

Stratix GX Device Handbook, Volume 1



101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com

SGX5V1-1.2

Copyright © 2006 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Altera Lead-Free

I.S. EN ISO 9001

Contents

Chapter Revision Dates vii

About This Handbook ix
How to Contact Altera ix
Typographic Conventions ix

Section I. Stratix GX Device Family Data Sheet
Revision History ..... Section I–2

Chapter 1. Introduction to the Stratix GX Device Data Sheet
Overview 1–1
Features 1–1
High-Speed I/O Interface Functional Description 1–4
FPGA Functional Description 1–5

Chapter 2. Stratix GX Transceivers
Transmitter Path 2–5
Receiver Path 2–13
Loopback Modes 2–26
BIST (Built-In Self Test) 2–28
Stratix GX Clocking 2–30
Other Transceiver Features 2–37
Individual Power-Down & Reset for the Transmitter & Receiver 2–37
Voltage Reference Capabilities 2–38
Hot-Socketing Capabilities 2–39
Applications & Protocols Supported with Stratix GX Devices 2–39
Stratix GX Example Application Support 2–39
High-Speed Serial Bus Protocols 2–40

Chapter 3. Source-Synchronous Signaling With DPA
Introduction 3–1
Stratix GX I/O Banks 3–1
Principles of SERDES Operation 3–1
DPA Block Overview 3–5
DPA Operation 3–10

Chapter 4. Stratix GX Architecture

Logic Array Blocks 4–1
LAB Interconnects 4–1
LAB Control Signals 4–2
Logic Elements 4–3
LUT Chain & Register Chain 4–5
addnsub Signal 4–5
LE Operating Modes 4–5
Clear & Preset Logic Control 4–10
MultiTrack Interconnect 4–11
TriMatrix Memory 4–18
Memory Modes 4–19
Parity Bit Support 4–21
Shift Register Support 4–21
Memory Block Size 4–22
Independent Clock Mode 4–40
Input/Output Clock Mode 4–42
Read/Write Clock Mode 4–44
Single-Port Mode 4–45
Digital Signal Processing Block 4–46
Multiplier Block 4–52
Adder/Output Blocks 4–56
Modes of Operation 4–59
DSP Block Interface 4–65
PLLs & Clock Networks 4–68
Global & Hierarchical Clocking 4–68
Enhanced & Fast PLLs 4–76
Enhanced PLLs 4–82
Fast PLLs 4–93
I/O Structure 4–96
Double-Data Rate I/O Pins 4–103
External RAM Interfacing 4–107
Programmable Drive Strength 4–110
Open-Drain Output 4–111
Slew-Rate Control 4–112
Bus Hold 4–112
Programmable Pull-Up Resistor 4–113
Advanced I/O Standard Support 4–113
Differential On-Chip Termination 4–118
MultiVolt I/O Interface 4–120
Power Sequencing & Hot Socketing 4–121
IEEE Std. 1149.1 (JTAG) Boundary-Scan Support 4–122

Chapter 5. Configuration & Testing

SignalTap Embedded Logic Analyzer 5–1
Configuration 5–1
Operating Modes 5–1

Configuration Schemes 5–2
Partial Reconfiguration 5–3
Remote Update Configuration Modes 5–3
Stratix GX Automated Single Event Upset (SEU) Detection 5–7
Custom-Built Circuitry 5–8
Software Interface 5–8
Temperature-Sensing Diode 5–8

Chapter 6. DC & Switching Characteristics

Operating Conditions 6–1
Power Consumption 6–22
Timing Model 6–22
Preliminary & Final Timing 6–23
Performance 6–23
Internal Timing Parameters 6–26
External Timing Parameters 6–35
External I/O Delay Parameters 6–44
Maximum Input & Output Clock Rates 6–54
High-Speed I/O Specification 6–58
PLL Timing 6–62
DLL Jitter 6–68

Chapter 7. Reference & Ordering Information

Software 7–1
Device Pin-Outs 7–1
Ordering Information 7–1

Chapter Revision Dates

The chapters in this book, *Stratix GX Device Handbook, Volume 1*, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1. Introduction to the Stratix GX Device Data Sheet
Revised: *February 2005*
Part number: *SGX51001-1.0*

Chapter 2. Stratix GX Transceivers
Revised: *June 2006*
Part number: *SGX51002-1.1*

Chapter 3. Source-Synchronous Signaling With DPA
Revised: *August 2005*
Part number: *SGX51003-1.1*

Chapter 4. Stratix GX Architecture
Revised: *February 2005*
Part number: *SGX51004-1.0*

Chapter 5. Configuration & Testing
Revised: *February 2005*
Part number: *SGX51005-1.0*

Chapter 6. DC & Switching Characteristics
Revised: *June 2006*
Part number: *SGX51006-1.2*

Chapter 7. Reference & Ordering Information
Revised: *February 2005*
Part number: *SGX51007-1.0*

About This Handbook

This handbook provides comprehensive information about the Altera® Stratix® GX family of devices.

How to Contact Altera

For the most up-to-date information about Altera products, go to the Altera world-wide web site at www.altera.com. For technical support on this product, go to www.altera.com/mysupport. For additional information about Altera products, consult the sources shown below.

Information Type	USA & Canada	All Other Locations
Technical support	www.altera.com/mysupport/	www.altera.com/mysupport/
	(800) 800-EPLD (3753) (7:00 a.m. to 5:00 p.m. Pacific Time)	+1 408-544-8767 7:00 a.m. to 5:00 p.m. (GMT -8:00) Pacific Time
Product literature	www.altera.com	www.altera.com
Altera literature services	literature@altera.com	literature@altera.com
Non-technical customer service	(800) 767-3753	+ 1 408-544-7000 7:00 a.m. to 5:00 p.m. (GMT -8:00) Pacific Time
FTP site	ftp.altera.com	ftp.altera.com

Typographic Conventions

This document uses the typographic conventions shown below.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: **Save As** dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: $\mathbf{f_{MAX}}$, **\qdesigns** directory, **d:** drive, **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Document titles are shown in italic type with initial capital letters. Example: *AN 75: High-Speed Board Design.*

Visual Cue	Meaning
Italic type	Internal timing parameters and variables are shown in italic type. Examples: t_{PIA}, $n + 1$. Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: *<file name>*, *<project name>*.**pof** file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
"Subheading Title"	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: "Typographic Conventions."
`Courier type`	Signal and port names are shown in lowercase Courier type. Examples: `data1`, `tdi`, `input.` Active-low signals are denoted by suffix `n`, e.g., `resetn`. Anything that must be typed exactly as it appears is shown in Courier type. For example: `c:\qdesigns\tutorial\chiptrip.gdf`. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword `SUBDESIGN`), as well as logic function names (e.g., `TRI`) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ● •	Bullets are used in a list of items when the sequence of the items is not important.
✓	The checkmark indicates a procedure that consists of one step only.
☞	The hand points to information that requires special attention.
CAUTION	The caution indicates required information that needs special consideration and understanding and should be read prior to starting or continuing with the procedure or process.
	The warning indicates information that should be read prior to starting or continuing the procedure or processes
↵	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information on a particular topic.

Section I. Stratix GX Device Family Data Sheet

This section provides the data sheet specifications for Stratix® GX devices. It contains feature definitions of the internal architecture, configuration information, testing information, DC operating conditions, and AC timing parameters.

This section includes the following chapters:

- Chapter 1, Introduction to the Stratix GX Device Data Sheet
- Chapter 2, Stratix GX Transceivers
- Chapter 3, Source-Synchronous Signaling With DPA
- Chapter 4, Stratix GX Architecture
- Chapter 5, Configuration & Testing
- Chapter 6, DC & Switching Characteristics
- Chapter 7, Reference & Ordering Information

Revision History

The table below shows the revision history for Chapters 1 through 7.

Chapter(s)	Date / Version	Changes Made	Comments
1	February 2005, v1.0	Initial Release.	
2	June 2006, v1.1	• Updated "Serial Loopback" section. • Updated Figures 2–1 through 2–3. • Updated Figure 2–13. • Updated Figures 2–26 and 2–27.	
	February 2005, v1.0	Initial Release.	
3	August 2005, v1.1	Added Note (3) to Figure 3-7.	
4	February 2005, v1.0	Initial Release.	
5	February 2005, v1.0	Initial Release.	
6	June 2006, v1.2	• Updated "Operating Conditions" section. • Updated Table 6–4. • Updated note 3 in Table 6–6. • Added note 12 in Table 6–7. • Updated Figure 6–1. • Added Figure 6–2. • Updated Tables 6–13 through 6–16.	• Changed V_{OD} to V_{ID} for receiver input voltage and `refclkb` input voltage in Table 6–4. • Changed value for undershoot during transition from -0.5 V to -2.0 V in note 3 of Table 6–6. • Changed value of V_{OCM} from mV to V in Table 6–15. • Changed unit value of W to Ω.
	August 2005, v1.1	Updated Tables 6-7 and 6-50.	
7	February 2005, v1.0	Initial Release.	

1. Introduction to the Stratix GX Device Data Sheet

SGX51001-1.0

Overview

The Stratix® GX family of devices is Altera's second FPGA family to combine high-speed serial transceivers with a scalable, high-performance logic array. Stratix GX devices include 4 to 20 high-speed transceiver channels, each incorporating clock data recovery (CDR) technology and embedded SERDES capability at data rates of up to 3.1875 gigabits per second (Gbps). These transceivers are grouped by four-channel transceiver blocks, and are designed for low power consumption and small die size. The Stratix GX FPGA technology is built upon the Stratix architecture, and offers a 1.5-V logic array with unmatched performance, flexibility, and time-to-market capabilities. This scalable, high-performance architecture makes Stratix GX devices ideal for high-speed backplane interface, chip-to-chip, and communications protocol-bridging applications.

Features

- Transceiver block features are as follows:
  - High-speed serial transceiver channels with CDR provides 500-megabits per second (Mbps) to 3.1875-Gbps full-duplex operation
  - Devices are available with 4, 8, 16, or 20 high-speed serial transceiver channels providing up to 127.5 Gbps of full-duplex serial bandwidth
  - Support for transceiver-based protocols, including 10 Gigabit Ethernet attachment unit interface (XAUI), Gigabit Ethernet (GigE), and SONET/SDH
  - Compatible with PCI Express, SMPTE 292M, Fibre Channel, and Serial RapidIO I/O standards
  - Programmable differential output voltage (V_{OD}), pre-emphasis, and equalization settings for improved signal integrity
  - Individual transmitter and receiver channel power-down capability implemented automatically by the Quartus® II software for reduced power consumption during non-operation
  - Programmable transceiver-to-FPGA interface with support for 8-, 10-, 16-, and 20-bit wide data paths
  - 1.5-V pseudo current mode logic (PCML) for 500 Mbps to 3.1875 Gbps
  - Support for LVDS, LVPECL, and 3.3-V PCML on reference clocks and receiver input pins (AC-coupled)
  - Built-in self test (BIST)
  - Hot insertion/removal protection circuitry

- Pattern detector and word aligner supports programmable patterns
- 8B/10B encoder/decoder performs 8- to 10-bit encoding and 10- to 8-bit decoding
- Rate matcher compliant with IEEE 802.3-2002 for GigE mode and with IEEE 802-3ae for XAUI mode
- Channel bonding compliant with IEEE 802.3ae (for XAUI mode only)
- Device can bypass some transceiver block features if necessary

- FPGA features are as follows:
  - 10,570 to 41,250 logic elements (LEs); see Table 1–1
  - Up to 3,423,744 RAM bits (427,968 bytes) available without reducing logic resources
  - TriMatrix™ memory consisting of three RAM block sizes to implement true dual-port memory and first-in-out (FIFO) buffers
  - Up to 16 global clock networks with up to 22 regional clock networks per device region
  - High-speed DSP blocks provide dedicated implementation of multipliers (faster than 300 MHz), multiply-accumulate functions, and finite impulse response (FIR) filters
  - Up to eight general usage phase-locked loops (four enhanced PLLs and four fast PLLs) per device provide spread spectrum, programmable bandwidth, clock switchover, real-time PLL reconfiguration, and advanced multiplication and phase shifting
  - Support for numerous single-ended and differential I/O standards
  - High-speed source-synchronous differential I/O support on up to 45 channels for 1-Gbps performance
  - Support for source-synchronous bus standards, including 10-Gigabit Ethernet XSBI, Parallel RapidIO, UTOPIA IV, Network Packet Streaming Interface (NPSI), HyperTransport™ technology, SPI-4 Phase 2 (POS-PHY Level 4), and SFI-4
  - Support for high-speed external memory, including zero bus turnaround (ZBT) SRAM, quad data rate (QDR and QDRII) SRAM, double data rate (DDR) SDRAM, DDR fast cycle RAM (FCRAM), and single data rate (SDR) SDRAM
  - Support for multiple intellectual property megafunctions from Altera® MegaCore® functions and Altera Megafunction Partners Program (AMPP℠) megafunctions
  - Support for remote configuration updates
  - Dynamic phase alignment on LVDS receiver channels

Table 1–1. Stratix GX Device Features

Feature	EP1SGX10C EP1SGX10D	EP1SGX25C EP1SGX25D EP1SGX25F	EP1SGX40D EP1SGX40G
LEs	10,570	25,660	41,250
Transceiver channels	4, 8	4, 8, 16	8, 20
Source-synchronous channels	22	39	45
M512 RAM blocks (32 × 18 bits)	94	224	384
M4K RAM blocks (128 × 36 bits)	60	138	183
M-RAM blocks (4K ×144 bits)	1	2	4
Total RAM bits	920,448	1,944,576	3,423,744
Digital signal processing (DSP) blocks	6	10	14
Embedded multipliers *(1)*	48	80	112
PLLs	4	4	8

Note to Table 1–1:

(1) This parameter lists the total number of 9- × 9-bit multipliers for each device. For the total number of 18- × 18-bit multipliers per device, divide the total number of 9- × 9-bit multipliers by 2. For the total number of 36- × 36-bit multipliers per device, decide the total number of 9- × 9-bit multipliers by 8.

Stratix GX devices are available in space-saving FineLine BGA® packages (refer to Tables 1–2 and 1–3), and in multiple speed grades (refer to Table 1–4). Stratix GX devices support vertical migration within the same package (that is, you can migrate between the EP1SGX10C and EP1SGX25C devices in the 672-pin FineLine BGA package). See the Stratix GX device pin tables for more information. Vertical migration means that you can migrate to devices whose dedicated pins, configuration pins, and power pins are the same for a given package across device densities. For I/O pin migration across densities, you must cross-reference the available I/O pins using the device pin-outs for all planned densities of a given package type, to identify which I/O pins it is possible to migrate. The Quartus II software can automatically cross reference and place all pins for migration when given a device migration list.

Table 1–2. Stratix GX Package Options & I/O Pin Counts (Part 1 of 2) *Note (1)*

Device	672-Pin FineLine BGA	1,020-Pin FineLine BGA
EP1SGX10C	362	
EP1SGX10D	362	
EP1SGX25C	455	

Table 1–2. Stratix GX Package Options & I/O Pin Counts (Part 2 of 2) *Note (1)*

Device	672-Pin FineLine BGA	1,020-Pin FineLine BGA
EP1SGX25D	455	607
EP1SGX25F		607
EP1SGX40D		624
EP1SGX40G		624

Note to Table 1–2:

(1) The number of I/O pins listed for each package includes dedicated clock pins and dedicated fast I/O pins. However, these numbers do not include high-speed or clock reference pins for high-speed I/O standards.

Table 1–3. Stratix GX FineLine BGA Package Sizes

Dimension	672 Pin	1,020 Pin
Pitch (mm)	1.00	1.00
Area (mm^2)	729	1,089
Length × width (mm × mm)	27 × 27	33 × 33

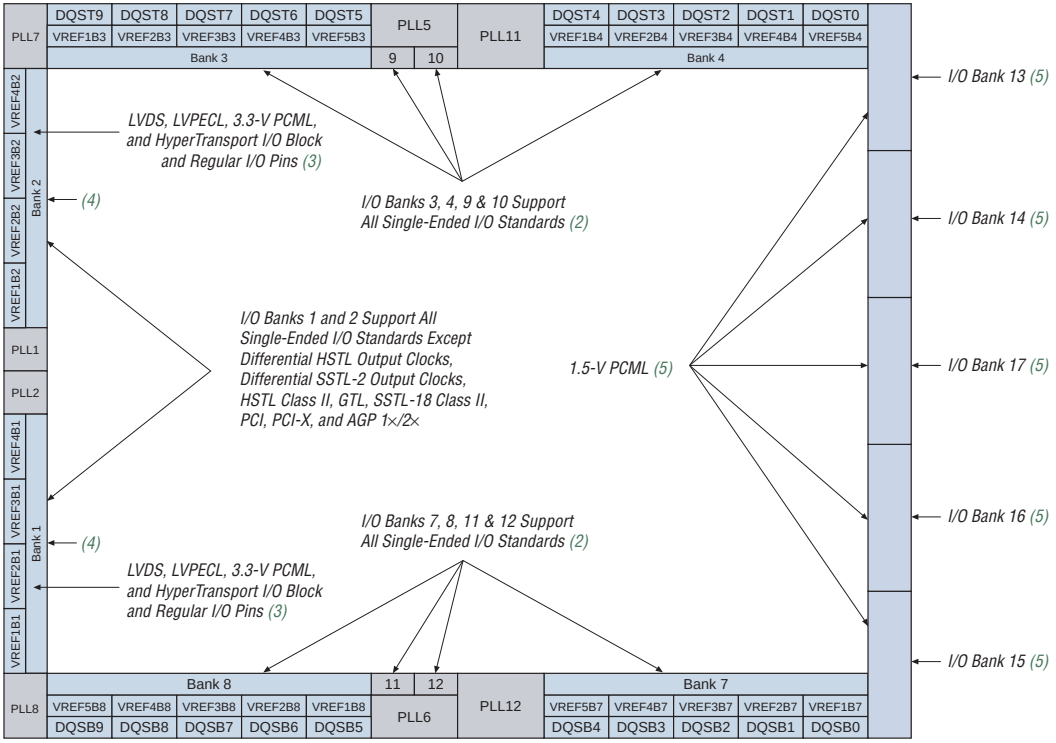
Table 1–4. Stratix GX Device Speed Grades

Device	672-Pin FineLine BGA	1,020-pin FineLine BGA
EP1SGX10	-5, -6, -7	
EP1SGX25	-5, -6, -7	-5, -6, -7
EP1SGX40		-5, -6, -7

High-Speed I/O Interface Functional Description

The Stratix GX device family supports high-speed serial transceiver blocks with CDR circuitry as well as source-synchronous interfaces. The channels on the right side of the device use an embedded circuit dedicated for receiving and transmitting high-speed serial data streams to and from the system board. These channels are clustered in a four-channel serial transceiver building block and deliver high-speed bidirectional point-to-point data transmissions to provide up to 3.1875 Gbps of full-duplex data transmission per channel. The channels on the left side of the device support source-synchronous data transfers at up to 1 Gbps using LVDS, LVPECL, 3.3-V PCML, or HyperTransport technology I/O standards. Figure 1–1 shows the Stratix GX I/O blocks. The differential source-synchronous serial interface and the high-speed serial interface are described in the *Stratix GX Transceivers* chapter of the *Stratix GX Device Handbook, Volume 1*.

Figure 1–1. Stratix GX I/O Blocks *Note (1)*



Notes to Figure 1–1:

(1) Figure 1–1 is a top view of the Stratix GX silicon die.
(2) Banks 9 through 12 are enhanced PLL external clock output banks.
(3) If the high-speed differential I/O pins are not used for high-speed differential signaling, they can support all of the I/O standards except HSTL class I and II, GTL, SSTL-18 Class II, PCI, PCI-X, and AGP 1×/2×.
(4) For guidelines for placing single-ended I/O pads next to differential I/O pads, see the *Selectable I/O Standards in Stratix & Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2*.
(5) These I/O banks in Stratix GX devices also support the LVDS, LVPECL, and 3.3-V PCML I/O standards on reference clocks and receiver input pins (AC coupled).

FPGA Functional Description

Stratix GX devices contain a two-dimensional row- and column-based architecture to implement custom logic. A series of column and row interconnects of varying length and speed provide signal interconnects between logic array blocks (LABs), memory block structures, and DSP blocks.

The logic array consists of LABs, with 10 logic elements (LEs) in each LAB. An LE is a small unit of logic providing efficient implementation of user logic functions. LABs are grouped into rows and columns across the device.

M512 RAM blocks are simple dual-port memory blocks with 512 bits plus parity (576 bits). These blocks provide dedicated simple dual-port or single-port memory up to 18-bits wide at up to 318 MHz. M512 blocks are grouped into columns across the device in between certain LABs.

M4K RAM blocks are true dual-port memory blocks with 4K bits plus parity (4,608 bits). These blocks provide dedicated true dual-port, simple dual-port, or single-port memory up to 36-bits wide at up to 291 MHz. These blocks are grouped into columns across the device in between certain LABs.

M-RAM blocks are true dual-port memory blocks with 512K bits plus parity (589,824 bits). These blocks provide dedicated true dual-port, simple dual-port, or single-port memory up to 144-bits wide at up to 269 MHz. Several M-RAM blocks are located individually or in pairs within the device's logic array.

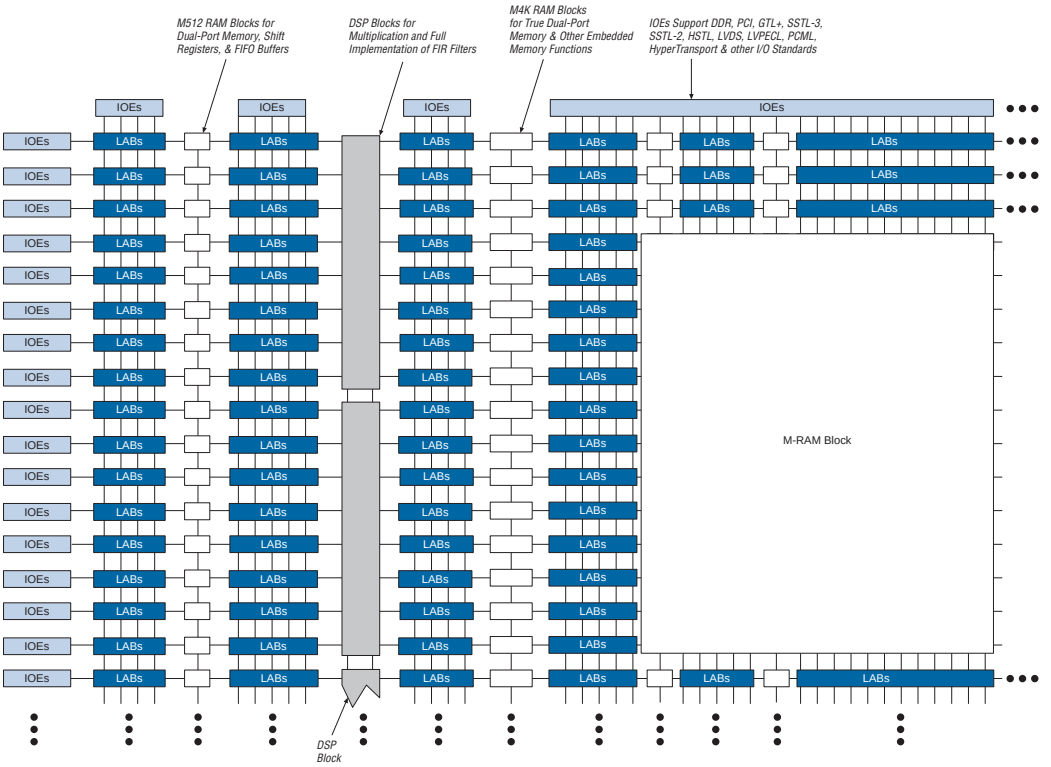
Digital signal processing (DSP) blocks can implement up to either eight full-precision 9 × 9-bit multipliers, four full-precision 18 × 18-bit multipliers, or one full-precision 36 × 36-bit multiplier with add or subtract features. These blocks also contain 18-bit input shift registers for digital signal processing applications, including FIR and infinite impulse response (IIR) filters. DSP blocks are grouped into two columns in each device.

Each Stratix GX device I/O pin is fed by an I/O element (IOE) located at the end of LAB rows and columns around the periphery of the device. I/O pins support numerous single-ended and differential I/O standards. Each IOE contains a bidirectional I/O buffer and six registers for registering input, output, and output-enable signals. When used with dedicated clocks, these registers provide exceptional performance and interface support with external memory devices such as DDR SDRAM, FCRAM, ZBT, and QDR SRAM devices.

High-speed serial interface channels support transfers at up to 840 Mbps using LVDS, LVPECL, 3.3-V PCML, or HyperTransport technology I/O standards.

Figure 1–2 shows an overview of the Stratix GX device.

Figure 1–2. Stratix GX Block Diagram



The number of M512 RAM, M4K RAM, and DSP blocks varies by device along with row and column numbers and M-RAM blocks. Table 1–5 lists the resources available in Stratix GX devices.

Table 1–5. Stratix GX Device Resources

Device	M512 RAM Columns/Blocks	M4K RAM Columns/Blocks	M-RAM Blocks	DSP Block Columns/Blocks	LAB Columns	LAB Rows
EP1SGX10	4 / 94	2 / 60	1	2 / 6	40	30
EP1SGX25	6 / 224	3 / 138	2	2 / 10	62	46
EP1SGX40	8 / 384	3 / 183	4	2 / 14	77	61

2. Stratix GX Transceivers

SGX51002-1.1

Transceiver Blocks

Stratix® GX devices incorporate dedicated embedded circuitry on the right side of the device, which contains up to 20 high-speed 3.1875-Gbps serial transceiver channels. Each Stratix GX transceiver block contains four full-duplex channels and supporting logic to transmit and receive high-speed serial data streams. The transceiver block uses the channels to deliver bidirectional point-to-point data transmissions with up to 3.1875 Gbps of data transition per channel.

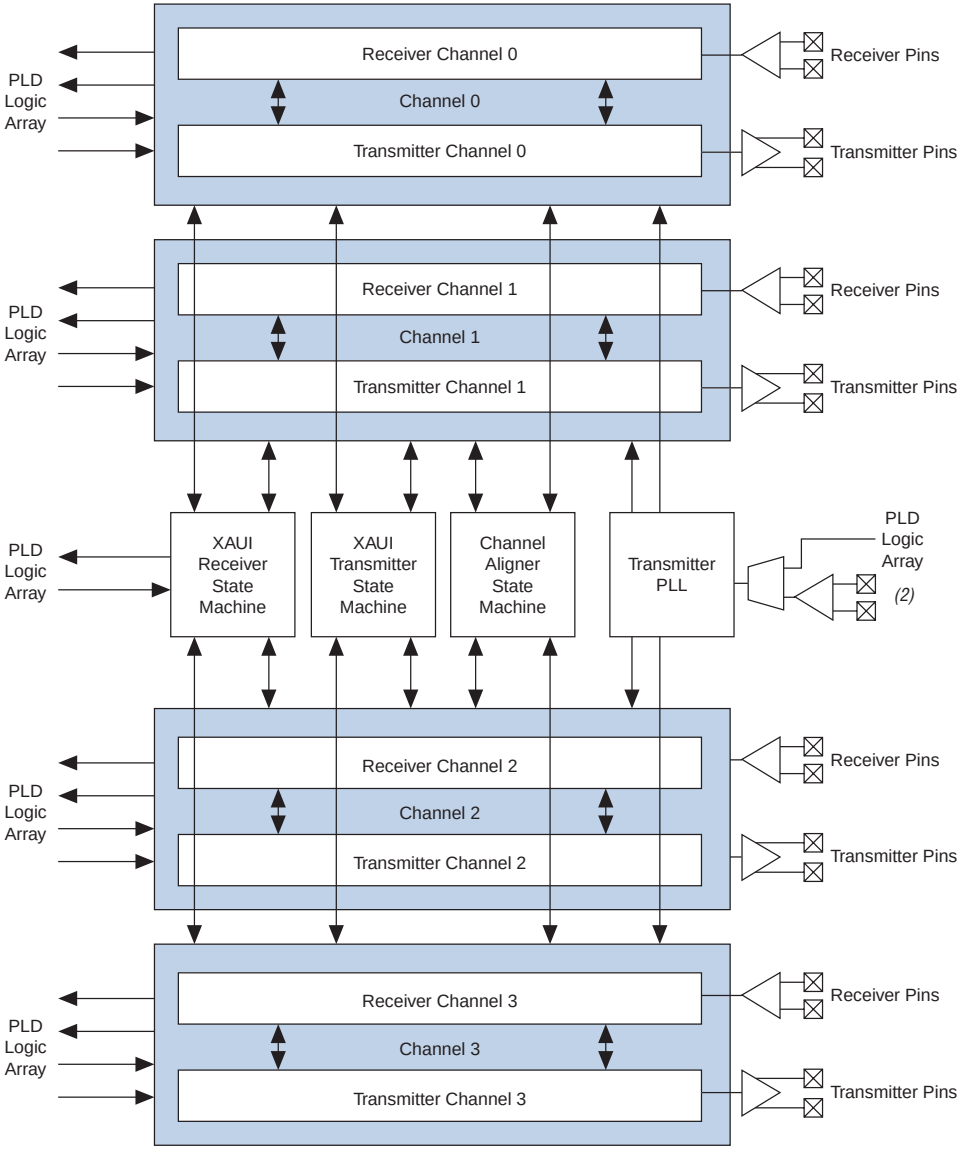
There are up to 20 transceiver channels available on a single Stratix GX device. Table 2–1 shows the number of transceiver channels available on each Stratix GX device.

Table 2–1. Stratix GX Transceiver Channels

Device	Number of Transceiver Channels
EP1SGX10C	4
EP1SGX10D	8
EP1SGX25C	4
EP1SGX25D	8
EP1SGX25F	16
EP1SGX40D	8
EP1SGX40G	20

Figure 2–1 shows the elements of the transceiver block, including the four channels, supporting logic, and I/O buffers. Each transceiver channel consists of a receiver and transmitter. The supporting logic contains a transmitter PLL to generate a high-speed clock used by the four transmitters. The receiver PLL within each transceiver channel generates the receiver reference clocks. The supporting logic also contains state machines to manage rate matching for XAUI and GIGE applications, in addition to channel bonding for XAUI applications.

Figure 2–1. Stratix GX Transceiver Block *Note (1)*



Notes to Figure 2–1:

(1) Each receiver channel has its own PLL and CRU, which are not shown in this diagram. For more information, refer to the section "Receiver Path" on page 2–13.

(2) For possible transmitter PLL clock inputs, refer to the section "Transmitter Path" on page 2–5.

Each Stratix GX transceiver channel consists of a transmitter and receiver. The transmitter contains the following:

- Transmitter PLL
- Transmitter phase compensation FIFO buffer
- Byte serializer
- 8B/10B encoder
- Serializer (parallel to serial converter)
- Transmitter output buffer

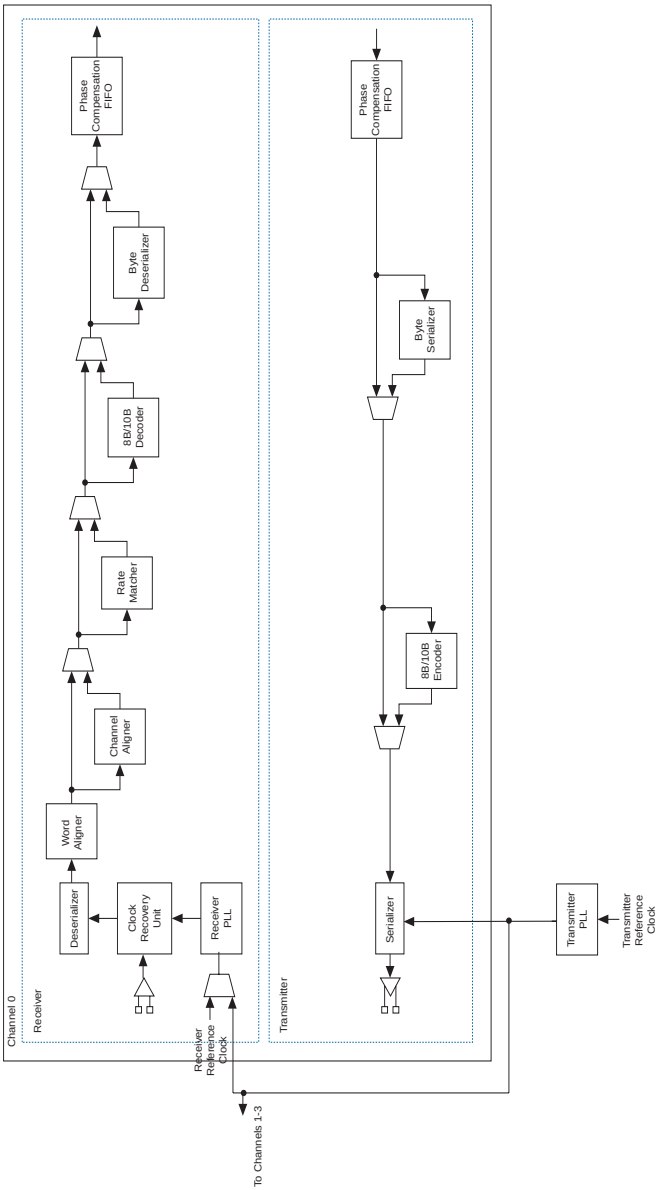
The receiver contains the following:

- Input buffer
- Clock recovery unit (CRU)
- Deserializer
- Pattern detector and word aligner
- Rate matcher and channel aligner
- 8B/10B decoder
- Receiver logic array interface

You can set all the Stratix GX transceiver functions through the Quartus II software. You can set programmable pre-emphasis, programmable equalizer, and programmable V_{OD} dynamically as well. Each Stratix GX transceiver channel is also capable of BIST generation and verification in addition to various loopback modes. Figure 2–2 shows the block diagram for the Stratix GX transceiver channel.

Stratix GX transceivers provide physical coding sublayer (PCS) and physical media attachment (PMA) implementation for protocols such as 10-gigabit XAUI and GIGE. The PCS portion of the transceiver consists of the logic array interface, 8B/10B encoder/decoder, pattern detector, word aligner, rate matcher, channel aligner, and the BIST and pseudo-random binary sequence pattern generator/verifier. The PMA portion of the transceiver consists of the serializer/deserializer, the CRU, and the I/O buffers.

Figure 2–2. Stratix GX Transceiver ChanneL Note (1)



Note to Figure 2–2:
(1) There are four transceiver channels in a transceiver block.

Transmitter Path

This section describes the data path through the Stratix GX transmitter (see Figure 2–2). Data travels through the Stratix GX transmitter via the following modules:

- Transmitter PLL
- Transmitter phase compensation FIFO buffer
- Byte serializer
- 8B/10B encoder
- Serializer (parallel to serial converter)
- Transmitter output buffer

Transmitter PLL

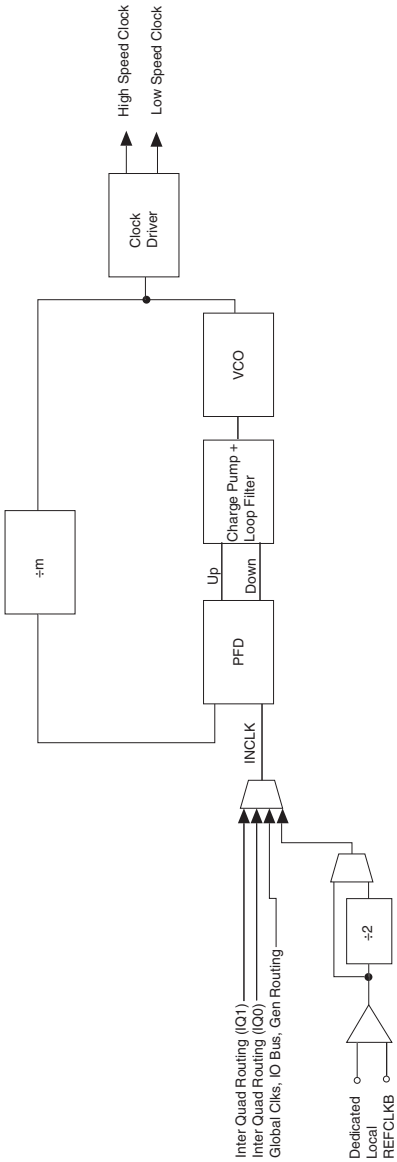
Each transceiver block has one transmitter PLL, which receives the reference clock and generates the following signals:

- High-speed serial clock used by the serializer
- Slow-speed reference clock used by the receiver
- Slow-speed clock used by the logic array (divisible by two for double-width mode)

The INCLK clock is the input into the transmitter PLL. There is one INCLK clock per transceiver block. This clock can be fed by either the REFCLKB pin, PLD routing, or the inter-transceiver routing line. See the section "Stratix GX Clocking" on page 2–30 for more information about the inter-transceiver lines.

The transmitter PLL in each transceiver block clocks the circuits in the transmit path. The transmitter PLL is also used to train the receiver PLL. If no transmit channels are used in the transceiver block, the transmitter PLL can be turned off. Figure 2–3 is a block diagram of the transmitter PLL.

Figure 2–3. Transmitter PLL Block Diagram *Note (1)*



Note to Figure 2–3:
(1) The divider in the PLL divides by 4, 8, 10, 16, or 20.

The transmitter PLL can support up to 3.1875 Mbps. The input clock frequency for –5 and –6 speed grade devices is limited to 650 MHz if you use the REFCLKB pin or to 325 MHz if you use the other clock routing resources. For –7 speed grade devices, the maximum input clock frequency is 312.5 MHz with the REFCLKB pin, and the maximum is 156.25 MHz for all other clock routing resources. An optional PLL_LOCKED port is available to indicate whether the transmitter PLL is locked to the reference clock. The transmitter PLL has a programmable loop bandwidth that can be set to low or high. The loop bandwidth parameter can be statically set in the Quartus II software.

Table 2–2 lists the adjustable parameters in the transmitter PLL.

Table 2–2. Transmitter PLL Specifications

Parameter	Specifications
Input reference frequency range	25 MHz to 650 MHz
Data rate support	500 Mbps to 3.1875 Gbps
Multiplication factor (W)	2, 4, 5, 8, 10, 16, or 20 *(1)*
Bandwidth	Low, high

Note to Table 2–2:
(1) Multiplication factors 2 and 5 can only be achieved with the use of the pre-divider on the REFCLKB pin.

Transmitter Phase Compensation FIFO Buffer

The transmitter phase compensation FIFO buffer resides in the transceiver block at the PLD boundary. This FIFO buffer compensates for the phase differences between the transmitter reference clock (inclk) and the PLD interface clock (tx_coreclk). The phase difference between the two clocks must be less than 360° . The PLD interface clock must also be frequency locked to the transmitter reference clock. The phase compensation FIFO buffer is four words deep and cannot be bypassed.

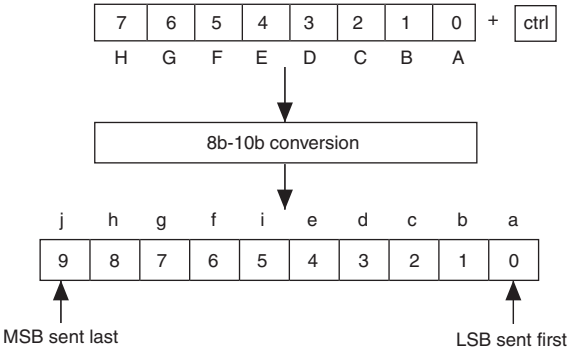
Byte Serializer

The byte serializer takes double-width words (16 or 20 bits) from the PLD interface and converts them to a single width word (8 or 10 bits) for use in the transceiver. The transmit data path after the byte serializer is single width (8 or 10 bits). The byte serializer is bypassed when single width mode (8 or 10 bits) is used at the PLD interface.

8B/10B Encoder

The 8B/10B encoder translates 8-bit wide data + 1 control enable bit into a 10-bit encoded data. The encoded data has a maximum run length of 5. The 8B/10B encoder can be bypassed. Figure 2–4 diagrams the encoding process.

Figure 2–4. Encoding Process



Transmit State Machine

The transmit state machine operates in either XAUI mode or in GIGE mode, depending on the protocol used.

GIGE Mode

In GIGE mode, the transmit state machines convert all idle ordered sets (/K28.5/, /Dx.y/) to either /I1/ or /I2/ ordered sets. /I1/ consists of a negative-ending disparity /K28.5/ (denoted by /K28.5/-) followed by a neutral /D5.6/. /I2/ consists of a positive-ending disparity /K28.5/ (denoted by /K28.5/+) and a negative-ending disparity /D16.2/ (denoted by /D16.2/-). The transmit state machines do not convert any of the ordered sets to match /C1/ or /C2/, which are the configuration ordered sets. (/C1/ and /C2/ are defined by (/K28.5/, /D21.5/) and (/K28.5/, /D2.2/), respectively.) Both the /I1/ and /I2/ ordered sets guarantee a negative-ending disparity after each ordered set. The GIGE transmit state machine can be statically disabled in the Quartus II software, even if using the GIGE protocol mode.

XAUI Mode

The transmit state machine translates the XAUI XGMII code group to the XAUI PCS code group. Table 2–3 shows the code conversion.

Table 2–3. Code Conversion

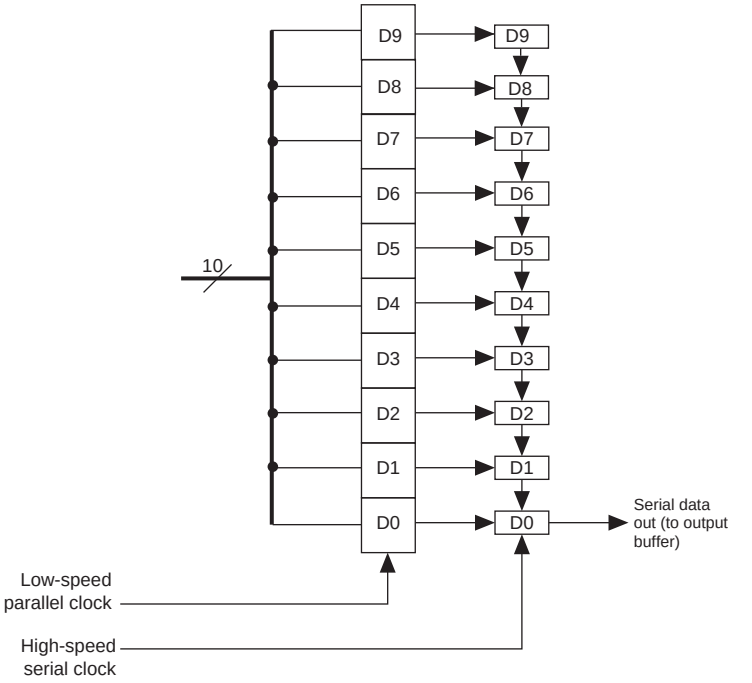
XGMII TXC	XGMII TXD	PCS Code-Group	Description
0	00 through FF	Dxx.y	Normal data
1	07	K28.0 or K28.3 or K28.5	Idle in ||**I**||
1	07	K28.5	Idle in ||**T**||
1	9C	K28.4	Sequence
1	FB	K27.7	Start
1	FD	K29.7	Terminate
1	FE	K30.7	Error
1	See IEEE 802.3 reserved code groups	See IEEE 802.3 reserved code groups	Reserved code groups
1	Other value	K30.7	Invalid XGMII character

The XAUI PCS idle code groups, `/K28.0/` (`/R/`) and `/K28.5/` (`/K/`), are automatically randomized based on a PRBS7 pattern with an x^7+x^6+1 polynomial. The `/K28.3/` (`/A/`) code group is automatically generated between 16 and 31 idle code groups. The idle randomization on the `/A/`, `/K/`, and `/R/` code groups are done automatically by the transmit state machine.

Serializer (Parallel-to-Serial Converter)

The serializer converts the parallel 8-bit or 10-bit data into a serial stream, transmitting the LSB first. The serialized stream is then fed to the transmit buffer. Figure 2–5 is a diagram of the serializer.

Figure 2–5. Serializer

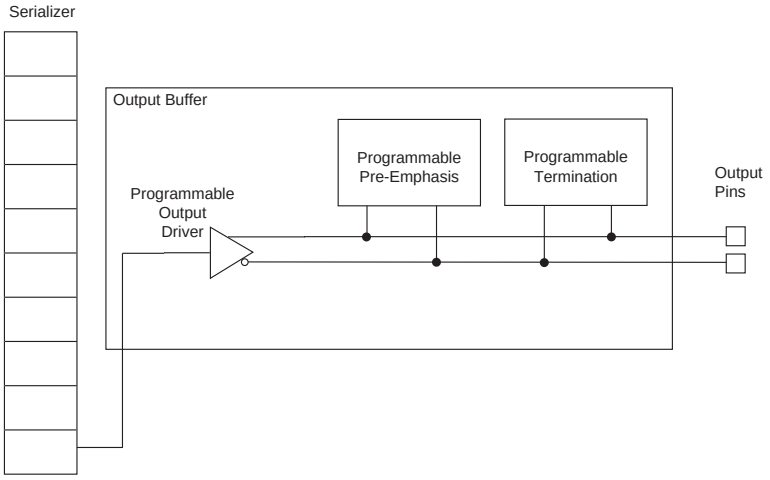


Transmit Buffer

The Stratix GX transceiver buffers support the 1.5-V pseudo current mode logic (PCML) I/O standard at a rate up to 3.1875 Gbps, across up to 40 inches of FR4 trace, and across 2 connectors. Additional I/O standards, LVDS, 3.3-V PCML, LVPECL, can be supported when AC coupled. The common mode of the output driver is 750 mV.

The output buffer, as shown in Figure 2–6, consists of a programmable output driver and a programmable pre-emphasis circuit.

Figure 2–6. Output Buffer



Programmable Output Driver

The programmable output driver can be set to drive out 400 to 1,600 mV. Table 2–4 shows the available settings for each termination value. The V_{OD} can be dynamically or statically set. The output driver requires either internal or external termination at the source.

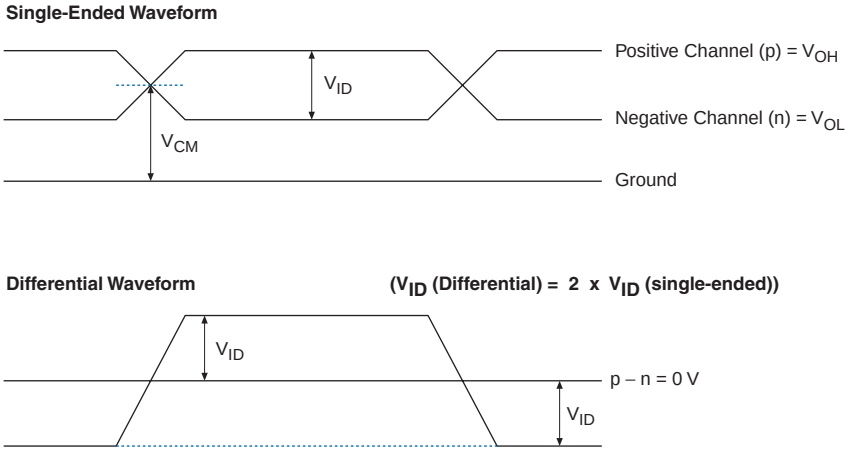
Table 2–4. Programmable V_{OD} (Differential) *Note (1)*

Termination Setting (Ω)	V_{OD} Setting (mV)
100	400, 800, 1000, 1200, 1400, 1600
120	480, 960, 1200, 1440
150	600, 1200, 1500

Note to Table 2–4:

(1) V_{OD} differential is measured as $V_A - V_B$ (see Figure 2–7).

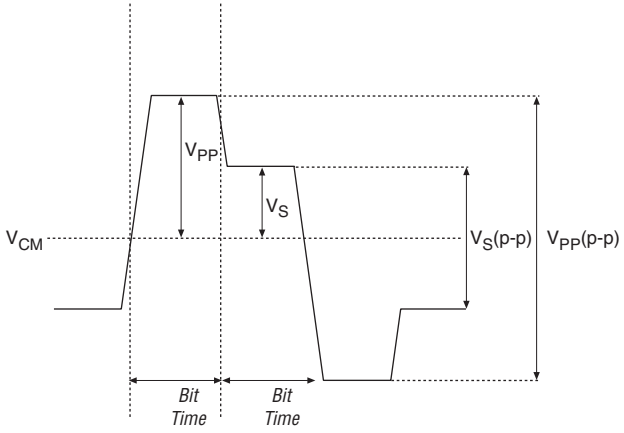
Figure 2–7. V_{OD} Differential



Programmable Pre-Emphasis

The programmable pre-emphasis module controls the output driver to boost the high frequency components, to compensate for losses in the transmission medium, as shown in Figure 2–8. The pre-emphasis can be dynamically or statically set. There are five possible pre-emphasis settings (1 through 5), with 5 being the highest and 0 being no pre-emphasis.

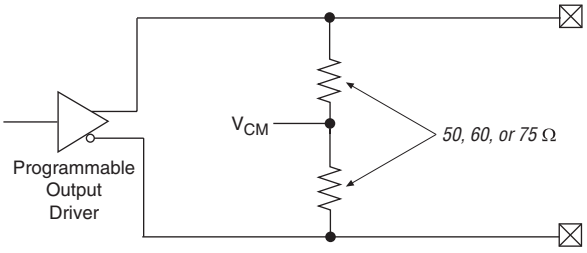
Figure 2–8. Programmable Pre-Emphasis Model



Pre-emphasis percentage is defined as $V_{PP}/V_S - 1$, where V_{PP} is the differential emphasized voltage (peak-to-peak) and V_S is the differential steady-state voltage (peak-to-peak).

Programmable Transmitter Termination
The programmable termination can be statically set in the Quartus II software. The values are 100 Ω, 120 Ω, 150 Ω, and off. Figure 2–9 shows the setup for programmable termination.

Figure 2–9. Programmable Transmitter Termination



Receiver Path

This section describes the data path through the Stratix GX receiver (refer to Figure 2–2 on page 2–4). Data travels through the Stratix GX receiver via the following modules:

- Input buffer
- Clock Recovery Unit (CRU)
- Deserializer
- Pattern detector and word aligner
- Rate matcher and channel aligner
- 8B/10B decoder
- Receiver logic array interface

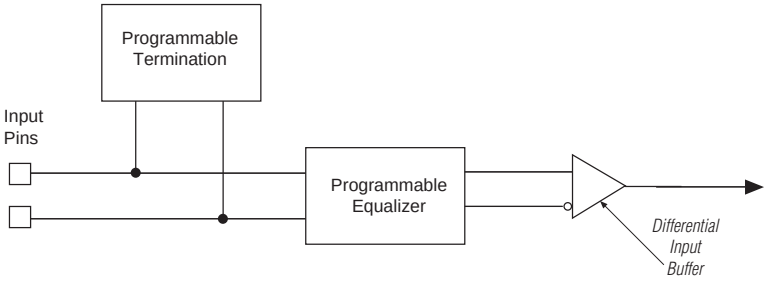
Receiver Input Buffer

The Stratix GX receiver input buffer supports the 1.5-V PCML I/O standard at a rate up to 3.1875 Gbps. Additional I/O standards, LVDS, 3.3-V PCML, and LVPECL can be supported when AC coupled. The common mode of the input buffer is 1.1 V. The receiver can support Stratix GX-to-Stratix GX DC coupling.

Figure 2–10 shows a diagram of the receiver input buffer, which contains:

- Programmable termination
- Programmable equalizer

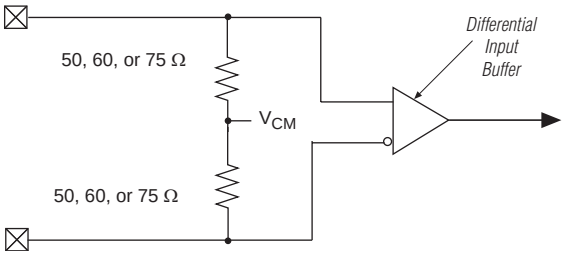
Figure 2–10. Receiver Input Buffer



Programmable Termination

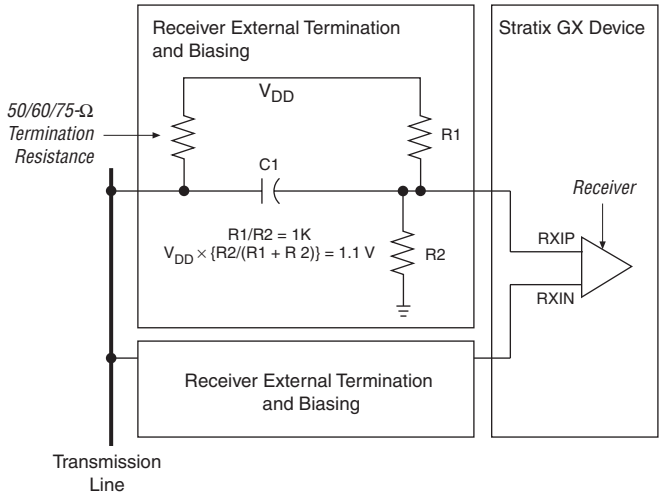
The programmable termination can be statically set in the Quartus II software. Figure 2–11 shows the setup for programmable receiver termination.

Figure 2–11. Programmable Receiver Termination



If you use external termination, then the receiver must be externally terminated and biased to 1.1 V. Figure 2–12 shows an example of an external termination/biasing circuit.

Figure 2–12. External Termination & Biasing Circuit



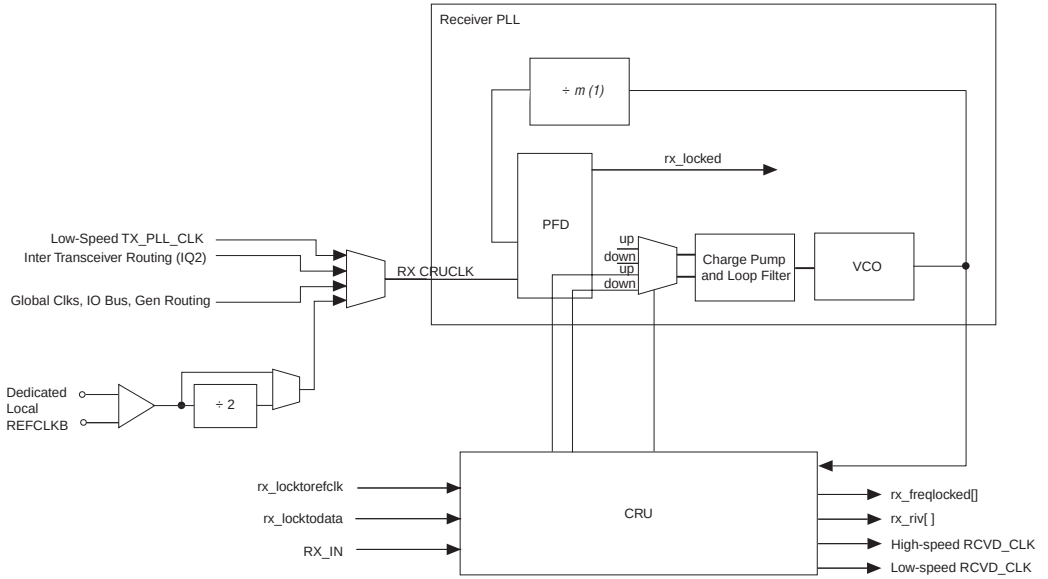
Programmable Equalizer

The programmable equalizer module boosts the high frequency components of the incoming signal to compensate for losses in the transmission medium. There are five possible equalization settings (0, 1, 2, 3, 4) to compensate for 0", 10", 20", 30", and 40" of FR4 trace. These settings should be interpreted loosely. The programmable equalizer can be set dynamically or statically.

Receiver PLL & CRU

Each transceiver block has four receiver PLLs and CRUs, each of which is dedicated to a receive channel. If the receive channel associated with a particular receiver PLL or CRU is not used, then the receiver PLL or CRU is powered down for the channel. Figure 2–13 is a diagram of the receiver PLL and CRU circuits.

Figure 2–13. Receiver PLL & CRU Circuit



Note to Figure 2–13:
(1) m = 8, 10 16, or 20.

The receiver PLLs and CRUs are capable of supporting up to 3.1875 Gbps. The input clock frequency for –5 and –6 speed grade devices is limited to 650 MHz if you use the REFCLKB pin or 325 MHz if you use the other clock routing resources. The maximum input clock frequency for –7 speed grade devices is 312.5 MHz if you use the REFCLKB pin or 156.25 MHz with the other clock routing resources. An optional RX_LOCKED port (active low signal) is available to indicate whether the PLL is locked to the reference clock. The receiver PLL has a programmable loop bandwidth, which can be set to low, medium, or high. The loop bandwidth parameter can be statically set by the Quartus II software.

Table 2–5 lists the adjustable parameters of the receiver PLL and CRU. All the parameters listed are statically programmable in the Quartus II software.

Table 2–5. Receiver PLL & CRU Adjustable Parameters (Part 1 of 2)

Parameter	Specifications
Input reference frequency range	25 MHz to 650 MHz
Data rate support	500 Mbps to 3.1875 Gbps

Table 2–5. Receiver PLL & CRU Adjustable Parameters (Part 2 of 2)	
Multiplication factor (W)	2, 4, 5, 8, 10, 16, or 20 *(1)*
PPM detector	125, 250, 500, 1,000
Bandwidth	Low, medium, high
Run length detector	10-bit or 20-bit mode: 5 to 160 in steps of 5
	8-bit or 16-bit mode: 4 to 128 in steps of 4

Note to Table 2–5:

(1) Multiplication factors 2, 4, and 5 can only be achieved with the use of the pre-divider on the REFCLKB port or if the CRU is trained with the low speed clock from the transmitter PLL.

The CRU has a built-in switchover circuit to select whether the voltage-controlled oscillator of the PLL is trained by the reference clock or the data. The optional port `rx_freqlocked` monitors when the CRU is in locked to data mode.

In the automatic mode, the following conditions must be met for the CRU to switch from locked to reference to locked to data mode:

- The CRU PLL is within the prescribed PPM frequency threshold setting (125 PPM, 250 PPM, 500 PPM, 1,000 PPM) of the CRU reference clock.
- The reference clock and CRU PLL output are phase matched (phases are within .08 UI).

The automatic switchover circuit can be overridden by using the optional ports `rx_lockedtorefclk` and `rx_locktodata`. Table 2–6 shows the possible combinations of these two signals.

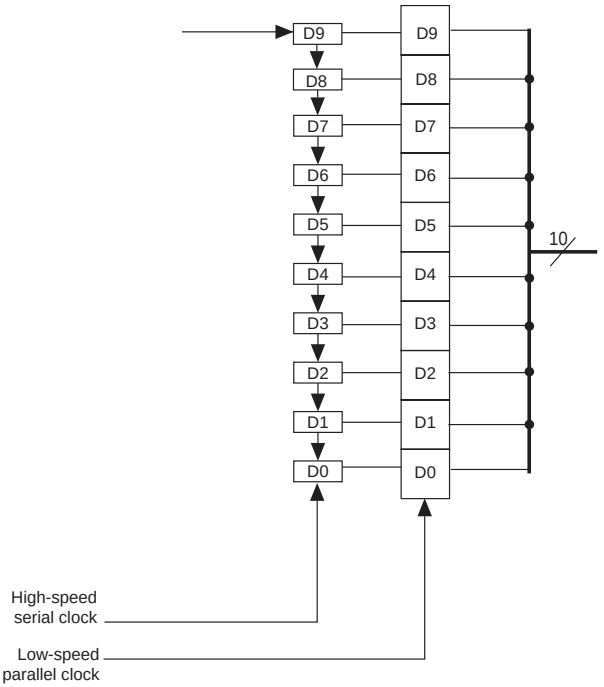
Table 2–6. Possible Combinations of rx_lockedtorefclk & rx_locktodata		
rx_locktodata	**rx_lockedtorefclk**	**VCO (lock to mode)**
0	0	Auto
0	1	Reference CLK
1	x	DATA

If the `rx_lockedtorefclk` and `rx_locktodata` ports are not used, the default is auto mode.

Deserializer (Serial-to-Parallel Converter)

The deserializer converts the serial stream into a parallel 8- or 10-bit data bus. The deserializer receives the least significant bit first. Figure 2–14 is a diagram of the deserializer.

Figure 2–14. Deserializer

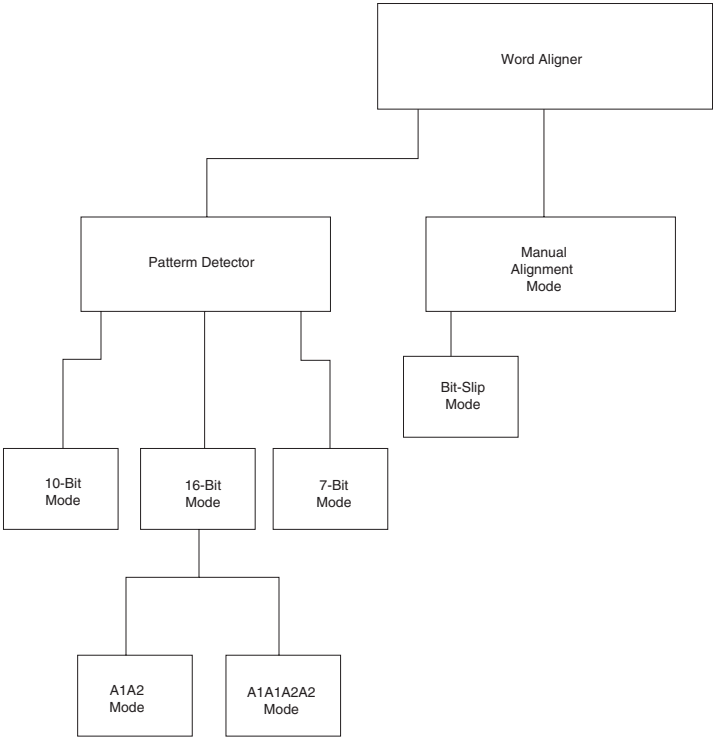


Word Aligner

The word aligner aligns the incoming data based on the specific byte boundaries. The word aligner has three customizable modes of operation: bit-slip mode, 16-bit mode, and 10-bit mode, the last of which is available for the basic and SONET modes. The word aligner also has two non-customizable modes of operation, which are the XAUI and GIGE modes.

Figure 2–15 shows the word aligner in bit-slip mode.

Figure 2–15. Word Aligner in Bit-Slip Mode



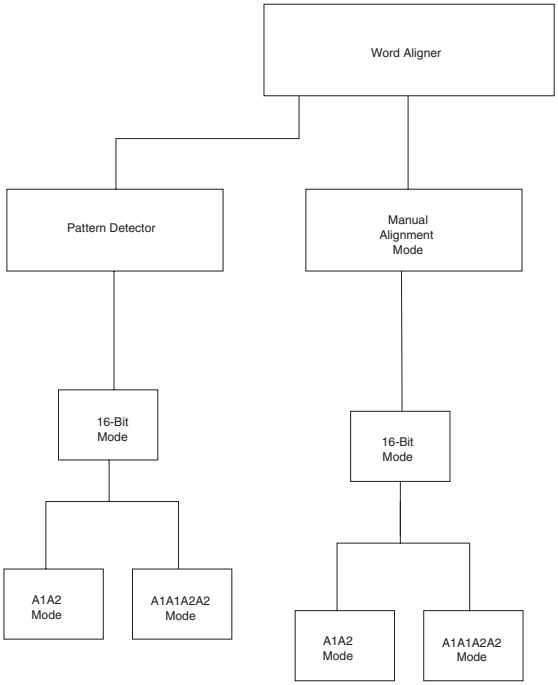
In the bit-slip mode, the byte boundary can be modified by a barrel shifter to slip the byte boundary one bit at a time via a user-controlled bit-slip port. The bit-slip mode supports both 8-bit and 10-bit data paths operating in a single or double-width mode.

The pattern detector is active in the bit-slip mode, and it detects the user-defined pattern that is specified in the MegaWizard® Plug-In Manager.

The bit-slip mode is available only in Custom mode and SONET mode.

Figure 2–16 shows the word aligner in 16-bit mode.

Figure 2–16. Word Aligner in 16-Bit Mode

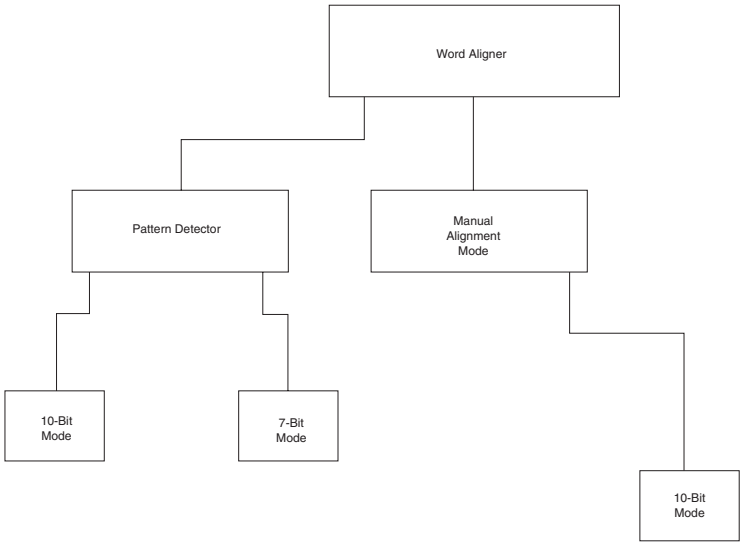


In the 16-bit mode, the word aligner and pattern detector automatically aligns and detects a user-defined 16-bit alignment pattern. This pattern can be in the format of A1A2 or A1A1A2A2 (for the SONET protocol). The re-alignment of the byte boundary can be done via a user-controlled port. The 16-bit mode supports only the 8-bit data path in a single-width or double-width mode.

The 16-bit mode is available only for the Custom mode and SONET mode. The A1A1A2A2 word alignment pattern option is available only for the SONET mode and cannot be used in the Custom mode.

Figure 2–17 shows the word aligner in 10-bit mode.

Figure 2–17. Word Aligner in 10-Bit Mode

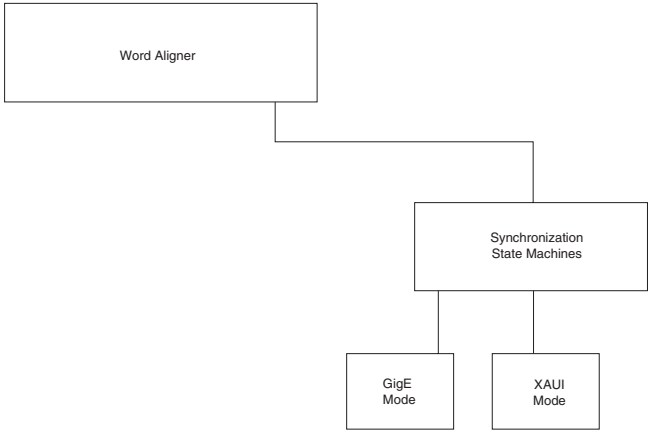


In the 10-bit mode, the word aligner automatically aligns the user's predefined 10-bit alignment pattern. The pattern detector can detect the full 10-bit pattern or only the lower seven bits of the pattern. The word aligner and pattern detector detect both the positive and the negative disparity of the pattern. A user-controlled enable port is available for the word aligner.

The 10-bit mode is available only for the Custom mode.

Figure 2–18 shows the word aligner in XAUI mode.

Figure 2–18. Word Aligner in XAUI Mode



In the XAUI and GIGE modes, the word alignment is controlled by a state machine that adheres to the IEEE 802.3ae standard for XAUI and the IEEE 802.3 standard for GIGE. The alignment pattern is predefined to be a /K28.5/ code group.

The XAUI mode is available only for the XAUI protocol, and the GIGE mode is available only for the GIGE protocol.

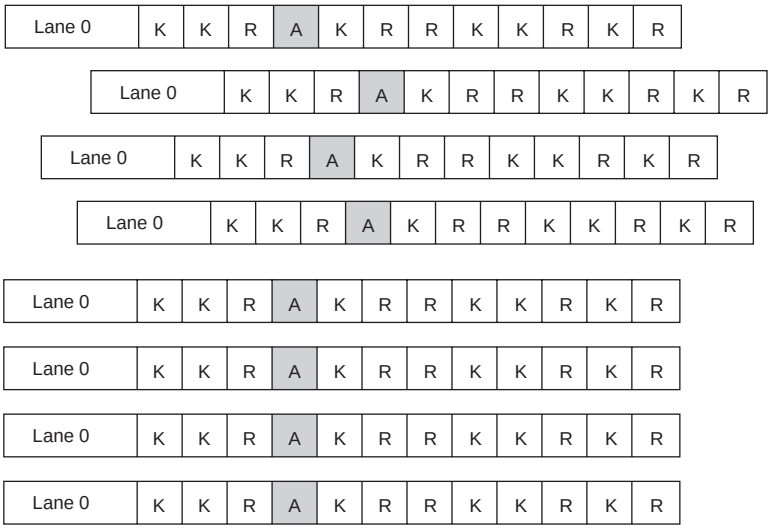
Channel Aligner

The channel aligner is available only in XAUI mode and bonds all four channels within a transceiver. The channel aligner adheres to the IEEE 802.3ae, clause 48 specification for channel bonding.

The channel aligner is a 16-word deep FIFO buffer with a state machine overlooking the channel bonding process. The state machine looks for an /A/ (/K28.3/) in each channel and aligns all the /A/s in the transceiver. When four columns of /A/ (denoted by //A//) are detected, the rx_channelalign port goes high, signifying that all the channels in the transceiver have been bonded. The reception of four consecutive misaligned /A/s restarts the channel alignment sequence and de-asserts rx_channelalign.

Figure 2–19 shows misaligned channels before the channel aligner and the channel alignment after the channel aligner.

Figure 2–19. Before & After the Channel Aligner



Rate Matcher

The rate matcher, which is available only in XAUI and GIGE modes, consists of a 12-word deep FIFO buffer and a FIFO controller. The rate matcher is bypassed when the device is not in XAUI or GIGE mode.

In a multi-crystal environment, the rate matcher compensates for up to a 100-ppm difference between the source and receiver clocks.

GIGE Mode

In the GIGE mode, the rate matcher adheres to the specifications in clause 36 of the IEEE 802.3 documentation, for idle additions or removals. The rate matcher performs clock compensation only on `/I2/` ordered sets, composing a `/K28.5/+` followed by a `/D16.2/-`. The rate matcher does not perform a clock compensation on any other ordered set combinations. An /I2/ is added or deleted automatically based on the number of words in the FIFO buffer. A `9'h19C` is given at the control and data ports when the FIFO is in an overflow or underflow condition.

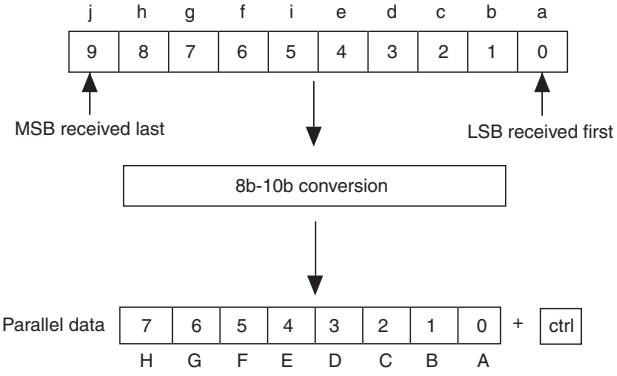
XAUI Mode

In XAUI mode, the rate matcher adheres to clause 48 of the IEEE 802.3ae specification for clock rate compensation. The rate matcher performs clock compensation on columns of /R/ (/K28.0/), denoted by //R//. An //R// is added or deleted automatically based on the number of words in the FIFO buffer.

8B/10B Decoder

The 8B/10B decoder converts the 10-bit encoded code group into 8-bit data and 1 control bit. The 8B/10B decoder can be bypassed. The following is a diagram of the conversion from a 10-bit encoded code group into 8-bit data + 1-bit control.

Figure 2–20. 8B/10B Decoder Conversion



There are two optional error status ports available in the 8B/10B decoder, rx_errdetect and rx_disperr. Table 2–7 shows the values of the ports from a given error. These status signals are aligned with the code group in which the error occurred.

Table 2–7. Error Signal Values

Types of Errors	rx_errdetect	rx_disperr
No errors	1'b0	1'b0
Invalid code groups	1'b1	1'b0
Disparity errors	1'b1	1'b1

Receiver State Machine

The receiver state machine operates in GIGE and XAUI modes. In GIGE mode, the receiver state machine replaces invalid code groups with `9'h1FE`. In XAUI mode, the receiver state machine translates the XAUI PCS code group to the XAUI XGMII code group. Table 2–8 shows the code conversion. The conversion adheres to the IEEE 802.3ae specification.

Table 2–8. Code Conversion

XGMII RXC	XGMII RXD	PCS code-group	Description
0	00 through FF	Dxx.y	Normal Data
1	07	K28.0 or K28.3 or K28.5	Idle in ||**I**||
1	07	K28.5	Idle in ||**T**||
1	9C	K28.4	Sequence
1	FB	K27.7	Start
1	FD	K29.7	Terminate
1	FE	K30.7	Error
1	FE	Invalid code group	Invalid XGMII character
1	See IEEE 802.3 reserved code groups	See IEEE 802.3 reserved code groups	Reserved code groups

Byte Deserializer

The byte deserializer takes a single width word (8 or 10 bits) from the transceiver logic and converts it into double-width words (16 or 20 bits) to the phase compensation FIFO buffer. The byte deserializer is bypassed when single width mode (8 or 10 bits) is used at the PLD interface.

Phase Compensation FIFO Buffer

The receiver phase compensation FIFO buffer resides in the transceiver block at the programmable logic device (PLD) boundary. This buffer compensates for the phase difference between the recovered clock within the transceiver and the recovered clock after it has transferred to the PLD core. The phase compensation FIFO buffer is four words deep and cannot be bypassed.

Loopback Modes

The Stratix GX transceiver has built-in loopback modes to aid in debug and testing. The loopback modes are set in the Stratix GX MegaWizard Plug-In Manager in the Quartus II software. Only one loopback mode can be set at any single instance of the transceiver block. The loopback mode applies to all used channels in a transceiver block.

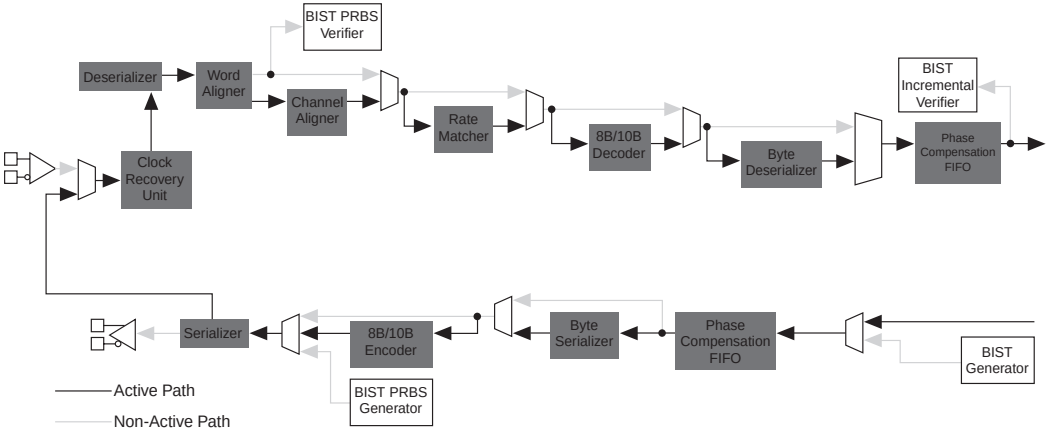
The available loopback modes are:

- Serial loopback
- Parallel loopback
- Reverse serial loopback

Serial Loopback

Serial loopback exercises all the transceiver logic except for the output buffer and input buffer. The loopback function is dynamically switchable through the `rx_slpbk` port on a channel by channel basis. The V_{OD} of the output reduced. If you select 400 mV, the output is tri-stated when the serial loopback option is selected. Figure 2–21 shows the data path in serial loopback mode.

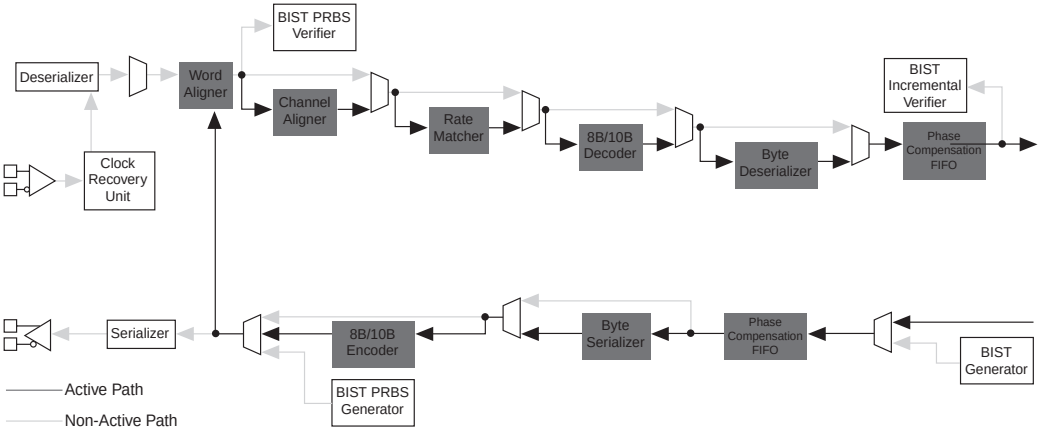
Figure 2–21. Data Path in Serial Loopback Mode



Parallel Loopback

The parallel loopback mode exercises the digital logic portion of the transceiver data path. The analog portions are not use in the loopback path. The received data is not retimed. Figure 2–22 shows the data path in parallel loopback mode. This option is not dynamically switchable. Reception of an external signal is not possible in this mode.

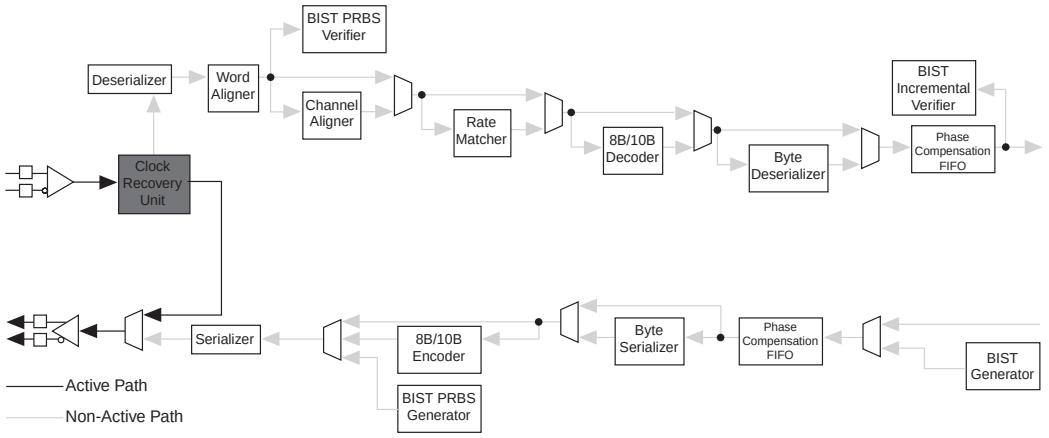
Figure 2–22. Data Path in Parallel Loopback Mode



Reverse Serial Loopback

The reverse serial loopback exercises the analog portion of the transceiver. This loopback mode is dynamically switchable through the `tx_srlpbk` port on a channel by channel basis. Asserting `rxanalogreset` in reverse serial loopback mode powers down the receiver buffer and CRU, preventing data loopback. Figure 2–23 shows the data path in reverse serial loopback mode.

Figure 2–23. Data Path in Reverse Serial Loopback Mode



BIST (Built-In Self Test)

The Stratix GX transceiver has built-in self test modes to aid in debug and testing. The BIST modes are set in the Stratix GX MegaWizard Plug-In Manager in the Quartus II software. Only one BIST mode can be set for any single instance of the transceiver block. The BIST mode applies to all channels used in a transceiver.

The following is a list of the available BIST modes:

- PRBS generator and verifier
- Incremental mode generator and verifier
- High-frequency generator
- Low-frequency generator
- Mixed-frequency generator

Figures 2–24 and 2–25 are diagrams of the BIST PRBS data path and the BIST incremental data path, respectively.

Figure 2–24. BIST PRBS Data Path

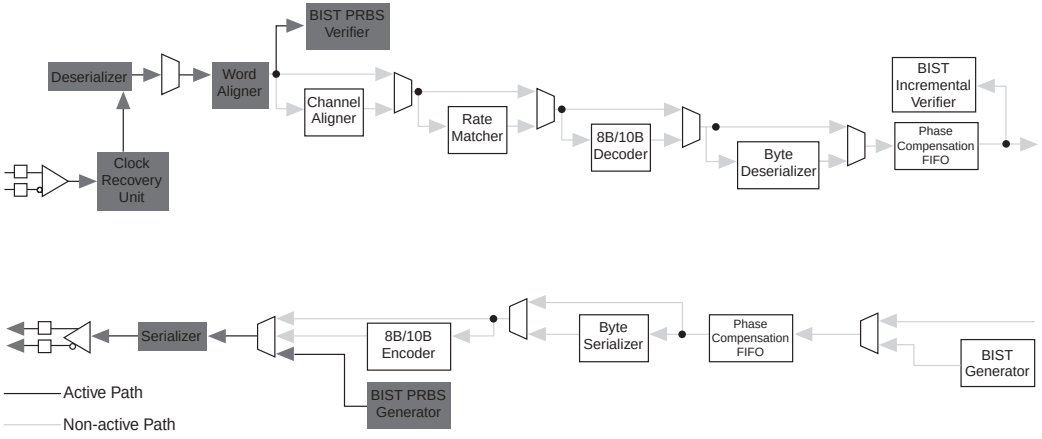


Figure 2–25. BIST Incremental Data Path

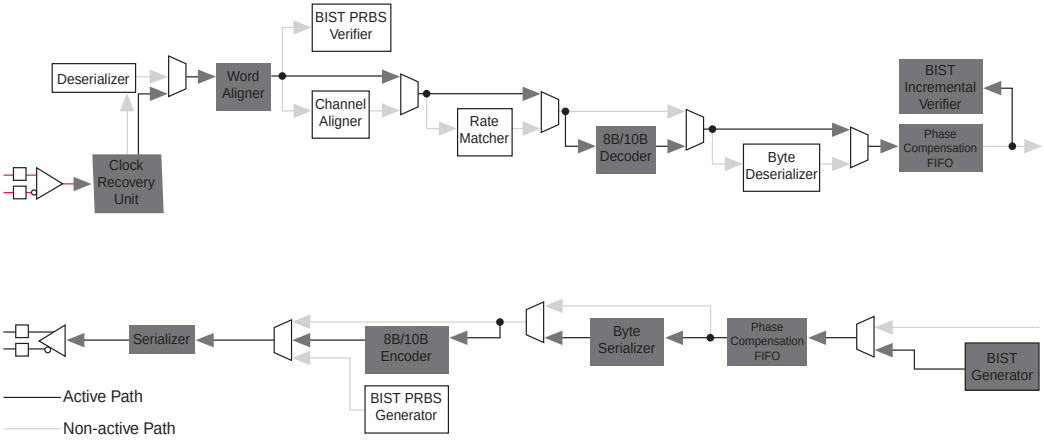


Table 2–9 shows the BIST data output and verifier alignment pattern.

Table 2–9. BIST Data Output & Verifier Alignment Pattern (Part 1 of 2)

BIST Mode	Output	Polynomials	Verifier Word Alignment Pattern
PRBS 8-bit	$2^8 - 1$	$x^8 + x^7 + x^5 + x^3 + 1$	1000000011111111
PRBS 10-bit	$2^{10} - 1$	$x^{10} + x^7 + 1$	1111111111

Table 2–9. BIST Data Output & Verifier Alignment Pattern (Part 2 of 2)

BIST Mode	Output	Polynomials	Verifier Word Alignment Pattern
PRBS 16-bit	$2^8 - 1$	$x^8 + x^7 + x^5 + x^3 + 1$	1000000011111111
PRBS 20-bit	$2^{10} - 1$	$x^{10} + x^7 + 1$	1111111111
Incremental 10-bit	K28.5, K27.7, Data (00-FF incremental), K28.0, K28.1, K28.2, K28.3, K28.4, K28.6, K28.7, K23.7, K30.7, K29.7 *(1)*		0101111100 (K28.5)
Incremental 20-bit	K28.5, K27.7, Data (00-FF incremental), K28.0, K28.1, K28.2, K28.3, K28.4, K28.6, K28.7, K23.7, K30.7, K29.7 *(1)*		0101111100 (K28.5)
High frequency	1010101010		
Low frequency	0011111000		
Mixed frequency	0011111010 or 1100000101		

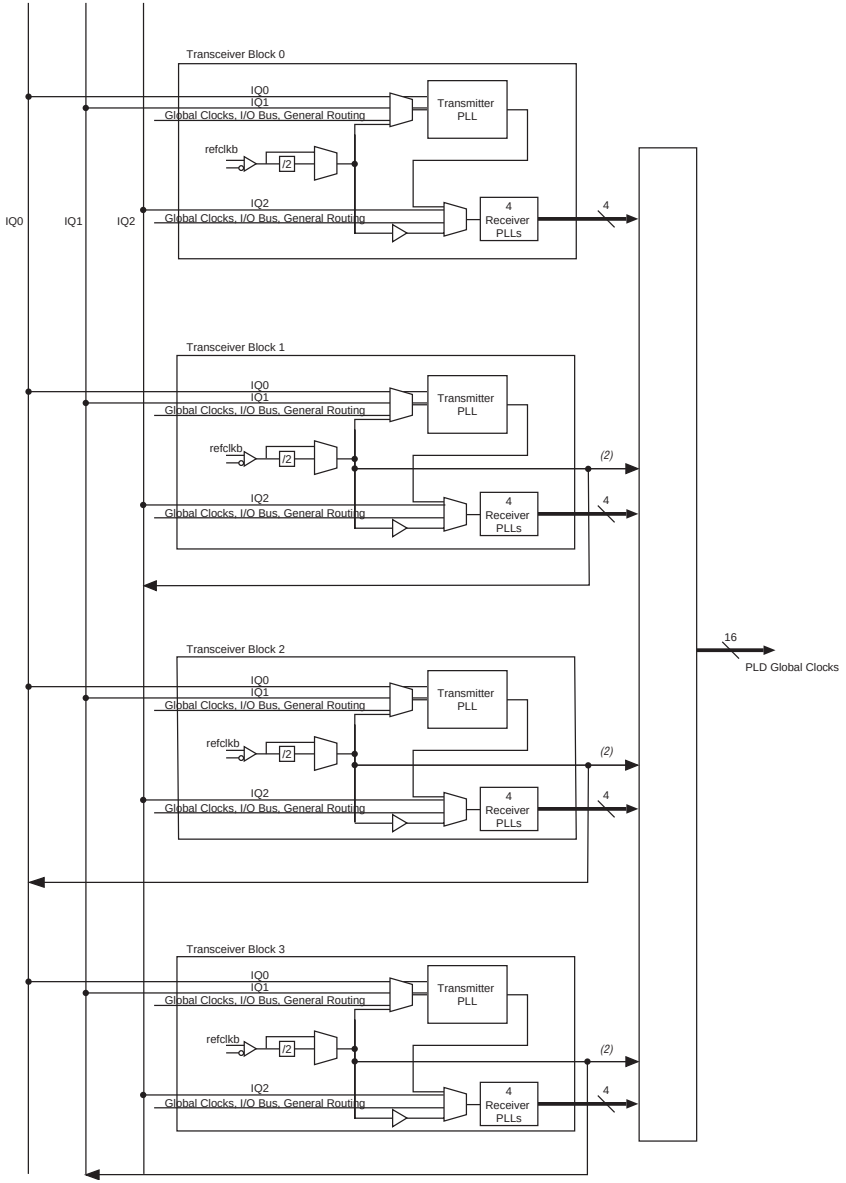
Note to Table 2–9:
(1) This output repeats.

Stratix GX Clocking

The Stratix GX global clock can be driven by certain REFCLKB pins, all transmitter PLL outputs, and all receiver PLL outputs. The REFCLKB pins (except for transceiver block 0 and transceiver block 4) can drive inter-transceiver and global clock lines as well as feed the transmitter and receiver PLLs. The output of the transmitter PLL can only feed global clock lines and the reference clock port of the receiver PLL.

Figures 2–26 and 2–27 are diagrams of the Inter-Transceiver line connections as well as the global clock connections for the EP1SGX25F and EP1SGX40G devices. For devices with fewer transceivers, ignore the information about the unavailable transceiver blocks.

Figure 2–26. EP1SGX25F Device Inter-Transceiver & Global Clock Connections *Note (1)*



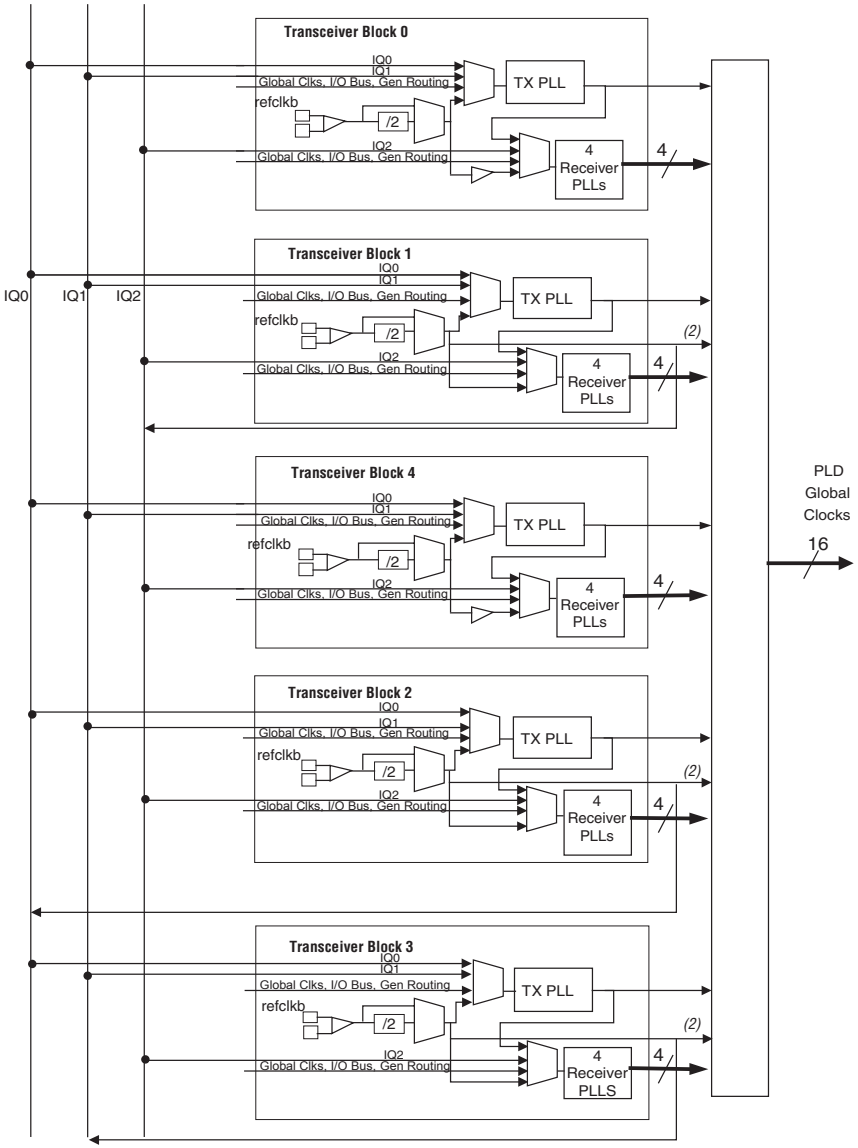
Notes to Figure 2–26:

(1) IQ lines are inter-transceiver block lines.

(2) If the /2 pre-divider is used, the path to drive the PLD logic array, local, or global clocks is not allowed.

(3) There are four receiver PLLs in each transceiver block.

Figure 2–27. EP1SGX40G Device Inter-Transceiver & Global Clock Connections *Note (1)*



Notes to Figure 2–27:

(1) IQ lines are inter-transceiver block lines.

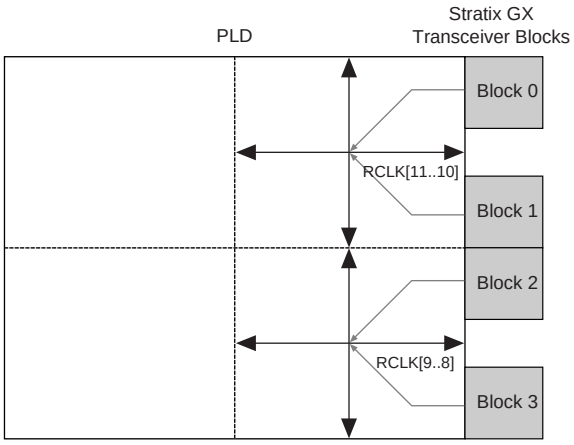
(2) If the /2 pre-divider is used, the path to drive the PLD logic array, local, or global clocks is not allowed.

(3) There are four receiver PLLs in each transceiver block.

The receiver PLL can also drive the fast regional, regional clocks, and local routing adjacent to the associated transceiver block. Figures 2–28 through 2–31 show which fast regional and regional clock resource can be used by the recovered clock.

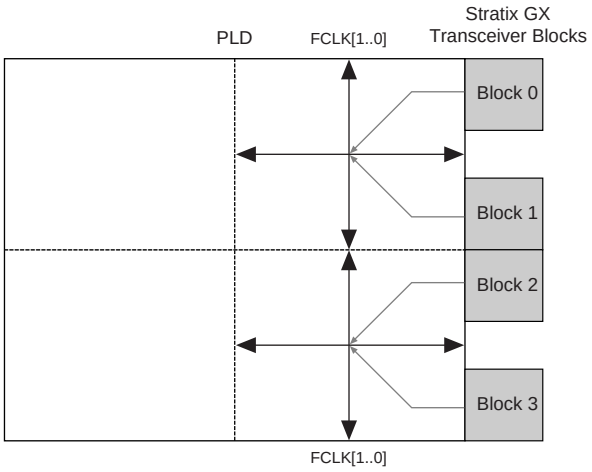
In the EP1SGX25 device, the receiver PLL recovered clocks from transceiver blocks 0 and 1 drive `RCLK[1..0]` while transceiver blocks 2 and 3 drive `RCLK[7..6]`. The regional clocks feed logic in their associated regions.

Figure 2–28. EP1SGX25 Receiver PLL Recovered Clock to Regional Clock Connection



In addition, the receiver PLL's recovered clocks can drive fast regional lines (`FCLK`) as shown Figure 2–29. The fast regional clocks can feed logic in their associated regions.

Figure 2–29. EP1SGX25 Receiver PLL Recovered Clock to Fast Regional Clock Connection



In the EP1SGX40 device, the receiver PLL recovered clocks from transceivers 0 and 1 drive `RCLK[1..0]` while transceivers 2, 3, and 4 drive `RCLK[7..6]`. The regional clocks feed logic in their associated regions.

Figure 2–30. EP1SGX40 Receiver PLL Recovered Clock to Regional Clock Connection

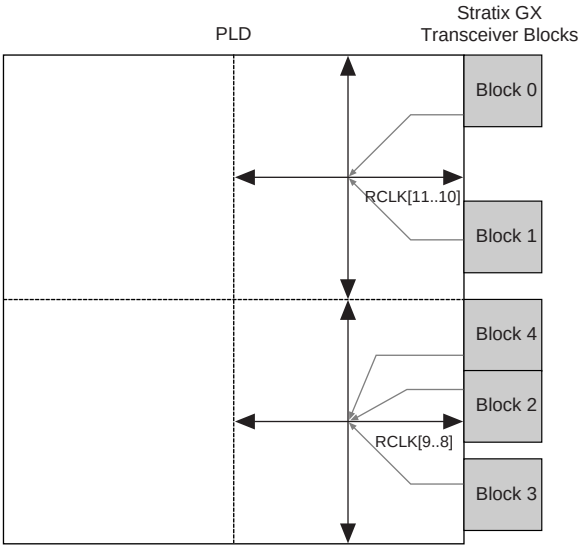


Figure 2–31 shows the possible recovered clock connection to the fast regional clock resource. The fast regional clocks can drive logic in their associated regions.

Figure 2–31. EP1SGX40 Receiver PLL Recovered Clock to Fast Regional Clock Connection

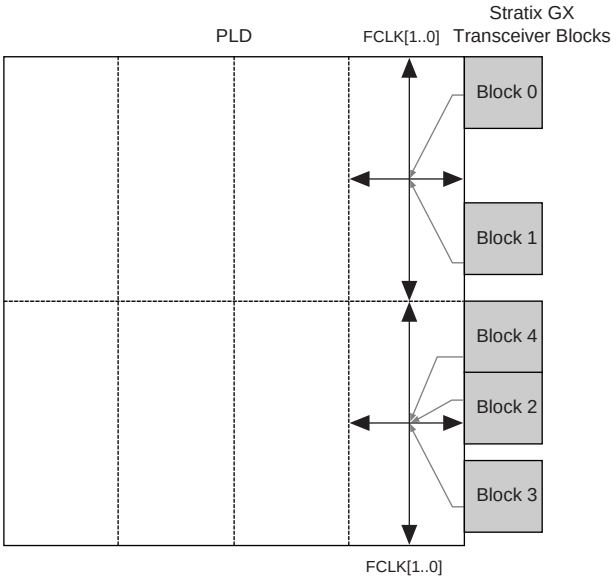


Table 2–10 summarizes the possible clocking connections for the transceivers.

Table 2–10. Possible Clocking Connections for Transceivers (Part 1 of 2)

Source	Destination					
	Transmitter PLL	Receiver PLL	GCLK	RCLK	FCLK	IQ Lines
REFCLKB	✓	✓	✓ *(1)*	✓		✓ *(1)*
Transmitter PLL		✓	✓	✓	✓	
Receiver PLL			✓	✓	✓	
GCLK	✓	✓				
RCLK	✓	✓				
FCLK	✓	✓				

Table 2–10. Possible Clocking Connections for Transceivers (Part 2 of 2)

Source	Destination					
	Transmitter PLL	Receiver PLL	GCLK	RCLK	FCLK	IQ Lines
IQ lines	✓ (2)	✓ (2)				

Notes to Table 2–10:

(1) REFCLKB from transceiver block 0 and transceiver block 4 does not drive the inter-transceiver lines or the GCLK lines.

(2) Inter-transceiver line 0 and inter-transceiver line 1 drive the transmitter PLL, while inter-transceiver line 2 drives the receiver PLLs.

Other Transceiver Features

Other important features of the Stratix GX transceivers are the power down and reset capabilities, the external voltage reference and bias circuitry, and hot swapping.

Individual Power-Down & Reset for the Transmitter & Receiver

Stratix GX transceivers offer a power saving advantage with their ability to shut off functions that are not needed. The device can individually reset the receiver and transmitter blocks and the PLLs. The Stratix GX device can either globally power down and reset the transmitter and receiver channels or do each channel separately. Table 2–11 shows the connectivity between the reset signals and the Stratix GX logical blocks.

Power-down functions are static, in other words., they are implemented upon device configuration and programmed, through the Quartus II software, to static values. Resets can be static as well as dynamic inputs coming from the logic array or pins.

Table 2–11. Reset Signal Map to Stratix GX Blocks

Reset Signal	Transmitter Phase Compensation FIFO Module/ Byte Serializer	Transmitter 8B/10B Encoder	Transmitter Serializer	Transmitter Analog Circuits	Transmitter PLL	Transmitter XAUI State Machine	Transmitter Analog Circuits	BIST Generators	Receiver Deserializer	Receiver Word Aligner	Receiver Deskew FIFO Module	Receiver Rate Matcher	Receiver 8B/10B Decoder	Receiver Phase Comp FIFO Module/ Byte Deserializer	Receiver PLL / CRU	Receiver XAUI State Machine	BIST Verifiers	Receiver Analog Circuits
rxdigitalreset										✓	✓	✓	✓	✓		✓	✓	
rxanalogreset									✓						✓			✓
txdigitalreset	✓	✓				✓		✓										
pll_areset	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
pllenable	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Voltage Reference Capabilities

Stratix GX transceivers provide voltage reference and bias circuitry. To set-up internal bias for controlling the transmitter output drivers' voltage swing—as well as to provide voltage/current biasing for other analog circuitry—use the internal bandgap voltage reference at 0.7 V. To provide bias for internal pull-up PMOS resistors for I/O termination at the serial interface of receiver and transmitter channels (independent of power supply drift, process changes, or temperature variation) an external resistor, which is connected to the external low voltage power supply, is

accurately tracked by the internal bias circuit. Moreover, the reference voltage and internal resistor bias current is generated and replicated to the analog circuitry in each channel.

Hot-Socketing Capabilities

Each Stratix GX device is capable of hot-socketing. Because Stratix GX devices can be used in a mixed-voltage environment, they have been designed specifically to tolerate any possible power-up sequence. Signals can be driven into Stratix GX devices before and during power-up without damaging the device. Once operating conditions are reached and the device is configured, Stratix GX devices operate according to your specifications. This feature provides the Stratix GX transceiver line card behavior, so you can insert it into the system without powering the system down, offering more flexibility.

Applications & Protocols Supported with Stratix GX Devices

Each Stratix GX transceiver block is designed to operate at any serial bit rate from 500 Mbps to 3.1875 Gbps per channel. The wide, data rate range allows Stratix GX transceivers to support a wide variety of standard and future protocols such as 10-Gigabit Ethernet XAUI, InfiniBand, Fibre Channel, and Serial RapidIO. Stratix GX devices are ideal for many high-speed communication applications such as high-speed backplanes, chip-to-chip bridges, and high-speed serial communications standards support.

Stratix GX Example Application Support

Stratix GX devices can be used for many applications, including:

- Backplanes for traffic management and quality of service (QOS)
- Switch fabric applications for complete set for backplane and switch fabric transceivers
- Chip-to-chip applications such as: 10 Gigabit Ethernet XAUI to XGMII bridge, 10 Gigabit Ethernet XGMII to POS-PHY4 bridge, POS-PHY4 to NPSI bridge, or NPSI to backplane bridge

High-Speed Serial Bus Protocols

With wide, serial data rate range, Stratix GX devices can support multiple, high-speed serial bus protocols. Table 2–12 shows some of the protocols that Stratix GX devices can support.

Table 2–12. High-Speed Serial Bus Protocols

Bus Transfer Protocol	Stratix GX (Gbps) (Supports up to 3.1875 Gbps)
SONET backplane	2.488
10 Gigabit Ethernet XAUI	3.125
10 Gigabit fibre channel	3.1875
InfiniBand	2.5
Fibre channel (1G, 2G)	1.0625, 2.125
Serial RapidIO™	1.25, 2.5, 3.125
PCI Express	2.5
SMPTE 292M	1.485

3. Source-Synchronous Signaling With DPA

SGX51003-1.1

Introduction

Expansion in the telecommunications market and growth in Internet use requires systems to move more data faster than ever. To meet this demand, rely on solutions such as differential signaling and emerging high-speed interface standards including RapidIO, POS-PHY 4, SFI-4, or XSBI.

These new protocols support differential data rates up to 1 Gbps and higher. At these high data rates, it becomes more challenging to manage the skew between the clock and data signals. One solution to this challenge is to use CDR to eliminate skew between data channels and clock signals. Another potential solution, DPA, is beginning to be incorporated into some of these protocols.

The source-synchronous high-speed interface in Stratix GX devices is a dedicated circuit embedded into the PLD allowing for high-speed communications. The *High-Speed Source-Synchronous Differential I/O Interfaces in Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2* provides information on the high-speed I/O standard features and functions of the Stratix GX device.

Stratix GX I/O Banks

Stratix GX devices contain 17 I/O banks. I/O banks one and two support high-speed LVDS, LVPECL, and 3.3-V PCML inputs and outputs. These two banks also incorporate an embedded dynamic phase aligner within the source-synchronous interface (see Figure 3–8 on page 3–10). The dynamic phase aligner corrects for the phase difference between the clock and data lines caused by skew. The dynamic phase aligner operates automatically and continuously without requiring a fixed training pattern, and allows the source-synchronous circuitry to capture data correctly regardless of the channel-to-clock skew.

Principles of SERDES Operation

Stratix GX devices support source-synchronous differential signaling up to 1 Gbps in DPA mode, and up to 840 Mbps in non-DPA mode. Serial data is transmitted and received along with a low-frequency clock. The PLL can multiply the incoming low-frequency clock by a factor of 1 to 10. The SERDES factor *J* can be 8 or 10 for the DPA mode, or 4, 7, 8, or 10 for all other modes. The SERDES factor does not have to equal the clock

multiplication value. The ×1 and ×2 operation is also possible by bypassing the SERDES. The SERDES DPA cannot support ×1, ×2, or ×4 natively.

On the receiver side, the high-frequency clock generated by the PLL shifts the serial data through a shift register (also called deserializer). The parallel data is clocked out to the logic array synchronized with the low-frequency clock. On the transmitter side, the parallel data from the logic array is first clocked into a parallel-in, serial-out shift register synchronized with the low-frequency clock and then transmitted out by the output buffers.

There are two dedicated fast PLLs each in EP1SGX10 to EP1SGX25 devices, and four in EP1SGX40 devices. These PLLs are used for the SERDES operations as well as general-purpose use.

Stratix GX Differential I/O Receiver Operation (Non-DPA Mode)

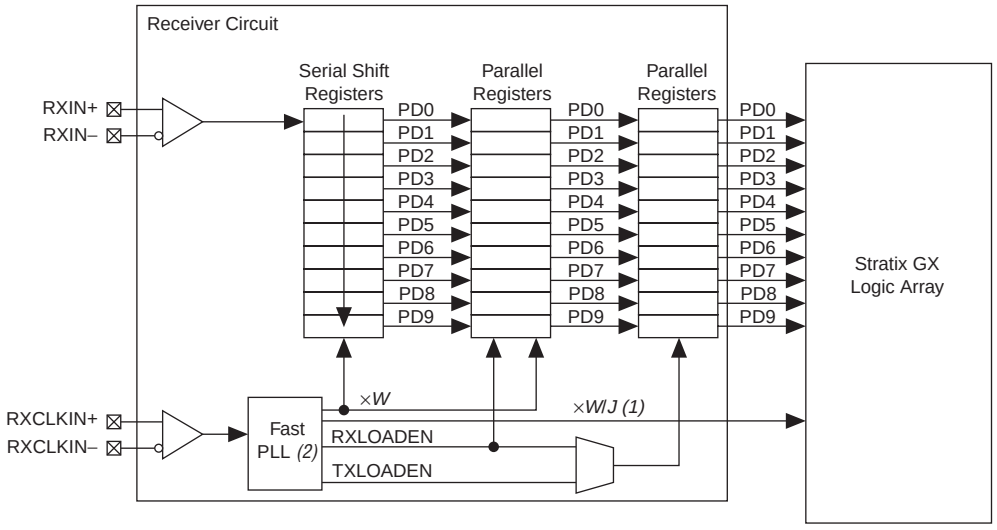
You can configure any of the Stratix GX source synchronous differential input channels as a receiver channel (see Figure 3–1). The differential receiver deserializes the incoming high-speed data. The input shift register continuously clocks the incoming data on the negative transition of the high-frequency clock generated by the PLL clock (×*W*).

The data in the serial shift register is shifted into a parallel register by the RXLOADEN signal generated by the fast PLL counter circuitry on the third falling edge of the high-frequency clock. However, you can select which falling edge of the high frequency clock loads the data into the parallel register, using the data-realignment circuit.

In normal mode, the enable signal RXLOADEN loads the parallel data into the next parallel register on the second rising edge of the low-frequency clock. You can also load data to the parallel register through the TXLOADEN signal when using the data-realignment circuit.

Figure 3–1 shows the block diagram of a single SERDES receiver channel. Figure 3–2 shows the timing relationship between the data and clocks in Stratix GX devices in ×10 mode. *W* is the low-frequency multiplier and *J* is the data parallelization division factor.

Figure 3–1. Stratix GX High-Speed Interface Deserialized in ×10 Mode

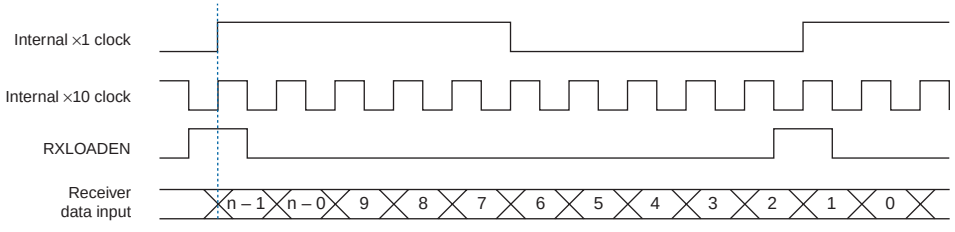


Notes to Figure 3–1:

(1) W = 1, 2, 4, 7, 8, or 10.
J = 4, 7, 8, or 10 for non-DPA (J = 8 or 10 for DPA).
W does not have to equal J. When J = 1 or 2, the deserializer is bypassed. When J = 2, the device uses DDRIO registers.

(2) This figure does not show additional circuitry for clock or data manipulation.

Figure 3–2. Receiver Timing Diagram



Stratix GX Differential I/O Transmitter Operation

You can configure any of the Stratix GX differential output channels as a transmitter channel. The differential transmitter serializes outbound parallel data.

The logic array sends parallel data to the SERDES transmitter circuit when the TXLOADEN signal is asserted. This signal is generated by the high-speed counter circuitry of the logic array low-frequency clock's rising edge. The data is then transferred from the parallel register into the serial shift register by the TXLOADEN signal on the third rising edge of the high-frequency clock.

Figure 3–3 shows the block diagram of a single SERDES transmitter channel and Figure 3–4 shows the timing relationship between the data and clocks in Stratix GX devices in ×10 mode. *W* is the low-frequency multiplier and *J* is the data parallelization division factor.

Figure 3–3. Stratix GX High-Speed Interface Serialized in ×10 Mode

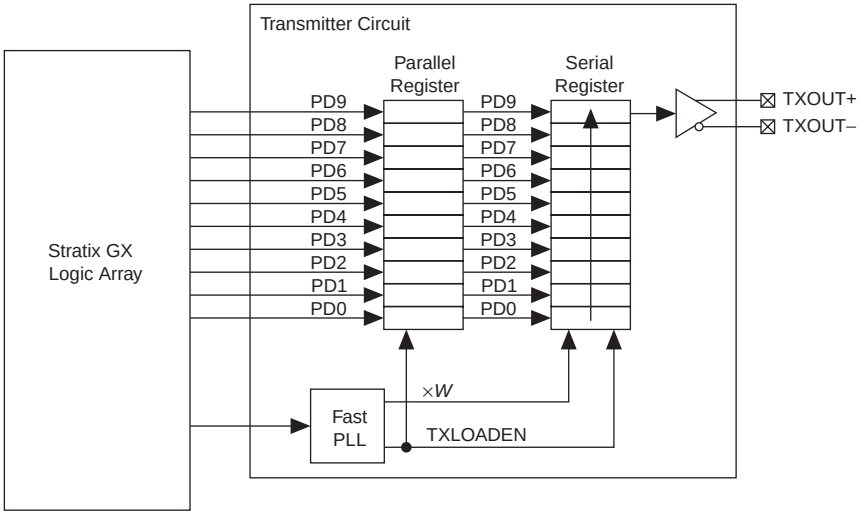
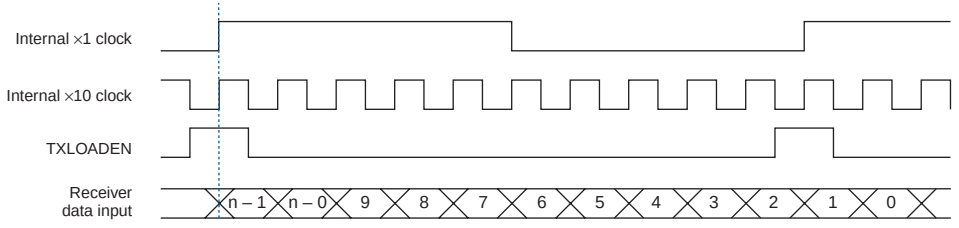


Figure 3–4. Transmitter Timing Diagram

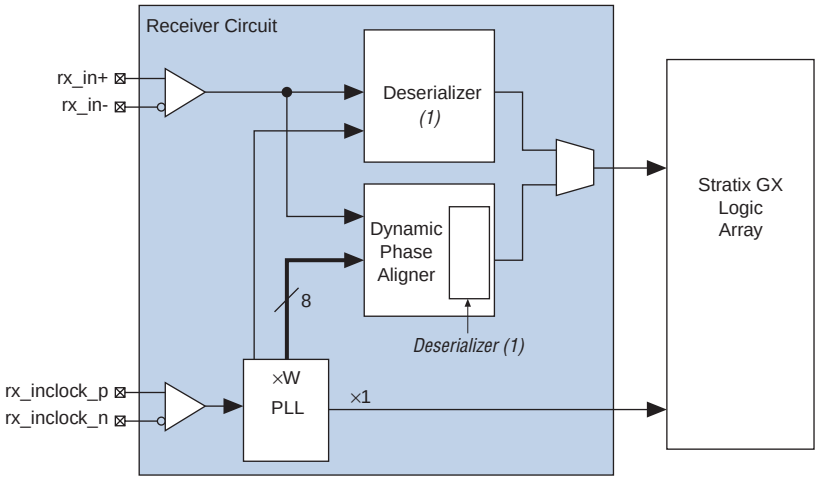


DPA Block Overview

Each Stratix GX receiver channel features a DPA block. The block contains a dynamic phase selector for phase detection and selection, a SERDES, a synchronizer, and a data realigner circuit. You can bypass the dynamic phase aligner without affecting the basic source-synchronous operation of the channel by using a separate deserializer shown in Figure 3–5.

The dynamic phase aligner uses both the source clock and the serial data. The dynamic phase aligner automatically and continuously tracks fluctuations caused by system variations and self-adjusts to eliminate the phase skew between the multiplied clock and the serial data. Figure 3–5 shows the relationship between Stratix GX source-synchronous circuitry and the Stratix GX source-synchronous circuitry with DPA.

Figure 3–5. Source-Synchronous DPA Circuitry



Note to Figure 3–5:

(1) Both deserializers are identical. The deserializer operation is described in the "Principles of SERDES Operation" section.

Unlike the de-skew function in APEX™ 20KE and APEX 20KC devices, you do not have to use a fixed training pattern with DPA in Stratix GX devices. Table 3–1 shows the differences between source-synchronous circuitry with DPA and source-synchronous circuitry without DPA circuitry in Stratix GX devices.

Table 3–1. Source-Synchronous Circuitry With & Without DPA

Feature	Source-Synchronous Circuitry	
	Without DPA	With DPA
Data rate	300 to 840 Megabits per second (Mbps)	300 Mbps to 1 Gbps
Deserialization factors	1, 2, 4, 8, 10	8, 10
Clock frequency	10 to 717 MHz	74 to 717 MHz
Interface pins	I/O banks 1 and 2	I/O banks 1 and 2
Receiver pins	Dedicated inputs	Dedicated inputs

DPA Input Support

Stratix GX device I/O banks 1 and 2 contain dedicated circuitry to support differential I/O standards at speeds up to 1 Gbps with DPA (or up to 840 Mbps without DPA). Stratix GX device source-synchronous circuitry supports LVDS, LVPECL, and 3.3-V PCML I/O standards, each with a supply voltage of 3.3 V. Refer to the *High-Speed Source-Synchronous Differential I/O Interfaces in Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2* for more information on these I/O standards. Transmitter pins can be either input or output pins for single-ended I/O standards. Refer to Table 3–2.

Table 3–2. Bank 1 & 2 Input Pins

Input Pin Type	I/O Standard	Receiver Pin	Transmitter Pin
Differential	Differential	Input only	Output only
Single ended	Single ended	Input only	Input or output

Interface & Fast PLL

This section describes the number of channels that support DPA and their relationship with the PLL in Stratix GX devices. EP1SGX10 and EP1SGX25 devices have two dedicated fast PLLs and EP1SGX40 devices

have four dedicated fast PLLs for clock multiplication. Table 3–3 shows the maximum number of channels in each Stratix GX device that support DPA.

Table 3–3. Stratix GX Source-Synchronous Differential I/O Resources

Device	Fast PLLs	Pin Count	Receiver Channels *(1)*	Transmitter Channels *(1)*	Receiver & Transmitter Channel Speed (Gbps) *(2)*	LEs
EP1SGX10C	2 *(3)*	672	22	22	1	10,570
EP1SGX10D	2 *(3)*	672	22	22	1	10,570
EP1SGX25C	2	672	39	39	1	25,660
EP1SGX25D	2	672	39	39	1	25,660
		1,020	39	39	1	25,660
EP1SGX25F	2	1,020	39	39	1	25,660
EP1SGX40D	4 *(4)*	1,020	45	45	1	41,250
EP1SGX40G	4 *(4)*	1,020	45	45	1	41,250

Notes to Table 3–3:

(1) This is the number of receiver or transmitter channels in the source-synchronous (I/O bank 1 and 2) interface of the device.

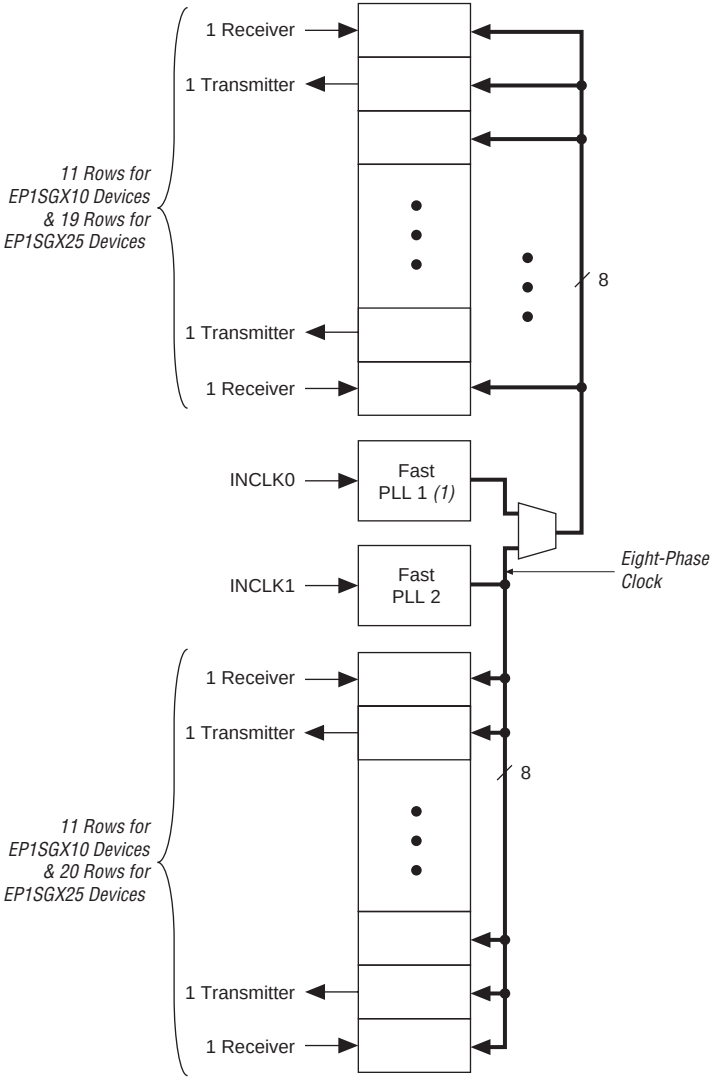
(2) Receiver channels operate at 1,000 Mbps with DPA. Without DPA, the receiver channels operate at 840 Mbps.

(3) One of the two fast PLLs in EP1SGX10C and EP1SGX10D devices supports DPA.

(4) Two of the four fast PLLs in EP1SGX40D and EP1SGX40G devices support DPA

The receiver and transmitter channels are interleaved so that each I/O row in I/O banks 1 and 2 of the device has one receiver channel and one transmitter channel per row. Figures 3–6 and 3–7 show the fast PLL and channels with DPA layout in EP1SGX10, EP1SGX25, and EP1SGX40 devices. In EP1SGX10 devices, only fast PLL 2 supports DPA operations.

Figure 3–6. PLL & Channel Layout in EP1SGX10 & EP1SGX25 Devices *Notes (1), (2)*

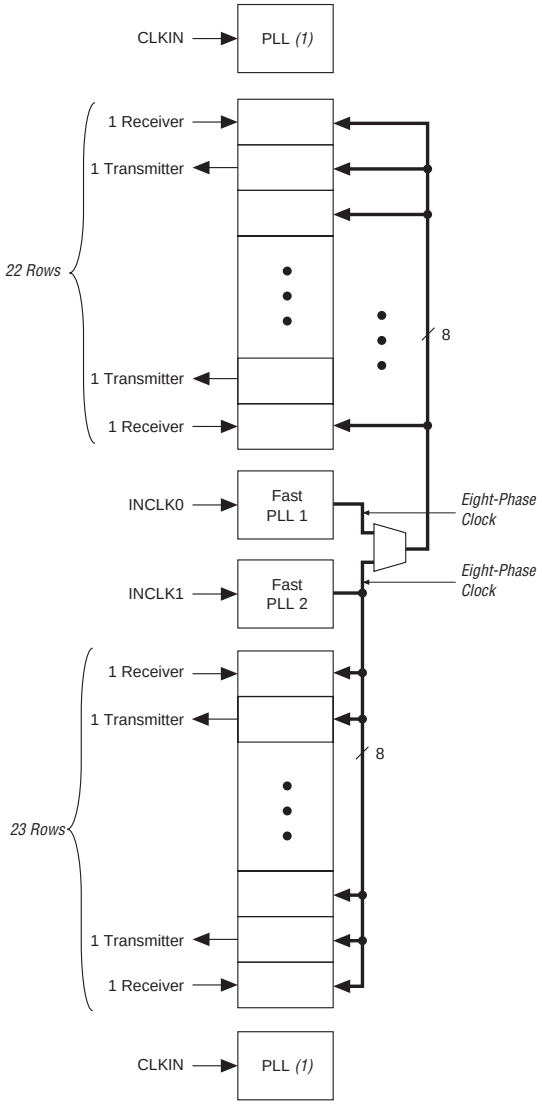


Notes to Figure 3–6:

(1) Fast PLL 1 in EP1SGX10 devices does not support DPA.

(2) Not all eight phases are used by the receiver channel or transmitter channel in non-DPA mode.

Figure 3–7. PLL & Channel Layout in EP1SGX40 Devices *Notes (1), (2), (3)*



Notes to Figure 3–7:

(1) Corner PLLs do not support DPA.

(2) Not all eight phases are used by the receiver channel or transmitter channel in non-DPA mode.

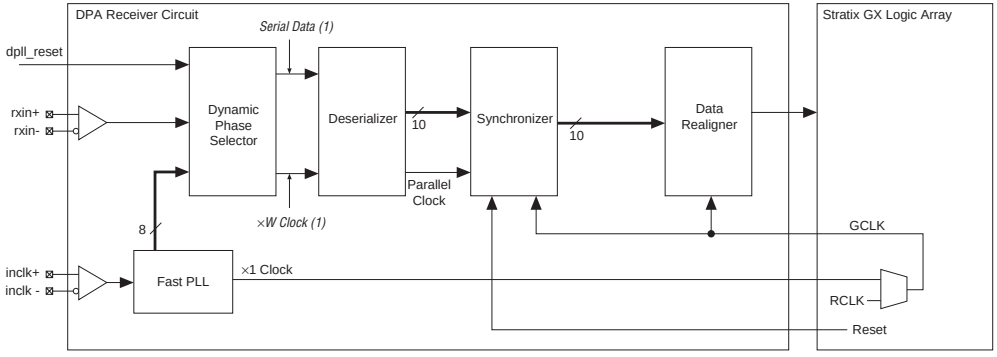
(3) The center PLLs can only clock 20 transceivers in either direction. Using Fast PLL2, you can clock a total of 40 transceivers, 20 in each direction.

DPA Operation

The DPA receiver circuitry contains the dynamic phase selector, the deserializer, the synchronizer, and the data realigner (see Figure 3–8). This section describes the DPA operation, synchronization and data realignment. In the SERDES with DPA mode, the source clock is fed to the fast PLL through the dedicated clock input pins. This clock is multiplied by the multiplication value *W* to match the serial data rate.

For information on the deserializer, see "Principles of SERDES Operation" on page 3–1.

Figure 3–8. DPA Receiver Circuit

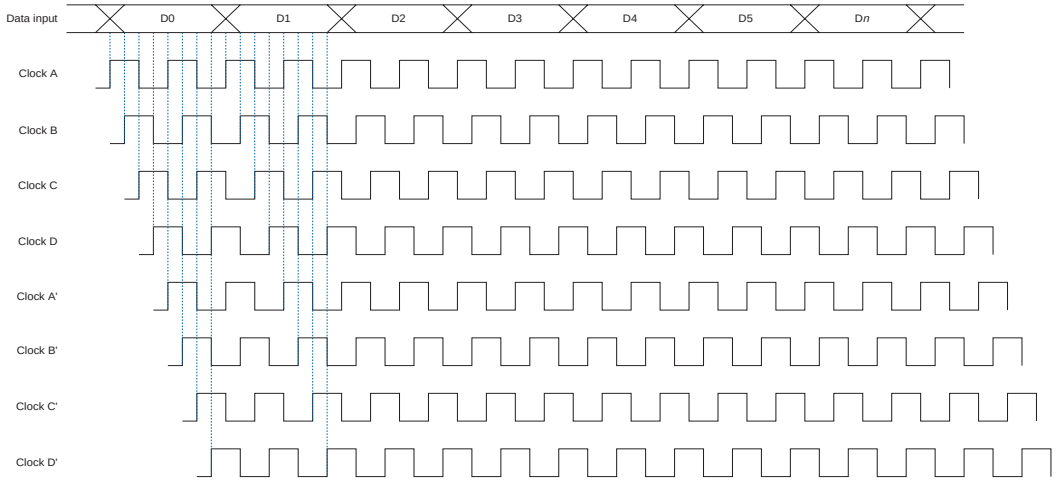


Note to Figure 3–8:
(1) These are phase-matched and retimed high-speed clocks and data.

The dynamic phase selector matches the phase of the high-speed clock and data before sending them to the deserializer.

The fast PLL supplies eight phases of the same clock (each a separate tap from a four-stage differential VCO) to all the differential channels associated with the selected fast PLL. The DPA circuitry inside each channel locks to a phase closest to the serial data's phase and sends the retimed data and the selected clock to the deserializer. The DPA circuitry automatically performs this operation and is not something you select. Each channel's DPA circuit can independently choose a different clock phase. The data phase detection and the clock phase selection process is automatic and continuous. The eight phases of the clock give the DPA circuit a granularity of one eighth of the unit interval (UI) or 125 ps at 1 Gbps. Figure 3–9 illustrates the clocks generated by the fast PLL circuitry and their relationship to a data stream.

Figure 3–9. Fast PLL Clocks & Data Input



Protocols, Training Pattern & DPA Lock Time

The dynamic phase aligner uses a fast PLL for clock multiplication, and the dynamic phase selector for the phase detection and alignment. The dynamic phase aligner uses the high-speed clock out of the dynamic phase selector to deserialize high-speed data and the receiver's source synchronous operations.

At each rising edge of the clock, the dynamic phase selector determines the phase difference between the clock and the data and automatically compensates for the phase difference between the data and clock.

The actual lock time for different data patterns varies depending on the data's transition density (how often the data switches between 1 and 0) and jitter characteristic. The DPA circuitry is designed to lock onto any data pattern with sufficient transition density, so the circuitry works with current and future protocols. Experiments and simulations show that the DPA circuitry locks when the data patterns listed in Table 3–4 are repeated for the specified number of times. There are other suitable patterns not shown in Table 3–4 and/or pattern lengths, but the lock time may vary. The circuit can adjust for any phase variation that may occur during operation.

Table 3–4. Training Patterns for Different Protocols

Protocols	Training Pattern	Number of Repetitions
SPI-4, NPSI	Ten 0's, ten 1's (00000000001111111111)	256
RapidIO	Four 0's, four 1's (00001111) or one 1, two 0's, one 1, four 0's (10010000)	
Other designs	Eight alternating 1's and 0's (10101010 or 01010101)	
SFI-4, XSBI	Not specified	

Phase Synchronizer

Each receiver has its own phase synchronizer. The receiver phase synchronizer aligns the phase of the parallel data from all the receivers to one global clock. The synchronizers in each channel consist of a 4-bit deep and *J*-bit wide FIFO buffer. The parallel clock writes to the FIFO buffer and the global clock (GCLK) reads from the FIFO buffer. The global and parallel clock inputs into the synchronizers must have identical frequencies and differ only in phase. The FIFO buffer never becomes full or empty (because the source and receive signals are frequency locked) when operating within the DPA specifications, and the operation does not require an empty/full flag or read/write enable signals.

Receiver Data Realignment In DPA Mode

While DPA operation aligns the incoming clock phase to the incoming data phase, it does not guarantee the parallelization boundary or byte boundary. When the dynamic phase aligner realigns the data bits, the bits may be shifted out of byte alignment, as shown in Figure 3–10.

Figure 3–10. Misaligned Captured Bits



The dynamic phase selector and synchronizer align the clock and data based on the power-up of both communicating devices, and the channel to channel skew. However, the dynamic phase selector and synchronizer cannot determine the byte boundary, and the data may need to be byte-aligned. The dynamic phase aligner's data realignment circuitry shifts data bits to correct bit misalignments.

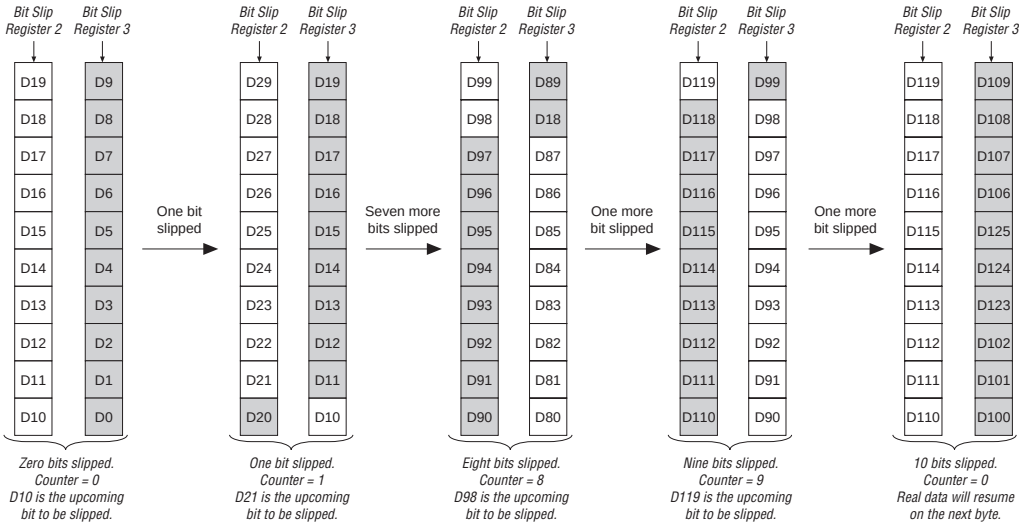
The Stratix GX circuitry contains a data-realignment feature controlled by the logic array. Stratix GX devices perform data realignment on the parallel data after the deserialization block. The data realignment can be performed per channel for more flexibility. The data alignment operation requires a state machine to recognize a specific pattern. The procedure requires the bits to be slipped on the data stream to correctly align the incoming data to the start of the byte boundary.

The DPA uses its realignment circuitry and the global clock for data realignment. Either a device pin or the logic array asserts the internal `rx_channel_data_align` node to activate the DPA data-realignment circuitry. Switching this node from low to high activates the realignment circuitry and the data being transferred to the logic array is shifted by one bit. The data realignment block cannot be bypassed. However, if the `rx_channel_data_align` is not turned on (through the `altvlds` MegaWizard Plug-In Manager), or when it is not toggled, it only acts as a register latency.

A state machine and additional logic can monitor the incoming parallel data and compare it against a known pattern. If the incoming data pattern does not match the known pattern, you can activate the `rx_channel_data_align` node again. Repeat this process until the realigner detects the desired match between the known data pattern and incoming parallel data pattern.

The DPA data-realignment circuitry allows further realignment beyond what the *J* multiplication factor allows. You can set the *J* multiplication factor to be 8 or 10. However, because data must be continuously clocked in on each low-speed clock cycle, the upcoming bit to be realigned and previous *n* –1 bits of data are selected each time the data realignment logic's counter passes *n* –1. At this point the data is selected entirely from bit-slip register 3 (see Figure 3–11) as the counter is reset to 0. The logic array receives a new valid byte of data on the next divided low speed clock cycle. Figure 3–11 shows the data realignment logic output selection from data in the data realignment register 2 and data realignment register 3 based on its current counter value upon continuous request of data slipping from the logic array.

Figure 3–11. DPA Data Realigner



Use the `rx_channel_data_align` signal within the device to activate the data realigner. You can use internal logic or an external pin to control the `rx_channel_data_align` signal. To ensure the rising edge of the `rx_channel_data_align` signal is latched into the control logic, the `rx_channel_data_align` signal should stay high for at least two low-frequency clock cycles.

To manage the alignment procedure, a state machine should be built in the FPGA logic array to generate the realignment signal. The following guidelines outline the requirements for this state machine.

- The design must include an input synchronizing register to ensure that data is synchronized to the ×*W*/*J* clock.
- After the state machine, use another synchronizing register to capture the generated `rx_channel_data_align` signal and synchronize it to the ×*W*/*J* clock.
- Because the skew in the path from the output of this synchronizing register to the PLL is undefined, the state machine must generate a pulse that is high for two *W*/*J* clock periods.
- To guarantee the state machine does not incorrectly generate multiple `rx_channel_data_align` pulses to shift a single bit, the state machine must hold the `rx_channel_data_align` signal low for at least three ×1 clock periods between pulses.

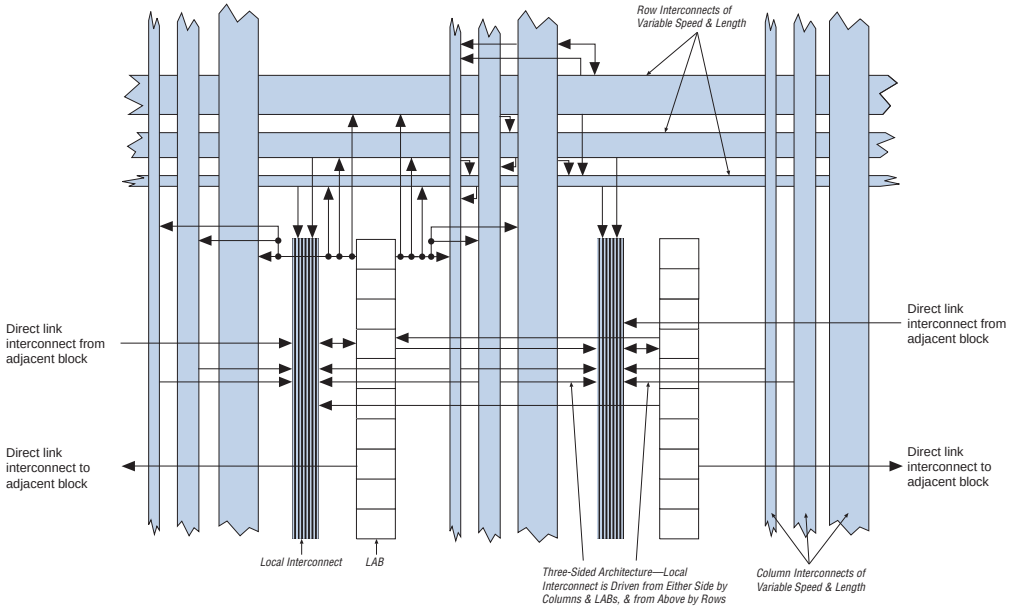
4. Stratix GX Architecture

SGX51004-1.0

Logic Array Blocks

Each LAB consists of 10 LEs, LE carry chains, LAB control signals, local interconnect, LUT chain, and register chain connection lines. The local interconnect transfers signals between LEs in the same LAB. LUT chain connections transfer the output of one LE's LUT to the adjacent LE for fast sequential LUT connections within the same LAB. Register chain connections transfer the output of one LE's register to the adjacent LE's register within an LAB. The Quartus® II Compiler places associated logic within an LAB or adjacent LABs, allowing the use of local, LUT chain, and register chain connections for performance and area efficiency. Figure 4–1 shows the Stratix® GX LAB.

Figure 4–1. Stratix GX LAB Structure

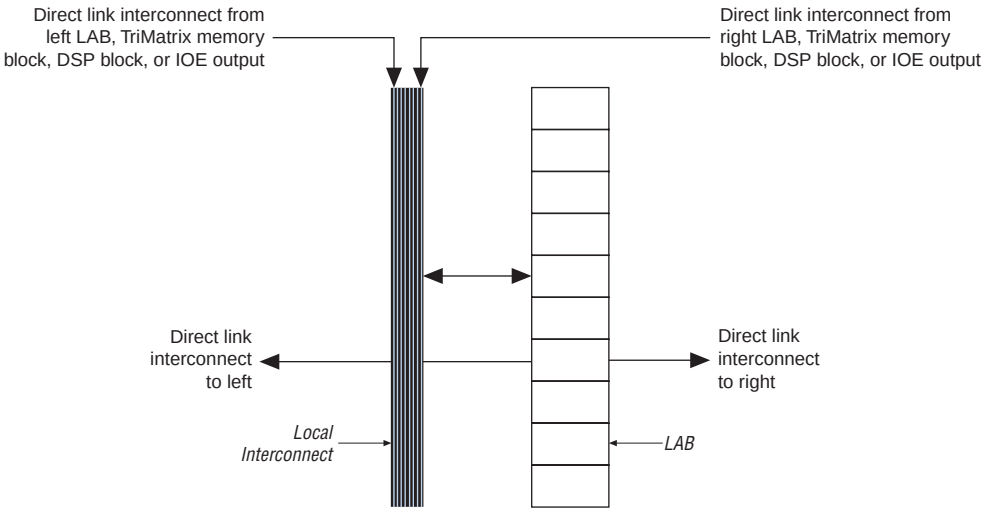


LAB Interconnects

The LAB local interconnect can drive LEs within the same LAB. The LAB local interconnect is driven by column and row interconnects and LE outputs within the same LAB. Neighboring LABs, M512 RAM blocks,

M4K RAM blocks, or DSP blocks from the left and right can also drive an LAB's local interconnect through the direct link connection. The direct link connection feature minimizes the use of row and column interconnects, providing higher performance and flexibility. Each LE can drive 30 other LEs through fast local and direct link interconnects. Figure 4–2 shows the direct link connection.

Figure 4–2. Direct Link Connection



LAB Control Signals

Each LAB contains dedicated logic for driving control signals to its LEs. The control signals include two clocks, two clock enables, two asynchronous clears, synchronous clear, asynchronous preset/load, synchronous load, and add/subtract control signals. This gives a maximum of 10 control signals at a time. Although synchronous load and clear signals are generally used when implementing counters, they can also be used with other functions.

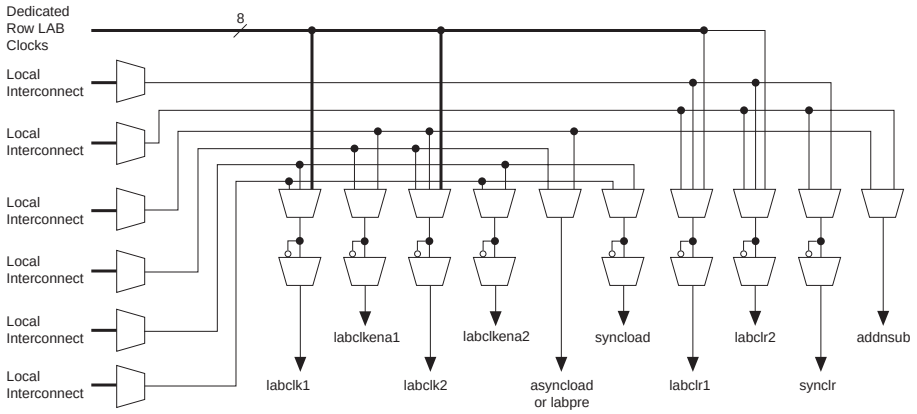
Each LAB can use two clocks and two clock enable signals. Each LAB's clock and clock enable signals are linked. For example, any LE in a particular LAB using the `labclk1` signal also uses `labclkena1`. If the LAB uses both the rising and falling edges of a clock, it also uses both LAB-wide clock signals. De-asserting the clock enable signal turns off the LAB-wide clock.

Each LAB can use two asynchronous clear signals and an asynchronous load/preset signal. The asynchronous load acts as a preset when the asynchronous load data input is tied high.

With the LAB-wide `addnsub` control signal, a single LE can implement a one-bit adder and subtractor. This saves LE resources and improves performance for logic functions such as DSP correlators and signed multipliers that alternate between addition and subtraction depending on data.

The LAB row clocks [7..0] and LAB local interconnect generate the LAB-wide control signals. The MultiTrack™ interconnect's inherent low skew allows clock and control signal distribution in addition to data. Figure 4–3 shows the LAB control signal generation circuit.

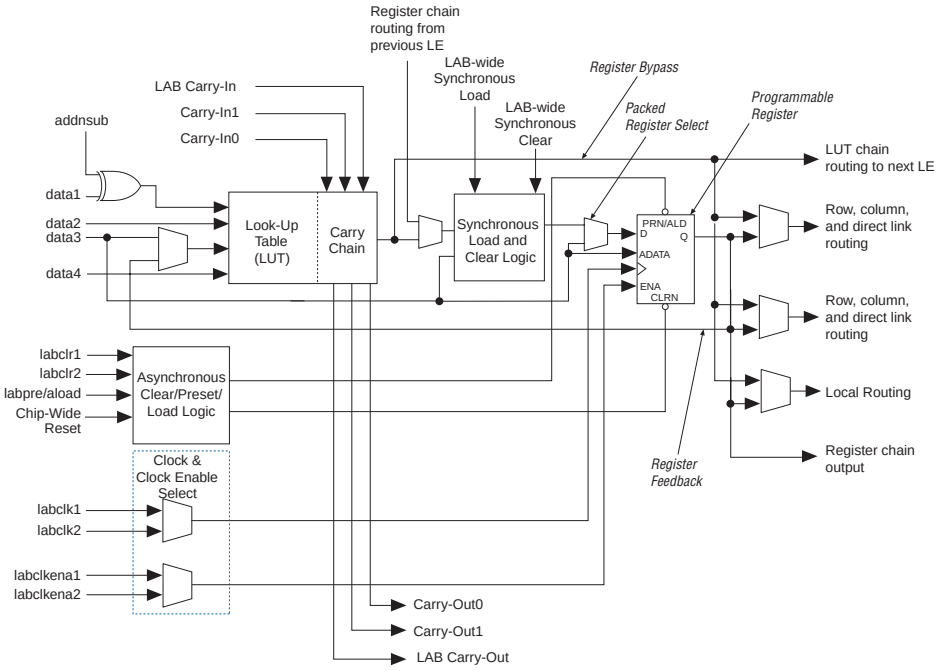
Figure 4–3. LAB-Wide Control Signals



Logic Elements

The smallest unit of logic in the Stratix GX architecture, the LE, is compact and provides advanced features with efficient logic utilization. Each LE contains a four-input LUT, which is a function generator that can implement any function of four variables. In addition, each LE contains a programmable register and carry chain with carry select capability. A single LE also supports dynamic single bit addition or subtraction mode selectable by an LAB-wide control signal. Each LE drives all types of interconnects: local, row, column, LUT chain, register chain, and direct link interconnects. See Figure 4–4.

Figure 4–4. Stratix GX LE



Each LE's programmable register can be configured for D, T, JK, or SR operation. Each register has data, true asynchronous load data, clock, clock enable, clear, and asynchronous load/preset inputs. Global signals, general-purpose I/O pins, or any internal logic can drive the register's clock and clear control signals. Either general-purpose I/O pins or internal logic can drive the clock enable, preset, asynchronous load, and asynchronous data. The asynchronous load data input comes from the `data3` input of the LE. For combinatorial functions, the register is bypassed and the output of the LUT drives directly to the outputs of the LE.

Each LE has three outputs that drive the local, row, and column routing resources. The LUT or register output can drive these three outputs independently. Two LE outputs drive column or row and direct link routing connections and one drives local interconnect resources. This allows the LUT to drive one output while the register drives another output. This feature, called register packing, improves device utilization because the device can use the register and the LUT for unrelated functions. Another special packing mode allows the register output to feed back into the LUT of the same LE so that the register is packed with

its own fan-out LUT. This provides another mechanism for improved fitting. The LE can also drive out registered and unregistered versions of the LUT output.

LUT Chain & Register Chain

In addition to the three general routing outputs, the LEs within an LAB have LUT chain and register chain outputs. LUT chain connections allow LUTs within the same LAB to cascade together for wide input functions. Register chain outputs allow registers within the same LAB to cascade together. The register chain output allows an LAB to use LUTs for a single combinatorial function and the registers to be used for an unrelated shift register implementation. These resources speed up connections between LABs while saving local interconnect resources. See "MultiTrack Interconnect" on page 4–11 for more information on LUT chain and register chain connections.

addnsub Signal

The LE's dynamic adder/subtractor feature saves logic resources by using one set of LEs to implement both an adder and a subtractor. This feature is controlled by the LAB-wide control signal `addnsub`. The `addnsub` signal sets the LAB to perform either A + B or A – B. The LUT computes addition, and subtraction is computed by adding the two's complement of the intended subtractor. The LAB-wide signal converts to two's complement by inverting the B bits within the LAB and setting carry-in = 1 to add one to the least significant bit (LSB). The LSB of an adder/subtractor must be placed in the first LE of the LAB, where the LAB-wide `addnsub` signal automatically sets the carry-in to 1. The Quartus II Compiler automatically places and uses the adder/subtractor feature when using adder/subtractor parameterized functions.

LE Operating Modes

The Stratix GX LE can operate in one of the following modes:

- Normal mode
- Dynamic arithmetic mode

Each mode uses LE resources differently. In each mode, eight available inputs to the LE—the four data inputs from the LAB local interconnect; `carry-in0` and `carry-in1` from the previous LE; the LAB carry-in from the previous carry-chain LAB; and the register chain connection—are directed to different destinations to implement the desired logic function. LAB-wide signals provide clock, asynchronous clear, asynchronous preset load, synchronous clear, synchronous load, and

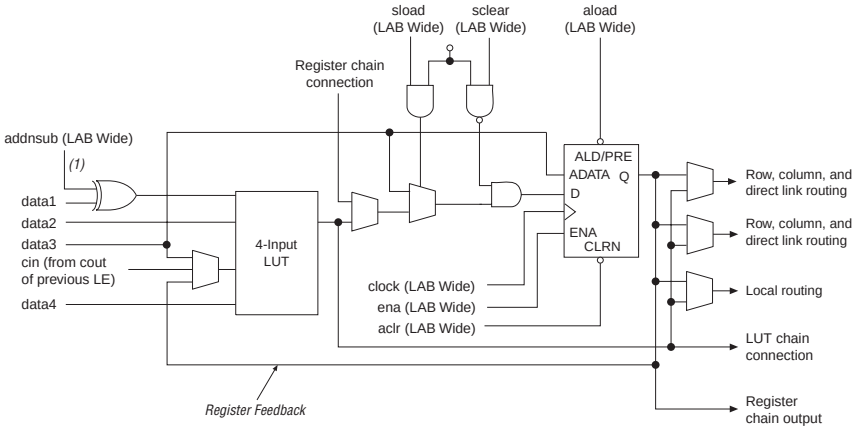
clock enable control for the register. These LAB-wide signals are available in all LE modes. The addnsub control signal is allowed in arithmetic mode.

The Quartus II software, in conjunction with parameterized functions such as library of parameterized modules (LPM) functions, automatically chooses the appropriate mode for common functions such as counters, adders, subtractors, and arithmetic functions. If required, you can also create special-purpose functions that specify which LE operating mode to use for optimal performance.

Normal Mode

The normal mode is suitable for general logic applications and combinatorial functions. In normal mode, four data inputs from the LAB local interconnect are inputs to a four-input LUT (see Figure 4–5). The Quartus II Compiler automatically selects the carry-in or the data3 signal as one of the inputs to the LUT. Each LE can use LUT chain connections to drive its combinatorial output directly to the next LE in the LAB. Asynchronous load data for the register comes from the data3 input of the LE. LEs in normal mode support packed registers.

Figure 4–5. LE in Normal Mode



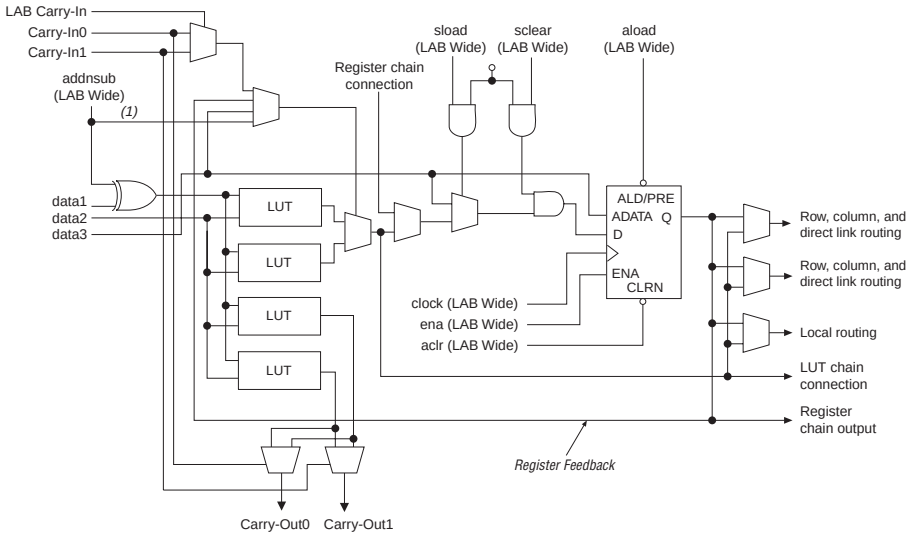
Note to Figure 4–5:
(1) This signal is only allowed in normal mode if the LE is at the end of an adder/subtractor chain.

Dynamic Arithmetic Mode

The dynamic arithmetic mode is ideal for implementing adders, counters, accumulators, wide parity functions, and comparators. An LE in dynamic arithmetic mode uses four 2-input LUTs configurable as a dynamic adder/subtractor. The first two 2-input LUTs compute two summations based on a possible carry-in of 1 or 0; the other two LUTs generate carry outputs for the two chains of the carry select circuitry. As shown in Figure 4–6, the LAB carry-in signal selects either the `carry-in0` or `carry-in1` chain. The selected chain's logic level in turn determines which parallel sum is generated as a combinatorial or registered output. For example, when implementing an adder, the sum output is the selection of two possible calculated sums: `data1` + `data2` + `carry-in0` or `data1` + `data2` + `carry-in1`. The other two LUTs use the `data1` and `data2` signals to generate two possible carry-out signals—one for a carry of 1 and the other for a carry of 0. The `carry-in0` signal acts as the carry select for the `carry-out0` output and `carry-in1` acts as the carry select for the `carry-out1` output. LEs in arithmetic mode can drive out registered and unregistered versions of the LUT output.

The dynamic arithmetic mode also offers clock enable, counter enable, synchronous up/down control, synchronous clear, synchronous load, and dynamic adder/subtractor options. The LAB local interconnect data inputs generate the counter enable and synchronous up/down control signals. The synchronous clear and synchronous load options are LAB-wide signals that affect all registers in the LAB. The Quartus II software automatically places any registers that are not used by the counter into other LABs. The `addnsub` LAB-wide signal controls whether the LE acts as an adder or subtractor.

Figure 4–6. LE in Dynamic Arithmetic Mode



Note to Figure 4–6:
(1) The addnsub signal is tied to the carry input for the first LE of a carry chain only.

Carry-Select Chain

The carry-select chain provides a very fast carry-select function between LEs in arithmetic mode. The carry-select chain uses the redundant carry calculation to increase the speed of carry functions. The LE is configured to calculate outputs for a possible carry-in of 1 and carry-in of 0 in parallel. The carry-in0 and carry-in1 signals from a lower-order bit feed forward into the higher-order bit via the parallel carry chain and feed into both the LUT and the next portion of the carry chain. Carry-select chains can begin in any LE within an LAB.

The speed advantage of the carry-select chain is in the parallel pre-computation of carry chains. Because the LAB carry-in selects the precomputed carry chain, not every LE is in the critical path. Only the propagation delay between LAB carry-in generation (LE 5 and LE 10) are now part of the critical path. This feature allows the Stratix GX architecture to implement high-speed counters, adders, multipliers, parity functions, and comparators of arbitrary width.

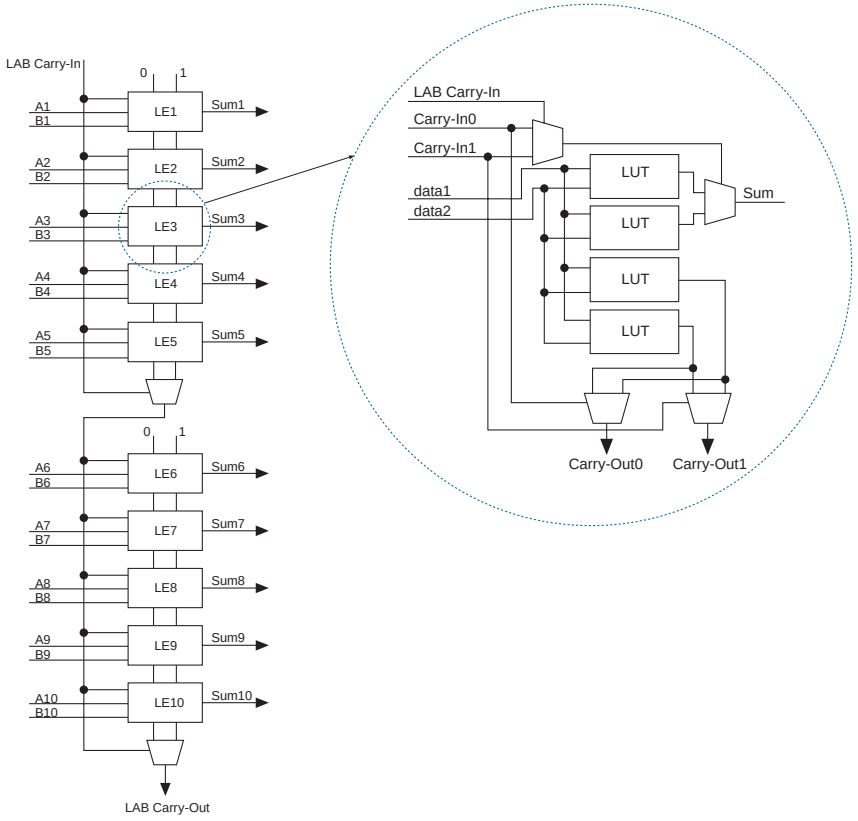
Figure 4–7 shows the carry-select circuitry in an LAB for a 10-bit full adder. One portion of the LUT generates the sum of two bits using the input signals and the appropriate carry-in bit; the sum is routed to the output of the LE. The register can be bypassed for simple adders or used

for accumulator functions. Another portion of the LUT generates carry-out bits. An LAB-wide carry in bit selects which chain to use for the addition of given inputs. The carry-in signal for each chain, `carry-in0` or `carry-in1`, selects the carry-out to carry forward to the carry-in signal of the next-higher-order bit. The final carry-out signal is routed to an LE, where it is fed to local, row, or column interconnects.

The Quartus II Compiler automatically creates carry chain logic during design processing, or you can create it manually during design entry. Parameterized functions such as LPM functions automatically take advantage of carry chains for the appropriate functions.

The Quartus II Compiler creates carry chains longer than 10 LEs by linking LABs together automatically. For enhanced fitting, a long carry chain runs vertically allowing fast horizontal connections to TriMatrix™ memory and DSP blocks. A carry chain can continue as far as a full column.

Figure 4–7. Carry Select Chain



Clear & Preset Logic Control

LAB-wide signals control the logic for the register's clear and preset signals. The LE directly supports an asynchronous clear and preset function. The register preset is achieved through the asynchronous load of a logic high. The direct asynchronous preset does not require a NOT-gate push-back technique. Stratix GX devices support simultaneous preset/ asynchronous load, and clear signals. An asynchronous clear signal takes precedence if both signals are asserted simultaneously. Each LAB supports up to two clears and one preset signal.

In addition to the clear and preset ports, Stratix GX devices provide a chip-wide reset pin (DEV_CLRn) that resets all registers in the device. An option set before compilation in the Quartus II software controls this pin. This chip-wide reset overrides all other control signals.

MultiTrack Interconnect

In the Stratix GX architecture, connections between LEs, TriMatrix memory, DSP blocks, and device I/O pins are provided by the MultiTrack interconnect structure with DirectDrive™ technology. The MultiTrack interconnect consists of continuous, performance-optimized routing lines of different lengths and speeds used for inter- and intra-design block connectivity. The Quartus II Compiler automatically places critical design paths on faster interconnects to improve design performance.

DirectDrive technology is a deterministic routing technology that ensures identical routing resource usage for any function regardless of placement within the device. The MultiTrack interconnect and DirectDrive technology simplify the integration stage of block-based designing by eliminating the re-optimization cycles that typically follow design changes and additions.

The MultiTrack interconnect consists of row and column interconnects that span fixed distances. A routing structure with fixed length resources for all devices allows predictable and repeatable performance when migrating through different device densities. Dedicated row interconnects route signals to and from LABs, DSP blocks, and TriMatrix memory within the same row. These row resources include:

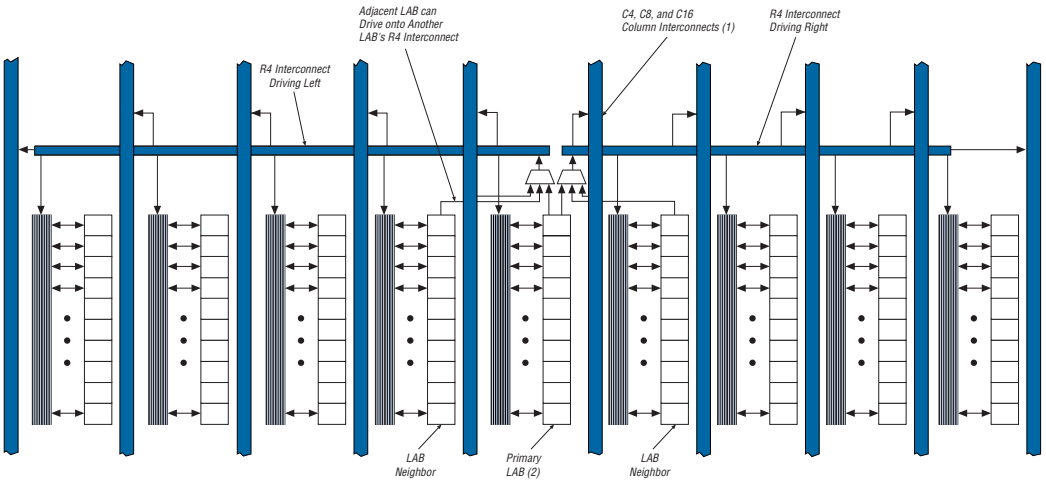
- Direct link interconnects between LABs and adjacent blocks.
- R4 interconnects traversing four blocks to the right or left.
- R8 interconnects traversing eight blocks to the right or left.
- R24 row interconnects for high-speed access across the length of the device.

The direct link interconnect allows an LAB, DSP block, or TriMatrix memory block to drive into the local interconnect of its left and right neighbors and then back into itself. Only one side of a M-RAM block interfaces with direct link and row interconnects. This provides fast communication between adjacent LABs and/or blocks without using row interconnect resources.

The R4 interconnects span four LABs, three LABs and one M512 RAM block, two LABs and one M4K RAM block, or two LABs and one DSP block to the right or left of a source LAB. These resources are used for fast row connections in a four-LAB region. Every LAB has its own set of R4 interconnects to drive either left or right. Figure 4–8 shows R4 interconnect connections from an LAB. R4 interconnects can drive and be driven by DSP blocks and RAM blocks and horizontal IOEs. For LAB interfacing, a primary LAB or LAB neighbor can drive a given R4 interconnect. For R4 interconnects that drive to the right, the primary LAB and right neighbor can drive on to the interconnect. For R4 interconnects that drive to the left, the primary LAB and its left neighbor

can drive on to the interconnect. R4 interconnects can drive other R4 interconnects to extend the range of LABs they can drive. R4 interconnects can also drive C4 and C16 interconnects for connections from one row to another. Additionally, R4 interconnects can drive R24 interconnects.

Figure 4–8. R4 Interconnect Connections



Notes to Figure 4–8:
(1) C4 interconnects can drive R4 interconnects.
(2) This pattern is repeated for every LAB in the LAB row.

The R8 interconnects span eight LABs, M512 or M4K RAM blocks, or DSP blocks to the right or left from a source LAB. These resources are used for fast row connections in an eight-LAB region. Every LAB has its own set of R8 interconnects to drive either left or right. R8 interconnect connections between LABs in a row are similar to the R4 connections shown in Figure 4–8, with the exception that they connect to eight LABs to the right or left, not four. Like R4 interconnects, R8 interconnects can drive and be driven by all types of architecture blocks. R8 interconnects can drive other R8 interconnects to extend their range as well as C8 interconnects for row-to-row connections. One R8 interconnect is faster than two R4 interconnects connected together.

R24 row interconnects span 24 LABs and provide the fastest resource for long row connections between LABs, TriMatrix memory, DSP blocks, and IOEs. The R24 row interconnects can cross M-RAM blocks. R24 row interconnects drive to other row or column interconnects at every fourth

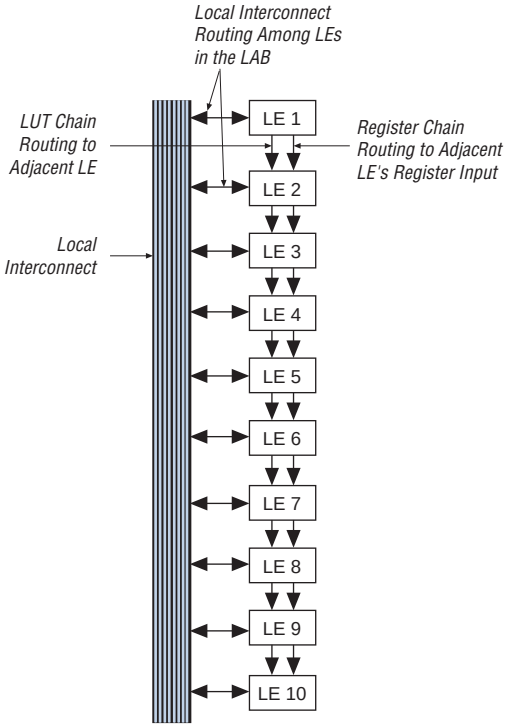
LAB and do not drive directly to LAB local interconnects. R24 row interconnects drive LAB local interconnects via R4 and C4 interconnects. R24 interconnects can drive R24, R4, C16, and C4 interconnects.

The column interconnect operates similarly to the row interconnect and vertically routes signals to and from LABs, TriMatrix memory, DSP blocks, and IOEs. Each column of LABs is served by a dedicated column interconnect, which vertically routes signals to and from LABs, TriMatrix memory and DSP blocks, and horizontal IOEs. These column resources include:

- LUT chain interconnects within an LAB
- Register chain interconnects within an LAB
- C4 interconnects traversing a distance of four blocks in up and down direction
- C8 interconnects traversing a distance of eight blocks in up and down direction
- C16 column interconnects for high-speed vertical routing through the device

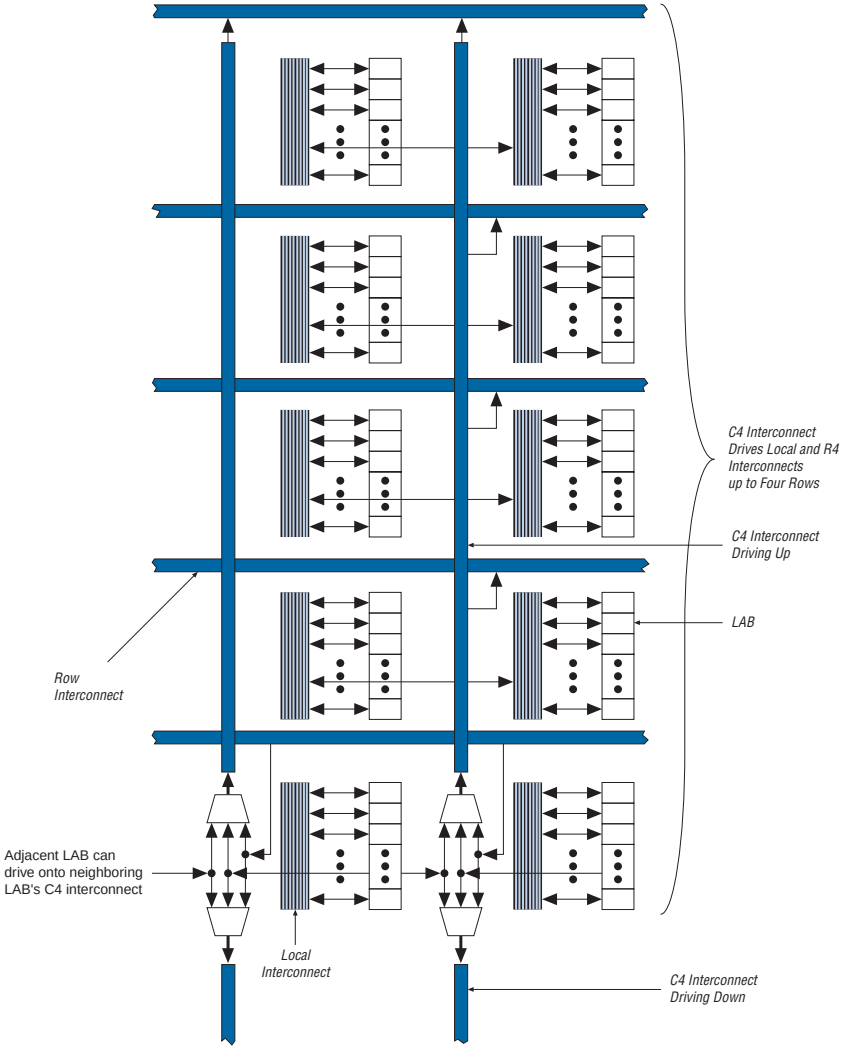
Stratix GX devices include an enhanced interconnect structure within LABs for routing LE output to LE input connections faster using LUT chain connections and register chain connections. The LUT chain connection allows the combinatorial output of an LE to directly drive the fast input of the LE right below it, bypassing the local interconnect. These resources can be used as a high-speed connection for wide fan-in functions from LE 1 to LE 10 in the same LAB. The register chain connection allows the register output of one LE to connect directly to the register input of the next LE in the LAB for fast shift registers. The Quartus II Compiler automatically takes advantage of these resources to improve utilization and performance. Figure 4–9 shows the LUT chain and register chain interconnects.

Figure 4–9. LUT Chain & Register Chain Interconnects



The C4 interconnects span four LABs, M512, or M4K blocks up or down from a source LAB. Every LAB has its own set of C4 interconnects to drive either up or down. Figure 4–10 shows the C4 interconnect connections from an LAB in a column. The C4 interconnects can drive and be driven by all types of architecture blocks, including DSP blocks, TriMatrix memory blocks, and vertical IOEs. For LAB interconnection, a primary LAB or its LAB neighbor can drive a given C4 interconnect.
C4 interconnects can drive each other to extend their range as well as drive row interconnects for column-to-column connections.

Figure 4–10. C4 Interconnect Connections *Note (1)*



Note to Figure 4–10:
(1) Each C4 interconnect can drive either up or down four rows.

C8 interconnects span eight LABs, M512, or M4K blocks up or down from a source LAB. Every LAB has its own set of C8 interconnects to drive either up or down. C8 interconnect connections between the LABs in a column are similar to the C4 connections shown in Figure 4–10 with the exception that they connect to eight LABs above and below. The C8

interconnects can drive and be driven by all types of architecture blocks similar to C4 interconnects. C8 interconnects can drive each other to extend their range as well as R8 interconnects for column-to-column connections. C8 interconnects are faster than two C4 interconnects.

C16 column interconnects span a length of 16 LABs and provide the fastest resource for long column connections between LABs, TriMatrix memory blocks, DSP blocks, and IOEs. C16 interconnects can cross M-RAM blocks and also drive to row and column interconnects at every fourth LAB. C16 interconnects drive LAB local interconnects via C4 and R4 interconnects and do not drive LAB local interconnects directly.

All embedded blocks communicate with the logic array similar to LAB-to-LAB interfaces. Each block (that is, TriMatrix memory and DSP blocks) connects to row and column interconnects and has local interconnect regions driven by row and column interconnects. These blocks also have direct link interconnects for fast connections to and from a neighboring LAB. All blocks are fed by the row LAB clocks, `labclk[7..0]`.

Table 4–1 shows the Stratix GX device's routing scheme.

Table 4–1. Stratix GX Device Routing Scheme

Source	Destination																
	LUT Chain	Register Chain	Local Interconnect	Direct Link Interconnect	R4 Interconnect	R8 Interconnect	R24 Interconnect	C4 Interconnect	C8 Interconnect	C16 Interconnect	LE	M512 RAM Block	M4K RAM Block	M-RAM Block	DSP Blocks	Column IOE	Row IOE
LUT Chain											✓						
Register Chain											✓						
Local Interconnect											✓	✓	✓	✓	✓	✓	✓
Direct Link Interconnect			✓														
R4 Interconnect			✓		✓		✓	✓		✓							
R8 Interconnect			✓			✓			✓								
R24 Interconnect					✓		✓	✓		✓							
C4 Interconnect			✓		✓			✓									
C8 Interconnect			✓			✓			✓								
C16 Interconnect					✓		✓	✓		✓							
LE	✓	✓	✓	✓	✓	✓		✓	✓								
M512 RAM Block			✓	✓	✓	✓		✓	✓								
M4K RAM Block			✓	✓	✓	✓		✓	✓								
M-RAM Block								✓	✓								
DSP Blocks			✓	✓	✓	✓		✓	✓								
Column IOE				✓				✓	✓	✓							
Row IOE				✓		✓	✓	✓	✓	✓							

TriMatrix Memory

TriMatrix memory consists of three types of RAM blocks: M512, M4K, and M-RAM blocks. Although these memory blocks are different, they can all implement various types of memory with or without parity, including true dual-port, simple dual-port, and single-port RAM, ROM, and FIFO buffers. Table 4–2 shows the size and features of the different RAM blocks.

Table 4–2. TriMatrix Memory Features (Part 1 of 2)

Memory Feature	M512 RAM Block (32 × 18 Bits)	M4K RAM Block (128 × 36 Bits)	M-RAM Block (4K × 144 Bits)
Maximum performance	*(1)*	*(1)*	*(1)*
True dual-port memory		✓	✓
Simple dual-port memory	✓	✓	✓
Single-port memory	✓	✓	✓
Shift register	✓	✓	
ROM	✓	✓	*(2)*
FIFO buffer	✓	✓	✓
Byte enable		✓	✓
Parity bits	✓	✓	✓
Mixed clock mode	✓	✓	✓
Memory initialization	✓	✓	
Simple dual-port memory mixed width support	✓	✓	✓
True dual-port memory mixed width support		✓	✓
Power-up conditions	Outputs cleared	Outputs cleared	Outputs unknown
Register clears	Input and output registers	Input and output registers	Output registers
Mixed-port read-during-write	Unknown output/old data	Unknown output/old data	Unknown output

Table 4–2. TriMatrix Memory Features (Part 2 of 2)

Memory Feature	M512 RAM Block (32 × 18 Bits)	M4K RAM Block (128 × 36 Bits)	M-RAM Block (4K × 144 Bits)
Configurations	512 ×1 256 × 2 128 × 4 64 × 8 64 × 9 32 × 16 32 × 18	4K × 1 2K × 2 1K × 4 512 × 8 512 × 9 256 × 16 256 × 18 128 × 32 128 × 36	64K × 8 64K × 9 32K × 16 32K × 18 16K × 32 16K × 36 8K × 64 8K × 72 4K × 128 4K × 144

Notes to Table 4–2:

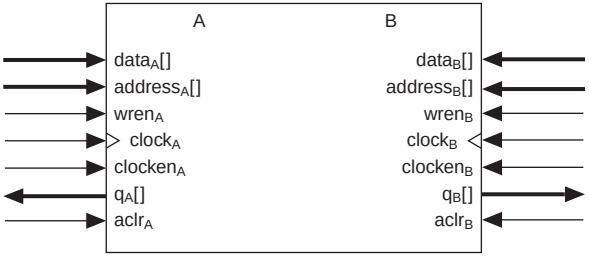
(1) See the *DC & Switching Characteristics* chapter of the *Stratix GX Device Handbook, Volume 1* for maximum performance information.

(2) The M-RAM block does not support memory initializations. However, the M-RAM block can emulate a ROM function using a dual-port RAM bock. The Stratix GX device must write to the dual-port memory once and then disable the write-enable ports afterwards.

Memory Modes

TriMatrix memory blocks include input registers that synchronize writes and output registers to pipeline designs and improve system performance. M4K and M-RAM memory blocks offer a true dual-port mode to support any combination of two-port operations: two reads, two writes, or one read and one write at two different clock frequencies. Figure 4–11 shows true dual-port memory.

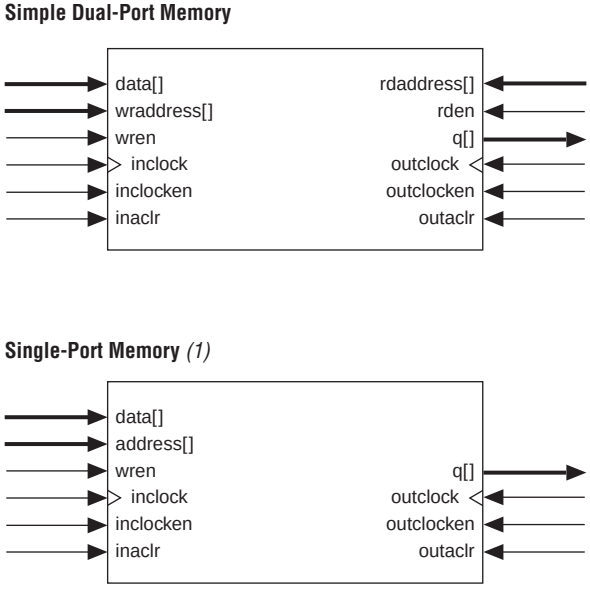
Figure 4–11. True Dual-Port Memory Configuration



In addition to true dual-port memory, the memory blocks support simple dual-port and single-port RAM. Simple dual-port memory supports a simultaneous read and write and can either read old data before the write

occurs or just read the don't care bits. Single-port memory supports non-simultaneous reads and writes, but the q[] port outputs the data once it has been written to the memory (if the outputs are not registered) or after the next rising edge of the clock (if the outputs are registered). For more information, see the *TriMatrix Embedded Memory Blocks in Stratix & Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2*. Figure 4–12 shows these different RAM memory port configurations for TriMatrix memory.

Figure 4–12. Simple Dual-Port & Single-Port Memory Configurations



Note to Figure 4–12:

(1) Two single-port memory blocks can be implemented in a single M4K block as long as each of the two independent block sizes is equal to or less than half of the M4K block size.

The memory blocks also enable mixed-width data ports for reading and writing to the RAM ports in dual-port RAM configuration. For example, the memory block can be written in ×1 mode at port A and read out in ×16 mode from port B.

TriMatrix memory architecture can implement pipelined RAM by registering both the input and output signals to the RAM block. All TriMatrix memory block inputs are registered providing synchronous write cycles. In synchronous operation, the memory block generates its own self-timed strobe write enable (WREN) signal derived from the global

or regional clock. In contrast, a circuit using asynchronous RAM must generate the RAM WREN signal while ensuring its data and address signals meet setup and hold time specifications relative to the WREN signal. The output registers can be bypassed. Flow-through reading is possible in the simple dual-port mode of M512 and M4K RAM blocks by clocking the read enable and read address registers on the negative clock edge and bypassing the output registers.

Two single-port memory blocks can be implemented in a single M4K block as long as each of the two independent block sizes is equal to or less than half of the M4K block size.

The Quartus II software automatically implements larger memory by combining multiple TriMatrix memory blocks. For example, two 256 × 16-bit RAM blocks can be combined to form a 256 × 32-bit RAM block. Memory performance does not degrade for memory blocks using the maximum number of words available in one memory block. Logical memory blocks using less than the maximum number of words use physical blocks in parallel, eliminating any external control logic that would increase delays. To create a larger high-speed memory block, the Quartus II software automatically combines memory blocks with LE control logic.

Parity Bit Support

The memory blocks support a parity bit for each byte. The parity bit, along with internal LE logic, can implement parity checking for error detection to ensure data integrity. You can also use parity-size data words to store user-specified control bits. In the M4K and M-RAM blocks, byte enables are also available for data input masking during write operations.

Shift Register Support

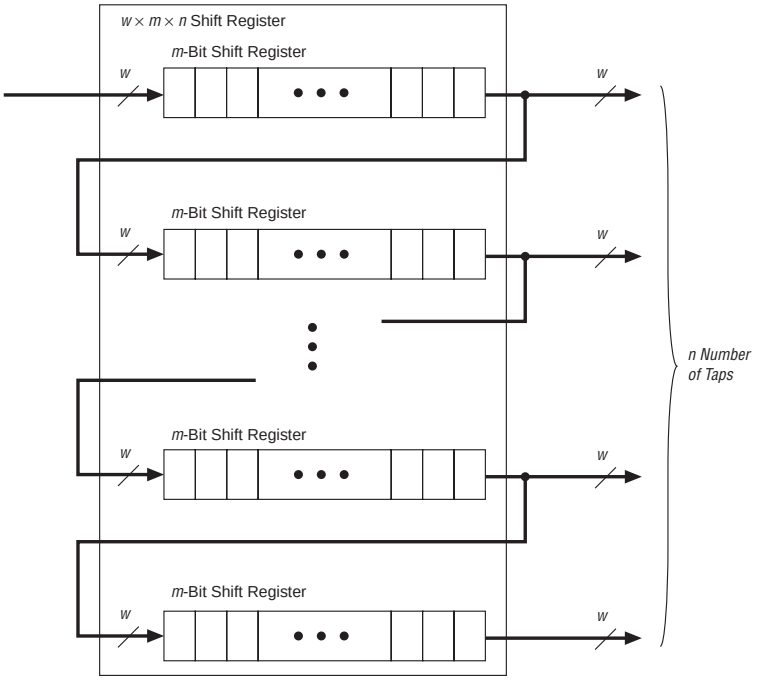
You can configure embedded memory blocks to implement shift registers for DSP applications such as pseudo-random number generators, multi-channel filtering, auto-correlation, and cross-correlation functions. These and other DSP applications require local data storage, traditionally implemented with standard flip-flops, which can quickly consume many logic cells and routing resources for large shift registers. A more efficient alternative is to use embedded memory as a shift register block, which saves logic cell and routing resources and provides a more efficient implementation with the dedicated circuitry.

The size of a $w \times m \times n$ shift register is determined by the input data width (w), the length of the taps (m), and the number of taps (n). The size of a $w \times m \times n$ shift register must be less than or equal to the maximum number of memory bits in the respective block: 576 bits for the M512

RAM block and 4,608 bits for the M4K RAM block. The total number of shift register outputs (number of taps n × width w) must be less than the maximum data width of the RAM block (18 for M512 blocks, 36 for M4K blocks). To create larger shift registers, the memory blocks are cascaded together.

Data is written into each address location at the falling edge of the clock and read from the address at the rising edge of the clock. The shift register mode logic automatically controls the positive and negative edge clocking to shift the data in one clock cycle. Figure 4–13 shows the TriMatrix memory block in the shift register mode.

Figure 4–13. Shift Register Memory Configuration



Memory Block Size

TriMatrix memory provides three different memory sizes for efficient application support. The large number of M512 blocks are ideal for designs with many shallow first-in first-out (FIFO) buffers. M4K blocks provide additional resources for channelized functions that do not require large amounts of storage. The M-RAM blocks provide a large

single block of RAM ideal for data packet storage. The different-sized blocks allow Stratix GX devices to efficiently support variable-sized memory in designs.

The Quartus II software automatically partitions the user-defined memory into the embedded memory blocks using the most efficient size combinations. You can also manually assign the memory to a specific block size or a mixture of block sizes.

M512 RAM Block

The M512 RAM block is a simple dual-port memory block and is useful for implementing small FIFO buffers, DSP, and clock domain transfer applications. Each block contains 576 RAM bits (including parity bits). M512 RAM blocks can be configured in the following modes:

- Simple dual-port RAM
- Single-port RAM
- FIFO
- ROM
- Shift register

When configured as RAM or ROM, you can use an initialization file to pre-load the memory contents.

The memory address depths and output widths can be configured as 512 × 1, 256 × 2, 128 × 4, 64 × 8 (64 × 9 bits with parity), and 32 × 16 (32 × 18 bits with parity). Mixed-width configurations are also possible, allowing different read and write widths. Table 4–3 summarizes the possible M512 RAM block configurations.

Table 4–3. M512 RAM Block Configurations (Simple Dual-Port RAM)

Read Port	Write Port						
	512 × 1	256 × 2	128 × 4	64 × 8	32 × 16	64 × 9	32 × 18
512 × 1	✓	✓	✓	✓	✓		
256 × 2	✓	✓	✓	✓	✓		
128 × 4	✓	✓	✓		✓		
64 × 8	✓	✓		✓			
32 × 16	✓	✓	✓		✓		
64 × 9						✓	
32 × 18							✓

When the M512 RAM block is configured as a shift register block, a shift register of size up to 576 bits is possible.

The M512 RAM block can also be configured to support serializer and deserializer applications. By using the mixed-width support in combination with DDR I/O standards, the block can function as a SERDES to support low-speed serial I/O standards using global or regional clocks. See "I/O Structure" on page 4–96 for details on dedicated SERDES in Stratix GX devices.

M512 RAM blocks can have different clocks on its inputs and outputs. The `wren`, `datain`, and write address registers are all clocked together from one of the two clocks feeding the block. The read address, `rden`, and output registers can be clocked by either of the two clocks driving the block. This allows the RAM block to operate in read/write or input/output clock modes. Only the output register can be bypassed. The eight `labclk` signals or local interconnect can drive the `inclock`, `outclock`, `wren`, `rden`, `inclr`, and `outclr` signals. Because of the advanced interconnect between the LAB and M512 RAM blocks, LEs can also control the `wren` and `rden` signals and the RAM clock, clock enable, and asynchronous clear signals. Figure 4–14 shows the M512 RAM block control signal generation logic.

The RAM blocks within Stratix GX devices have local interconnects to allow LEs and interconnects to drive into RAM blocks. The M512 RAM block local interconnect is driven by the R4, R8, C4, C8, and direct link interconnects from adjacent LABs. The M512 RAM blocks can communicate with LABs on either the left or right side through these row interconnects or with LAB columns on the left or right side with the column interconnects. Up to 10 direct link input connections to the M512 RAM block are possible from the left adjacent LABs and another 10 possible from the right adjacent LAB. M512 RAM outputs can also connect to left and right LABs through 10 direct link interconnects. The M512 RAM block has equal opportunity for access and performance to and from LABs on either its left or right side. Figure 4–15 shows the M512 RAM block to logic array interface.

Figure 4–14. M512 RAM Block Control Signals

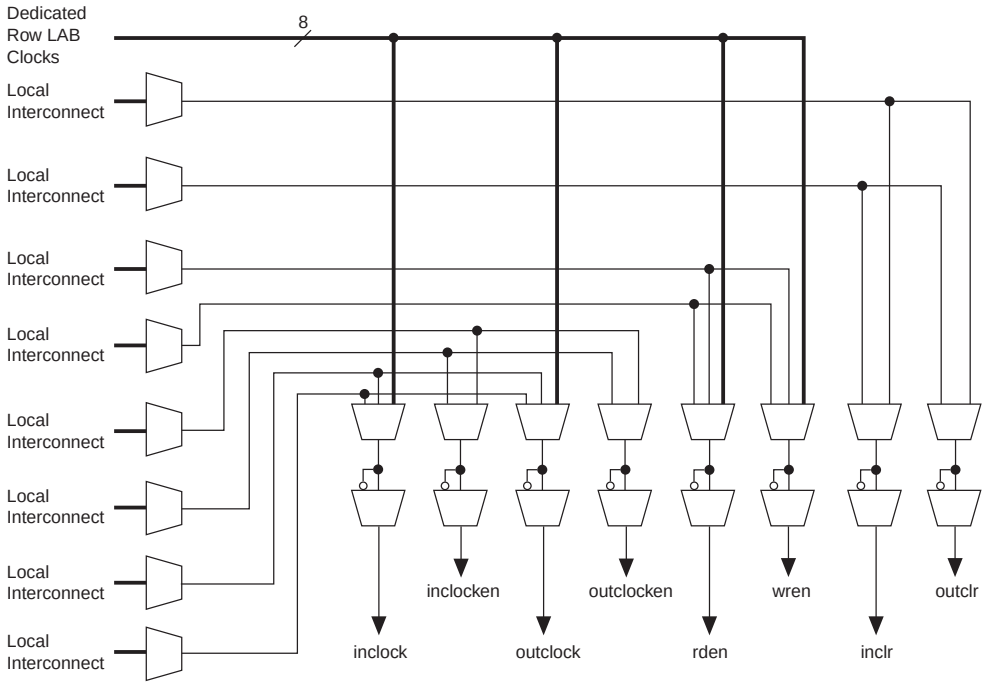
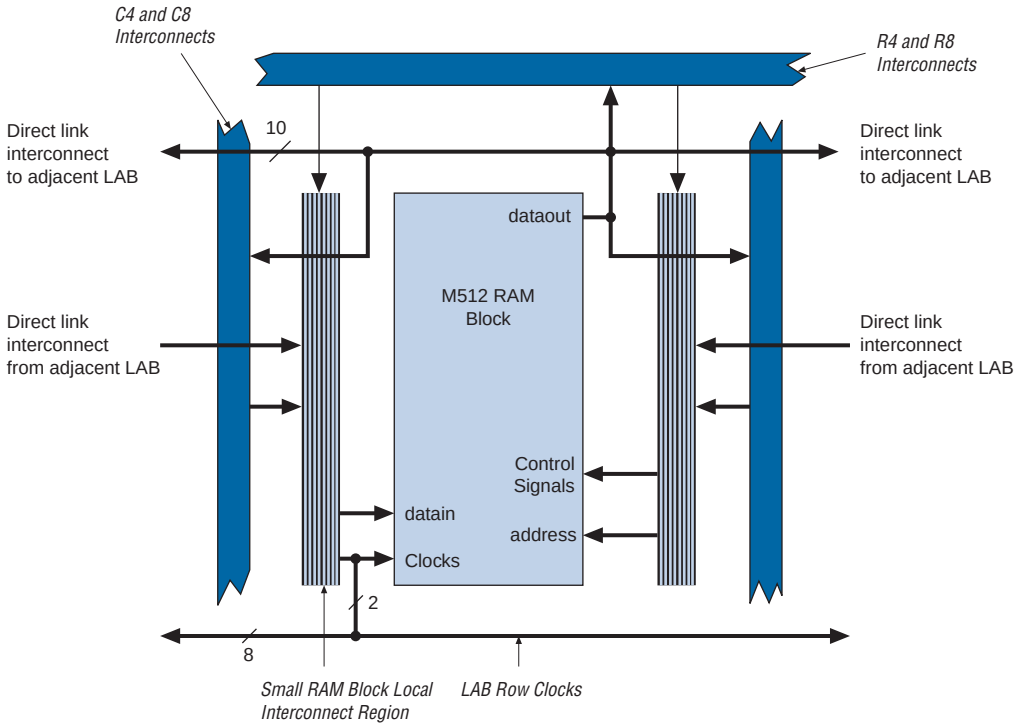


Figure 4–15. M512 RAM Block LAB Row Interface



M4K RAM Blocks

The M4K RAM block includes support for true dual-port RAM. The M4K RAM block implements buffers for a wide variety of applications such as storing processor code, implementing lookup schemes, and implementing larger memory applications. Each block contains 4,608 RAM bits (including parity bits). M4K RAM blocks can be configured in the following modes:

- True dual-port RAM
- Simple dual-port RAM
- Single-port RAM
- FIFO
- ROM
- Shift register

When configured as RAM or ROM, you can use an initialization file to pre-load the memory contents.

The memory address depths and output widths can be configured as 4,096 × 1, 2,048 × 2, 1,024 × 4, 512 × 8 (or 512 × 9 bits), 256 × 16 (or 256 × 18 bits), and 128 × 32 (or 128 × 36 bits). The 128 × 32- or 36-bit configuration is not available in the true dual-port mode. Mixed-width configurations are also possible, allowing different read and write widths. Tables 4–4 and 4–5 summarize the possible M4K RAM block configurations.

Table 4–4. M4K RAM Block Configurations (Simple Dual-Port)

Read Port	Write Port								
	4K 1	2K × 2	1K ° 4	512 ° 8	256 ° 16	128 ° 32	512 ° 9	256 ° 18	128 ° 36
4K × 1	✓	✓	✓	✓	✓	✓			
2K × 2	✓	✓	✓	✓	✓	✓			
1K × 4	✓	✓	✓	✓	✓	✓			
512 × 8	✓	✓	✓	✓	✓	✓			
256 × 16	✓	✓	✓	✓	✓	✓			
128 × 32	✓	✓	✓	✓	✓	✓			
512 × 9							✓	✓	✓
256 × 18							✓	✓	✓
128 × 36							✓	✓	✓

Table 4–5. M4K RAM Block Configurations (True Dual-Port)

Port A	Port B						
	4K × 1	2K × 2	1K × 4	512 × 8	256 × 16	512 × 9	256 × 18
4K × 1	✓	✓	✓	✓	✓		
2K × 2	✓	✓	✓	✓	✓		
1K × 4	✓	✓	✓	✓	✓		
512 × 8	✓	✓	✓	✓	✓		
256 × 16	✓	✓	✓	✓	✓		
512 × 9						✓	✓
256 × 18						✓	✓

When the M4K RAM block is configured as a shift register block, you can create a shift register up to 4,608 bits ($w \times m \times n$).

M4K RAM blocks support byte writes when the write port has a data width of 16, 18, 32, or 36 bits. The byte enables allow the input data to be masked so the device can write to specific bytes. The unwritten bytes retain the previous written value. Table 4–6 summarizes the byte selection.

Table 4–6. Byte Enable for M4K Blocks Notes (1), (2)

byteena[3..0]	datain ×18	datain ×36
[0] = 1	[8..0]	[8..0]
[1] = 1	[17..9]	[17..9]
[2] = 1	–	[26..18]
[3] = 1	–	[35..27]

Notes to Table 4–6:
(1) Any combination of byte enables is possible.
(2) Byte enables can be used in the same manner with 8-bit words, that is, in ×16 and ×32 modes.

The M4K RAM blocks allow for different clocks on their inputs and outputs. Either of the two clocks feeding the block can clock M4K RAM block registers (`renwe`, address, byte enable, `datain`, and output registers). Only the output register can be bypassed. The eight `labclk` signals or local interconnects can drive the control signals for the A and B ports of the M4K RAM block. LEs can also control the `clock_a`, `clock_b`, `renwe_a`, `renwe_b`, `clr_a`, `clr_b`, `clocken_a`, and `clocken_b` signals, as shown in Figure 4–16.

The R4, R8, C4, C8, and direct link interconnects from adjacent LABs drive the M4K RAM block local interconnect. The M4K RAM blocks can communicate with LABs on either the left or right side through these row resources or with LAB columns on either the right or left with the column resources. Up to 10 direct link input connections to the M4K RAM Block are possible from the left adjacent LABs and another 10 possible from the right adjacent LAB. M4K RAM block outputs can also connect to left and right LABs through 10 direct link interconnects each. Figure 4–17 shows the M4K RAM block to logic array interface.

Figure 4–16. M4K RAM Block Control Signals

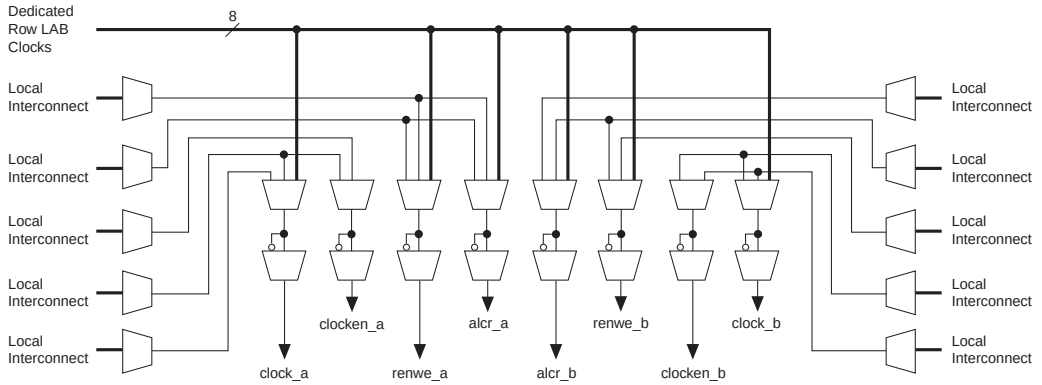
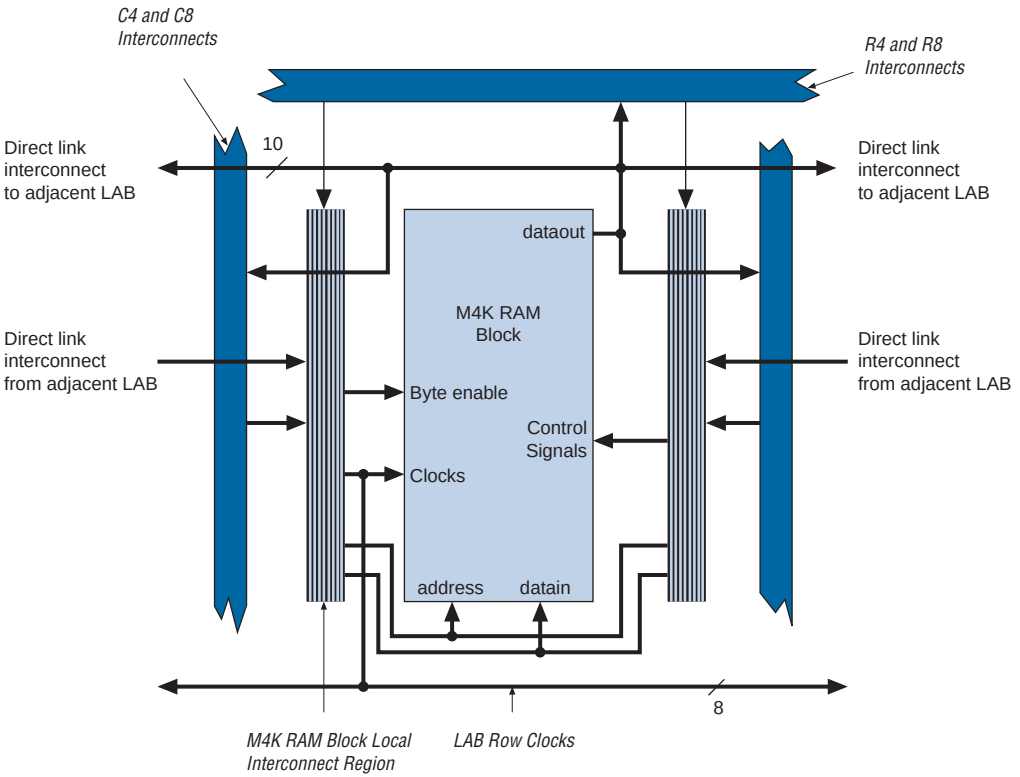


Figure 4–17. M4K RAM Block LAB Row Interface



M-RAM Block

The largest TriMatrix memory block, the M-RAM block, is useful for applications where a large volume of data must be stored on-chip. Each block contains 589,824 RAM bits (including parity bits). The M-RAM block can be configured in the following modes:

- True dual-port RAM
- Simple dual-port RAM
- Single-port RAM
- FIFO RAM

You cannot use an initialization file to initialize the contents of a M-RAM block. All M-RAM block contents power up to an undefined value. Only synchronous operation is supported in the M-RAM block, so all inputs are registered. Output registers can be bypassed. The memory address and output width can be configured as 64K × 8 (or 64K × 9 bits), 32K × 16 (or 32K × 18 bits), 16K × 32 (or 16K × 36 bits), 8K × 64 (or 8K × 72 bits), and 4K × 128 (or 4K × 144 bits). The 4K × 128 configuration is unavailable in true dual-port mode because there are a total of 144 data output drivers in the block. Mixed-width configurations are also possible, allowing different read and write widths. Tables 4–7 and 4–8 summarize the possible M-RAM block configurations:

Table 4–7. M-RAM Block Configurations (Simple Dual-Port)

Read Port	Write Port				
	64K × 9	32K × 18	16K × 36	8K × 72	4K × 144
64K × 9	✓	✓	✓	✓	
32K × 18	✓	✓	✓	✓	
16K × 36	✓	✓	✓	✓	
8K × 72	✓	✓	✓	✓	
4K × 144					✓

Table 4–8. M-RAM Block Configurations (True Dual-Port)				
Port A	Port B			
	64K × 9	32K × 18	16K × 36	8K × 72
64K × 9	✓	✓	✓	✓
32K × 18	✓	✓	✓	✓
16K × 36	✓	✓	✓	✓
8K × 72	✓	✓	✓	✓

The read and write operation of the memory is controlled by the WREN signal, which sets the ports into either read or write modes. There is no separate read enable (RE) signal.

Writing into RAM is controlled by both the WREN and byte enable (byteena) signals for each port. The default value for the byteena signal is high, in which case writing is controlled only by the WREN signal. The byte enables are available for the ×18, ×36, and ×72 modes. In the ×144 simple dual-port mode, the two sets of byteena signals (byteena_a and byteena_b) are combined to form the necessary 16 byte enables. Tables 4–9 and 4–10 summarize the byte selection.

Table 4–9. Byte Enable for M-RAM Blocks Notes (1), (2)			
byteena[3..0]	datain ×18	datain ×36	datain ×72
[0] = 1	[8..0]	[8..0]	[8..0]
[1] = 1	[17..9]	[17..9]	[17..9]
[2] = 1	–	[26..18]	[26..18]
[3] = 1	–	[35..27]	[35..27]
[4] = 1	–	–	[44..36]
[5] = 1	–	–	[53..45]
[6] = 1	–	–	[62..54]
[7] = 1	–	–	[71..63]

Table 4–10. M-RAM Combined Byte Selection for ×144 Mode Notes (1), (2)	
byteena[15..0]	**datain ×144**
[0] = 1	[8..0]
[1] = 1	[17..9]
[2] = 1	[26..18]
[3] = 1	[35..27]
[4] = 1	[44..36]
[5] = 1	[53..45]
[6] = 1	[62..54]
[7] = 1	[71..63]
[8] = 1	[80..72]
[9] = 1	[89..81]
[10] = 1	[98..90]
[11] = 1	[107..99]
[12] = 1	[116..108]
[13] = 1	[125..117]
[14] = 1	[134..126]
[15] = 1	[143..135]

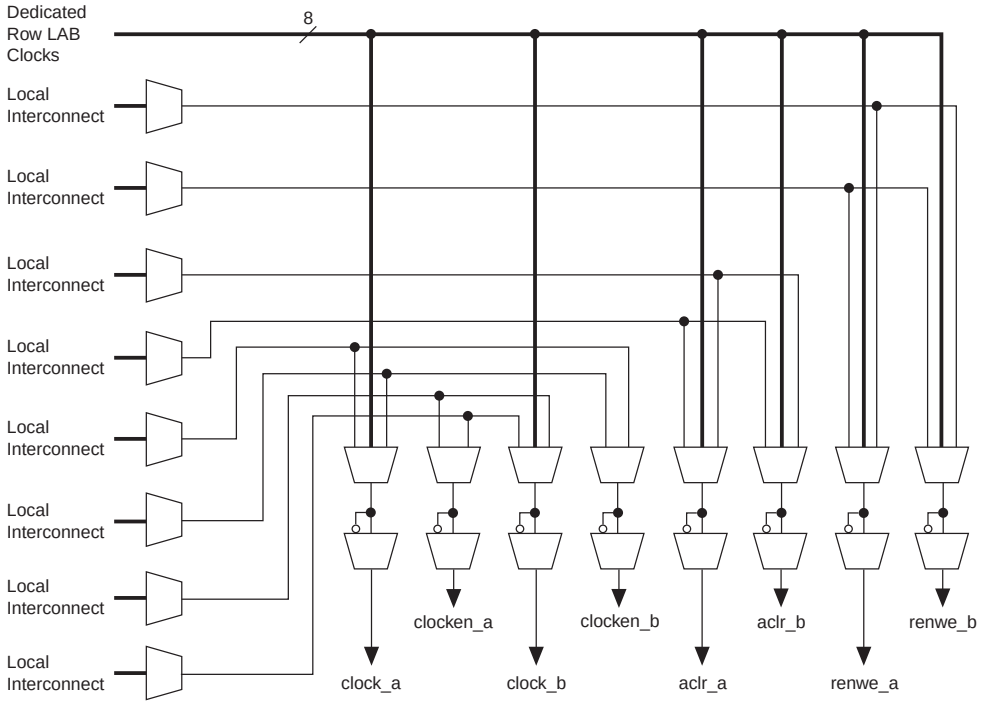
Notes to Tables 4–9 and 4–10:

(1) Any combination of byte enables is possible.

(2) Byte enables can be used in the same manner with 8-bit words, that is, in ×16, ×32, ×64, and ×128 modes.

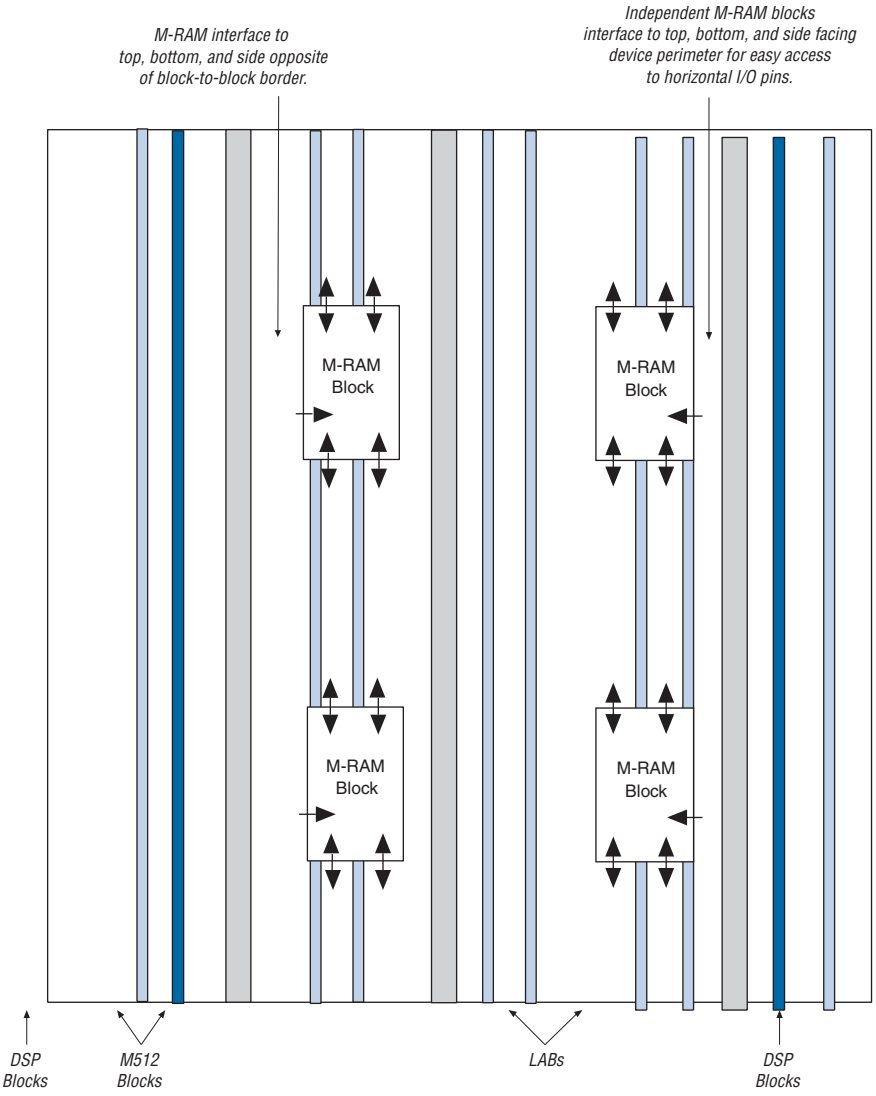
Similar to all RAM blocks, M-RAM blocks can have different clocks on their inputs and outputs. All input registers—`renwe`, `datain`, address, and byte enable registers—are clocked together from either of the two clocks feeding the block. The output register can be bypassed. The eight `labclk` signals or local interconnect can drive the control signals for the A and B ports of the M-RAM block. LEs can also control the `clock_a`, `clock_b`, `renwe_a`, `renwe_b`, `clr_a`, `clr_b`, `clocken_a`, and `clocken_b` signals as shown in Figure 4–18.

Figure 4–18. M-RAM Block Control Signals



One of the M-RAM block's horizontal sides drive the address and control signal (clock, `renwe`, `byteena`, etc.) inputs. Typically, the horizontal side closest to the device perimeter contains the interfaces. The one exception is when two M-RAM blocks are paired next to each other. In this case, the side of the M-RAM block opposite the common side of the two blocks contains the input interface. The top and bottom sides of any M-RAM block contain data input and output interfaces to the logic array. The top side has 72 data inputs and 72 data outputs for port B, and the bottom side has another 72 data inputs and 72 data outputs for port A. Figure 4–19 shows an example floorplan for the EP1SGX40 device and the location of the M-RAM interfaces.

Figure 4–19. EP1SGX40 Device with M-RAM Interface Locations *Note (1)*

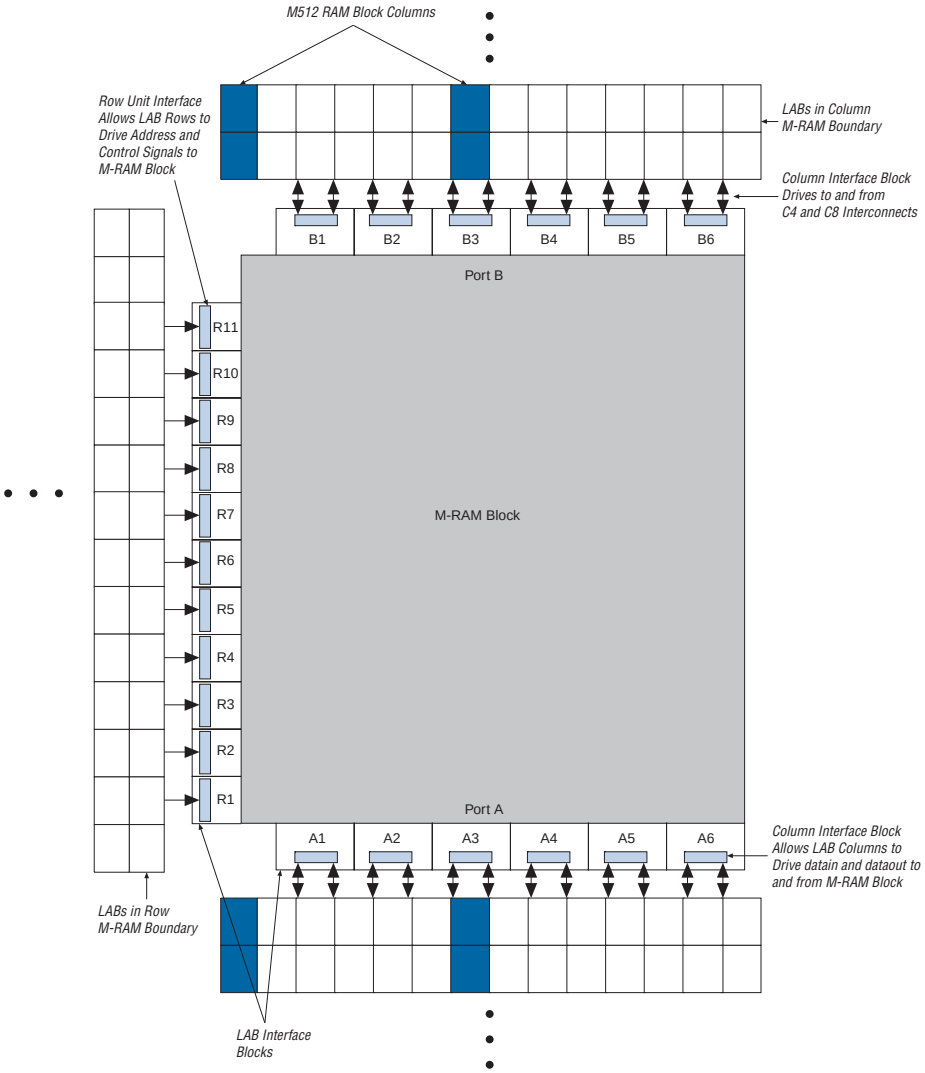


Note to Figure 4–19:
(1) Device shown is an EP1SGX40 device. The number and position of M-RAM blocks varies in other devices.

The M-RAM block local interconnect is driven by the R4, R8, C4, C8, and direct link interconnects from adjacent LABs. For independent M-RAM blocks, up to 10 direct link address and control signal input connections to the M-RAM block are possible from the left adjacent LABs for M-RAM

blocks facing to the left, and another 10 possible from the right adjacent LABs for M-RAM blocks facing to the right. For column interfacing, every M-RAM column unit connects to the right and left column lines, allowing each M-RAM column unit to communicate directly with three columns of LABs. Figures 4–20 through 4–22 show the interface between the M-RAM block and the logic array.

Figure 4–20. Left-Facing M-RAM to Interconnect Interface *Notes (1), (2)*



Notes to Figure 4–20:

(1) Only R24 and C16 interconnects cross the M-RAM block boundaries.

(2) The right-facing M-RAM block has interface blocks on the right side, but none on the left. B1 to B6 and A1 to A6 orientation is clipped across the vertical axis for right-facing M-RAM blocks.

Figure 4–21. M-RAM Row Unit Interface to Interconnect

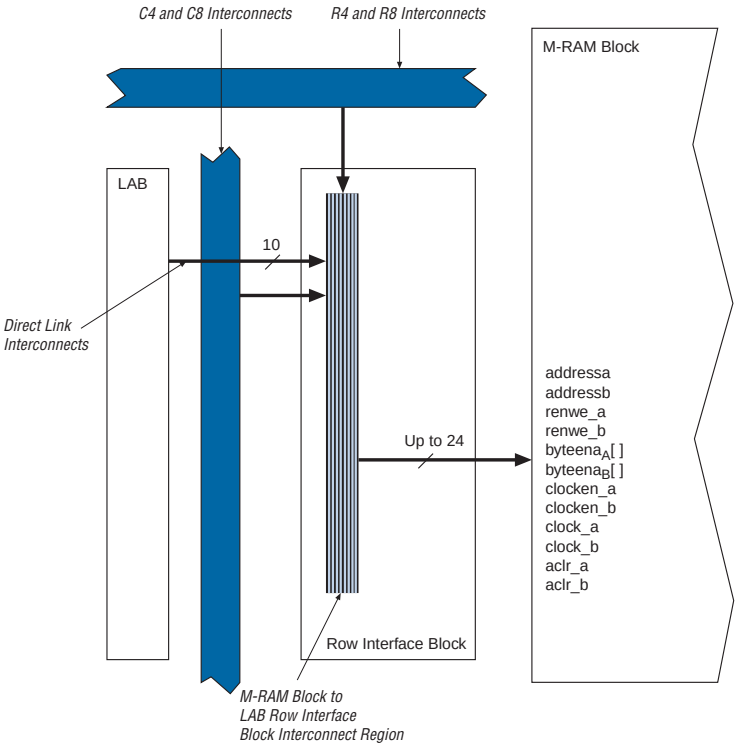


Figure 4–22. M-RAM Column Unit Interface to Interconnect

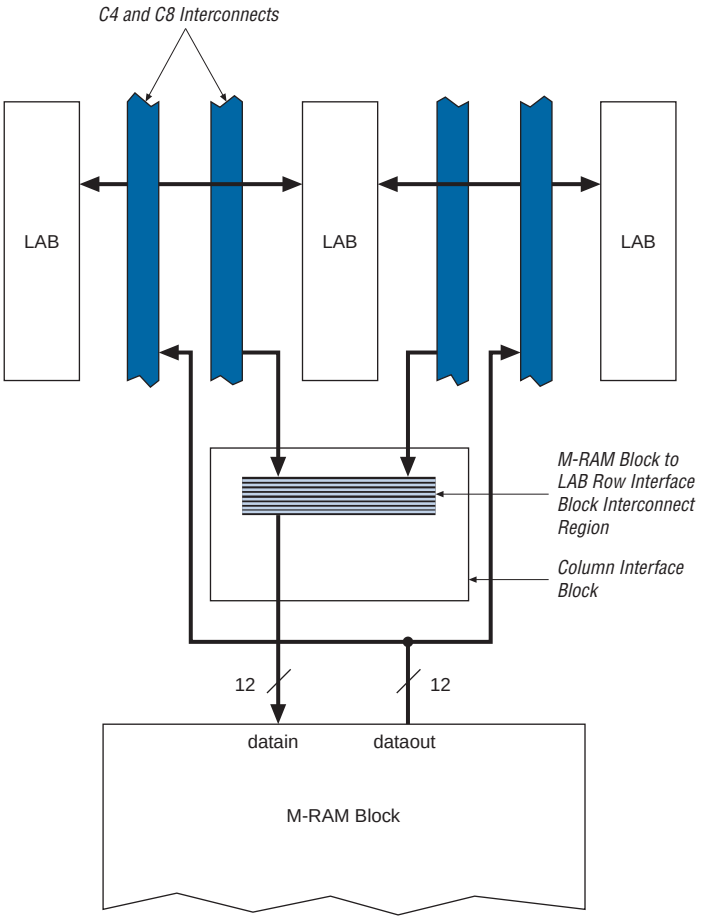


Table 4–11 shows the input and output data signal connections for the column units (B1 to B6 and A1 to A6). It also shows the address and control signal input connections to the row units (R1 to R11).

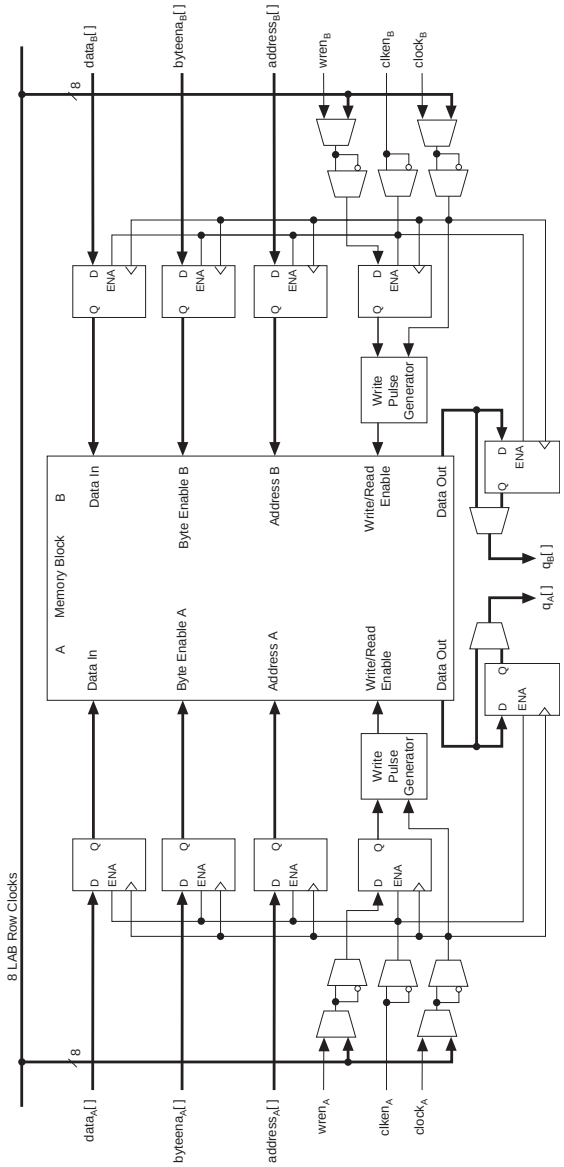
Table 4–11. M-RAM Row & Column Interface Unit Signals

Unit Interface Block	Input SIgnals	Output Signals
R1	addressa[7..0]	
R2	addressa[15..8]	
R3	byte_enable_a[7..0] renwe_a	
R4	-	
R5	-	
R6	clock_a clocken_a clock_b clocken_b	
R7	-	
R8	-	
R9	byte_enable_b[7..0] renwe_b	
R10	addressb[15..8]	
R11	addressb[7..0]	
B1	datain_b[71..60]	dataout_b[71..60]
B2	datain_b[59..48]	dataout_b[59..48]
B3	datain_b[47..36]	dataout_b[47..36]
B4	datain_b[35..24]	dataout_b[35..24]
B5	datain_b[23..12]	dataout_b[23..12]
B6	datain_b[11..0]	dataout_b[11..0]
A1	datain_a[71..60]	dataout_a[71..60]
A2	datain_a[59..48]	dataout_a[59..48]
A3	datain_a[47..36]	dataout_a[47..36]
A4	datain_a[35..24]	dataout_a[35..24]
A5	datain_a[23..12]	dataout_a[23..12]
A6	datain_a[11..0]	dataout_a[11..0]

Independent Clock Mode

The memory blocks implement independent clock mode for true dual-port memory. In this mode, a separate clock is available for each port (ports A and B). Clock A controls all registers on the port A side, while clock B controls all registers on the port B side. Each port, A and B, also supports independent clock enables and asynchronous clear signals for port A and B registers. Figure 4–23 shows a TriMatrix memory block in independent clock mode.

Figure 4–23. Independent Clock Mode *Note (1)*

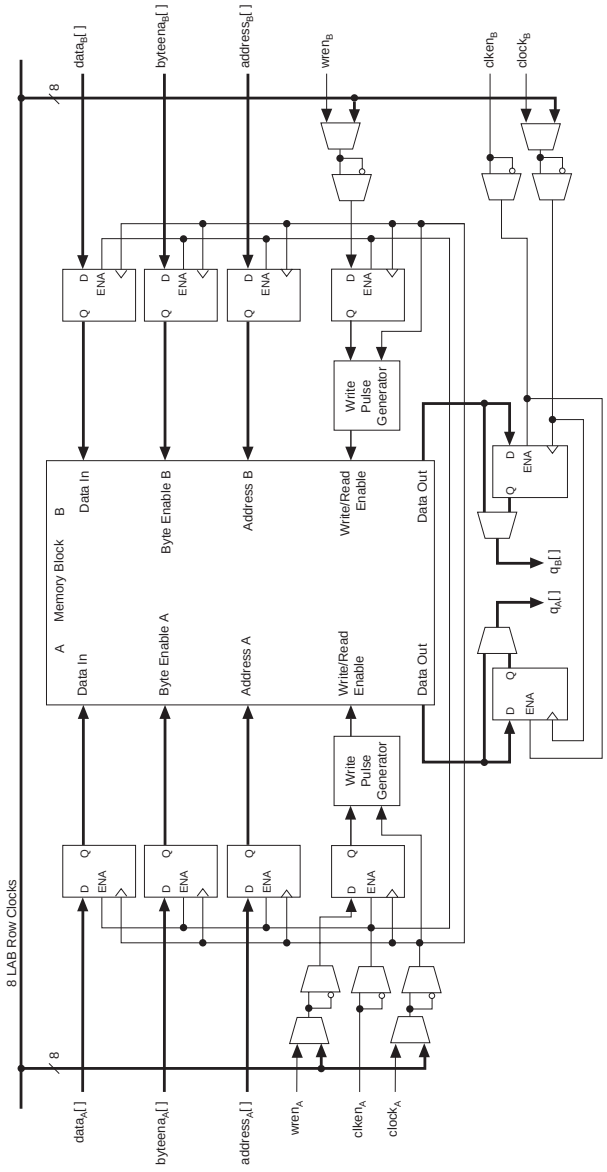


Note to Figure 4–23:
(1) All registers shown have asynchronous clear ports.

Input/Output Clock Mode

Input/output clock mode can be implemented for both the true and simple dual-port memory modes. On each of the two ports, A or B, one clock controls all registers for inputs into the memory block: data input, `wren`, and address. The other clock controls the block's data output registers. Each memory block port, A or B, also supports independent clock enables and asynchronous clear signals for input and output registers. Figures 4–24 and 4–25 show the memory block in input/output clock mode.

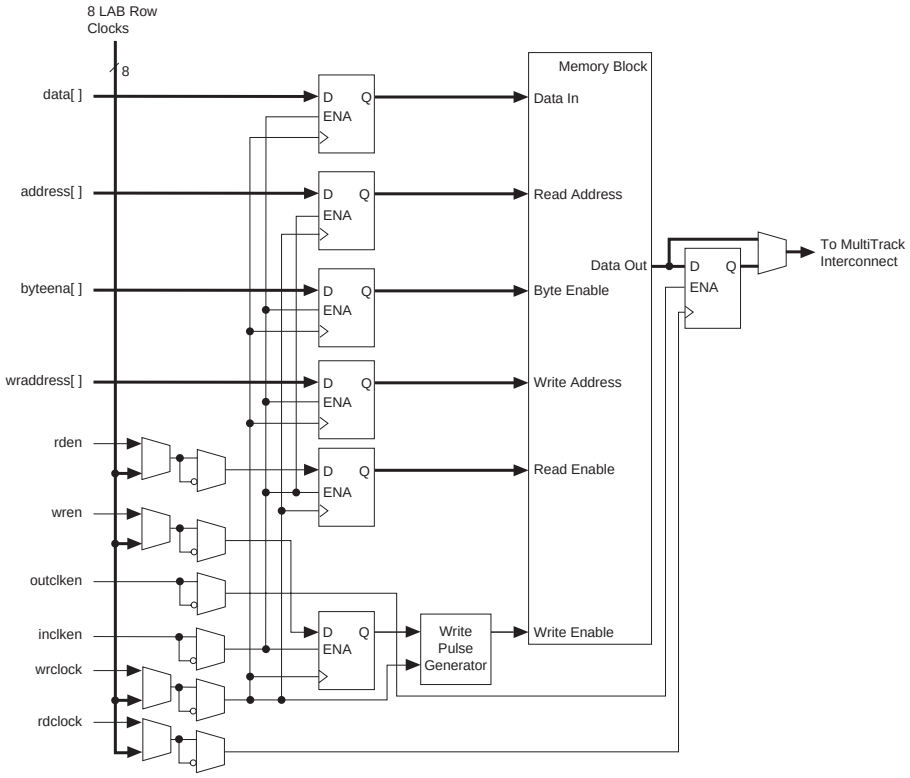
Figure 4–24. Input/Output Clock Mode in True Dual-Port Mode *Note (1)*



Note to Figure 4–24:

(1) All registers shown have asynchronous clear ports.

Figure 4–25. Input/Output Clock Mode in Simple Dual-Port Mode *Note (1)*

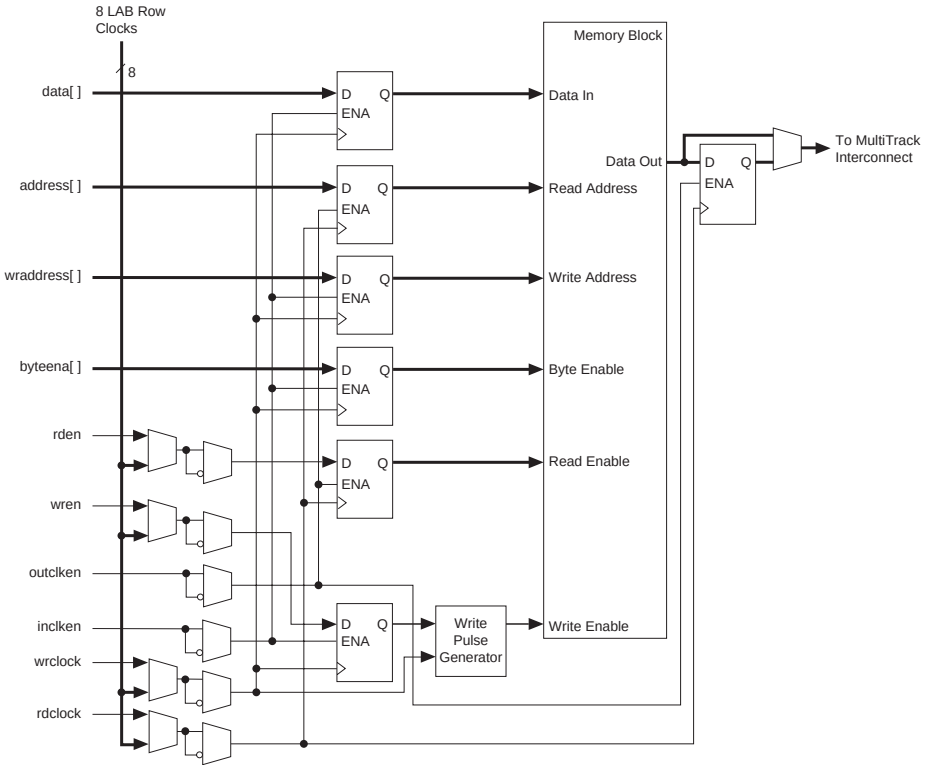


Note to Figure 4–25:
(1) All registers shown except the `rden` register have asynchronous clear ports.

Read/Write Clock Mode

The memory blocks implement read/write clock mode for simple dual-port memory. You can use up to two clocks in this mode. The write clock controls the block's data inputs, `wraddress`, and `wren`. The read clock controls the data output, `rdaddress`, and `rden`. The memory blocks support independent clock enables for each clock and asynchronous clear signals for the read- and write-side registers. Figure 4–26 shows a memory block in read/write clock mode.

Figure 4–26. Read/Write Clock Mode in Simple Dual-Port Mode *Note (1)*

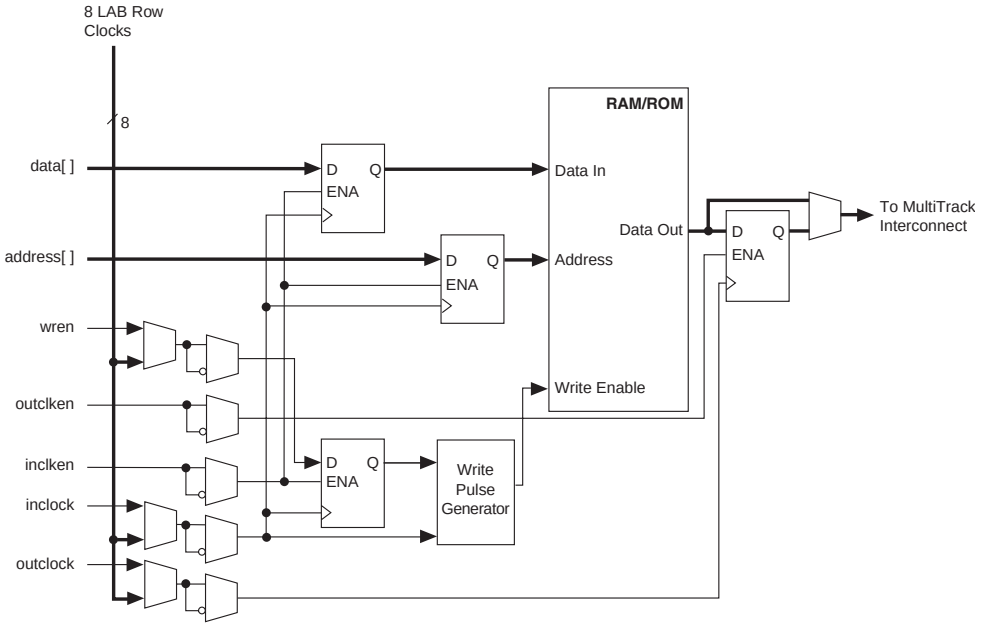


Note to Figure 4–26:
(1) All registers shown except the `rden` register have asynchronous clear ports.

Single-Port Mode

The memory blocks also support single-port mode, used when simultaneous reads and writes are not required. See Figure 4–27. A single block in a memory block can support up to two single-port mode RAM blocks in the M4K RAM blocks if each RAM block is less than or equal to 2K bits in size.

Figure 4–27. Single-Port Mode



Digital Signal Processing Block

The most commonly used DSP functions are finite impulse response (FIR) filters, complex FIR filters, infinite impulse response (IIR) filters, fast Fourier transform (FFT) functions, direct cosine transform (DCT) functions, and correlators. All of these blocks have the same fundamental building block: the multiplier. Additionally, some applications need specialized operations such as multiply-add and multiply-accumulate operations. Stratix GX devices provide DSP blocks to meet the arithmetic requirements of these functions.

Each Stratix GX device has two columns of DSP blocks to efficiently implement DSP functions faster than LE-based implementations. Larger Stratix GX devices have more DSP blocks per column (see Table 4–12). Each DSP block can be configured to support up to:

- Eight 9 × 9-bit multipliers
- Four 18 × 18-bit multipliers
- One 36 × 36-bit multiplier

As indicated, the Stratix GX DSP block can support one 36 × 36-bit multiplier in a single DSP block. This is true for any matched sign multiplications (either unsigned by unsigned or signed by signed), but

the capabilities for dynamic and mixed sign multiplications are handled differently. The following list provides the largest functions that can fit into a single DSP block.

- 36 × 36-bit unsigned by unsigned multiplication
- 36 × 36-bit signed by signed multiplication
- 35 × 36-bit unsigned by signed multiplication
- 36 × 35-bit signed by unsigned multiplication
- 36 × 35-bit signed by dynamic sign multiplication
- 35 × 36-bit dynamic sign by signed multiplication
- 35 × 36-bit unsigned by dynamic sign multiplication
- 36 × 35-bit dynamic sign by unsigned multiplication
- 35 × 35-bit dynamic sign multiplication when the sign controls for each operand are different
- 36 × 36-bit dynamic sign multiplication when the same sign control is used for both operands



This list only shows functions that can fit into a single DSP block. Multiple DSP blocks can support larger multiplication functions.

Figure 4–28 shows one of the columns with surrounding LAB rows.

Figure 4–28. DSP Blocks Arranged in Columns

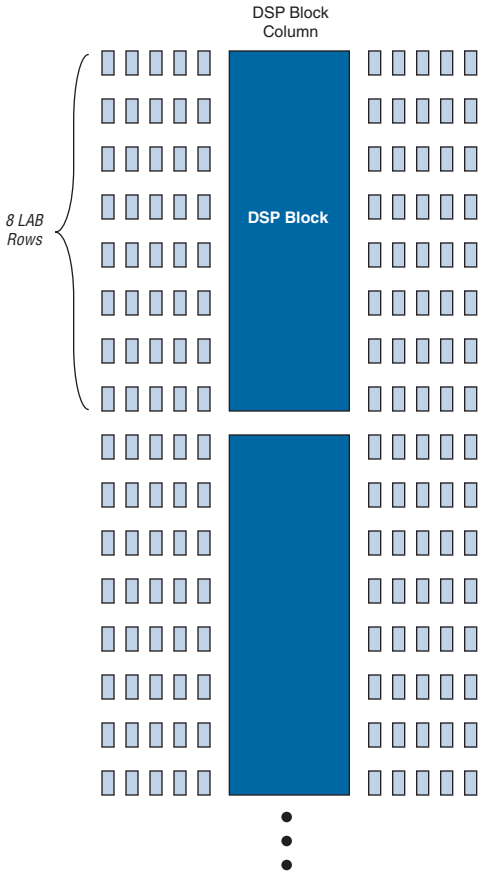


Table 4–12 shows the number of DSP blocks in each Stratix GX device.

Table 4–12. DSP Blocks in Stratix GX Devices *Notes (1), (2)*

Device	DSP Blocks	Total 9 × 9 Multipliers	Total 18 × 18 Multipliers	Total 36 × 36 Multipliers
EP1SGX10	6	48	24	6
EP1SGX25	10	80	40	10
EP1SGX40	14	112	56	14

Notes to Table 4–12:

(1) Each device has either the number of 9 × 9-, 18 × 18-, or 36 × 36-bit multipliers shown. The total number of multipliers for each device is not the sum of all the multipliers.

(2) The number of supported multiply functions shown is based on signed/signed or unsigned/unsigned implementations.

DSP block multipliers can optionally feed an adder/subtractor or accumulator within the block depending on the configuration. This makes routing to LEs easier, saves LE routing resources, and increases performance, because all connections and blocks are within the DSP block. Additionally, the DSP block input registers can efficiently implement shift registers for FIR filter applications.

Figure 4–29 shows the top-level diagram of the DSP block configured for 18 × 18-bit multiplier mode. Figure 4–30 shows the 9 × 9-bit multiplier configuration of the DSP block.

Figure 4–29. DSP Block Diagram for 18 × 18-Bit Configuration

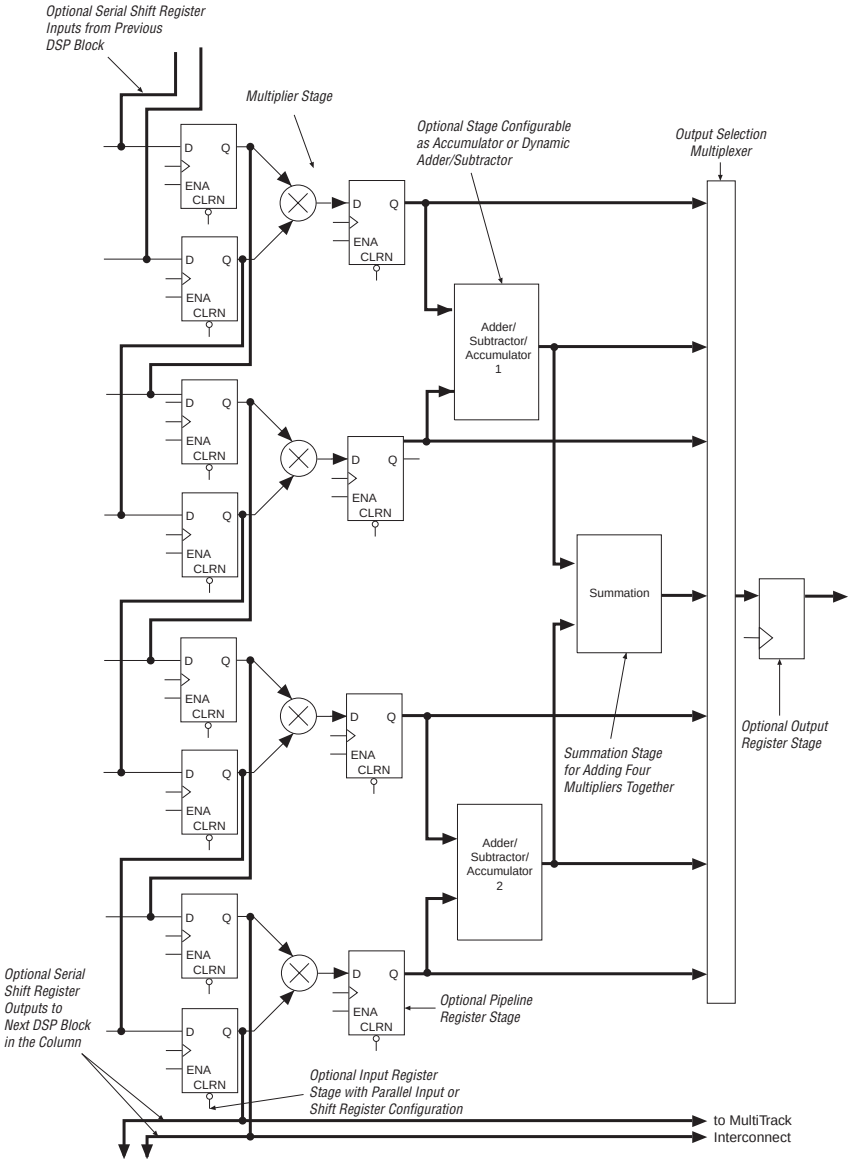
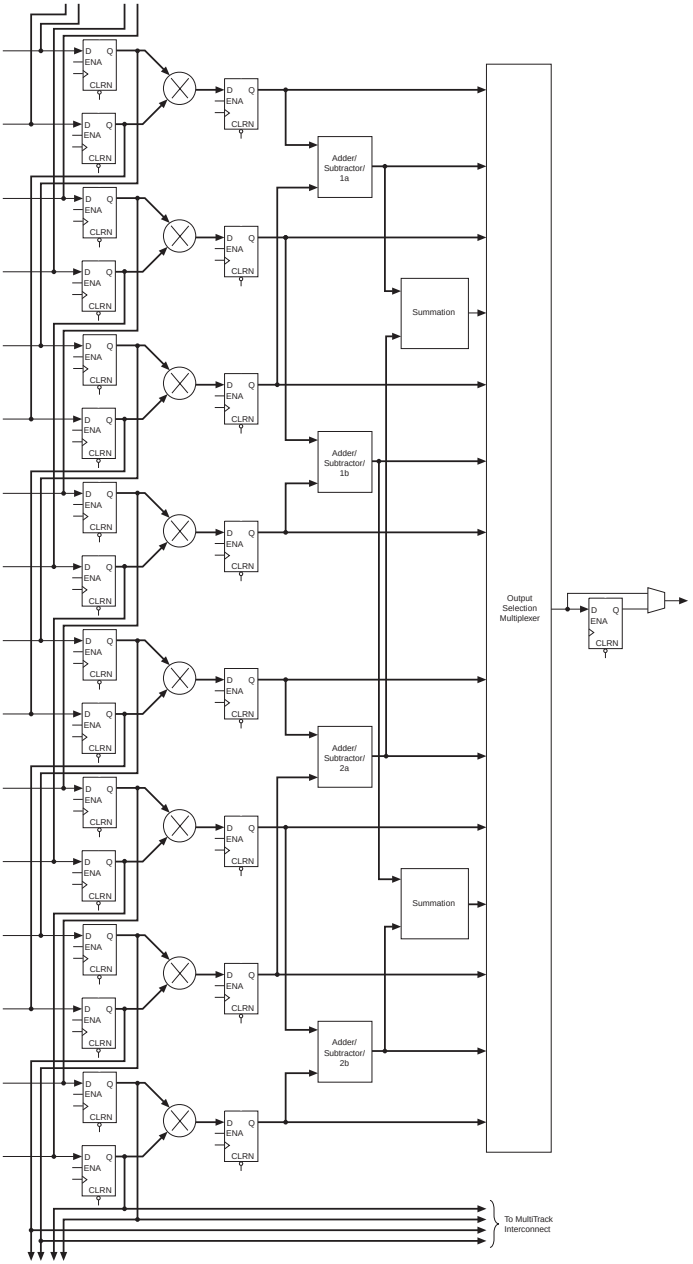


Figure 4–30. DSP Block Diagram for 9 × 9-Bit Configuration



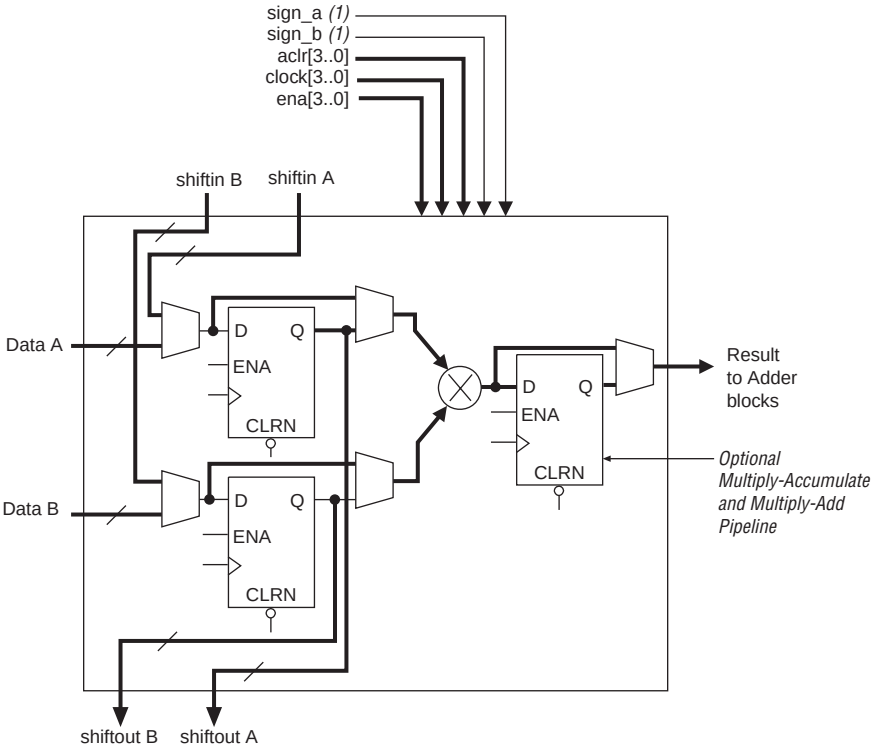
The DSP block consists of the following elements:

- Multiplier block
- Adder/output block

Multiplier Block

The DSP block multiplier block consists of the input registers, a multiplier, and pipeline register for pipelining multiply-accumulate and multiply-add/subtract functions as shown in Figure 4–31.

Figure 4–31. Multiplier Sub-Block Within Stratix GX DSP Block



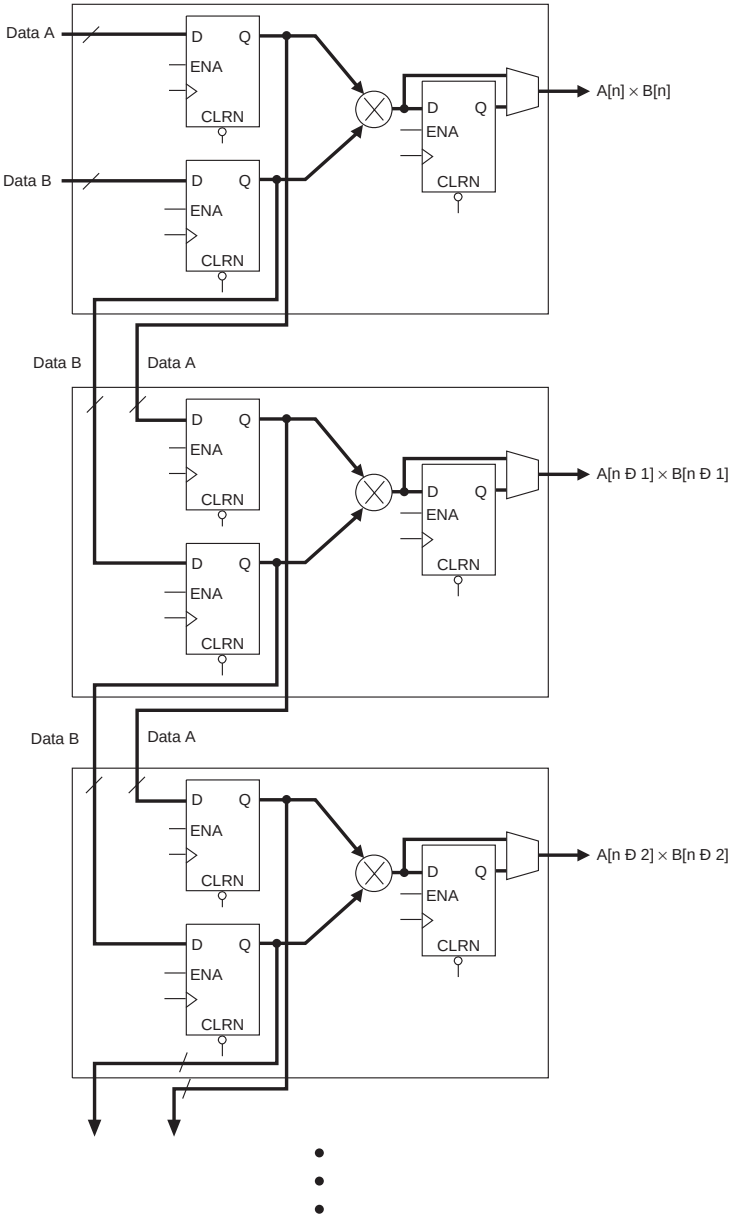
Note to Figure 4–31:
(1) These signals can be unregistered or registered once to match data path pipelines if required.

Input Registers

A bank of optional input registers is located at the input of each multiplier and multiplicand inputs to the multiplier. When these registers are configured for parallel data inputs, they are driven by regular routing resources. You can use a clock signal, asynchronous clear signal, and a clock enable signal to independently control each set of A and B inputs for each multiplier in the DSP block. You select these control signals from a set of four different `clock[3..0]`, `aclr[3..0]`, and `ena[3..0]` signals that drive the entire DSP block.

You can also configure the input registers for a shift register application. In this case, the input registers feed the multiplier and drive two dedicated shift output lines: `shiftoutA` and `shiftoutB`. The shift outputs of one multiplier block directly feed the adjacent multiplier block in the same DSP block (or the next DSP block) as shown in Figure 4–32, to form a shift register chain. This chain can terminate in any block, that is, you can create any length of shift register chain up to 224 registers. You can use the input shift registers for FIR filter applications. One set of shift inputs can provide data for a filter, and the other are coefficients that are optionally loaded in serial or parallel. When implementing 9 × 9- and 18 × 18-bit multipliers, you do not need to implement external shift registers in LAB LEs. You implement all the filter circuitry within the DSP block and its routing resources, saving LE and general routing resources for general logic. External registers are needed for shift register inputs when using 36 × 36-bit multipliers.

Figure 4–32. Multiplier Sub-Blocks Using Input Shift Register Connections *Note (1)*



Note to Figure 4–32:
(1) Either Data A or Data B input can be set to a parallel input for constant coefficient multiplication.

Table 4–13 shows the summary of input register modes for the DSP block.

Table 4–13. Input Register Modes

Register Input Mode	9 × 9	18 × 18	36 × 36
Parallel input	✓	✓	✓
Shift register input	✓	✓	

Multiplier

The multiplier supports 9 × 9-, 18 × 18-, or 36 × 36-bit multiplication. Each DSP block supports eight possible 9 × 9-bit or smaller multipliers. There are four multiplier blocks available for multipliers larger than 9 × 9 bits but smaller than 18 × 18 bits. There is one multiplier block available for multipliers larger than 18 × 18 bits but smaller than or equal to 36 × 36 bits. The ability to have several small multipliers is useful in applications such as video processing. Large multipliers greater than 18 × 18 bits are useful for applications such as the mantissa multiplication of a single-precision floating-point number.

The multiplier operands can be signed or unsigned numbers, where the result is signed if either input is signed as shown in Table 4–14. The `sign_a` and `sign_b` signals provide dynamic control of each operand's representation: a logic 1 indicates the operand is a signed number, a logic 0 indicates the operand is an unsigned number. These sign signals affect all multipliers and adders within a single DSP block and you can register them to match the data path pipeline. The multipliers are full precision (that is, 18 bits for the 18-bit multiply, 36-bits for the 36-bit multiply, and so on), regardless of whether `sign_a` or `sign_b` set the operands as signed or unsigned numbers.

Table 4–14. Multiplier Signed Representation

Data A	Data B	Result
Unsigned	Unsigned	Unsigned
Unsigned	Signed	Signed
Signed	Unsigned	Signed
Signed	Signed	Signed

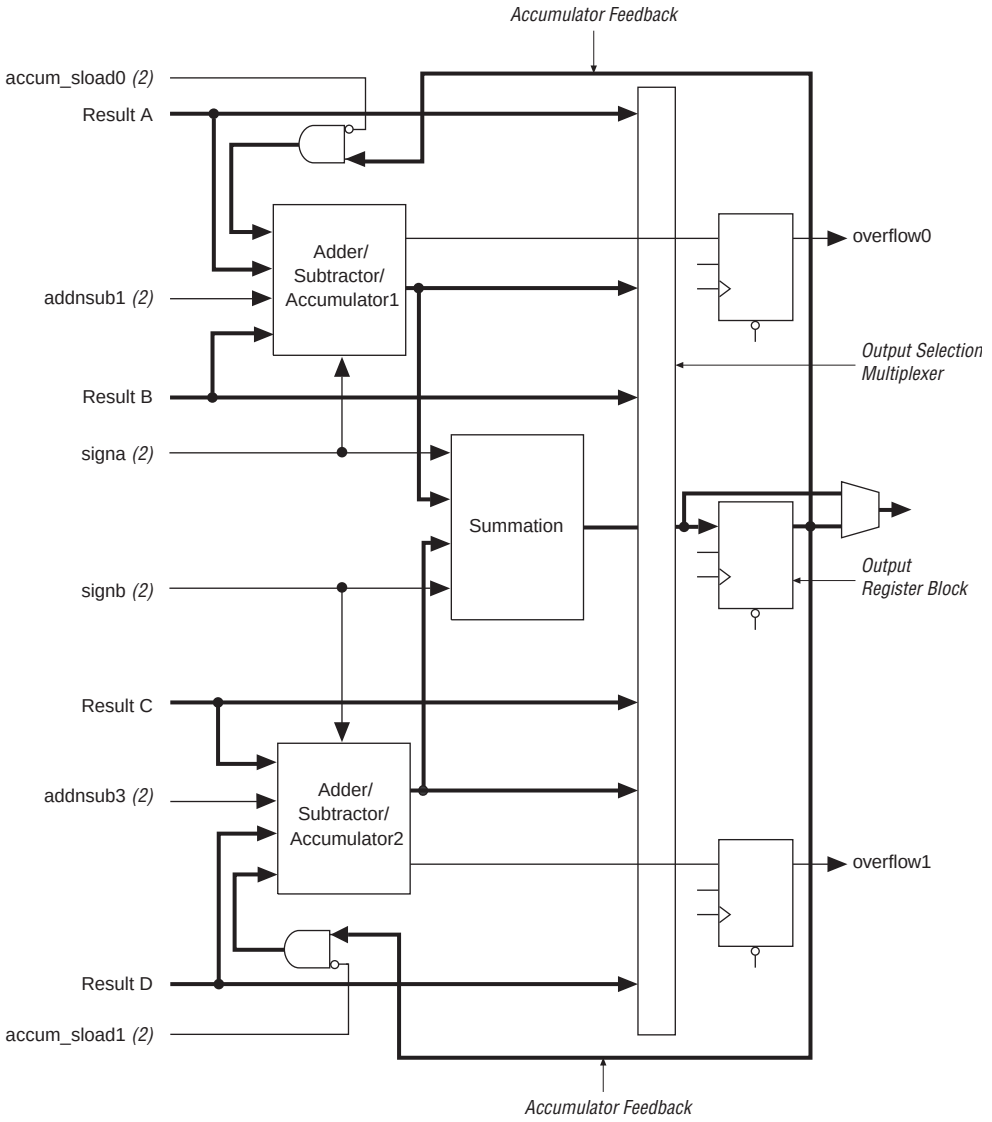
Pipeline/Post Multiply Register

The output of 9 × 9- or 18 × 18-bit multipliers can optionally feed a register to pipeline multiply-accumulate and multiply-add/subtract functions. For 36 × 36-bit multipliers, this register pipelines the multiplier function.

Adder/Output Blocks

The result of the multiplier sub-blocks are sent to the adder/output block which consist of an adder/subtractor/accumulator unit, summation unit, output select multiplexer, and output registers. The results are used to configure the adder/output block as a pure output, accumulator, a simple two-multiplier adder, four-multiplier adder, or final stage of the 36-bit multiplier. You can configure the adder/output block to use output registers in any mode, and must use output registers for the accumulator. The system cannot use adder/output blocks independently of the multiplier. Figure 4–33 shows the adder and output stages.

Figure 4–33. Adder/Output Blocks *Note (1)*



Notes to Figure 4–33:

(1) Adder/output block shown in Figure 4–33 is in 18 × 18-bit mode. In 9 × 9-bit mode, there are four adder/subtractor blocks and two summation blocks.

(2) These signals are either not registered, registered once, or registered twice to match the data path pipeline.

Adder/Subtractor/Accumulator

The adder/subtractor/accumulator is the first level of the adder/output block and can be used as an accumulator or as an adder/subtractor.

Adder/Subtractor
Each adder/subtractor/accumulator block can perform addition or subtraction using the `addnsub` independent control signal for each first-level adder in 18 × 18-bit mode. There are two `addnsub[1..0]` signals available in a DSP block for any configuration. For 9 × 9-bit mode, one `addnsub[1..0]` signal controls the top two one-level adders and another `addnsub[1..0]` signal controls the bottom two one-level adders. A high `addnsub` signal indicates addition, and a low signal indicates subtraction. The `addnsub` control signal can be unregistered or registered once or twice when feeding the adder blocks to match data path pipelines.

The `signa` and `signb` signals serve the same function as the multiplier block `signa` and `signb` signals. The only difference is that these signals can be registered up to two times. These signals are tied to the same `signa` and `signb` signals from the multiplier and must be connected to the same clocks and control signals.

Accumulator
When configured for accumulation, the adder/output block output feeds back to the accumulator as shown in Figure 4–33. The `accum_sload[1..0]` signal synchronously loads the multiplier result to the accumulator output. This signal can be unregistered or registered once or twice. Additionally, the `overflow` signal indicates the accumulator has overflowed or underflowed in accumulation mode. This signal is always registered and must be externally latched in LEs if the design requires a latched `overflow` signal.

Summation

The output of the adder/subtractor/accumulator block feeds to an optional summation block. This block sums the outputs of the DSP block multipliers. In 9 × 9-bit mode, there are two summation blocks providing the sums of two sets of four 9 × 9-bit multipliers. In 18 × 18-bit mode, there is one summation providing the sum of one set of four 18 × 18-bit multipliers.

Output Selection Multiplexer

The outputs from the various elements of the adder/output block are routed through an output selection multiplexer. Based on the DSP block operational mode and user settings, the multiplexer selects whether the output from the multiplier, the adder/subtractor/accumulator, or summation block feeds to the output.

Output Registers

Optional output registers for the DSP block outputs are controlled by four sets of control signals: `clock[3..0]`, `aclr[3..0]`, and `ena[3..0]`. Output registers can be used in any mode.

Modes of Operation

The adder, subtractor, and accumulate functions of a DSP block have four modes of operation:

- Simple multiplier
- Multiply-accumulator
- Two-multipliers adder
- Four-multipliers adder

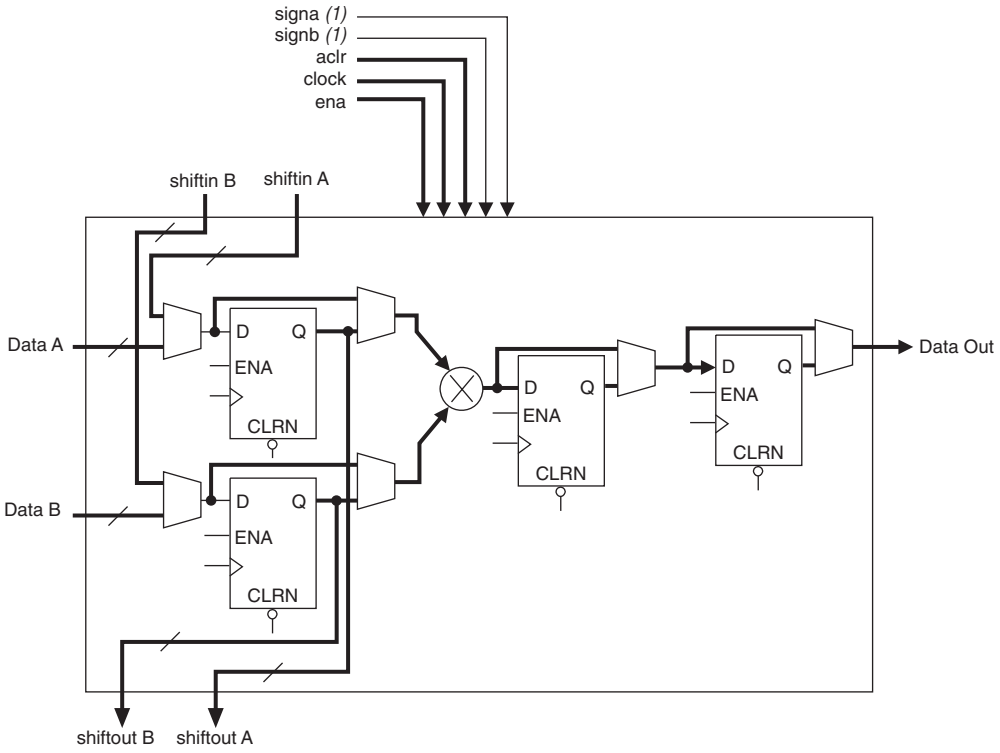


Each DSP block can only support one mode. Mixed modes in the same DSP block is not supported.

Simple Multiplier Mode

In simple multiplier mode, the DSP block drives the multiplier sub-block result directly to the output with or without an output register. Up to four 18 × 18-bit multipliers or eight 9 × 9-bit multipliers can drive their results directly out of one DSP block. See Figure 4–34.

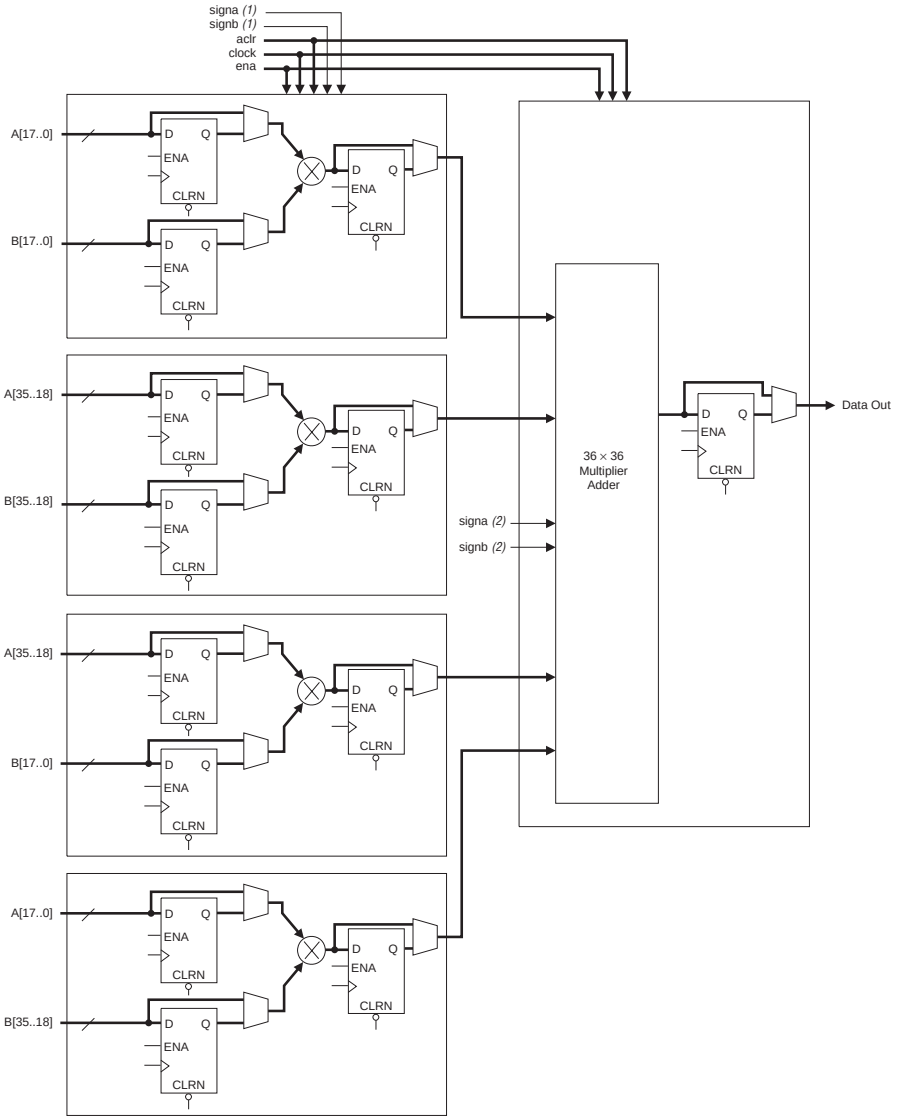
Figure 4–34. Simple Multiplier Mode



Note to Figure 4–34:
(1) These signals are not registered or registered once to match the data path pipeline.

DSP blocks can also implement one 36 × 36-bit multiplier in multiplier mode. DSP blocks use four 18 × 18-bit multipliers combined with dedicated adder and internal shift circuitry to achieve 36-bit multiplication. The input shift register feature is not available for the 36 × 36-bit multiplier. In 36 × 36-bit mode, the device can use the register that is normally a multiplier-result-output register as a pipeline stage for the 36 × 36-bit multiplier. Figure 4–35 shows the 36 × 36-bit multiply mode.

Figure 4–35. 36 × 36 Multiply Mode



Notes to Figure 4–35:

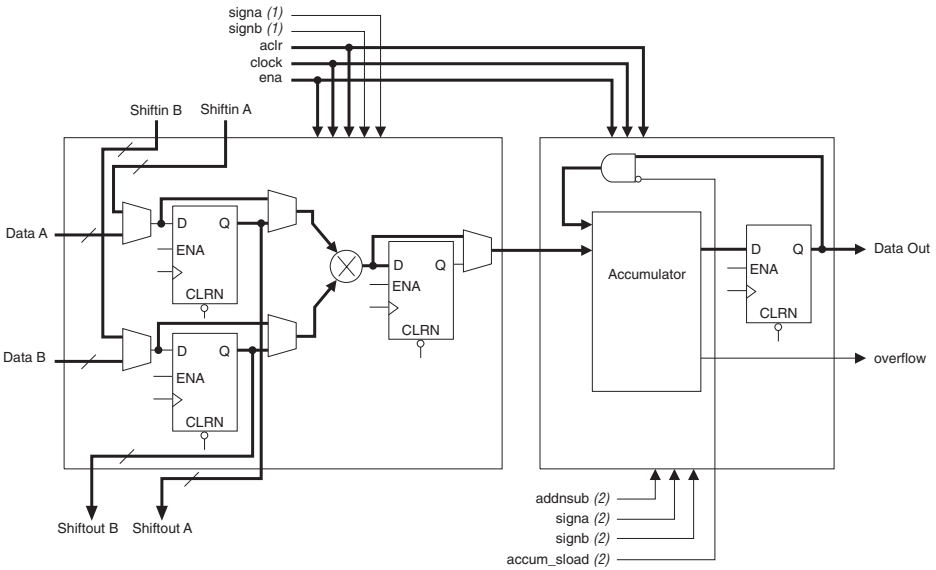
(1) These signals are not registered or registered once to match the pipeline.

(2) These signals are not registered, registered once, or registered twice for latency to match the pipeline.

Multiply-Accumulator Mode

In multiply-accumulator mode (see Figure 4–36), the DSP block drives multiplied results to the adder/subtractor/accumulator block configured as an accumulator. You can implement one or two multiply-accumulators up to 18 × 18 bits in one DSP block. The first and third multiplier sub-blocks are unused in this mode, since only one multiplier can feed one of two accumulators. The multiply-accumulator output can be up to 52 bits—a maximum of a 36-bit result with 16 bits of accumulation. The `accum_sload` and `overflow` signals are only available in this mode. The `addnsub` signal can set the accumulator for decimation and the `overflow` signal indicates underflow condition.

Figure 4–36. Multiply-Accumulate Mode



Notes to Figure 4–36:
(1) These signals are not registered or registered once to match the data path pipeline.
(2) These signals are not registered, registered once, or registered twice for latency to match the data path pipeline.

Two-Multipliers Adder Mode

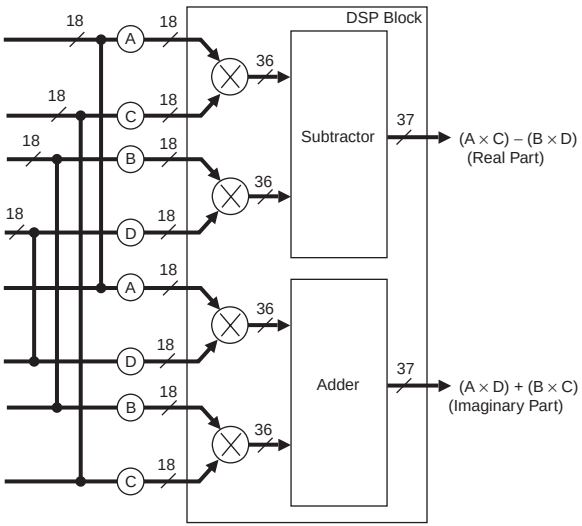
The two-multipliers adder mode uses the adder/subtractor/accumulator block to add or subtract the outputs of the multiplier block, which is useful for applications such as FFT functions and complex FIR filters. A single DSP block can implement two sums or differences from two 18 × 18-bit multipliers each or four sums or differences from two 9 × 9-bit multipliers each.

You can use the two-multipliers adder mode for complex multiplications, which are written as:

$$(a + jb) \times (c + jd) = [(a \times c) - (b \times d)] + j \times [(a \times d) + (b \times c)]$$

The two-multipliers adder mode allows a single DSP block to calculate the real part $[(a \times c) - (b \times d)]$ using one subtractor and the imaginary part $[(a \times d) + (b \times c)]$ using one adder, for data widths up to 18 bits. Two complex multiplications are possible for data widths up to 9 bits using four adder/subtractor/accumulator blocks. Figure 4–37 shows an 18-bit two-multipliers adder.

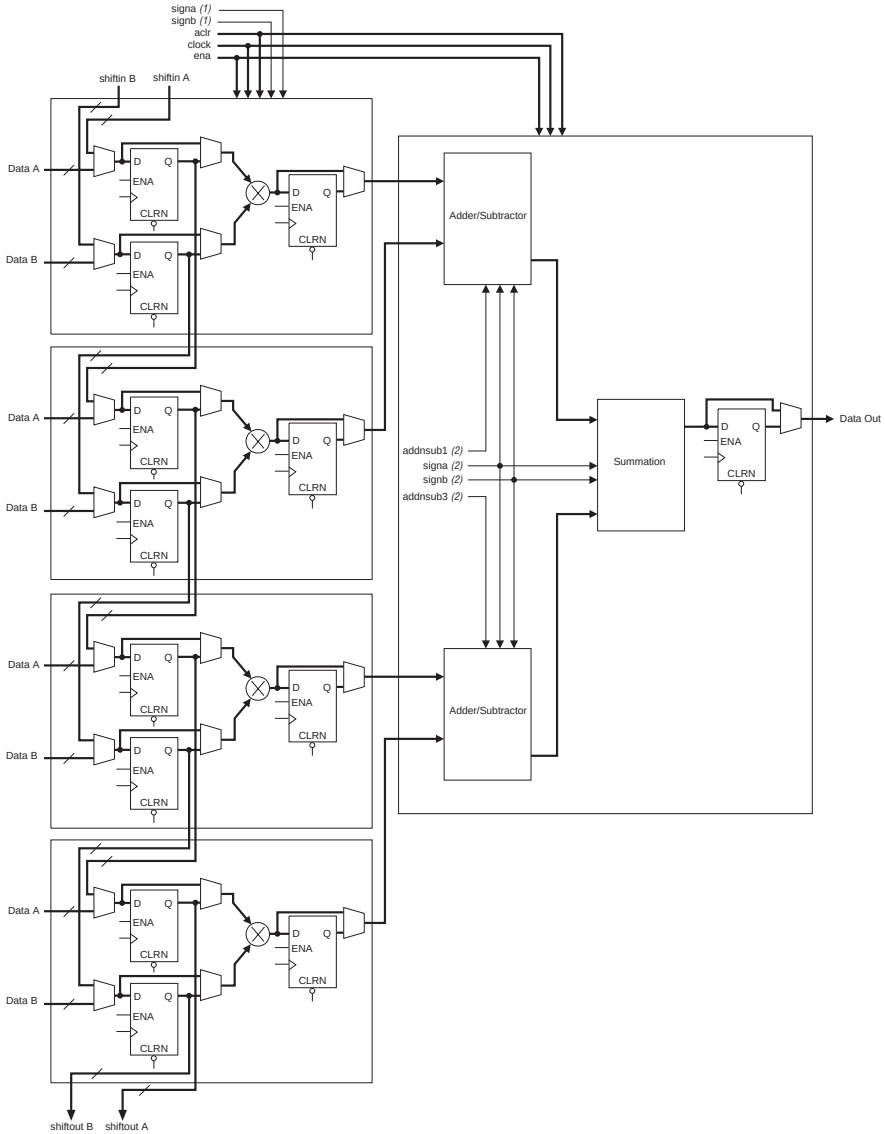
Figure 4–37. Two-Multipliers Adder Mode Implementing Complex Multiply



Four-Multipliers Adder Mode

In the four-multipliers adder mode, the DSP block adds the results of two first -stage adder/subtractor blocks. One sum of four 18 × 18-bit multipliers or two different sums of two sets of four 9 × 9-bit multipliers can be implemented in a single DSP block. The product width for each multiplier must be the same size. The four-multipliers adder mode is useful for FIR filter applications. Figure 4–38 shows the four multipliers adder mode.

Figure 4–38. Four-Multipliers Adder Mode



Notes to Figure 4–38:

(1) These signals are not registered or registered once to match the data path pipeline.

(2) These signals are not registered, registered once, or registered twice for latency to match the data path pipeline.

For FIR filters, the DSP block combines the four-multipliers adder mode with the shift register inputs. One set of shift inputs contains the filter data, while the other holds the coefficients loaded in serial or parallel. The input shift register eliminates the need for shift registers external to the DSP block (that is, implemented in LEs). This architecture simplifies filter design since the DSP block implements all of the filter circuitry.

One DSP block can implement an entire 18-bit FIR filter with up to four taps. For FIR filters larger than four taps, DSP blocks can be cascaded with additional adder stages implemented in LEs.

Table 4–15 shows the different number of multipliers possible in each DSP block mode according to size. These modes allow the DSP blocks to implement numerous applications for DSP including FFTs, complex FIR, FIR, and 2D FIR filters, equalizers, IIR, correlators, matrix multiplication and many other functions.

Table 4–15. Multiplier Size & Configurations per DSP block

DSP Block Mode	9 × 9	18 × 18	36 × 36 *(1)*
Multiplier	Eight multipliers with eight product outputs	Four multipliers with four product outputs	One multiplier with one product output
Multiply-accumulator	Two multiply and accumulate (52 bits)	Two multiply and accumulate (52 bits)	–
Two-multipliers adder	Four sums of two multiplier products each	Two sums of two multiplier products each	–
Four-multipliers adder	Two sums of four multiplier products each	One sum of four multiplier products each	–

Note to Table 4–15:
(1) The number of supported multiply functions shown is based on signed/signed or unsigned/unsigned implementations.

DSP Block Interface

Stratix GX device DSP block outputs can cascade down within the same DSP block column. Dedicated connections between DSP blocks provide fast connections between the shift register inputs to cascade the shift register chains. You can cascade DSP blocks for 9 × 9- or 18 × 18-bit FIR filters larger than four taps, with additional adder stages implemented in LEs. If the DSP block is configured as 36 × 36 bits, the adder, subtractor, or accumulator stages are implemented in LEs. Each DSP block can route the shift register chain out of the block to cascade two full columns of DSP blocks.

The DSP block is divided into eight block units that interface with eight LAB rows on the left and right. Each block unit can be considered half of an 18 × 18-bit multiplier sub-block with 18 inputs and 18 outputs. A local interconnect region is associated with each DSP block. Like an LAB, this interconnect region can be fed with 10 direct link interconnects from the LAB to the left or right of the DSP block in the same row. All row and column routing resources can access the DSP block's local interconnect region. The outputs also work similarly to LAB outputs as well. Nine outputs from the DSP block can drive to the left LAB through direct link interconnects and nine can drive to the right LAB though direct link interconnects. All 18 outputs can drive to all types of row and column routing. Outputs can drive right- or left-column routing. Figures 4–39 and 4–40 show the DSP block interfaces to LAB rows.

Figure 4–39. DSP Block Interconnect Interface

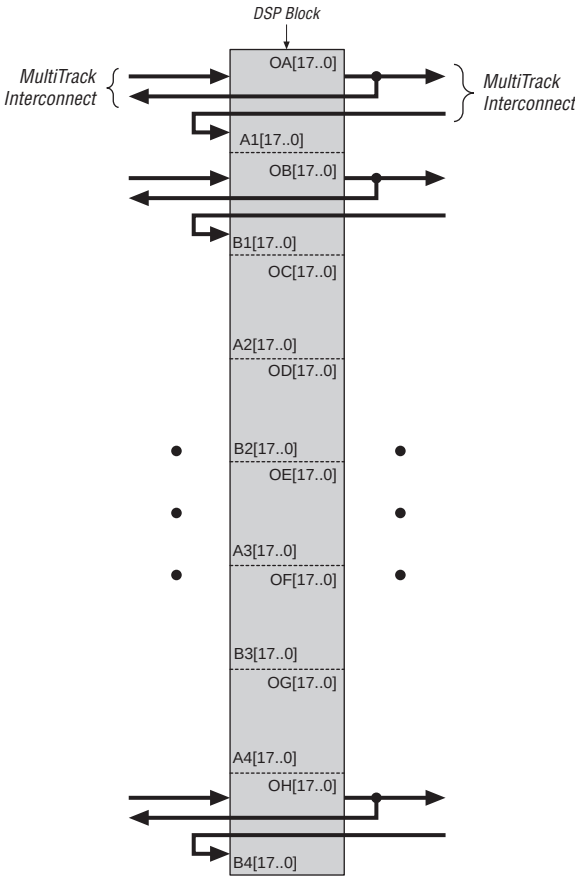
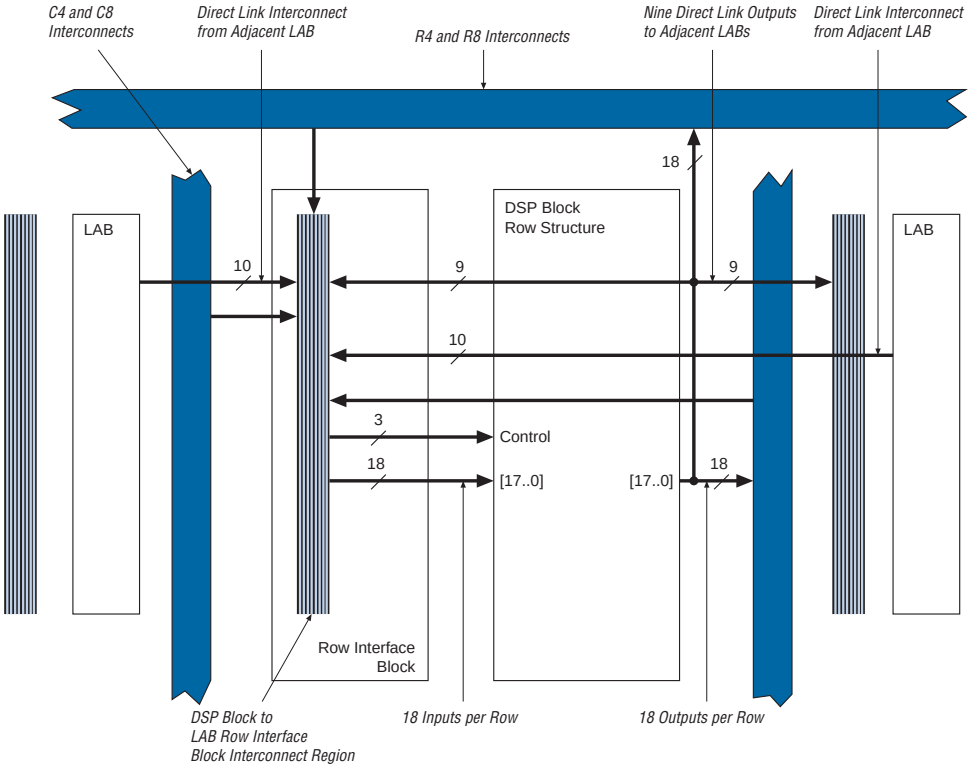


Figure 4–40. DSP Block Interface to Interconnect



A bus of 18 control signals feeds the entire DSP block. These signals include `clock[0..3]` clocks, `aclr[0..3]` asynchronous clears, `ena[1..4]` clock enables, `signa`, `signb` signed/unsigned control signals, `addnsub1` and `addnsub3` addition and subtraction control signals, and `accum_sload[0..1]` accumulator synchronous loads. The

clock signals are routed from LAB row clocks and are generated from specific LAB rows at the DSP block interface. The LAB row source for control signals, data inputs, and outputs is shown in Table 4–16.

Table 4–16. DSP Block Signal Sources & Destinations

LAB Row at Interface	Control Signals Generated	Data Inputs	Data Outputs
1	signa	A1[17..0]	OA[17..0]
2	aclr0 accum_sload0	B1[17..0]	OB[17..0]
3	addnsub1 clock0 ena0	A2[17..0]	OC[17..0]
4	aclr1 clock1 ena1	B2[17..0]	OD[17..0]
5	aclr2 clock2 ena2	A3[17..0]	OE[17..0]
6	sign_b clock3 ena3	B3[17..0]	OF[17..0]
7	clear3 accum_sload1	A4[17..0]	OG[17..0]
8	addnsub3	B4[17..0]	OH[17..0]

PLLs & Clock Networks

Stratix GX devices provide a hierarchical clock structure and multiple PLLs with advanced features. The large number of clocking resources in combination with the clock synthesis precision provided by enhanced and fast PLLs provides a complete clock management solution. Stratix GX devices contain up to four enhanced PLLs and up to four fast PLLs. In addition, there are four receiver PLLs and one transmitter PLL per transceiver block located on the right side of Stratix GX devices.

Global & Hierarchical Clocking

Stratix GX devices provide 16 dedicated global clock networks, 16 regional clock networks (four per device quadrant), 8 dedicated fast regional clock networks within EP1SGX10 and EP1SGX25, and 16 dedicated fast regional clock networks within EP1SGX40 devices.

These clocks are organized into a hierarchical clock structure that allows for up to 22 clocks per device region with low skew and delay. This hierarchical clocking scheme provides up to 40 unique clock domains within EP1SGX10 and EP1SGX25 devices, and 48 unique clock domains within EP1SGX40 devices.

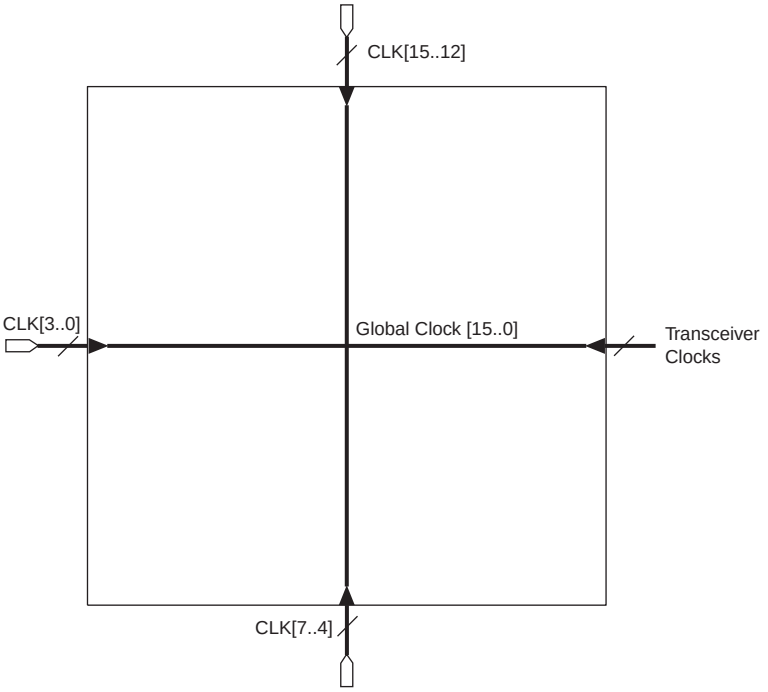
There are 12 dedicated clock pins (CLK[15..12], and CLK[7..0]) to drive either the global or regional clock networks. Three clock pins drive the top, bottom, and left side of the device. Enhanced and fast PLL outputs as well as an I/O interface can also drive these global and regional clock networks.

There are up to 20 recovered clocks (rxclkout[20..0]) and up to 5 transmitter clock outputs (coreclk_out) which can drive any of the global clock networks (CLK[15..0]), as shown in Figure 4–41.

Global Clock Network

These clocks drive throughout the entire device, feeding all device quadrants. The global clock networks can be used as clock sources for all resources within the device IOEs, LEs, DSP blocks, and all memory blocks. These resources can also be used for control signals, such as clock enables and synchronous or asynchronous clears fed from the external pin. The global clock networks can also be driven by internal logic for internally generated global clocks and asynchronous clears, clock enables, or other control signals with large fanout. Figure 4–41 shows the 12 dedicated CLK pins and the transceiver clocks driving global clock networks.

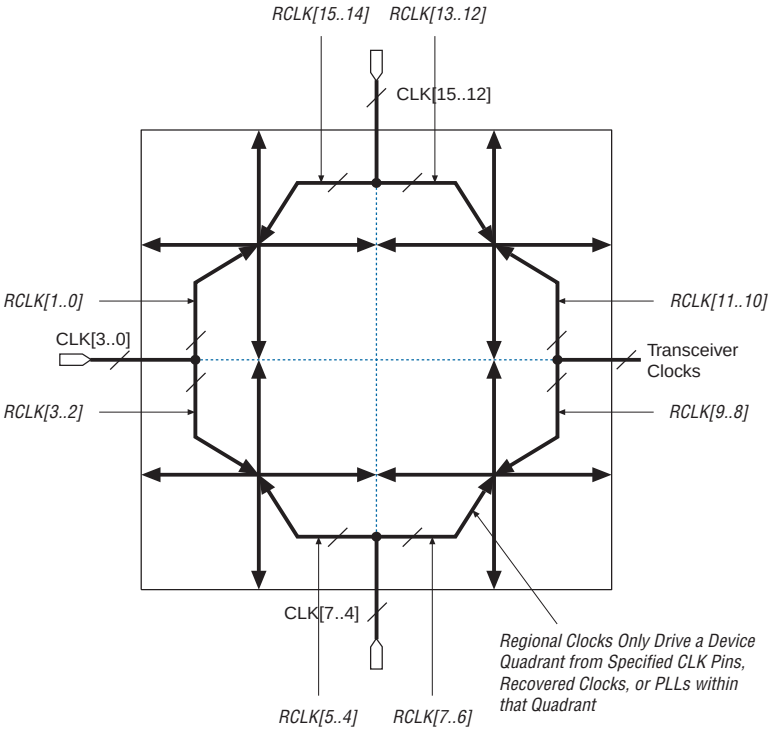
Figure 4–41. Global Clock Resources



Regional Clock Network

There are four regional clock networks RCLK[3..0] within each quadrant of the Stratix GX device that are driven by the same dedicated CLK[7..0] and CLK[15..12] input pins, PLL outputs, or transceiver clocks. The regional clock networks only pertain to the quadrant they drive into. The regional clock networks provide the lowest clock delay and skew for logic contained within a single quadrant. The CLK clock pins symmetrically drive the RCLK networks within a particular quadrant, as shown in Figure 4–42.

Figure 4–42. Regional Clocks



Fast Regional Clock Network

In EP1SGX25 and EP1SGX10 devices, there are two fast regional clock networks, FCLK[1..0], within each quadrant, fed by input pins (see Figure 4–43). In EP1SGX40 devices, there are two fast regional clock networks within each half-quadrant (see Figure 4–44). The FCLK[1..0] clocks can also be used for high fanout control signals, such as asynchronous clears, presets, clock enables, or protocol control signals such as TRDY and IRDY for PCI. Dual-purpose FCLK pins drive the fast clock networks. All devices have eight FCLK pins to drive fast regional clock networks. Any I/O pin can drive a clock or control signal onto any fast regional clock network with the addition of a delay. The I/O interconnect drives this signal.

Figure 4–43. EP1SGX25 & EP1SGX10 Device Fast Clock Pin Connections to Fast Regional Clocks

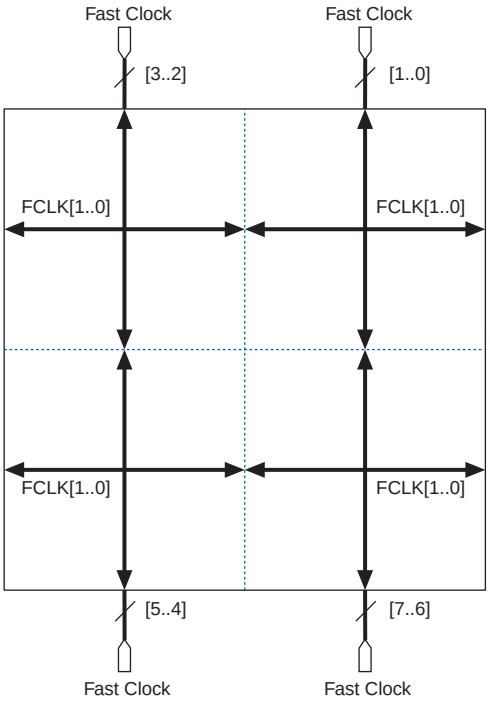
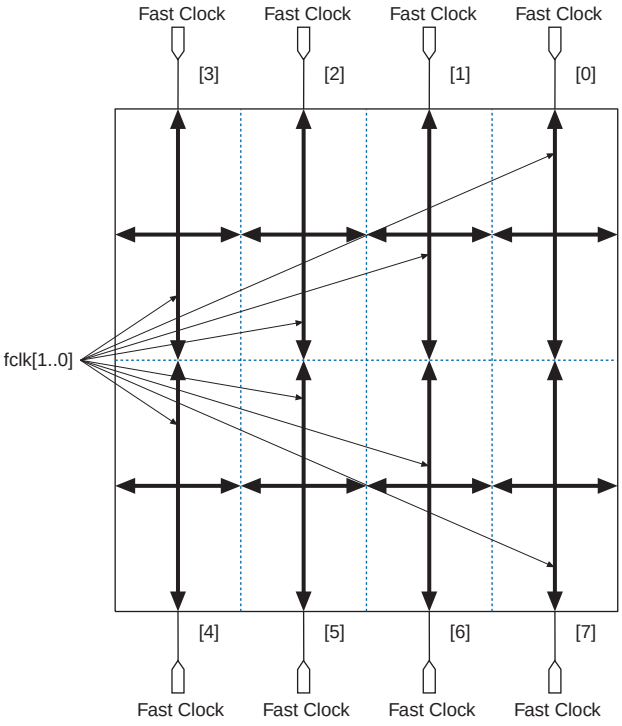


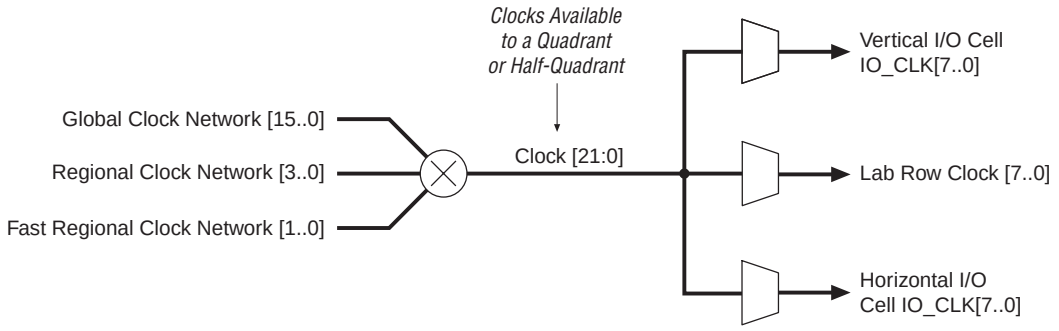
Figure 4–44. EP1SGX40 Device Fast Regional Clock Pin Connections to Fast Regional Clocks



Combined Resources

Within each region, there are 22 distinct dedicated clocking resources consisting of 16 global clock lines, 4 regional clock lines, and 2 fast regional clock lines. Multiplexers are used with these clocks to form 8-bit busses to drive LAB row clocks, column IOE clocks, or row IOE clocks. Another multiplexer at the LAB level selects two of the eight row clocks to feed the LE registers within the LAB. See Figure 4–45.

Figure 4–45. Regional Clock Bus



IOE clocks have horizontal and vertical block regions that are clocked by eight I/O clock signals chosen from the 22-quadrant or half-quadrant clock resources. Figures 4–46 and 4–47 show the quadrant and half-quadrant relationship to the I/O clock regions, respectively. The vertical regions (column pins) have less clock delay than the horizontal regions (row pins).

Figure 4–46. EP1SGX25 & EP1SGX10 Device I/O Clock Groups

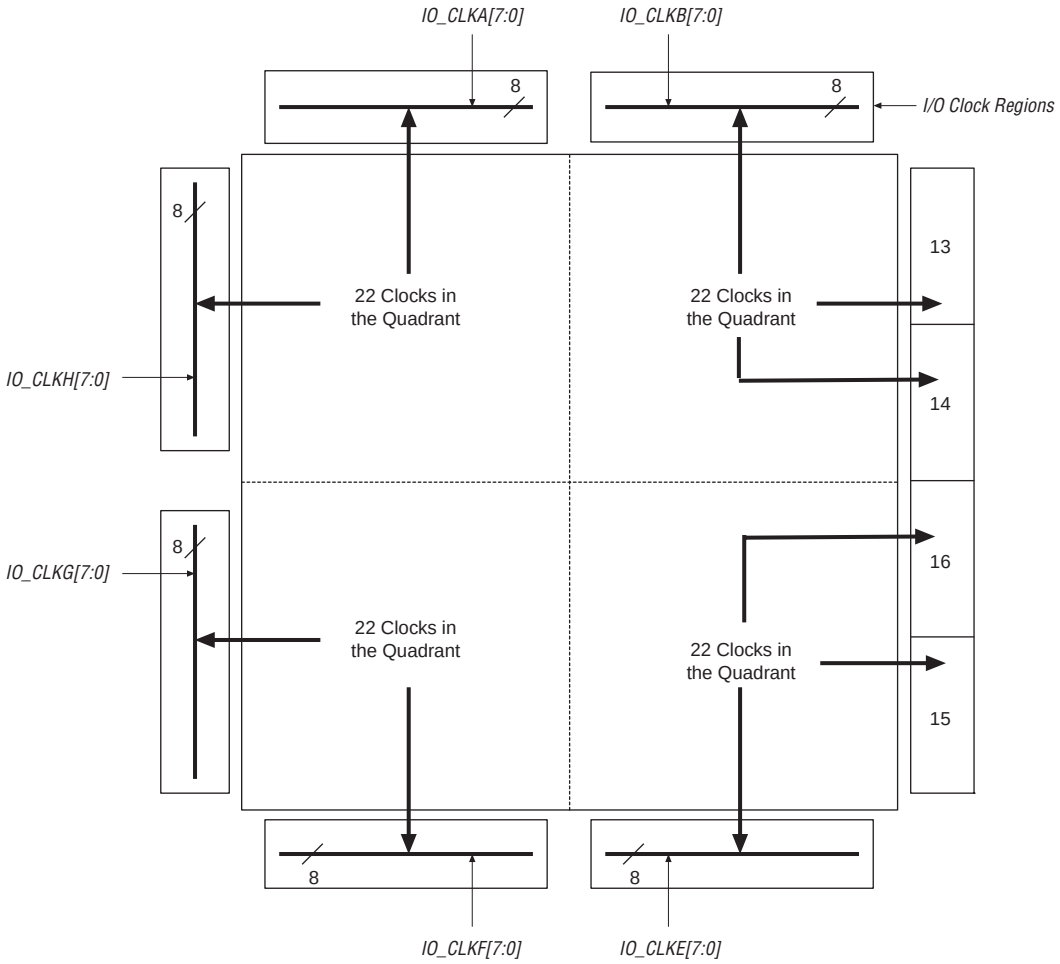
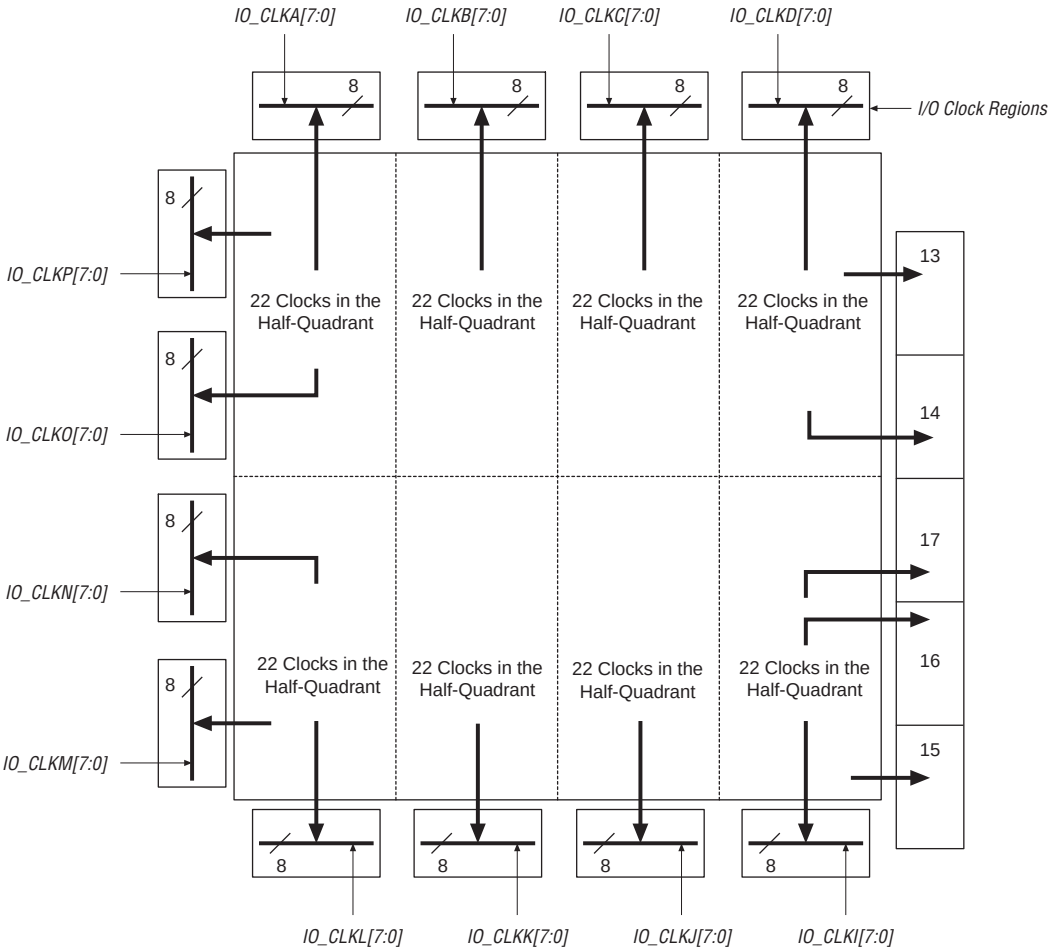


Figure 4–47. EP1SGX40 Device I/O Clock Groups



You can use the Quartus II software to control whether a clock input pin is either global, regional, or fast regional. The Quartus II software automatically selects the clocking resources if not specified.

Enhanced & Fast PLLs

Stratix GX devices provide robust clock management and synthesis using up to four enhanced PLLs and four fast PLLs. These PLLs increase performance and provide advanced clock interfacing and clock frequency synthesis. With features such as clock switchover, spread spectrum

clocking, programmable bandwidth, phase and delay control, and dynamic PLL reconfiguration, the Stratix GX device's enhanced PLLs provide you with complete control of your clocks and system timing. The fast PLLs provide general purpose clocking with multiplication and phase shifting as well as high-speed outputs for high-speed differential I/O support. Enhanced and fast PLLs work together with the Stratix GX high-speed I/O and advanced clock architecture to provide significant improvements in system performance and bandwidth.

The Quartus II software enables the PLLs and their features without requiring any external devices. Table 4–17 shows which PLLs are available for each Stratix GX device and their type. Table 4–18 shows the enhanced PLL and fast PLL features in Stratix GX devices.

Table 4–17. Stratix GX Device PLL Availability

Device	Fast PLLs								Enhanced PLLs			
	1	2	3 *(1)*	4 *(1)*	7	8	9 *(1)*	10 *(1)*	5 *(2)*	6 *(2)*	11 *(3)*	12 *(3)*
EP1SGX10	✓	✓							✓	✓		
EP1SGX25	✓	✓							✓	✓		
EP1SGX40	✓	✓			✓	✓			✓	✓	✓	✓

Notes to Table 4–17:

(1) PLLs 3, 4, 9, and 10 are not available in Stratix GX devices. However, these PLLs are listed in Table 4–17 because the Stratix GX PLL numbering scheme is consistent with Stratix devices.

(2) PLLs 5 and 6 each have eight single-ended outputs or four differential outputs.

(3) PLLs 11 and 12 each have one single-ended output.

Table 4–18. Stratix GX Enhanced PLL & Fast PLL Features (Part 1 of 2) *Notes (1)–(8)*

Feature	Enhanced PLL	Fast PLL
Clock multiplication and division	*m*/ (*n* × post-scale counter) *(1)*	*m*/(post-scale counter) *(2)*
Phase shift	Down to 156.25-ps increments *(3)*, *(4)*	Down to 125-ps increments *(3)*, *(4)*
Delay shift	250-ps increments for ±3 ns	
Clock switchover	✓	
PLL reconfiguration	✓	
Programmable bandwidth	✓	
Spread spectrum clocking	✓	
Programmable duty cycle	✓	✓
Number of internal clock outputs	6	3 *(5)*

Table 4–18. Stratix GX Enhanced PLL & Fast PLL Features (Part 2 of 2) Notes (1)–(8)		
Feature	**Enhanced PLL**	**Fast PLL**
Number of external clock outputs	Four differential/eight singled-ended or one single-ended *(6)*	*(7)*
Number of feedback clock inputs	4 *(8)*	

Notes to Table 4–18:

(1) The maximum count value is 1024, with a 50% duty cycle setting on the counter. The maximum count value for any other duty cycle setting is 512.
(2) For fast PLLs, *m* and post-scale counters range from 1 to 32.
(3) The smallest phase shift is determined by the VCO period divided by 8.
(4) For degree increments, Stratix GX devices can shift all output frequencies in increments of at least 45°. Smaller degree increments are possible depending on the frequency and divide parameters.
(5) PLLs 7 and 8 have two output ports per PLL. PLLs 1 and 2 have three output ports per PLL.
(6) Every Stratix GX device has two enhanced PLLs (PLLs 5 and 6) with eight single-ended or four differential outputs each. Two additional enhanced PLLs (PLLs 11 and 12) in EP1SGX40 devices each have one single-ended output.
(7) Fast PLLs can drive to any I/O pin as an external clock. For high-speed differential I/O pins, the device uses a data channel to generate `txclkout`.
(8) Every Stratix GX device has two enhanced PLLs with one single-ended or differential external feedback input per PLL.

Figure 4–48 shows a top-level diagram of the Stratix GX device and the PLL floorplan.

Figure 4–48. PLL Floorplan

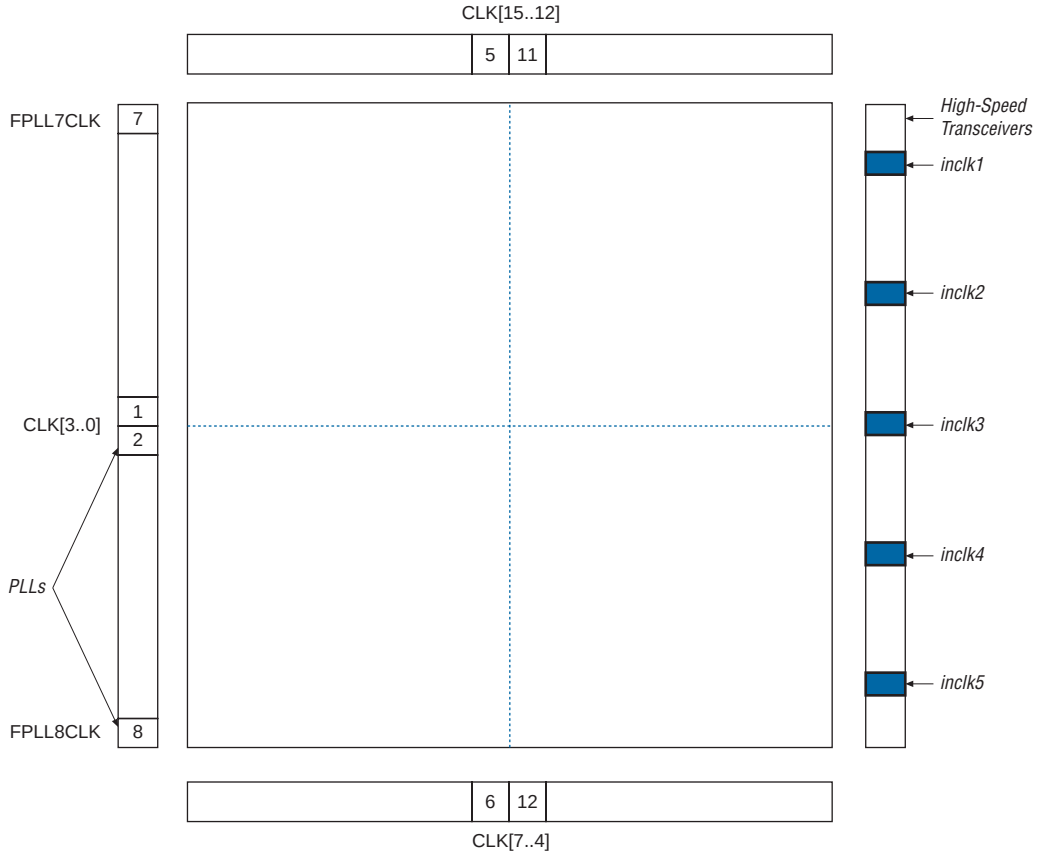
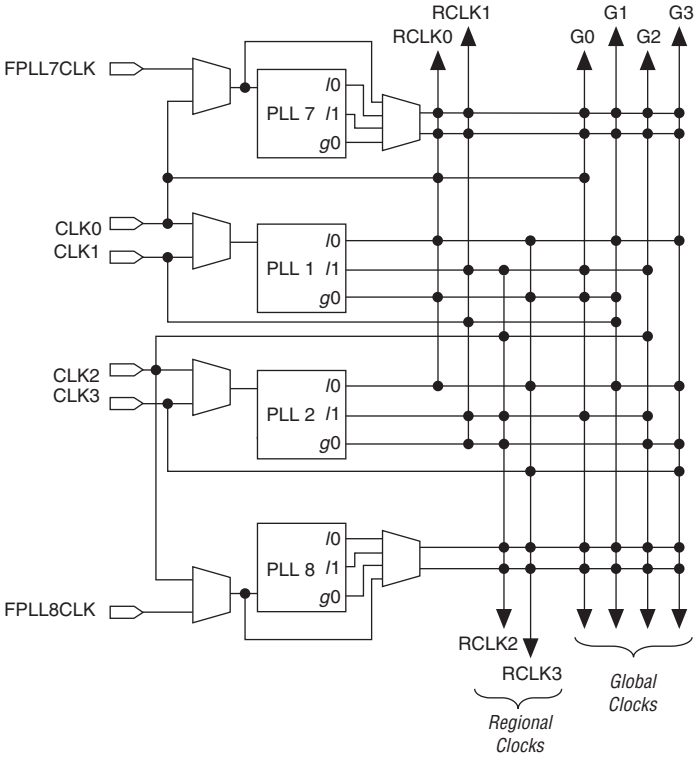


Figure 4–49 shows the global and regional clock connections from the PLL outputs and the CLK pins.

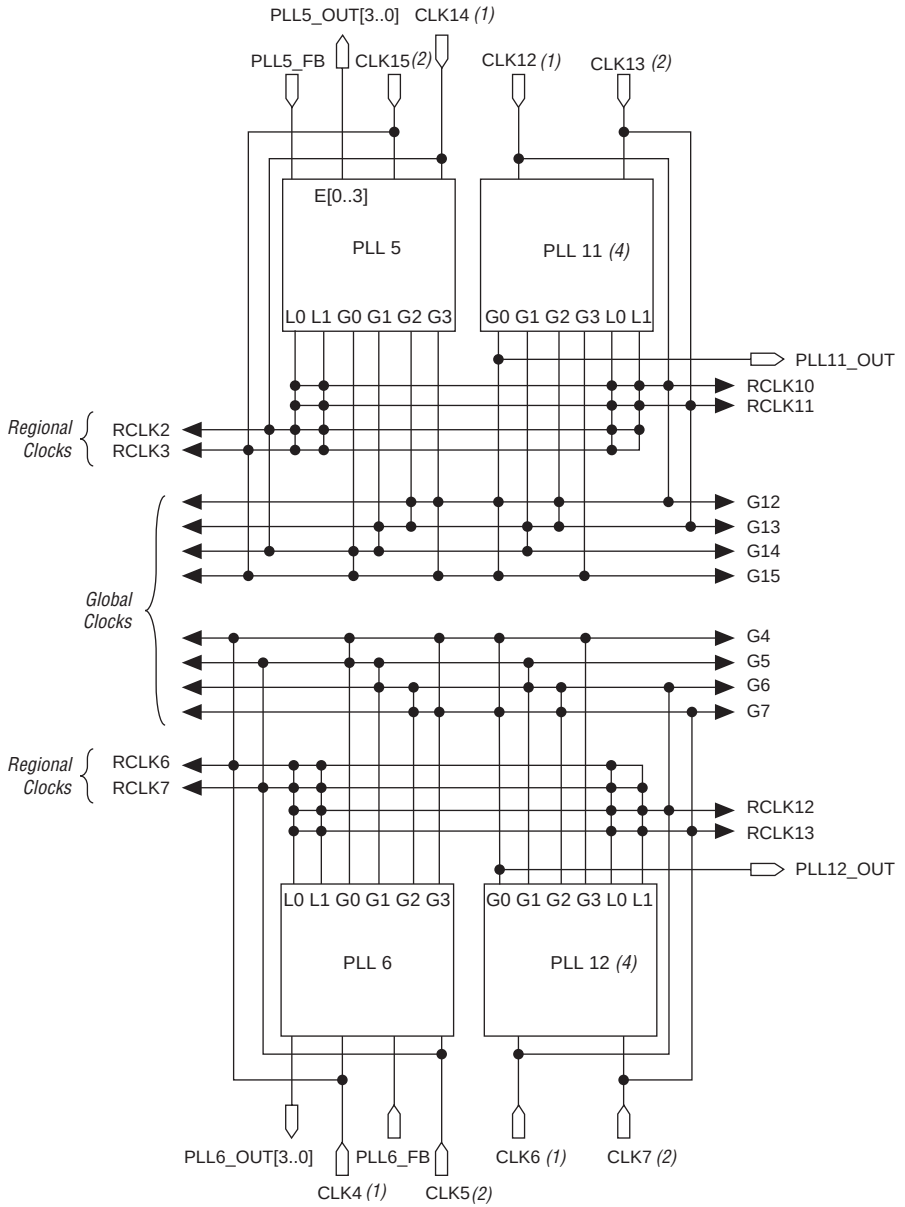
Figure 4–49. Global & Regional Clock Connections From Side Pins & Fast PLL Outputs *Note (1)*



Note to Figure 4–49:
(1) PLLs 1,2 7, and 8 are fast PLLs. PLLs 7 and 8 do not drive global clocks.

Figure 4–50 shows the global and regional clocking from enhanced PLL outputs and top CLK pins.

Figure 4–50. Global & Regional Clock Connections From Top Clock Pins & Enhanced PLL Outputs *Note (1)*

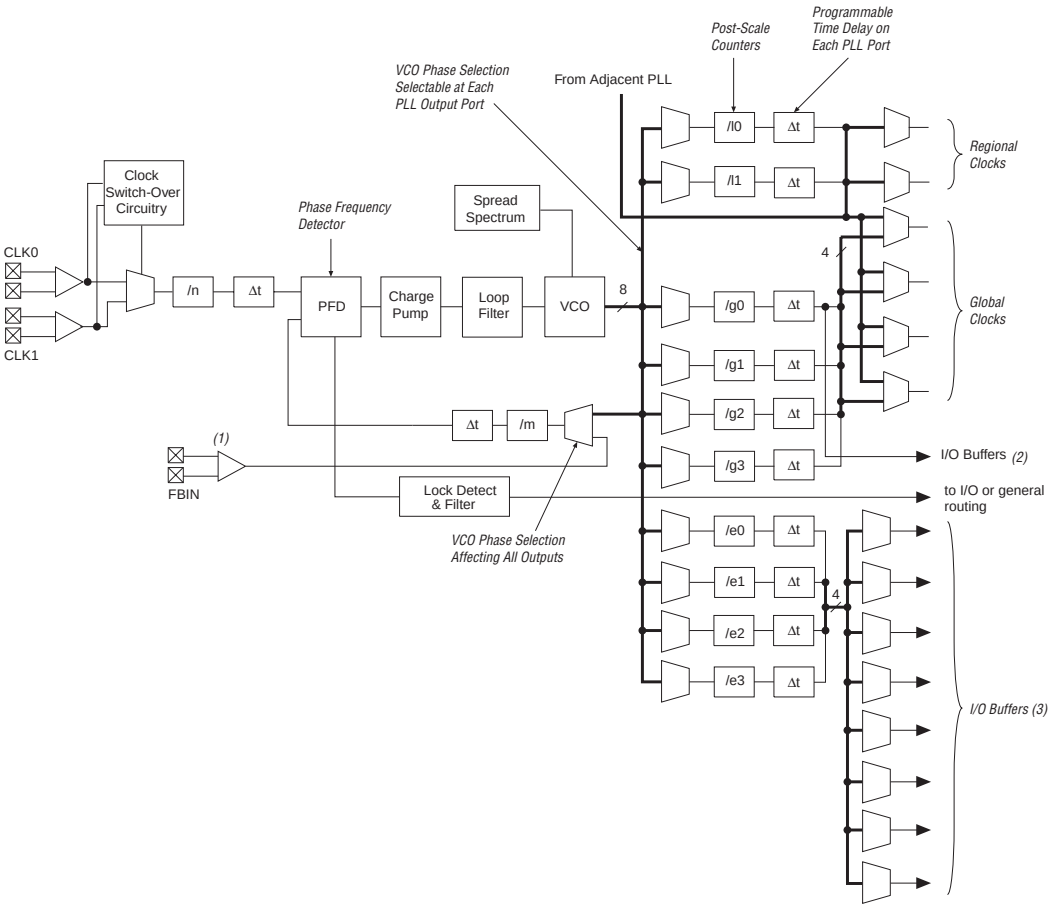


Note to Figure 4–50:
(1) PLLs 5, 6, 11, and 12 are enhanced PLLs.

Enhanced PLLs

Stratix GX devices contain up to four enhanced PLLs with advanced clock management features. Figure 4–51 shows a diagram of the enhanced PLL.

Figure 4–51. Stratix GX Enhanced PLL



Notes to Figure 4–51:

(1) External feedback is available in PLLs 5 and 6.

(2) This external output is available from the *g*0 counter for PLLs 11 and 12.

(3) These counters and external outputs are available in PLLs 5 and 6.

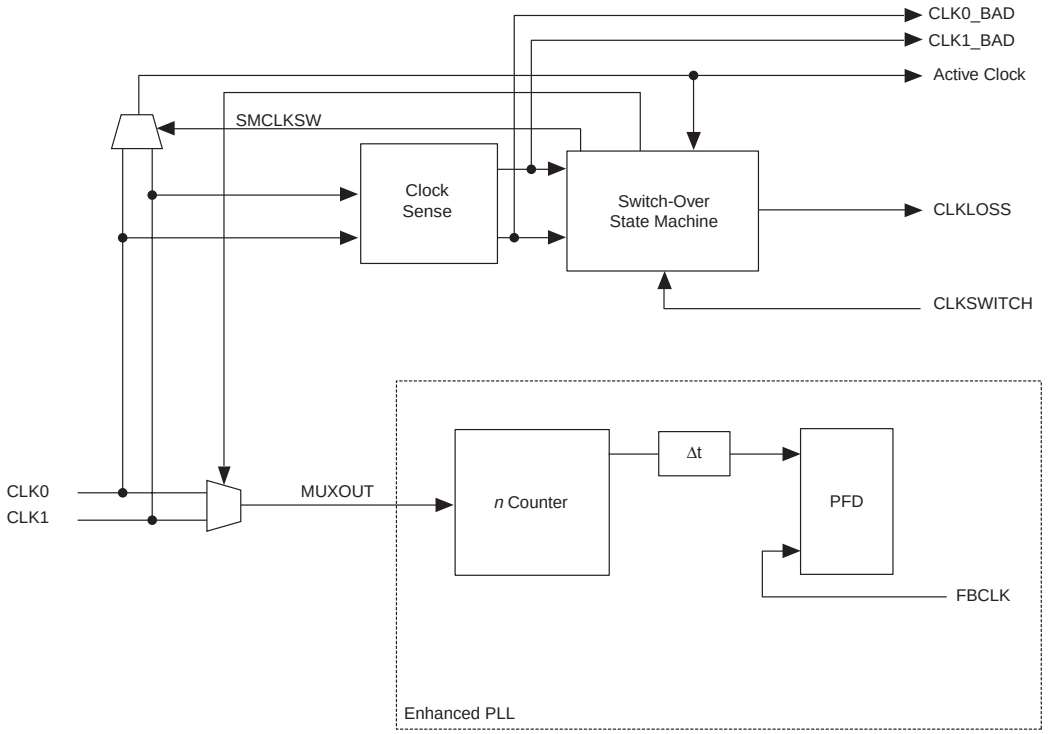
Clock Multiplication & Division

Each Stratix GX device enhanced PLL provides clock synthesis for PLL output ports using $m/(n \times$ post-scale counter) scaling factors. The input clock is divided by a pre-scale divider, n, and is then multiplied by the m feedback factor. The control loop drives the VCO to match $f_{IN} \times (m/n)$. Each output port has a unique post-scale counter that divides down the high-frequency VCO. For multiple PLL outputs with different frequencies, the VCO is set to the least common multiple of the output frequencies that meets its frequency specifications. Then, the post-scale dividers scale down the output frequency for each output port. For example, if output frequencies required from one PLL are 33 and 66 MHz, set the VCO to 330 MHz (the least common multiple in the VCO's range). There is one pre-scale divider, n, and one multiply divider, m, per PLL, with a range of 1 to 512 on each. There are two post-scale dividers (l) for regional clock output ports, four counters (g) for global clock output ports, and up to four counters (e) for external clock outputs, all ranging from 1 to 512. The Quartus II software automatically chooses the appropriate scaling factors according to the input frequency, multiplication, and division values entered.

Clock Switchover

To effectively develop high-reliability network systems, clocking schemes must support multiple clocks to provide redundancy. For this reason, Stratix GX device enhanced PLLs support a flexible clock switchover capability. Figure 4–52 shows a block diagram of the switchover circuit.The switchover circuit is configurable, so you can define how to implement it. Clock-sense circuitry automatically switches from the primary to secondary clock for PLL reference when the primary clock signal is not present.

Figure 4–52. Clock Switchover Circuitry



Note to Figure 4–52:
(1) PFD: phase frequency detector.

There are two possible ways to use the clock switchover feature.

- You can use automatic switchover circuitry for switching between inputs of the same frequency. For example, in applications that require a redundant clock with the same frequency as the primary clock, the switchover state machine generates a signal that controls the multiplexer select input on the bottom of Figure 4–52. In this case, the secondary clock becomes the reference clock for the PLL.
- You can use the `clkswitch` input for user- or system-controlled switch conditions. This is possible for same-frequency switchover or to switch between inputs of different frequencies. For example, if `inclk0` is 66 MHz and `inclk1` is 100 MHz, you must control the switchover because the automatic clock-sense circuitry cannot monitor primary and secondary clock frequencies with a frequency difference of more than ±20%. This feature is useful when clock sources can originate from multiple cards on the backplane,

requiring a system-controlled switchover between frequencies of operation. You can use `clkswitch` together with the lock signal to trigger the switch from a clock that is running but becomes unstable and cannot be locked onto.

During switchover, the PLL VCO continues to run and either slows down or speeds up, generating frequency drift on the PLL outputs. The clock switchover transitions without any glitches. After the switch, there is a finite resynchronization period to lock onto new clock as the VCO ramps up. The exact amount of time it takes for the PLL to relock relates to the PLL configuration and may be adjusted by using the programmable bandwidth feature of the PLL. The preliminary specification for the maximum time to relock is 100 µs.



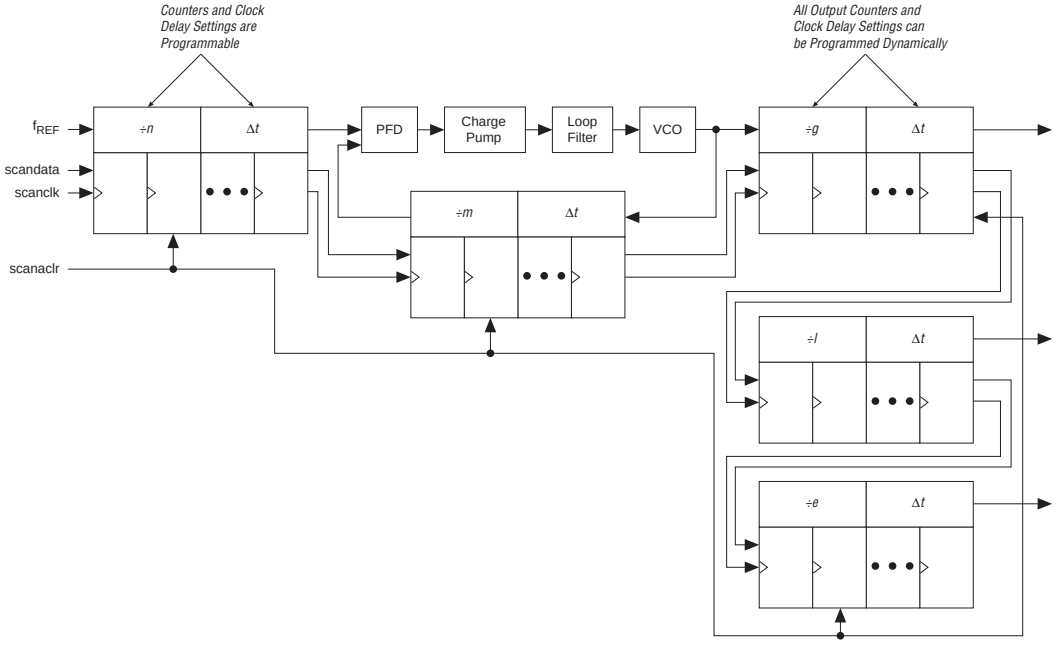
For more information on clock switchover, see *AN313: Implementing Clock Switchover in Stratix & Stratix GX Devices*.

PLL Reconfiguration

The PLL reconfiguration feature enables system logic to change Stratix GX device enhanced PLL counters and delay elements without reloading a Programmer Object File (**.pof**). This provides considerable flexibility for frequency synthesis, allowing real-time PLL frequency and output clock delay variation. You can sweep the PLL output frequencies and clock delay in prototype environments. The PLL reconfiguration feature can also dynamically or intelligently control system clock speeds or t_{CO} delays in end systems.

Clock delay elements at each PLL output port implement variable delay. Figure 4–53 shows a diagram of the overall dynamic PLL control feature for the counters and the clock delay elements. The configuration time is less than 20 µs for the enhanced PLL using a input shift clock rate of 25 MHz. The charge pump, loop filter components, and phase shifting using VCO phase taps cannot be dynamically adjusted.

Figure 4–53. Dynamically Programmable Counters & Delays in Stratix GX Device Enhanced PLLs



PLL reconfiguration data is shifted into serial registers from the logic array or external devices. The PLL input shift data uses a reference input shift clock. Once the last bit of the serial chain is clocked in, the register chain is synchronously loaded into the PLL configuration bits. The shift circuitry also provides an asynchronous clear for the serial registers.

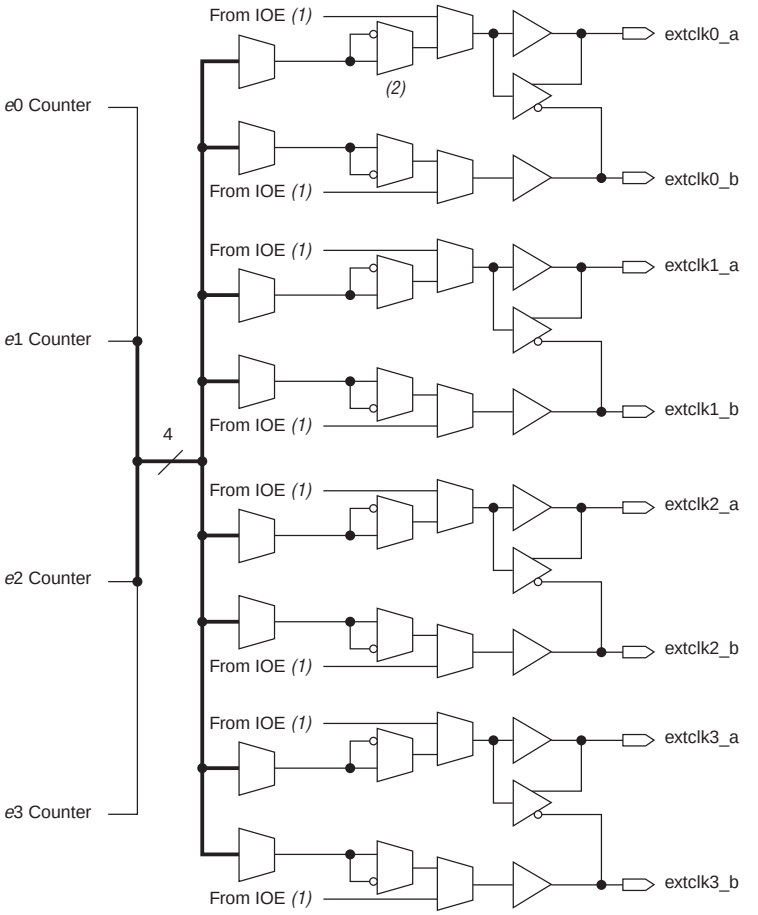
Programmable Bandwidth

You have advanced control of the PLL bandwidth using the programmable control of the PLL loop characteristics, including loop filter and charge pump. The PLL's bandwidth is a measure of its ability to track the input clock and jitter. A high-bandwidth PLL can quickly lock onto a reference clock and react to any changes in the clock. It also allows a wide band of input jitter spectrum to pass to the output. A low-bandwidth PLL takes longer to lock, but it attenuates all high-frequency jitter components. The Quartus II software can adjust PLL characteristics to achieve the desired bandwidth. The programmable bandwidth is tuned by varying the charge pump current, loop filter resistor value, high frequency capacitor value, and *m* counter value. You can manually adjust these values if desired. Bandwidth is programmable from 150 kHz to 2 MHz.

External Clock Outputs

Enhanced PLLs 5 and 6 each support up to eight single-ended clock outputs (or four differential pairs). See Figure 4–54.

Figure 4–54. External Clock Outputs for PLLs 5 & 6



Notes to Figure 4–54:

(1) Each external clock output pin can be used as a general purpose output pin from the logic array. These pins are multiplexed with IOE outputs.

(2) Two single-ended outputs are possible per output counter—either two outputs of the same frequency and phase or one shifted 180°.

Any of the four external output counters can drive the single-ended or differential clock outputs for PLLs 5 and 6. This means one counter or frequency can drive all output pins available from PLL 5 or PLL 6. Each

pair of output pins (four pins total) has dedicated VCC and GND pins to reduce the output clock's overall jitter by providing improved isolation from switching I/O pins.

For PLLs 5 and 6, each pin of a single-ended output pair can either be in phase or 180° out of phase. The clock output pin pairs support the same I/O standards as standard output pins (in the top and bottom banks) as well as LVDS, LVPECL, 3.3-V PCML, HyperTransport technology, differential HSTL, and differential SSTL. Table 4–19 shows which I/O standards the enhanced PLL clock pins support. When in single-ended or differential mode, the two outputs operate off the same power supply. Both outputs use the same standards in single-ended mode to maintain performance. You can also use the external clock output pins as user output pins if external enhanced PLL clocking is not needed.

Table 4–19. I/O Standards Supported for Enhanced PLL Pins (Part 1 of 2)

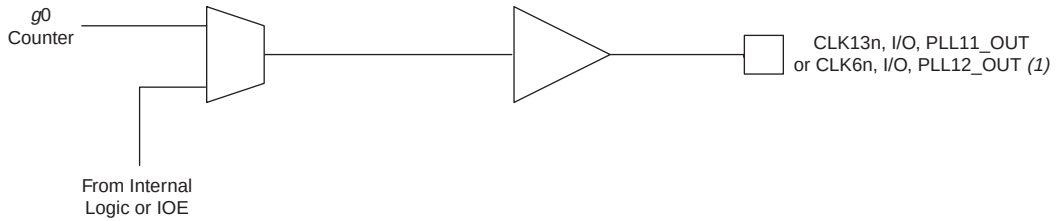
I/O Standard	Input			Output
	INCLK	FBIN	PLLENABLE	EXTCLK
LVTTL	✓	✓	✓	✓
LVCMOS	✓	✓	✓	✓
2.5 V	✓	✓		✓
1.8 V	✓	✓		✓
1.5 V	✓	✓		✓
3.3-V PCI	✓	✓		✓
3.3-V PCI-X	✓	✓		✓
LVPECL	✓	✓		✓
3.3-V PCML	✓	✓		✓
LVDS	✓	✓		✓
HyperTransport technology	✓	✓		✓
Differential HSTL	✓			✓
Differential SSTL				✓
3.3-V GTL	✓	✓		✓
3.3-V GTL+	✓	✓		✓
1.5-V HSTL class I	✓	✓		✓
1.5-V HSTL class II	✓	✓		✓
SSTL-18 class I	✓	✓		✓
SSTL-18 class II	✓	✓		✓
SSTL-2 class I	✓	✓		✓
SSTL-2 class II	✓	✓		✓

Table 4–19. I/O Standards Supported for Enhanced PLL Pins (Part 2 of 2)

I/O Standard	Input			Output
	INCLK	FBIN	PLLENABLE	EXTCLK
SSTL-3 class I	✓	✓		✓
SSTL-3 class II	✓	✓		✓
AGP (1× and 2×)	✓	✓		✓
CTT	✓	✓		✓

Enhanced PLLs 11 and 12 support one single-ended output each (see Figure 4–55). These outputs do not have their own VCC and GND signals. Therefore, to minimize jitter, do not place switching I/O pins next to this output pin.

Figure 4–55. External Clock Outputs for Enhanced PLLs 11 & 12



Note to Figure 4–55:
(1) For PLL 11, this pin is CLK13n; for PLL 12 this pin is CLK7n.

Stratix GX devices can drive any enhanced PLL driven through the global clock or regional clock network to any general I/O pin as an external output clock. The jitter on the output clock is not guaranteed for these cases.

Clock Feedback

The following four feedback modes in Stratix GX device enhanced PLLs allow multiplication and/or phase and delay shifting:

- Zero delay buffer: The external clock output pin is phase-aligned with the clock input pin for zero delay.
- External feedback: The external feedback input pin, FBIN, is phase-aligned with the clock input, CLK, pin. Aligning these clocks allows you to remove clock delay and skew between devices. This mode is only possible for PLLs 5 and 6. PLLs 5 and 6 each support

feedback for one of the dedicated external outputs, either one single-ended or one differential pair. In this mode, one *e* counter feeds back to the PLL `FBIN` input, becoming part of the feedback loop.

- Normal mode: If an internal clock is used in this mode, it is phase-aligned to the input clock pin. The external clock output pin has a phase delay relative to the clock input pin if connected in this mode. You define which internal clock output from the PLL should be phase-aligned to the internal clock pin.
- No compensation: In this mode, the PLL does not compensate for any clock networks or external clock outputs.

Phase & Delay Shifting

Stratix GX device enhanced PLLs provide advanced programmable phase and clock delay shifting. For phase shifting, you can specify a phase shift (in degrees or time units) for each PLL clock output port or for all outputs together in one shift. Phase-shifting values in time units are allowed with a resolution range of 160 to 420 ps. This resolution is a function of frequency input and the multiplication and division factors. In other words, it is a function of the VCO period equal to one-eighth of the VCO period. Each clock output counter can choose a different phase of the VCO period from up to eight taps. You can use this clock output counter along with an initial setting on the post-scale counter to achieve a phase-shift range for the entire period of the output clock. The phase tap feedback to the *m* counter can shift all outputs to a single phase or delay. The Quartus II software automatically sets the phase taps and counter settings according to the phase shift entered.

In addition to the phase-shift feature, the fine tune clock delay shift feature provides advanced time delay shift control on each of the four PLL outputs. Each PLL output shifts in 250-ps increments for a range of –3.0 ns to +3.0 ns between any two outputs using discrete delay elements. Total delay shift between any two PLL outputs must be less than 3 ns. For example, shifts on outputs of –1 and +2 ns is allowed, but not –1 and +2.5 ns. There is some delay variation due to process, voltage, and temperature. Only the clock delay shift blocks can be controlled during system operation for dynamic clock delay control.

Spread-Spectrum Clocking

The Stratix GX device's enhanced PLLs use spread-spectrum technology to reduce electromagnetic interference generation from a system by distributing the energy over a broader frequency range. The enhanced

PLL typically provides 0.5% down spread modulation using a triangular profile. The modulation frequency is programmable. Enabling spread spectrum for a PLL affects all of its outputs.

Lock Detect

The lock output indicates that there is a stable clock output signal in phase with the reference clock. Without any additional circuitry, the lock signal may toggle as the PLL begins tracking the reference clock. You may need to gate the lock signal for use as a system control. The lock signal from the locked port can drive the logic array or an output pin.

Whenever the PLL loses lock for any reason (be it excessive inclk jitter, clock switchover, PLL reconfiguration, power supply noise etc.), the PLL must be reset with the `areset` signal for correct phase shift operation. If the phase relationship between the input clock versus output clock, and between different output clocks from the PLL is not important in the design, then the PLL need not be reset.



See the *Stratix GX FPGA Errata Sheet* for more information on implementing the gated lock signal in the design.

Programmable Duty Cycle

The programmable duty cycle allows enhanced PLLs to generate clock outputs with a variable duty cycle. This feature is supported on each enhanced PLL post-scale counter (*g*0..*g*3, *l*0..*l*3, *e*0..*e*3). The duty cycle setting is achieved by a low and high time count setting for the post-scale dividers. The Quartus II software uses the frequency input and the required multiply or divide rate to determine the duty cycle choices.

Advanced Clear & Enable Control

There are several control signals for clearing and enabling PLLs and their outputs. You can use these signals to control PLL resynchronization and gate PLL output clocks for low-power applications.

The `pllenable` pin is a dedicated pin that enables/disables PLLs. When the `pllenable` pin is low, the clock output ports are driven by `GND` and all the PLLs go out of lock. When the `pllenable` pin goes high again, the PLLs relock and resynchronize to the input clocks. You can choose which PLLs are controlled by the `pllenable` signal by connecting the `pllenable` input port of the `altpll` megafunction to the common `pllenable` input pin.

The areset signals are reset/resynchronization inputs for each PLL. The areset signal should be asserted every time the PLL loses lock to guarantee correct phase relationship between the PLL output clocks. Users should include the areset signal in designs if any of the following conditions are true:

- PLL Reconfiguration or Clock switchover enables in the design.
- Phase relationships between output clocks need to be maintained after a loss of lock condition

The device input pins or logic elements (LEs) can drive these input signals. When driven high, the PLL counters resets, clearing the PLL output and placing the PLL out of lock. The VCO sets back to its nominal setting (~700 MHz). When driven low again, the PLL resynchronizes to its input as it relocks. If the target VCO frequency is below this nominal frequency, then the output frequency starts at a higher value than desired as the PLL locks. If the system cannot tolerate this, the clkena signal can disable the output clocks until the PLL locks.

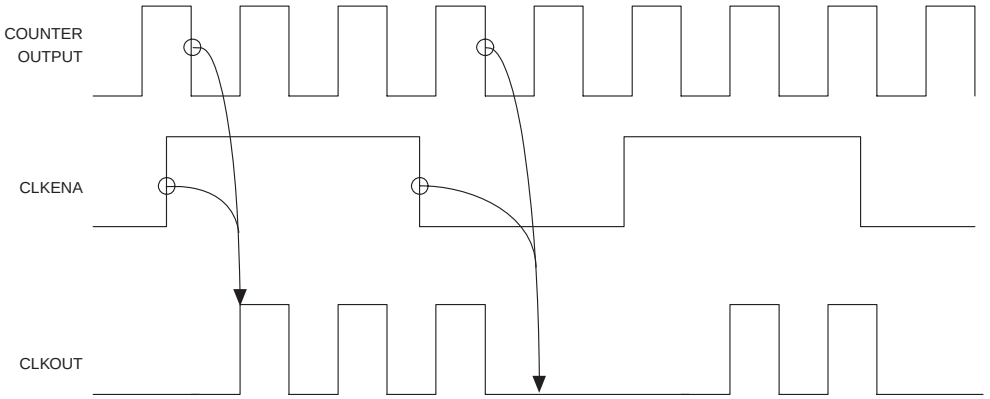
The pfdena signals control the phase frequency detector (PFD) output with a programmable gate. If you disable the PFD, the VCO operates at its last set value of control voltage and frequency with some long-term drift to a lower frequency. The system continues running when the PLL goes out of lock or the input clock is disabled. By maintaining the last locked frequency, the system has time to store its current settings before shutting down. You can either use your own control signal or a clkloss status signal to trigger pfdena.

The clkena signals control the enhanced PLL regional and global outputs. Each regional and global output port has its own clkena signal. The clkena signals synchronously disable or enable the clock at the PLL output port by gating the outputs of the *g* and *l* counters. The clkena signals are registered on the falling edge of the counter output clock to enable or disable the clock without glitches. Figure 4–56 shows the waveform example for a PLL clock port enable. The PLL can remain locked independent of the clkena signals since the loop-related counters are not affected. This feature is useful for applications that require a low power or sleep mode. Upon re-enabling, the PLL does not need a resynchronization or relock period. The clkena signal can also disable clock outputs if the system is not tolerant to frequency overshoot during resynchronization.

The extclkena signals work in the same way as the clkena signals, but they control the external clock output counters (*e*0, *e*1, *e*2, and *e*3). Upon re-enabling, the PLL does not need a resynchronization or relock period

unless the PLL is using external feedback mode. In order to lock in external feedback mode, the external output must drive the board trace back to the FBIN pin.

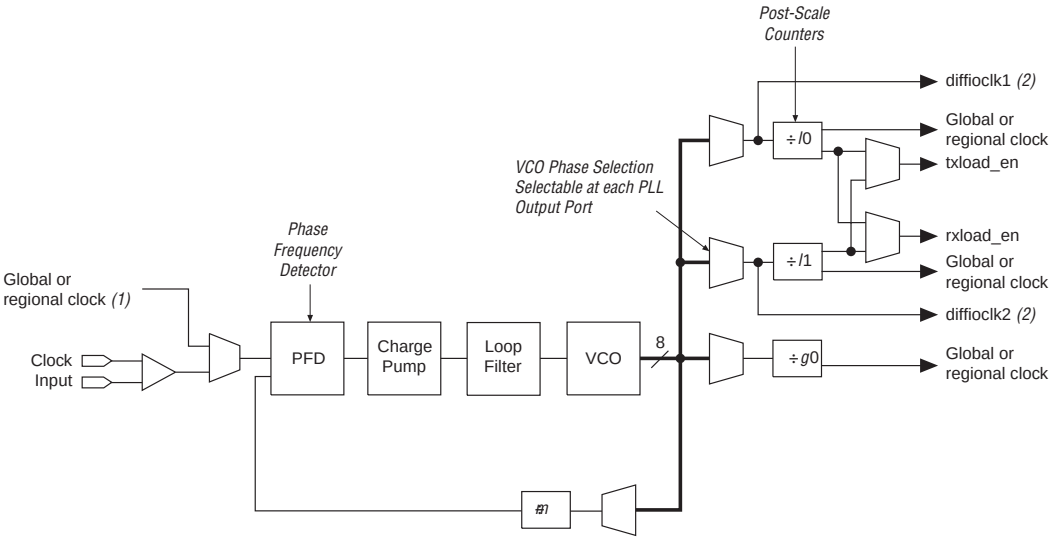
Figure 4–56. extclkena Signals



Fast PLLs

Stratix GX devices contain up to four fast PLLs with high-speed serial interfacing ability, along with general-purpose features. Figure 4–57 shows a diagram of the fast PLL.

Figure 4–57. Stratix GX Device Fast PLL



Notes to Figure 4–57:

(1) In high-speed differential I/O support mode, this high-speed PLL clock feeds the SERDES. Stratix GX devices only support one rate of data transfer per fast PLL in high-speed differential I/O support mode.

(2) This signal is a high-speed differential I/O support SERDES control signal.

Clock Multiplication & Division

The Stratix GX device's fast PLLs provide clock synthesis for PLL output ports using *m*/(post scaler) scaling factors. The input clock is multiplied by the *m* feedback factor. Each output port has a unique post scale counter to divide down the high-frequency VCO. There is one multiply divider, *m*, per fast PLL with a range of 1 to 32. There are two post scale L dividers for regional and/or LVDS interface clocks, and *g*0 counter for global clock output port; all range from 1 to 32.

In the case of a high-speed differential interface, you can set the output counter to 1 to allow the high-speed VCO frequency to drive the SERDES.

External Clock Outputs

Each fast PLL supports differential or single-ended outputs for source-synchronous transmitters or for general-purpose external clocks. There are no dedicated external clock output pins. Any I/O pin can be driven by the fast PLL global or regional outputs as an external output

pin. The I/O standards supported by any particular bank determines what standards are possible for an external clock output driven by the fast PLL in that bank.

Table 4–20 shows the I/O standards supported by fast PLL input pins.

Table 4–20. Fast PLL Port Input Pin I/O Standards

I/O Standard	Input	
	INCLK	PLLENABLE
LVTTL	✓	✓
LVCMOS	✓	✓
2.5 V	✓	
1.8 V	✓	
1.5 V	✓	
3.3-V PCI		
3.3-V PCI-X		
LVPECL	✓	
3.3-V PCML	✓	
LVDS	✓	
HyperTransport technology	✓	
Differential HSTL	✓	
Differential SSTL		
3.3-V GTL	✓	
3.3-V GTL+	✓	
1.5V HSTL class I	✓	
1.5V HSTL class II	✓	
SSTL-18 class I	✓	
SSTL-18 class II	✓	
SSTL-2 class I	✓	
SSTL-2 class II	✓	
SSTL-3 class I	✓	
SSTL-3 class II	✓	
AGP (1× and 2×)	✓	
CTT	✓	

Phase Shifting

Stratix GX device fast PLLs have advanced clock shift capability that enables programmable phase shifts. You can enter a phase shift (in degrees or time units) for each PLL clock output port or for all outputs together in one shift. You can perform phase shifting in time units with a resolution range of 150 to 400 ps. This resolution is a function of the VCO period.

Control Signals

The fast PLL has the same `lock` output, `pllenable` input, and `areset` input control signals as the enhanced PLL.

For more information on high-speed differential I/O support, see the *High-Speed Source-Synchronous Differential I/O Interfaces in Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2*.

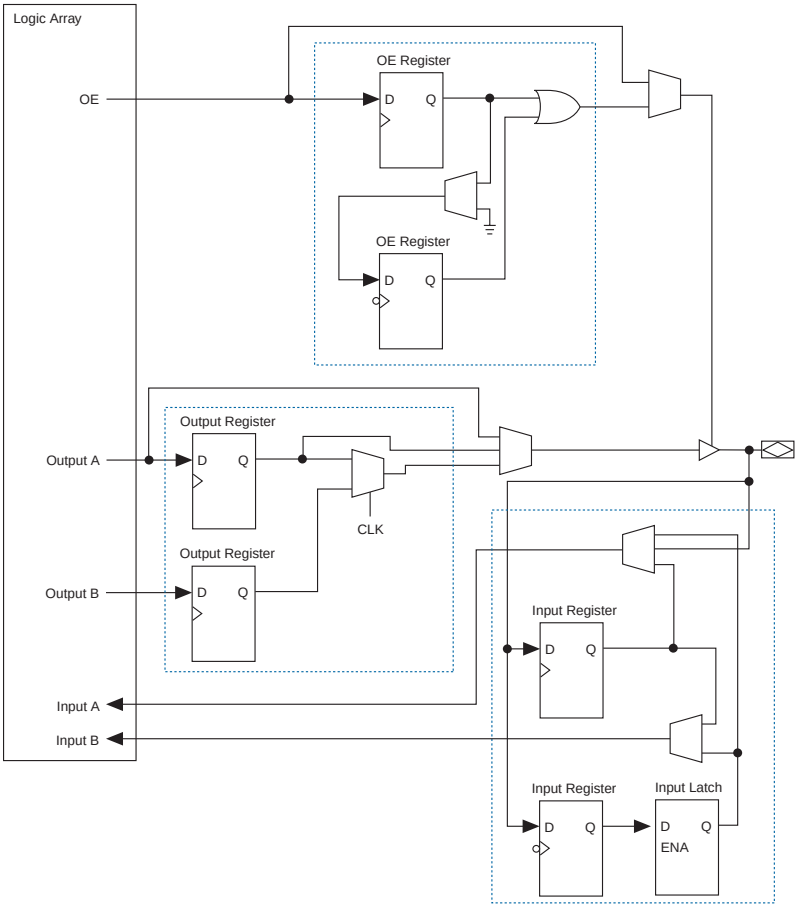
I/O Structure

IOEs provide many features, including:

- Dedicated differential and single-ended I/O buffers
- 3.3-V, 64-bit, 66-MHz PCI compliance
- 3.3-V, 64-bit, 133-MHz PCI-X 1.0 compliance
- Joint Test Action Group (JTAG) boundary-scan test (BST) support
- Differential on-chip termination for LVDS I/O standard
- Programmable pull-up during configuration
- Output drive strength control
- Slew-rate control
- Tri-state buffers
- Bus-hold circuitry
- Programmable pull-up resistors
- Programmable input and output delays
- Open-drain outputs
- DQ and DQS I/O pins
- Double-data rate (DDR) Registers

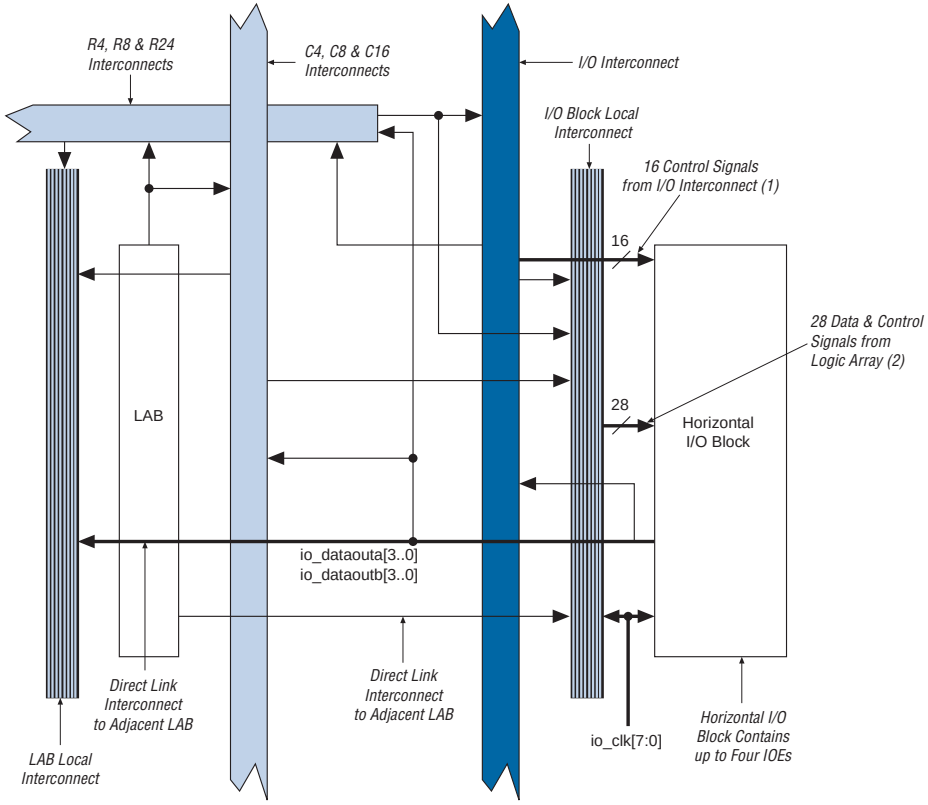
The IOE in Stratix GX devices contains a bidirectional I/O buffer, six registers, and a latch for a complete embedded bidirectional single data rate or DDR transfer. Figure 4–58 shows the Stratix GX IOE structure. The IOE contains two input registers (plus a latch), two output registers, and two output enable registers. The design can use both input registers and the latch to capture DDR input and both output registers to drive DDR outputs. Additionally, the design can use the output enable (OE) register for fast clock-to-output enable timing. The negative edge-clocked OE register is used for DDR SDRAM interfacing. The Quartus II software automatically duplicates a single OE register that controls multiple output or bidirectional pins.

Figure 4–58. Stratix GX IOE Structure



The IOEs are located in I/O blocks around the periphery of the Stratix GX device. There are up to four IOEs per row I/O block and six IOEs per column I/O block. The row I/O blocks drive row, column, or direct link interconnects. The column I/O blocks drive column interconnects. Figure 4–59 shows how a row I/O block connects to the logic array. Figure 4–60 shows how a column I/O block connects to the logic array.

Figure 4–59. Row I/O Block Connection to the Interconnect

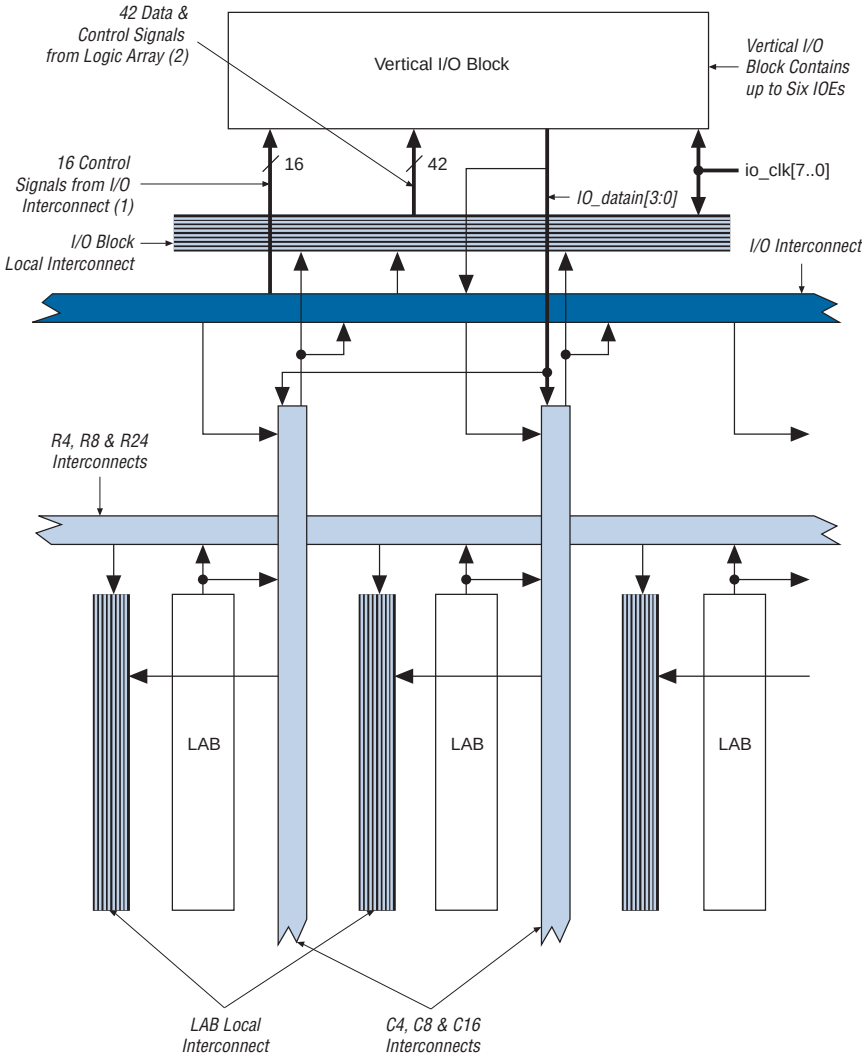


Notes to Figure 4–59:

(1) The 16 control signals are composed of four output enables `io_boe[3..0]`, four clock enables `io_bce[3..0]`, four clocks `io_clk[3..0]`, and four clear signals `io_bclr[3..0]`.

(2) The 28 data and control signals consist of eight data out lines: four lines each for DDR applications `io_dataouta[3..0]` and `io_dataoutb[3..0]`, four output enables `io_coe[3..0]`, four input clock enables `io_cce_in[3..0]`, four output clock enables `io_cce_out[3..0]`, four clocks `io_cclk[3..0]`, and four clear signals `io_cclr[3..0]`.

Figure 4–60. Column I/O Block Connection to the Interconnect



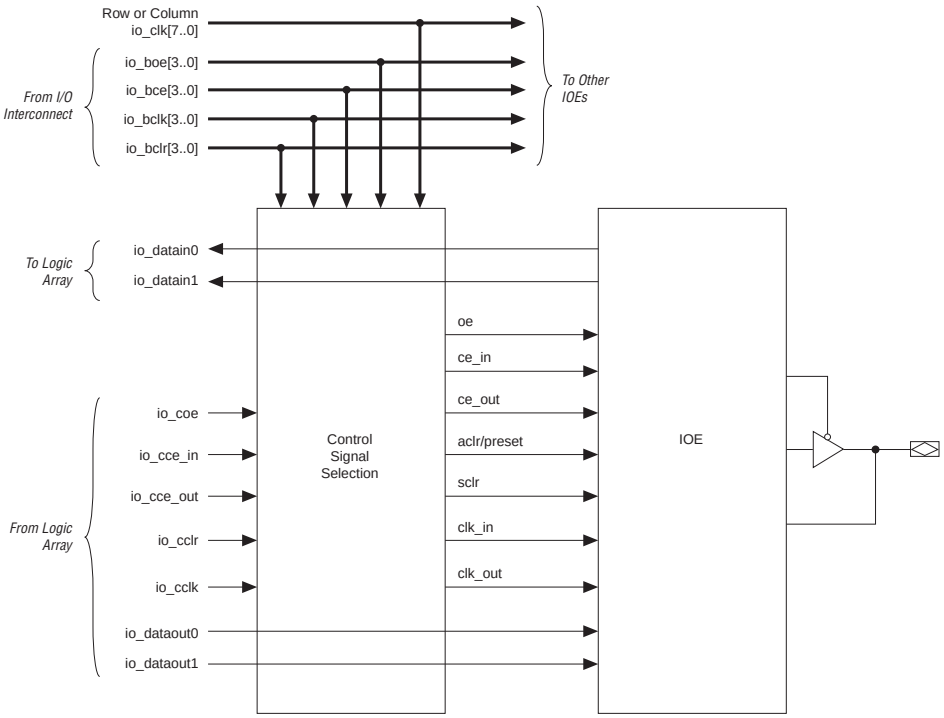
Notes to Figure 4–60:

(1) The 16 control signals are composed of four output enables `io_boe[3..0]`, four clock enables `io_bce[3..0]`, four clocks `io_bclk[3..0]`, and four clear signals `io_bclr[3..0]`.

(2) The 42 data and control signals consist of 12 data out lines; six lines each for DDR applications `io_dataouta[5..0]` and `io_dataoutb[5..0]`, six output enables `io_coe[5..0]`, six input clock enables `io_cce_in[5..0]`, six output clock enables `io_cce_out[5..0]`, six clocks `io_cclk[5..0]`, and six clear signals `io_cclr[5..0]`.

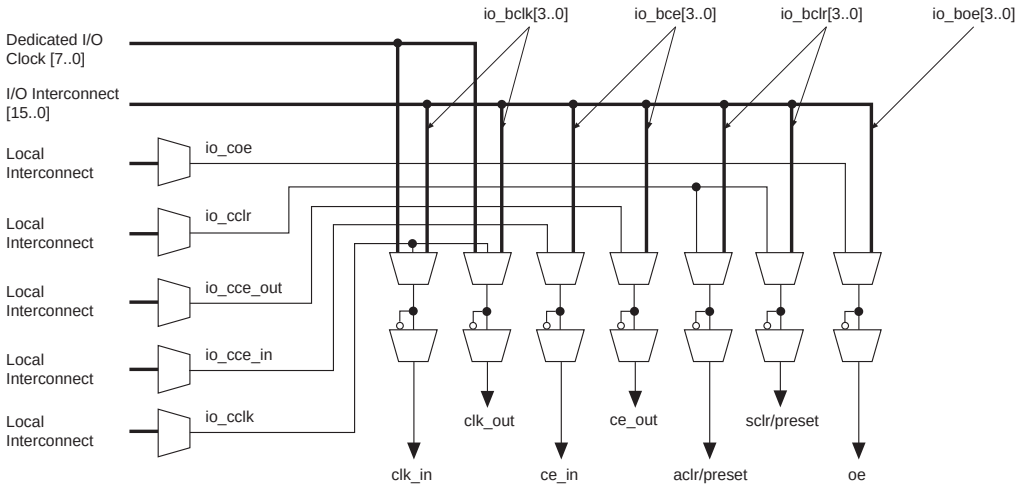
Stratix GX devices have an I/O interconnect similar to the R4 and C4 interconnect to drive high-fanout signals to and from the I/O blocks. There are 16 signals that drive into the I/O blocks composed of four output enables `io_boe[3..0]`, four clock enables `io_bce[3..0]`, four clocks `io_bclk[3..0]`, and four clear signals `io_bclr[3..0]`. The pin's `datain` signals can drive the IO interconnect, which in turn drives the logic array or other I/O blocks. In addition, the control and data signals can be driven from the logic array, providing a slower but more flexible routing resource. The row or column IOE clocks, `io_clk[7..0]`, provide a dedicated routing resource for low-skew, high-speed clocks. I/O clocks are generated from regional, global, or fast regional clocks (see "PLLs & Clock Networks" on page 4–68). Figure 4–61 illustrates the signal paths through the I/O block.

Figure 4–61. Signal Path Through the I/O Block



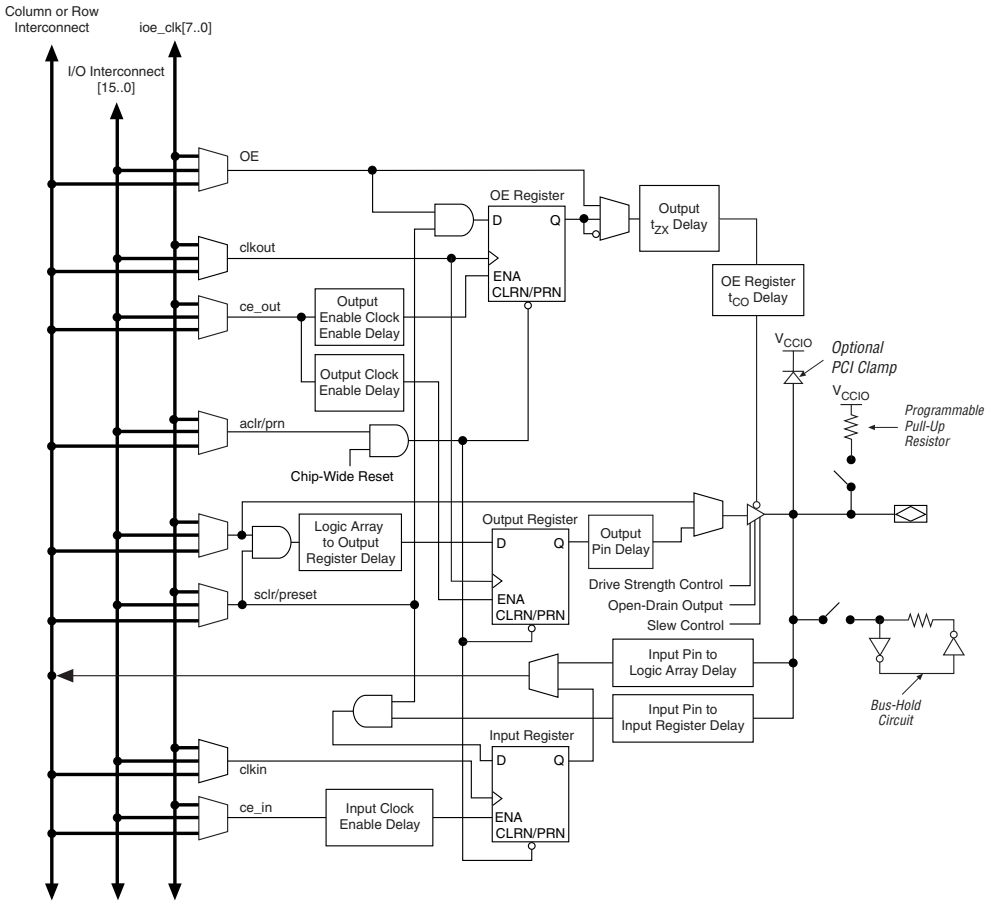
Each IOE contains its own control signal selection for the following control signals: `oe`, `ce_in`, `ce_out`, `aclr/preset`, `sclr/preset`, `clk_in`, and `clk_out`. Figure 4–62 illustrates the control signal selection.

Figure 4–62. Control Signal Selection per IOE



In normal bidirectional operation, the input register can be used for input data requiring fast setup times. The input register can have its own clock input and clock enable separate from the OE and output registers. The output register can be used for data requiring fast clock-to-output performance. The OE register can be used for fast clock-to-output enable timing. The OE and output register share the same clock source and the same clock enable source from local interconnect in the associated LAB, dedicated I/O clocks, and the column and row interconnects. Figure 4–63 shows the IOE in bidirectional configuration.

Figure 4–63. Stratix GX IOE in Bidirectional I/O Configuration *Note (1)*



Note to Figure 4–63:
(1) All input signals to the IOE can be inverted at the IOE.

The Stratix GX device IOE includes programmable delays that can be activated to ensure zero hold times, input IOE register-to-logic array register transfers, or logic array-to-output IOE register transfers.

A path in which a pin directly drives a register may require the delay to ensure zero hold time, whereas a path in which a pin drives a register through combinatorial logic may not require the delay. Programmable delays exist for decreasing input-pin-to-logic-array and IOE input register delays. The Quartus II Compiler can program these delays to automatically minimize setup time while providing a zero hold time.

Programmable delays can increase the register-to-pin delays for output and/or output enable registers. A programmable delay exists to increase the t_{ZX} delay to the output pin, which is required for ZBT interfaces. Table 4–21 shows the programmable delays for Stratix GX devices.

Table 4–21. Stratix GX Programmable Delay Chain

Programmable Delays	Quartus II Logic Option
Input pin to logic array delay	Decrease input delay to internal cells
Input pin to input register delay	Decrease input delay to input register
Output pin delay	Increase delay to output pin
Output enable register t_{CO} delay	Increase delay to output enable pin
Output t_{ZX} delay	Increase t_{ZX} delay to output pin
Output clock enable delay	Increase output clock enable delay
Input clock enable delay	Increase input clock enable delay
Logic array to output register delay	Decrease input delay to output register
Output enable clock enable delay	Increase output enable clock enable delay

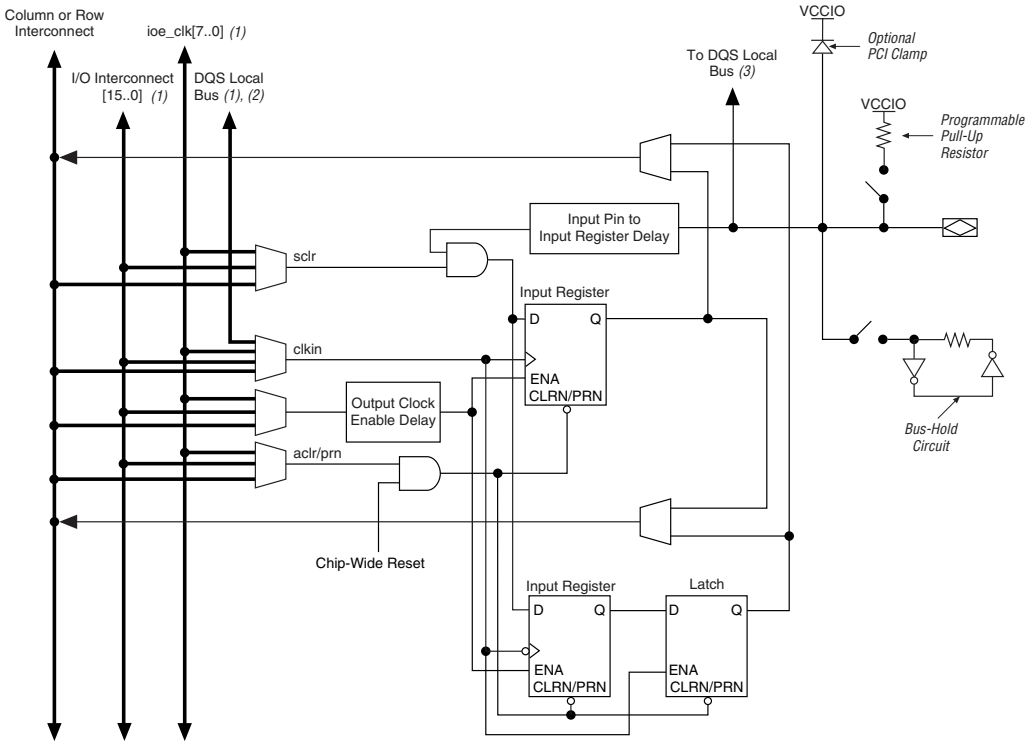
The IOE registers in Stratix GX devices share the same source for clear or preset. You can program preset or clear for each individual IOE. You can also program the registers to power up high or low after configuration is complete. If programmed to power up low, an asynchronous clear can control the registers. If programmed to power up high, an asynchronous preset can control the registers. This feature prevents the inadvertent activation of another device's active-low input upon power-up. If one register in an IOE uses a preset or clear signal then all registers in the IOE must use that same signal if they require preset or clear. Additionally, a synchronous reset signal is available for the IOE registers.

Double-Data Rate I/O Pins

Stratix GX devices have six registers in the IOE, which support DDR interfacing by clocking data on both positive and negative clock edges. The IOEs in Stratix GX devices support DDR inputs, DDR outputs, and bidirectional DDR modes.

When using the IOE for DDR inputs, the two input registers clock double rate input data on alternating edges. An input latch is also used within the IOE for DDR input acquisition. The latch holds the data that is present during the clock high times. This allows both bits of data to be synchronous with the same clock edge (either rising or falling). Figure 4–64 shows an IOE configured for DDR input. Figure 4–65 shows the DDR input timing diagram.

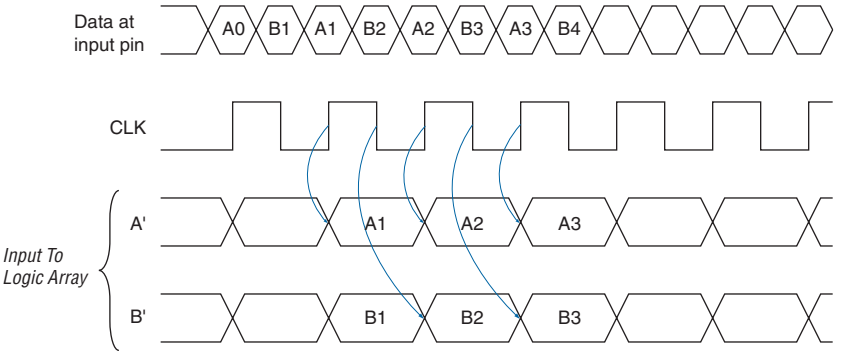
Figure 4–64. Stratix GX IOE in DDR Input I/O Configuration *Note (1)*



Notes to Figure 4–64:

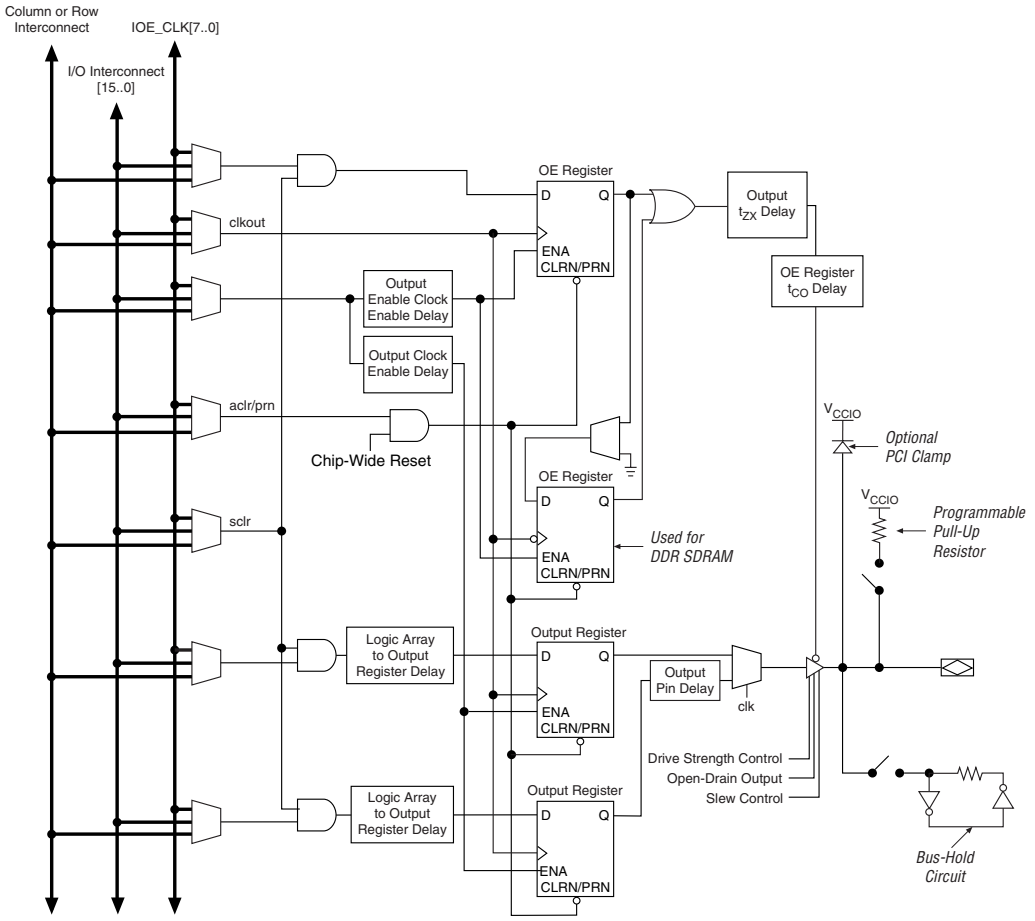
(1) All input signals to the IOE can be inverted at the IOE.
(2) This signal connection is only allowed on dedicated DQ function pins.
(3) This signal is for dedicated DQS function pins only.

Figure 4–65. Input Timing Diagram in DDR Mode



When using the IOE for DDR outputs, the two output registers are configured to clock two data paths from LEs on rising clock edges. These output registers are multiplexed by the clock to drive the output pin at a ×2 rate. One output register clocks the first bit out on the clock high time, while the other output register clocks the second bit out on the clock low time. Figure 4–66 shows the IOE configured for DDR output. Figure 4–67 shows the DDR output timing diagram.

Figure 4–66. Stratix GX IOE in DDR Output I/O Configuration *Notes (1), (2)*

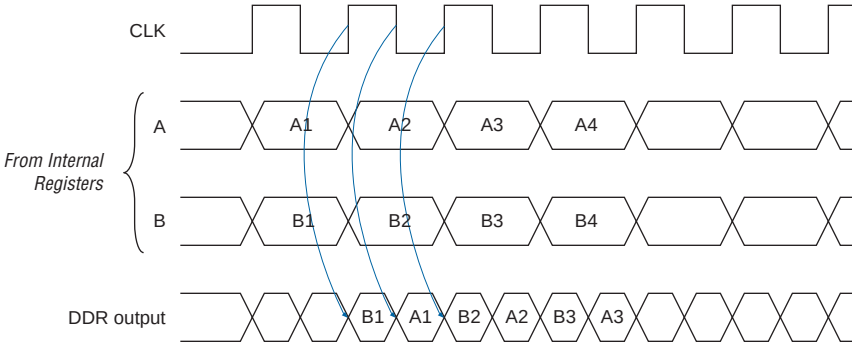


Notes to Figure 4–66:

(1) All input signals to the IOE can be inverted at the IOE.

(2) The tristate is by default active high. It can, however, be designed to be active low.

Figure 4–67. Output Timing Diagram in DDR Mode



The Stratix GX IOE operates in bidirectional DDR mode by combining the DDR input and DDR output configurations. Stratix GX device I/O pins transfer data on a DDR bidirectional bus to support DDR SDRAM. The negative-edge-clocked OE register holds the OE signal inactive until the falling edge of the clock. This is done to meet DDR SDRAM timing requirements.

External RAM Interfacing

Stratix GX devices support DDR SDRAM at up to 200 MHz (400-Mbps data rate) through dedicated phase-shift circuitry, QDR and QDRII SRAM interfaces up to 167 MHz, and ZBT SRAM interfaces up to 200 MHz. Stratix GX devices also provide preliminary support for reduced latency DRAM II (RLDRAM II) at rates up to 200 MHz through the dedicated phase-shift circuitry.

In addition to the required signals for external memory interfacing, Stratix GX devices offer the optional clock enable signal. By default the Quartus II software sets the clock enable signal high, which tells the output register to update with new values. The output registers hold their own values if the design sets the clock enable signal low. See Figure 4–63.

To find out more about the DDR SDRAM specification, see the JEDEC web site (**www.jedec.org**). For information on memory controller megafunctions for Stratix GX devices, see the Altera web site (**www.altera.com**). See *AN 342: Interfacing DDR SDRAM with Stratix & Stratix GX Devices* for more information on DDR SDRAM interface in Stratix GX. Also see *AN 349: QDR SRAM Controller Reference Design for Stratix & Stratix GX Devices* and *AN 329: ZBT SRAM Controller Reference Design for Stratix & Stratix GX Devices*.

Table 4–22 shows the performance specification for DDR SDRAM, RLDRAM II, QDR SRAM, QDRII SRAM, and ZBT SRAM interfaces in EP1SGX10 through EP1SGX40 devices. The DDR SDRAM and QDR SRAM numbers in Table 4–22 have been verified with hardware characterization with third-party DDR SDRAM and QDR SRAM devices over temperature and voltage extremes.

Table 4–22. External RAM Support in EP1SGX10 Through EP1SGX40 Devices

DDR Memory Type	I/O Standard	Maximum Clock Rate (MHz)		
		-5 Speed Grade	-6 Speed Grade	-7 Speed Grade
DDR SDRAM *(1)*, *(2)*	SSTL-2	200	167	133
DDR SDRAM - side banks *(2)*, *(3)*, *(4)*	SSTL-2	150	133	133
RLDRAM II *(4)*	1.8-V HSTL	200	*(5)*	*(5)*
QDR SRAM *(6)*	1.5-V HSTL	167	167	133
QDRII SRAM *(6)*	1.5-V HSTL	200	167	133
ZBT SRAM *(7)*	LVTTL	200	200	167

Notes to Table 4–22:

(1) These maximum clock rates apply if the Stratix GX device uses DQS phase-shift circuitry to interface with DDR SDRAM. DQS phase-shift circuitry is only available in the top and bottom I/O banks (I/O banks 3, 4, 7, and 8).
(2) For more information on DDR SDRAM, see *AN 342: Interfacing DDR SDRAM with Stratix & Stratix GX Devices.*
(3) DDR SDRAM is supported on the Stratix GX device side I/O banks (I/O banks 1, 2, 5, and 6) without dedicated DQS phase-shift circuitry. The read DQS signal is ignored in this mode.
(4) These performance specifications are preliminary.
(5) This device does not support RLDRAM II.
(6) For more information on QDR or QDRII SRAM, see *AN 349: QDR SRAM Controller Reference Design for Stratix & Stratix GX Devices*.
(7) For more information on ZBT SRAM, see *AN 329: ZBT SRAM Controller Reference Design for Stratix & Stratix GX Devices*.

In addition to six I/O registers and one input latch in the IOE for interfacing to these high-speed memory interfaces, Stratix GX devices also have dedicated circuitry for interfacing with DDR SDRAM. In every Stratix GX device, the I/O banks at the top (I/O banks 3 and 4) and bottom (I/O banks 7 and 8) of the device support DDR SDRAM up to 200 MHz. These pins support DQS signals with DQ bus modes of ×8, ×16, or ×32.

Table 4–23 shows the number of DQ and DQS buses that are supported per device.

Table 4–23. DQS & DQ Bus Mode Support *Note (1)*

Device	Package	Number of ×8 Groups	Number of ×16 Groups	Number of ×32 Groups
EP1SGX10	672-pin FineLine BGA	12 *(2)*	0	0
EP1SGX25	672-pin FineLine BGA	16 *(3)*	8	4
	1,020-pin FineLine BGA	20	8	4
EP1SGX40	1,020-pin FineLine BGA	20	8	4

Notes to Table 4–23:

(1) See the *Selectable I/O Standards in Stratix & Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2* for V_{REF} guidelines.

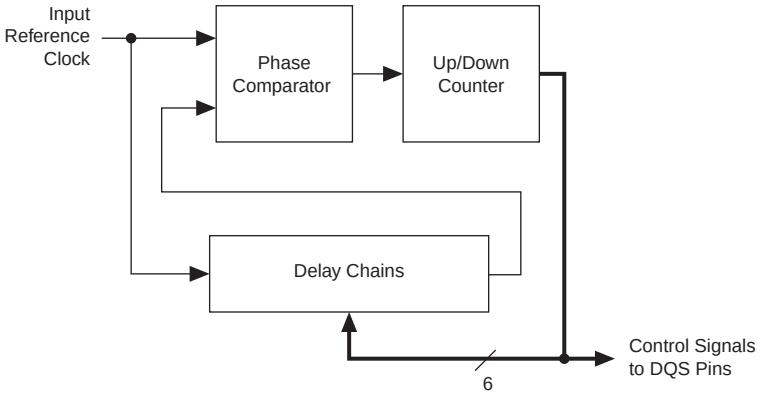
(2) These packages have six groups in I/O banks 3 and 4 and six groups in I/O banks 7 and 8.

(3) These packages have eight groups in I/O banks 3 and 4 and eight groups in I/O banks 7 and 8.

A compensated delay element on each DQS pin automatically aligns input DQS synchronization signals with the data window of their corresponding DQ data signals. The DQS signals drive a local DQS bus in the top and bottom I/O banks. This DQS bus is an additional resource to the I/O clocks and clocks DQ input registers with the DQS signal.

Two separate single phase-shifting reference circuits are located on the top and bottom of the Stratix GX device. Each circuit is driven by a system reference clock through the `CLK` pins that is the same frequency as the DQS signal. Clock pins `CLK[15..12]p` feed the phase-shift circuitry on the top of the device and clock pins `CLK[7..4]p` feed the phase-shift circuitry on the bottom of the device. The phase-shifting reference circuit on the top of the device controls the compensated delay elements for all 10 DQS pins located at the top of the device. The phase-shifting reference circuit on the bottom of the device controls the compensated delay elements for all 10 DQS pins located on the bottom of the device. All 10 delay elements (DQS signals) on either the top or bottom of the device shift by the same degree amount. For example, all 10 DQS pins on the top of the device can be shifted by 90° and all 10 DQS pins on the bottom of the device can be shifted by 72°. The reference circuits require a maximum of 256 system reference clock cycles to set the correct phase on the DQS delay elements. Figure 4–68 illustrates the phase-shift reference circuit control of each DQS delay shift on the top of the device. This same circuit is duplicated on the bottom of the device.

Figure 4–68. Simplified Diagram of the DQS Phase-Shift Circuitry



See the *External Memory Interfaces* chapter of the *Stratix GX Device Handbook, Volume 2* for more information on external memory interfaces.

Programmable Drive Strength

The output buffer for each Stratix GX device I/O pin has a programmable drive strength control for certain I/O standards. The LVTTL and LVCMOS standard has several levels of drive strength that the user can control. SSTL-3 class I and II, SSTL-2 class I and II, HSTL class I and II, and 3.3-V GTL+ support a minimum setting, the lowest drive strength that guarantees the I_{OH}/I_{OL} of the standard. Using minimum settings provides signal slew rate control to reduce system noise and signal overshoot.

Table 4–24 shows the possible settings for the I/O standards with drive strength control.

Table 4–24. Programmable Drive Strength

I/O Standard	I_{OH} / I_{OL} Current Strength Setting (mA)
3.3-V LVTTL	24 *(1)*, 16, 12, 8, 4
3.3-V LVCMOS	24 *(2)*, 12 *(1)*, 8, 4, 2
2.5-V LVTTL/LVCMOS	16 *(1)*, 12, 8, 2
1.8-V LVTTL/LVCMOS	12 *(1)*, 8, 2
1.5-V LVCMOS	8 *(1)*, 4, 2
GTL/GTL+ 1.5-V HSTL class I and II 1.8-V HSTL class I and II SSTL-3 class I and II SSTL-2 class I and II SSTL-18 class I and II	Support maximum and minimum strength

Notes to Table 4–24:
(1) This is the Quartus II software default current setting.
(2) I/O banks 1 and 2 do not support this setting.

The Quartus II software, beginning with version 4.2, reports current strength as "PCI Compliant" for 3.3-V PCI, 3.3-V PCI-X 1.0, and Compact PCI I/O standards.

Stratix GX devices support series on-chip termination (OCT) using programmable drive strength. For more information, contact your Altera Support Representative.

Open-Drain Output

Stratix GX devices provide an optional open-drain (equivalent to an open-collector) output for each I/O pin. This open-drain output enables the device to provide system-level control signals (that is, interrupt and write-enable signals) that can be asserted by any of several devices.

Slew-Rate Control

The output buffer for each Stratix GX device I/O pin has a programmable output slew-rate control that can be configured for low-noise or high-speed performance. A faster slew rate provides high-speed transitions for high-performance systems. However, these fast transitions may introduce noise transients into the system. A slow slew rate reduces system noise, but adds a nominal delay to rising and falling edges. Each I/O pin has an individual slew-rate control, allowing you to specify the slew rate on a pin-by-pin basis. The slew-rate control affects both the rising and falling edges.

Bus Hold

Each Stratix GX device I/O pin provides an optional bus-hold feature. The bus-hold circuitry can weakly hold the signal on an I/O pin at its last-driven state. Since the bus-hold feature holds the last-driven state of the pin until the next input signal is present, an external pull-up or pull-down resistor is not needed to hold a signal level when the bus is tri-stated.

Table 4–25 shows bus hold support for different pin types.

Table 4–25. Bus Hold Support

Pin Type	Bus Hold
I/O pins	✓
CLK[15..0]	
CLK[0,1,2,3,8,9,10,11]	
FCLK	✓
FPLL[7..10]CLK	

The bus-hold circuitry also pulls undriven pins away from the input threshold voltage where noise can cause unintended high-frequency switching. You can select this feature individually for each I/O pin. The bus-hold output drives no higher than V_{CCIO} to prevent overdriving signals. If the bus-hold feature is enabled, the programmable pull-up option cannot be used. Disable the bus-hold feature when using open-drain outputs with the GTL+ I/O standard or when the I/O pin has been configured for differential signals.

The bus-hold circuitry uses a resistor with a nominal resistance (R_{BH}) of approximately 7 kΩ to weakly pull the signal level to the last-driven state. The chapter *DC & Switching Characteristics* of the *Stratix GX Device Handbook, Volume 1* gives the specific sustaining current driven through this resistor and the overdrive current used to identify the next-driven input level. This information is provided for each V_{CCIO} voltage level.

The bus-hold circuitry is active only after configuration. When going into user mode, the bus-hold circuit captures the value on the pin present at the end of configuration.

Programmable Pull-Up Resistor

Each Stratix GX device I/O pin provides an optional programmable pull-up resistor during user mode. If this feature is enabled for an I/O pin, the pull-up resistor (typically 25 kΩ) weakly holds the output to the V_{CCIO} level of the output pin's bank. Table 4–26 shows which pin types support the weak pull-up resistor feature.

Table 4–26. Programmable Weak Pull-Up Resistor Support

Pin Type	Programmable Weak Pull-Up Resistor
I/O pins	✓
CLK[15..0]	
FCLK	✓
FPLL[7..10]CLK	
Configuration pins	
JTAG pins	✓ *(1)*

Note to Table 4–26:
(1) TDO pins do not support programmable weak pull-up resistors.

Advanced I/O Standard Support

Stratix GX device IOEs support the following I/O standards:

- LVTTL
- LVCMOS
- 1.5 V
- 1.8 V
- 2.5 V
- 3.3-V PCI
- 3.3-V PCI-X 1.0
- 3.3-V AGP (1× and 2×)

- LVDS
- LVPECL
- 3.3-V PCML
- HyperTransport
- Differential HSTL (on input/output clocks only)
- Differential SSTL (on output column clock pins only)
- GTL/GTL+
- 1.5-V HSTL class I and II
- 1.8-V HSTL Class I and II
- SSTL-3 class I and II
- SSTL-2 class I and II
- SSTL-18 class I and II
- CTT

Table 4–27 describes the I/O standards supported by Stratix GX devices.

Table 4–27. Stratix GX Supported I/O Standards (Part 1 of 2)

I/O Standard	Type	Input Reference Voltage (V_{REF}) (V)	Output Supply Voltage (V_{CCIO}) (V)	Board Termination Voltage (V_{TT}) (V)
LVTTL	Single-ended	N/A	3.3	N/A
LVCMOS	Single-ended	N/A	3.3	N/A
2.5 V	Single-ended	N/A	2.5	N/A
1.8 V	Single-ended	N/A	1.8	N/A
1.5 V	Single-ended	N/A	1.5	N/A
3.3-V PCI	Single-ended	N/A	3.3	N/A
3.3-V PCI-X 1.0	Single-ended	N/A	3.3	N/A
LVDS	Differential	N/A	3.3	N/A
LVPECL	Differential	N/A	3.3	N/A
3.3-V PCML	Differential	N/A	3.3	N/A
HyperTransport	Differential	N/A	2.5	N/A
Differential HSTL *(1)*	Differential	0.75	1.5	0.75
Differential SSTL *(2)*	Differential	1.25	2.5	1.25
GTL	Voltage-referenced	0.8	N/A	1.20
GTL+	Voltage-referenced	1.0	N/A	1.5
1.5-V HSTL class I and II	Voltage-referenced	0.75	1.5	0.75
1.8-V HSTL class I and II	Voltage-referenced	0.9	1.8	0.9
SSTL-18 class I and II	Voltage-referenced	0.90	1.8	0.90
SSTL-2 class I and II	Voltage-referenced	1.25	2.5	1.25

Table 4–27. Stratix GX Supported I/O Standards (Part 2 of 2)

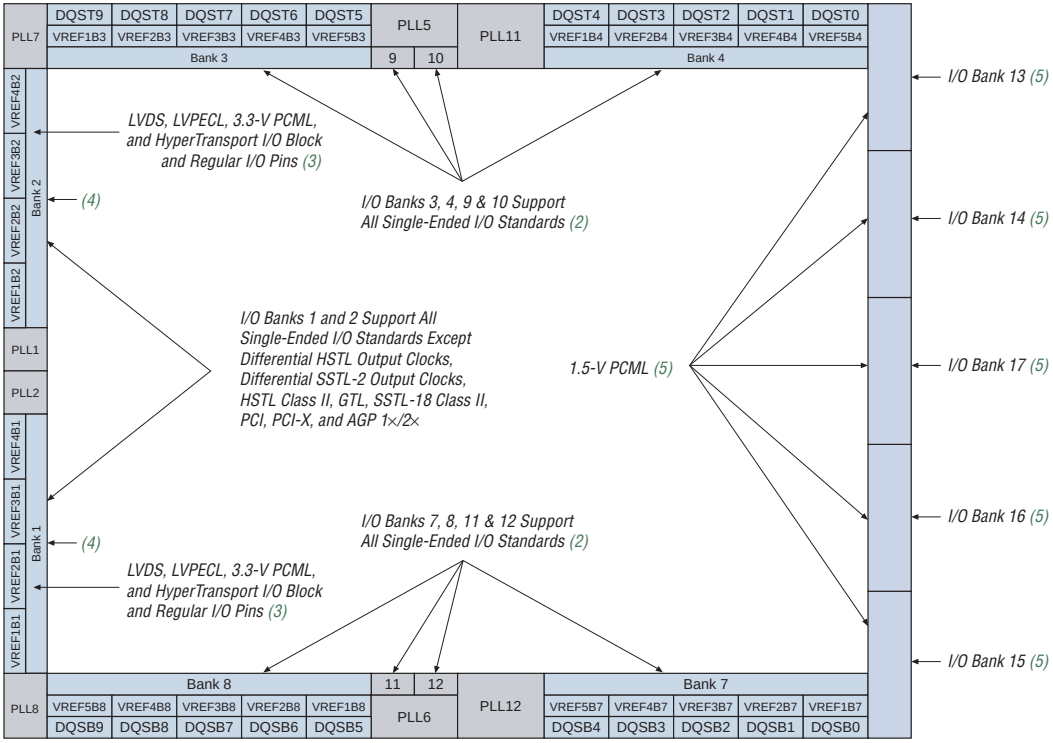
I/O Standard	Type	Input Reference Voltage (V_{REF}) (V)	Output Supply Voltage (V_{CCIO}) (V)	Board Termination Voltage (V_{TT}) (V)
SSTL-3 class I and II	Voltage-referenced	1.5	3.3	1.5
AGP (1× and 2×)	Voltage-referenced	1.32	3.3	N/A
CTT	Voltage-referenced	1.5	3.3	1.5

Notes to Table 4–27:
(1) This I/O standard is only available on input and output clock pins.
(2) This I/O standard is only available on output column clock pins.

For more information on I/O standards supported by Stratix GX devices, see the *Selectable I/O Standards in Stratix & Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2*.

Stratix GX devices contain eight I/O banks in addition to the four enhanced PLL external clock out banks, as shown in Figure 4–69. The four I/O banks on the right and left of the device contain circuitry to support high-speed differential I/O for LVDS, LVPECL, 3.3-V PCML, and HyperTransport inputs and outputs. These banks support all I/O standards listed in Table 4–27 except PCI I/O pins or PCI-X 1.0, GTL, SSTL-18 Class II, and HSTL Class II outputs. The top and bottom I/O banks support all single-ended I/O standards. Additionally, Stratix GX devices support four enhanced PLL external clock output banks, allowing clock output capabilities such as differential support for SSTL and HSTL. Table 4–28 shows I/O standard support for each I/O bank.

Figure 4–69. Stratix GX I/O Banks *Notes (1), (2), (3)*



Notes to Figure 4–69:

(1) Figure 4–69 is a top view of the Stratix GX silicon die.

(2) Banks 9 through 12 are enhanced PLL external clock output banks.

(3) If the high-speed differential I/O pins are not used for high-speed differential signaling, they can support all of the I/O standards except HSTL class I and II, GTL, SSTL-18 Class II, PCI, PCI-X, and AGP 1×/2×.

(4) For guidelines for placing single-ended I/O pads next to differential I/O pads, see the *Selectable I/O Standards in Stratix & Stratix GX Devices* chapter in the *Stratix GX Device Handbook, Volume 2*.

(5) These I/O banks in Stratix GX devices also support the LVDS, LVPECL, and 3.3-V PCML I/O standards on reference clocks and receiver input pins (AC coupled)

Table 4–28 shows I/O standard support for each I/O bank.

Table 4–28. I/O Support by Bank (Part 1 of 2)

I/O Standard	Top & Bottom Banks (3, 4, 7 & 8)	Left Banks (1 & 2)	Enhanced PLL External Clock Output Banks (9, 10, 11 & 12)
LVTTL	✓	✓	✓
LVCMOS	✓	✓	✓
2.5 V	✓	✓	✓
1.8 V	✓	✓	✓
1.5 V	✓	✓	✓
3.3-V PCI	✓		✓
3.3-V PCI-X 1.0	✓		✓
LVPECL		✓	✓
3.3-V PCML		✓	✓
LVDS		✓	✓
HyperTransport technology		✓	✓
Differential HSTL (clock inputs)	✓	✓	
Differential HSTL (clock outputs)			✓
Differential SSTL (clock outputs)			✓
3.3-V GTL	✓		✓
3.3-V GTL+	✓	✓	✓
1.5-V HSTL class I	✓	✓	✓
1.5-V HSTL class II	✓		✓
1.8-V HSTL class I	✓	✓	✓
1.8-V HSTL class II	✓		✓
SSTL-18 class I	✓	✓	✓
SSTL-18 class II	✓		✓
SSTL-2 class I	✓	✓	✓
SSTL-2 class II	✓	✓	✓
SSTL-3 class I	✓	✓	✓

Table 4–28. I/O Support by Bank (Part 2 of 2)

I/O Standard	Top & Bottom Banks (3, 4, 7 & 8)	Left Banks (1 & 2)	Enhanced PLL External Clock Output Banks (9, 10, 11 & 12)
SSTL-3 class II	✓	✓	✓
AGP (1× and 2×)	✓		✓
CTT	✓	✓	✓

Each I/O bank has its own VCCIO pins. A single device can support 1.5-, 1.8-, 2.5-, and 3.3-V interfaces; each bank can support a different standard independently. Each bank also has dedicated VREF pins to support any one of the voltage-referenced standards (such as SSTL-3) independently.

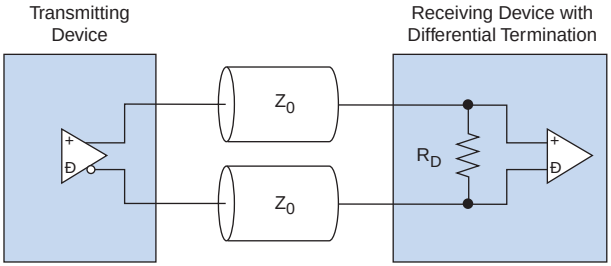
Each I/O bank can support multiple standards with the same V_{CCIO} for input and output pins. Each bank can support one voltage-referenced I/O standard. For example, when V_{CCIO} is 3.3 V, a bank can support LVTTL, LVCMOS, 3.3-V PCI, and SSTL-3 for inputs and outputs.

Differential On-Chip Termination

Stratix GX devices provide differential on-chip termination (LVDS I/O standard) to reduce reflections and maintain signal integrity. Differential on-chip termination simplifies board design by minimizing the number of external termination resistors required. Termination can be placed inside the package, eliminating small stubs that can still lead to reflections. The internal termination is designed using transistors in the linear region of operation.

Stratix GX devices support internal differential termination with a nominal resistance value of 137.5 Ω for LVDS input receiver buffers. LVPECL signals require an external termination resistor. Figure 4–70 shows the device with differential termination.

Figure 4–70. LVDS Input Differential On-Chip Termination



I/O banks on the left and right side of the device support LVDS receiver (far-end) differential termination.

Table 4–29 shows the Stratix GX device differential termination support.

Table 4–29. Differential Termination Supported by I/O Banks

Differential Termination Support	I/O Standard Support	Top & Bottom Banks (3, 4, 7 & 8)	Left Banks (1 & 2)
Differential termination *(1)*, *(2)*	LVDS		✓

Notes to Table 4–29:
(1) Clock pin CLK0, CLK2, CLK9, CLK11, and pins FPLL[7..10]CLK do not support differential termination.
(2) Differential termination is only supported for LVDS because of a 3.3-V V_{CCIO}.

Table 4–30 shows the termination support for different pin types.

Table 4–30. Differential Termination Support Across Pin Types

Pin Type	R_D
Top and bottom I/O banks (3, 4, 7, and 8)	
DIFFIO_RX[]	✓
CLK[0,2,9,11],CLK[4-7],CLK[12-15]	
CLK[1,3,8,10]	✓
FCLK	
FPLL[7..10]CLK	

The differential on-chip resistance at the receiver input buffer is 118 Ω ±20 %.

However, there is additional resistance present between the device ball and the input of the receiver buffer, as shown in Figure 4–71. This resistance is because of package trace resistance (which can be calculated as the resistance from the package ball to the pad) and the parasitic layout metal routing resistance (which is shown between the pad and the intersection of the on-chip termination and input buffer).

Figure 4–71. Differential Resistance of LVDS Differential Pin Pair (R_D)

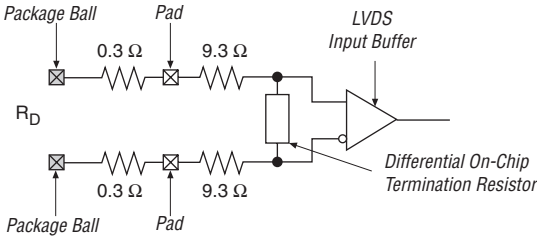


Table 4–31 defines the specification for internal termination resistance for commercial devices.

Table 4–31. Differential On-Chip Termination

Symbol	Description	Conditions	Resistance			Unit
			Min	Typ	Max	
R_D *(2)*	Internal differential termination for LVDS	Commercial *(1)*, *(3)*	110	135	165	Ω
		Industrial *(2)*, *(3)*	100	135	170	Ω

Notes to Table 4–31:

(1) Data measured over minimum conditions (T_j = 0 C, V_{CCIO} +5%) and maximum conditions (T_j = 85 C, V_{CCIO} = –5%).

(2) Data measured over minimum conditions (T_j = –40 C, V_{CCIO} +5%) and maximum conditions (T_j = 100 C, V_{CCIO} = –5%).

(3) LVDS data rate is supported for 840 Mbps using internal differential termination.

MultiVolt I/O Interface

The Stratix GX architecture supports the MultiVolt I/O interface feature, which allows Stratix GX devices in all packages to interface with systems of different supply voltages.

The Stratix GX VCCINT pins must always be connected to a 1.5-V power supply. With a 1.5-V V_{CCINT} level, input pins are 1.5-V, 1.8-V, 2.5-V, and 3.3-V tolerant. The VCCIO pins can be connected to either a 1.5-V, 1.8-V,

2.5-V, or 3.3-V power supply, depending on the output requirements. The output levels are compatible with systems of the same voltage as the power supply (for example, when VCCIO pins are connected to a 1.5-V power supply, the output levels are compatible with 1.5-V systems). When VCCIO pins are connected to a 3.3-V power supply, the output high is 3.3 V and is compatible with 3.3-V or 5.0-V systems.

Table 4–32 summarizes Stratix GX MultiVolt I/O support.

Table 4–32. Stratix GX MultiVolt I/O Support *Note (1)*

V_{CCIO} (V)	Input Signal *(5)*					Output Signal *(6)*				
	1.5 V	1.8 V	2.5 V	3.3 V	5.0 V	1.5 V	1.8 V	2.5 V	3.3 V	5.0 V
1.5	✓	✓	✓ *(2)*	✓ *(2)*		✓				
1.8	✓ *(2)*	✓	✓ *(2)*	✓ *(2)*		✓ *(3)*	✓			
2.5			✓	✓		✓ *(3)*	✓ *(3)*	✓		
3.3			✓ *(2)*	✓	✓ *(4)*	✓ *(3)*	✓ *(3)*	✓ *(3)*	✓	✓

Notes to Table 4–32:

(1) To drive inputs higher than V_{CCIO} but less than 4.1 V, disable the PCI clamping diode. However, to drive 5.0-V inputs to the device, enable the PCI clamping diode to prevent V_I from rising above 4.0 V.

(2) The input pin current may be slightly higher than the typical value.

(3) Although V_{CCIO} specifies the voltage necessary for the Stratix GX device to drive out, a receiving device powered at a different level can still interface with the Stratix GX device if it has inputs that tolerate the V_{CCIO} value.

(4) Stratix GX devices can be 5.0-V tolerant with the use of an external resistor and the internal PCI clamp diode.

(5) This is the external signal that is driving the Stratix GX device.

(6) This represents the system voltage that Stratix GX supports when a VCCIO pin is connected to a specific voltage level. For example, when VCCIO is 3.3 V and if the I/O standard is LVTTL/LVCMOS, the output high of the signal coming out from Stratix GX is 3.3 V and is compatible with 3.3-V or 5.0-V systems.

Power Sequencing & Hot Socketing

Because Stratix GX devices can be used in a mixed-voltage environment, they have been designed specifically to tolerate any possible power-up sequence. Therefore, the VCCIO and VCCINT power supplies may be powered in any order.

Signals can be driven into Stratix GX devices before and during power up without damaging the device. In addition, Stratix GX devices do not drive out during power up. Once operating conditions are reached and the device is configured, Stratix GX devices operate as specified by the user. For more information, see the *Selectable I/O Standards in Stratix & Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2*.

IEEE Std. 1149.1 (JTAG) Boundary-Scan Support

All Stratix GX devices provide JTAG BST circuitry that complies with the IEEE Std. 1149.1a-1990 specification. JTAG boundary-scan testing can be performed either before or after, but not during configuration. Stratix GX devices can also use the JTAG port for configuration together with either the Quartus II software or hardware using either Jam Files (**.jam**) or Jam Byte-Code Files (**.jbc**).

Stratix GX devices support IOE I/O standard setting reconfiguration through the JTAG BST chain. The JTAG chain can update the I/O standard for all input and output pins any time before or during user mode. You can use this ability for JTAG testing before configuration when some of the Stratix GX pins drive or receive from other devices on the board using voltage-referenced standards. Because the Stratix GX device may not be configured before JTAG testing, the I/O pins may not be configured for appropriate electrical standards for chip-to-chip communication. Programming those I/O standards via JTAG allows you to fully test I/O connection to other devices.

The enhanced PLL reconfiguration bits are part of the JTAG chain before configuration and after power-up. After device configuration, the PLL reconfiguration bits are not part of the JTAG chain.

Stratix GX devices also use the JTAG port to monitor the logic operation of the device with the SignalTap® embedded logic analyzer. Stratix GX devices support the JTAG instructions shown in Table 4–33.

Table 4–33. Stratix GX JTAG Instructions (Part 1 of 2)

JTAG Instruction	Description
SAMPLE/PRELOAD	Allows a snapshot of signals at the device pins to be captured and examined during normal device operation, and permits an initial data pattern to be output at the device pins. Also used by the SignalTap® embedded logic analyzer.
EXTEST *(1)*	Allows the external circuitry and board-level interconnects to be tested by forcing a test pattern at the output pins and capturing test results at the input pins.
BYPASS	Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through selected devices to adjacent devices during normal device operation.
USERCODE	Selects the 32-bit USERCODE register and places it between the TDI and TDO pins, allowing the USERCODE to be serially shifted out of TDO.
IDCODE	Selects the IDCODE register and places it between TDI and TDO, allowing the IDCODE to be serially shifted out of TDO.
HIGHZ *(1)*	Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through selected devices to adjacent devices during normal device operation, while tri-stating all of the I/O pins.

Table 4–33. Stratix GX JTAG Instructions (Part 2 of 2)

JTAG Instruction	Description
CLAMP *(1)*	Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through selected devices to adjacent devices during normal device operation while holding I/O pins to a state defined by the data in the boundary-scan register.
ICR instructions	Used when configuring a Stratix GX device through the JTAG port with a MasterBlaster™ or ByteBlasterMV™ download cable, or when using a .jam file or .jbc file with an embedded processor.
PULSE_NCONFIG	Emulates pulsing the nCONFIG pin low to trigger reconfiguration even though the physical pin is unaffected.
CONFIG_IO	Allows the IOE standards to be configured through the JTAG chain. Stops configuration if executed during configuration. Can be executed before or after configuration.
SignalTap instructions	Monitors internal device operation with the SignalTap embedded logic analyzer.

Note to Table 4–33:
(1) Bus hold and weak pull-up resistor features override the high-impedance state of HIGHZ, CLAMP, and EXTEST.

The Stratix GX device instruction register length is 10 bits, and the USERCODE register length is 32 bits. Tables 4–34 and 4–35 show the boundary-scan register length and IDCODE information for Stratix GX devices.

Table 4–34. Stratix GX Boundary-Scan Register Length

Device	Boundary-Scan Register Length
EP1SGX10	1,029
EP1SGX25	1,665
EP1SGX40	1,941

Table 4–35. 32-Bit Stratix GX Device IDCODE (Part 1 of 2)

Device	IDCODE (32 Bits) *(1)*			
	Version (4 Bits)	Part Number (16 Bits)	Manufacturer Identity (11 Bits)	LSB (1 Bit) *(2)*
EP1SGX10	0000	0010 0000 0100 0001	000 0110 1110	1
EP1SGX25	0000	0010 0000 0100 0011	000 0110 1110	1

Table 4–35. 32-Bit Stratix GX Device IDCODE (Part 2 of 2)

Device	IDCODE (32 Bits) *(1)*			
	Version (4 Bits)	Part Number (16 Bits)	Manufacturer Identity (11 Bits)	LSB (1 Bit) *(2)*
EP1SGX40	0000	0010 0000 0100 0101	000 0110 1110	1

Notes to Table 4–35:
(1) The most significant bit (MSB) is at the left end of the string.
(2) The IDCODE's least significant bit (LSB) is always 1.

Figure 4–72 shows the timing requirements for the JTAG signals.

Figure 4–72. Stratix GX JTAG Waveforms

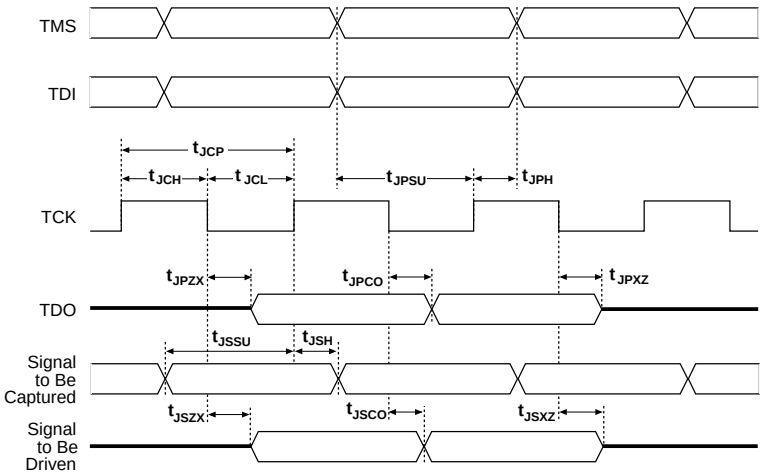


Table 4–36 shows the JTAG timing parameters and values for Stratix GX devices.

Table 4–36. Stratix GX JTAG Timing Parameters & Values (Part 1 of 2)

Symbol	Parameter	Min (ns)	Max (ns)
t_{JCP}	TCK clock period	100	
t_{JCH}	TCK clock high time	50	
t_{JCL}	TCK clock low time	50	
t_{JPSU}	JTAG port setup time	20	

Table 4–36. Stratix GX JTAG Timing Parameters & Values (Part 2 of 2)

Symbol	Parameter	Min (ns)	Max (ns)
t_{JPH}	JTAG port hold time	45	
t_{JPCO}	JTAG port clock to output		25
t_{JPZX}	JTAG port high impedance to valid output		25
t_{JPXZ}	JTAG port valid output to high impedance		25
t_{JSSU}	Capture register setup time	20	
t_{JSH}	Capture register hold time	45	
t_{JSCO}	Update register clock to output		35
t_{JSZX}	Update register high impedance to valid output		35
t_{JSXZ}	Update register valid output to high impedance		35

5. Configuration & Testing

SGX51005-1.0

SignalTap Embedded Logic Analyzer

Stratix® GX devices feature the SignalTap® embedded logic analyzer, which monitors design operation over a period of time through the IEEE Std. 1149.1 (JTAG) circuitry. You can analyze internal logic at speed without bringing internal signals to the I/O pins. This feature is particularly important for advanced packages, such as FineLine BGA® packages, because it can be difficult to add a connection to a pin during the debugging process after a board is designed and manufactured.

Configuration

The logic, circuitry, and interconnects in the Stratix GX architecture are configured with CMOS SRAM elements. Stratix GX devices are reconfigurable and are 100% tested prior to shipment. As a result, you do not have to generate test vectors for fault coverage purposes, and can instead focus on simulation and design verification. In addition, you do not need to manage inventories of different ASIC designs. Stratix GX devices can be configured on the board for the specific functionality required.

Stratix GX devices are configured at system power-up with data stored in an Altera serial configuration device or provided by a system controller. Altera offers in-system programmability (ISP)-capable configuration devices that configure Stratix GX devices via a serial data stream. Stratix GX devices can be configured in under 100 ms using 8-bit parallel data at 100 MHz. The Stratix GX device's optimized interface allows microprocessors to configure it serially or in parallel, and synchronously or asynchronously. The interface also enables microprocessors to treat Stratix GX devices as memory and configure them by writing to a virtual memory location, making reconfiguration easy. After a Stratix GX device has been configured, it can be reconfigured in-circuit by resetting the device and loading new data. Real-time changes can be made during system operation, enabling innovative reconfigurable computing applications.

Operating Modes

The Stratix GX architecture uses SRAM configuration elements that require configuration data to be loaded each time the circuit powers up. The process of physically loading the SRAM data into the device is called configuration. During initialization, which occurs immediately after configuration, the device resets registers, enables I/O pins, and begins to operate as a logic device. The I/O pins are tri-stated during power up,

and before and during configuration. Together, the configuration and initialization processes are called command mode. Normal device operation is called user mode.

A built-in weak pull-up resistor pulls all user I/O pins to V_{CCIO} before and during device configuration.

SRAM configuration elements allow Stratix GX devices to be reconfigured in-circuit by loading new configuration data into the device. With real-time reconfiguration, the device is forced into command mode with a device pin. The configuration process loads different configuration data, reinitializes the device, and resumes user-mode operation. You can perform in-field upgrades by distributing new configuration files either within the system or remotely.

Configuration Schemes

You can load the configuration data for a Stratix GX device with one of five configuration schemes (see Table 5–1), chosen on the basis of the target application. You can use a configuration device, intelligent controller, or the JTAG port to configure a Stratix GX device. A configuration device can automatically configure a Stratix GX device at system power-up.

You can configure multiple Stratix GX devices in any of five configuration schemes by connecting the configuration enable (nCE) and configuration enable output (nCEO) pins on each device.

Table 5–1. Data Sources for Configuration

Configuration Scheme	Data Source
Configuration device	Enhanced or EPC2 configuration device
Passive serial (PS)	ByteBlasterMV™ or MasterBlaster™ download cable or serial data source
Passive parallel asynchronous (PPA)	Parallel data source
Fast passive parallel	Parallel data source
JTAG	MasterBlaster or ByteBlasterMV download cable or a microprocessor with a Jam or JBC file (.jam or .jbc)

Partial Reconfiguration

The enhanced PLLs within the Stratix GX device family support partial reconfiguration of their multiply, divide, and time delay settings without reconfiguring the entire device. You can use either serial data from the logic array or regular I/O pins to program the PLL's counter settings in a serial chain. This option provides considerable flexibility for frequency synthesis, allowing real-time variation of the PLL frequency and delay. The rest of the device is functional while reconfiguring the PLL. See the *Stratix GX Architecture* chapter of the *Stratix GX Device Handbook, Volume 1* for more information on Stratix GX PLLs.

Remote Update Configuration Modes

Stratix GX devices also support remote configuration using an Altera enhanced configuration device (for example, EPC16, EPC8, and EPC4 devices) with page mode selection. Factory configuration data is stored in the default page of the configuration device. This is the default configuration which contains the design required to control remote updates and handle or recover from errors. You write the factory configuration once into the flash memory or configuration device. Remote update data can update any of the remaining pages of the configuration device. If there is an error or corruption in a remote update configuration, the configuration device reverts back to the factory configuration information.

There are two remote configuration modes: remote and local configuration. You can use the remote update configuration mode for all three configuration modes: serial, parallel synchronous, and parallel asynchronous. Configuration devices (for example, EPC16 devices) only support serial and parallel synchronous modes. Asynchronous parallel mode allows remote updates when an intelligent host is used to configure the Stratix GX device. This host must support page mode settings similar to an EPC16 device.

Remote Update Mode

When the Stratix GX device is first powered-up in remote update programming mode, it loads the configuration located at page address `000`. The factory configuration should always be located at page address `000`, and should never be remotely updated. The factory configuration contains the required logic to perform the following operations:

- Determine the page address/load location for the next application's configuration data
- Recover from a previous configuration error

- Receive new configuration data and write it into the configuration device

The factory configuration is the default and takes control if an error occurs while loading the application configuration.

While in the factory configuration, the factory-configuration logic performs the following operations:

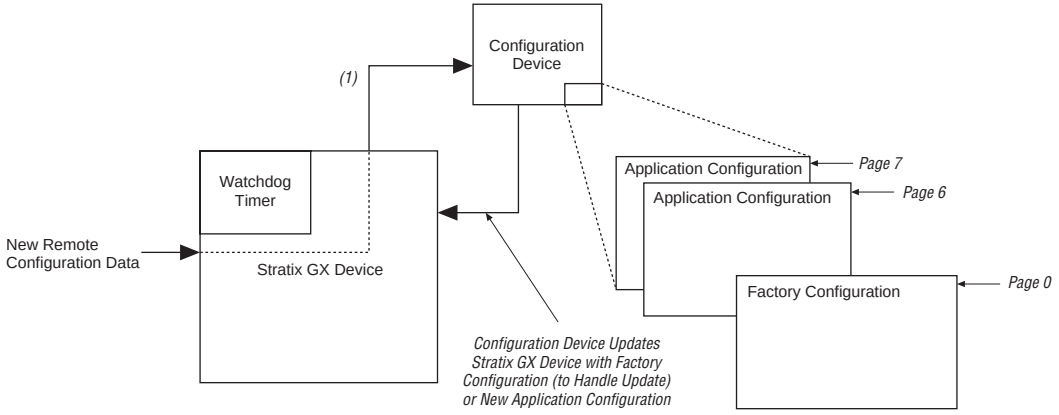
- Loads a remote update-control register to determine the page address of the new application configuration
- Determines whether to enable a user watchdog timer for the application configuration
- Determines what the watchdog timer setting should be if it is enabled

The user watchdog timer is a counter that must be continually reset within a specific amount of time in the user mode of an application configuration to ensure that valid configuration occurred during a remote update. Only valid application configurations designed for remote update can reset the user watchdog timer in user mode. If a valid application configuration does not reset the user watchdog timer in a specific amount of time, the timer updates a status register and loads the factory configuration. The user watchdog timer is automatically disabled for factory configurations.

If an error occurs in loading the application configuration, the configuration logic writes a status register to specify the cause of the reconfiguration. Once this occurs, the Stratix GX device automatically loads the factory configuration, which reads the status register and determines the reason for reconfiguration. Based on the reason, the factory configuration takes appropriate steps and writes the remote update control register to specify the next application configuration page to be loaded.

When the Stratix GX device successfully loads the application configuration, it enters into user mode. The Stratix GX device then executes the main application of the user. Intellectual property (IP), such as a Nios® embedded processor, can help the Stratix GX device determine when remote update is coming. The Nios embedded processor or user logic receives incoming data, writes it to the configuration device, and loads the factory configuration. The factory configuration reads the remote update status register and determine the valid application configuration to load. Figure 5–1 shows the Stratix GX remote update. Figure 5–2 shows the transition diagram for remote update mode.

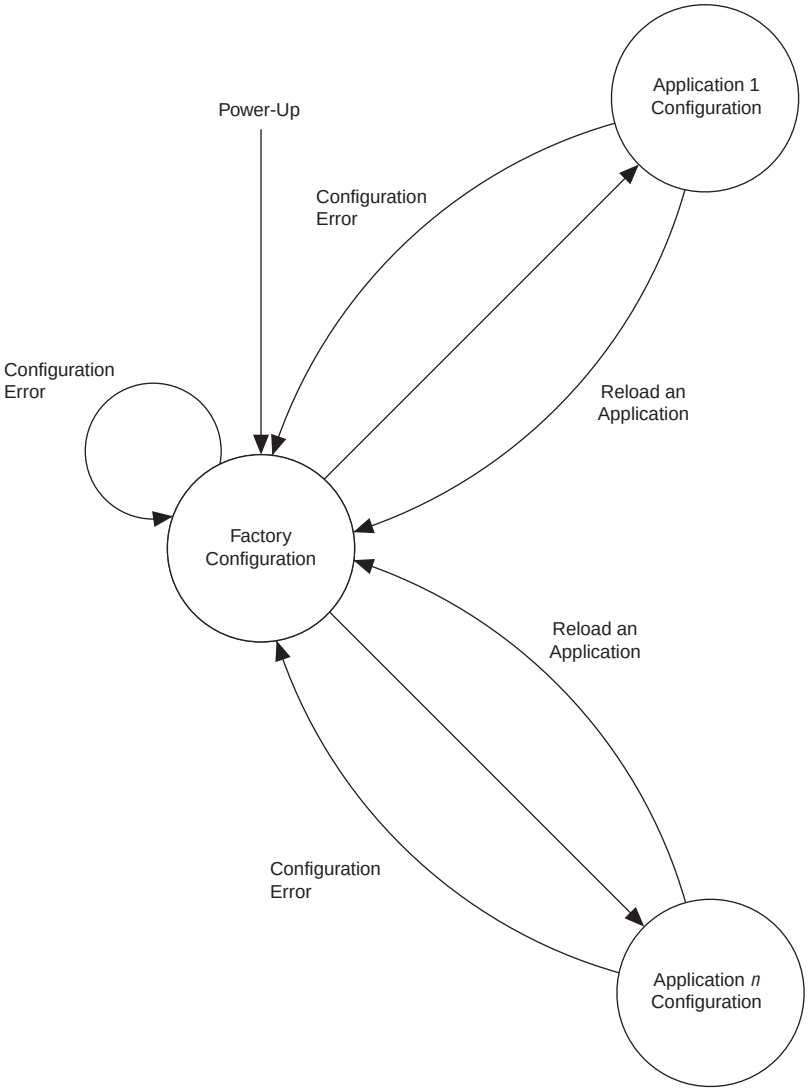
Figure 5–1. Stratix GX Device Remote Update



Note to Figure 5–1:

(1) When the Stratix GX device is configured with the factory configuration, it can handle update data from EPC16, EPC8, or EPC4 configuration device pages and point to the next page in the configuration device.

Figure 5–2. Remote Update Transition Diagram *Notes (1), (2)*



Notes to Figure 5–2:

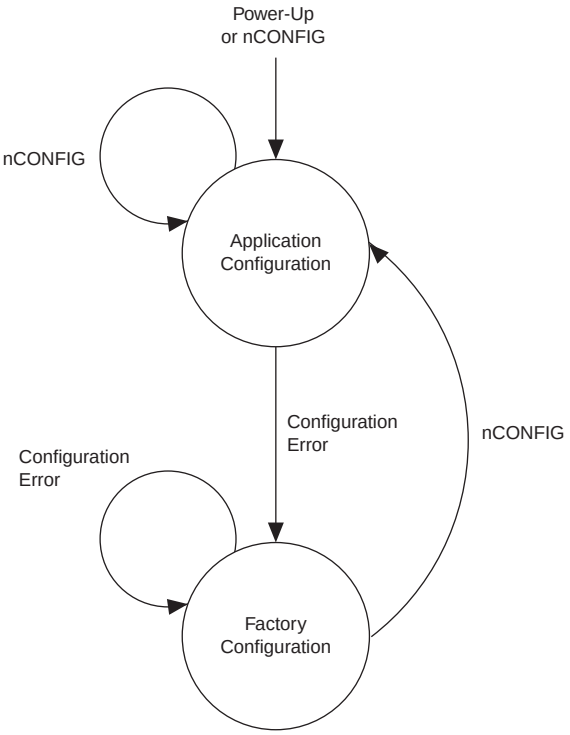
(1) Remote update of application configuration is controlled by a Nios embedded processor or user logic programmed in the factory or application configurations.

(2) Up to seven pages can be specified allowing up to seven different configuration applications.

Local Update Mode

Local update mode is a simplified version of the remote update. This feature is intended for simple systems that need to load a single application configuration immediately upon power-up without loading the factory configuration first. Local update designs have only one application configuration to load, so it does not require a factory configuration to determine which application configuration to use. Figure 5–3 shows the transition diagram for local update mode.

Figure 5–3. Local Update Transition Diagram



Stratix GX Automated Single Event Upset (SEU) Detection

Stratix GX devices offer on-chip circuitry for automated checking of single event upset (SEU) detection. Some applications that require the device to operate error free at high elevations or in close proximity to earth's North or South Pole require periodic checks to ensure continued data integrity. The error detection cyclic redundancy code (CRC) feature controlled by the **Device & Pin Options** dialog box in the Quartus II software uses a 32-bit CRC circuit to ensure data reliability and is one of the best options for mitigating SEU.

You can implement the error detection CRC feature with existing circuitry in Stratix GX devices, eliminating the need for external logic. For Stratix GX devices, the CRC is computed by Quartus II and downloaded into the device as a part of the configuration bit stream. The CRC_ERROR pin reports a soft error when configuration SRAM data is corrupted, triggering device reconfiguration.

Custom-Built Circuitry

Dedicated circuitry is built into Stratix GX devices to perform error detection automatically. This error detection circuitry constantly checks for errors in the configuration SRAM cells while the device is in user mode. You can monitor one external pin for the error and use it to trigger a reconfiguration cycle. You can select the desired time between checks by adjusting a built-in clock divider.

Software Interface

In the Quartus II software version 4.1 and later, you can turn on the automated error detection CRC feature in the **Device & Pin Options** dialog box. This dialog box allows you to enable the feature and set the internal frequency of the CRC between 400 kHz to 100 MHz. This controls the rate that the CRC circuitry verifies the internal configuration SRAM bits in the FPGA device.

For more information on CRC, refer to *AN 357: Error Detection Using CRC in Altera FPGA Devices*.

Temperature-Sensing Diode

Stratix GX devices include a diode-connected transistor for use as a temperature sensor in power management. This diode is used with an external digital thermometer device such as a MAX1617A or MAX1619 from MAXIM Integrated Products. These devices steer bias current through the Stratix GX diode, measuring forward voltage and converting this reading to temperature in the form of an 8-bit signed number (7 bits plus sign). The external device's output represents the package temperature of the Stratix GX device and can be used for intelligent power management.

The diode requires two pins (tempdiodep and tempdioden) on the Stratix GX device to connect to the external temperature-sensing device, as shown in Figure 5–4. The temperature-sensing diode is a passive element and therefore can be used before the Stratix GX device is powered.

Figure 5–4. External Temperature-Sensing Diode

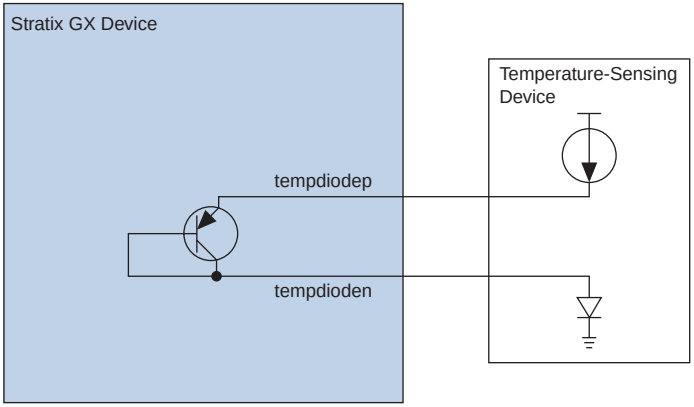


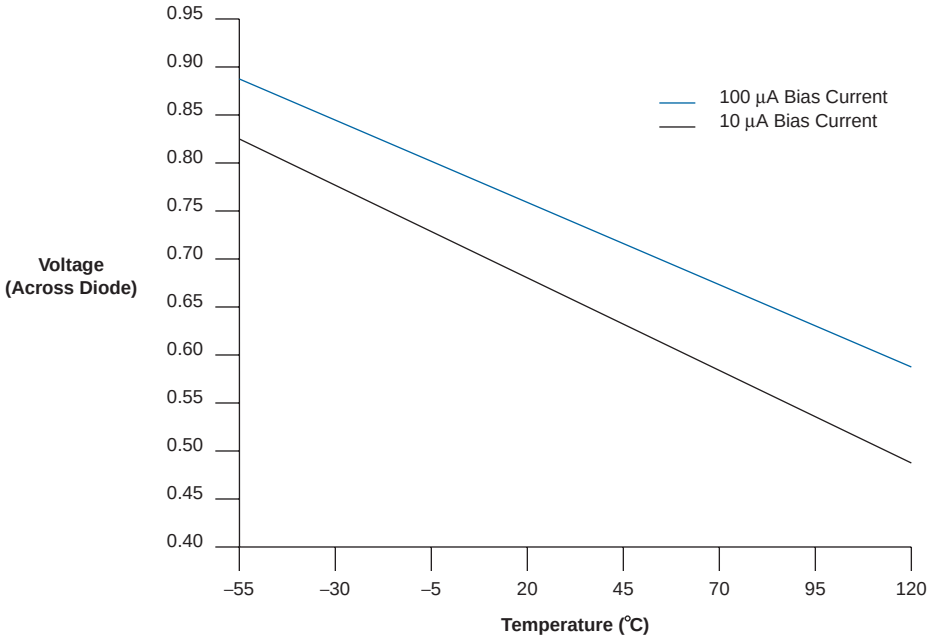
Table 5–2 shows the specifications for bias voltage and current of the Stratix GX temperature-sensing diode.

Table 5–2. Temperature-Sensing Diode Electrical Characteristics

Parameter	Minimum	Typical	Maximum	Units
I_{BIAS} high	80	100	120	µA
I_{BIAS} low	8	10	12	µA
$V_{BP} - V_{BN}$	0.3		0.9	V
V_{BN}		0.7		V
Series resistance			3	W

The temperature-sensing diode works for the entire operating range shown in Figure 5–5.

Figure 5–5. Temperature Versus Temperature-Sensing Diode Voltage



6. DC & Switching Characteristics

SGX51006-1.2

Operating Conditions

Stratix® GX devices are offered in both commercial and industrial grades. However, industrial-grade devices may have limited speed-grade availability.

Tables 6–1 through 6–12 provide information on absolute maximum ratings, recommended operating conditions, DC operating conditions, and transceiver block absolute maximum ratings. Notes for Tables 6–1 through 6–6 immediately follow Table 6–6, notes for Table 6–7 immediately follow that table, and notes for Tables 6–8 through 6–12 immediately follow Table 6–12.

Table 6–1. Stratix GX Device Absolute Maximum Ratings *Notes (1), (2)*

Symbol	Parameter	Conditions	Minimum	Maximum	Unit
V_{CCINT}	Supply voltage	With respect to ground *(3)*	–0.5	2.4	V
V_{CCIO}			–0.5	4.6	V
V_I	DC input voltage		–0.5	4.6	V
I_{OUT}	DC output current, per pin		–25	25	mA
T_{STG}	Storage temperature	No bias	–65	150	° C
T_{AMB}	Ambient temperature	Under bias	–65	135	° C
T_J	Junction temperature	BGA packages under bias		135	° C

Table 6–2. Stratix GX Device Recommended Operating Conditions (Part 1 of 2) *Note (7), (12), (13)*

Symbol	Parameter	Conditions	Minimum	Maximum	Unit
V_{CCINT}	Supply voltage for internal logic and input buffers	*(4)*	1.425	1.575	V
V_{CCIO}	Supply voltage for output buffers, 3.3-V operation	*(4), (5)*	3.00 (3.135)	3.60 (3.465)	V
	Supply voltage for output buffers, 2.5-V operation	*(4)*	2.375	2.625	V
	Supply voltage for output buffers, 1.8-V operation	*(4)*	1.71	1.89	V
	Supply voltage for output buffers, 1.5-V operation	*(4)*	1.4	1.6	V
V_I	Input voltage	*(3), (6)*	–0.5	4.1	V

Table 6–2. Stratix GX Device Recommended Operating Conditions (Part 2 of 2) *Note (7), (12), (13)*

Symbol	Parameter	Conditions	Minimum	Maximum	Unit
V_O	Output voltage		0	V_{CCIO}	V
T_J	Operating junction temperature	For commercial use	0	85	° C
		For industrial use	–40	100	° C

Table 6–3. Stratix GX Device DC Operating Conditions *Note (12)*

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
I_I	Input pin leakage current	$V_I = V_{CCIOmax}$ to 0 V *(8)*	–10		10	µA
I_{OZ}	Tri-stated I/O pin leakage current	$V_O = V_{CCIOmax}$ to 0 V *(8)*	–10		10	µA
R_{CONF}	Value of I/O pin pull-up resistor before and during configuration	V_{CCIO} = 3.0 V *(9)*	20		50	kΩ
		V_{CCIO} = 2.375 V *(9)*	30		80	kΩ
		V_{CCIO} = 1.71 V *(9)*	60		150	kΩ

Table 6–4. Stratix GX Transceiver Block Absolute Maximum Ratings

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{CCA}	Transceiver block supply voltage	Commercial and industrial	–0.5	4.6	V
V_{CCP}	Transceiver block supply voltage	Commercial and industrial	–0.5	2.4	V
V_{CCR}	Transceiver block supply Voltage	Commercial and industrial	–0.5	2.4	V
V_{CCT}	Transceiver block supply voltage	Commercial and industrial	–0.5	2.4	V
V_{CCG}	Transceiver block supply voltage	Commercial and industrial	–0.5	2.4	V
Receiver input voltage	V_{ICM} ±V_{ID} single / 2	Commercial and industrial		1.675 *(10)*, *(13)*	V
`refclkb` input voltage	V_{ICM} ±V_{ID} single / 2	Commercial and industrial		1.675 *(10)*, *(13)*	V

Table 6–5. Stratix GX Transceiver Block Operating Conditions

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCA}	Transceiver block supply voltage	Commercial and industrial	3.135	3.3	3.465	V
V_{CCP}	Transceiver block supply voltage	Commercial and industrial	1.425	1.5	1.575	V
V_{CCR}	Transceiver block supply voltage	Commercial and industrial	1.425	1.5	1.575	V
V_{CCT}	Transceiver block supply voltage	Commercial and industrial	1.425	1.5	1.575	V
V_{CCG}	Transceiver block supply voltage	Commercial and industrial	1.425	1.5	1.575	V
V_{ID} (differential p-p)	Receiver input differential voltage swing	Commercial and industrial	170		2,000	mV
	`refclkb` input differential voltage swing	Commercial and industrial	400		2,000	mV
V_{ICM}	Receiver input common mode voltage	Commercial and industrial	1,025	1,100	1,175	mV
V_{OD} (differential p-p)	Transmitter output differential voltage	Commercial and industrial	350		1,600	mV
V_{OCM}	Transmitter output common mode voltage	Commercial and industrial		750		mV
R_{REF} *(11)*	Reference resistor	Commercial and industrial	2K –1%	2K	2K +1%	Ω

Table 6–6. Stratix GX Transceiver Block On-Chip Termination (Part 1 of 2)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Rx	Receiver termination	Commercial and industrial, 100-Ω setting	103	108	113	Ω
		Commercial and industrial, 120-Ω setting	120	128	134	Ω
		Commercial and industrial, 150-Ω setting	149	158	167	Ω
Tx	Transmitter termination	Commercial and industrial, 100-Ω setting	103	108	113	Ω
		Commercial and industrial, 120-Ω setting	120	128	134	Ω
		Commercial and industrial, 150-Ω setting	149	158	167	Ω

Table 6–6. Stratix GX Transceiver Block On-Chip Termination (Part 2 of 2)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Refclkb	Dedicated transceiver clock termination	Commercial and industrial, 100-Ω setting	103	108	113	Ω
		Commercial and industrial, 120-Ω setting	120	128	134	Ω
		Commercial and industrial, 150-Ω setting	149	158	167	Ω

Notes to Tables 6–1 through 6–6:

(1) See the *Operating Requirements for Altera Devices Data Sheet*.
(2) Conditions beyond those listed in Table 6–1 may cause permanent damage to a device. Additionally, device operation at the absolute maximum ratings for extended periods of time may have adverse affects on the device.
(3) Minimum DC input is –0.5 V. During transitions, the inputs may undershoot to –2.0 V or overshoot to 4.6 V for input currents less than 100 mA and periods shorter than 20 ns. (The information in this note does not include the transceiver pins. See note 13 for information about the transient voltage on the transceiver pins.)
(4) Maximum V_{CC} rise time is 100 ms, and V_{CC} must rise monotonically.
(5) V_{CCIO} maximum and minimum conditions for LVPECL, LVDS, and 3.3-V PCML are shown in parentheses.
(6) All pins, including dedicated inputs, clock, I/O, and JTAG pins, may be driven before V_{CCINT} and V_{CCIO} are powered.
(7) Typical values are for T_A = 25° C, V_{CCINT} = 1.5 V, and V_{CCIO} = 1.5 V, 1.8 V, 2.5 V, and 3.3 V.
(8) This value is specified for normal device operation. The value may vary during power-up. This applies for all V_{CCIO} settings (3.3, 2.5, 1.8, and 1.5 V).
(9) Pin pull-up resistance values decrease if an external source drives the pin higher than V_{CCIO}.
(10) The device can tolerate prolonged operation at this absolute maximum, as long as the maximum specification is not violated.
(11) Each usable quad requires its own R_{REF} resistor path to ground. For example, the "D" in the EP1SGX25DC1020 device code means it has two usable quad so two different R_{REF} pins must be connected to a R_{REF} resistor each to ground. The DC signal on the R_{REF} pin must be as clean as possible. Ensure that no noise is coupled to this pin.
(12) The Stratix GX device's recommended operating conditions do not include the transceiver. Refer to Tables 6–4 to 6–7.
(13) Minimum DC input to the transceiver pins is –0.5 V. During transitions, the transceiver pins may undershoot to –0.5 V or overshoot to 3.5 V for input currents less than 100 mA and periods shorter than 20 ns.

Table 6–7. Stratix GX Transceiver Block AC Specification (Part 1 of 7)

Symbol / Description	Conditions	-5 Commercial Speed Grade *(1)*			-6 Commercial & Industrial Speed Grade *(1)*			-7 Commercial & Industrial Speed Grade *(1)*			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Power per quadrant (PCS + PMA)	3.125 Gbps, 400-mV V_{od} 0 pre-emphasis		450			450					mW

Table 6–7. Stratix GX Transceiver Block AC Specification (Part 2 of 7)

Symbol / Description	Conditions	-5 Commercial Speed Grade *(1)*			-6 Commercial & Industrial Speed Grade *(1)*			-7 Commercial & Industrial Speed Grade *(1)*			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Reference Clock											
Jitter tolerance (peak-to-peak)	Jitter components <20 MHz			20			20			20	ps
	Wideband			50			50			50	ps
Reference input clock frequency	Dedicated `refclkb` pins	25		650	25		650	25		312.5	MHz
	PLD clock resources	25		325	25		325	25		156.25	MHz
Receiver											
Serial data rate (general)	Commercial / industrial	614		3,187.5	614		3,187.5	614		2,500	Mbps
Serial data rate (8B/10B encoded)	Commercial / industrial	500		3,187.5	500		3,187.5	500		2,500	Mbps
Parallel transceiver/ logic array interface speed		20		398.4	20		375	20		312.5	MHz
Rate matching frequency tolerance	XAUI mode only			±100			±100			±100	ppm
8B/10B Custom Receiver Jitter Tolerance using Encoded CJPAT *Note (2)*											
Deterministic jitter	500 Mbps			0.45			0.45			0.45	UI
Total jitter	500 Mbps			0.71			0.71			0.71	UI
Fibre Channel Receiver Jitter Tolerance using 8B/10B Encoded CJTPAT *Note (2)*											
Deterministic jitter	1.0625 Gbps			0.37			0.37			0.37	UI
Total jitter	1.0625 Gbps			0.68			0.68			0.68	UI

Table 6–7. Stratix GX Transceiver Block AC Specification (Part 3 of 7)

Symbol / Description	Conditions	-5 Commercial Speed Grade *(1)*			-6 Commercial & Industrial Speed Grade *(1)*			-7 Commercial & Industrial Speed Grade *(1)*			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Sinusoidal jitter	f = 42.5 kHz at 1.0625 Gbps			1.5			1.5			1.5	UI
	f = 637 kHz at 1.0625 Gbps			0.1			0.1			0.1	UI
Deterministic jitter	2.125 Gbps			0.33			0.33			0.33	UI
Total jitter	2.125 Gbps			0.62			0.62			0.62	UI
Sinusoidal jitter	f = 85 kHz at 2.125 Gbps			1.5			1.5			1.5	UI
	f = 1,274 kHz at 2.125 Gbps			0.1			0.1			0.1	UI
Serial Rapid I/O Receiver Jitter Tolerance using 8B/10B Encoded CJPAT *Note (2)*											
Deterministic jitter	1.25 Gbps			0.45			0.45			0.45	UI
Total jitter	1.25 Gbps			0.71			0.71			0.71	UI
Deterministic jitter	2.5 Gbps			0.41			0.41			0.41	UI
Total jitter	2.5 Gbps			0.65			0.65			0.65	UI
Deterministic jitter	3.125 Gbps			0.36			0.36			N/A	UI
Total jitter	3.125 Gbps			0.60			0.60			N/A	UI
SONET Receiver Jitter Tolerance using PRBS23 *Note (2)*											
Sinusoidal jitter	f = 6 kHz at 2.48832 Gbps			1.5			1.5			1.5	UI
	f = 1 MHz at 2.48832 Gbps			0.15			0.15			0.15	UI
XAUI Receiver Jitter Tolerance using 8B/10B Encoded CJPAT *Note (2)*											
Deterministic jitter	3.125 Gbps			0.37			0.37			N/A	UI
Total jitter	3.125 Gbps			0.65			0.65			N/A	UI

Table 6–7. Stratix GX Transceiver Block AC Specification (Part 4 of 7)

Symbol / Description	Conditions	-5 Commercial Speed Grade *(1)*			-6 Commercial & Industrial Speed Grade *(1)*			-7 Commercial & Industrial Speed Grade *(1)*			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Sinusoidal jitter	f = 22.1 kHz at 3.125 Gbps			8.5			8.5			N/A	
	f = 1.875 MHz at 3.125 Gbps			0.1			0.1			N/A	
	f = 20 MHz at 3.125 Gbps			0.1			0.1			N/A	
BER *(12)*				10^{-12}			10^{-12}			10^{-12}	
Receive latency *(4)*	Single width	7		32	7		32	7		32	*(3)*
	Double width	5		19	5		19	5		19	*(3)*
Channel to channel bit skew tolerance *(5)*, *(6)*	XAUI mode / inter-quadrant only			40			40			N/A	UI *(7)*
Run-length	*(8)*			80			80			80	UI
Receive return loss (differential)	100 MHz to 2.5 Ghz			–10			–10			–10	dB
Receive return loss (common mode)	100 MHz to 2.5 Ghz			–6			–6			–6	dB
Transmitter											
Serial data rate	Commercial / industrial	500		3,187.5	500		3,187.5	500		2,500	Mbps
Parallel transceiver/ core interface speed		20		398.4	20		375	20		312.5	MHz
8B/10B Custom Transmitter Jitter using Encoded CRPAT *Note (9)*											
Deterministic jitter	500 Mbps Pre-emphasis = 1 V_{OD} = 1,400 mV			0.11			0.11			0.11	UI
Total jitter				0.18			0.18			0.18	UI

Table 6–7. Stratix GX Transceiver Block AC Specification (Part 5 of 7)

Symbol / Description	Conditions	-5 Commercial Speed Grade (1)			-6 Commercial & Industrial Speed Grade (1)			-7 Commercial & Industrial Speed Grade (1)			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Fibre Channel Transmitter Jitter using 8B/10B Encoded CRPAT *Note (9)*											
Deterministic jitter	1.0625 Gbps Pre-emphasis = 0 V_{OD} = 1,200 mV			0.09			0.09			0.09	UI
Total jitter				0.17			0.17			0.17	UI
Deterministic jitter	2.125 Gbps Pre-emphasis= 1 V_{OD} = 1,200 mV			0.16			0.16			0.16	UI
Total jitter				0.33			0.33			0.33	UI
Serial Rapid I/O Short Run Transmitter Jitter using 8B/10B Encoded CRPAT *Note (9)*											
Deterministic jitter	1.25 Gbps Pre-emphasis = 1 V_{OD} = 1,600 mV			0.09			0.09			0.09	UI
Total jitter				0.17			0.17			0.17	UI
Deterministic jitter	2.5 Gbps Pre-emphasis = 1 V_{OD} = 800 mV			0.15			0.15			0.15	UI
Total jitter				0.32			0.32			0.32	UI
Deterministic jitter	3.125 Gbps Pre-emphasis = 1 V_{OD} = 800 mV			0.15			0.15			N/A	UI
Total jitter				0.32			0.32			N/A	UI
Serial Rapid I/O Long Run Transmitter Jitter using 8B/10B Encoded CRPAT *Note (9)*											
Deterministic jitter	1.25 Gbps Pre-emphasis = 1 V_{OD} = 1,600 mV			0.09			0.09			0.09	UI
Total jitter				0.17			0.17			0.17	UI
Deterministic jitter	2.5 Gbps Pre-emphasis = 2 V_{OD} = 1,400 mV			0.18			0.18			0.18	UI
Total jitter				0.35			0.35			0.35	UI
Deterministic jitter	3.125 Gbps Pre-emphasis = 2 V_{OD} = 1,400 mV			0.20			0.20			N/A	UI
Total jitter				0.37			0.37			N/A	UI
SONET Transmitter Jitter PRBS23 *Note (9)*											
Total jitter	2.48832 Gbps Pre-emphasis = 1 V_{OD} = 800 mV			0.20			0.20			0.20	UI

Table 6–7. Stratix GX Transceiver Block AC Specification (Part 6 of 7)

Symbol / Description	Conditions	-5 Commercial Speed Grade *(1)*			-6 Commercial & Industrial Speed Grade *(1)*			-7 Commercial & Industrial Speed Grade *(1)*			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
XAUI Transmitter Jitter using 8B/10B Encoded CJPAT *Note (9)*											
Deterministic jitter	3.125 Gbps Pre-emphasis = 0 V_{OD} = 1,200 mV			0.15			0.15			N/A	UI
Total jitter				0.32			0.32			N/A	UI
Jitter transfer bandwidth *(10)*	Low bandwidth setting at 3.125 Gbps		3			3				N/A	MHz
	High bandwidth setting at 3.125 Gbps		4.7			4.7				N/A	MHz
	Low bandwidth setting at 2.5 Gbps		3.2			3.2				3.2	MHz
	High bandwidth setting at 2.5 Gbps		4.3			4.3				4.3	MHz
Output t_{RISE}	20% to 80%	60		130	60		130	60		130	ps
Output t_{FALL}	80% to 20%	60		130	60		130	60		130	ps
Transmit latency *(11)*	Single width	3		8	3		8	3		8	*(3)*
	Double width	3		7	3		7	3		7	*(3)*
Intra differential pair skew				10			10			10	ps
Channel to channel skew	Within a single quadrant			50			50			50	ps

Table 6–7. Stratix GX Transceiver Block AC Specification (Part 7 of 7)

Symbol / Description	Conditions	-5 Commercial Speed Grade (1)			-6 Commercial & Industrial Speed Grade (1)			-7 Commercial & Industrial Speed Grade (1)			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Output return loss	100 MHz to 2.5 GHz	–10			–10			–10			dB

Notes to Table 6–7:

(1) All numbers for the -6 and -7 speed grades are for both commercial and industrial unless specified otherwise in the Conditions column. Speed grade -5 is available only for commercial specifications.
(2) Not all V_{ID} and equalizer values will get the same results. The condition for the specification was that the V_{ID} before jitter was added is 1,000 mV and the equalizer was set to the maximum condition of 111 (equalizer control setting = 4 in the MegaWizard Plug-In Manager).
(3) Number of parallel clocks.
(4) Receive latency delay from serial receiver indata to parallel receiver data.
(5) Per IEEE Standard 802.3ae @ 3.125 for –5 and –6.
(6) The specification is for channel aligner tolerance.
(7) UI = Unit Interval.
(8) Run-length conditions are true for all data rates, but the average transition density must be enough to keep the receiver phase aligned and the overall data must be DC balanced.
(9) Not all combinations of V_{OD} and pre-emphasis will get the same results.
(10) The numbers are for 3.125-Gbps data rate for –5 and –6 devices and 2.5 Gbps for –7 devices.
(11) Transmitter latency delay from parallel transceiver data to serial transceiver out data.
(12) The receiver operates with a BER of better than 10^{-12} in the presence of an input signal as defined in the XAUI driver template for 3.125 Gbps and in the PCI Exp transmitter eye mask for 2.5 Gbps.

Table 6–8. LVTTL Specifications

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{CCIO}	Output supply voltage		3.0	3.6	V
V_{IH}	High-level input voltage		1.7	4.1	V
V_{IL}	Low-level input voltage		–0.5	0.7	V
V_{OH}	High-level output voltage	I_{OH} = –4 to –24 mA (1)	2.4		V
V_{OL}	Low-level output voltage	I_{OL} = 4 to 24 mA (1)		0.45	V

Table 6–9. LVCMOS Specifications

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{CCIO}	Output supply voltage		3.0	3.6	V
V_{IH}	High-level input voltage		1.7	4.1	V
V_{IL}	Low-level input voltage		–0.5	0.7	V

Table 6–9. LVCMOS Specifications

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{OH}	High-level output voltage	V_{CCIO} = 3.0, I_{OH} = –0.1 mA	V_{CCIO} – 0.2		V
V_{OL}	Low-level output voltage	V_{CCIO} = 3.0, I_{OL} = 0.1 mA		0.2	V

Table 6–10. 2.5-V I/O Specifications *Note (1)*

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{CCIO}	Output supply voltage		2.375	2.625	V
V_{IH}	High-level input voltage		1.7	4.1	V
V_{IL}	Low-level input voltage		–0.5	0.7	V
V_{OH}	High-level output voltage	I_{OH} = –0.1 mA	2.1		V
		I_{OH} = –1 mA	2.0		V
		I_{OH} = –2 to –16 mA *(1)*	1.7		V
V_{OL}	Low-level output voltage	I_{OL} = 0.1 mA		0.2	V
		I_{OH} = 1 mA		0.4	V
		I_{OH} = 2 to 16 mA *(1)*		0.7	V

Table 6–11. 1.8-V I/O Specifications

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{CCIO}	Output supply voltage		1.65	1.95	V
V_{IH}	High-level input voltage		0.65 × V_{CCIO}	2.25	V
V_{IL}	Low-level input voltage		–0.3	0.35 × V_{CCIO}	V
V_{OH}	High-level output voltage	I_{OH} = –2 to –8 mA *(1)*	V_{CCIO} – 0.45		V
V_{OL}	Low-level output voltage	I_{OL} = 2 to 8 mA *(1)*		0.45	V

Table 6–12. 1.5-V I/O Specifications (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{CCIO}	Output supply voltage		1.4	1.6	V
V_{IH}	High-level input voltage		0.65 × V_{CCIO}	V_{CCIO} + 0.3	V
V_{IL}	Low-level input voltage		–0.3	0.35 × V_{CCIO}	V
V_{OH}	High-level output voltage	I_{OH} = –2 mA *(1)*	0.75 × V_{CCIO}		V

Table 6–12. 1.5-V I/O Specifications (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{OL}	Low-level output voltage	I_{OL} = 2 mA *(1)*		0.25 × V_{CCIO}	V

Note to Tables 6–8 through 6–12:

(1) Drive strength is programmable according to values in found in the *Stratix GX Architecture* chapter of the *Stratix GX Device Handbook, Volume 1*.

Figures 6–1 through 6–3 show receiver input and transmitter output waveforms, respectively, for all differential I/O standards (LVDS, 3.3-V PCML, LVPECL, and HyperTransport technology).

Figure 6–1. Receiver Input Waveforms for Differential I/O Standards

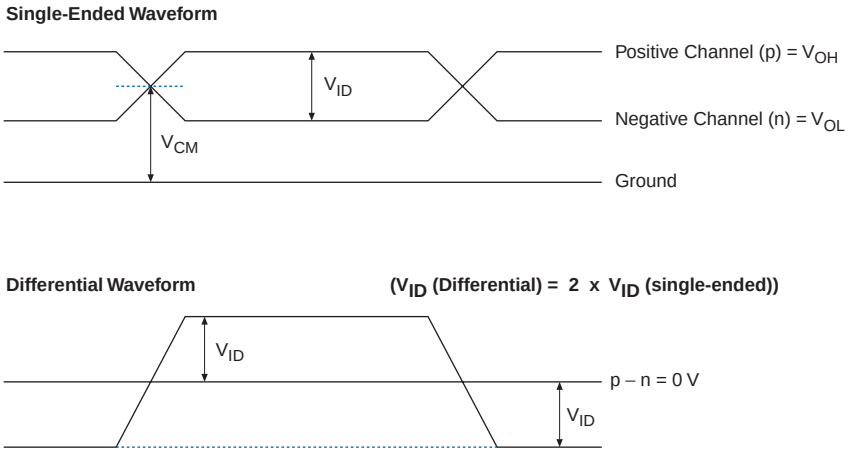
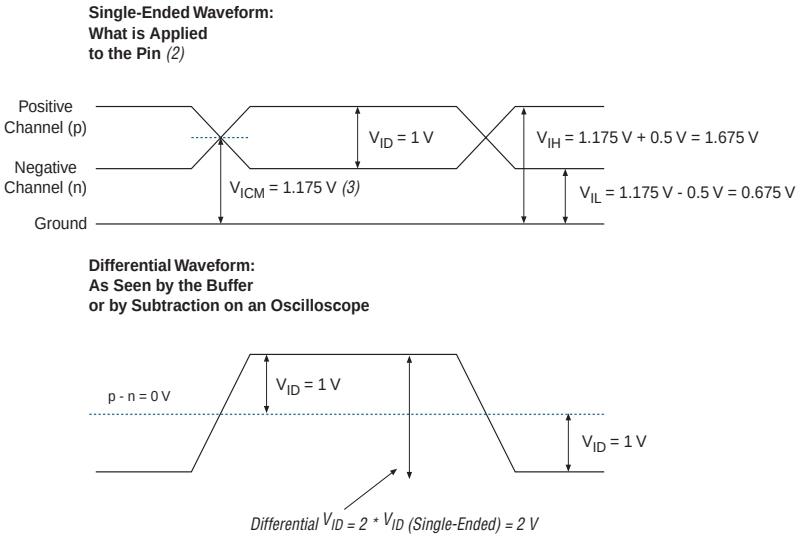


Figure 6–2. Receiver Input Waveform Example with Values



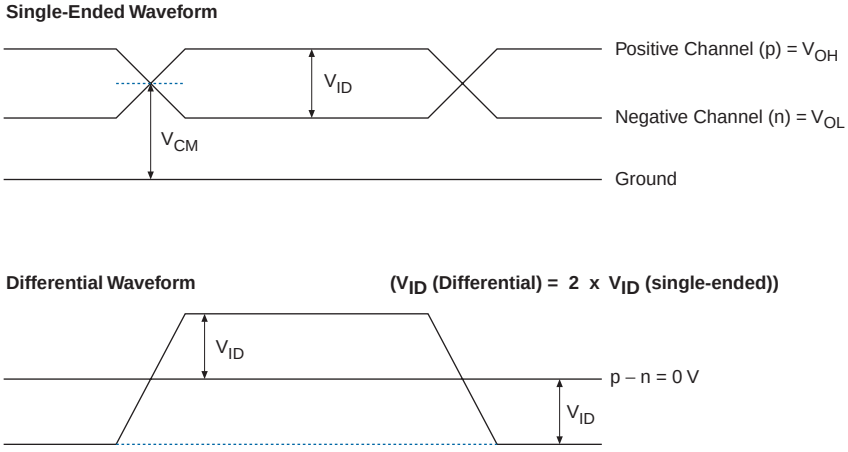
Notes to Figure 6–2:

(1) The values in this figure are for example only.

(2) These values must meet the voltages specified in the section "Operating Conditions" on page 6–1.

(3) If internal termination is used, the common mode is generated after the pins.

Figure 6–3. Transmitter Output Waveforms for Differential I/O Standards



Tables 6–13 through 6–33 provide information about specifications and bus hold parameters for 1.5-V Stratix GX devices. Notes for Tables 6–14 through 6–33 immediately follow Table 6–33.

Table 6–13. 3.3-V LVDS I/O Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	I/O supply voltage		3.135	3.3	3.465	V
V_{ID} *(1)*	Input differential voltage swing (single-ended)	0.1 V < V_{CM} < 1.1 V *W* = 1 through 10	300		1,000	mV
		1.1 V < V_{CM} < 1.6 V *W* = 1	200		1,000	mV
		1.1 V < V_{CM} < 1.6 V *W* = 2 through 10	100		1,000	mV
		1.6 V < V_{CM} < 1.8 V *W* = 1 through 10	300		1,000	mV
V_{ICM} *(1)*	Input common-mode voltage	LVDS 0.3 V < V_{ID} < 1.0 V *W* = 1 through 10	100		1,100	mV
		LVDS 0.3 V < V_{ID} < 1.0 V *W* = 1 through 10	1,600		1,800	mV
		LVDS 0.2 V < V_{ID} < 1.0 V *W* = 1	1,100		1,600	mV
		LVDS 0.1 V < V_{ID} < 1.0 V *W* = 2 through 10	1,100		1,600	mV
V_{OD}	Differential output voltage (single ended)	R_L = 100 Ω	250	375	550	mV
ΔV_{OD}	Change in V_{OD} between high and low	R_L = 100 Ω			50	mV
V_{OCM}	Output common-mode voltage	R_L = 100 Ω	1,125	1,200	1,375	mV
ΔV_{OCM}	Change in V_{OCM} between high and low	R_L = 100 Ω			50	mV
R_L	Receiver differential input resistor, external		90	100	110	Ω

Note to Table 6–13:
(1) For up to 1 Gbps in DPA mode and 840 Mbps in non-DPA mode

Table 6–14. 3.3-V PCML Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	I/O supply voltage		3.135	3.3	3.465	V
V_{ID}	Input differential voltage swing (single-ended)		300		600	mV
V_{ICM}	Input common mode voltage		1.5		3.465	V
V_{OD}	Output differential voltage (single-ended)		300	370	500	mV
ΔV_{OD}	Change in V_{OD} between high and low				50	mV
V_{OCM}	Output common mode voltage		2.5	2.85	3.3	V
ΔV_{OCM}	Change in V_{OCM} between high and low				50	mV
V_T	Output termination voltage			V_{CCIO}		V
R_1	Output external pull-up resistors		45	50	55	Ω
R_2	Output external pull-up resistors		45	50	55	Ω

Table 6–15. LVPECL Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	I/O supply voltage		3.135	3.3	3.465	V
V_{ID}	Input differential voltage swing (single-ended)		300		1,000	mV
V_{ICM}	Input common mode voltage		1		2	V
V_{OD}	Differential output voltage (single ended)	R_L = 100 Ω	525	700	970	mV
V_{OCM}	Output common mode voltage	R_L = 100 Ω	1.5	1.7	1.9	V
R_L	Receiver differential input resistor, external		90	100	110	Ω

Table 6–16. HyperTransport Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	I/O supply voltage		2.375	2.5	2.625	V
V_{OD}	Differential output voltage (single ended)	R_L = 100 Ω	380	485	820	mV
ΔV_{OD}	Change in between high and low	R_L = 100 Ω			50	mV
V_{OCM}	Output common mode voltage	R_L = 100 Ω	440	650	780	mV
ΔV_{OCM}	Change in between high and low	R_L = 100 Ω			50	mV
V_{ID}	Differential input voltage swing (single-ended)		300		900	mV
V_{ICM}	Input common mode voltage		300		900	mV
R_L	Receiver differential input resistor, external		90	100	110	Ω

Table 6–17. 3.3-V PCI Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		3.0	3.3	3.6	V
V_{IH}	High-level input voltage		0.5 × V_{CCIO}		V_{CCIO} + 0.5	V
V_{IL}	Low-level input voltage		–0.5		0.3 × V_{CCIO}	V
V_{OH}	High-level output voltage	I_{OUT} = –500 µA	0.9 × V_{CCIO}			V
V_{OL}	Low-level output voltage	I_{OUT} = 1,500 µA			0.1 × V_{CCIO}	V

Table 6–18. PCI-X Specifications (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		3.0		3.6	V
V_{IH}	High-level input voltage		0.5 × V_{CCIO}		V_{CCIO} + 0.5	V
V_{IL}	Low-level input voltage		–0.5		0.35 × V_{CCIO}	V

Table 6–18. PCI-X Specifications (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{IPU}	Input pull-up voltage		0.7 × V_{CCIO}			V
V_{OH}	High-level output voltage	I_{OUT} = –500 µA	0.9 × V_{CCIO}			V
V_{OL}	Low-level output voltage	I_{OUT} = 1,500 µA			0.1 × V_{CCIO}	V

Table 6–19. GTL+ I/O Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{TT}	Termination voltage		1.35	1.5	1.65	V
V_{REF}	Reference voltage		0.88	1.0	1.12	V
V_{IH}	High-level input voltage		V_{REF} + 0.1			V
V_{IL}	Low-level input voltage				V_{REF} – 0.1	V
V_{OL}	Low-level output voltage	I_{OL} = 36 mA *(1)*			0.65	V

Table 6–20. GTL I/O Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{TT}	Termination voltage		1.14	1.2	1.26	V
V_{REF}	Reference voltage		0.74	0.8	0.86	V
V_{IH}	High-level input voltage		V_{REF} + 0.05			V
V_{IL}	Low-level input voltage				V_{REF} – 0.05	V
V_{OL}	Low-level output voltage	I_{OL} = 40 mA *(1)*			0.4	V

Table 6–21. SSTL-18 Class I Specifications (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		1.65	1.8	1.95	V
V_{REF}	Reference voltage		0.8	0.9	1.0	V
V_{TT}	Termination voltage		V_{REF} – 0.04	V_{REF}	V_{REF} + 0.04	V
$V_{IH(DC)}$	High-level DC input voltage		V_{REF} + 0.125			V

Table 6–21. SSTL-18 Class I Specifications (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
$V_{IL(DC)}$	Low-level DC input voltage				V_{REF} – 0.125	V
$V_{IH(AC)}$	High-level AC input voltage		V_{REF} + 0.275			V
$V_{IL(AC)}$	Low-level AC input voltage				V_{REF} – 0.275	V
V_{OH}	High-level output voltage	I_{OH} = –6.7 mA *(1)*	V_{TT} + 0.475			V
V_{OL}	Low-level output voltage	I_{OL} = 6.7 mA *(1)*			V_{TT} – 0.475	V

Table 6–22. SSTL-18 Class II Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		1.65	1.8	1.95	V
V_{REF}	Reference voltage		0.8	0.9	1.0	V
V_{TT}	Termination voltage		V_{REF} – 0.04	V_{REF}	V_{REF} + 0.04	V
$V_{IH(DC)}$	High-level DC input voltage		V_{REF} + 0.125			V
$V_{IL(DC)}$	Low-level DC input voltage				V_{REF} – 0.125	V
$V_{IH(AC)}$	High-level AC input voltage		V_{REF} + 0.275			V
$V_{IL(AC)}$	Low-level AC input voltage				V_{REF} – 0.275	V
V_{OH}	High-level output voltage	I_{OH} = –13.4 mA *(1)*	V_{TT} + 0.630			V
V_{OL}	Low-level output voltage	I_{OL} = 13.4 mA *(1)*			V_{TT} – 0.630	V

Table 6–23. SSTL-2 Class I Specifications (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		2.375	2.5	2.625	V
V_{TT}	Termination voltage		V_{REF} – 0.04	V_{REF}	V_{REF} + 0.04	V
V_{REF}	Reference voltage		1.15	1.25	1.35	V
V_{IH}	High-level input voltage		V_{REF} + 0.18		3.0	V
V_{IL}	Low-level input voltage		–0.3		V_{REF} – 0.18	V

Table 6–23. SSTL-2 Class I Specifications (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{OH}	High-level output voltage	I_{OH} = –8.1 mA *(1)*	V_{TT} + 0.57			V
V_{OL}	Low-level output voltage	I_{OL} = 8.1 mA *(1)*			V_{TT} – 0.57	V

Table 6–24. SSTL-2 Class II Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		2.3	2.5	2.7	V
V_{TT}	Termination voltage		V_{REF} – 0.04	V_{REF}	V_{REF} + 0.04	V
V_{REF}	Reference voltage		1.15	1.25	1.35	V
V_{IH}	High-level input voltage		V_{REF} + 0.18		V_{CCIO} + 0.3	V
V_{IL}	Low-level input voltage		–0.3		V_{REF} – 0.18	V
V_{OH}	High-level output voltage	I_{OH} = –16.4 mA *(1)*	V_{TT} + 0.76			V
V_{OL}	Low-level output voltage	I_{OL} = 16.4 mA *(1)*			V_{TT} – 0.76	V

Table 6–25. SSTL-3 Class I Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		3.0	3.3	3.6	V
V_{TT}	Termination voltage		V_{REF} – 0.05	V_{REF}	V_{REF} + 0.05	V
V_{REF}	Reference voltage		1.3	1.5	1.7	V
V_{IH}	High-level input voltage		V_{REF} + 0.2		V_{CCIO} + 0.3	V
V_{IL}	Low-level input voltage		–0.3		V_{REF} – 0.2	V
V_{OH}	High-level output voltage	I_{OH} = –8 mA *(1)*	V_{TT} + 0.6			V
V_{OL}	Low-level output voltage	I_{OL} = 8 mA *(1)*			V_{TT} – 0.6	V

Table 6–26. SSTL-3 Class II Specifications (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		3.0	3.3	3.6	V
V_{TT}	Termination voltage		V_{REF} – 0.05	V_{REF}	V_{REF} + 0.05	V
V_{REF}	Reference voltage		1.3	1.5	1.7	V
V_{IH}	High-level input voltage		V_{REF} + 0.2		V_{CCIO} + 0.3	V

Table 6–26. SSTL-3 Class II Specifications (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{IL}	Low-level input voltage		–0.3		V_{REF} – 0.2	V
V_{OH}	High-level output voltage	I_{OH} = –16 mA *(1)*	V_{TT} + 0.8			V
V_{OL}	Low-level output voltage	I_{OL} = 16 mA *(1)*			V_{TT} – 0.8	V

Table 6–27. 3.3-V AGP 2× Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		3.15	3.3	3.45	V
V_{REF}	Reference voltage		0.39 × V_{CCIO}		0.41 × V_{CCIO}	V
V_{IH}	High-level input voltage *(2)*		0.5 × V_{CCIO}		V_{CCIO} + 0.5	V
V_{IL}	Low-level input voltage *(2)*				0.3 × V_{CCIO}	V
V_{OH}	High-level output voltage	I_{OUT} = –0.5 mA	0.9 × V_{CCIO}		3.6	V
V_{OL}	Low-level output voltage	I_{OUT} = 1.5 mA			0.1 × V_{CCIO}	V

Table 6–28. 3.3-V AGP 1× Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		3.15	3.3	3.45	V
V_{IH}	High-level input voltage *(2)*		0.5 × V_{CCIO}		V_{CCIO} + 0.5	V
V_{IL}	Low-level input voltage *(2)*				0.3 × V_{CCIO}	V
V_{OH}	High-level output voltage	I_{OUT} = –0.5 mA	0.9 × V_{CCIO}		3.6	V
V_{OL}	Low-level output voltage	I_{OUT} = 1.5 mA			0.1 × V_{CCIO}	V

Table 6–29. 1.5-V HSTL Class I Specifications (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		1.4	1.5	1.6	V
V_{REF}	Input reference voltage		0.68	0.75	0.9	V
V_{TT}	Termination voltage		0.7	0.75	0.8	V
V_{IH} (DC)	DC high-level input voltage		V_{REF} + 0.1			V
V_{IL} (DC)	DC low-level input voltage		–0.3		V_{REF} – 0.1	V
V_{IH} (AC)	AC high-level input voltage		V_{REF} + 0.2			V
V_{IL} (AC)	AC low-level input voltage				V_{REF} – 0.2	V

Table 6–29. 1.5-V HSTL Class I Specifications (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{OH}	High-level output voltage	I_{OH} = 8 mA *(1)*	V_{CCIO} – 0.4			V
V_{OL}	Low-level output voltage	I_{OH} = –8 mA *(1)*			0.4	V

Table 6–30. 1.5-V HSTL Class II Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		1.4	1.5	1.6	V
V_{REF}	Input reference voltage		0.68	0.75	0.9	V
V_{TT}	Termination voltage		0.7	0.75	0.8	V
V_{IH} (DC)	DC high-level input voltage		V_{REF} + 0.1			V
V_{IL} (DC)	DC low-level input voltage		–0.3		V_{REF} – 0.1	V
V_{IH} (AC)	AC high-level input voltage		V_{REF} + 0.2			V
V_{IL} (AC)	AC low-level input voltage				V_{REF} – 0.2	V
V_{OH}	High-level output voltage	I_{OH} = 16 mA *(1)*	V_{CCIO} – 0.4			V
V_{OL}	Low-level output voltage	I_{OH} = –16 mA *(1)*			0.4	V

Table 6–31. 1.5-V Differential HSTL Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	I/O supply voltage		1.4	1.5	1.6	V
V_{DIF} (DC)	DC input differential voltage		0.2			V
V_{CM} (DC)	DC common mode input voltage		0.68		0.9	V
V_{DIF} (AC)	AC differential input voltage		0.4			V

Table 6–32. CTT I/O Specifications (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		3.0	3.3	3.6	V
V_{TT}/V_{REF}	Termination and input reference voltage		1.35	1.5	1.65	V
V_{IH}	High-level input voltage		V_{REF} + 0.2			V
V_{IL}	Low-level input voltage				V_{REF} – 0.2	V

Table 6–32. CTT I/O Specifications (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{OH}	High-level output voltage	I_{OH} = –8 mA	V_{REF} + 0.4			V
V_{OL}	Low-level output voltage	I_{OL} = 8 mA			V_{REF} – 0.4	V
I_O	Output leakage current (when output is high *Z*)	GND ≤ V_{OUT} ≤ V_{CCIO}	–10		10	µA

Table 6–33. Bus Hold Parameters

Parameter	Conditions	V_{CCIO} Level								Units
		1.5 V		1.8 V		2.5 V		3.3 V		
		Min	Max	Min	Max	Min	Max	Min	Max	
Low sustaining current	V_{IN} > V_{IL} (maximum)	25		30		50		70		µA
High sustaining current	V_{IN} < V_{IH} (minimum)	–25		–30		–50		–70		µA
Low overdrive current	0 V < V_{IN} < V_{CCIO}		160		200		300		500	µA
High overdrive current	0 V < V_{IN} < V_{CCIO}		–160		–200		–300		–500	µA
Bus-hold trip point		0.5	1.0	0.68	1.07	0.7	1.7	0.8	2.0	V

Notes to Tables 6–14 through 6–33:

(1) Drive strength is programmable according to values in the *Stratix GX Architecture* chapter of the *Stratix GX Device Handbook, Volume 1*.

(2) V_{REF} specifies the center point of the switching range.

Power Consumption

Detailed power consumption information for Stratix GX devices will be released when available.

Timing Model

The DirectDrive™ technology and MultiTrack™ interconnect ensure predictable performance, accurate simulation, and accurate timing analysis across all Stratix GX device densities and speed grades. This section describes and specifies the performance, internal, external, and PLL timing specifications.

All specifications are representative of worst-case supply voltage and junction temperature conditions.

Preliminary & Final Timing

Timing models can have either preliminary or final status. The Quartus® II software displays an informational message during the design compilation if the timing models are preliminary. Table 6–34 shows the status of the Stratix GX device timing models.

Preliminary status means the timing model is subject to change. Initially, timing numbers are created using simulation results, process data, and other known parameters. These tests are used to make the preliminary numbers as close to the actual timing parameters as possible.

Final timing numbers are based on actual device operation and testing. These numbers reflect the actual performance of the device under worst-case voltage and junction temperature conditions.

Table 6–34. Stratix GX Device Timing Model Status

Device	Preliminary	Final
EP1SGX10	—	✓
EP1SGX25	—	✓
EP1SGX40	—	✓

Performance

Table 6–35 shows Stratix GX device performance for some common designs. All performance values were obtained with Quartus II software compilation of LPM, or MegaCore® functions for the FIR and FFT designs.

Table 6–35. Stratix GX Device Performance (Part 1 of 3) *Notes (1), (2)*

Applications		Resources Used			Performance			
		LEs	TriMatrix Memory Blocks	DSP Blocks	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Units
LE	16-to-1 multiplexer *(1)*	22	0	0	407.83	324.56	288.68	MHz
	32-to-1 multiplexer *(3)*	46	0	0	318.26	255.29	242.89	MHz
	16-bit counter	16	0	0	422.11	422.11	390.01	MHz
	64-bit counter	64	0	0	321.85	290.52	261.23	MHz

Table 6–35. Stratix GX Device Performance (Part 2 of 3) *Notes (1), (2)*

Applications		Resources Used			Performance			
		LEs	TriMatrix Memory Blocks	DSP Blocks	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Units
TriMatrix memory M512 block	Simple dual-port RAM 32 × 18 bit	0	1	0	317.76	277.62	241.48	MHz
	FIFO 32 × 18 bit	30	1	0	319.18	278.86	242.54	MHz
TriMatrix memory M4K block	Simple dual-port RAM 128 × 36 bit	0	1	0	290.86	255.55	222.27	MHz
	True dual-port RAM 128 × 18 bit	0	1	0	290.86	255.55	222.27	MHz
	FIFO 128 × 36 bit	34	1	0	290.86	255.55	222.27	MHz
TriMatrix memory M-RAM block	Single port RAM 4K × 144 bit	1	1	0	255.95	223.06	194.06	MHz
	Simple dual-port RAM 4K × 144 bit	0	1	0	255.95	233.06	194.06	MHz
	True dual-port RAM 4K × 144 bit	0	1	0	255.95	233.06	194.06	MHz
	Single port RAM 8K × 72 bit	0	1	0	278.94	243.19	211.59	MHz
	Simple dual-port RAM 8K × 72 bit	0	1	0	255.95	223.06	194.06	MHz
	True dual-port RAM 8K × 72 bit	0	1	0	255.95	223.06	194.06	MHz
	Single port RAM 16K × 36 bit	0	1	0	280.66	254.32	221.28	MHz
	Simple dual-port RAM 16K × 36 bit	0	1	0	269.83	237.69	206.82	MHz

Table 6–35. Stratix GX Device Performance (Part 3 of 3) *Notes (1), (2)*

Applications		Resources Used			Performance			
		LEs	TriMatrix Memory Blocks	DSP Blocks	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Units
TriMatrix memory M-RAM block	True dual-port RAM 16K × 36 bit	0	1	0	269.83	237.69	206.82	MHz
	Single port RAM 32K × 18 bit	0	1	0	275.86	244.55	212.76	MHz
	Simple dual-port RAM 32K × 18 bit	0	1	0	275.86	244.55	212.76	MHz
	True dual-port RAM 32K × 18 bit	0	1	0	275.86	244.55	212.76	MHz
	Single port RAM 64K × 9 bit	0	1	0	287.85	253.29	220.36	MHz
	Simple dual-port RAM 64K × 9 bit	0	1	0	287.85	253.29	220.36	MHz
	True dual-port RAM 64K × 9 bit	0	1	0	287.85	253.29	220.36	MHz
DSP block	9 × 9-bit multiplier *(3)*	0	0	1	335.0	293.94	255.68	MHz
	18 × 18-bit multiplier *(4)*	0	0	1	278.78	237.41	206.52	MHz
	36 × 36-bit multiplier *(4)*	0	0	1	148.25	134.71	117.16	MHz
	36 × 36-bit multiplier *(5)*	0	0	1	278.78	237.41	206.52	MHz
	18-bit, 4-tap FIR filter	0	0	1	278.78	237.41	206.52	MHz
Larger Designs	8-bit, 16-tap parallel FIR filter	58	0	4	141.26	133.49	114.88	MHz
	8-bit, 1,024-point FFT function	870	5	1	261.09	235.51	205.21	MHz

Notes to Table 6–35:

(1) These design performance numbers were obtained using the Quartus II software.
(2) Numbers not listed will be included in a future version of the data sheet.
(3) This application uses registered inputs and outputs.
(4) This application uses registered multiplier input and output stages within the DSP block.
(5) This application uses registered multiplier input, pipeline, and output stages within the DSP block.

Internal Timing Parameters

Internal timing parameters are specified on a speed grade basis independent of device density. Tables 6–36 through 6–42 describe the Stratix GX device internal timing microparameters for LEs, IOEs, TriMatrix™ memory structures, DSP blocks, and MultiTrack interconnects.

Table 6–36. LE Internal Timing Microparameter Descriptions

Symbol	Parameter
t_{SU}	LE register setup time before clock
t_{H}	LE register hold time after clock
t_{CO}	LE register clock-to-output delay
t_{LUT}	LE combinational LUT delay for data-in to data-out
t_{CLR}	Minimum clear pulse width
t_{PRE}	Minimum preset pulse width
t_{CLKHL}	Minimum clock high or low time

Table 6–37. IOE Internal Timing Microparameter Descriptions

Symbol	Parameter
t_{SU}	IOE input and output register setup time before clock
t_{H}	IOE input and output register hold time after clock
t_{CO}	IOE input and output register clock-to-output delay
$t_{PIN2COMBOUT_R}$	Row input pin to IOE combinational output
$t_{PIN2COMBOUT_C}$	Column input pin to IOE combinational output
$t_{COMBIN2PIN_R}$	Row IOE data input to combinational output pin
$t_{COMBIN2PIN_C}$	Column IOE data input to combinational output pin
t_{CLR}	Minimum clear pulse width
t_{PRE}	Minimum preset pulse width
t_{CLKHL}	Minimum clock high or low time

Table 6–38. DSP Block Internal Timing Microparameter Descriptions	
Symbol	**Parameter**
t_{SU}	Input, pipeline, and output register setup time before clock
t_{H}	Input, pipeline, and output register hold time after clock
t_{CO}	Input, pipeline, and output register clock-to-output delay
$t_{INREG2PIPE9}$	Input register to DSP block pipeline register in 9 × 9-bit mode
$t_{INREG2PIPE18}$	Input register to DSP block pipeline register in 18 × 18-bit mode
$t_{PIPE2OUTREG2ADD}$	DSP block pipeline register to output register delay in two-multipliers adder mode
$t_{PIPE2OUTREG4ADD}$	DSP Block Pipeline Register to output register delay in four-multipliers adder mode
t_{PD9}	Combinational input to output delay for 9 × 9-bit mode
t_{PD18}	Combinational input to output delay for 18 × 18-bit mode
t_{PD36}	Combinational input to output delay for 36 × 36-bit mode
t_{CLR}	Minimum clear pulse width
t_{CLKHL}	Minimum clock high or low time

Table 6–39. M512 Block Internal Timing Microparameter Descriptions	
Symbol	**Parameter**
t_{M512RC}	Synchronous read cycle time
t_{M512WC}	Synchronous write cycle time
$t_{M512WERESU}$	Write or read enable setup time before clock
$t_{M512WEREH}$	Write or read enable hold time after clock
$t_{M512DATASU}$	Data setup time before clock
$t_{M512DATAH}$	Data hold time after clock
$t_{M512WADDRSU}$	Write address setup time before clock
$t_{M512WADDRH}$	Write address hold time after clock
$t_{M512RADDRSU}$	Read address setup time before clock
$t_{M512RADDRH}$	Read address hold time after clock
$t_{M512DATACO1}$	Clock-to-output delay when using output registers
$t_{M512DATACO2}$	Clock-to-output delay without output registers
$t_{M512CLKHL}$	Minimum clock high or low time
$t_{M512CLR}$	Minimum clear pulse width

Table 6–40. M4K Block Internal Timing Microparameter Descriptions	
Symbol	**Parameter**
t_{M4KRC}	Synchronous read cycle time
t_{M4KWC}	Synchronous write cycle time
$t_{M4KWERESU}$	Write or read enable setup time before clock
$t_{M4KWEREH}$	Write or read enable hold time after clock
$t_{M4KBESU}$	Byte enable setup time before clock
t_{M4KBEH}	Byte enable hold time after clock
$t_{M4KDATAASU}$	A port data setup time before clock
$t_{M4KDATAAH}$	A port data hold time after clock
$t_{M4KADDRASU}$	A port address setup time before clock
$t_{M4KADDRAH}$	A port address hold time after clock
$t_{M4KDATABSU}$	B port data setup time before clock
$t_{M4KDATABH}$	B port data hold time after clock
$t_{M4KADDRBSU}$	B port address setup time before clock
$t_{M4KADDRBH}$	B port address hold time after clock
$t_{M4KDATACO1}$	Clock-to-output delay when using output registers
$t_{M4KDATACO2}$	Clock-to-output delay without output registers
$t_{M4KCLKHL}$	Minimum clock high or low time
t_{M4KCLR}	Minimum clear pulse width

Table 6–41. M-RAM Block Internal Timing Microparameter Descriptions (Part 1 of 2)	
Symbol	**Parameter**
t_{MRAMRC}	Synchronous read cycle time
t_{MRAMWC}	Synchronous write cycle time
$t_{MRAMWERESU}$	Write or read enable setup time before clock
$t_{MRAMWEREH}$	Write or read enable hold time after clock
$t_{MRAMBESU}$	Byte enable setup time before clock
$t_{MRAMBEH}$	Byte enable hold time after clock
$t_{MRAMDATAASU}$	A port data setup time before clock
$t_{MRAMDATAAH}$	A port data hold time after clock
$t_{MRAMADDRASU}$	A port address setup time before clock
$t_{MRAMADDRAH}$	A port address hold time after clock

Table 6–41. M-RAM Block Internal Timing Microparameter Descriptions (Part 2 of 2)

Symbol	Parameter
$t_{MRAMDATABSU}$	B port setup time before clock
$t_{MRAMDATABH}$	B port hold time after clock
$t_{MRAMADDRBSU}$	B port address setup time before clock
$t_{MRAMADDRBH}$	B port address hold time after clock
$t_{MRAMDATACO1}$	Clock-to-output delay when using output registers
$t_{MRAMDATACO2}$	Clock-to-output delay without output registers
$t_{MRAMCLKHL}$	Minimum clock high or low time
$t_{MRAMCLR}$	Minimum clear pulse width

Table 6–42. Routing Delay Internal Timing Microparameter Descriptions

Symbol	Parameter
t_{R4}	Delay for an R4 line with average loading; covers a distance of four LAB columns
t_{R8}	Delay for an R8 line with average loading; covers a distance of eight LAB columns
t_{R24}	Delay for an R24 line with average loading; covers a distance of 24 LAB columns
t_{C4}	Delay for an C4 line with average loading; covers a distance of four LAB rows
t_{C8}	Delay for an C8 line with average loading; covers a distance of eight LAB rows
t_{C16}	Delay for an C16 line with average loading; covers a distance of 16 LAB rows
t_{LOCAL}	Local interconnect delay

Table 6–43. Stratix GX Reset & PLL Lock Time Parameter Descriptions (Part 1 of 2)

Symbol	Parameter
$t_{ANALOGRESETPW}$	Pulse width to power down analog circuits.
$t_{DIGITALRESETPW}$	Pulse width to reset digital circuits
$t_{TX_PLL_LOCK}$	The time it takes the `tx_pll` to lock to the reference clock.

Table 6–43. Stratix GX Reset & PLL Lock Time Parameter Descriptions (Part 2 of 2)	
Symbol	**Parameter**
$t_{RX_FREQLOCK}$	The time until the clock recovery unit (CRU) switches to data mode from lock to reference mode.
$t_{RX_FREQLOCK2PHASELOCK}$	The time until CRU phase locks to data after switching from lock to data mode.

Figure 6–4 shows the TriMatrix memory waveforms for the M512, M4K, and M-RAM timing parameters shown in Tables 6–39 through 6–41.

Figure 6–4. Dual-Port RAM Timing Microparameter Waveform

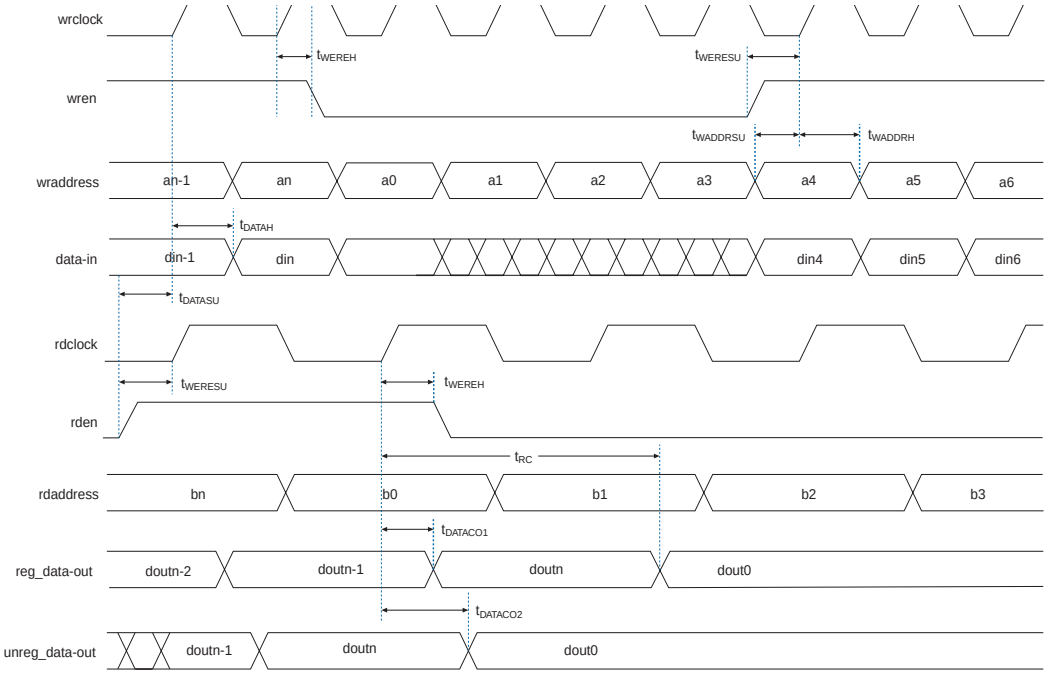
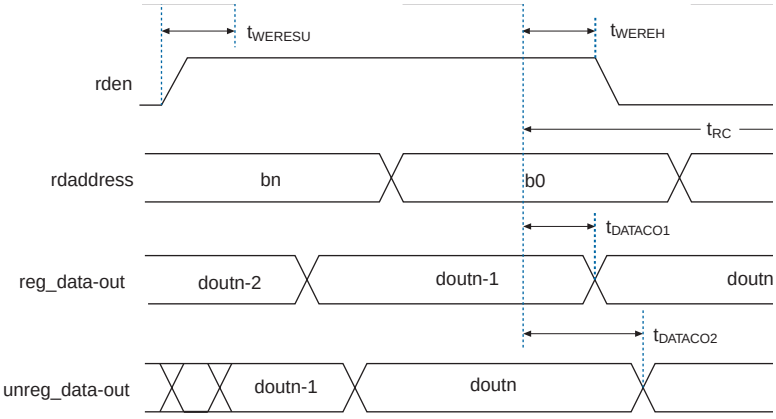


Figure 6–5. Stratix GX Transceiver Reset & PLL Lock Time Waveform *Note (1)*



Note to Figure 6–5:

(1) Waveforms are for minimum pulse width timing and output timing only. Please refer to the *Stratix GX Transceiver User Guide* for the complete reset sequence.

Tables 6–44 through 6–50 show the internal timing microparameters for all Stratix GX devices.

Table 6–44. LE Internal Timing Microparameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{SU}	10		10		11		ps
t_{H}	100		100		114		ps
t_{CO}		156		176		202	ps
t_{LUT}		366		459		527	ps
t_{CLR}	100		100		114		ps
t_{PRE}	100		100		114		ps
t_{CLKHL}	100		100		114		ps

Table 6–45. IOE Internal Timing Microparameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{SU}	64		68		68		ps
t_{H}	76		80		80		ps
t_{CO}		162		171		171	ps
$t_{PIN2COMBOUT_R}$		1,038		1,093		1,256	ps
$t_{PIN2COMBOUT_C}$		927		976		1,122	ps
$t_{COMBIN2PIN_R}$		2,944		3,099		3,563	ps
$t_{COMBIN2PIN_C}$		3,189		3,357		3,860	ps
t_{CLR}	262		276		317		ps
t_{PRE}	262		276		317		ps
t_{CLKHL}	90		95		109		ps

Table 6–46. DSP Block Internal Timing Microparameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{SU}	0		0		0		ps
t_{H}	67		75		86		ps
t_{CO}		142		158		181	ps
$t_{INREG2PIPE18}$		2,613		2,982		3,429	ps
$t_{INREG2PIPE9}$		3,390		3,993		4,591	ps
$t_{PIPE2OUTREG2ADD}$		2,002		2,203		2,533	ps
$t_{PIPE2OUTREG4ADD}$		2,899		3,189		3,667	ps
t_{PD9}		3,709		4,081		4,692	ps
t_{PD18}		4,795		5,275		6,065	ps
t_{PD36}		7,495		8,245		9,481	ps
t_{CLR}	450		500		575		ps
t_{CLKHL}	1,350		1,500		1,724		ps

Table 6–47. M512 Block Internal Timing Microparameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{M512RC}		3,340		3,816		4,387	ps
t_{M512WC}		3,318		3,590		4,128	ps
$t_{M512WERESU}$	110		123		141		ps
$t_{M512WERH}$	34		38		43		ps
$t_{M512DATASU}$	110		123		141		ps
$t_{M512DATAH}$	34		38		43		ps
$t_{M512WADDRASU}$	110		123		141		ps
$t_{M512WADDRH}$	34		38		43		ps
$t_{M512DATACO1}$		424		472		541	ps
$t_{M512DATACO2}$		3,366		3,846		4,421	ps
$t_{M512CLKHL}$	150		167		192		ps
$t_{M512CLR}$	170		189		217		ps

Table 6–48. M4K Block Internal Timing Microparameters (Part 1 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{M4KRC}		3,807		4,320		4,967	ps
t_{M4KWC}		2,556		2,840		3,265	ps
$t_{M4KWERESU}$	131		149		171		ps
$t_{M4KWERH}$	34		38		43		ps
$t_{M4KDATASU}$	131		149		171		ps
$t_{M4KDATAH}$	34		38		43		ps
$t_{M4KWADDRASU}$	131		149		171		ps
$t_{M4KWADDRH}$	34		38		43		ps
$t_{M4KRADDRASU}$	131		149		171		ps
$t_{M4KRADDRH}$	34		38		43		ps
$t_{M4KDATABSU}$	131		149		171		ps
$t_{M4KDATABH}$	34		38		43		ps
$t_{M4KADDRBSU}$	131		149		171		ps
$t_{M4KADDRBH}$	34		38		43		ps

Table 6–48. M4K Block Internal Timing Microparameters (Part 2 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
$t_{M4KDATACO1}$		571		635		729	ps
$t_{M4KDATACO2}$		3,984		4,507		5,182	ps
$t_{M4KCLKHL}$	150		167		192		ps
t_{M4KCLR}	170		189		255		ps

Table 6–49. M-RAM Block Internal Timing Microparameters

Symbol	-5		-6		-7		Unit
	Min	Max	Min	Max	Min	Max	
t_{MRAMRC}		4,364		4,838		5,562	ps
t_{MRAMWC}		3,654		4,127		4,746	ps
$t_{MRAMWERESU}$	25		25		28		ps
$t_{MRAMWERH}$	18		20		23		ps
$t_{MRAMDATASU}$	25		25		28		ps
$t_{MRAMDATAH}$	18		20		23		ps
$t_{MRAMWADDRASU}$	25		25		28		ps
$t_{MRAMWADDRH}$	18		20		23		ps
$t_{MRAMRADDRASU}$	25		25		28		ps
$t_{MRAMRADDRH}$	18		20		23		ps
$t_{MRAMDATABSU}$	25		25		28		ps
$t_{MRAMDATABH}$	18		20		23		ps
$t_{MRAMADDRBSU}$	25		25		28		ps
$t_{MRAMADDRBH}$	18		20		23		ps
$t_{MRAMDATACO1}$		1,038		1,053		1,210	ps
$t_{MRAMDATACO2}$		4,362		4,939		5,678	ps
$t_{MRAMCLKHL}$	270		300		345		ps
$t_{MRAMCLR}$	135		150		172		ps

Table 6–50. Stratix GX Transceiver Reset & PLL Lock Time Parameters

Symbol	Min	Typ	Max	Units
$t_{ANALOGRESETPW}$ (5)	1			mS
$t_{DIGITALRESETPW}$ (5)	4			Parallel clock cycle
$t_{TX_PLL_LOCK}$ (3)			10	µS
$t_{RX_FREQLOCK}$ (4)			5	mS
$t_{RX_FREQLOCK2PHASELOCK}$ (2)			5	µS

Notes to Table 6–50:

(1) The minimum pulse width specified is associated with the power-down of circuits.

(2) The clock recovery unit (CRU) phase locked-to-data time is based on a data rate of 500 Mbps and 8B/10B encoded data.

(3) After `#pll_areset`, `pll_enable`, or PLL power-up, the time required for the transceiver PLL to lock to the reference clock.

(4) After `#rx_analogreset`, the time for the CRU to switch to lock-to-data mode.

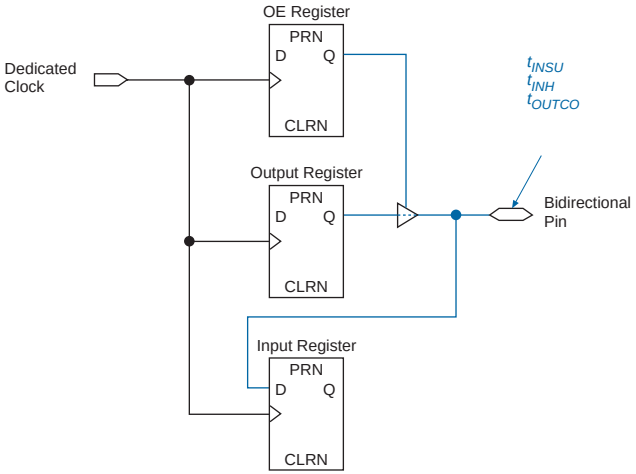
(5) There is no maximum pulse width specification. The GXB can be held in reset indefinitely.

Routing delays vary depending on the load on a specific routing line. The Quartus II software reports the routing delay information when running the timing analysis for a design. Contact Altera Applications Engineering for more details.

External Timing Parameters

External timing parameters are specified by device density and speed grade. Figure 6–6 shows the timing model for bidirectional IOE pin timing. All registers are within the IOE.

Figure 6–6. External Timing in Stratix GX Devices



All external I/O timing parameters shown are for 3.3-V LVTTL or LVCMOS I/O standards with the maximum current strength. For external I/O timing using standards other than LVTTL or LVCMOS use the I/O standard input and output delay adders in Tables 6–72 through 6–76.

Table 6–51 shows the external I/O timing parameters when using fast regional clock networks.

Table 6–51. Stratix GX Fast Regional Clock External I/O Timing Parameters *Notes (1), (2)*

Symbol	Parameter	Conditions
t_{INSU}	Setup time for input or bidirectional pin using column IOE input register with fast regional clock fed by FCLK pin	
t_{INH}	Hold time for input or bidirectional pin using column IOE input register with fast regional clock fed by FCLK pin	
t_{OUTCO}	Clock-to-output delay output or bidirectional pin using column IOE output register with fast regional clock fed by FCLK pin	C_{LOAD} = 10 pF

Notes to Table 6–51:
(1) These timing parameters are sample-tested only.
(2) These timing parameters are for column IOE pins. Row IOE pins are 100- to 250-ps slower depending on device and speed grade and whether it is t_{CO} or t_{SU}. You should use the Quartus II software to verify the external timing for any pin.

Table 6–52 shows the external I/O timing parameters when using regional clock networks.

Table 6–52. Stratix GX Regional Clock External I/O Timing Parameters *Notes (1), (2)*		
Symbol	**Parameter**	**Conditions**
t_{INSU}	Setup time for input or bidirectional pin using column IOE input register with regional clock fed by CLK pin	
t_{INH}	Hold time for input or bidirectional pin using column IOE input register with regional clock fed by CLK pin	
t_{OUTCO}	Clock-to-output delay output or bidirectional pin using column IOE output register with regional clock fed by CLK pin	C_{LOAD} = 10 pF
$t_{INSUPLL}$	Setup time for input or bidirectional pin using column IOE input register with regional clock fed by Enhanced PLL with default phase setting	
t_{INHPLL}	Hold time for input or bidirectional pin using column IOE input register with regional clock fed by Enhanced PLL with default phase setting	
$t_{OUTCOPLL}$	Clock-to-output delay output or bidirectional pin using column IOE output register with regional clock Enhanced PLL with default phase setting	C_{LOAD} = 10 pF

Notes to Table 6–52:
(1) These timing parameters are sample-tested only.
(2) These timing parameters are for column IOE pins. Row IOE pins are 100- to 250-ps slower depending on device, speed grade, and the specific parameter in question. You should use the Quartus II software to verify the external timing for any pin.

Table 6–53 shows the external I/O timing parameters when using global clock networks.

Table 6–53. Stratix GX Global Clock External I/O Timing Parameters (Part 1 of 2) *Notes (1), (2)*		
Symbol	**Parameter**	**Conditions**
t_{INSU}	Setup time for input or bidirectional pin using column IOE input register with global clock fed by CLK pin	
t_{INH}	Hold time for input or bidirectional pin using column IOE input register with global clock fed by CLK pin	
t_{OUTCO}	Clock-to-output delay output or bidirectional pin using column IOE output register with global clock fed by CLK pin	C_{LOAD} = 10 pF
$t_{INSUPLL}$	Setup time for input or bidirectional pin using column IOE input register with global clock fed by Enhanced PLL with default phase setting	

Table 6–53. Stratix GX Global Clock External I/O Timing Parameters (Part 2 of 2) Notes (1), (2)		
Symbol	**Parameter**	**Conditions**
t_{INHPLL}	Hold time for input or bidirectional pin using column IOE input register with global clock fed by enhanced PLL with default phase setting	
$t_{OUTCOPLL}$	Clock-to-output delay output or bidirectional pin using column IOE output register with global clock enhanced PLL with default phase setting	C_{LOAD} = 10 pF

Notes to Table 6–53:
(1) These timing parameters are sample-tested only.
(2) These timing parameters are for column IOE pins. Row IOE pins are 100- to 250-ps slower depending on device, speed grade, and the specific parameter in question. You should use the Quartus II software to verify the external timing for any pin.

Tables 6–54 through 6–59 show the external timing parameters on column and row pins for EP1SGX10 devices.

Table 6–54. EP1SGX10 Column Pin Fast Regional Clock External I/O Timing Parameters							
Symbol	**-5 Speed Grade**		**-6 Speed Grade**		**-7 Speed Grade**		**Unit**
	Min	**Max**	**Min**	**Max**	**Min**	**Max**	
t_{INSU}	2.245		2.332		2.666		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.597	2.000	4.920	2.000	5.635	ns

Table 6–55. EP1SGX10 Column Pin Regional Clock External I/O Timing Parameters							
Symbol	**-5 Speed Grade**		**-6 Speed Grade**		**-7 Speed Grade**		**Unit**
	Min	**Max**	**Min**	**Max**	**Min**	**Max**	
t_{INSU}	2.114		2.218		2.348		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.728	2.000	5.078	2.000	6.004	ns
$t_{INSUPLL}$	1.035		0.941		1.070		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.629	0.500	2.769	0.500	3.158	ns

Table 6–56. EP1SGX10 Column Pin Global Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	1.785		1.814		2.087		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	5.057	2.000	5.438	2.000	6.214	ns
$t_{INSUPLL}$	0.988		0.936		1.066		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.634	0.500	2.774	0.500	3.162	ns

Table 6–57. EP1SGX10 Row Pin Fast Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.194		2.384		2.727		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.956	2.000	4.971	2.000	5.463	ns

Table 6–58. EP1SGX10 Row Pin Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.244		2.413		2.574		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.906	2.000	4.942	2.000	5.616	ns
$t_{INSUPLL}$	1.126		1.186		1.352		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.804	0.500	2.627	0.500	2.765	ns

Table 6–59. EP1SGX10 Row Pin Global Clock External I/O Timing Parameters (Part 1 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	1.919		2.062		2.368		ns
t_{INH}	0.000		0.000		0.000		ns

Table 6–59. EP1SGX10 Row Pin Global Clock External I/O Timing Parameters (Part 2 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{OUTCO}	2.000	5.231	2.000	5.293	2.000	5.822	ns
$t_{INSUPLL}$	1.126		1.186		1.352		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.804	0.500	2.627	0.500	2.765	ns

Tables 6–60 through 6–65 show the external timing parameters on column and row pins for EP1SGX25 devices.

Table 6–60. EP1SGX25 Column Pin Fast Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.418		2.618		3.014		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.524	2.000	4.834	2.000	5.538	ns

Table 6–61. EP1SGX25 Column Pin Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	1.713		1.838		2.069		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	5.229	2.000	5.614	2.000	6.432	ns
$t_{INSUPLL}$	1.061		1.155		1.284		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.661	0.500	2.799	0.500	3.195	ns

Table 6–62. EP1SGX25 Column Pin Global Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	1.790		1.883		2.120		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	5.194	2.000	5.569	2.000	6.381	ns
$t_{INSUPLL}$	1.046		1.141		1.220		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.676	0.500	2.813	0.500	3.208	ns

Table 6–63. EP1SGX25 Row Pin Fast Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.394		2.594		2.936		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.456	2.000	4.761	2.000	5.454	ns

Table 6–64. EP1SGX25 Row Pin Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	1.970		2.109		2.377		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.880	2.000	5.246	2.000	6.013	ns
$t_{INSUPLL}$	1.326		1.386		1.552		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.304	0.500	2.427	0.500	2.765	ns

Table 6–65. EP1SGX25 Row Pin Global Clock External I/O Timing Parameters (Part 1 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	1.963		2.108		2.379		ns
t_{INH}	0.000		0.000		0.000		ns

Table 6–65. EP1SGX25 Row Pin Global Clock External I/O Timing Parameters (Part 2 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{OUTCO}	2.000	4.887	2.000	5.247	2.000	6.011	ns
$t_{INSUPLL}$	1.326		1.386		1.552		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.304	0.500	2.427	0.500	2.765	ns

Tables 6–66 through 6–71 show the external timing parameters on column and row pins for EP1SGX40 devices.

Table 6–66. EP1SGX40 Column Pin Fast Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.704		2.912		3.235		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	5.060	2.000	5.432	2.000	6.226	ns

Table 6–67. EP1SGX40 Column Pin Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.467		2.671		3.011		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	5.255	2.000	5.673	2.000	6.501	ns
$t_{INSUPLL}$	1.254		1.259		1.445		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.610	0.500	2.751	0.500	3.134	ns

Table 6–68. EP1SGX40 Column Pin Global Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.033		2.184		2.451		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	5.689	2.000	6.116	2.000	7.010	ns
$t_{INSUPLL}$	1.228		1.278		1.415		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.594	0.500	2.732	0.500	3.113	ns

Table 6–69. EP1SGX40 Row Pin Fast Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.450		2.662		3.046		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.880	2.000	5.241	2.000	6.004	ns

Table 6–70. EP1SGX40 Row Pin Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.398		2.567		2.938		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.932	2.000	5.336	2.000	6.112	ns
$t_{INSUPLL}$	1.126		1.186		1.352		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.304	0.500	2.427	0.500	2.765	ns

Table 6–71. EP1SGX40 Row Pin Global Clock External I/O Timing Parameters (Part 1 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	1.965		2.128		2.429		ns
t_{INH}	0.000		0.000		0.000		ns

Table 6–71. EP1SGX40 Row Pin Global Clock External I/O Timing Parameters (Part 2 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{OUTCO}	2.000	5.365	2.000	5.775	2.000	6.621	ns
$t_{INSUPLL}$	1.126		1.186		1.352		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.304	0.500	2.427	0.500	2.765	ns

External I/O Delay Parameters

External I/O delay timing parameters, both for I/O standard input and output adders and programmable input and output delays, are specified by speed grade, independent of device density.

Tables 6–72 through 6–77 show the adder delays associated with column and row I/O pins. If an I/O standard is selected other than LVTTL 24 mA with a fast slew rate, add the selected delay to the external t_{CO} and t_{SU} I/O parameters.

Table 6–72. Stratix GX I/O Standard Column Pin Input Delay Adders (Part 1 of 2)

I/O Standard	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
LVCMOS		0		0		0	ps
3.3-V LVTTL		0		0		0	ps
2.5-V LVTTL		30		31		35	ps
1.8-V LVTTL		150		157		180	ps
1.5-V LVTTL		210		220		252	ps
GTL		220		231		265	ps
GTL+		220		231		265	ps
3.3-V PCI		0		0		0	ps
3.3-V PCI-X 1.0		0		0		0	ps
Compact PCI		0		0		0	ps
AGP 1×		0		0		0	ps
AGP 2×		0		0		0	ps
CTT		120		126		144	ps
SSTL-3 class I		–30		–32		–37	ps
SSTL-3 class II		–30		–32		–37	ps

Table 6–72. Stratix GX I/O Standard Column Pin Input Delay Adders (Part 2 of 2)

I/O Standard	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
SSTL-2 class I		–70		–74		–86	ps
SSTL-2 class II		–70		–74		–86	ps
SSTL-18 class I		180		189		217	ps
SSTL-18 class II		180		189		217	ps
1.5-V HSTL class I		120		126		144	ps
1.5-V HSTL class II		120		126		144	ps
1.8-V HSTL class I		70		73		83	ps
1.8-V HSTL class II		70		73		83	ps

Table 6–73. Stratix GX I/O Standard Row Pin Input Delay Adders (Part 1 of 2)

I/O Standard	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
LVCMOS		0		0		0	ps
3.3-V LVTTL		0		0		0	ps
2.5-V LVTTL		30		31		35	ps
1.8-V LVTTL		150		157		180	ps
1.5-V LVTTL		210		220		252	ps
GTL		0		0		0	ps
GTL+		220		231		265	ps
3.3-V PCI		0		0		0	ps
3.3-V PCI-X 1.0		0		0		0	ps
Compact PCI		0		0		0	ps
AGP 1×		0		0		0	ps
AGP 2×		0		0		0	ps
CTT		80		84		96	ps
SSTL-3 class I		–30		–32		–37	ps
SSTL-3 class II		–30		–32		–37	ps
SSTL-2 class I		–70		–74		–86	ps
SSTL-2 class II		–70		–74		–86	ps
SSTL-18 class I		180		189		217	ps
SSTL-18 class II		0		0		0	ps
1.5-V HSTL class I		130		136		156	ps

Table 6–73. Stratix GX I/O Standard Row Pin Input Delay Adders (Part 2 of 2)

I/O Standard	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
1.5-V HSTL class II		0		0		0	ps
1.8-V HSTL class I		70		73		83	ps
1.8-V HSTL class II		70		73		83	ps
LVDS *(1)*		40		42		48	ps
LVPECL *(1)*		–50		–53		–61	ps
3.3-V PCML *(1)*		330		346		397	ps
HyperTransport *(1)*		80		84		96	ps

Table 6–74. Stratix GX I/O Standard Output Delay Adders for Fast Slew Rate on Column Pins (Part 1 of 2)

Standard		-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
LVCMOS	2 mA		570		599		689	ps
	4 mA		570		599		689	ps
	8 mA		350		368		423	ps
	12 mA		130		137		157	ps
	24 mA		0		0		0	ps
3.3-V LVTTL	4 mA		570		599		689	ps
	8 mA		350		368		423	ps
	12 mA		130		137		157	ps
	16 mA		70		74		85	ps
	24 mA		0		0		0	ps
2.5-V LVTTL	2 mA		830		872		1,002	ps
	8 mA		250		263		302	ps
	12 mA		140		147		169	ps
	16 mA		100		105		120	ps
1.8-V LVTTL	2 mA		420		441		507	ps
	8 mA		350		368		423	ps
	12 mA		350		368		423	ps
1.5-V LVTTL	2 mA		1,740		1,827		2,101	ps
	4 mA		1,160		1,218		1,400	ps
	8 mA		690		725		833	ps
GTL			–150		–157		–181	ps

Table 6–74. Stratix GX I/O Standard Output Delay Adders for Fast Slew Rate on Column Pins (Part 2 of 2)

Standard	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
GTL+		–110		–115		–133	ps
3.3-V PCI		–230		–241		–277	ps
3.3-V PCI-X 1.0		–230		–241		–277	ps
Compact PCI		–230		–241		–277	ps
AGP 1×		–30		–31		–36	ps
AGP 2×		–30		–31		–36	ps
CTT		50		53		61	ps
SSTL-3 class I		90		95		109	ps
SSTL-3 class II		–50		–52		–60	ps
SSTL-2 class I		100		105		120	ps
SSTL-2 class II		20		21		24	ps
SSTL-18 class I		230		242		278	ps
SSTL-18 class II		0		0		0	ps
1.5-V HSTL class I		380		399		459	ps
1.5-V HSTL class II		190		200		230	ps
1.8-V HSTL class I		380		399		459	ps
1.8-V HSTL class II		390		410		471	ps

Table 6–75. Stratix GX I/O Standard Output Delay Adders for Fast Slew Rate on Row Pins (Part 1 of 2)

Standard		-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
LVCMOS	2 mA		570		599		689	ps
	4 mA		570		599		689	ps
	8 mA		350		368		423	ps
	12 mA		130		137		157	ps
	24 mA		0		0		0	ps
3.3-V LVTTL	4 mA		570		599		689	ps
	8 mA		350		368		423	ps
	12 mA		130		137		157	ps
	16 mA		70		74		85	ps
	24 mA		0		0		0	ps

Table 6–75. Stratix GX I/O Standard Output Delay Adders for Fast Slew Rate on Row Pins (Part 2 of 2)

Standard		-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
2.5-V LVTTL	2 mA		830		872		1,002	ps
	8 mA		250		263		302	ps
	12 mA		140		147		169	ps
	16 mA		100		105		120	ps
1.8-V LVTTL	2 mA		1,510		1,586		1,824	ps
	8 mA		420		441		507	ps
	12 mA		350		368		423	ps
1.5-V LVTTL	2 mA		1,740		1,827		2,101	ps
	4 mA		1,160		1,218		1,400	ps
	8 mA		690		725		833	ps
CTT			50		53		61	ps
SSTL-3 class I			90		95		109	ps
SSTL-3 class II			–50		–52		–60	ps
SSTL-2 class I			100		105		120	ps
SSTL-2 class II			20		21		24	ps
LVDS *(1)*			–20		–21		–24	ps
LVPECL *(1)*			40		42		48	ps
PCML *(1)*			–60		–63		–73	ps
HyperTransport Technology *(1)*			70		74		85	ps

Table 6–76. Stratix GX I/O Standard Output Delay Adders for Slow Slew Rate on Column Pins (Part 1 of 2)

I/O Standard		-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
LVCMOS	2 mA		1,911		2,011		2,312	ps
	4 mA		1,911		2,011		2,312	ps
	8 mA		1,691		1,780		2,046	ps
	12 mA		1,471		1,549		1,780	ps
	24 mA		1,341		1,412		1,623	ps

Table 6–76. Stratix GX I/O Standard Output Delay Adders for Slow Slew Rate on Column Pins (Part 2 of 2)

I/O Standard		-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
3.3-V LVTTL	4 mA		1,993		2,097		2,411	ps
	8 mA		1,773		1,866		2,145	ps
	12 mA		1,553		1,635		1,879	ps
	16 mA		1,493		1,572		1,807	ps
	24 mA		1,423		1,498		1,722	ps
2.5-V LVTTL	2 mA		2,631		2,768		3,182	ps
	8 mA		2,051		2,159		2,482	ps
	12 mA		1,941		2,043		2,349	ps
	16 mA		1,901		2,001		2,300	ps
1.8-V LVTTL	2 mA		4,632		4,873		5,604	ps
	8 mA		3,542		3,728		4,287	ps
	12 mA		3,472		3,655		4,203	ps
1.5-V LVTTL	2 mA		6,620		6,964		8,008	ps
	4 mA		6,040		6,355		7,307	ps
	8 mA		5,570		5,862		6,740	ps
GTL			1,191		1,255		1,442	ps
GTL+			1,231		1,297		1,90	ps
3.3-V PCI			1,111		1,171		1,346	ps
3.3-V PCI-X 1.0			1,111		1,171		1,346	ps
Compact PCI			1,111		1,171		1,346	ps
AGP 1×			1,311		1,381		1,587	ps
AGP 2×			1,311		1,381		1,587	ps
CTT			1,391		1,465		1,684	ps
SSTL-3 class I			1,431		1,507		1,732	ps
SSTL-3 class II			1,291		1,360		1,563	ps
SSTL-2 class I			1,912		2,013		2,314	ps
SSTL-2 class II			1,832		1,929		2,218	ps
SSTL-18 class I			3,097		3,260		3,748	ps
SSTL-18 class II			2,867		3,018		3,470	ps
1.5-V HSTL class I			4,916		5,174		5,950	ps
1.5-V HSTL class II			4,726		4,975		5,721	ps
1.8-V HSTL class I			3,247		3,417		3,929	ps
1.8-V HSTL class II			3,257		3,428		3,941	ps

Table 6–77. Stratix GX I/O Standard Output Delay Adders for Slow Slew Rate on Row Pins

I/O Standard		-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
LVCMOS	2 mA		1,930		2,031		2,335	ps
	4 mA		1,930		2,031		2,335	ps
	8 mA		1,710		1,800		2,069	ps
	12 mA		1,490		1,569		1,803	ps
3.3-V LVTTL	4 mA		1,953		2,055		2,363	ps
	8 mA		1,733		1,824		2,097	ps
	12 mA		1,513		1,593		1,831	ps
	16 mA		1,453		1,530		1,759	ps
2.5-V LVTTL	2 mA		2,632		2,769		3,183	ps
	8 mA		2,052		2,160		2,483	ps
	12 mA		1,942		2,044		2,350	ps
	16 mA		1,902		2,002		2,301	ps
1.8-V LVTTL	2 mA		4,537		4,773		5,489	ps
	8 mA		3,447		3,628		4,172	ps
	12 mA		3,377		3,555		4,088	ps
1.5-V LVTTL	2 mA		6,575		6,917		7,954	ps
	4 mA		5,995		6,308		7,253	ps
	8 mA		5,525		5,815		6,686	ps
CTT			1,410		1,485		1,707	ps
SSTL-3 class I			1,450		1,527		1,755	ps
SSTL-3 class II			1,310		1,380		1,586	ps
SSTL-2 class I			1,797		1,892		2,175	ps
SSTL-2 class II			1,717		1,808		2,079	ps
LVDS *(1)*			1,340		1,411		1,622	ps
LVPECL *(1)*			1,400		1,474		1,694	ps
3.3-V PCML *(1)*			1,300		1,369		1,573	ps
HyperTransport technology *(1)*			1,430		1,506		1,731	ps

Note to Tables 6–72 through 6–77:
(1) These parameters are only available on the left side row I/O pins.

Tables 6–78 and 6–79 show the adder delays for the column and row IOE programmable delays, respectively. These delays are controlled with the Quartus II software logic options listed in the Parameter column.

Table 6–78. Stratix GX IOE Programmable Delays on Column Pins

Parameter	Setting	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
Decrease input delay to internal cells	Off		3,970		4,367		5,022	ps
	On		3,390		3,729		4,288	ps
	Small		2,810		3,091		3,554	ps
	Medium		212		224		257	ps
	Large		212		224		257	ps
Decrease input delay to input register	Off		3900		4,290		4,933	ps
	On		0		0		0	ps
Decrease input delay to output register	Off		1,240		1,364		1,568	ps
	On		0		0		0	ps
Increase delay to output pin	Off		0		0		0	ps
	On		377		397		456	ps
Increase delay to output enable pin	Off		0		0		0	ps
	On		338		372		427	ps
Increase output clock enable delay	Off		0		0		0	ps
	On		540		594		683	ps
	Small		1,016		1,118		1,285	ps
	Large		1,016		1,118		1,285	ps
Increase input clock enable delay	Off		0		0		0	ps
	On		540		594		683	ps
	Small		1,016		1,118		1,285	ps
	Large		1,016		1,118		1,285	ps
Increase output enable clock enable delay	Off		0		0		0	ps
	On		540		594		683	ps
	Small		1,016		1,118		1,285	ps
	Large		1,016		1,118		1,285	ps

Table 6–79. Stratix GX IOE Programmable Delays on Row Pins

Parameter	Setting	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
Decrease input delay to internal cells	Off		3,970		4,367		5,022	ps
	On		3,390		3,729		4,288	ps
	Small		2,810		3,091		3,554	ps
	Medium		164		173		198	ps
	Large		164		173		198	ps
Decrease input delay to input register	Off		3,900		4,290		4,933	ps
	On		0		0		0	ps
Decrease input delay to output register	Off		1,240		1,364		1,568	ps
	On		0		0		0	ps
Increase delay to output pin	Off		0		0		0	ps
	On		377		397		456	ps
Increase delay to output enable pin	Off		0		0		0	ps
	On		348		383		441	ps
Increase output clock enable delay	Off		0		0		0	ps
	On		180		198		227	ps
	Small		260		286		328	ps
	Large		260		286		328	ps
Increase input clock enable delay	Off		0		0		0	ps
	On		180		198		227	ps
	Small		260		286		328	ps
	Large		260		286		328	ps
Increase output enable clock enable delay	Off		0		0		0	ps
	On		540		594		683	ps
	Small		1,016		1,118		1,285	ps
	Large		1,016		1,118		1,285	ps

The scaling factors for output pin timing in Table 6–80 are shown in units of time per pF unit of capacitance (ps/pF). Add this delay to the combinational timing path for output or bidirectional pins in addition to the "I/O Adder" delays shown in Tables 6–72 through 6–77 and the "IOE Programmable Delays" in Tables 6–78 and 6–79.

Table 6–80. Output Delay Adder for Loading on LVTTL/LVCMOS Output Buffers

LVTTL/LVCMOS Standards						
Conditions		**Output Pin Adder Delay (ps/pF)**				
Parameter	**Value**	3.3-V LVTTL	2.5-V LVTTL	1.8-V LVTTL	1.5-V LVTTL	LVCMOS
Drive Strength	24 mA	15	–	–	-	8
	16 mA	25	18	–	–	–
	12 mA	30	25	25	–	15
	8 mA	50	35	40	35	20
	4 mA	60	–	–	80	30
	2 mA	–	75	120	160	60
SSTL/HSTL Standards						
Conditions		**Output Pin Adder Delay (ps/pF)**				
		SSTL-3	SSTL-2	SSTL-1.8	1.5-V HSTL	1.8-V HSTL
Class I		25	25	25	25	25
Class II		25	20	25	20	20
GTL+/GTL/CTT/PCI Standards						
Conditions		**Output Pin Adder Delay (ps/pF)**				
Parameter	**Value**	GTL+	GTL	CTT	PCI	AGP
V_{CCIO} voltage level	3.3 V	18	18	25	20	20
	2.5 V	15	18	-	-	-

Maximum Input & Output Clock Rates

Tables 6–81 through 6–83 show the maximum input clock rate for column and row pins in Stratix GX devices.

Table 6–81. Stratix GX Maximum Input Clock Rate for CLK[7..4] & CLK[15..12] Pins

I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
LVTTL	422	422	390	MHz
2.5 V	422	422	390	MHz
1.8 V	422	422	390	MHz
1.5 V	422	422	390	MHz
LVCMOS	422	422	390	MHz
GTL	300	250	200	MHz
GTL+	300	250	200	MHz
SSTL-3 class I	400	350	300	MHz
SSTL-3 class II	400	350	300	MHz
SSTL-2 class I	400	350	300	MHz
SSTL-2 class II	400	350	300	MHz
SSTL-18 class I	400	350	300	MHz
SSTL-18 class II	400	350	300	MHz
1.5-V HSTL class I	400	350	300	MHz
1.5-V HSTL class II	400	350	300	MHz
1.8-V HSTL class I	400	350	300	MHz
1.8-V HSTL class II	400	350	300	MHz
3.3-V PCI	422	422	390	MHz
3.3-V PCI-X 1.0	422	422	390	MHz
Compact PCI	422	422	390	MHz
AGP 1×	422	422	390	MHz
AGP 2×	422	422	390	MHz
CTT	300	250	200	MHz
Differential HSTL	400	350	300	MHz
LVDS	645	645	622	MHz
LVPECL	645	645	622	MHz
PCML	300	275	275	MHz
HyperTransport technology	500	500	450	MHz

Table 6–82. Stratix GX Maximum Input Clock Rate for CLK[0, 2, 9, 11] Pins & FPLL[8..7]CLK Pins

I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
LVTTL	422	422	390	MHz
2.5 V	422	422	390	MHz
1.8 V	422	422	390	MHz
1.5 V	422	422	390	MHz
LVCMOS	422	422	390	MHz
GTL	300	250	200	MHz
GTL+	300	250	200	MHz
SSTL-3 class I	400	350	300	MHz
SSTL-3 class II	400	350	300	MHz
SSTL-2 class I	400	350	300	MHz
SSTL-2 class II	400	350	300	MHz
SSTL-18 class I	400	350	300	MHz
SSTL-18 class II	400	350	300	MHz
1.5-V HSTL class I	400	350	300	MHz
1.5-V HSTL class II	400	350	300	MHz
1.8-V HSTL class I	400	350	300	MHz
1.8-V HSTL class II	400	350	300	MHz
3.3-V PCI	422	422	390	MHz
3.3-V PCI-X 1.0	422	422	390	MHz
Compact PCI	422	422	390	MHz
AGP 1×	422	422	390	MHz
AGP 2×	422	422	390	MHz
CTT	300	250	200	MHz
Differential HSTL	400	350	300	MHz
LVDS	717	717	640	MHz
LVPECL	717	717	640	MHz
PCML	400	375	350	MHz
HyperTransport technology	717	717	640	MHz

Table 6–83. Stratix GX Maximum Input Clock Rate for CLK[1, 3, 8, 10] Pins (Part 1 of 2)

I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
LVTTL	422	422	390	MHz
2.5 V	422	422	390	MHz

Table 6–83. Stratix GX Maximum Input Clock Rate for CLK[1, 3, 8, 10] Pins (Part 2 of 2)				
I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
1.8 V	422	422	390	MHz
1.5 V	422	422	390	MHz
LVCMOS	422	422	390	MHz
GTL	300	250	200	MHz
GTL+	300	250	200	MHz
SSTL-3 class I	400	350	300	MHz
SSTL-3 class II	400	350	300	MHz
SSTL-2 class I	400	350	300	MHz
SSTL-2 class II	400	350	300	MHz
SSTL-18 class I	400	350	300	MHz
SSTL-18 class II	400	350	300	MHz
1.5-V HSTL class I	400	350	300	MHz
1.5-V HSTL class II	400	350	300	MHz
1.8-V HSTL class I	400	350	300	MHz
1.8-V HSTL class II	400	350	300	MHz
3.3-V PCI	422	422	390	MHz
3.3-V PCI-X 1.0	422	422	390	MHz
Compact PCI	422	422	390	MHz
AGP 1×	422	422	390	MHz
AGP 2×	422	422	390	MHz
CTT	300	250	200	MHz
Differential HSTL	400	350	300	MHz
LVDS	645	645	640	MHz
LVPECL	645	645	640	MHz
PCML	300	275	275	MHz
HyperTransport technology	645	645	640	MHz

Tables 6–84 and 6–85 show the maximum output clock rate for column and row pins in Stratix GX devices.

Table 6–84. Stratix GX Maximum Output Clock Rate for PLL[5, 6, 11, 12] Pins (Part 1 of 2)				
I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
LVTTL	350	300	250	MHz
2.5 V	350	300	300	MHz

Table 6–84. Stratix GX Maximum Output Clock Rate for PLL[5, 6, 11, 12] Pins (Part 2 of 2)

I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
1.8 V	250	250	250	MHz
1.5 V	225	200	200	MHz
LVCMOS	350	300	250	MHz
GTL	200	167	125	MHz
GTL+	200	167	125	MHz
SSTL-3 class I	167	150	133	MHz
SSTL-3 class II	167	150	133	MHz
SSTL-2 class I	200	200	167	MHz
SSTL-2 class II	200	200	167	MHz
SSTL-18 class I	150	133	133	MHz
SSTL-18 class II	150	133	133	MHz
1.5-V HSTL class I	250	225	200	MHz
1.5-V HSTL class II	225	200	200	MHz
1.8-V HSTL class I	250	225	200	MHz
1.8-V HSTL class II	225	200	200	MHz
3.3-V PCI	350	300	250	MHz
3.3-V PCI-X 1.0	350	300	250	MHz
Compact PCI	350	300	250	MHz
AGP 1×	350	300	250	MHz
AGP 2×	350	300	250	MHz
CTT	200	200	200	MHz
Differential HSTL	225	200	200	MHz
Differential SSTL-2	200	200	167	MHz
LVDS	500	500	500	MHz
LVPECL	500	500	500	MHz
PCML	350	350	350	MHz
HyperTransport technology	350	350	350	MHz

Table 6–85. Stratix GX Maximum Output Clock Rate (Using I/O Pins) for PLL[1, 2] Pins (Part 1 of 2)

I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
LVTTL	400	350	300	MHz
2.5 V	400	350	300	MHz
1.8 V	400	350	300	MHz

Table 6–85. Stratix GX Maximum Output Clock Rate (Using I/O Pins) for PLL[1, 2] Pins (Part 2 of 2)

I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
1.5 V	350	300	300	MHz
LVCMOS	400	350	300	MHz
GTL	200	167	125	MHz
GTL+	200	167	125	MHz
SSTL-3 class I	167	150	133	MHz
SSTL-3 class II	167	150	133	MHz
SSTL-2 class I	150	133	133	MHz
SSTL-2 class II	150	133	133	MHz
SSTL-18 class I	150	133	133	MHz
SSTL-18 class II	150	133	133	MHz
HSTL class I	250	225	200	MHz
HSTL class II	225	225	200	MHz
3.3-V PCI	250	225	200	MHz
3.3-V PCI-X 1.0	225	225	200	MHz
Compact PCI	400	350	300	MHz
AGP 1×	400	350	300	MHz
AGP 2×	400	350	300	MHz
CTT	300	250	200	MHz
Differential HSTL	225	225	200	MHz
LVDS	717	717	500	MHz
LVPECL	717	717	500	MHz
PCML	420	420	420	MHz
HyperTransport technology	420	420	420	MHz

High-Speed I/O Specification

Table 6–86 provides high-speed timing specifications definitions.

Table 6–86. High-Speed Timing Specifications & Definitions (Part 1 of 2)

High-Speed Timing Specification	Definitions
t_C	High-speed receiver/transmitter input and output clock period.
f_{HSCLK}	High-speed receiver/transmitter input and output clock frequency.
t_{RISE}	Low-to-high transmission time.

Table 6–86. High-Speed Timing Specifications & Definitions (Part 2 of 2)	
High-Speed Timing Specification	**Definitions**
t_{FALL}	High-to-low transmission time.
Timing unit interval (TUI)	The timing budget allowed for skew, propagation delays, and data sampling window. (TUI = 1/(Receiver Input Clock Frequency × Multiplication Factor) = t_C/w).
f_{HSDR}	Maximum/minimum LVDS data transfer rate (f_{HSDR} = 1/TUI), non-DPA.
$f_{HSDRDPA}$	Maximum/minimum LVDS data transfer rate ($f_{HSDRDPA}$ = 1/TUI), DPA.
Channel-to-channel skew (TCCS)	The timing difference between the fastest and slowest output edges, including t_{CO} variation and clock skew. The clock is included in the TCCS measurement.
Sampling window (SW)	The period of time during which the data must be valid in order to capture it correctly. The setup and hold times determine the ideal strobe position within the sampling window. SW = t_{SW} (max) – t_{SW} (min).
Input jitter (peak-to-peak)	Peak-to-peak input jitter on high-speed PLLs.
Output jitter (peak-to-peak)	Peak-to-peak output jitter on high-speed PLLs.
t_{DUTY}	Duty cycle on high-speed transmitter output clock.
t_{LOCK}	Lock time for high-speed transmitter and receiver PLLs.

Table 6–87 shows the high-speed I/O timing specifications for Stratix GX devices.

Table 6–87. High-Speed I/O Specifications (Part 1 of 4) Notes (1), (2)											
Symbol	**Conditions**	**-5 Speed Grade**			**-6 Speed Grade**			**-7 Speed Grade**			**Unit**
		Min	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	
f_{HSCLK} (Clock frequency) (LVDS, LVPECL, HyperTransport technology) f_{HSCLK} = f_{HSDR} / W	*W* = 1 to 30 for ≤717 Mbps W = 2 to 30 for > 717 Mbps	10		717	10		717	10		624	MHz
f_{HSCLK_DPA}		74		717	74		717	74		717	MHz

Table 6–87. High-Speed I/O Specifications (Part 2 of 4) *Notes (1), (2)*

Symbol	Conditions	-5 Speed Grade			-6 Speed Grade			-7 Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
f_{HSDR} Device operation (LVDS, LVPECL, HyperTransport technology)	*J* = 10	300		840	300		840	300		840	Mbps
	J = 8	300		840	300		840	300		840	Mbps
	J = 7	300		840	300		840	300		840	Mbps
	J = 4	300		840	300		840	300		840	Mbps
	J = 2	100		624	100		624	100		462	Mbps
	J = 1 (LVDS and LVPECL only)	100		462	100		462	100		462	Mbps
$f_{HSDRDPA}$ (LVDS, LVPECL)	J=10	300		1000	300		840	300		840	Mbps
	J=8	300		1000	300		840	300		840	Mbps
f_{HSCLK} (Clock frequency) (PCML) f_{HSCLK} = f_{HSDR} / W	*W* = 1 to 30	10		400	10		400	10		311	MHz
f_{HSDR} Device operation (PCML)	*J* = 10	300		400	300		400	300		311	Mbps
	J = 8	300		400	300		400	300		311	Mbps
	J = 7	300		400	300		400	300		311	Mbps
	J = 4	300		400	300		400	300		311	Mbps
	J = 2	100		400	100		400	100		300	Mbps
	J = 1	100		250	100		250	100		200	Mbps
DPA Run Length				6400			6400			6400	UI
DPA Jitter Tolerance$_{(p-p)}$	all data rates			0.44			0.44			0.44	UI
DPA Minimum Eye opening (p-p)		0.56			0.56			0.56			UI
DPA Receiver Latency		5		9	5		9	5		9	*(3)*

Table 6–87. High-Speed I/O Specifications (Part 3 of 4) *Notes (1), (2)*

Symbol	Conditions			-5 Speed Grade			-6 Speed Grade			-7 Speed Grade			Unit
				Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
DPA Lock Time	Standard	Training Pattern	Transition Density										
	SPI-4, CSIX	0000 0000 0011 1111 1111	10%	256			256			256			*(4)*
	Rapid IO	0000 1111	25%	256			256			256			*(4)*
		1001 0000	50%	256			256			256			*(4)*
	Misc	1010 1010	100 %	256			256			256			*(4)*
		0101 0101		256			256			256			*(4)*
TCCS	All					200			200			300	ps
SW	PCML (J = 4, 7, 8, 10)			750			750			800			ps
	PCML (J = 2)			900			900			1,200			ps
	PCML (J = 1)			1,500			1,500			1,700			ps
	LVDS and LVPECL (J = 1)			500			500			550			ps
	LVDS, LVPECL, HyperTransport technology (J = 2 through 10)			440			440			500			ps
Input jitter tolerance (peak-to-peak)	All					250			250			250	ps
Output jitter (peak-to-peak)	All					160			160			200	ps
Output t_{RISE}	LVDS			80	110	120	80	110	120	80	110	120	ps
	HyperTransport technology			110	170	200	110	170	200	120	170	200	ps
	LVPECL			90	130	150	90	130	150	100	135	150	ps
	PCML			80	110	135	80	110	135	80	110	135	ps

Table 6–87. High-Speed I/O Specifications (Part 4 of 4) *Notes (1), (2)*

Symbol	Conditions	-5 Speed Grade			-6 Speed Grade			-7 Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Output t_{FALL}	LVDS	80	110	120	80	110	120	80	110	120	ps
	HyperTransport technology	110	170	200	110	170	200	110	170	200	ps
	LVPECL	90	130	160	90	130	160	100	135	160	ps
	PCML	105	140	175	105	140	175	110	145	175	ps
t_{DUTY}	LVDS (*J* = 2 through 10)	47.5	50	52.5	47.5	50	52.5	47.5	50	52.5	%
	LVDS (*J* =1) and LVPECL, PCML, HyperTransport technology	45	50	55	45	50	55	45	50	55	%
t_{LOCK}	All			100			100			100	µs

Notes to Table 6–87:
(1) When J = 4, 7, 8, and 10, the SERDES block is used.
(2) When J = 2 or J = 1, the SERDES is bypassed.
(3) Number of parallel CLK cycles.
(4) Number of repetitions.

PLL Timing

Tables 6–88 through 6–90 describe the Stratix GX device enhanced PLL specifications.

Table 6–88. Enhanced PLL Specifications for -5 Speed Grades (Part 1 of 2)

Symbol	Parameter	Min	Typ	Max	Unit
f_{IN}	Input clock frequency	3 *(1)*		684	MHz
f_{INDUTY}	Input clock duty cycle	40		60	%
$f_{EINDUTY}$	External feedback clock input duty cycle	40		60	%
$t_{INJITTER}$	Input clock period jitter			±200 *(2)*	ps
$t_{EINJITTER}$	External feedback clock period jitter			±200 *(2)*	ps
t_{FCOMP}	External feedback clock compensation time *(3)*			6	ns
f_{OUT}	Output frequency for internal global or regional clock	0.3		500	MHz
f_{OUT_EXT}	Output frequency for external clock *(2)*	0.3		526	MHz

Table 6–88. Enhanced PLL Specifications for -5 Speed Grades (Part 2 of 2)

Symbol	Parameter	Min	Typ	Max	Unit
$t_{OUTDUTY}$	Duty cycle for external clock output (when set to 50%)	45		55	%
t_{JITTER}	Period jitter for external clock output *(5)*			±100 ps for >200 MHz `outclk` ±20 mUI for <200 MHz `outclk`	ps or mUI
$t_{CONFIG5,6}$	Time required to reconfigure the scan chains for PLLs 5 and 6			$289/f_{SCANCLK}$	
$t_{CONFIG11,12}$	Time required to reconfigure the scan chains for PLLs 11 and 12			$193/f_{SCANCLK}$	
$t_{SCANCLK}$	`scanclk` frequency *(4)*			22	MHz
t_{DLOCK}	Time required to lock dynamically (after switchover or reconfiguring any non-post-scale counters/delays) *(6)*			100	µs
t_{LOCK}	Time required to lock from end of device configuration	10		400	µs
f_{VCO}	PLL internal VCO operating range	300		800 *(7)*	MHz
t_{LSKEW}	Clock skew between two external clock outputs driven by the same counter		±50		ps
t_{SKEW}	Clock skew between two external clock outputs driven by the different counters with the same settings		±75		ps
f_{SS}	Spread spectrum modulation frequency	30		150	kHz
% spread	Percentage spread for spread spectrum frequency *(9)*	0.4	0.5	0.6	%
t_{ARESET}	Minimum pulse width on `areset` signal	10			ns

Table 6–89. Enhanced PLL Specifications for -6 Speed Grades (Part 1 of 2)

Symbol	Parameter	Min	Typ	Max	Unit
f_{IN}	Input clock frequency	3 *(1)*		650	MHz
f_{INDUTY}	Input clock duty cycle	40		60	%
$f_{EINDUTY}$	External feedback clock input duty cycle	40		60	%
$t_{INJITTER}$	Input clock period jitter			±200 *(2)*	ps
$t_{EINJITTER}$	External feedback clock period jitter			±200 *(2)*	ps
t_{FCOMP}	External feedback clock compensation time *(3)*			6	ns

Table 6–89. Enhanced PLL Specifications for -6 Speed Grades (Part 2 of 2)

Symbol	Parameter	Min	Typ	Max	Unit
f_{OUT}	Output frequency for internal global or regional clock	0.3		450	MHz
f_{OUT_EXT}	Output frequency for external clock *(2)*	0.3		500	MHz
$t_{OUTDUTY}$	Duty cycle for external clock output (when set to 50%)	45		55	%
t_{JITTER}	Period jitter for external clock output *(5)*			±100 ps for >200 MHz `outclk` ±20 mUI for <200 MHz `outclk`	ps or mUI
$t_{CONFIG5,6}$	Time required to reconfigure the scan chains for PLLs 5 and 6			$289/f_{SCANCLK}$	
$t_{CONFIG11,12}$	Time required to reconfigure the scan chains for PLLs 11 and 12			$193/f_{SCANCLK}$	
$t_{SCANCLK}$	`scanclk` frequency *(4)*			22	MHz
t_{DLOCK}	Time required to lock dynamically (after switchover or reconfiguring any non-post-scale counters/delays) *(6)* *(10)*	*(8)*		100	µs
t_{LOCK}	Time required to lock from end of device configuration *(10)*	10		400	µs
f_{VCO}	PLL internal VCO operating range	300		800 *(7)*	MHz
t_{LSKEW}	Clock skew between two external clock outputs driven by the same counter		±50		ps
t_{SKEW}	Clock skew between two external clock outputs driven by the different counters with the same settings		±75		ps
f_{SS}	Spread spectrum modulation frequency	30		150	kHz
% spread	Percentage spread for spread spectrum frequency *(9)*	0.4	0.5	0.6	%
t_{ARESET}	Minimum pulse width on `areset` signal	10			ns

Table 6–90. Enhanced PLL Specifications for -7 Speed Grade (Part 1 of 3)

Symbol	Parameter	Min	Typ	Max	Unit
f_{IN}	Input clock frequency	3 *(1)*		565	MHz
f_{INDUTY}	Input clock duty cycle	40		60	%
$f_{EINDUTY}$	External feedback clock input duty cycle	40		60	%
$t_{INJITTER}$	Input clock period jitter			±200 *(2)*	ps
$t_{EINJITTER}$	External feedback clock period jitter			±200 *(2)*	ps

Table 6–90. Enhanced PLL Specifications for -7 Speed Grade (Part 2 of 3)

Symbol	Parameter	Min	Typ	Max	Unit
t_{FCOMP}	External feedback clock compensation time *(3)*			6	ns
f_{OUT}	Output frequency for internal global or regional clock	0.3		420	MHz
f_{OUT_EXT}	Output frequency for external clock *(2)*	0.3		434	MHz
$t_{OUTDUTY}$	Duty cycle for external clock output (when set to 50%)	45		55	%
t_{JITTER}	Period jitter for external clock output *(5)*			±100 ps for >200 MHz `outclk` ±20 mUI for <200 MHz `outclk`	ps or mUI
$t_{CONFIG5,6}$	Time required to reconfigure the scan chains for PLLs 5 and 6			$289/f_{SCANCLK}$	
$t_{CONFIG11,12}$	Time required to reconfigure the scan chains for PLLs 11 and 12			$193/f_{SCANCLK}$	
$t_{SCANCLK}$	`scanclk` frequency *(4)*			22	MHz
t_{DLOCK}	Time required to lock dynamically (after switchover or reconfiguring any non-post-scale counters/delays) *(6)* *(10)*	*(8)*		100	µs
t_{LOCK}	Time required to lock from end of device configuration *(10)*	10		400	µs
f_{VCO}	PLL internal VCO operating range	300		600 *(7)*	MHz

Table 6–90. Enhanced PLL Specifications for -7 Speed Grade (Part 3 of 3)

Symbol	Parameter	Min	Typ	Max	Unit
t_{LSKEW}	Clock skew between two external clock outputs driven by the same counter		±50		ps
t_{SKEW}	Clock skew between two external clock outputs driven by the different counters with the same settings		±75		ps
f_{SS}	Spread spectrum modulation frequency	30		150	kHz
% spread	Percentage spread for spread spectrum frequency *(9)*	0.5		0.6	%
t_{ARESET}	Minimum pulse width on `areset` signal	10			ns

Notes to Tables 6–88 through 6–90:

(1) The minimum input clock frequency to the PFD (f_{IN}/N) must be at least 3 MHz for Stratix device enhanced PLLs.
(2) See "Maximum Input & Output Clock Rates" on page 6–54.
(3) t_{FCOMP} can also equal 50% of the input clock period multiplied by the pre-scale divider *n* (whichever is less).
(4) This parameter is timing analyzed by the Quartus II software because the `scanclk` and `scandata` ports can be driven by the logic array.
(5) Actual jitter performance may vary based on the system configuration.
(6) Total required time to reconfigure and lock is equal to $t_{DLOCK} + t_{CONFIG}$. If only post-scale counters and delays are changed, then t_{DLOCK} is equal to 0.
(7) The VCO range is limited to 500 to 800 MHz when the spread spectrum feature is selected.
(8) Lock time is a function of PLL configuration and may be significantly faster depending on bandwidth settings or feedback counter change increment.
(9) Exact, user-controllable value depends on the PLL settings.
(10) The LOCK circuit on Stratix PLLs does not work for industrial devices below -20C unless the PFD frequency > 200 MHz. See the *Stratix FPGA Errata Sheet* for more information on the PLL.

Table 6–91 describes the Stratix GX device fast PLL specifications.

Table 6–91. Fast PLL Specifications for -5 & -6 Speed Grade Devices

Symbol	Parameter	Min	Max	Unit
f_{IN}	CLKIN frequency (for *m* = 1) *(1)*	300	717	MHz
	CLKIN frequency (for *m* = 2 to 19)	300/*m*	1,000/*m*	MHz
	CLKIN frequency (for *m* = 20 to 32)	10	1,000/*m*	MHz
f_{OUT}	Output frequency for internal global or regional clock *(2)*	9.4	420	MHz
f_{OUT_EXT}	Output frequency for external clock	9.375	717	MHz
f_{VCO}	VCO operating frequency	300	1,000	MHz
t_{INDUTY}	CLKIN duty cycle	40	60	%
$t_{INJITTER}$	Period jitter for CLKIN pin		±200	ps
t_{DUTY}	Duty cycle for DFFIO 1× CLKOUT pin *(3)*	45	55	%
t_{JITTER}	Period jitter for DIFFIO clock out *(3)*		±80	ps
	Period jitter for internal global or regional clock		±100 ps for >200-MHz outclk ±20 mUI for <200-MHz outclk	ps or mUI
t_{LOCK}	Time required for PLL to acquire lock	10	100	µs
m	Multiplication factors for *m* counter *(3)*	1	32	Integer
*l*0, *l*1, *g*0	Multiplication factors for *l*0, *l*1, and *g*0 counter *(4)*, *(5)*	1	32	Integer
t_{ARESET}	Minimum pulse width on areset signal	10		ns

Table 6–92. Fast PLL Specifications for -7 & -8 Speed Grades (Part 1 of 2)

Symbol	Parameter	Min	Max	Unit
f_{IN}	CLKIN frequency (for *m* = 1) *(1)*,	300	640	MHz
	CLKIN frequency (for *m* = 2 to 19)	300/*m*	700/*m*	MHz
	CLKIN frequency (for *m* = 20 to 32)	10	700/*m*	MHz
f_{OUT}	Output frequency for internal global or regional clock *(2)*	9.375	420	MHz
f_{OUT_EXT}	Output frequency for external clock	9.4	500	MHz
f_{VCO}	VCO operating frequency	300	700	MHz
t_{INDUTY}	CLKIN duty cycle	40	60	%
$t_{INJITTER}$	Period jitter for CLKIN pin		±200	ps

Table 6–92. Fast PLL Specifications for -7 & -8 Speed Grades (Part 2 of 2)

Symbol	Parameter	Min	Max	Unit
t_{DUTY}	Duty cycle for DFFIO 1× CLKOUT pin *(3)*	45	55	%
t_{JITTER}	Period jitter for DIFFIO clock out *(3)*		±80	ps
	Period jitter for internal global or regional clock		±100 ps for >200 MHz outclk ±20 mUI for <200 MHz outclk	ps or mUI
t_{LOCK}	Time required for PLL to acquire lock	10	100	µs
m	Multiplication factors for *m* counter *(4)*	1	32	Integer
*l*0, *l*1, *g*0	Multiplication factors for *l*0, *l*1, and *g*0 counter *(4)*, *(5)*	1	32	Integer
t_{ARESET}	Minimum pulse width on areset signal	10		ns

Notes to Tables 6–91 & 6–92:

(1) See "Maximum Input & Output Clock Rates" on page 6–54.
(2) When using the SERDES, high-speed differential I/O mode supports a maximum output frequency of 210 MHz to the global or regional clocks (that is, the maximum data rate 840 Mbps divided by the smallest SERDES J factor of 4).
(3) This parameter is for high-speed differential I/O mode only.
(4) These counters have a maximum of 32 if programmed for 50/50 duty cycle. Otherwise, they have a maximum of 16.
(5) High-speed differential I/O mode supports $W = 1$ to 16 and $J = 4, 7, 8,$ or 10.

DLL Jitter

Table 6–93 reports the jitter for the DLL in the DQS phase-shift reference circuit.

Table 6–93. DLL Jitter for DQS Phase Shift Reference Circuit

Frequency (MHz)	DLL Jitter (ps)
197 to 200	± 100
160 to 196	± 300
100 to 159	± 500

7. Reference & Ordering Information

SGX51007-1.0

Software

Stratix® GX devices are supported by the Altera® Quartus® II design software, which provides a comprehensive environment for system-on-a-programmable-chip (SOPC) design. The Quartus II software includes hardware description language and schematic design entry, compilation and logic synthesis, full simulation and advanced timing analysis, SignalTap® logic analysis, and device configuration. See the *Design Software Selector Guide* for more details on the Quartus II software features.

The Quartus II software supports the Windows 2000/NT/98, Sun Solaris, Linux Red Hat v6.2 and HP-UX operating systems. It also supports seamless integration with industry-leading EDA tools through the NativeLink® interface.

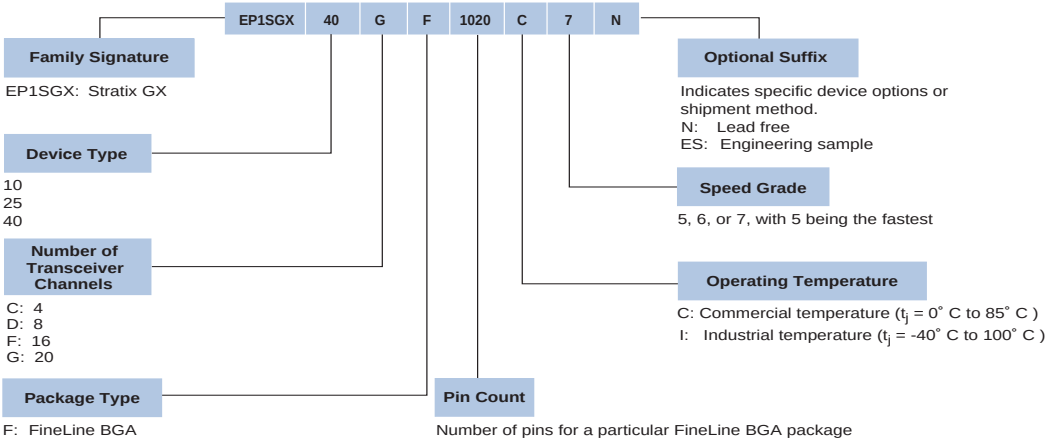
Device Pin-Outs

Device pin-outs for Stratix GX devices will be released on the Altera web site (**www.altera.com**).

Ordering Information

Figure 7–1 describes the ordering codes for Stratix GX devices.

Figure 7–1. Stratix GX Device Packaging Ordering Information





Stratix GX Device Handbook, Volume 1



101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com

SGX5V1-1.2

Copyright © 2006 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Altera Lead-Free

I.S. EN ISO 9001

Contents

Chapter Revision Dates vii

About This Handbook ix
How to Contact Altera ix
Typographic Conventions ix

Section I. Stratix GX Device Family Data Sheet
Revision History ..... Section I–2

Chapter 1. Introduction to the Stratix GX Device Data Sheet
Overview 1–1
Features 1–1
High-Speed I/O Interface Functional Description 1–4
FPGA Functional Description 1–5

Chapter 2. Stratix GX Transceivers
Transmitter Path 2–5
Receiver Path 2–13
Loopback Modes 2–26
BIST (Built-In Self Test) 2–28
Stratix GX Clocking 2–30
Other Transceiver Features 2–37
Individual Power-Down & Reset for the Transmitter & Receiver 2–37
Voltage Reference Capabilities 2–38
Hot-Socketing Capabilities 2–39
Applications & Protocols Supported with Stratix GX Devices 2–39
Stratix GX Example Application Support 2–39
High-Speed Serial Bus Protocols 2–40

Chapter 3. Source-Synchronous Signaling With DPA
Introduction 3–1
Stratix GX I/O Banks 3–1
Principles of SERDES Operation 3–1
DPA Block Overview 3–5
DPA Operation 3–10

Chapter 4. Stratix GX Architecture

Logic Array Blocks 4–1
LAB Interconnects 4–1
LAB Control Signals 4–2
Logic Elements 4–3
LUT Chain & Register Chain 4–5
addnsub Signal 4–5
LE Operating Modes 4–5
Clear & Preset Logic Control 4–10
MultiTrack Interconnect 4–11
TriMatrix Memory 4–18
Memory Modes 4–19
Parity Bit Support 4–21
Shift Register Support 4–21
Memory Block Size 4–22
Independent Clock Mode 4–40
Input/Output Clock Mode 4–42
Read/Write Clock Mode 4–44
Single-Port Mode 4–45
Digital Signal Processing Block 4–46
Multiplier Block 4–52
Adder/Output Blocks 4–56
Modes of Operation 4–59
DSP Block Interface 4–65
PLLs & Clock Networks 4–68
Global & Hierarchical Clocking 4–68
Enhanced & Fast PLLs 4–76
Enhanced PLLs 4–82
Fast PLLs 4–93
I/O Structure 4–96
Double-Data Rate I/O Pins 4–103
External RAM Interfacing 4–107
Programmable Drive Strength 4–110
Open-Drain Output 4–111
Slew-Rate Control 4–112
Bus Hold 4–112
Programmable Pull-Up Resistor 4–113
Advanced I/O Standard Support 4–113
Differential On-Chip Termination 4–118
MultiVolt I/O Interface 4–120
Power Sequencing & Hot Socketing 4–121
IEEE Std. 1149.1 (JTAG) Boundary-Scan Support 4–122

Chapter 5. Configuration & Testing

SignalTap Embedded Logic Analyzer 5–1
Configuration 5–1
Operating Modes 5–1

Configuration Schemes 5–2
Partial Reconfiguration 5–3
Remote Update Configuration Modes 5–3
Stratix GX Automated Single Event Upset (SEU) Detection 5–7
Custom-Built Circuitry 5–8
Software Interface 5–8
Temperature-Sensing Diode 5–8

Chapter 6. DC & Switching Characteristics

Operating Conditions 6–1
Power Consumption 6–22
Timing Model 6–22
Preliminary & Final Timing 6–23
Performance 6–23
Internal Timing Parameters 6–26
External Timing Parameters 6–35
External I/O Delay Parameters 6–44
Maximum Input & Output Clock Rates 6–54
High-Speed I/O Specification 6–58
PLL Timing 6–62
DLL Jitter 6–68

Chapter 7. Reference & Ordering Information

Software 7–1
Device Pin-Outs 7–1
Ordering Information 7–1

Chapter Revision Dates

The chapters in this book, *Stratix GX Device Handbook, Volume 1*, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1. Introduction to the Stratix GX Device Data Sheet
Revised: *February 2005*
Part number: *SGX51001-1.0*

Chapter 2. Stratix GX Transceivers
Revised: *June 2006*
Part number: *SGX51002-1.1*

Chapter 3. Source-Synchronous Signaling With DPA
Revised: *August 2005*
Part number: *SGX51003-1.1*

Chapter 4. Stratix GX Architecture
Revised: *February 2005*
Part number: *SGX51004-1.0*

Chapter 5. Configuration & Testing
Revised: *February 2005*
Part number: *SGX51005-1.0*

Chapter 6. DC & Switching Characteristics
Revised: *June 2006*
Part number: *SGX51006-1.2*

Chapter 7. Reference & Ordering Information
Revised: *February 2005*
Part number: *SGX51007-1.0*

About This Handbook

This handbook provides comprehensive information about the Altera® Stratix® GX family of devices.

How to Contact Altera

For the most up-to-date information about Altera products, go to the Altera world-wide web site at www.altera.com. For technical support on this product, go to www.altera.com/mysupport. For additional information about Altera products, consult the sources shown below.

Information Type	USA & Canada	All Other Locations
Technical support	www.altera.com/mysupport/	www.altera.com/mysupport/
	(800) 800-EPLD (3753) (7:00 a.m. to 5:00 p.m. Pacific Time)	+1 408-544-8767 7:00 a.m. to 5:00 p.m. (GMT -8:00) Pacific Time
Product literature	www.altera.com	www.altera.com
Altera literature services	literature@altera.com	literature@altera.com
Non-technical customer service	(800) 767-3753	+ 1 408-544-7000 7:00 a.m. to 5:00 p.m. (GMT -8:00) Pacific Time
FTP site	ftp.altera.com	ftp.altera.com

Typographic Conventions

This document uses the typographic conventions shown below.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: **Save As** dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: $\mathbf{f_{MAX}}$, **\qdesigns** directory, **d:** drive, **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Document titles are shown in italic type with initial capital letters. Example: *AN 75: High-Speed Board Design.*

Visual Cue	Meaning
Italic type	Internal timing parameters and variables are shown in italic type. Examples: t_{PIA}, $n + 1$. Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: *<file name>*, *<project name>*.**pof** file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
"Subheading Title"	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: "Typographic Conventions."
`Courier type`	Signal and port names are shown in lowercase Courier type. Examples: `data1`, `tdi`, `input.` Active-low signals are denoted by suffix `n`, e.g., `resetn`. Anything that must be typed exactly as it appears is shown in Courier type. For example: `c:\qdesigns\tutorial\chiptrip.gdf`. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword `SUBDESIGN`), as well as logic function names (e.g., `TRI`) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ● •	Bullets are used in a list of items when the sequence of the items is not important.
✓	The checkmark indicates a procedure that consists of one step only.
☞	The hand points to information that requires special attention.
CAUTION	The caution indicates required information that needs special consideration and understanding and should be read prior to starting or continuing with the procedure or process.
	The warning indicates information that should be read prior to starting or continuing the procedure or processes
↵	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information on a particular topic.

Section I. Stratix GX Device Family Data Sheet

This section provides the data sheet specifications for Stratix® GX devices. It contains feature definitions of the internal architecture, configuration information, testing information, DC operating conditions, and AC timing parameters.

This section includes the following chapters:

- Chapter 1, Introduction to the Stratix GX Device Data Sheet
- Chapter 2, Stratix GX Transceivers
- Chapter 3, Source-Synchronous Signaling With DPA
- Chapter 4, Stratix GX Architecture
- Chapter 5, Configuration & Testing
- Chapter 6, DC & Switching Characteristics
- Chapter 7, Reference & Ordering Information

Revision History

The table below shows the revision history for Chapters 1 through 7.

Chapter(s)	Date / Version	Changes Made	Comments
1	February 2005, v1.0	Initial Release.	
2	June 2006, v1.1	• Updated "Serial Loopback" section. • Updated Figures 2–1 through 2–3. • Updated Figure 2–13. • Updated Figures 2–26 and 2–27.	
	February 2005, v1.0	Initial Release.	
3	August 2005, v1.1	Added Note (3) to Figure 3-7.	
4	February 2005, v1.0	Initial Release.	
5	February 2005, v1.0	Initial Release.	
6	June 2006, v1.2	• Updated "Operating Conditions" section. • Updated Table 6–4. • Updated note 3 in Table 6–6. • Added note 12 in Table 6–7. • Updated Figure 6–1. • Added Figure 6–2. • Updated Tables 6–13 through 6–16.	• Changed V_{OD} to V_{ID} for receiver input voltage and `refclkb` input voltage in Table 6–4. • Changed value for undershoot during transition from -0.5 V to -2.0 V in note 3 of Table 6–6. • Changed value of V_{OCM} from mV to V in Table 6–15. • Changed unit value of W to Ω.
	August 2005, v1.1	Updated Tables 6-7 and 6-50.	
7	February 2005, v1.0	Initial Release.	

1. Introduction to the Stratix GX Device Data Sheet

SGX51001-1.0

Overview

The Stratix® GX family of devices is Altera's second FPGA family to combine high-speed serial transceivers with a scalable, high-performance logic array. Stratix GX devices include 4 to 20 high-speed transceiver channels, each incorporating clock data recovery (CDR) technology and embedded SERDES capability at data rates of up to 3.1875 gigabits per second (Gbps). These transceivers are grouped by four-channel transceiver blocks, and are designed for low power consumption and small die size. The Stratix GX FPGA technology is built upon the Stratix architecture, and offers a 1.5-V logic array with unmatched performance, flexibility, and time-to-market capabilities. This scalable, high-performance architecture makes Stratix GX devices ideal for high-speed backplane interface, chip-to-chip, and communications protocol-bridging applications.

Features

- Transceiver block features are as follows:
  - High-speed serial transceiver channels with CDR provides 500-megabits per second (Mbps) to 3.1875-Gbps full-duplex operation
  - Devices are available with 4, 8, 16, or 20 high-speed serial transceiver channels providing up to 127.5 Gbps of full-duplex serial bandwidth
  - Support for transceiver-based protocols, including 10 Gigabit Ethernet attachment unit interface (XAUI), Gigabit Ethernet (GigE), and SONET/SDH
  - Compatible with PCI Express, SMPTE 292M, Fibre Channel, and Serial RapidIO I/O standards
  - Programmable differential output voltage (V_{OD}), pre-emphasis, and equalization settings for improved signal integrity
  - Individual transmitter and receiver channel power-down capability implemented automatically by the Quartus® II software for reduced power consumption during non-operation
  - Programmable transceiver-to-FPGA interface with support for 8-, 10-, 16-, and 20-bit wide data paths
  - 1.5-V pseudo current mode logic (PCML) for 500 Mbps to 3.1875 Gbps
  - Support for LVDS, LVPECL, and 3.3-V PCML on reference clocks and receiver input pins (AC-coupled)
  - Built-in self test (BIST)
  - Hot insertion/removal protection circuitry

- Pattern detector and word aligner supports programmable patterns
- 8B/10B encoder/decoder performs 8- to 10-bit encoding and 10- to 8-bit decoding
- Rate matcher compliant with IEEE 802.3-2002 for GigE mode and with IEEE 802-3ae for XAUI mode
- Channel bonding compliant with IEEE 802.3ae (for XAUI mode only)
- Device can bypass some transceiver block features if necessary

- FPGA features are as follows:
  - 10,570 to 41,250 logic elements (LEs); see Table 1–1
  - Up to 3,423,744 RAM bits (427,968 bytes) available without reducing logic resources
  - TriMatrix™ memory consisting of three RAM block sizes to implement true dual-port memory and first-in-out (FIFO) buffers
  - Up to 16 global clock networks with up to 22 regional clock networks per device region
  - High-speed DSP blocks provide dedicated implementation of multipliers (faster than 300 MHz), multiply-accumulate functions, and finite impulse response (FIR) filters
  - Up to eight general usage phase-locked loops (four enhanced PLLs and four fast PLLs) per device provide spread spectrum, programmable bandwidth, clock switchover, real-time PLL reconfiguration, and advanced multiplication and phase shifting
  - Support for numerous single-ended and differential I/O standards
  - High-speed source-synchronous differential I/O support on up to 45 channels for 1-Gbps performance
  - Support for source-synchronous bus standards, including 10-Gigabit Ethernet XSBI, Parallel RapidIO, UTOPIA IV, Network Packet Streaming Interface (NPSI), HyperTransport™ technology, SPI-4 Phase 2 (POS-PHY Level 4), and SFI-4
  - Support for high-speed external memory, including zero bus turnaround (ZBT) SRAM, quad data rate (QDR and QDRII) SRAM, double data rate (DDR) SDRAM, DDR fast cycle RAM (FCRAM), and single data rate (SDR) SDRAM
  - Support for multiple intellectual property megafunctions from Altera® MegaCore® functions and Altera Megafunction Partners Program (AMPP℠) megafunctions
  - Support for remote configuration updates
  - Dynamic phase alignment on LVDS receiver channels

Table 1–1. Stratix GX Device Features

Feature	EP1SGX10C EP1SGX10D	EP1SGX25C EP1SGX25D EP1SGX25F	EP1SGX40D EP1SGX40G
LEs	10,570	25,660	41,250
Transceiver channels	4, 8	4, 8, 16	8, 20
Source-synchronous channels	22	39	45
M512 RAM blocks (32 × 18 bits)	94	224	384
M4K RAM blocks (128 × 36 bits)	60	138	183
M-RAM blocks (4K ×144 bits)	1	2	4
Total RAM bits	920,448	1,944,576	3,423,744
Digital signal processing (DSP) blocks	6	10	14
Embedded multipliers *(1)*	48	80	112
PLLs	4	4	8

Note to Table 1–1:

(1) This parameter lists the total number of 9- × 9-bit multipliers for each device. For the total number of 18- × 18-bit multipliers per device, divide the total number of 9- × 9-bit multipliers by 2. For the total number of 36- × 36-bit multipliers per device, decide the total number of 9- × 9-bit multipliers by 8.

Stratix GX devices are available in space-saving FineLine BGA® packages (refer to Tables 1–2 and 1–3), and in multiple speed grades (refer to Table 1–4). Stratix GX devices support vertical migration within the same package (that is, you can migrate between the EP1SGX10C and EP1SGX25C devices in the 672-pin FineLine BGA package). See the Stratix GX device pin tables for more information. Vertical migration means that you can migrate to devices whose dedicated pins, configuration pins, and power pins are the same for a given package across device densities. For I/O pin migration across densities, you must cross-reference the available I/O pins using the device pin-outs for all planned densities of a given package type, to identify which I/O pins it is possible to migrate. The Quartus II software can automatically cross reference and place all pins for migration when given a device migration list.

Table 1–2. Stratix GX Package Options & I/O Pin Counts (Part 1 of 2) *Note (1)*

Device	672-Pin FineLine BGA	1,020-Pin FineLine BGA
EP1SGX10C	362	
EP1SGX10D	362	
EP1SGX25C	455	

Table 1–2. Stratix GX Package Options & I/O Pin Counts (Part 2 of 2) *Note (1)*

Device	672-Pin FineLine BGA	1,020-Pin FineLine BGA
EP1SGX25D	455	607
EP1SGX25F		607
EP1SGX40D		624
EP1SGX40G		624

Note to Table 1–2:

(1) The number of I/O pins listed for each package includes dedicated clock pins and dedicated fast I/O pins. However, these numbers do not include high-speed or clock reference pins for high-speed I/O standards.

Table 1–3. Stratix GX FineLine BGA Package Sizes

Dimension	672 Pin	1,020 Pin
Pitch (mm)	1.00	1.00
Area (mm^2)	729	1,089
Length × width (mm × mm)	27 × 27	33 × 33

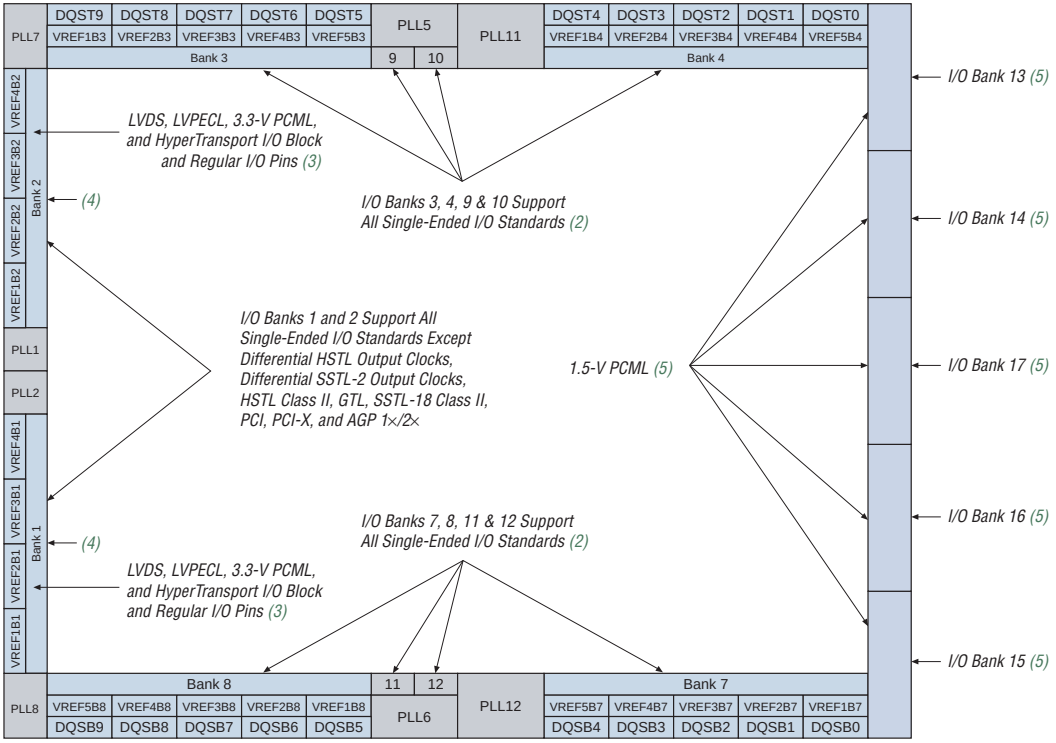
Table 1–4. Stratix GX Device Speed Grades

Device	672-Pin FineLine BGA	1,020-pin FineLine BGA
EP1SGX10	-5, -6, -7	
EP1SGX25	-5, -6, -7	-5, -6, -7
EP1SGX40		-5, -6, -7

High-Speed I/O Interface Functional Description

The Stratix GX device family supports high-speed serial transceiver blocks with CDR circuitry as well as source-synchronous interfaces. The channels on the right side of the device use an embedded circuit dedicated for receiving and transmitting high-speed serial data streams to and from the system board. These channels are clustered in a four-channel serial transceiver building block and deliver high-speed bidirectional point-to-point data transmissions to provide up to 3.1875 Gbps of full-duplex data transmission per channel. The channels on the left side of the device support source-synchronous data transfers at up to 1 Gbps using LVDS, LVPECL, 3.3-V PCML, or HyperTransport technology I/O standards. Figure 1–1 shows the Stratix GX I/O blocks. The differential source-synchronous serial interface and the high-speed serial interface are described in the *Stratix GX Transceivers* chapter of the *Stratix GX Device Handbook, Volume 1*.

Figure 1–1. Stratix GX I/O Blocks *Note (1)*



Notes to Figure 1–1:

(1) Figure 1–1 is a top view of the Stratix GX silicon die.
(2) Banks 9 through 12 are enhanced PLL external clock output banks.
(3) If the high-speed differential I/O pins are not used for high-speed differential signaling, they can support all of the I/O standards except HSTL class I and II, GTL, SSTL-18 Class II, PCI, PCI-X, and AGP 1×/2×.
(4) For guidelines for placing single-ended I/O pads next to differential I/O pads, see the *Selectable I/O Standards in Stratix & Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2*.
(5) These I/O banks in Stratix GX devices also support the LVDS, LVPECL, and 3.3-V PCML I/O standards on reference clocks and receiver input pins (AC coupled).

FPGA Functional Description

Stratix GX devices contain a two-dimensional row- and column-based architecture to implement custom logic. A series of column and row interconnects of varying length and speed provide signal interconnects between logic array blocks (LABs), memory block structures, and DSP blocks.

The logic array consists of LABs, with 10 logic elements (LEs) in each LAB. An LE is a small unit of logic providing efficient implementation of user logic functions. LABs are grouped into rows and columns across the device.

M512 RAM blocks are simple dual-port memory blocks with 512 bits plus parity (576 bits). These blocks provide dedicated simple dual-port or single-port memory up to 18-bits wide at up to 318 MHz. M512 blocks are grouped into columns across the device in between certain LABs.

M4K RAM blocks are true dual-port memory blocks with 4K bits plus parity (4,608 bits). These blocks provide dedicated true dual-port, simple dual-port, or single-port memory up to 36-bits wide at up to 291 MHz. These blocks are grouped into columns across the device in between certain LABs.

M-RAM blocks are true dual-port memory blocks with 512K bits plus parity (589,824 bits). These blocks provide dedicated true dual-port, simple dual-port, or single-port memory up to 144-bits wide at up to 269 MHz. Several M-RAM blocks are located individually or in pairs within the device's logic array.

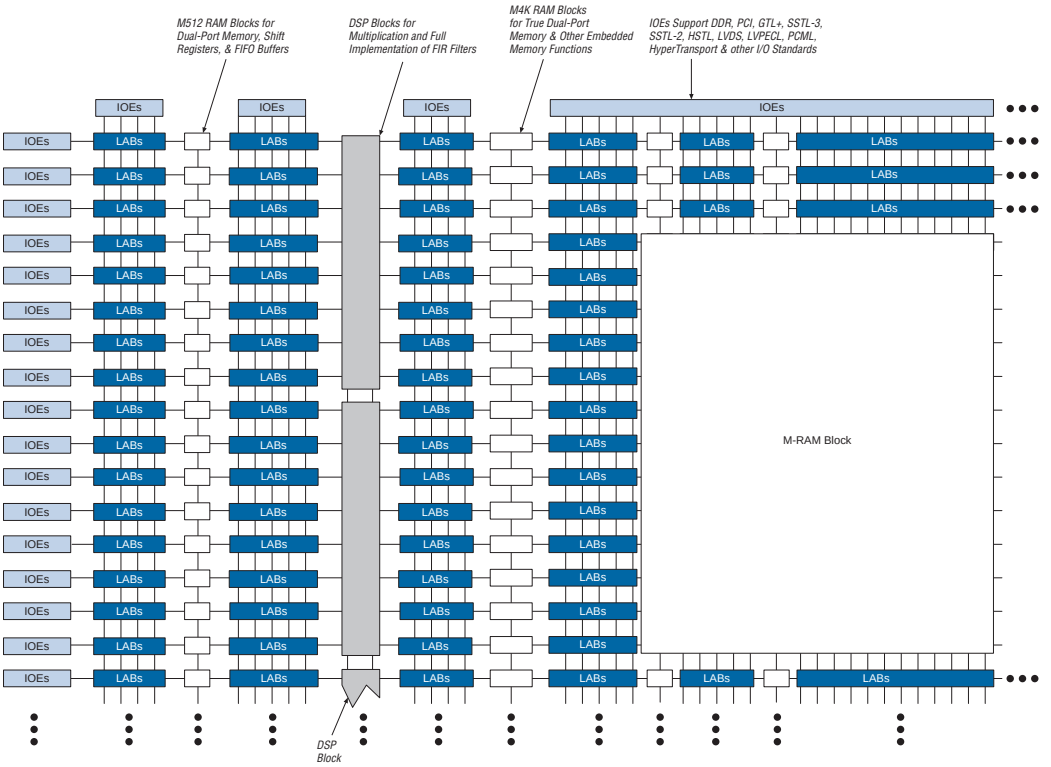
Digital signal processing (DSP) blocks can implement up to either eight full-precision 9 × 9-bit multipliers, four full-precision 18 × 18-bit multipliers, or one full-precision 36 × 36-bit multiplier with add or subtract features. These blocks also contain 18-bit input shift registers for digital signal processing applications, including FIR and infinite impulse response (IIR) filters. DSP blocks are grouped into two columns in each device.

Each Stratix GX device I/O pin is fed by an I/O element (IOE) located at the end of LAB rows and columns around the periphery of the device. I/O pins support numerous single-ended and differential I/O standards. Each IOE contains a bidirectional I/O buffer and six registers for registering input, output, and output-enable signals. When used with dedicated clocks, these registers provide exceptional performance and interface support with external memory devices such as DDR SDRAM, FCRAM, ZBT, and QDR SRAM devices.

High-speed serial interface channels support transfers at up to 840 Mbps using LVDS, LVPECL, 3.3-V PCML, or HyperTransport technology I/O standards.

Figure 1–2 shows an overview of the Stratix GX device.

Figure 1–2. Stratix GX Block Diagram



The number of M512 RAM, M4K RAM, and DSP blocks varies by device along with row and column numbers and M-RAM blocks. Table 1–5 lists the resources available in Stratix GX devices.

Table 1–5. Stratix GX Device Resources

Device	M512 RAM Columns/Blocks	M4K RAM Columns/Blocks	M-RAM Blocks	DSP Block Columns/Blocks	LAB Columns	LAB Rows
EP1SGX10	4 / 94	2 / 60	1	2 / 6	40	30
EP1SGX25	6 / 224	3 / 138	2	2 / 10	62	46
EP1SGX40	8 / 384	3 / 183	4	2 / 14	77	61

2. Stratix GX Transceivers

SGX51002-1.1

Transceiver Blocks

Stratix® GX devices incorporate dedicated embedded circuitry on the right side of the device, which contains up to 20 high-speed 3.1875-Gbps serial transceiver channels. Each Stratix GX transceiver block contains four full-duplex channels and supporting logic to transmit and receive high-speed serial data streams. The transceiver block uses the channels to deliver bidirectional point-to-point data transmissions with up to 3.1875 Gbps of data transition per channel.

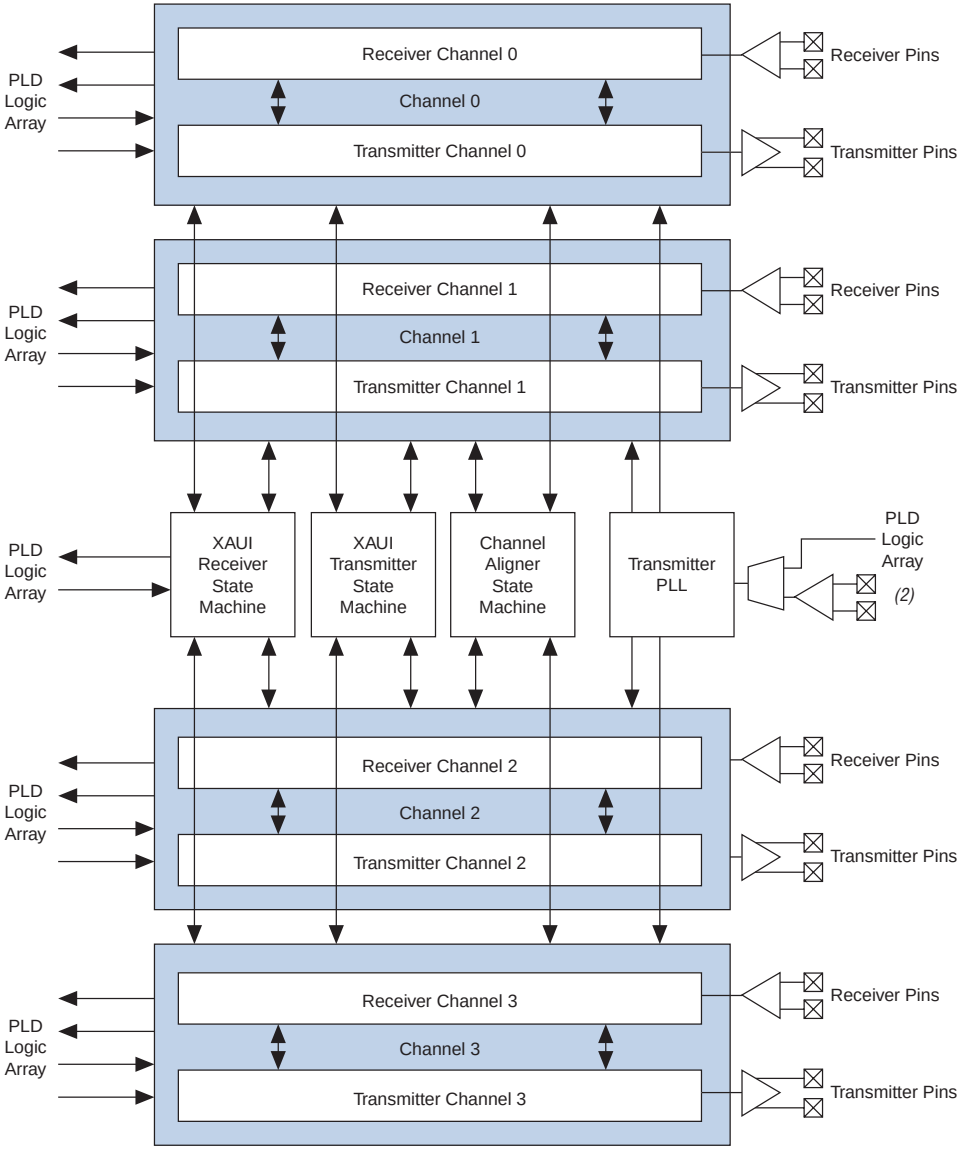
There are up to 20 transceiver channels available on a single Stratix GX device. Table 2–1 shows the number of transceiver channels available on each Stratix GX device.

Table 2–1. Stratix GX Transceiver Channels

Device	Number of Transceiver Channels
EP1SGX10C	4
EP1SGX10D	8
EP1SGX25C	4
EP1SGX25D	8
EP1SGX25F	16
EP1SGX40D	8
EP1SGX40G	20

Figure 2–1 shows the elements of the transceiver block, including the four channels, supporting logic, and I/O buffers. Each transceiver channel consists of a receiver and transmitter. The supporting logic contains a transmitter PLL to generate a high-speed clock used by the four transmitters. The receiver PLL within each transceiver channel generates the receiver reference clocks. The supporting logic also contains state machines to manage rate matching for XAUI and GIGE applications, in addition to channel bonding for XAUI applications.

Figure 2–1. Stratix GX Transceiver Block *Note (1)*



Notes to Figure 2–1:

(1) Each receiver channel has its own PLL and CRU, which are not shown in this diagram. For more information, refer to the section "Receiver Path" on page 2–13.

(2) For possible transmitter PLL clock inputs, refer to the section "Transmitter Path" on page 2–5.

Each Stratix GX transceiver channel consists of a transmitter and receiver. The transmitter contains the following:

- Transmitter PLL
- Transmitter phase compensation FIFO buffer
- Byte serializer
- 8B/10B encoder
- Serializer (parallel to serial converter)
- Transmitter output buffer

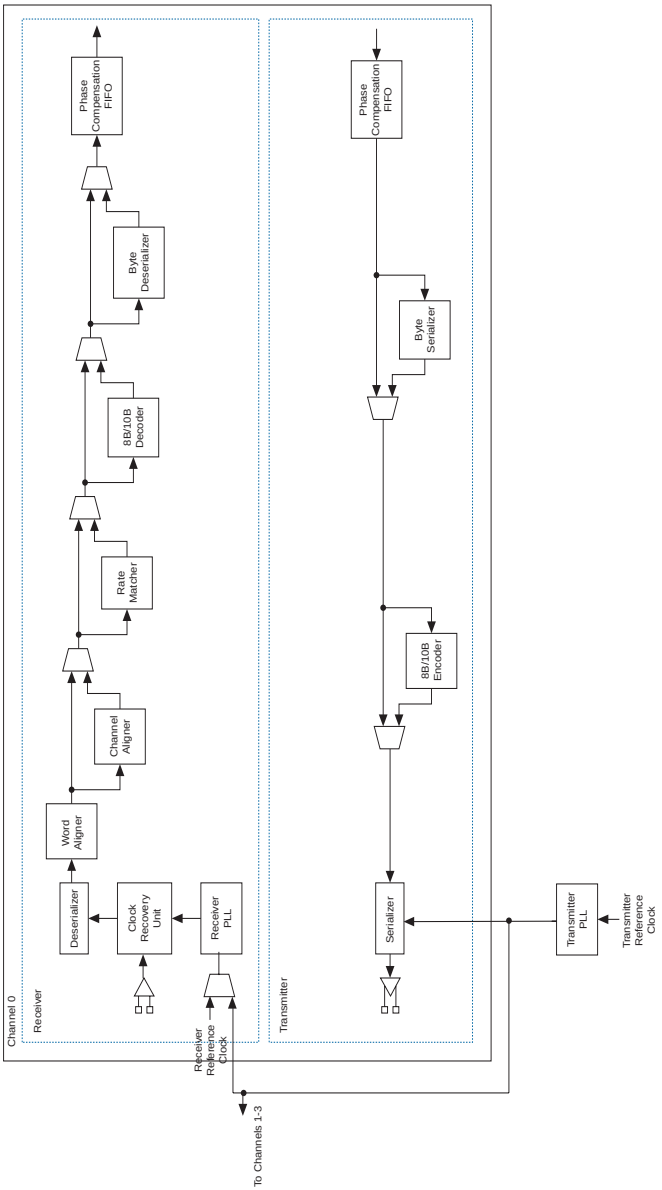
The receiver contains the following:

- Input buffer
- Clock recovery unit (CRU)
- Deserializer
- Pattern detector and word aligner
- Rate matcher and channel aligner
- 8B/10B decoder
- Receiver logic array interface

You can set all the Stratix GX transceiver functions through the Quartus II software. You can set programmable pre-emphasis, programmable equalizer, and programmable V_{OD} dynamically as well. Each Stratix GX transceiver channel is also capable of BIST generation and verification in addition to various loopback modes. Figure 2–2 shows the block diagram for the Stratix GX transceiver channel.

Stratix GX transceivers provide physical coding sublayer (PCS) and physical media attachment (PMA) implementation for protocols such as 10-gigabit XAUI and GIGE. The PCS portion of the transceiver consists of the logic array interface, 8B/10B encoder/decoder, pattern detector, word aligner, rate matcher, channel aligner, and the BIST and pseudo-random binary sequence pattern generator/verifier. The PMA portion of the transceiver consists of the serializer/deserializer, the CRU, and the I/O buffers.

Figure 2–2. Stratix GX Transceiver ChanneL Note (1)



Note to Figure 2–2:
(1) There are four transceiver channels in a transceiver block.

Transmitter Path

This section describes the data path through the Stratix GX transmitter (see Figure 2–2). Data travels through the Stratix GX transmitter via the following modules:

- Transmitter PLL
- Transmitter phase compensation FIFO buffer
- Byte serializer
- 8B/10B encoder
- Serializer (parallel to serial converter)
- Transmitter output buffer

Transmitter PLL

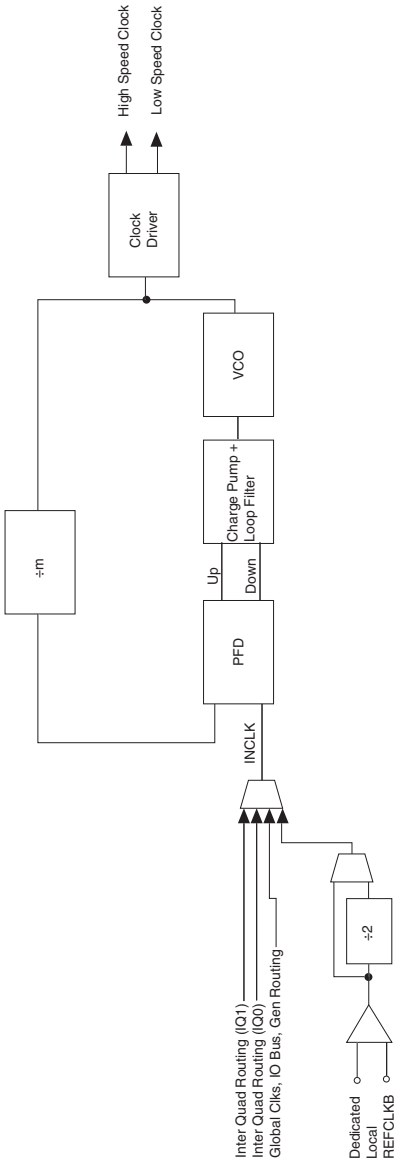
Each transceiver block has one transmitter PLL, which receives the reference clock and generates the following signals:

- High-speed serial clock used by the serializer
- Slow-speed reference clock used by the receiver
- Slow-speed clock used by the logic array (divisible by two for double-width mode)

The INCLK clock is the input into the transmitter PLL. There is one INCLK clock per transceiver block. This clock can be fed by either the REFCLKB pin, PLD routing, or the inter-transceiver routing line. See the section "Stratix GX Clocking" on page 2–30 for more information about the inter-transceiver lines.

The transmitter PLL in each transceiver block clocks the circuits in the transmit path. The transmitter PLL is also used to train the receiver PLL. If no transmit channels are used in the transceiver block, the transmitter PLL can be turned off. Figure 2–3 is a block diagram of the transmitter PLL.

Figure 2–3. Transmitter PLL Block Diagram *Note (1)*



Note to Figure 2–3:
(1) The divider in the PLL divides by 4, 8, 10, 16, or 20.

The transmitter PLL can support up to 3.1875 Mbps. The input clock frequency for –5 and –6 speed grade devices is limited to 650 MHz if you use the REFCLKB pin or to 325 MHz if you use the other clock routing resources. For –7 speed grade devices, the maximum input clock frequency is 312.5 MHz with the REFCLKB pin, and the maximum is 156.25 MHz for all other clock routing resources. An optional PLL_LOCKED port is available to indicate whether the transmitter PLL is locked to the reference clock. The transmitter PLL has a programmable loop bandwidth that can be set to low or high. The loop bandwidth parameter can be statically set in the Quartus II software.

Table 2–2 lists the adjustable parameters in the transmitter PLL.

Table 2–2. Transmitter PLL Specifications

Parameter	Specifications
Input reference frequency range	25 MHz to 650 MHz
Data rate support	500 Mbps to 3.1875 Gbps
Multiplication factor (W)	2, 4, 5, 8, 10, 16, or 20 *(1)*
Bandwidth	Low, high

Note to Table 2–2:
(1) Multiplication factors 2 and 5 can only be achieved with the use of the pre-divider on the REFCLKB pin.

Transmitter Phase Compensation FIFO Buffer

The transmitter phase compensation FIFO buffer resides in the transceiver block at the PLD boundary. This FIFO buffer compensates for the phase differences between the transmitter reference clock (inclk) and the PLD interface clock (tx_coreclk). The phase difference between the two clocks must be less than 360° . The PLD interface clock must also be frequency locked to the transmitter reference clock. The phase compensation FIFO buffer is four words deep and cannot be bypassed.

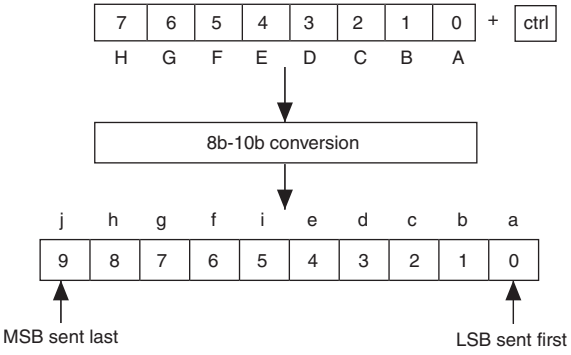
Byte Serializer

The byte serializer takes double-width words (16 or 20 bits) from the PLD interface and converts them to a single width word (8 or 10 bits) for use in the transceiver. The transmit data path after the byte serializer is single width (8 or 10 bits). The byte serializer is bypassed when single width mode (8 or 10 bits) is used at the PLD interface.

8B/10B Encoder

The 8B/10B encoder translates 8-bit wide data + 1 control enable bit into a 10-bit encoded data. The encoded data has a maximum run length of 5. The 8B/10B encoder can be bypassed. Figure 2–4 diagrams the encoding process.

Figure 2–4. Encoding Process



Transmit State Machine

The transmit state machine operates in either XAUI mode or in GIGE mode, depending on the protocol used.

GIGE Mode

In GIGE mode, the transmit state machines convert all idle ordered sets (/K28.5/, /Dx.y/) to either /I1/ or /I2/ ordered sets. /I1/ consists of a negative-ending disparity /K28.5/ (denoted by /K28.5/-) followed by a neutral /D5.6/. /I2/ consists of a positive-ending disparity /K28.5/ (denoted by /K28.5/+) and a negative-ending disparity /D16.2/ (denoted by /D16.2/-). The transmit state machines do not convert any of the ordered sets to match /C1/ or /C2/, which are the configuration ordered sets. (/C1/ and /C2/ are defined by (/K28.5/, /D21.5/) and (/K28.5/, /D2.2/), respectively.) Both the /I1/ and /I2/ ordered sets guarantee a negative-ending disparity after each ordered set. The GIGE transmit state machine can be statically disabled in the Quartus II software, even if using the GIGE protocol mode.

XAUI Mode

The transmit state machine translates the XAUI XGMII code group to the XAUI PCS code group. Table 2–3 shows the code conversion.

Table 2–3. Code Conversion

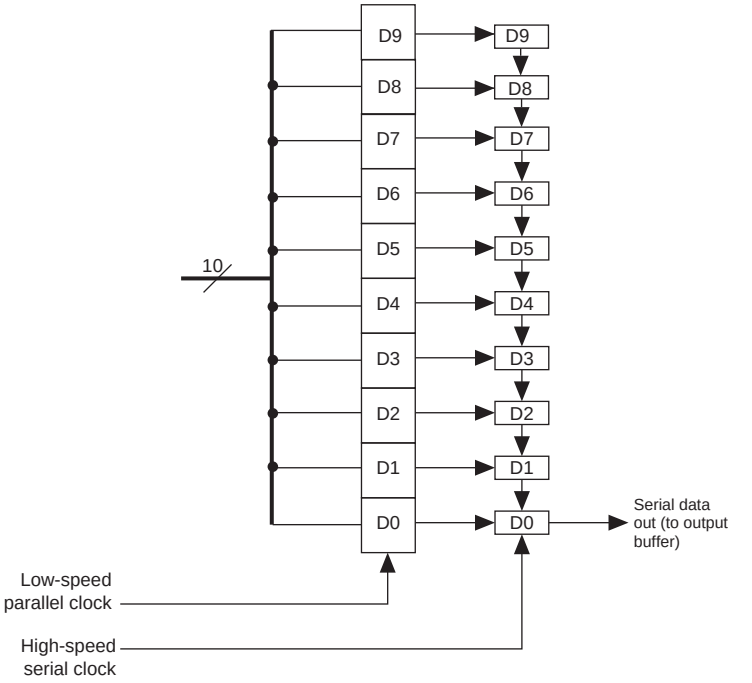
XGMII TXC	XGMII TXD	PCS Code-Group	Description
0	00 through FF	Dxx.y	Normal data
1	07	K28.0 or K28.3 or K28.5	Idle in ||**I**||
1	07	K28.5	Idle in ||**T**||
1	9C	K28.4	Sequence
1	FB	K27.7	Start
1	FD	K29.7	Terminate
1	FE	K30.7	Error
1	See IEEE 802.3 reserved code groups	See IEEE 802.3 reserved code groups	Reserved code groups
1	Other value	K30.7	Invalid XGMII character

The XAUI PCS idle code groups, `/K28.0/` (`/R/`) and `/K28.5/` (`/K/`), are automatically randomized based on a PRBS7 pattern with an x^7+x^6+1 polynomial. The `/K28.3/` (`/A/`) code group is automatically generated between 16 and 31 idle code groups. The idle randomization on the `/A/`, `/K/`, and `/R/` code groups are done automatically by the transmit state machine.

Serializer (Parallel-to-Serial Converter)

The serializer converts the parallel 8-bit or 10-bit data into a serial stream, transmitting the LSB first. The serialized stream is then fed to the transmit buffer. Figure 2–5 is a diagram of the serializer.

Figure 2–5. Serializer

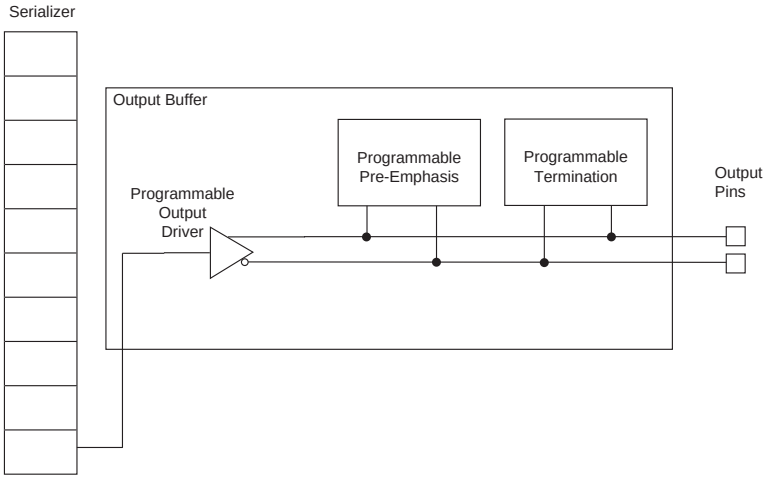


Transmit Buffer

The Stratix GX transceiver buffers support the 1.5-V pseudo current mode logic (PCML) I/O standard at a rate up to 3.1875 Gbps, across up to 40 inches of FR4 trace, and across 2 connectors. Additional I/O standards, LVDS, 3.3-V PCML, LVPECL, can be supported when AC coupled. The common mode of the output driver is 750 mV.

The output buffer, as shown in Figure 2–6, consists of a programmable output driver and a programmable pre-emphasis circuit.

Figure 2–6. Output Buffer



Programmable Output Driver

The programmable output driver can be set to drive out 400 to 1,600 mV. Table 2–4 shows the available settings for each termination value. The V_{OD} can be dynamically or statically set. The output driver requires either internal or external termination at the source.

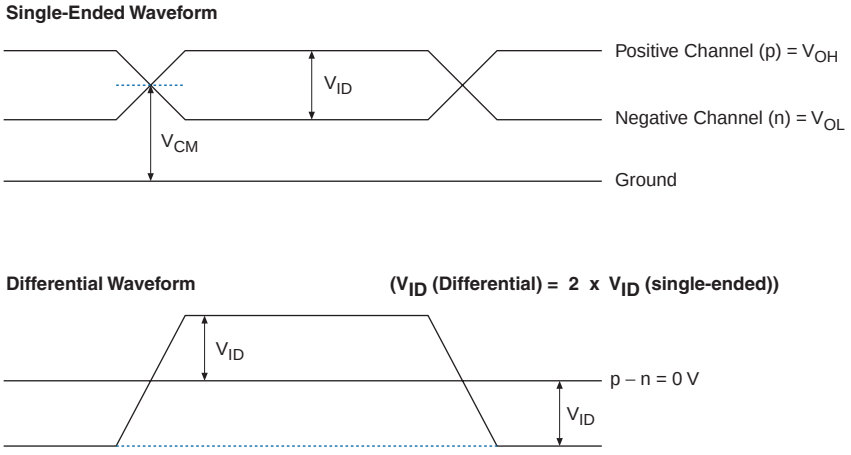
Table 2–4. Programmable V_{OD} (Differential) *Note (1)*

Termination Setting (Ω)	V_{OD} Setting (mV)
100	400, 800, 1000, 1200, 1400, 1600
120	480, 960, 1200, 1440
150	600, 1200, 1500

Note to Table 2–4:

(1) V_{OD} differential is measured as $V_A - V_B$ (see Figure 2–7).

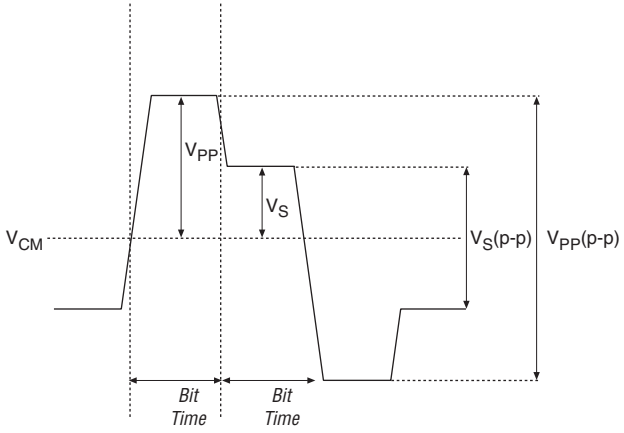
Figure 2–7. V_{OD} Differential



Programmable Pre-Emphasis

The programmable pre-emphasis module controls the output driver to boost the high frequency components, to compensate for losses in the transmission medium, as shown in Figure 2–8. The pre-emphasis can be dynamically or statically set. There are five possible pre-emphasis settings (1 through 5), with 5 being the highest and 0 being no pre-emphasis.

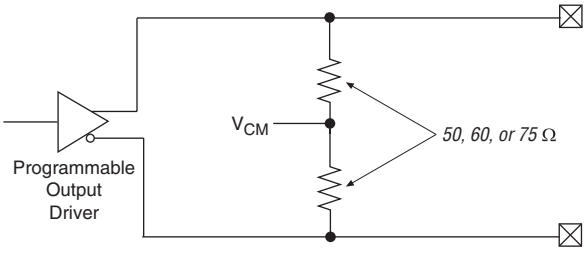
Figure 2–8. Programmable Pre-Emphasis Model



Pre-emphasis percentage is defined as $V_{PP}/V_S - 1$, where V_{PP} is the differential emphasized voltage (peak-to-peak) and V_S is the differential steady-state voltage (peak-to-peak).

Programmable Transmitter Termination
The programmable termination can be statically set in the Quartus II software. The values are 100 Ω, 120 Ω, 150 Ω, and off. Figure 2–9 shows the setup for programmable termination.

Figure 2–9. Programmable Transmitter Termination



Receiver Path

This section describes the data path through the Stratix GX receiver (refer to Figure 2–2 on page 2–4). Data travels through the Stratix GX receiver via the following modules:

- Input buffer
- Clock Recovery Unit (CRU)
- Deserializer
- Pattern detector and word aligner
- Rate matcher and channel aligner
- 8B/10B decoder
- Receiver logic array interface

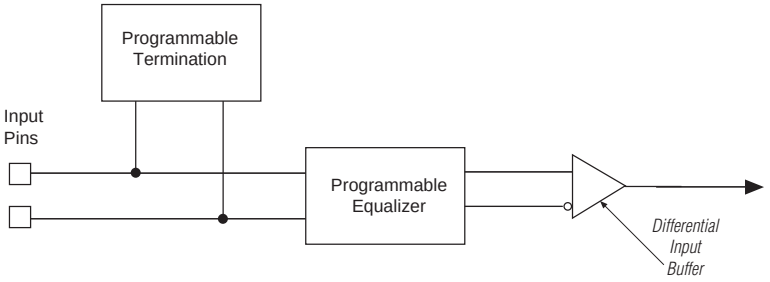
Receiver Input Buffer

The Stratix GX receiver input buffer supports the 1.5-V PCML I/O standard at a rate up to 3.1875 Gbps. Additional I/O standards, LVDS, 3.3-V PCML, and LVPECL can be supported when AC coupled. The common mode of the input buffer is 1.1 V. The receiver can support Stratix GX-to-Stratix GX DC coupling.

Figure 2–10 shows a diagram of the receiver input buffer, which contains:

- Programmable termination
- Programmable equalizer

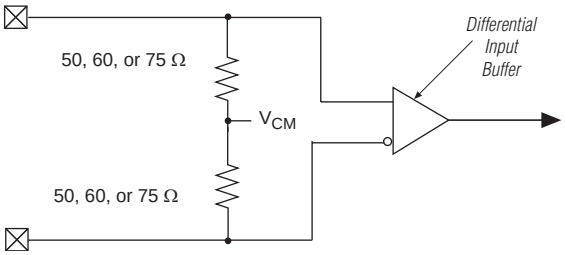
Figure 2–10. Receiver Input Buffer



Programmable Termination

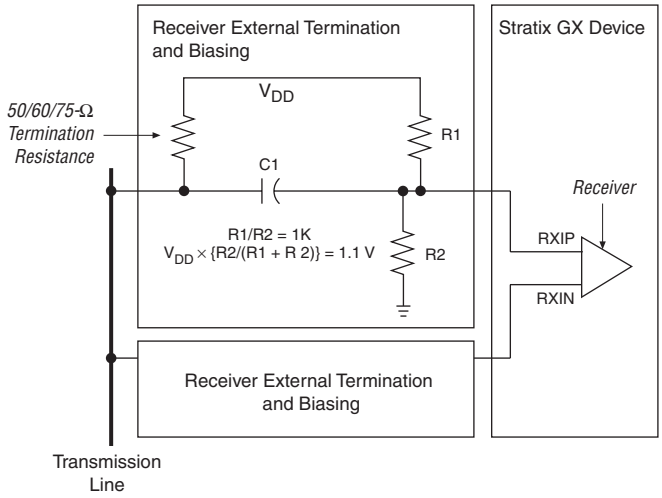
The programmable termination can be statically set in the Quartus II software. Figure 2–11 shows the setup for programmable receiver termination.

Figure 2–11. Programmable Receiver Termination



If you use external termination, then the receiver must be externally terminated and biased to 1.1 V. Figure 2–12 shows an example of an external termination/biasing circuit.

Figure 2–12. External Termination & Biasing Circuit



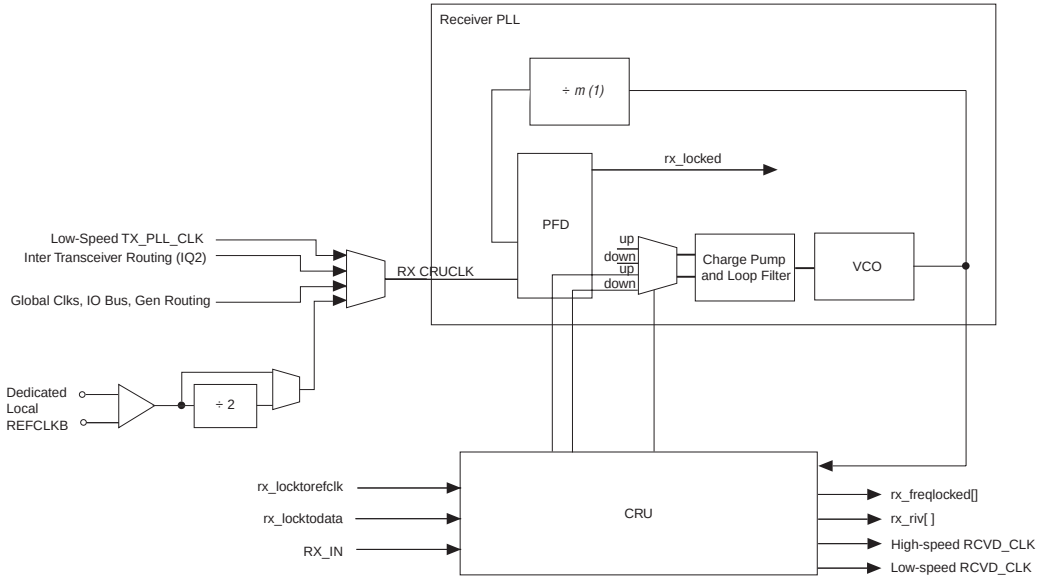
Programmable Equalizer

The programmable equalizer module boosts the high frequency components of the incoming signal to compensate for losses in the transmission medium. There are five possible equalization settings (0, 1, 2, 3, 4) to compensate for 0", 10", 20", 30", and 40" of FR4 trace. These settings should be interpreted loosely. The programmable equalizer can be set dynamically or statically.

Receiver PLL & CRU

Each transceiver block has four receiver PLLs and CRUs, each of which is dedicated to a receive channel. If the receive channel associated with a particular receiver PLL or CRU is not used, then the receiver PLL or CRU is powered down for the channel. Figure 2–13 is a diagram of the receiver PLL and CRU circuits.

Figure 2–13. Receiver PLL & CRU Circuit



Note to Figure 2–13:
(1) m = 8, 10 16, or 20.

The receiver PLLs and CRUs are capable of supporting up to 3.1875 Gbps. The input clock frequency for –5 and –6 speed grade devices is limited to 650 MHz if you use the REFCLKB pin or 325 MHz if you use the other clock routing resources. The maximum input clock frequency for –7 speed grade devices is 312.5 MHz if you use the REFCLKB pin or 156.25 MHz with the other clock routing resources. An optional RX_LOCKED port (active low signal) is available to indicate whether the PLL is locked to the reference clock. The receiver PLL has a programmable loop bandwidth, which can be set to low, medium, or high. The loop bandwidth parameter can be statically set by the Quartus II software.

Table 2–5 lists the adjustable parameters of the receiver PLL and CRU. All the parameters listed are statically programmable in the Quartus II software.

Table 2–5. Receiver PLL & CRU Adjustable Parameters (Part 1 of 2)

Parameter	Specifications
Input reference frequency range	25 MHz to 650 MHz
Data rate support	500 Mbps to 3.1875 Gbps

Table 2–5. Receiver PLL & CRU Adjustable Parameters (Part 2 of 2)	
Multiplication factor (W)	2, 4, 5, 8, 10, 16, or 20 *(1)*
PPM detector	125, 250, 500, 1,000
Bandwidth	Low, medium, high
Run length detector	10-bit or 20-bit mode: 5 to 160 in steps of 5
	8-bit or 16-bit mode: 4 to 128 in steps of 4

Note to Table 2–5:

(1) Multiplication factors 2, 4, and 5 can only be achieved with the use of the pre-divider on the REFCLKB port or if the CRU is trained with the low speed clock from the transmitter PLL.

The CRU has a built-in switchover circuit to select whether the voltage-controlled oscillator of the PLL is trained by the reference clock or the data. The optional port `rx_freqlocked` monitors when the CRU is in locked to data mode.

In the automatic mode, the following conditions must be met for the CRU to switch from locked to reference to locked to data mode:

- The CRU PLL is within the prescribed PPM frequency threshold setting (125 PPM, 250 PPM, 500 PPM, 1,000 PPM) of the CRU reference clock.
- The reference clock and CRU PLL output are phase matched (phases are within .08 UI).

The automatic switchover circuit can be overridden by using the optional ports `rx_lockedtorefclk` and `rx_locktodata`. Table 2–6 shows the possible combinations of these two signals.

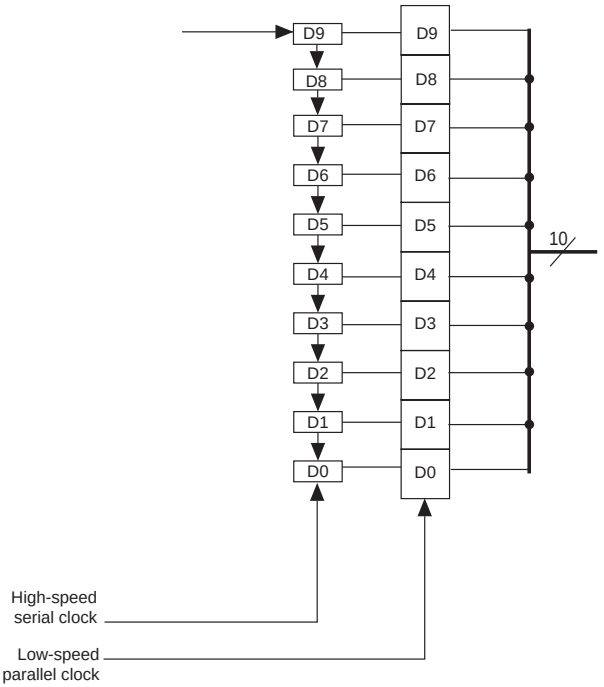
Table 2–6. Possible Combinations of rx_lockedtorefclk & rx_locktodata		
rx_locktodata	**rx_lockedtorefclk**	**VCO (lock to mode)**
0	0	Auto
0	1	Reference CLK
1	x	DATA

If the `rx_lockedtorefclk` and `rx_locktodata` ports are not used, the default is auto mode.

Deserializer (Serial-to-Parallel Converter)

The deserializer converts the serial stream into a parallel 8- or 10-bit data bus. The deserializer receives the least significant bit first. Figure 2–14 is a diagram of the deserializer.

Figure 2–14. Deserializer

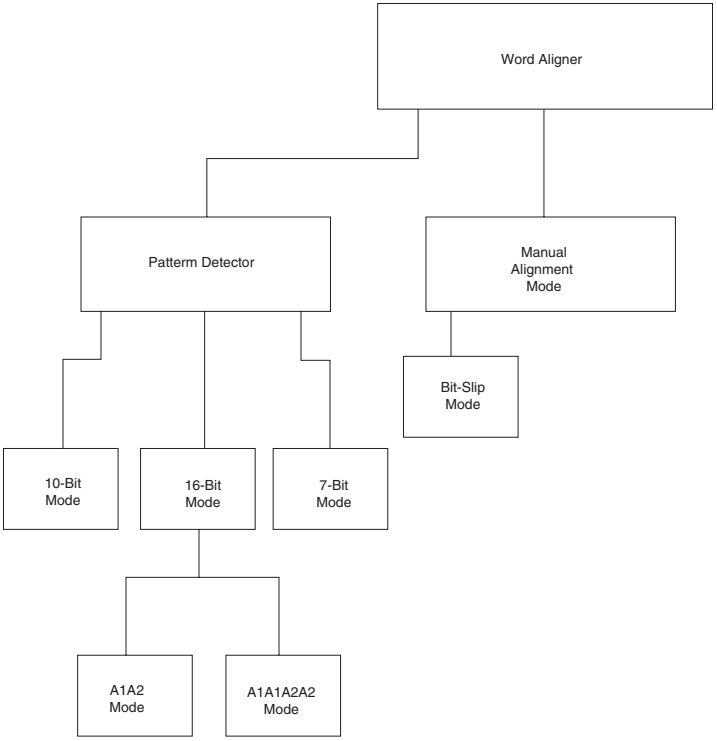


Word Aligner

The word aligner aligns the incoming data based on the specific byte boundaries. The word aligner has three customizable modes of operation: bit-slip mode, 16-bit mode, and 10-bit mode, the last of which is available for the basic and SONET modes. The word aligner also has two non-customizable modes of operation, which are the XAUI and GIGE modes.

Figure 2–15 shows the word aligner in bit-slip mode.

Figure 2–15. Word Aligner in Bit-Slip Mode



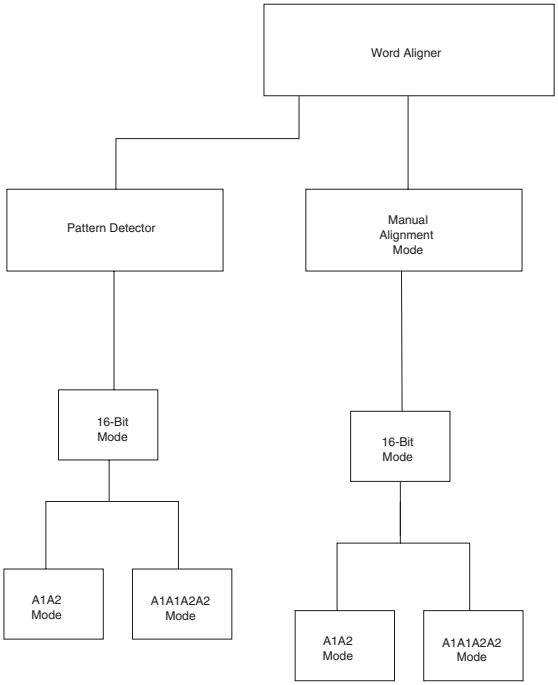
In the bit-slip mode, the byte boundary can be modified by a barrel shifter to slip the byte boundary one bit at a time via a user-controlled bit-slip port. The bit-slip mode supports both 8-bit and 10-bit data paths operating in a single or double-width mode.

The pattern detector is active in the bit-slip mode, and it detects the user-defined pattern that is specified in the MegaWizard® Plug-In Manager.

The bit-slip mode is available only in Custom mode and SONET mode.

Figure 2–16 shows the word aligner in 16-bit mode.

Figure 2–16. Word Aligner in 16-Bit Mode

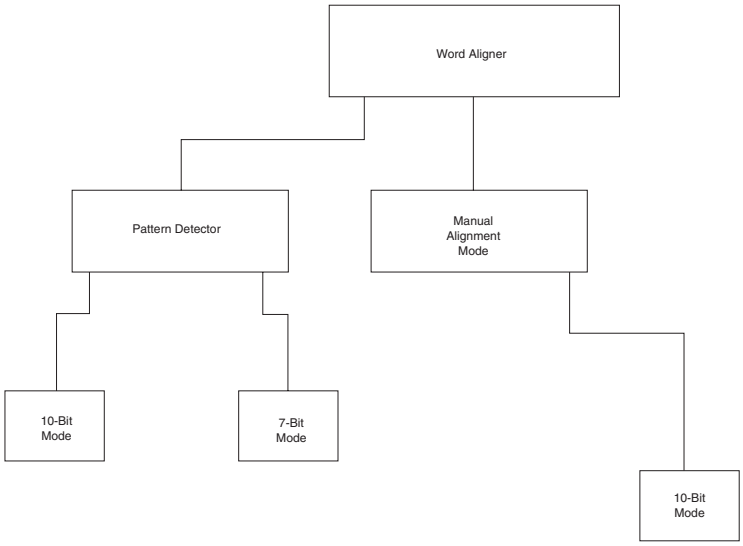


In the 16-bit mode, the word aligner and pattern detector automatically aligns and detects a user-defined 16-bit alignment pattern. This pattern can be in the format of A1A2 or A1A1A2A2 (for the SONET protocol). The re-alignment of the byte boundary can be done via a user-controlled port. The 16-bit mode supports only the 8-bit data path in a single-width or double-width mode.

The 16-bit mode is available only for the Custom mode and SONET mode. The A1A1A2A2 word alignment pattern option is available only for the SONET mode and cannot be used in the Custom mode.

Figure 2–17 shows the word aligner in 10-bit mode.

Figure 2–17. Word Aligner in 10-Bit Mode

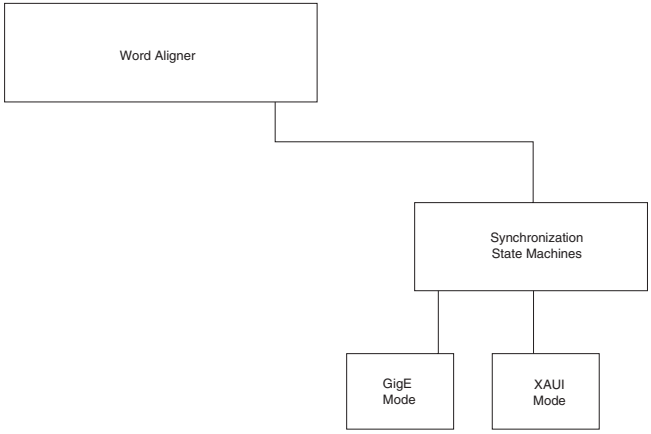


In the 10-bit mode, the word aligner automatically aligns the user's predefined 10-bit alignment pattern. The pattern detector can detect the full 10-bit pattern or only the lower seven bits of the pattern. The word aligner and pattern detector detect both the positive and the negative disparity of the pattern. A user-controlled enable port is available for the word aligner.

The 10-bit mode is available only for the Custom mode.

Figure 2–18 shows the word aligner in XAUI mode.

Figure 2–18. Word Aligner in XAUI Mode



In the XAUI and GIGE modes, the word alignment is controlled by a state machine that adheres to the IEEE 802.3ae standard for XAUI and the IEEE 802.3 standard for GIGE. The alignment pattern is predefined to be a /K28.5/ code group.

The XAUI mode is available only for the XAUI protocol, and the GIGE mode is available only for the GIGE protocol.

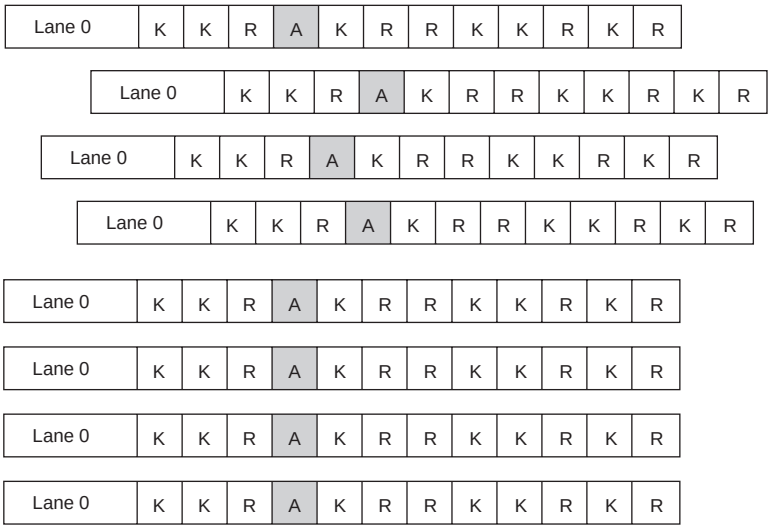
Channel Aligner

The channel aligner is available only in XAUI mode and bonds all four channels within a transceiver. The channel aligner adheres to the IEEE 802.3ae, clause 48 specification for channel bonding.

The channel aligner is a 16-word deep FIFO buffer with a state machine overlooking the channel bonding process. The state machine looks for an /A/ (/K28.3/) in each channel and aligns all the /A/s in the transceiver. When four columns of /A/ (denoted by //A//) are detected, the rx_channelalign port goes high, signifying that all the channels in the transceiver have been bonded. The reception of four consecutive misaligned /A/s restarts the channel alignment sequence and de-asserts rx_channelalign.

Figure 2–19 shows misaligned channels before the channel aligner and the channel alignment after the channel aligner.

Figure 2–19. Before & After the Channel Aligner



Rate Matcher

The rate matcher, which is available only in XAUI and GIGE modes, consists of a 12-word deep FIFO buffer and a FIFO controller. The rate matcher is bypassed when the device is not in XAUI or GIGE mode.

In a multi-crystal environment, the rate matcher compensates for up to a 100-ppm difference between the source and receiver clocks.

GIGE Mode

In the GIGE mode, the rate matcher adheres to the specifications in clause 36 of the IEEE 802.3 documentation, for idle additions or removals. The rate matcher performs clock compensation only on `/I2/` ordered sets, composing a `/K28.5/+` followed by a `/D16.2/-`. The rate matcher does not perform a clock compensation on any other ordered set combinations. An /I2/ is added or deleted automatically based on the number of words in the FIFO buffer. A `9'h19C` is given at the control and data ports when the FIFO is in an overflow or underflow condition.

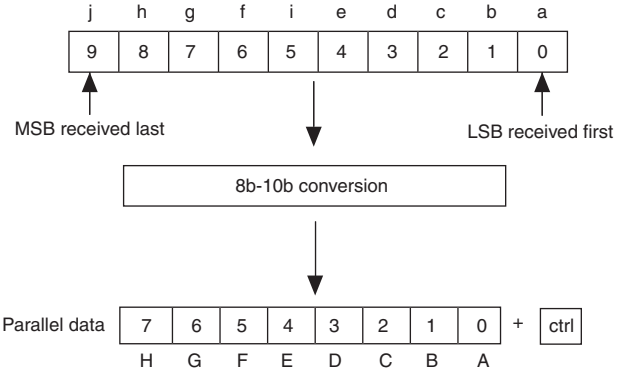
XAUI Mode

In XAUI mode, the rate matcher adheres to clause 48 of the IEEE 802.3ae specification for clock rate compensation. The rate matcher performs clock compensation on columns of /R/ (/K28.0/), denoted by //R//. An //R// is added or deleted automatically based on the number of words in the FIFO buffer.

8B/10B Decoder

The 8B/10B decoder converts the 10-bit encoded code group into 8-bit data and 1 control bit. The 8B/10B decoder can be bypassed. The following is a diagram of the conversion from a 10-bit encoded code group into 8-bit data + 1-bit control.

Figure 2–20. 8B/10B Decoder Conversion



There are two optional error status ports available in the 8B/10B decoder, rx_errdetect and rx_disperr. Table 2–7 shows the values of the ports from a given error. These status signals are aligned with the code group in which the error occurred.

Table 2–7. Error Signal Values

Types of Errors	rx_errdetect	rx_disperr
No errors	1'b0	1'b0
Invalid code groups	1'b1	1'b0
Disparity errors	1'b1	1'b1

Receiver State Machine

The receiver state machine operates in GIGE and XAUI modes. In GIGE mode, the receiver state machine replaces invalid code groups with `9'h1FE`. In XAUI mode, the receiver state machine translates the XAUI PCS code group to the XAUI XGMII code group. Table 2–8 shows the code conversion. The conversion adheres to the IEEE 802.3ae specification.

Table 2–8. Code Conversion

XGMII RXC	XGMII RXD	PCS code-group	Description
0	00 through FF	Dxx.y	Normal Data
1	07	K28.0 or K28.3 or K28.5	Idle in ||**I**||
1	07	K28.5	Idle in ||**T**||
1	9C	K28.4	Sequence
1	FB	K27.7	Start
1	FD	K29.7	Terminate
1	FE	K30.7	Error
1	FE	Invalid code group	Invalid XGMII character
1	See IEEE 802.3 reserved code groups	See IEEE 802.3 reserved code groups	Reserved code groups

Byte Deserializer

The byte deserializer takes a single width word (8 or 10 bits) from the transceiver logic and converts it into double-width words (16 or 20 bits) to the phase compensation FIFO buffer. The byte deserializer is bypassed when single width mode (8 or 10 bits) is used at the PLD interface.

Phase Compensation FIFO Buffer

The receiver phase compensation FIFO buffer resides in the transceiver block at the programmable logic device (PLD) boundary. This buffer compensates for the phase difference between the recovered clock within the transceiver and the recovered clock after it has transferred to the PLD core. The phase compensation FIFO buffer is four words deep and cannot be bypassed.

Loopback Modes

The Stratix GX transceiver has built-in loopback modes to aid in debug and testing. The loopback modes are set in the Stratix GX MegaWizard Plug-In Manager in the Quartus II software. Only one loopback mode can be set at any single instance of the transceiver block. The loopback mode applies to all used channels in a transceiver block.

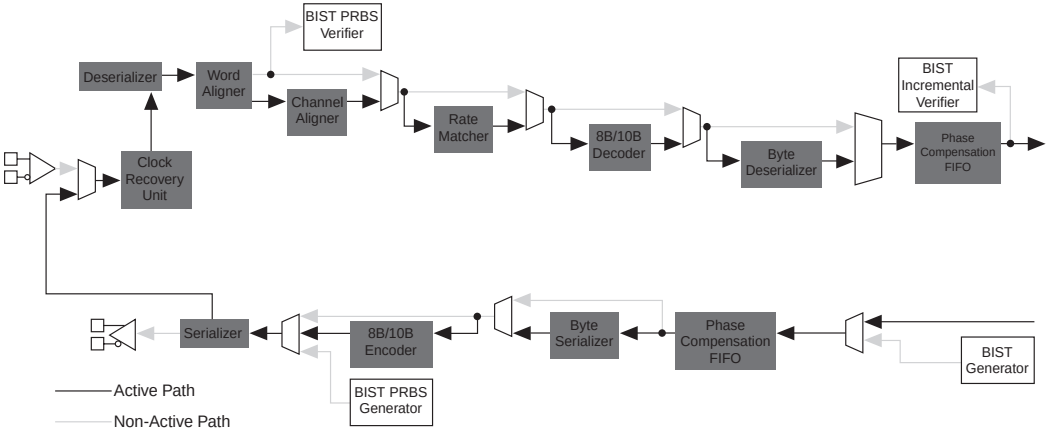
The available loopback modes are:

- Serial loopback
- Parallel loopback
- Reverse serial loopback

Serial Loopback

Serial loopback exercises all the transceiver logic except for the output buffer and input buffer. The loopback function is dynamically switchable through the `rx_slpbk` port on a channel by channel basis. The V_{OD} of the output reduced. If you select 400 mV, the output is tri-stated when the serial loopback option is selected. Figure 2–21 shows the data path in serial loopback mode.

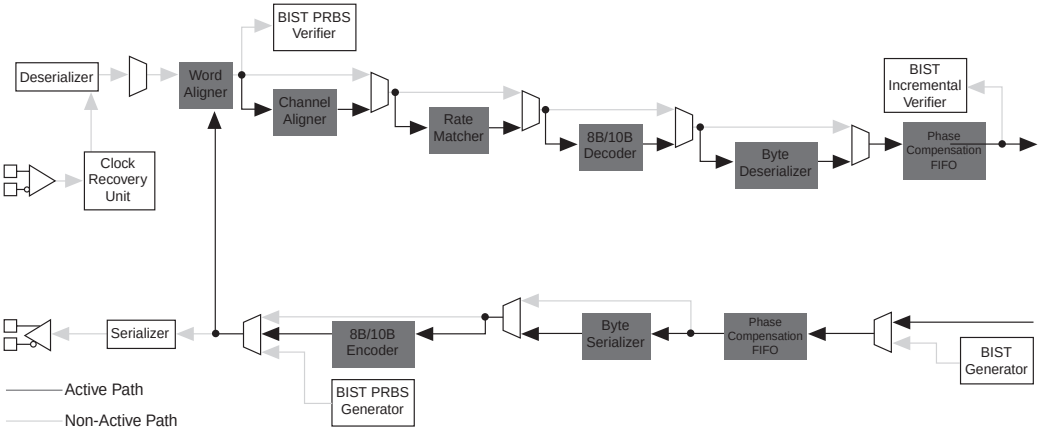
Figure 2–21. Data Path in Serial Loopback Mode



Parallel Loopback

The parallel loopback mode exercises the digital logic portion of the transceiver data path. The analog portions are not use in the loopback path. The received data is not retimed. Figure 2–22 shows the data path in parallel loopback mode. This option is not dynamically switchable. Reception of an external signal is not possible in this mode.

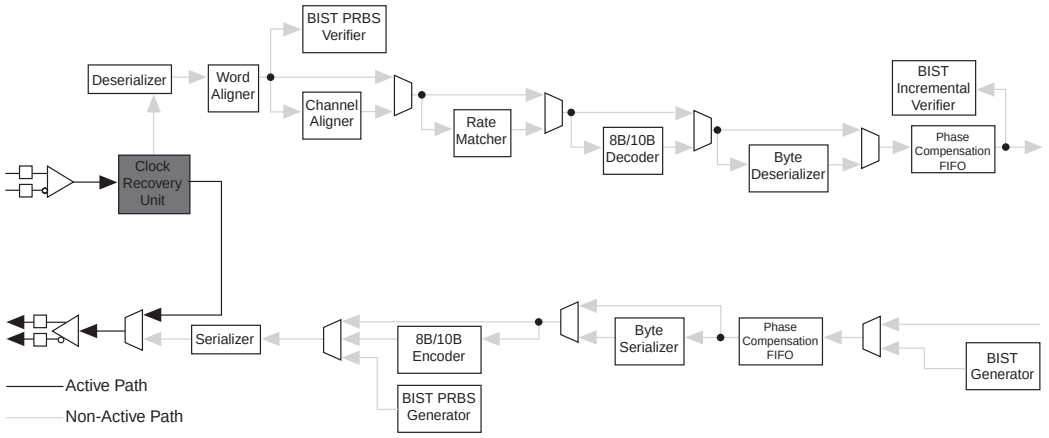
Figure 2–22. Data Path in Parallel Loopback Mode



Reverse Serial Loopback

The reverse serial loopback exercises the analog portion of the transceiver. This loopback mode is dynamically switchable through the `tx_srlpbk` port on a channel by channel basis. Asserting `rxanalogreset` in reverse serial loopback mode powers down the receiver buffer and CRU, preventing data loopback. Figure 2–23 shows the data path in reverse serial loopback mode.

Figure 2–23. Data Path in Reverse Serial Loopback Mode



BIST (Built-In Self Test)

The Stratix GX transceiver has built-in self test modes to aid in debug and testing. The BIST modes are set in the Stratix GX MegaWizard Plug-In Manager in the Quartus II software. Only one BIST mode can be set for any single instance of the transceiver block. The BIST mode applies to all channels used in a transceiver.

The following is a list of the available BIST modes:

- PRBS generator and verifier
- Incremental mode generator and verifier
- High-frequency generator
- Low-frequency generator
- Mixed-frequency generator

Figures 2–24 and 2–25 are diagrams of the BIST PRBS data path and the BIST incremental data path, respectively.

Figure 2–24. BIST PRBS Data Path

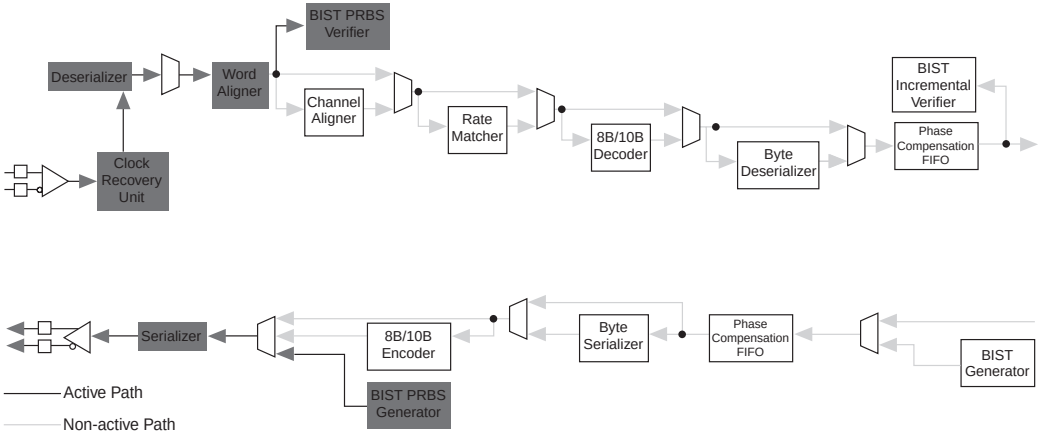


Figure 2–25. BIST Incremental Data Path

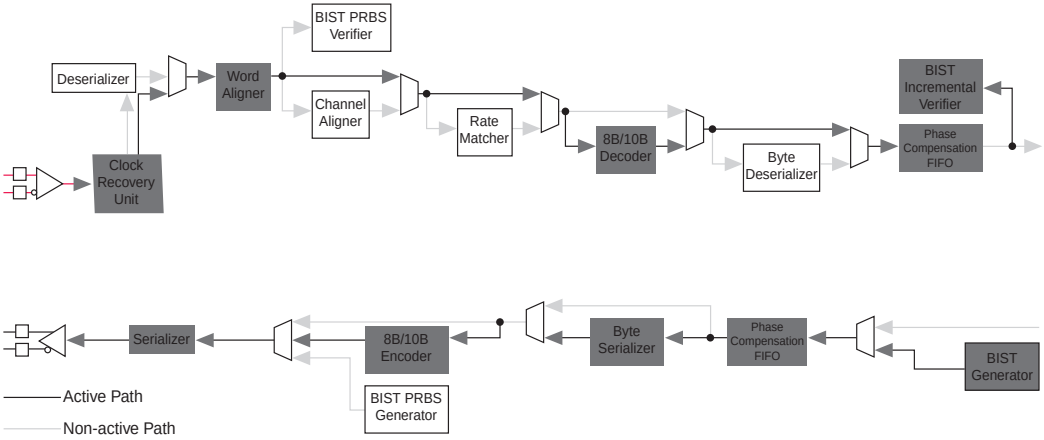


Table 2–9 shows the BIST data output and verifier alignment pattern.

Table 2–9. BIST Data Output & Verifier Alignment Pattern (Part 1 of 2)

BIST Mode	Output	Polynomials	Verifier Word Alignment Pattern
PRBS 8-bit	$2^8 - 1$	$x^8 + x^7 + x^5 + x^3 + 1$	1000000011111111
PRBS 10-bit	$2^{10} - 1$	$x^{10} + x^7 + 1$	1111111111

Table 2–9. BIST Data Output & Verifier Alignment Pattern (Part 2 of 2)

BIST Mode	Output	Polynomials	Verifier Word Alignment Pattern
PRBS 16-bit	$2^8 - 1$	$x^8 + x^7 + x^5 + x^3 + 1$	1000000011111111
PRBS 20-bit	$2^{10} - 1$	$x^{10} + x^7 + 1$	1111111111
Incremental 10-bit	K28.5, K27.7, Data (00-FF incremental), K28.0, K28.1, K28.2, K28.3, K28.4, K28.6, K28.7, K23.7, K30.7, K29.7 *(1)*		0101111100 (K28.5)
Incremental 20-bit	K28.5, K27.7, Data (00-FF incremental), K28.0, K28.1, K28.2, K28.3, K28.4, K28.6, K28.7, K23.7, K30.7, K29.7 *(1)*		0101111100 (K28.5)
High frequency	1010101010		
Low frequency	0011111000		
Mixed frequency	0011111010 or 1100000101		

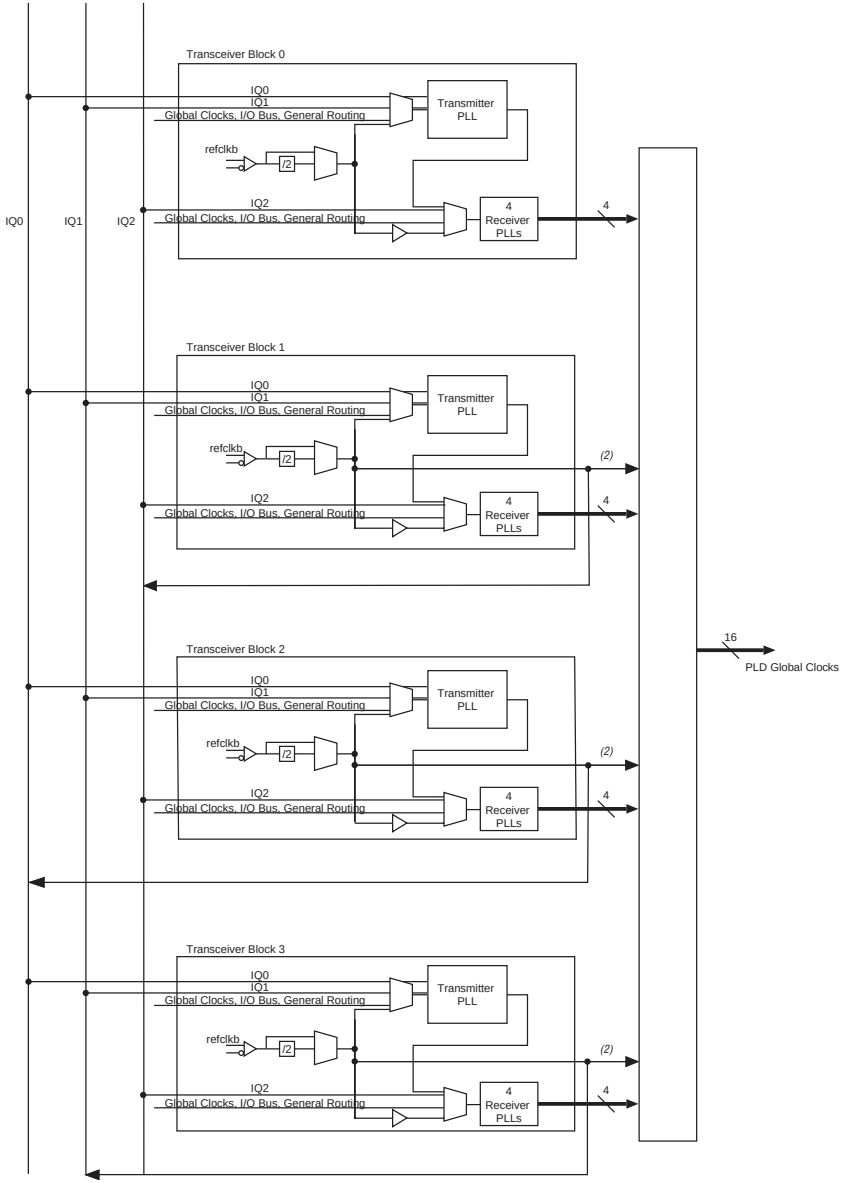
Note to Table 2–9:
(1) This output repeats.

Stratix GX Clocking

The Stratix GX global clock can be driven by certain REFCLKB pins, all transmitter PLL outputs, and all receiver PLL outputs. The REFCLKB pins (except for transceiver block 0 and transceiver block 4) can drive inter-transceiver and global clock lines as well as feed the transmitter and receiver PLLs. The output of the transmitter PLL can only feed global clock lines and the reference clock port of the receiver PLL.

Figures 2–26 and 2–27 are diagrams of the Inter-Transceiver line connections as well as the global clock connections for the EP1SGX25F and EP1SGX40G devices. For devices with fewer transceivers, ignore the information about the unavailable transceiver blocks.

Figure 2–26. EP1SGX25F Device Inter-Transceiver & Global Clock Connections *Note (1)*



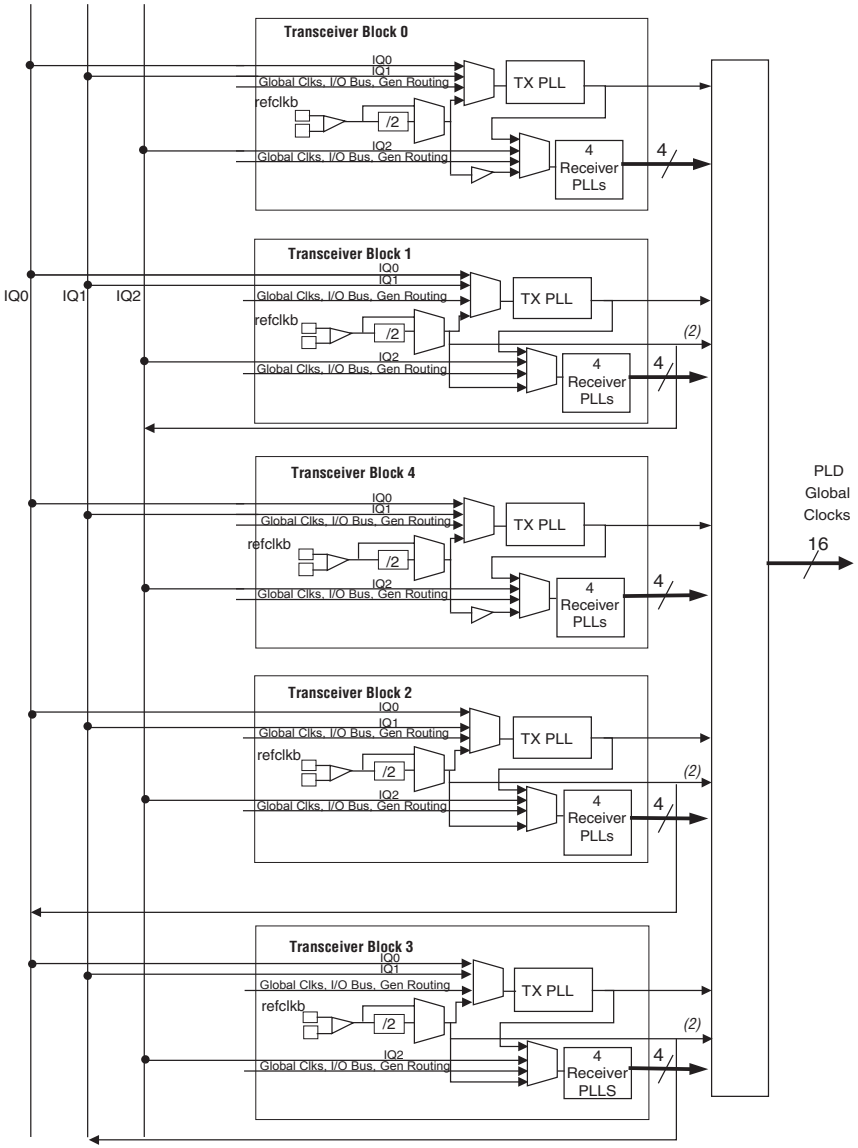
Notes to Figure 2–26:

(1) IQ lines are inter-transceiver block lines.

(2) If the /2 pre-divider is used, the path to drive the PLD logic array, local, or global clocks is not allowed.

(3) There are four receiver PLLs in each transceiver block.

Figure 2–27. EP1SGX40G Device Inter-Transceiver & Global Clock Connections *Note (1)*



Notes to Figure 2–27:

(1) IQ lines are inter-transceiver block lines.

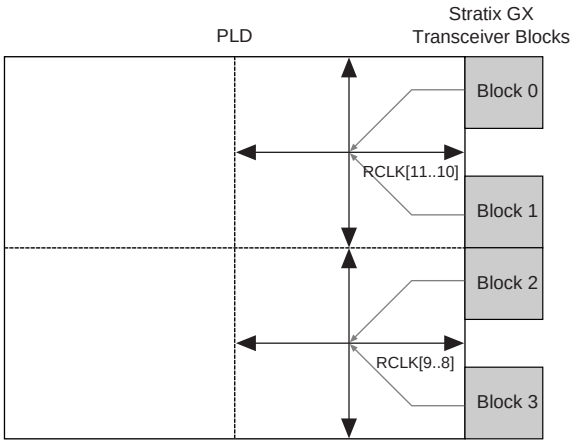
(2) If the /2 pre-divider is used, the path to drive the PLD logic array, local, or global clocks is not allowed.

(3) There are four receiver PLLs in each transceiver block.

The receiver PLL can also drive the fast regional, regional clocks, and local routing adjacent to the associated transceiver block. Figures 2–28 through 2–31 show which fast regional and regional clock resource can be used by the recovered clock.

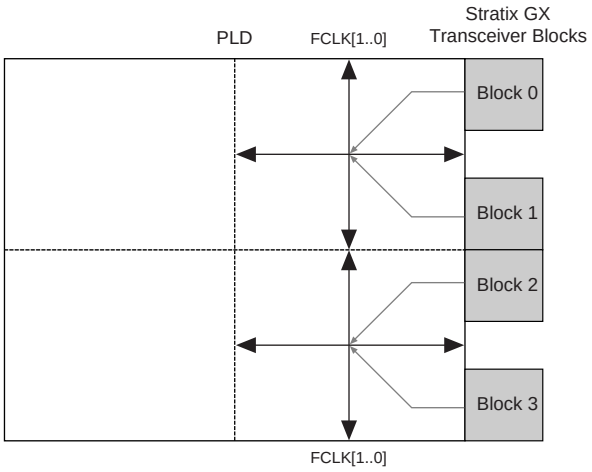
In the EP1SGX25 device, the receiver PLL recovered clocks from transceiver blocks 0 and 1 drive `RCLK[1..0]` while transceiver blocks 2 and 3 drive `RCLK[7..6]`. The regional clocks feed logic in their associated regions.

Figure 2–28. EP1SGX25 Receiver PLL Recovered Clock to Regional Clock Connection



In addition, the receiver PLL's recovered clocks can drive fast regional lines (`FCLK`) as shown Figure 2–29. The fast regional clocks can feed logic in their associated regions.

Figure 2–29. EP1SGX25 Receiver PLL Recovered Clock to Fast Regional Clock Connection



In the EP1SGX40 device, the receiver PLL recovered clocks from transceivers 0 and 1 drive `RCLK[1..0]` while transceivers 2, 3, and 4 drive `RCLK[7..6]`. The regional clocks feed logic in their associated regions.

Figure 2–30. EP1SGX40 Receiver PLL Recovered Clock to Regional Clock Connection

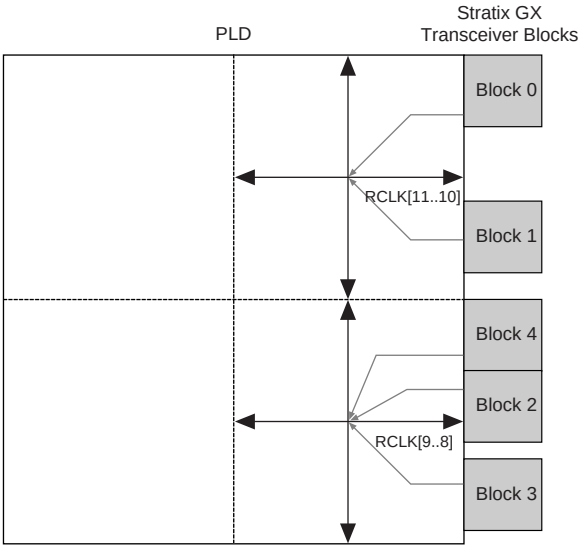


Figure 2–31 shows the possible recovered clock connection to the fast regional clock resource. The fast regional clocks can drive logic in their associated regions.

Figure 2–31. EP1SGX40 Receiver PLL Recovered Clock to Fast Regional Clock Connection

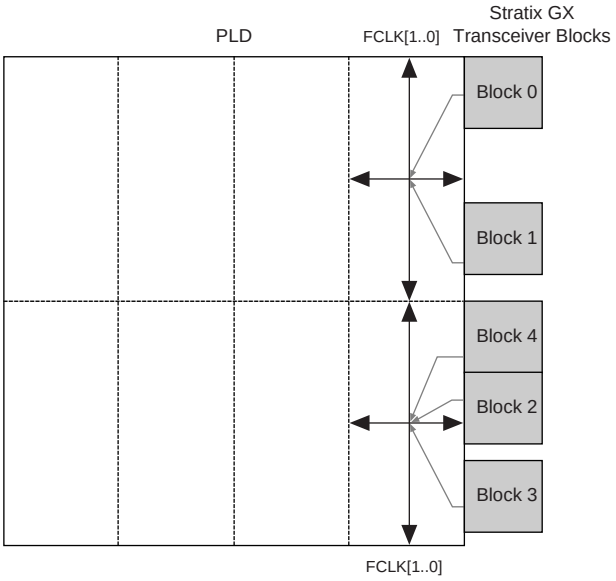


Table 2–10 summarizes the possible clocking connections for the transceivers.

Table 2–10. Possible Clocking Connections for Transceivers (Part 1 of 2)

Source	Destination					
	Transmitter PLL	Receiver PLL	GCLK	RCLK	FCLK	IQ Lines
REFCLKB	✓	✓	✓ *(1)*	✓		✓ *(1)*
Transmitter PLL		✓	✓	✓	✓	
Receiver PLL			✓	✓	✓	
GCLK	✓	✓				
RCLK	✓	✓				
FCLK	✓	✓				

Table 2–10. Possible Clocking Connections for Transceivers (Part 2 of 2)

Source	Destination					
	Transmitter PLL	Receiver PLL	GCLK	RCLK	FCLK	IQ Lines
IQ lines	✓ (2)	✓ (2)				

Notes to Table 2–10:

(1) REFCLKB from transceiver block 0 and transceiver block 4 does not drive the inter-transceiver lines or the GCLK lines.

(2) Inter-transceiver line 0 and inter-transceiver line 1 drive the transmitter PLL, while inter-transceiver line 2 drives the receiver PLLs.

Other Transceiver Features

Other important features of the Stratix GX transceivers are the power down and reset capabilities, the external voltage reference and bias circuitry, and hot swapping.

Individual Power-Down & Reset for the Transmitter & Receiver

Stratix GX transceivers offer a power saving advantage with their ability to shut off functions that are not needed. The device can individually reset the receiver and transmitter blocks and the PLLs. The Stratix GX device can either globally power down and reset the transmitter and receiver channels or do each channel separately. Table 2–11 shows the connectivity between the reset signals and the Stratix GX logical blocks.

Power-down functions are static, in other words., they are implemented upon device configuration and programmed, through the Quartus II software, to static values. Resets can be static as well as dynamic inputs coming from the logic array or pins.

Table 2–11. Reset Signal Map to Stratix GX Blocks

Reset Signal	Transmitter Phase Compensation FIFO Module/ Byte Serializer	Transmitter 8B/10B Encoder	Transmitter Serializer	Transmitter Analog Circuits	Transmitter PLL	Transmitter XAUI State Machine	Transmitter Analog Circuits	BIST Generators	Receiver Deserializer	Receiver Word Aligner	Receiver Deskew FIFO Module	Receiver Rate Matcher	Receiver 8B/10B Decoder	Receiver Phase Comp FIFO Module/ Byte Deserializer	Receiver PLL / CRU	Receiver XAUI State Machine	BIST Verifiers	Receiver Analog Circuits
rxdigitalreset										✓	✓	✓	✓	✓		✓	✓	
rxanalogreset									✓						✓			✓
txdigitalreset	✓	✓				✓		✓										
pll_areset	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
pllenable	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Voltage Reference Capabilities

Stratix GX transceivers provide voltage reference and bias circuitry. To set-up internal bias for controlling the transmitter output drivers' voltage swing—as well as to provide voltage/current biasing for other analog circuitry—use the internal bandgap voltage reference at 0.7 V. To provide bias for internal pull-up PMOS resistors for I/O termination at the serial interface of receiver and transmitter channels (independent of power supply drift, process changes, or temperature variation) an external resistor, which is connected to the external low voltage power supply, is

accurately tracked by the internal bias circuit. Moreover, the reference voltage and internal resistor bias current is generated and replicated to the analog circuitry in each channel.

Hot-Socketing Capabilities

Each Stratix GX device is capable of hot-socketing. Because Stratix GX devices can be used in a mixed-voltage environment, they have been designed specifically to tolerate any possible power-up sequence. Signals can be driven into Stratix GX devices before and during power-up without damaging the device. Once operating conditions are reached and the device is configured, Stratix GX devices operate according to your specifications. This feature provides the Stratix GX transceiver line card behavior, so you can insert it into the system without powering the system down, offering more flexibility.

Applications & Protocols Supported with Stratix GX Devices

Each Stratix GX transceiver block is designed to operate at any serial bit rate from 500 Mbps to 3.1875 Gbps per channel. The wide, data rate range allows Stratix GX transceivers to support a wide variety of standard and future protocols such as 10-Gigabit Ethernet XAUI, InfiniBand, Fibre Channel, and Serial RapidIO. Stratix GX devices are ideal for many high-speed communication applications such as high-speed backplanes, chip-to-chip bridges, and high-speed serial communications standards support.

Stratix GX Example Application Support

Stratix GX devices can be used for many applications, including:

- Backplanes for traffic management and quality of service (QOS)
- Switch fabric applications for complete set for backplane and switch fabric transceivers
- Chip-to-chip applications such as: 10 Gigabit Ethernet XAUI to XGMII bridge, 10 Gigabit Ethernet XGMII to POS-PHY4 bridge, POS-PHY4 to NPSI bridge, or NPSI to backplane bridge

High-Speed Serial Bus Protocols

With wide, serial data rate range, Stratix GX devices can support multiple, high-speed serial bus protocols. Table 2–12 shows some of the protocols that Stratix GX devices can support.

Table 2–12. High-Speed Serial Bus Protocols

Bus Transfer Protocol	Stratix GX (Gbps) (Supports up to 3.1875 Gbps)
SONET backplane	2.488
10 Gigabit Ethernet XAUI	3.125
10 Gigabit fibre channel	3.1875
InfiniBand	2.5
Fibre channel (1G, 2G)	1.0625, 2.125
Serial RapidIO™	1.25, 2.5, 3.125
PCI Express	2.5
SMPTE 292M	1.485

3. Source-Synchronous Signaling With DPA

SGX51003-1.2

Introduction

Expansion in the telecommunications market and growth in Internet use requires systems to move more data faster than ever. To meet this demand, rely on solutions such as differential signaling and emerging high-speed interface standards including RapidIO, POS-PHY 4, SFI-4, or XSBI.

These new protocols support differential data rates up to 1 Gbps and higher. At these high data rates, it becomes more challenging to manage the skew between the clock and data signals. One solution to this challenge is to use CDR to eliminate skew between data channels and clock signals. Another potential solution, DPA, is beginning to be incorporated into some of these protocols.

The source-synchronous high-speed interface in Stratix GX devices is a dedicated circuit embedded into the PLD allowing for high-speed communications. The *High-Speed Source-Synchronous Differential I/O Interfaces in Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2* provides information on the high-speed I/O standard features and functions of the Stratix GX device.

Stratix GX I/O Banks

Stratix GX devices contain 17 I/O banks. I/O banks one and two support high-speed LVDS, LVPECL, and 3.3-V PCML inputs and outputs. These two banks also incorporate an embedded dynamic phase aligner within the source-synchronous interface (see Figure 3–8 on page 3–10). The dynamic phase aligner corrects for the phase difference between the clock and data lines caused by skew. The dynamic phase aligner operates automatically and continuously without requiring a fixed training pattern, and allows the source-synchronous circuitry to capture data correctly regardless of the channel-to-clock skew.

Principles of SERDES Operation

Stratix GX devices support source-synchronous differential signaling up to 1 Gbps in DPA mode, and up to 840 Mbps in non-DPA mode. Serial data is transmitted and received along with a low-frequency clock. The PLL can multiply the incoming low-frequency clock by a factor of 1 to 10. The SERDES factor *J* can be 8 or 10 for the DPA mode, or 4, 7, 8, or 10 for all other modes. The SERDES factor does not have to equal the clock

multiplication value. The ×1 and ×2 operation is also possible by bypassing the SERDES. The SERDES DPA cannot support ×1, ×2, or ×4 natively.

On the receiver side, the high-frequency clock generated by the PLL shifts the serial data through a shift register (also called deserializer). The parallel data is clocked out to the logic array synchronized with the low-frequency clock. On the transmitter side, the parallel data from the logic array is first clocked into a parallel-in, serial-out shift register synchronized with the low-frequency clock and then transmitted out by the output buffers.

There are two dedicated fast PLLs each in EP1SGX10 to EP1SGX25 devices, and four in EP1SGX40 devices. These PLLs are used for the SERDES operations as well as general-purpose use.

Stratix GX Differential I/O Receiver Operation (Non-DPA Mode)

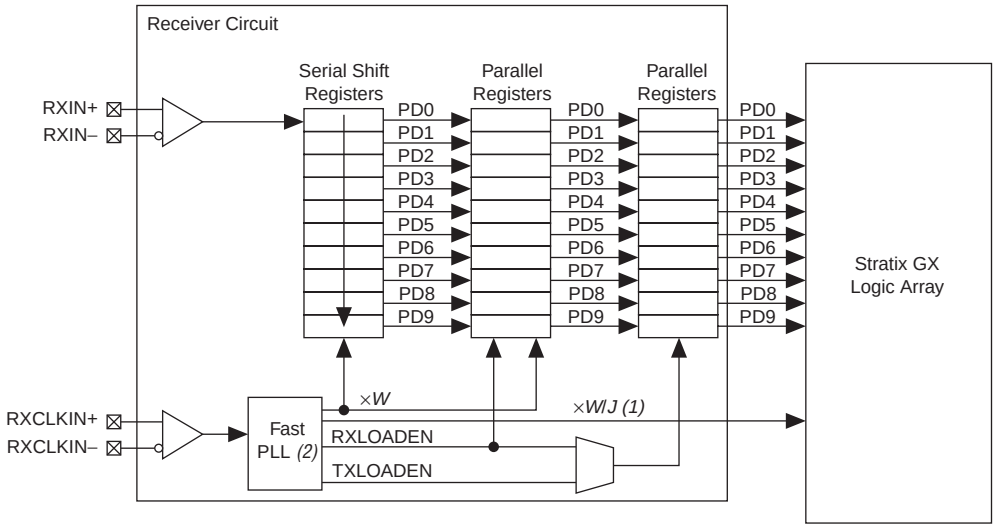
You can configure any of the Stratix GX source synchronous differential input channels as a receiver channel (see Figure 3–1). The differential receiver deserializes the incoming high-speed data. The input shift register continuously clocks the incoming data on the negative transition of the high-frequency clock generated by the PLL clock (×*W*).

The data in the serial shift register is shifted into a parallel register by the RXLOADEN signal generated by the fast PLL counter circuitry on the third falling edge of the high-frequency clock. However, you can select which falling edge of the high frequency clock loads the data into the parallel register, using the data-realignment circuit.

In normal mode, the enable signal RXLOADEN loads the parallel data into the next parallel register on the second rising edge of the low-frequency clock. You can also load data to the parallel register through the TXLOADEN signal when using the data-realignment circuit.

Figure 3–1 shows the block diagram of a single SERDES receiver channel. Figure 3–2 shows the timing relationship between the data and clocks in Stratix GX devices in ×10 mode. *W* is the low-frequency multiplier and *J* is the data parallelization division factor.

Figure 3–1. Stratix GX High-Speed Interface Deserialized in ×10 Mode

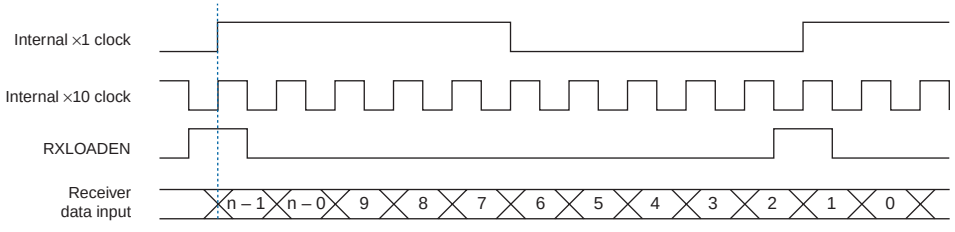


Notes to Figure 3–1:

(1) W = 1, 2, 4, 7, 8, or 10.
J = 4, 7, 8, or 10 for non-DPA (J = 8 or 10 for DPA).
W does not have to equal J. When J = 1 or 2, the deserializer is bypassed. When J = 2, the device uses DDRIO registers.

(2) This figure does not show additional circuitry for clock or data manipulation.

Figure 3–2. Receiver Timing Diagram



Stratix GX Differential I/O Transmitter Operation

You can configure any of the Stratix GX differential output channels as a transmitter channel. The differential transmitter serializes outbound parallel data.

The logic array sends parallel data to the SERDES transmitter circuit when the TXLOADEN signal is asserted. This signal is generated by the high-speed counter circuitry of the logic array low-frequency clock's rising edge. The data is then transferred from the parallel register into the serial shift register by the TXLOADEN signal on the third rising edge of the high-frequency clock.

Figure 3–3 shows the block diagram of a single SERDES transmitter channel and Figure 3–4 shows the timing relationship between the data and clocks in Stratix GX devices in ×10 mode. *W* is the low-frequency multiplier and *J* is the data parallelization division factor.

Figure 3–3. Stratix GX High-Speed Interface Serialized in ×10 Mode

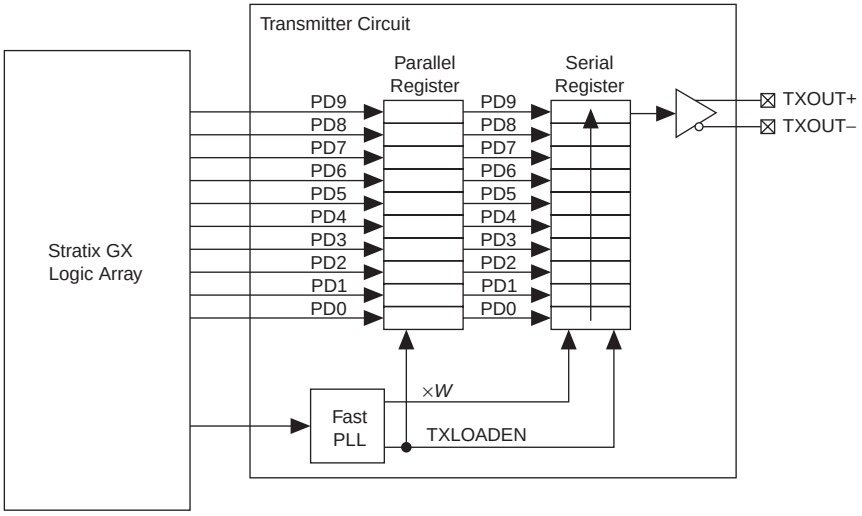
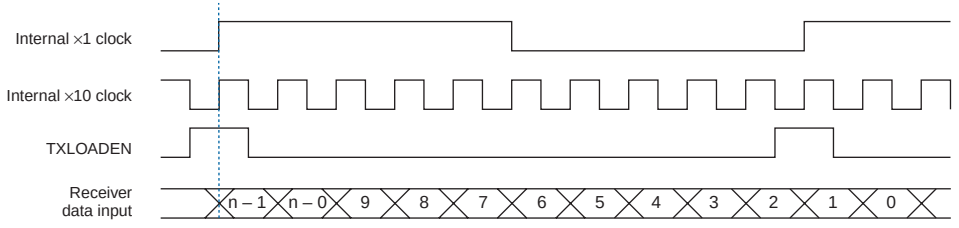


Figure 3–4. Transmitter Timing Diagram

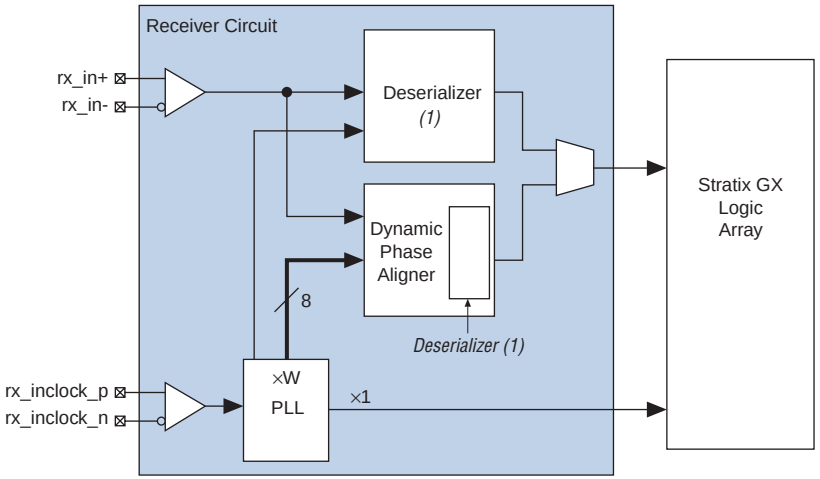


DPA Block Overview

Each Stratix GX receiver channel features a DPA block. The block contains a dynamic phase selector for phase detection and selection, a SERDES, a synchronizer, and a data realigner circuit. You can bypass the dynamic phase aligner without affecting the basic source-synchronous operation of the channel by using a separate deserializer shown in Figure 3–5.

The dynamic phase aligner uses both the source clock and the serial data. The dynamic phase aligner automatically and continuously tracks fluctuations caused by system variations and self-adjusts to eliminate the phase skew between the multiplied clock and the serial data. Figure 3–5 shows the relationship between Stratix GX source-synchronous circuitry and the Stratix GX source-synchronous circuitry with DPA.

Figure 3–5. Source-Synchronous DPA Circuitry



Note to Figure 3–5:

(1) Both deserializers are identical. The deserializer operation is described in the "Principles of SERDES Operation" section.

Unlike the de-skew function in APEX™ 20KE and APEX 20KC devices, you do not have to use a fixed training pattern with DPA in Stratix GX devices. Table 3–1 shows the differences between source-synchronous circuitry with DPA and source-synchronous circuitry without DPA circuitry in Stratix GX devices.

Table 3–1. Source-Synchronous Circuitry With & Without DPA

Feature	Source-Synchronous Circuitry	
	Without DPA	With DPA
Data rate	300 to 840 Megabits per second (Mbps)	300 Mbps to 1 Gbps
Deserialization factors	1, 2, 4, 8, 10	8, 10
Clock frequency	10 to 717 MHz	74 to 717 MHz
Interface pins	I/O banks 1 and 2	I/O banks 1 and 2
Receiver pins	Dedicated inputs	Dedicated inputs

DPA Input Support

Stratix GX device I/O banks 1 and 2 contain dedicated circuitry to support differential I/O standards at speeds up to 1 Gbps with DPA (or up to 840 Mbps without DPA). Stratix GX device source-synchronous circuitry supports LVDS, LVPECL, and 3.3-V PCML I/O standards, each with a supply voltage of 3.3 V. Refer to the *High-Speed Source-Synchronous Differential I/O Interfaces in Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2* for more information on these I/O standards. Transmitter pins can be either input or output pins for single-ended I/O standards. Refer to Table 3–2.

Table 3–2. Bank 1 & 2 Input Pins

Input Pin Type	I/O Standard	Receiver Pin	Transmitter Pin
Differential	Differential	Input only	Output only
Single ended	Single ended	Input only	Input or output

Interface & Fast PLL

This section describes the number of channels that support DPA and their relationship with the PLL in Stratix GX devices. EP1SGX10 and EP1SGX25 devices have two dedicated fast PLLs and EP1SGX40 devices

have four dedicated fast PLLs for clock multiplication. Table 3–3 shows the maximum number of channels in each Stratix GX device that support DPA.

Table 3–3. Stratix GX Source-Synchronous Differential I/O Resources

Device	Fast PLLs	Pin Count	Receiver Channels *(1)*	Transmitter Channels *(1)*	Receiver & Transmitter Channel Speed (Gbps) *(2)*	LEs
EP1SGX10C	2 *(3)*	672	22	22	1	10,570
EP1SGX10D	2 *(3)*	672	22	22	1	10,570
EP1SGX25C	2	672	39	39	1	25,660
EP1SGX25D	2	672	39	39	1	25,660
		1,020	39	39	1	25,660
EP1SGX25F	2	1,020	39	39	1	25,660
EP1SGX40D	4 *(4)*	1,020	45	45	1	41,250
EP1SGX40G	4 *(4)*	1,020	45	45	1	41,250

Notes to Table 3–3:

(1) This is the number of receiver or transmitter channels in the source-synchronous (I/O bank 1 and 2) interface of the device.

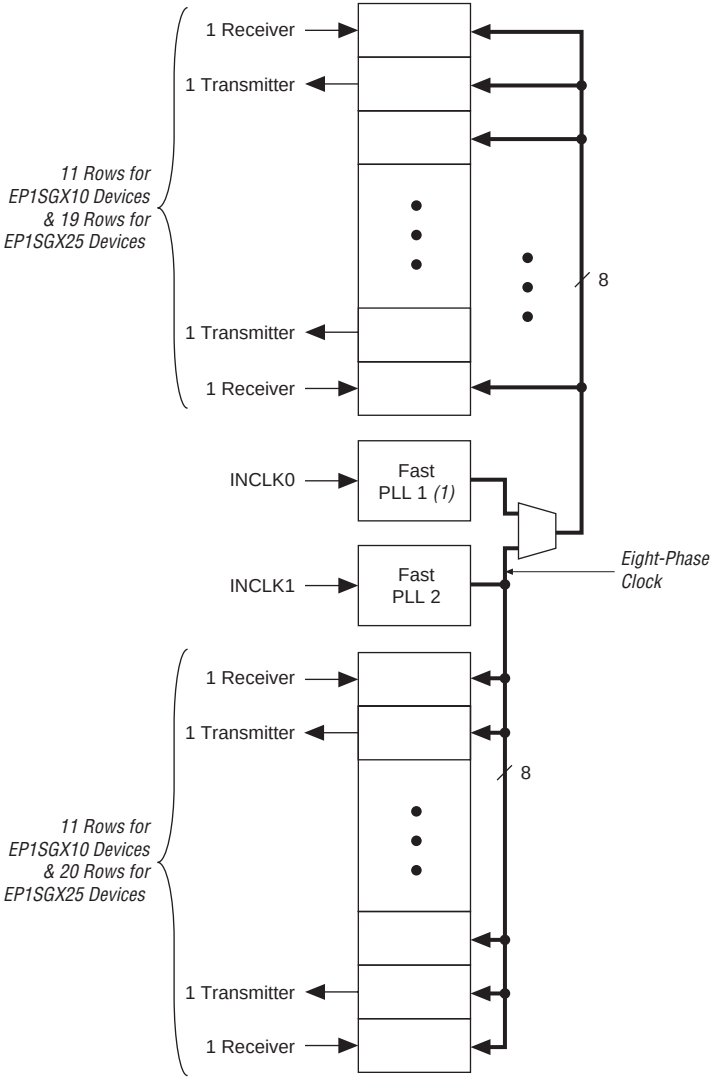
(2) Receiver channels operate at 1,000 Mbps with DPA. Without DPA, the receiver channels operate at 840 Mbps.

(3) One of the two fast PLLs in EP1SGX10C and EP1SGX10D devices supports DPA.

(4) Two of the four fast PLLs in EP1SGX40D and EP1SGX40G devices support DPA

The receiver and transmitter channels are interleaved so that each I/O row in I/O banks 1 and 2 of the device has one receiver channel and one transmitter channel per row. Figures 3–6 and 3–7 show the fast PLL and channels with DPA layout in EP1SGX10, EP1SGX25, and EP1SGX40 devices. In EP1SGX10 devices, only fast PLL 2 supports DPA operations.

Figure 3–6. PLL & Channel Layout in EP1SGX10 & EP1SGX25 Devices *Notes (1), (2)*

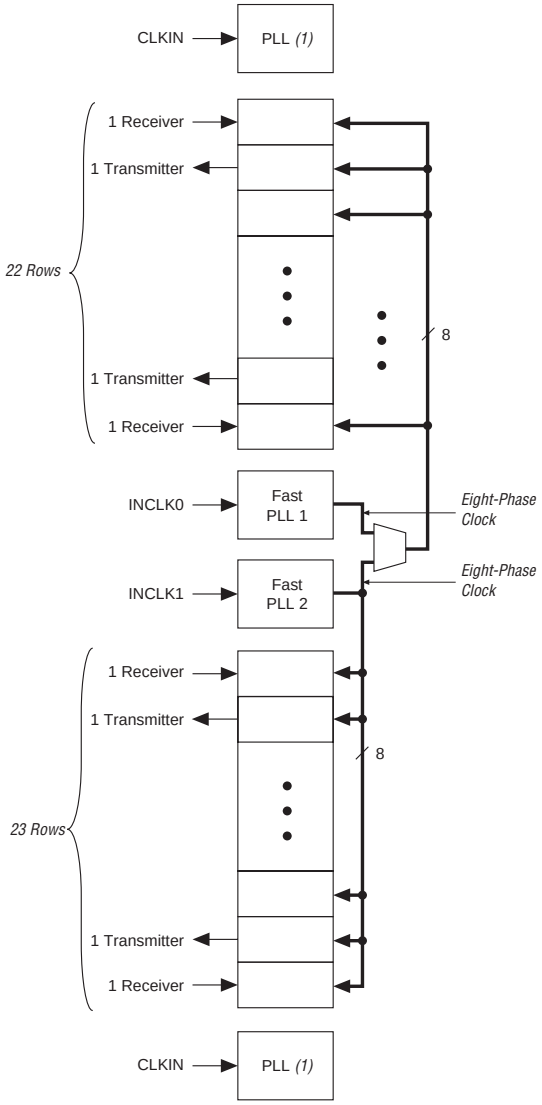


Notes to Figure 3–6:

(1) Fast PLL 1 in EP1SGX10 devices does not support DPA.

(2) Not all eight phases are used by the receiver channel or transmitter channel in non-DPA mode.

Figure 3–7. PLL & Channel Layout in EP1SGX40 Devices *Notes (1), (2), (3)*



Notes to Figure 3–7:

(1) Corner PLLs do not support DPA.

(2) Not all eight phases are used by the receiver channel or transmitter channel in non-DPA mode.

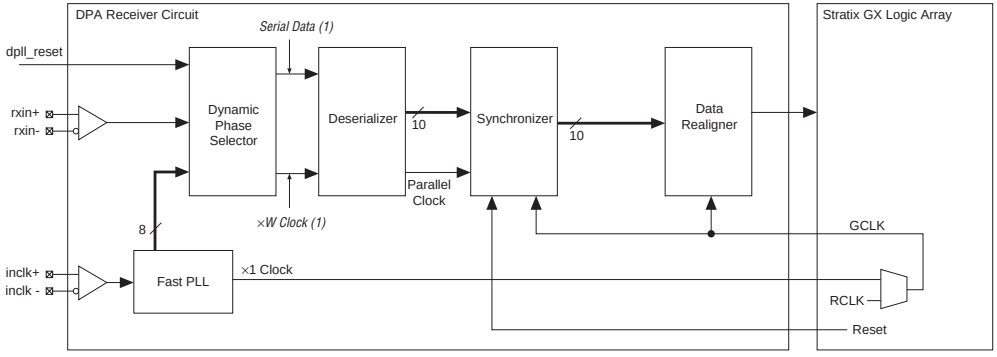
(3) The center PLLs can only clock 20 transceivers in either direction. Using Fast PLL2, you can clock a total of 40 transceivers, 20 in each direction.

DPA Operation

The DPA receiver circuitry contains the dynamic phase selector, the deserializer, the synchronizer, and the data realigner (see Figure 3–8). This section describes the DPA operation, synchronization and data realignment. In the SERDES with DPA mode, the source clock is fed to the fast PLL through the dedicated clock input pins. This clock is multiplied by the multiplication value *W* to match the serial data rate.

For information on the deserializer, see "Principles of SERDES Operation" on page 3–1.

Figure 3–8. DPA Receiver Circuit

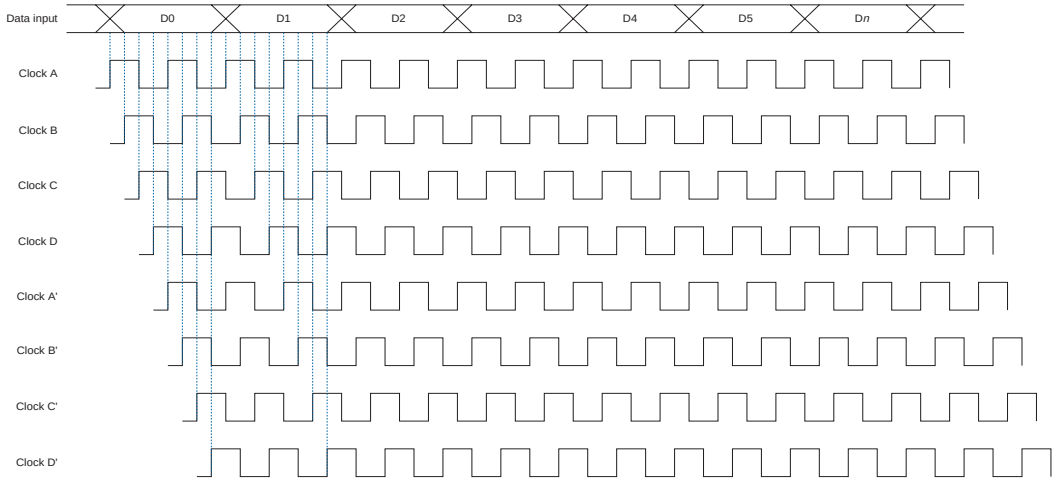


Note to Figure 3–8:
(1) These are phase-matched and retimed high-speed clocks and data.

The dynamic phase selector matches the phase of the high-speed clock and data before sending them to the deserializer.

The fast PLL supplies eight phases of the same clock (each a separate tap from a four-stage differential VCO) to all the differential channels associated with the selected fast PLL. The DPA circuitry inside each channel locks to a phase closest to the serial data's phase and sends the retimed data and the selected clock to the deserializer. The DPA circuitry automatically performs this operation and is not something you select. Each channel's DPA circuit can independently choose a different clock phase. The data phase detection and the clock phase selection process is automatic and continuous. The eight phases of the clock give the DPA circuit a granularity of one eighth of the unit interval (UI) or 125 ps at 1 Gbps. Figure 3–9 illustrates the clocks generated by the fast PLL circuitry and their relationship to a data stream.

Figure 3–9. Fast PLL Clocks & Data Input



Protocols, Training Pattern & DPA Lock Time

The dynamic phase aligner uses a fast PLL for clock multiplication, and the dynamic phase selector for the phase detection and alignment. The dynamic phase aligner uses the high-speed clock out of the dynamic phase selector to deserialize high-speed data and the receiver's source synchronous operations.

At each rising edge of the clock, the dynamic phase selector determines the phase difference between the clock and the data and automatically compensates for the phase difference between the data and clock.

The actual lock time for different data patterns varies depending on the data's transition density (how often the data switches between 1 and 0) and jitter characteristic. The DPA circuitry is designed to lock onto any data pattern with sufficient transition density, so the circuitry works with current and future protocols. Experiments and simulations show that the DPA circuitry locks when the data patterns listed in Table 3–4 are repeated for the specified number of times. There are other suitable patterns not shown in Table 3–4 and/or pattern lengths, but the lock time may vary. The circuit can adjust for any phase variation that may occur during operation.

Table 3–4. Training Patterns for Different Protocols

Protocols	Training Pattern	Number of Repetitions
SPI-4, NPSI	Ten 0's, ten 1's (00000000001111111111)	256
RapidIO	Four 0's, four 1's (00001111) or one 1, two 0's, one 1, four 0's (10010000)	
Other designs	Eight alternating 1's and 0's (10101010 or 01010101)	
SFI-4, XSBI	Not specified	

Phase Synchronizer

Each receiver has its own phase synchronizer. The receiver phase synchronizer aligns the phase of the parallel data from all the receivers to one global clock. The synchronizers in each channel consist of a 4-bit deep and *J*-bit wide FIFO buffer. The parallel clock writes to the FIFO buffer and the global clock (GCLK) reads from the FIFO buffer. The global and parallel clock inputs into the synchronizers must have identical frequencies and differ only in phase. The FIFO buffer never becomes full or empty (because the source and receive signals are frequency locked) when operating within the DPA specifications, and the operation does not require an empty/full flag or read/write enable signals.

Receiver Data Realignment In DPA Mode

While DPA operation aligns the incoming clock phase to the incoming data phase, it does not guarantee the parallelization boundary or byte boundary. When the dynamic phase aligner realigns the data bits, the bits may be shifted out of byte alignment, as shown in Figure 3–10.

Figure 3–10. Misaligned Captured Bits



The dynamic phase selector and synchronizer align the clock and data based on the power-up of both communicating devices, and the channel to channel skew. However, the dynamic phase selector and synchronizer cannot determine the byte boundary, and the data may need to be byte-aligned. The dynamic phase aligner's data realignment circuitry shifts data bits to correct bit misalignments.

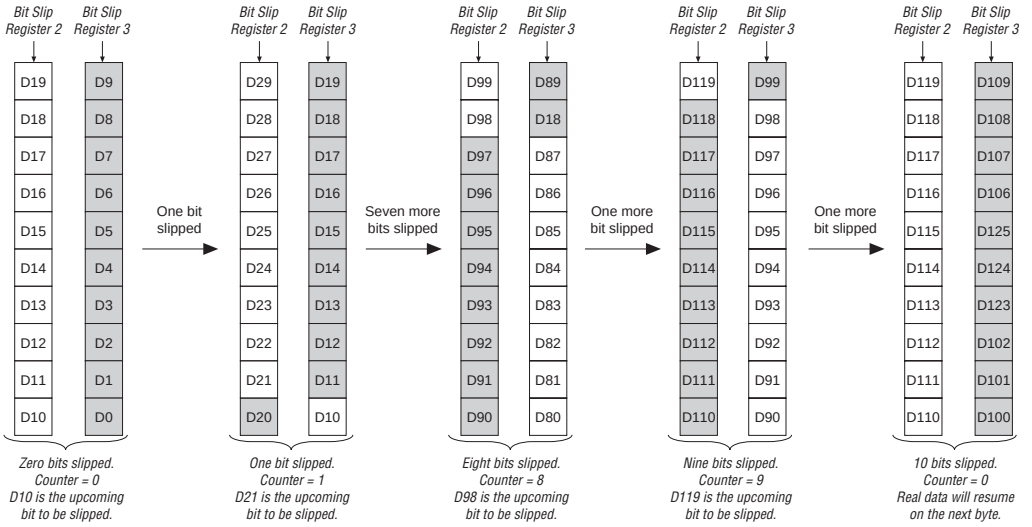
The Stratix GX circuitry contains a data-realignment feature controlled by the logic array. Stratix GX devices perform data realignment on the parallel data after the deserialization block. The data realignment can be performed per channel for more flexibility. The data alignment operation requires a state machine to recognize a specific pattern. The procedure requires the bits to be slipped on the data stream to correctly align the incoming data to the start of the byte boundary.

The DPA uses its realignment circuitry and the global clock for data realignment. Either a device pin or the logic array asserts the internal `rx_channel_data_align` node to activate the DPA data-realignment circuitry. Switching this node from low to high activates the realignment circuitry and the data being transferred to the logic array is shifted by one bit. The data realignment block cannot be bypassed. However, if the `rx_channel_data_align` is not turned on (through the `altvlds` MegaWizard Plug-In Manager), or when it is not toggled, it only acts as a register latency.

A state machine and additional logic can monitor the incoming parallel data and compare it against a known pattern. If the incoming data pattern does not match the known pattern, you can activate the `rx_channel_data_align` node again. Repeat this process until the realigner detects the desired match between the known data pattern and incoming parallel data pattern.

The DPA data-realignment circuitry allows further realignment beyond what the *J* multiplication factor allows. You can set the *J* multiplication factor to be 8 or 10. However, because data must be continuously clocked in on each low-speed clock cycle, the upcoming bit to be realigned and previous *n* –1 bits of data are selected each time the data realignment logic's counter passes *n* –1. At this point the data is selected entirely from bit-slip register 3 (see Figure 3–11) as the counter is reset to 0. The logic array receives a new valid byte of data on the next divided low speed clock cycle. Figure 3–11 shows the data realignment logic output selection from data in the data realignment register 2 and data realignment register 3 based on its current counter value upon continuous request of data slipping from the logic array.

Figure 3–11. DPA Data Realigner



Use the `rx_channel_data_align` signal within the device to activate the data realigner. You can use internal logic or an external pin to control the `rx_channel_data_align` signal. To ensure the rising edge of the `rx_channel_data_align` signal is latched into the control logic, the `rx_channel_data_align` signal should stay high for at least two low-frequency clock cycles.

To manage the alignment procedure, a state machine should be built in the FPGA logic array to generate the realignment signal. The following guidelines outline the requirements for this state machine.

- The design must include an input synchronizing register to ensure that data is synchronized to the ×*W*/*J* clock.
- After the state machine, use another synchronizing register to capture the generated `rx_channel_data_align` signal and synchronize it to the ×*W*/*J* clock.
- Because the skew in the path from the output of this synchronizing register to the PLL is undefined, the state machine must generate a pulse that is high for two *W*/*J* clock periods.
- To guarantee the state machine does not incorrectly generate multiple `rx_channel_data_align` pulses to shift a single bit, the state machine must hold the `rx_channel_data_align` signal low for at least three ×1 clock periods between pulses.

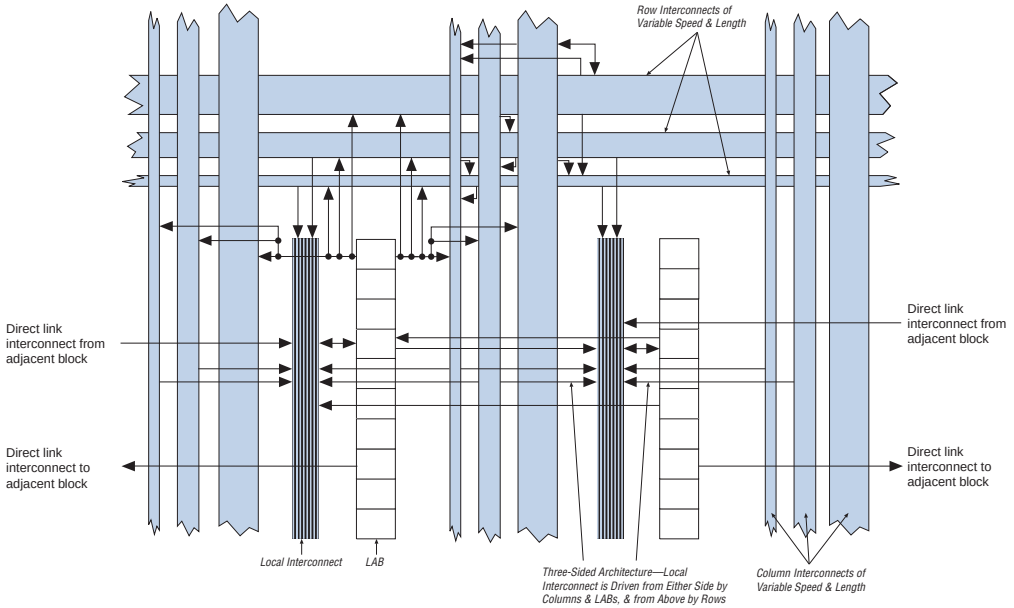
4. Stratix GX Architecture

SGX51004-1.0

Logic Array Blocks

Each LAB consists of 10 LEs, LE carry chains, LAB control signals, local interconnect, LUT chain, and register chain connection lines. The local interconnect transfers signals between LEs in the same LAB. LUT chain connections transfer the output of one LE's LUT to the adjacent LE for fast sequential LUT connections within the same LAB. Register chain connections transfer the output of one LE's register to the adjacent LE's register within an LAB. The Quartus® II Compiler places associated logic within an LAB or adjacent LABs, allowing the use of local, LUT chain, and register chain connections for performance and area efficiency. Figure 4–1 shows the Stratix® GX LAB.

Figure 4–1. Stratix GX LAB Structure

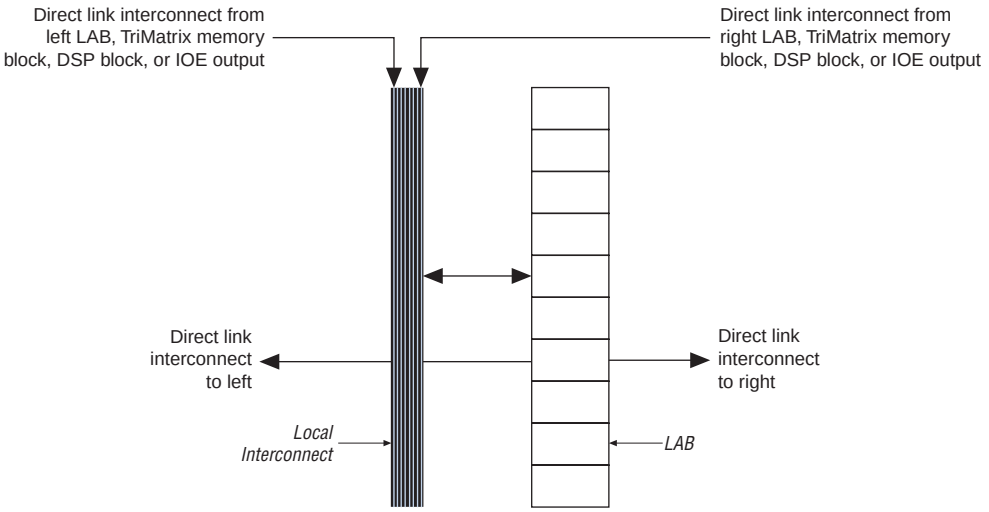


LAB Interconnects

The LAB local interconnect can drive LEs within the same LAB. The LAB local interconnect is driven by column and row interconnects and LE outputs within the same LAB. Neighboring LABs, M512 RAM blocks,

M4K RAM blocks, or DSP blocks from the left and right can also drive an LAB's local interconnect through the direct link connection. The direct link connection feature minimizes the use of row and column interconnects, providing higher performance and flexibility. Each LE can drive 30 other LEs through fast local and direct link interconnects. Figure 4–2 shows the direct link connection.

Figure 4–2. Direct Link Connection



LAB Control Signals

Each LAB contains dedicated logic for driving control signals to its LEs. The control signals include two clocks, two clock enables, two asynchronous clears, synchronous clear, asynchronous preset/load, synchronous load, and add/subtract control signals. This gives a maximum of 10 control signals at a time. Although synchronous load and clear signals are generally used when implementing counters, they can also be used with other functions.

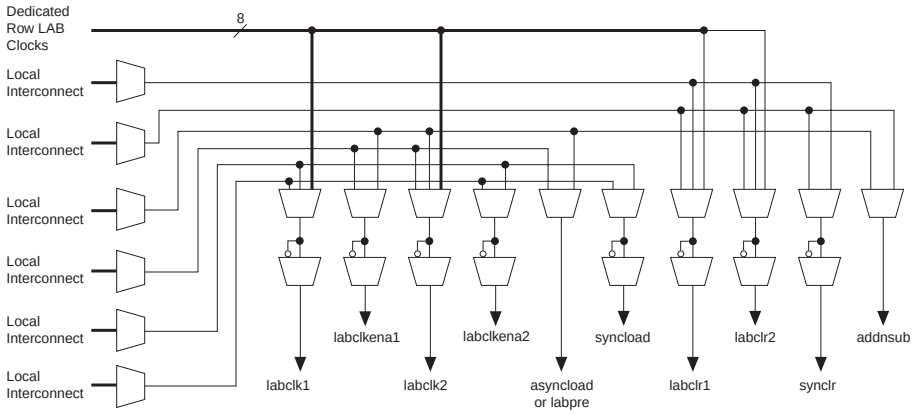
Each LAB can use two clocks and two clock enable signals. Each LAB's clock and clock enable signals are linked. For example, any LE in a particular LAB using the `labclk1` signal also uses `labclkena1`. If the LAB uses both the rising and falling edges of a clock, it also uses both LAB-wide clock signals. De-asserting the clock enable signal turns off the LAB-wide clock.

Each LAB can use two asynchronous clear signals and an asynchronous load/preset signal. The asynchronous load acts as a preset when the asynchronous load data input is tied high.

With the LAB-wide `addnsub` control signal, a single LE can implement a one-bit adder and subtractor. This saves LE resources and improves performance for logic functions such as DSP correlators and signed multipliers that alternate between addition and subtraction depending on data.

The LAB row clocks [7..0] and LAB local interconnect generate the LAB-wide control signals. The MultiTrack™ interconnect's inherent low skew allows clock and control signal distribution in addition to data. Figure 4–3 shows the LAB control signal generation circuit.

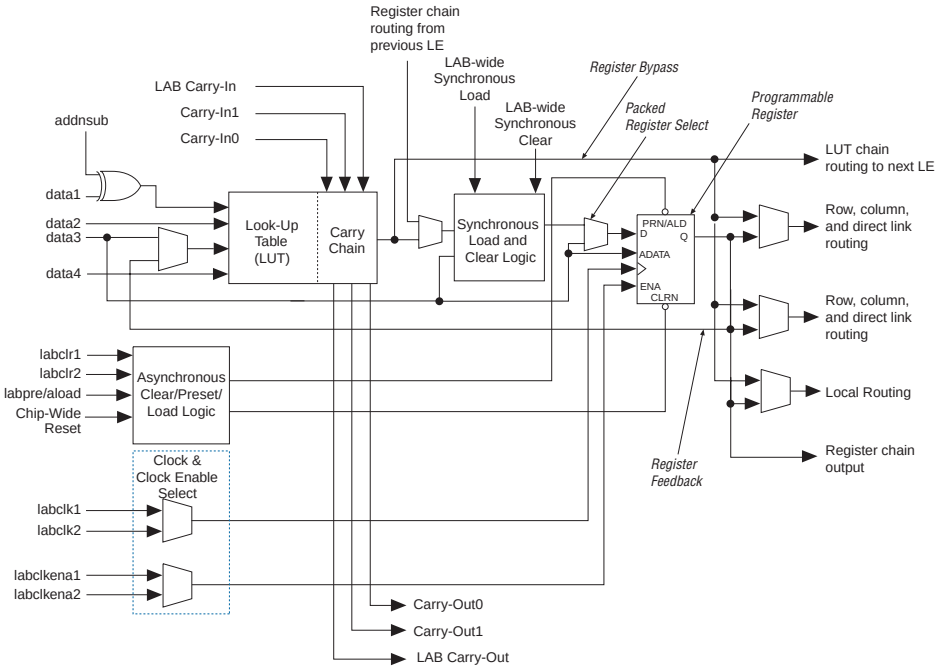
Figure 4–3. LAB-Wide Control Signals



Logic Elements

The smallest unit of logic in the Stratix GX architecture, the LE, is compact and provides advanced features with efficient logic utilization. Each LE contains a four-input LUT, which is a function generator that can implement any function of four variables. In addition, each LE contains a programmable register and carry chain with carry select capability. A single LE also supports dynamic single bit addition or subtraction mode selectable by an LAB-wide control signal. Each LE drives all types of interconnects: local, row, column, LUT chain, register chain, and direct link interconnects. See Figure 4–4.

Figure 4–4. Stratix GX LE



Each LE's programmable register can be configured for D, T, JK, or SR operation. Each register has data, true asynchronous load data, clock, clock enable, clear, and asynchronous load/preset inputs. Global signals, general-purpose I/O pins, or any internal logic can drive the register's clock and clear control signals. Either general-purpose I/O pins or internal logic can drive the clock enable, preset, asynchronous load, and asynchronous data. The asynchronous load data input comes from the `data3` input of the LE. For combinatorial functions, the register is bypassed and the output of the LUT drives directly to the outputs of the LE.

Each LE has three outputs that drive the local, row, and column routing resources. The LUT or register output can drive these three outputs independently. Two LE outputs drive column or row and direct link routing connections and one drives local interconnect resources. This allows the LUT to drive one output while the register drives another output. This feature, called register packing, improves device utilization because the device can use the register and the LUT for unrelated functions. Another special packing mode allows the register output to feed back into the LUT of the same LE so that the register is packed with

its own fan-out LUT. This provides another mechanism for improved fitting. The LE can also drive out registered and unregistered versions of the LUT output.

LUT Chain & Register Chain

In addition to the three general routing outputs, the LEs within an LAB have LUT chain and register chain outputs. LUT chain connections allow LUTs within the same LAB to cascade together for wide input functions. Register chain outputs allow registers within the same LAB to cascade together. The register chain output allows an LAB to use LUTs for a single combinatorial function and the registers to be used for an unrelated shift register implementation. These resources speed up connections between LABs while saving local interconnect resources. See "MultiTrack Interconnect" on page 4–11 for more information on LUT chain and register chain connections.

addnsub Signal

The LE's dynamic adder/subtractor feature saves logic resources by using one set of LEs to implement both an adder and a subtractor. This feature is controlled by the LAB-wide control signal `addnsub`. The `addnsub` signal sets the LAB to perform either A + B or A – B. The LUT computes addition, and subtraction is computed by adding the two's complement of the intended subtractor. The LAB-wide signal converts to two's complement by inverting the B bits within the LAB and setting carry-in = 1 to add one to the least significant bit (LSB). The LSB of an adder/subtractor must be placed in the first LE of the LAB, where the LAB-wide `addnsub` signal automatically sets the carry-in to 1. The Quartus II Compiler automatically places and uses the adder/subtractor feature when using adder/subtractor parameterized functions.

LE Operating Modes

The Stratix GX LE can operate in one of the following modes:

- Normal mode
- Dynamic arithmetic mode

Each mode uses LE resources differently. In each mode, eight available inputs to the LE—the four data inputs from the LAB local interconnect; `carry-in0` and `carry-in1` from the previous LE; the LAB carry-in from the previous carry-chain LAB; and the register chain connection—are directed to different destinations to implement the desired logic function. LAB-wide signals provide clock, asynchronous clear, asynchronous preset load, synchronous clear, synchronous load, and

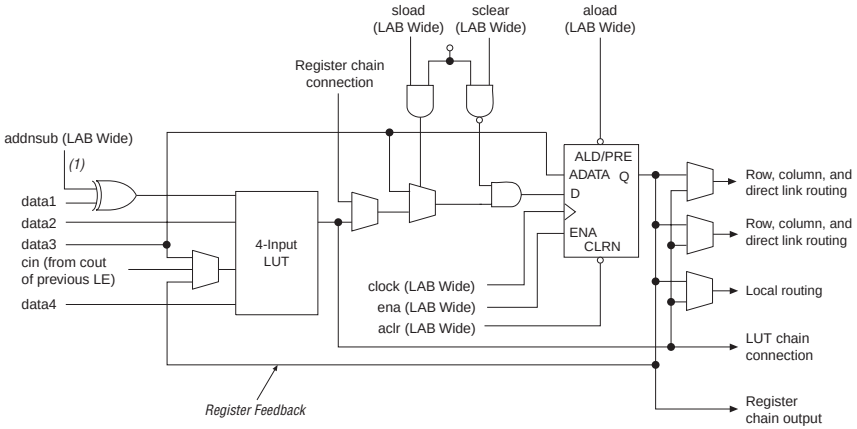
clock enable control for the register. These LAB-wide signals are available in all LE modes. The addnsub control signal is allowed in arithmetic mode.

The Quartus II software, in conjunction with parameterized functions such as library of parameterized modules (LPM) functions, automatically chooses the appropriate mode for common functions such as counters, adders, subtractors, and arithmetic functions. If required, you can also create special-purpose functions that specify which LE operating mode to use for optimal performance.

Normal Mode

The normal mode is suitable for general logic applications and combinatorial functions. In normal mode, four data inputs from the LAB local interconnect are inputs to a four-input LUT (see Figure 4–5). The Quartus II Compiler automatically selects the carry-in or the data3 signal as one of the inputs to the LUT. Each LE can use LUT chain connections to drive its combinatorial output directly to the next LE in the LAB. Asynchronous load data for the register comes from the data3 input of the LE. LEs in normal mode support packed registers.

Figure 4–5. LE in Normal Mode



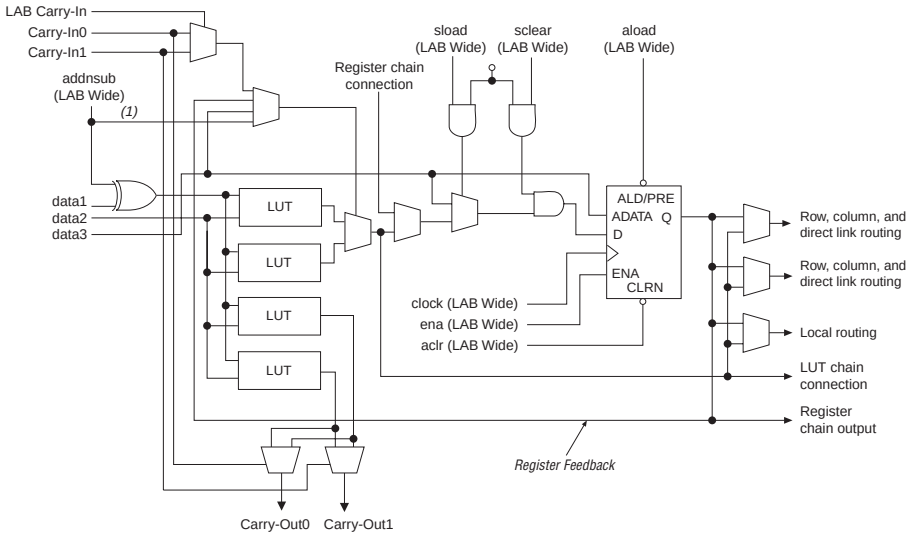
Note to Figure 4–5:
(1) This signal is only allowed in normal mode if the LE is at the end of an adder/subtractor chain.

Dynamic Arithmetic Mode

The dynamic arithmetic mode is ideal for implementing adders, counters, accumulators, wide parity functions, and comparators. An LE in dynamic arithmetic mode uses four 2-input LUTs configurable as a dynamic adder/subtractor. The first two 2-input LUTs compute two summations based on a possible carry-in of 1 or 0; the other two LUTs generate carry outputs for the two chains of the carry select circuitry. As shown in Figure 4–6, the LAB carry-in signal selects either the `carry-in0` or `carry-in1` chain. The selected chain's logic level in turn determines which parallel sum is generated as a combinatorial or registered output. For example, when implementing an adder, the sum output is the selection of two possible calculated sums: `data1` + `data2` + `carry-in0` or `data1` + `data2` + `carry-in1`. The other two LUTs use the `data1` and `data2` signals to generate two possible carry-out signals—one for a carry of 1 and the other for a carry of 0. The `carry-in0` signal acts as the carry select for the `carry-out0` output and `carry-in1` acts as the carry select for the `carry-out1` output. LEs in arithmetic mode can drive out registered and unregistered versions of the LUT output.

The dynamic arithmetic mode also offers clock enable, counter enable, synchronous up/down control, synchronous clear, synchronous load, and dynamic adder/subtractor options. The LAB local interconnect data inputs generate the counter enable and synchronous up/down control signals. The synchronous clear and synchronous load options are LAB-wide signals that affect all registers in the LAB. The Quartus II software automatically places any registers that are not used by the counter into other LABs. The `addnsub` LAB-wide signal controls whether the LE acts as an adder or subtractor.

Figure 4–6. LE in Dynamic Arithmetic Mode



Note to Figure 4–6:
(1) The addnsub signal is tied to the carry input for the first LE of a carry chain only.

Carry-Select Chain

The carry-select chain provides a very fast carry-select function between LEs in arithmetic mode. The carry-select chain uses the redundant carry calculation to increase the speed of carry functions. The LE is configured to calculate outputs for a possible carry-in of 1 and carry-in of 0 in parallel. The carry-in0 and carry-in1 signals from a lower-order bit feed forward into the higher-order bit via the parallel carry chain and feed into both the LUT and the next portion of the carry chain. Carry-select chains can begin in any LE within an LAB.

The speed advantage of the carry-select chain is in the parallel pre-computation of carry chains. Because the LAB carry-in selects the precomputed carry chain, not every LE is in the critical path. Only the propagation delay between LAB carry-in generation (LE 5 and LE 10) are now part of the critical path. This feature allows the Stratix GX architecture to implement high-speed counters, adders, multipliers, parity functions, and comparators of arbitrary width.

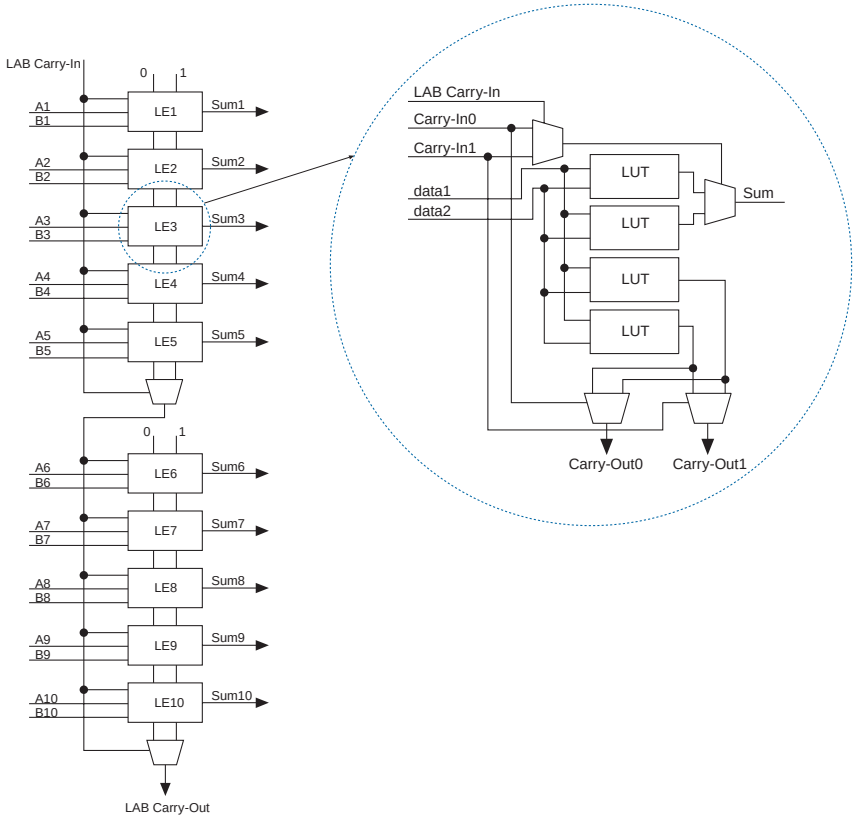
Figure 4–7 shows the carry-select circuitry in an LAB for a 10-bit full adder. One portion of the LUT generates the sum of two bits using the input signals and the appropriate carry-in bit; the sum is routed to the output of the LE. The register can be bypassed for simple adders or used

for accumulator functions. Another portion of the LUT generates carry-out bits. An LAB-wide carry in bit selects which chain to use for the addition of given inputs. The carry-in signal for each chain, `carry-in0` or `carry-in1`, selects the carry-out to carry forward to the carry-in signal of the next-higher-order bit. The final carry-out signal is routed to an LE, where it is fed to local, row, or column interconnects.

The Quartus II Compiler automatically creates carry chain logic during design processing, or you can create it manually during design entry. Parameterized functions such as LPM functions automatically take advantage of carry chains for the appropriate functions.

The Quartus II Compiler creates carry chains longer than 10 LEs by linking LABs together automatically. For enhanced fitting, a long carry chain runs vertically allowing fast horizontal connections to TriMatrix™ memory and DSP blocks. A carry chain can continue as far as a full column.

Figure 4–7. Carry Select Chain



Clear & Preset Logic Control

LAB-wide signals control the logic for the register's clear and preset signals. The LE directly supports an asynchronous clear and preset function. The register preset is achieved through the asynchronous load of a logic high. The direct asynchronous preset does not require a NOT-gate push-back technique. Stratix GX devices support simultaneous preset/ asynchronous load, and clear signals. An asynchronous clear signal takes precedence if both signals are asserted simultaneously. Each LAB supports up to two clears and one preset signal.

In addition to the clear and preset ports, Stratix GX devices provide a chip-wide reset pin (DEV_CLRn) that resets all registers in the device. An option set before compilation in the Quartus II software controls this pin. This chip-wide reset overrides all other control signals.

MultiTrack Interconnect

In the Stratix GX architecture, connections between LEs, TriMatrix memory, DSP blocks, and device I/O pins are provided by the MultiTrack interconnect structure with DirectDrive™ technology. The MultiTrack interconnect consists of continuous, performance-optimized routing lines of different lengths and speeds used for inter- and intra-design block connectivity. The Quartus II Compiler automatically places critical design paths on faster interconnects to improve design performance.

DirectDrive technology is a deterministic routing technology that ensures identical routing resource usage for any function regardless of placement within the device. The MultiTrack interconnect and DirectDrive technology simplify the integration stage of block-based designing by eliminating the re-optimization cycles that typically follow design changes and additions.

The MultiTrack interconnect consists of row and column interconnects that span fixed distances. A routing structure with fixed length resources for all devices allows predictable and repeatable performance when migrating through different device densities. Dedicated row interconnects route signals to and from LABs, DSP blocks, and TriMatrix memory within the same row. These row resources include:

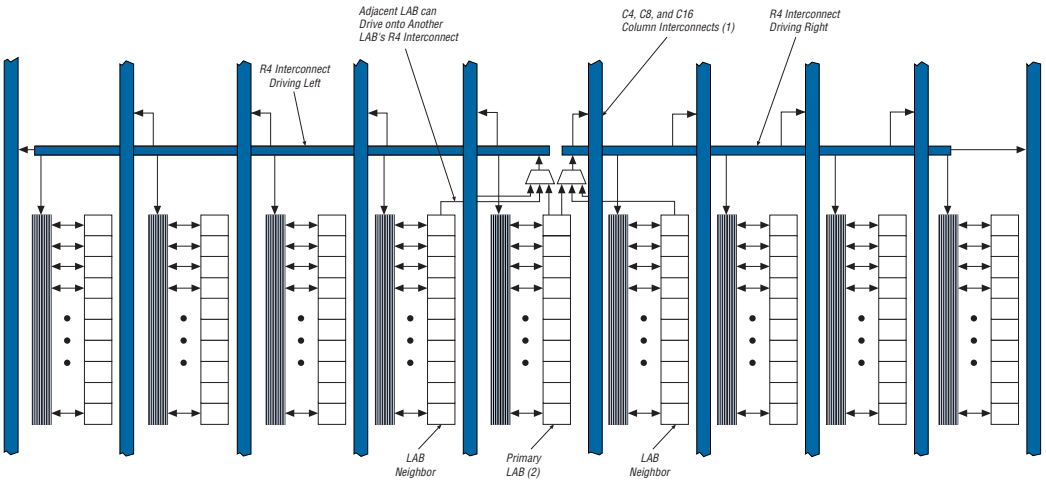
- Direct link interconnects between LABs and adjacent blocks.
- R4 interconnects traversing four blocks to the right or left.
- R8 interconnects traversing eight blocks to the right or left.
- R24 row interconnects for high-speed access across the length of the device.

The direct link interconnect allows an LAB, DSP block, or TriMatrix memory block to drive into the local interconnect of its left and right neighbors and then back into itself. Only one side of a M-RAM block interfaces with direct link and row interconnects. This provides fast communication between adjacent LABs and/or blocks without using row interconnect resources.

The R4 interconnects span four LABs, three LABs and one M512 RAM block, two LABs and one M4K RAM block, or two LABs and one DSP block to the right or left of a source LAB. These resources are used for fast row connections in a four-LAB region. Every LAB has its own set of R4 interconnects to drive either left or right. Figure 4–8 shows R4 interconnect connections from an LAB. R4 interconnects can drive and be driven by DSP blocks and RAM blocks and horizontal IOEs. For LAB interfacing, a primary LAB or LAB neighbor can drive a given R4 interconnect. For R4 interconnects that drive to the right, the primary LAB and right neighbor can drive on to the interconnect. For R4 interconnects that drive to the left, the primary LAB and its left neighbor

can drive on to the interconnect. R4 interconnects can drive other R4 interconnects to extend the range of LABs they can drive. R4 interconnects can also drive C4 and C16 interconnects for connections from one row to another. Additionally, R4 interconnects can drive R24 interconnects.

Figure 4–8. R4 Interconnect Connections



Notes to Figure 4–8:
(1) C4 interconnects can drive R4 interconnects.
(2) This pattern is repeated for every LAB in the LAB row.

The R8 interconnects span eight LABs, M512 or M4K RAM blocks, or DSP blocks to the right or left from a source LAB. These resources are used for fast row connections in an eight-LAB region. Every LAB has its own set of R8 interconnects to drive either left or right. R8 interconnect connections between LABs in a row are similar to the R4 connections shown in Figure 4–8, with the exception that they connect to eight LABs to the right or left, not four. Like R4 interconnects, R8 interconnects can drive and be driven by all types of architecture blocks. R8 interconnects can drive other R8 interconnects to extend their range as well as C8 interconnects for row-to-row connections. One R8 interconnect is faster than two R4 interconnects connected together.

R24 row interconnects span 24 LABs and provide the fastest resource for long row connections between LABs, TriMatrix memory, DSP blocks, and IOEs. The R24 row interconnects can cross M-RAM blocks. R24 row interconnects drive to other row or column interconnects at every fourth

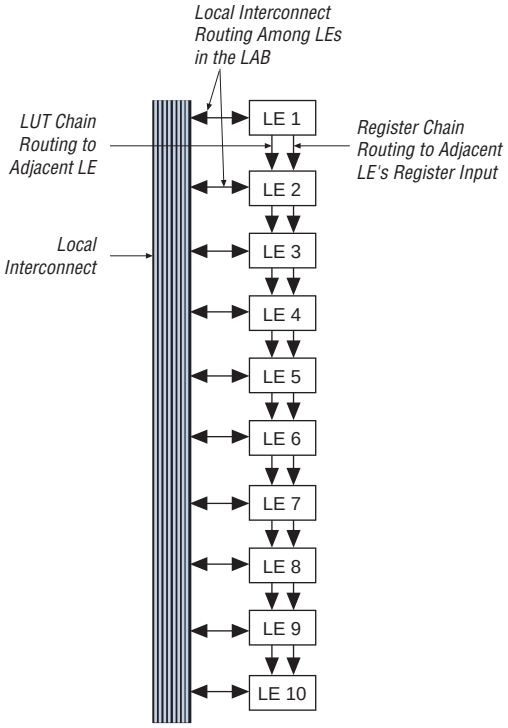
LAB and do not drive directly to LAB local interconnects. R24 row interconnects drive LAB local interconnects via R4 and C4 interconnects. R24 interconnects can drive R24, R4, C16, and C4 interconnects.

The column interconnect operates similarly to the row interconnect and vertically routes signals to and from LABs, TriMatrix memory, DSP blocks, and IOEs. Each column of LABs is served by a dedicated column interconnect, which vertically routes signals to and from LABs, TriMatrix memory and DSP blocks, and horizontal IOEs. These column resources include:

- LUT chain interconnects within an LAB
- Register chain interconnects within an LAB
- C4 interconnects traversing a distance of four blocks in up and down direction
- C8 interconnects traversing a distance of eight blocks in up and down direction
- C16 column interconnects for high-speed vertical routing through the device

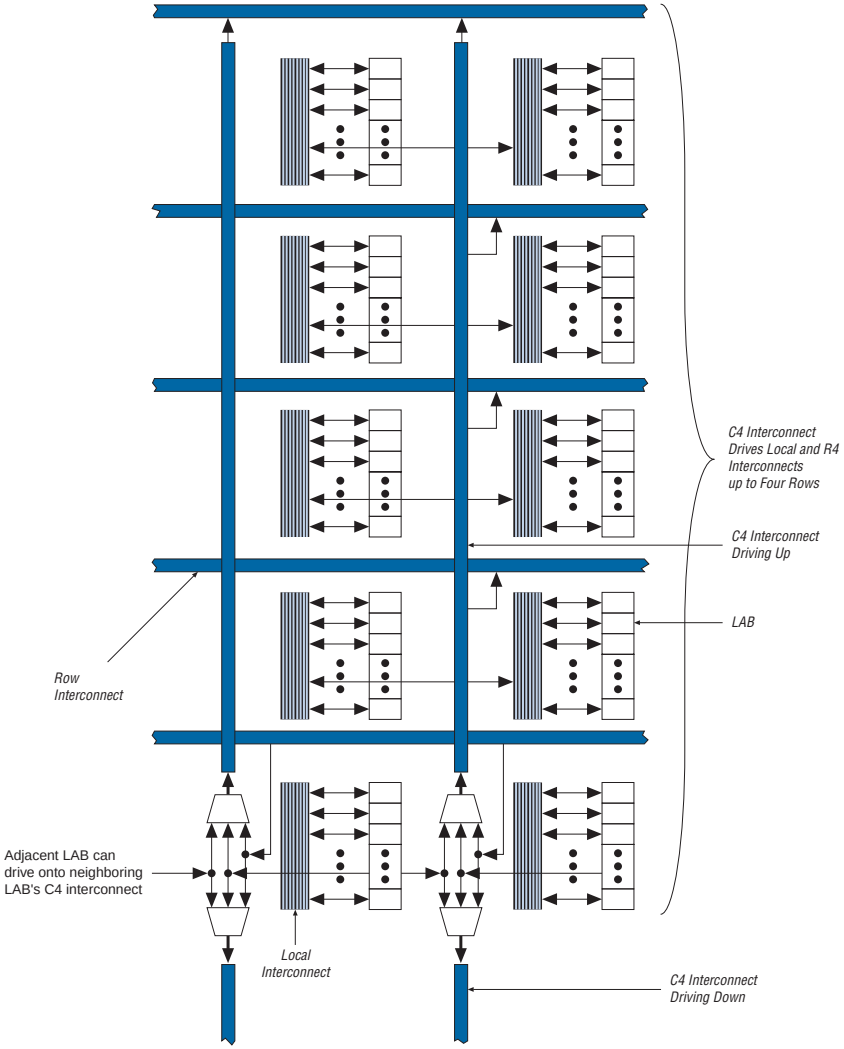
Stratix GX devices include an enhanced interconnect structure within LABs for routing LE output to LE input connections faster using LUT chain connections and register chain connections. The LUT chain connection allows the combinatorial output of an LE to directly drive the fast input of the LE right below it, bypassing the local interconnect. These resources can be used as a high-speed connection for wide fan-in functions from LE 1 to LE 10 in the same LAB. The register chain connection allows the register output of one LE to connect directly to the register input of the next LE in the LAB for fast shift registers. The Quartus II Compiler automatically takes advantage of these resources to improve utilization and performance. Figure 4–9 shows the LUT chain and register chain interconnects.

Figure 4–9. LUT Chain & Register Chain Interconnects



The C4 interconnects span four LABs, M512, or M4K blocks up or down from a source LAB. Every LAB has its own set of C4 interconnects to drive either up or down. Figure 4–10 shows the C4 interconnect connections from an LAB in a column. The C4 interconnects can drive and be driven by all types of architecture blocks, including DSP blocks, TriMatrix memory blocks, and vertical IOEs. For LAB interconnection, a primary LAB or its LAB neighbor can drive a given C4 interconnect.
C4 interconnects can drive each other to extend their range as well as drive row interconnects for column-to-column connections.

Figure 4–10. C4 Interconnect Connections *Note (1)*



Note to Figure 4–10:
(1) Each C4 interconnect can drive either up or down four rows.

C8 interconnects span eight LABs, M512, or M4K blocks up or down from a source LAB. Every LAB has its own set of C8 interconnects to drive either up or down. C8 interconnect connections between the LABs in a column are similar to the C4 connections shown in Figure 4–10 with the exception that they connect to eight LABs above and below. The C8

interconnects can drive and be driven by all types of architecture blocks similar to C4 interconnects. C8 interconnects can drive each other to extend their range as well as R8 interconnects for column-to-column connections. C8 interconnects are faster than two C4 interconnects.

C16 column interconnects span a length of 16 LABs and provide the fastest resource for long column connections between LABs, TriMatrix memory blocks, DSP blocks, and IOEs. C16 interconnects can cross M-RAM blocks and also drive to row and column interconnects at every fourth LAB. C16 interconnects drive LAB local interconnects via C4 and R4 interconnects and do not drive LAB local interconnects directly.

All embedded blocks communicate with the logic array similar to LAB-to-LAB interfaces. Each block (that is, TriMatrix memory and DSP blocks) connects to row and column interconnects and has local interconnect regions driven by row and column interconnects. These blocks also have direct link interconnects for fast connections to and from a neighboring LAB. All blocks are fed by the row LAB clocks, `labclk[7..0]`.

Table 4–1 shows the Stratix GX device's routing scheme.

Table 4–1. Stratix GX Device Routing Scheme

Source	Destination																
	LUT Chain	Register Chain	Local Interconnect	Direct Link Interconnect	R4 Interconnect	R8 Interconnect	R24 Interconnect	C4 Interconnect	C8 Interconnect	C16 Interconnect	LE	M512 RAM Block	M4K RAM Block	M-RAM Block	DSP Blocks	Column IOE	Row IOE
LUT Chain											✓						
Register Chain											✓						
Local Interconnect											✓	✓	✓	✓	✓	✓	✓
Direct Link Interconnect			✓														
R4 Interconnect			✓		✓		✓	✓		✓							
R8 Interconnect			✓			✓			✓								
R24 Interconnect					✓		✓	✓		✓							
C4 Interconnect			✓		✓			✓									
C8 Interconnect			✓			✓			✓								
C16 Interconnect					✓		✓	✓		✓							
LE	✓	✓	✓	✓	✓	✓		✓	✓								
M512 RAM Block			✓	✓	✓	✓		✓	✓								
M4K RAM Block			✓	✓	✓	✓		✓	✓								
M-RAM Block								✓	✓								
DSP Blocks			✓	✓	✓	✓		✓	✓								
Column IOE				✓				✓	✓	✓							
Row IOE				✓		✓	✓	✓	✓	✓							

TriMatrix Memory

TriMatrix memory consists of three types of RAM blocks: M512, M4K, and M-RAM blocks. Although these memory blocks are different, they can all implement various types of memory with or without parity, including true dual-port, simple dual-port, and single-port RAM, ROM, and FIFO buffers. Table 4–2 shows the size and features of the different RAM blocks.

Table 4–2. TriMatrix Memory Features (Part 1 of 2)

Memory Feature	M512 RAM Block (32 × 18 Bits)	M4K RAM Block (128 × 36 Bits)	M-RAM Block (4K × 144 Bits)
Maximum performance	*(1)*	*(1)*	*(1)*
True dual-port memory		✓	✓
Simple dual-port memory	✓	✓	✓
Single-port memory	✓	✓	✓
Shift register	✓	✓	
ROM	✓	✓	*(2)*
FIFO buffer	✓	✓	✓
Byte enable		✓	✓
Parity bits	✓	✓	✓
Mixed clock mode	✓	✓	✓
Memory initialization	✓	✓	
Simple dual-port memory mixed width support	✓	✓	✓
True dual-port memory mixed width support		✓	✓
Power-up conditions	Outputs cleared	Outputs cleared	Outputs unknown
Register clears	Input and output registers	Input and output registers	Output registers
Mixed-port read-during-write	Unknown output/old data	Unknown output/old data	Unknown output

Table 4–2. TriMatrix Memory Features (Part 2 of 2)

Memory Feature	M512 RAM Block (32 × 18 Bits)	M4K RAM Block (128 × 36 Bits)	M-RAM Block (4K × 144 Bits)
Configurations	512 ×1 256 × 2 128 × 4 64 × 8 64 × 9 32 × 16 32 × 18	4K × 1 2K × 2 1K × 4 512 × 8 512 × 9 256 × 16 256 × 18 128 × 32 128 × 36	64K × 8 64K × 9 32K × 16 32K × 18 16K × 32 16K × 36 8K × 64 8K × 72 4K × 128 4K × 144

Notes to Table 4–2:

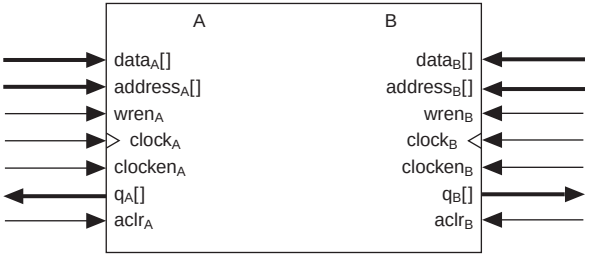
(1) See the *DC & Switching Characteristics* chapter of the *Stratix GX Device Handbook, Volume 1* for maximum performance information.

(2) The M-RAM block does not support memory initializations. However, the M-RAM block can emulate a ROM function using a dual-port RAM bock. The Stratix GX device must write to the dual-port memory once and then disable the write-enable ports afterwards.

Memory Modes

TriMatrix memory blocks include input registers that synchronize writes and output registers to pipeline designs and improve system performance. M4K and M-RAM memory blocks offer a true dual-port mode to support any combination of two-port operations: two reads, two writes, or one read and one write at two different clock frequencies. Figure 4–11 shows true dual-port memory.

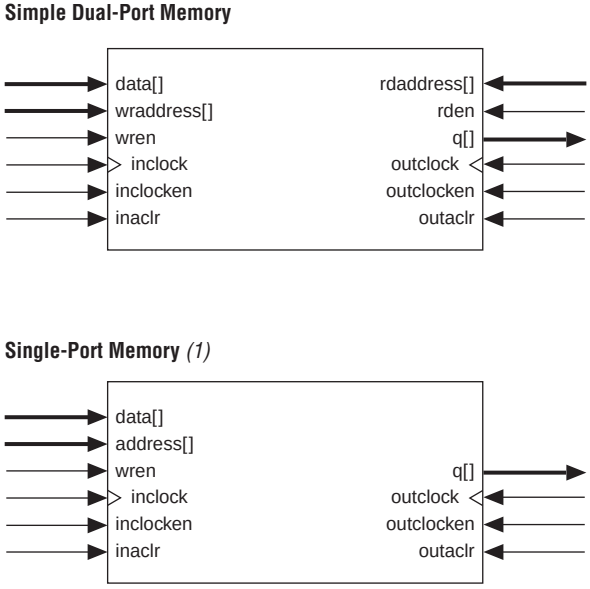
Figure 4–11. True Dual-Port Memory Configuration



In addition to true dual-port memory, the memory blocks support simple dual-port and single-port RAM. Simple dual-port memory supports a simultaneous read and write and can either read old data before the write

occurs or just read the don't care bits. Single-port memory supports non-simultaneous reads and writes, but the q[] port outputs the data once it has been written to the memory (if the outputs are not registered) or after the next rising edge of the clock (if the outputs are registered). For more information, see the *TriMatrix Embedded Memory Blocks in Stratix & Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2*. Figure 4–12 shows these different RAM memory port configurations for TriMatrix memory.

Figure 4–12. Simple Dual-Port & Single-Port Memory Configurations



Note to Figure 4–12:

(1) Two single-port memory blocks can be implemented in a single M4K block as long as each of the two independent block sizes is equal to or less than half of the M4K block size.

The memory blocks also enable mixed-width data ports for reading and writing to the RAM ports in dual-port RAM configuration. For example, the memory block can be written in ×1 mode at port A and read out in ×16 mode from port B.

TriMatrix memory architecture can implement pipelined RAM by registering both the input and output signals to the RAM block. All TriMatrix memory block inputs are registered providing synchronous write cycles. In synchronous operation, the memory block generates its own self-timed strobe write enable (WREN) signal derived from the global

or regional clock. In contrast, a circuit using asynchronous RAM must generate the RAM WREN signal while ensuring its data and address signals meet setup and hold time specifications relative to the WREN signal. The output registers can be bypassed. Flow-through reading is possible in the simple dual-port mode of M512 and M4K RAM blocks by clocking the read enable and read address registers on the negative clock edge and bypassing the output registers.

Two single-port memory blocks can be implemented in a single M4K block as long as each of the two independent block sizes is equal to or less than half of the M4K block size.

The Quartus II software automatically implements larger memory by combining multiple TriMatrix memory blocks. For example, two 256 × 16-bit RAM blocks can be combined to form a 256 × 32-bit RAM block. Memory performance does not degrade for memory blocks using the maximum number of words available in one memory block. Logical memory blocks using less than the maximum number of words use physical blocks in parallel, eliminating any external control logic that would increase delays. To create a larger high-speed memory block, the Quartus II software automatically combines memory blocks with LE control logic.

Parity Bit Support

The memory blocks support a parity bit for each byte. The parity bit, along with internal LE logic, can implement parity checking for error detection to ensure data integrity. You can also use parity-size data words to store user-specified control bits. In the M4K and M-RAM blocks, byte enables are also available for data input masking during write operations.

Shift Register Support

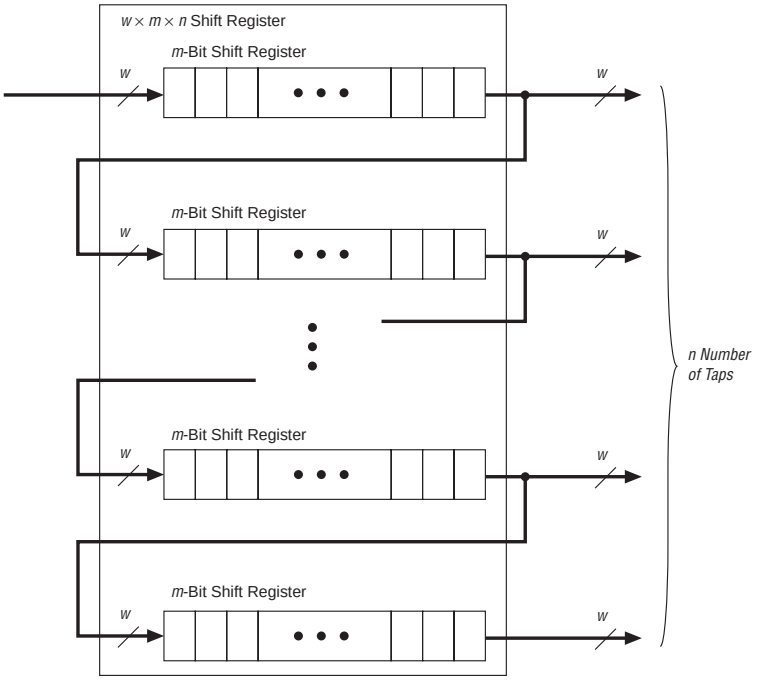
You can configure embedded memory blocks to implement shift registers for DSP applications such as pseudo-random number generators, multi-channel filtering, auto-correlation, and cross-correlation functions. These and other DSP applications require local data storage, traditionally implemented with standard flip-flops, which can quickly consume many logic cells and routing resources for large shift registers. A more efficient alternative is to use embedded memory as a shift register block, which saves logic cell and routing resources and provides a more efficient implementation with the dedicated circuitry.

The size of a $w \times m \times n$ shift register is determined by the input data width (w), the length of the taps (m), and the number of taps (n). The size of a $w \times m \times n$ shift register must be less than or equal to the maximum number of memory bits in the respective block: 576 bits for the M512

RAM block and 4,608 bits for the M4K RAM block. The total number of shift register outputs (number of taps n × width w) must be less than the maximum data width of the RAM block (18 for M512 blocks, 36 for M4K blocks). To create larger shift registers, the memory blocks are cascaded together.

Data is written into each address location at the falling edge of the clock and read from the address at the rising edge of the clock. The shift register mode logic automatically controls the positive and negative edge clocking to shift the data in one clock cycle. Figure 4–13 shows the TriMatrix memory block in the shift register mode.

Figure 4–13. Shift Register Memory Configuration



Memory Block Size

TriMatrix memory provides three different memory sizes for efficient application support. The large number of M512 blocks are ideal for designs with many shallow first-in first-out (FIFO) buffers. M4K blocks provide additional resources for channelized functions that do not require large amounts of storage. The M-RAM blocks provide a large

single block of RAM ideal for data packet storage. The different-sized blocks allow Stratix GX devices to efficiently support variable-sized memory in designs.

The Quartus II software automatically partitions the user-defined memory into the embedded memory blocks using the most efficient size combinations. You can also manually assign the memory to a specific block size or a mixture of block sizes.

M512 RAM Block

The M512 RAM block is a simple dual-port memory block and is useful for implementing small FIFO buffers, DSP, and clock domain transfer applications. Each block contains 576 RAM bits (including parity bits). M512 RAM blocks can be configured in the following modes:

- Simple dual-port RAM
- Single-port RAM
- FIFO
- ROM
- Shift register

When configured as RAM or ROM, you can use an initialization file to pre-load the memory contents.

The memory address depths and output widths can be configured as 512 × 1, 256 × 2, 128 × 4, 64 × 8 (64 × 9 bits with parity), and 32 × 16 (32 × 18 bits with parity). Mixed-width configurations are also possible, allowing different read and write widths. Table 4–3 summarizes the possible M512 RAM block configurations.

Table 4–3. M512 RAM Block Configurations (Simple Dual-Port RAM)

Read Port	Write Port						
	512 × 1	256 × 2	128 × 4	64 × 8	32 × 16	64 × 9	32 × 18
512 × 1	✓	✓	✓	✓	✓		
256 × 2	✓	✓	✓	✓	✓		
128 × 4	✓	✓	✓		✓		
64 × 8	✓	✓		✓			
32 × 16	✓	✓	✓		✓		
64 × 9						✓	
32 × 18							✓

When the M512 RAM block is configured as a shift register block, a shift register of size up to 576 bits is possible.

The M512 RAM block can also be configured to support serializer and deserializer applications. By using the mixed-width support in combination with DDR I/O standards, the block can function as a SERDES to support low-speed serial I/O standards using global or regional clocks. See "I/O Structure" on page 4–96 for details on dedicated SERDES in Stratix GX devices.

M512 RAM blocks can have different clocks on its inputs and outputs. The `wren`, `datain`, and write address registers are all clocked together from one of the two clocks feeding the block. The read address, `rden`, and output registers can be clocked by either of the two clocks driving the block. This allows the RAM block to operate in read/write or input/output clock modes. Only the output register can be bypassed. The eight `labclk` signals or local interconnect can drive the `inclock`, `outclock`, `wren`, `rden`, `inclr`, and `outclr` signals. Because of the advanced interconnect between the LAB and M512 RAM blocks, LEs can also control the `wren` and `rden` signals and the RAM clock, clock enable, and asynchronous clear signals. Figure 4–14 shows the M512 RAM block control signal generation logic.

The RAM blocks within Stratix GX devices have local interconnects to allow LEs and interconnects to drive into RAM blocks. The M512 RAM block local interconnect is driven by the R4, R8, C4, C8, and direct link interconnects from adjacent LABs. The M512 RAM blocks can communicate with LABs on either the left or right side through these row interconnects or with LAB columns on the left or right side with the column interconnects. Up to 10 direct link input connections to the M512 RAM block are possible from the left adjacent LABs and another 10 possible from the right adjacent LAB. M512 RAM outputs can also connect to left and right LABs through 10 direct link interconnects. The M512 RAM block has equal opportunity for access and performance to and from LABs on either its left or right side. Figure 4–15 shows the M512 RAM block to logic array interface.

Figure 4–14. M512 RAM Block Control Signals

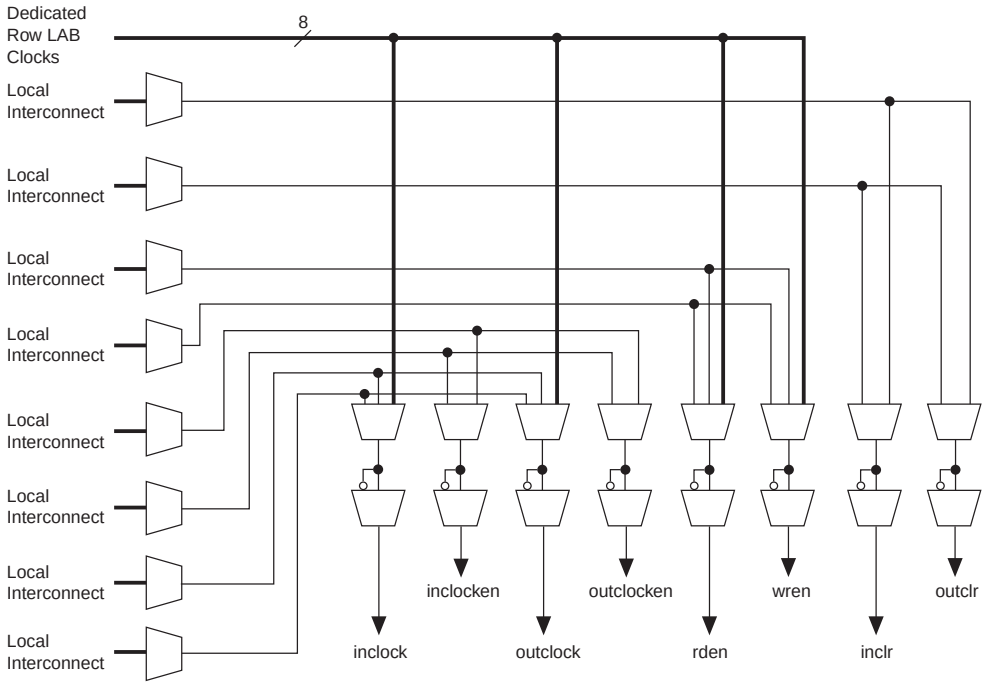
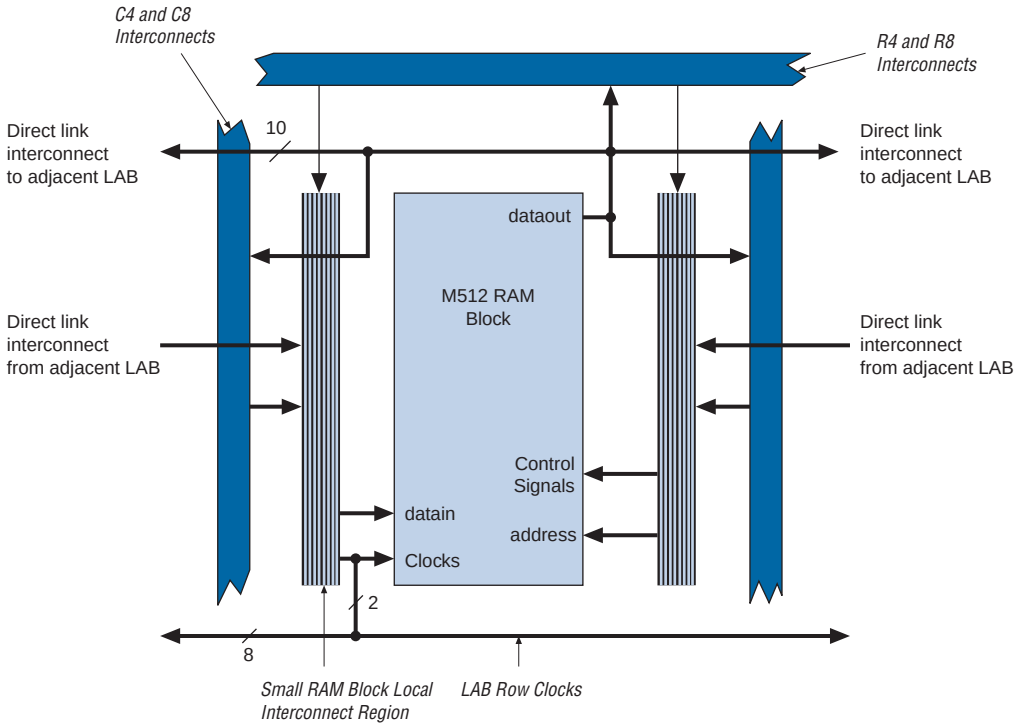


Figure 4–15. M512 RAM Block LAB Row Interface



M4K RAM Blocks

The M4K RAM block includes support for true dual-port RAM. The M4K RAM block implements buffers for a wide variety of applications such as storing processor code, implementing lookup schemes, and implementing larger memory applications. Each block contains 4,608 RAM bits (including parity bits). M4K RAM blocks can be configured in the following modes:

- True dual-port RAM
- Simple dual-port RAM
- Single-port RAM
- FIFO
- ROM
- Shift register

When configured as RAM or ROM, you can use an initialization file to pre-load the memory contents.

The memory address depths and output widths can be configured as 4,096 × 1, 2,048 × 2, 1,024 × 4, 512 × 8 (or 512 × 9 bits), 256 × 16 (or 256 × 18 bits), and 128 × 32 (or 128 × 36 bits). The 128 × 32- or 36-bit configuration is not available in the true dual-port mode. Mixed-width configurations are also possible, allowing different read and write widths. Tables 4–4 and 4–5 summarize the possible M4K RAM block configurations.

Table 4–4. M4K RAM Block Configurations (Simple Dual-Port)

Read Port	Write Port								
	4K 1	2K × 2	1K ° 4	512 ° 8	256 ° 16	128 ° 32	512 ° 9	256 ° 18	128 ° 36
4K × 1	✓	✓	✓	✓	✓	✓			
2K × 2	✓	✓	✓	✓	✓	✓			
1K × 4	✓	✓	✓	✓	✓	✓			
512 × 8	✓	✓	✓	✓	✓	✓			
256 × 16	✓	✓	✓	✓	✓	✓			
128 × 32	✓	✓	✓	✓	✓	✓			
512 × 9							✓	✓	✓
256 × 18							✓	✓	✓
128 × 36							✓	✓	✓

Table 4–5. M4K RAM Block Configurations (True Dual-Port)

Port A	Port B						
	4K × 1	2K × 2	1K × 4	512 × 8	256 × 16	512 × 9	256 × 18
4K × 1	✓	✓	✓	✓	✓		
2K × 2	✓	✓	✓	✓	✓		
1K × 4	✓	✓	✓	✓	✓		
512 × 8	✓	✓	✓	✓	✓		
256 × 16	✓	✓	✓	✓	✓		
512 × 9						✓	✓
256 × 18						✓	✓

When the M4K RAM block is configured as a shift register block, you can create a shift register up to 4,608 bits ($w \times m \times n$).

M4K RAM blocks support byte writes when the write port has a data width of 16, 18, 32, or 36 bits. The byte enables allow the input data to be masked so the device can write to specific bytes. The unwritten bytes retain the previous written value. Table 4–6 summarizes the byte selection.

Table 4–6. Byte Enable for M4K Blocks Notes (1), (2)

byteena[3..0]	datain ×18	datain ×36
[0] = 1	[8..0]	[8..0]
[1] = 1	[17..9]	[17..9]
[2] = 1	–	[26..18]
[3] = 1	–	[35..27]

Notes to Table 4–6:
(1) Any combination of byte enables is possible.
(2) Byte enables can be used in the same manner with 8-bit words, that is, in ×16 and ×32 modes.

The M4K RAM blocks allow for different clocks on their inputs and outputs. Either of the two clocks feeding the block can clock M4K RAM block registers (`renwe`, address, byte enable, `datain`, and output registers). Only the output register can be bypassed. The eight `labclk` signals or local interconnects can drive the control signals for the A and B ports of the M4K RAM block. LEs can also control the `clock_a`, `clock_b`, `renwe_a`, `renwe_b`, `clr_a`, `clr_b`, `clocken_a`, and `clocken_b` signals, as shown in Figure 4–16.

The R4, R8, C4, C8, and direct link interconnects from adjacent LABs drive the M4K RAM block local interconnect. The M4K RAM blocks can communicate with LABs on either the left or right side through these row resources or with LAB columns on either the right or left with the column resources. Up to 10 direct link input connections to the M4K RAM Block are possible from the left adjacent LABs and another 10 possible from the right adjacent LAB. M4K RAM block outputs can also connect to left and right LABs through 10 direct link interconnects each. Figure 4–17 shows the M4K RAM block to logic array interface.

Figure 4–16. M4K RAM Block Control Signals

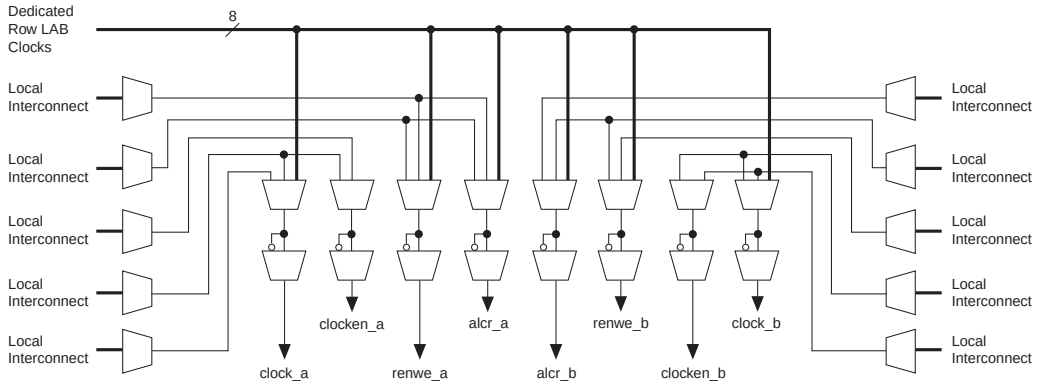
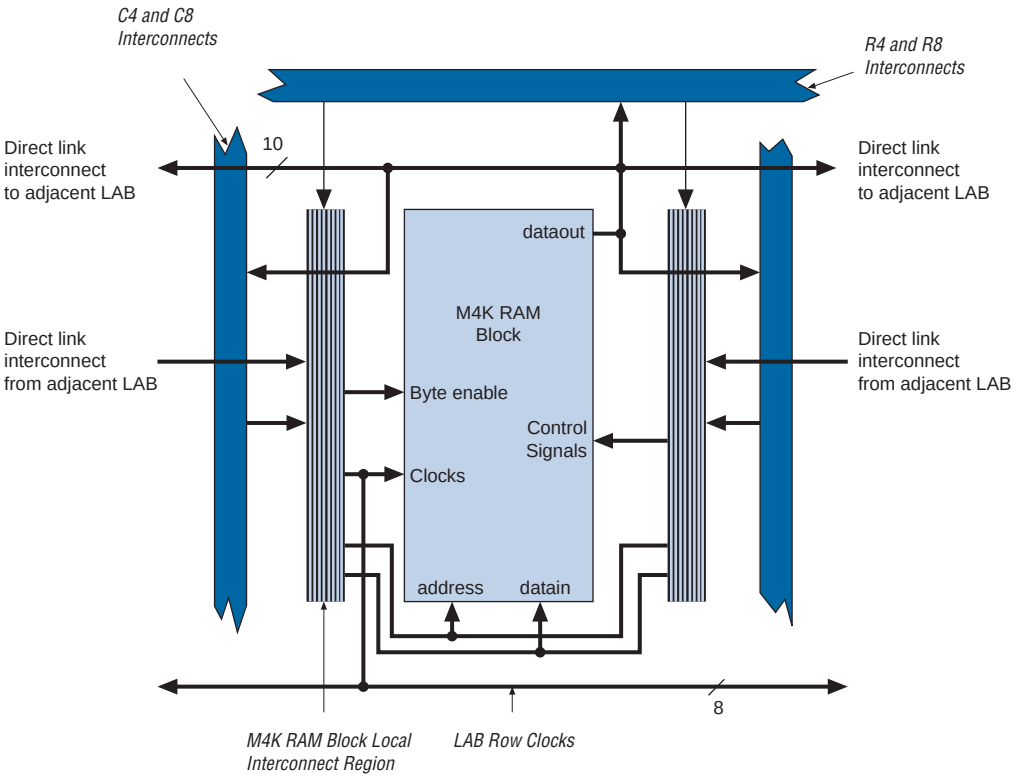


Figure 4–17. M4K RAM Block LAB Row Interface



M-RAM Block

The largest TriMatrix memory block, the M-RAM block, is useful for applications where a large volume of data must be stored on-chip. Each block contains 589,824 RAM bits (including parity bits). The M-RAM block can be configured in the following modes:

- True dual-port RAM
- Simple dual-port RAM
- Single-port RAM
- FIFO RAM

You cannot use an initialization file to initialize the contents of a M-RAM block. All M-RAM block contents power up to an undefined value. Only synchronous operation is supported in the M-RAM block, so all inputs are registered. Output registers can be bypassed. The memory address and output width can be configured as 64K × 8 (or 64K × 9 bits), 32K × 16 (or 32K × 18 bits), 16K × 32 (or 16K × 36 bits), 8K × 64 (or 8K × 72 bits), and 4K × 128 (or 4K × 144 bits). The 4K × 128 configuration is unavailable in true dual-port mode because there are a total of 144 data output drivers in the block. Mixed-width configurations are also possible, allowing different read and write widths. Tables 4–7 and 4–8 summarize the possible M-RAM block configurations:

Table 4–7. M-RAM Block Configurations (Simple Dual-Port)

Read Port	Write Port				
	64K × 9	32K × 18	16K × 36	8K × 72	4K × 144
64K × 9	✓	✓	✓	✓	
32K × 18	✓	✓	✓	✓	
16K × 36	✓	✓	✓	✓	
8K × 72	✓	✓	✓	✓	
4K × 144					✓

Table 4–8. M-RAM Block Configurations (True Dual-Port)				
Port A	Port B			
	64K × 9	32K × 18	16K × 36	8K × 72
64K × 9	✓	✓	✓	✓
32K × 18	✓	✓	✓	✓
16K × 36	✓	✓	✓	✓
8K × 72	✓	✓	✓	✓

The read and write operation of the memory is controlled by the WREN signal, which sets the ports into either read or write modes. There is no separate read enable (RE) signal.

Writing into RAM is controlled by both the WREN and byte enable (byteena) signals for each port. The default value for the byteena signal is high, in which case writing is controlled only by the WREN signal. The byte enables are available for the ×18, ×36, and ×72 modes. In the ×144 simple dual-port mode, the two sets of byteena signals (byteena_a and byteena_b) are combined to form the necessary 16 byte enables. Tables 4–9 and 4–10 summarize the byte selection.

Table 4–9. Byte Enable for M-RAM Blocks Notes (1), (2)			
byteena[3..0]	datain ×18	datain ×36	datain ×72
[0] = 1	[8..0]	[8..0]	[8..0]
[1] = 1	[17..9]	[17..9]	[17..9]
[2] = 1	–	[26..18]	[26..18]
[3] = 1	–	[35..27]	[35..27]
[4] = 1	–	–	[44..36]
[5] = 1	–	–	[53..45]
[6] = 1	–	–	[62..54]
[7] = 1	–	–	[71..63]

Table 4–10. M-RAM Combined Byte Selection for ×144 Mode Notes (1), (2)	
byteena[15..0]	**datain ×144**
[0] = 1	[8..0]
[1] = 1	[17..9]
[2] = 1	[26..18]
[3] = 1	[35..27]
[4] = 1	[44..36]
[5] = 1	[53..45]
[6] = 1	[62..54]
[7] = 1	[71..63]
[8] = 1	[80..72]
[9] = 1	[89..81]
[10] = 1	[98..90]
[11] = 1	[107..99]
[12] = 1	[116..108]
[13] = 1	[125..117]
[14] = 1	[134..126]
[15] = 1	[143..135]

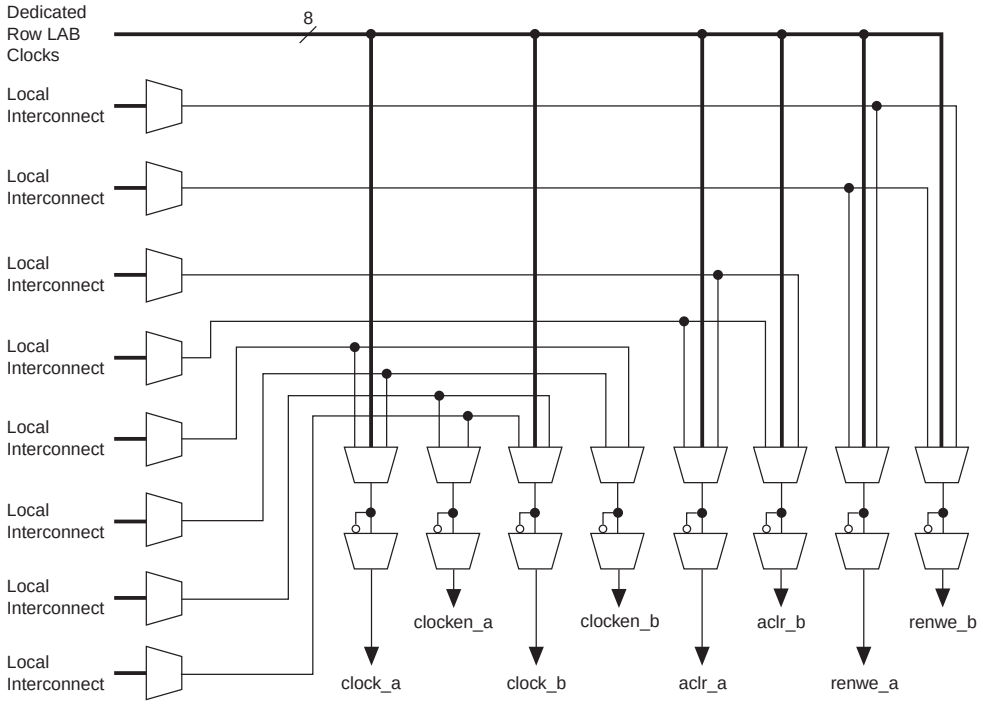
Notes to Tables 4–9 and 4–10:

(1) Any combination of byte enables is possible.

(2) Byte enables can be used in the same manner with 8-bit words, that is, in ×16, ×32, ×64, and ×128 modes.

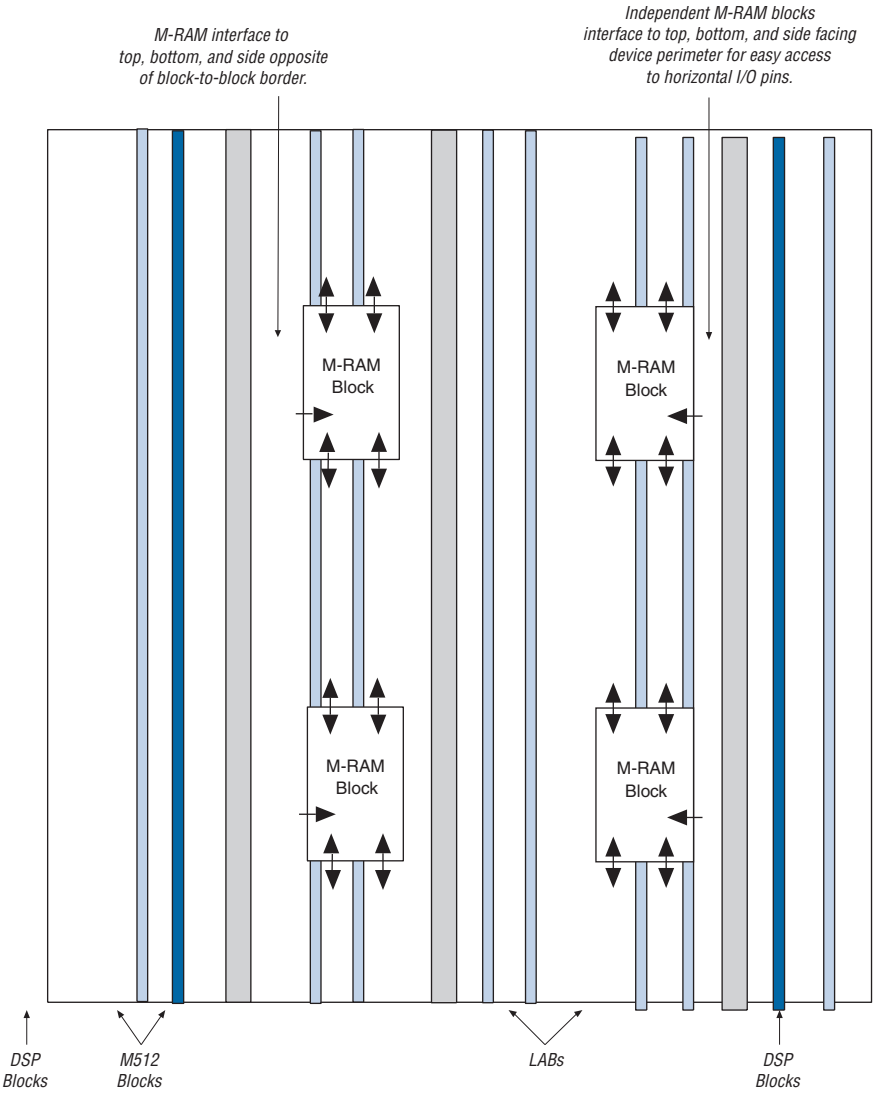
Similar to all RAM blocks, M-RAM blocks can have different clocks on their inputs and outputs. All input registers—`renwe`, `datain`, address, and byte enable registers—are clocked together from either of the two clocks feeding the block. The output register can be bypassed. The eight `labclk` signals or local interconnect can drive the control signals for the A and B ports of the M-RAM block. LEs can also control the `clock_a`, `clock_b`, `renwe_a`, `renwe_b`, `clr_a`, `clr_b`, `clocken_a`, and `clocken_b` signals as shown in Figure 4–18.

Figure 4–18. M-RAM Block Control Signals



One of the M-RAM block's horizontal sides drive the address and control signal (clock, `renwe`, `byteena`, etc.) inputs. Typically, the horizontal side closest to the device perimeter contains the interfaces. The one exception is when two M-RAM blocks are paired next to each other. In this case, the side of the M-RAM block opposite the common side of the two blocks contains the input interface. The top and bottom sides of any M-RAM block contain data input and output interfaces to the logic array. The top side has 72 data inputs and 72 data outputs for port B, and the bottom side has another 72 data inputs and 72 data outputs for port A. Figure 4–19 shows an example floorplan for the EP1SGX40 device and the location of the M-RAM interfaces.

Figure 4–19. EP1SGX40 Device with M-RAM Interface Locations *Note (1)*

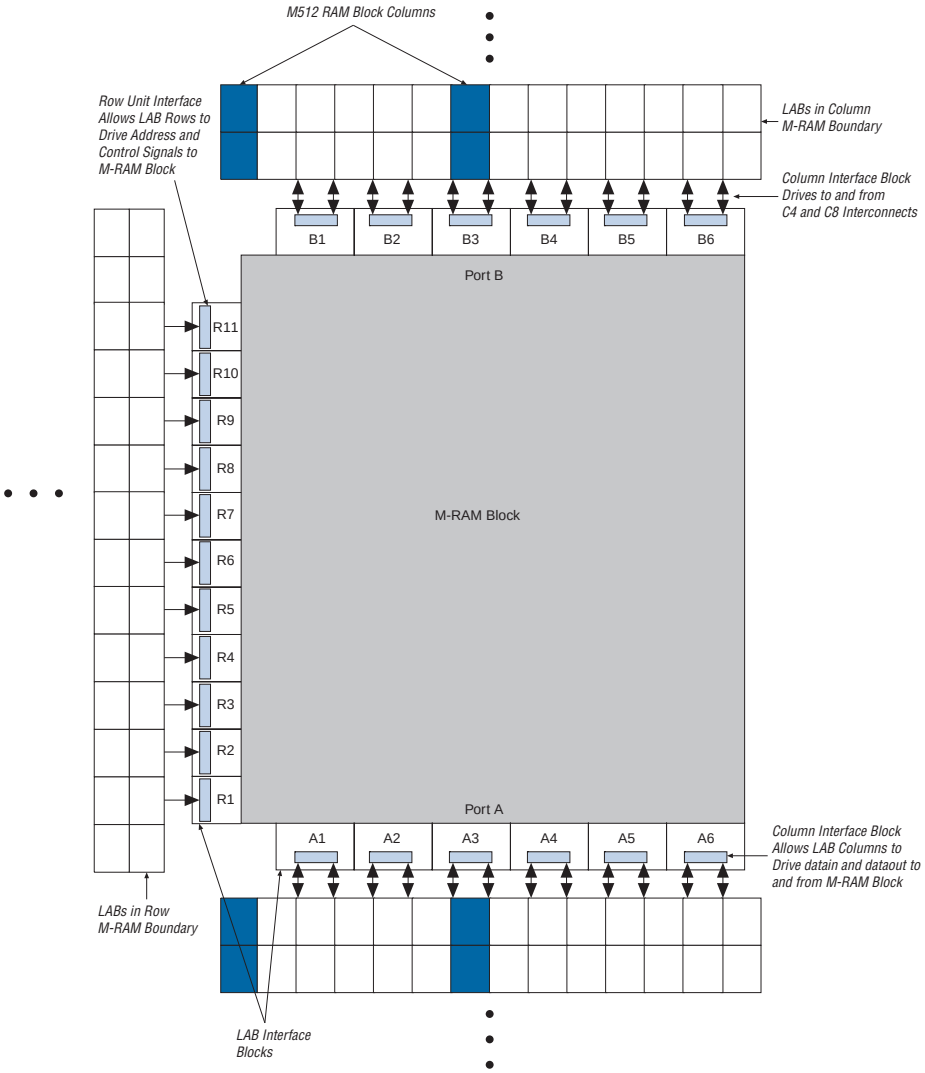


Note to Figure 4–19:
(1) Device shown is an EP1SGX40 device. The number and position of M-RAM blocks varies in other devices.

The M-RAM block local interconnect is driven by the R4, R8, C4, C8, and direct link interconnects from adjacent LABs. For independent M-RAM blocks, up to 10 direct link address and control signal input connections to the M-RAM block are possible from the left adjacent LABs for M-RAM

blocks facing to the left, and another 10 possible from the right adjacent LABs for M-RAM blocks facing to the right. For column interfacing, every M-RAM column unit connects to the right and left column lines, allowing each M-RAM column unit to communicate directly with three columns of LABs. Figures 4–20 through 4–22 show the interface between the M-RAM block and the logic array.

Figure 4–20. Left-Facing M-RAM to Interconnect Interface *Notes (1), (2)*



Notes to Figure 4–20:

(1) Only R24 and C16 interconnects cross the M-RAM block boundaries.

(2) The right-facing M-RAM block has interface blocks on the right side, but none on the left. B1 to B6 and A1 to A6 orientation is clipped across the vertical axis for right-facing M-RAM blocks.

Figure 4–21. M-RAM Row Unit Interface to Interconnect

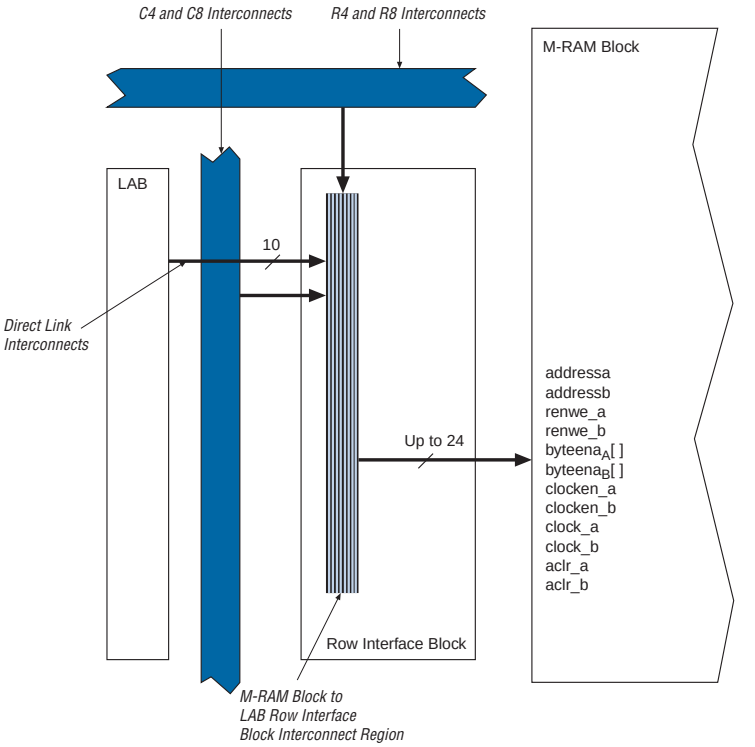


Figure 4–22. M-RAM Column Unit Interface to Interconnect

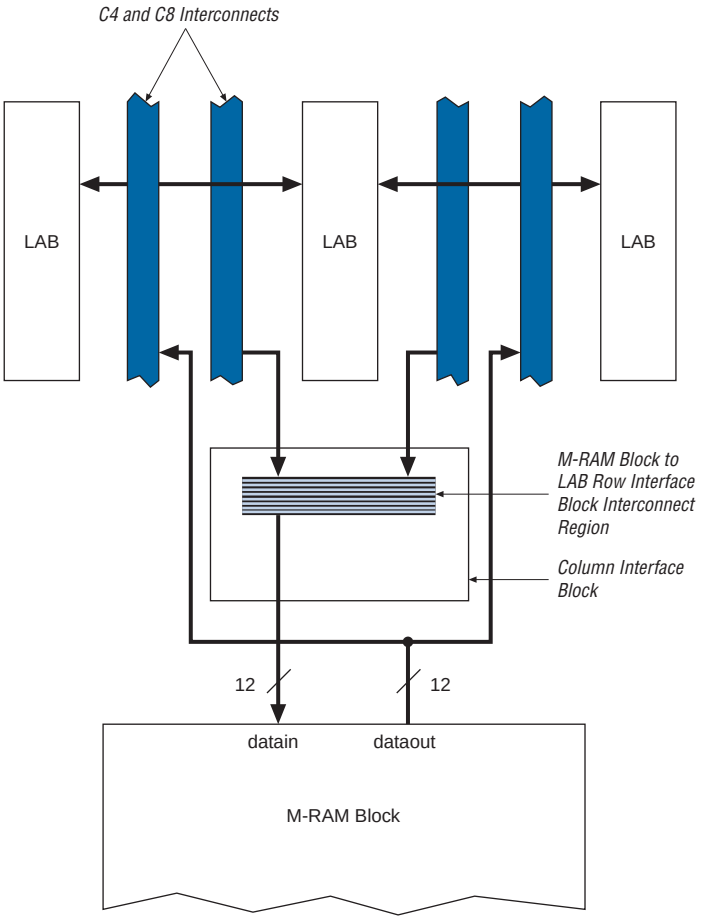


Table 4–11 shows the input and output data signal connections for the column units (B1 to B6 and A1 to A6). It also shows the address and control signal input connections to the row units (R1 to R11).

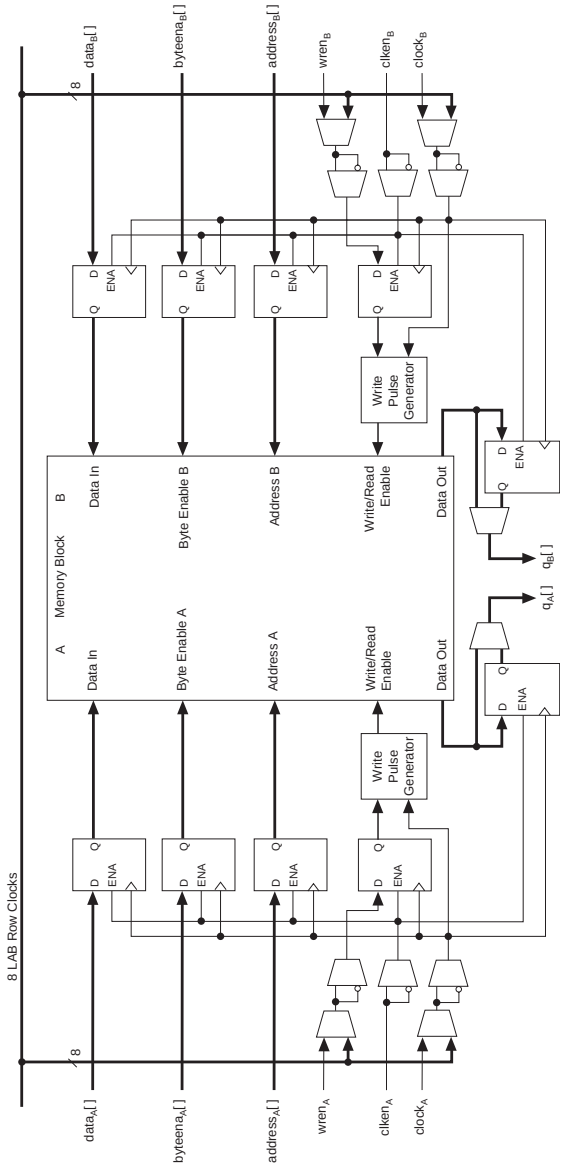
Table 4–11. M-RAM Row & Column Interface Unit Signals

Unit Interface Block	Input SIgnals	Output Signals
R1	addressa[7..0]	
R2	addressa[15..8]	
R3	byte_enable_a[7..0] renwe_a	
R4	-	
R5	-	
R6	clock_a clocken_a clock_b clocken_b	
R7	-	
R8	-	
R9	byte_enable_b[7..0] renwe_b	
R10	addressb[15..8]	
R11	addressb[7..0]	
B1	datain_b[71..60]	dataout_b[71..60]
B2	datain_b[59..48]	dataout_b[59..48]
B3	datain_b[47..36]	dataout_b[47..36]
B4	datain_b[35..24]	dataout_b[35..24]
B5	datain_b[23..12]	dataout_b[23..12]
B6	datain_b[11..0]	dataout_b[11..0]
A1	datain_a[71..60]	dataout_a[71..60]
A2	datain_a[59..48]	dataout_a[59..48]
A3	datain_a[47..36]	dataout_a[47..36]
A4	datain_a[35..24]	dataout_a[35..24]
A5	datain_a[23..12]	dataout_a[23..12]
A6	datain_a[11..0]	dataout_a[11..0]

Independent Clock Mode

The memory blocks implement independent clock mode for true dual-port memory. In this mode, a separate clock is available for each port (ports A and B). Clock A controls all registers on the port A side, while clock B controls all registers on the port B side. Each port, A and B, also supports independent clock enables and asynchronous clear signals for port A and B registers. Figure 4–23 shows a TriMatrix memory block in independent clock mode.

Figure 4–23. Independent Clock Mode *Note (1)*

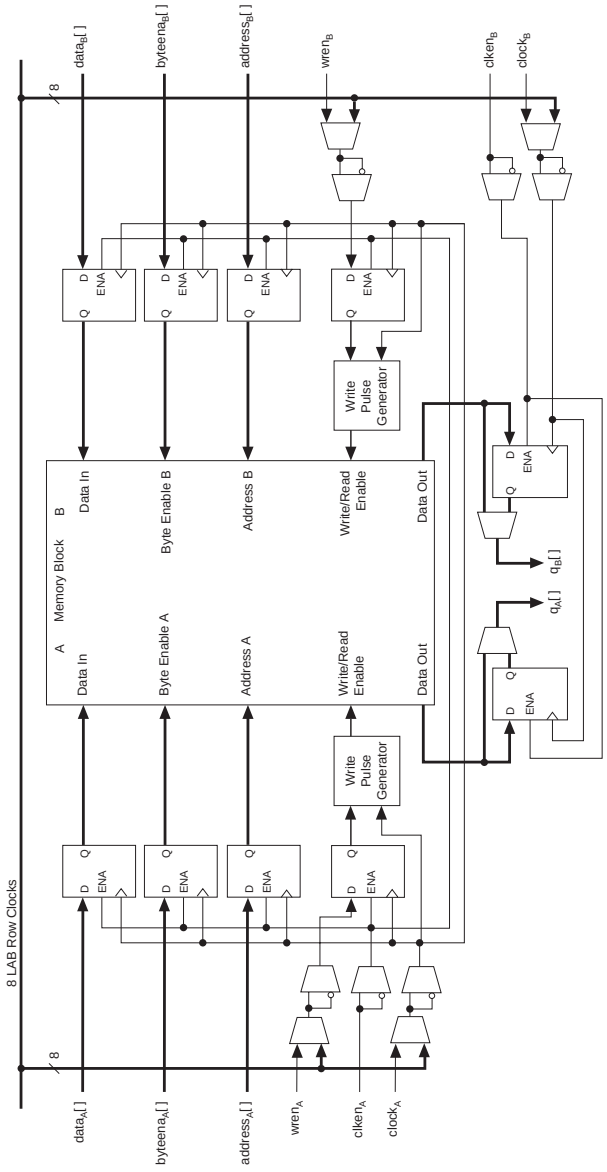


Note to Figure 4–23:
(1) All registers shown have asynchronous clear ports.

Input/Output Clock Mode

Input/output clock mode can be implemented for both the true and simple dual-port memory modes. On each of the two ports, A or B, one clock controls all registers for inputs into the memory block: data input, `wren`, and address. The other clock controls the block's data output registers. Each memory block port, A or B, also supports independent clock enables and asynchronous clear signals for input and output registers. Figures 4–24 and 4–25 show the memory block in input/output clock mode.

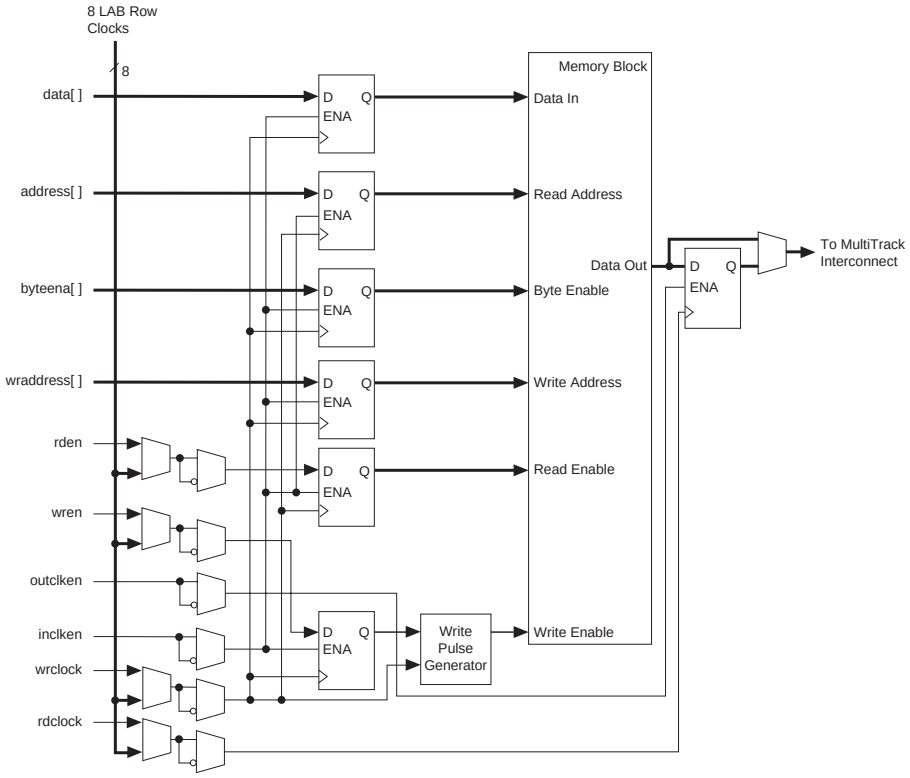
Figure 4–24. Input/Output Clock Mode in True Dual-Port Mode *Note (1)*



Note to Figure 4–24:

(1) All registers shown have asynchronous clear ports.

Figure 4–25. Input/Output Clock Mode in Simple Dual-Port Mode *Note (1)*

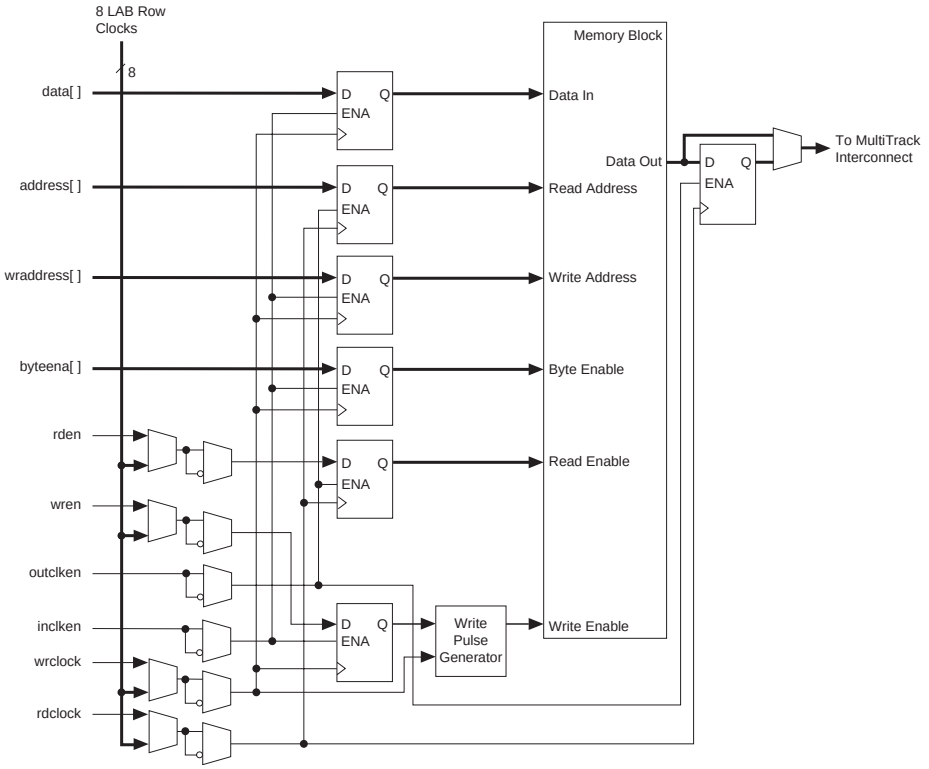


Note to Figure 4–25:
(1) All registers shown except the `rden` register have asynchronous clear ports.

Read/Write Clock Mode

The memory blocks implement read/write clock mode for simple dual-port memory. You can use up to two clocks in this mode. The write clock controls the block's data inputs, `wraddress`, and `wren`. The read clock controls the data output, `rdaddress`, and `rden`. The memory blocks support independent clock enables for each clock and asynchronous clear signals for the read- and write-side registers. Figure 4–26 shows a memory block in read/write clock mode.

Figure 4–26. Read/Write Clock Mode in Simple Dual-Port Mode *Note (1)*

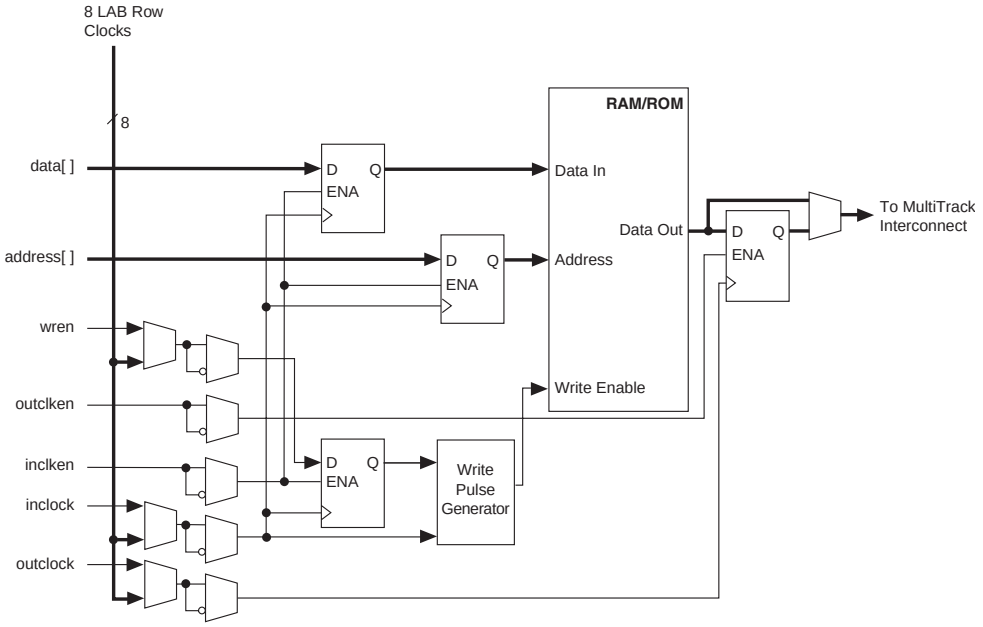


Note to Figure 4–26:
(1) All registers shown except the `rden` register have asynchronous clear ports.

Single-Port Mode

The memory blocks also support single-port mode, used when simultaneous reads and writes are not required. See Figure 4–27. A single block in a memory block can support up to two single-port mode RAM blocks in the M4K RAM blocks if each RAM block is less than or equal to 2K bits in size.

Figure 4–27. Single-Port Mode



Digital Signal Processing Block

The most commonly used DSP functions are finite impulse response (FIR) filters, complex FIR filters, infinite impulse response (IIR) filters, fast Fourier transform (FFT) functions, direct cosine transform (DCT) functions, and correlators. All of these blocks have the same fundamental building block: the multiplier. Additionally, some applications need specialized operations such as multiply-add and multiply-accumulate operations. Stratix GX devices provide DSP blocks to meet the arithmetic requirements of these functions.

Each Stratix GX device has two columns of DSP blocks to efficiently implement DSP functions faster than LE-based implementations. Larger Stratix GX devices have more DSP blocks per column (see Table 4–12). Each DSP block can be configured to support up to:

- Eight 9 × 9-bit multipliers
- Four 18 × 18-bit multipliers
- One 36 × 36-bit multiplier

As indicated, the Stratix GX DSP block can support one 36 × 36-bit multiplier in a single DSP block. This is true for any matched sign multiplications (either unsigned by unsigned or signed by signed), but

the capabilities for dynamic and mixed sign multiplications are handled differently. The following list provides the largest functions that can fit into a single DSP block.

- 36 × 36-bit unsigned by unsigned multiplication
- 36 × 36-bit signed by signed multiplication
- 35 × 36-bit unsigned by signed multiplication
- 36 × 35-bit signed by unsigned multiplication
- 36 × 35-bit signed by dynamic sign multiplication
- 35 × 36-bit dynamic sign by signed multiplication
- 35 × 36-bit unsigned by dynamic sign multiplication
- 36 × 35-bit dynamic sign by unsigned multiplication
- 35 × 35-bit dynamic sign multiplication when the sign controls for each operand are different
- 36 × 36-bit dynamic sign multiplication when the same sign control is used for both operands



This list only shows functions that can fit into a single DSP block. Multiple DSP blocks can support larger multiplication functions.

Figure 4–28 shows one of the columns with surrounding LAB rows.

Figure 4–28. DSP Blocks Arranged in Columns

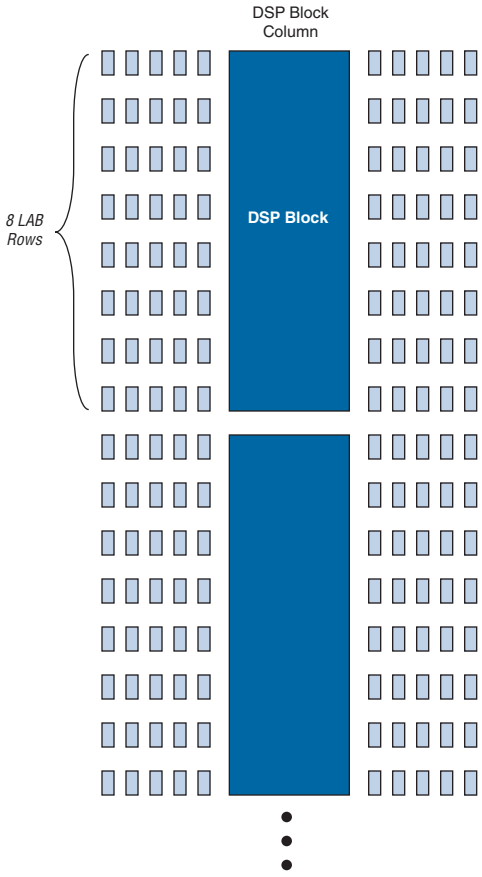


Table 4–12 shows the number of DSP blocks in each Stratix GX device.

Table 4–12. DSP Blocks in Stratix GX Devices *Notes (1), (2)*

Device	DSP Blocks	Total 9 × 9 Multipliers	Total 18 × 18 Multipliers	Total 36 × 36 Multipliers
EP1SGX10	6	48	24	6
EP1SGX25	10	80	40	10
EP1SGX40	14	112	56	14

Notes to Table 4–12:

(1) Each device has either the number of 9 × 9-, 18 × 18-, or 36 × 36-bit multipliers shown. The total number of multipliers for each device is not the sum of all the multipliers.

(2) The number of supported multiply functions shown is based on signed/signed or unsigned/unsigned implementations.

DSP block multipliers can optionally feed an adder/subtractor or accumulator within the block depending on the configuration. This makes routing to LEs easier, saves LE routing resources, and increases performance, because all connections and blocks are within the DSP block. Additionally, the DSP block input registers can efficiently implement shift registers for FIR filter applications.

Figure 4–29 shows the top-level diagram of the DSP block configured for 18 × 18-bit multiplier mode. Figure 4–30 shows the 9 × 9-bit multiplier configuration of the DSP block.

Figure 4–29. DSP Block Diagram for 18 × 18-Bit Configuration

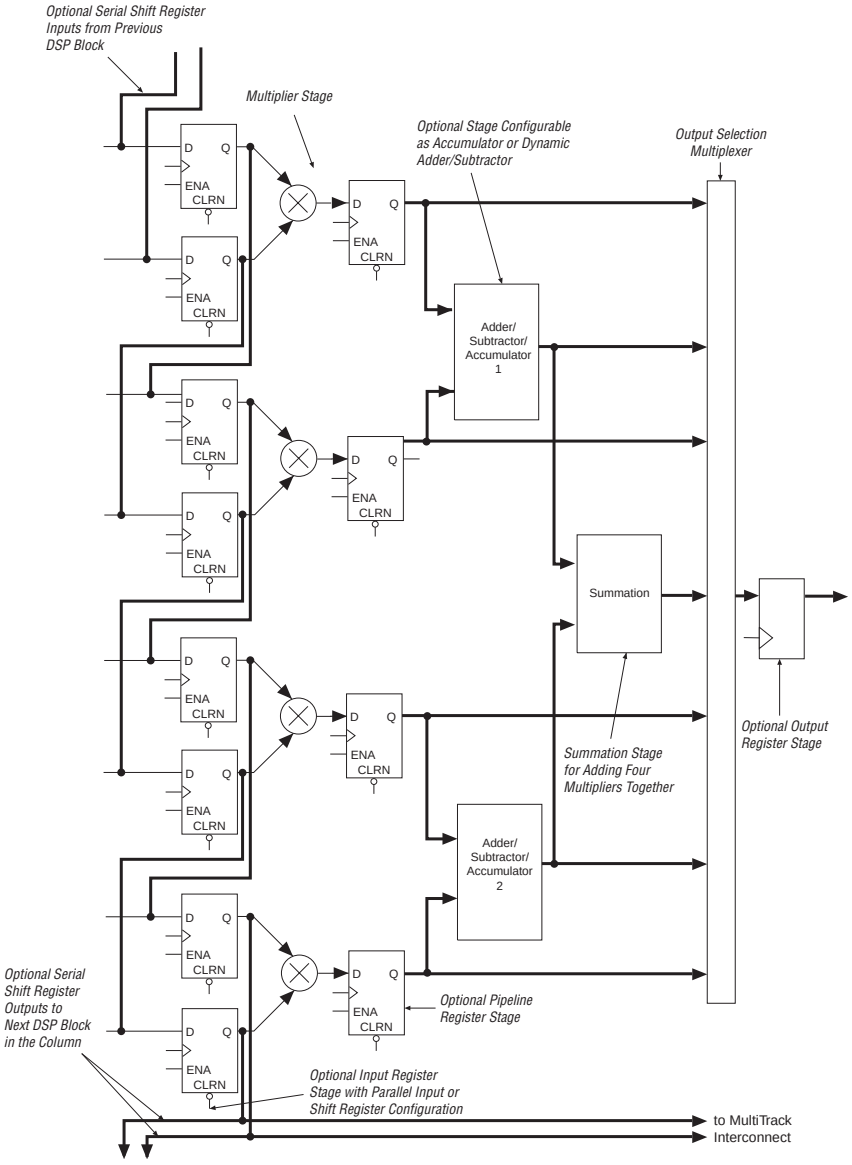
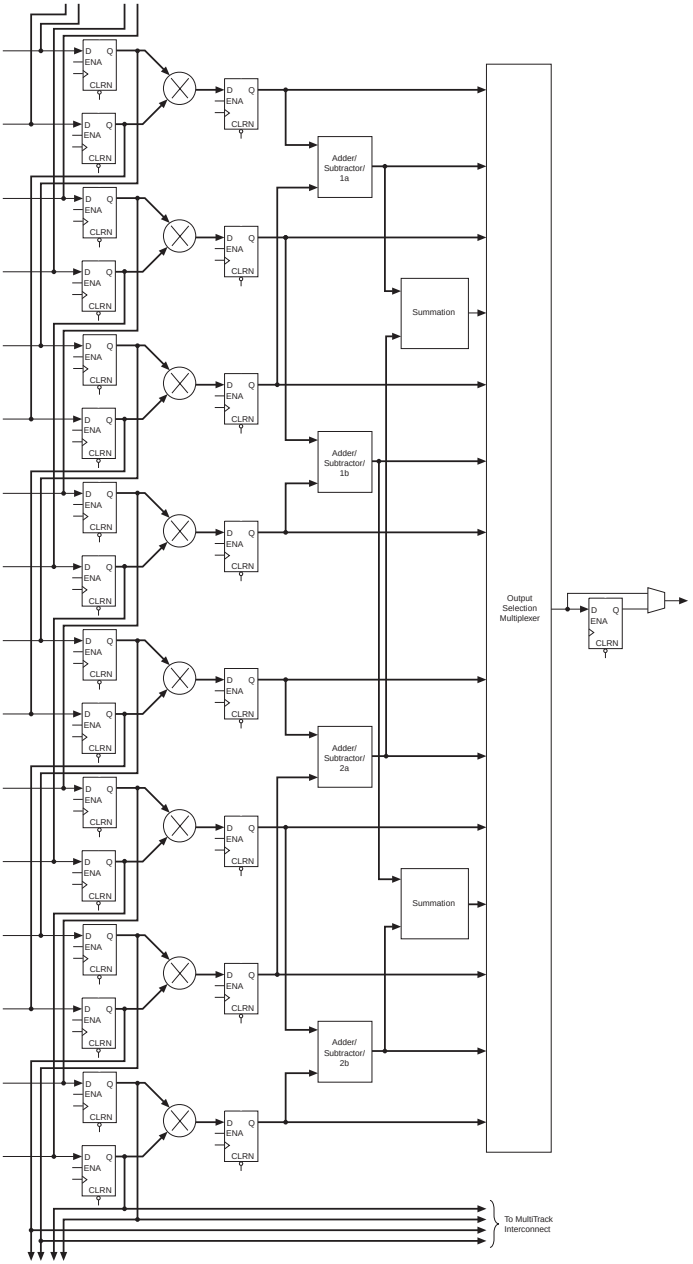


Figure 4–30. DSP Block Diagram for 9 × 9-Bit Configuration



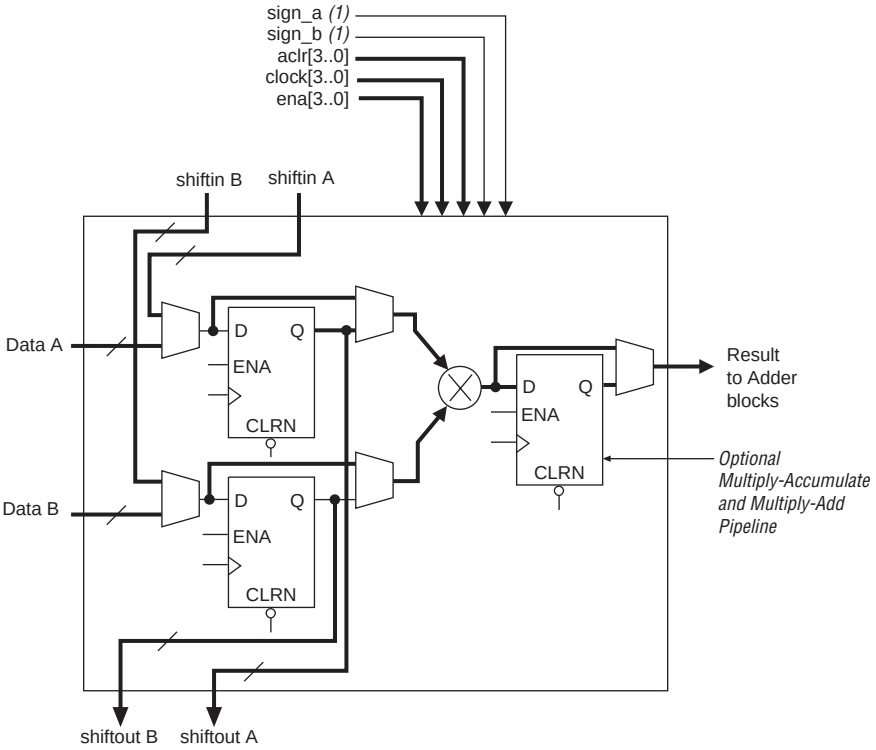
The DSP block consists of the following elements:

- Multiplier block
- Adder/output block

Multiplier Block

The DSP block multiplier block consists of the input registers, a multiplier, and pipeline register for pipelining multiply-accumulate and multiply-add/subtract functions as shown in Figure 4–31.

Figure 4–31. Multiplier Sub-Block Within Stratix GX DSP Block



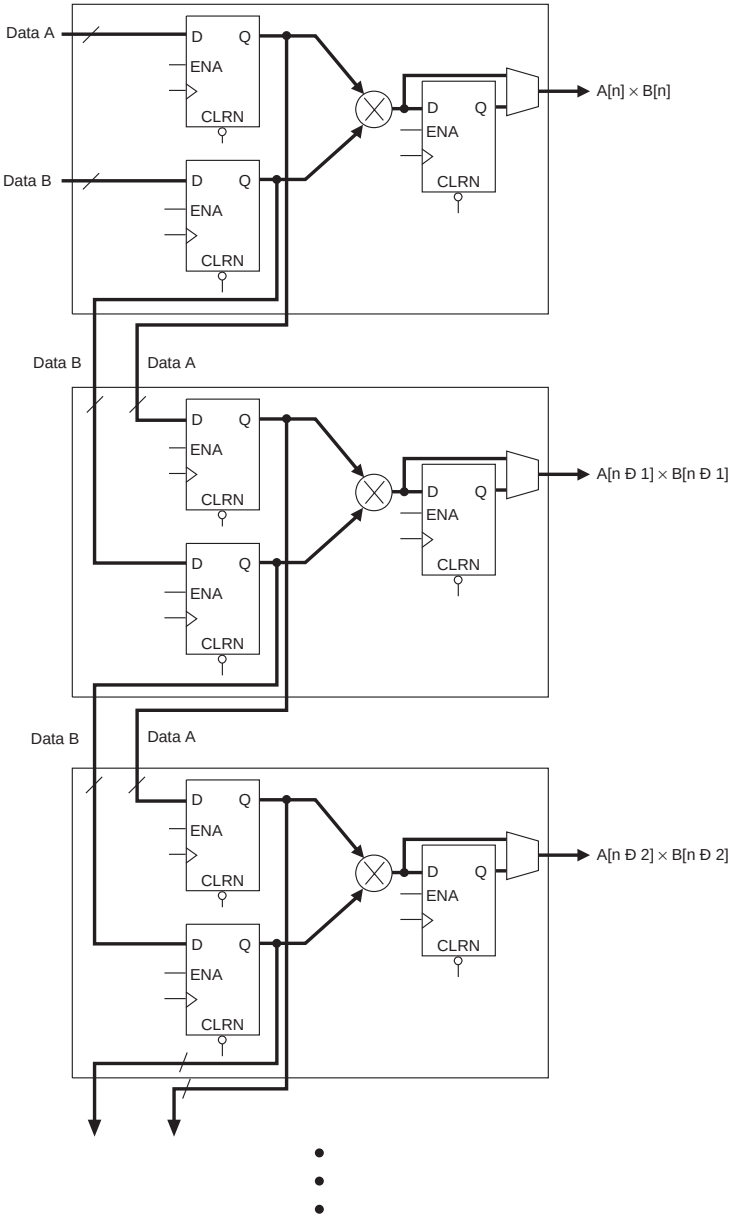
Note to Figure 4–31:
(1) These signals can be unregistered or registered once to match data path pipelines if required.

Input Registers

A bank of optional input registers is located at the input of each multiplier and multiplicand inputs to the multiplier. When these registers are configured for parallel data inputs, they are driven by regular routing resources. You can use a clock signal, asynchronous clear signal, and a clock enable signal to independently control each set of A and B inputs for each multiplier in the DSP block. You select these control signals from a set of four different `clock[3..0]`, `aclr[3..0]`, and `ena[3..0]` signals that drive the entire DSP block.

You can also configure the input registers for a shift register application. In this case, the input registers feed the multiplier and drive two dedicated shift output lines: `shiftoutA` and `shiftoutB`. The shift outputs of one multiplier block directly feed the adjacent multiplier block in the same DSP block (or the next DSP block) as shown in Figure 4–32, to form a shift register chain. This chain can terminate in any block, that is, you can create any length of shift register chain up to 224 registers. You can use the input shift registers for FIR filter applications. One set of shift inputs can provide data for a filter, and the other are coefficients that are optionally loaded in serial or parallel. When implementing 9 × 9- and 18 × 18-bit multipliers, you do not need to implement external shift registers in LAB LEs. You implement all the filter circuitry within the DSP block and its routing resources, saving LE and general routing resources for general logic. External registers are needed for shift register inputs when using 36 × 36-bit multipliers.

Figure 4–32. Multiplier Sub-Blocks Using Input Shift Register Connections *Note (1)*



Note to Figure 4–32:
(1) Either Data A or Data B input can be set to a parallel input for constant coefficient multiplication.

Table 4–13 shows the summary of input register modes for the DSP block.

Table 4–13. Input Register Modes

Register Input Mode	9 × 9	18 × 18	36 × 36
Parallel input	✓	✓	✓
Shift register input	✓	✓	

Multiplier

The multiplier supports 9 × 9-, 18 × 18-, or 36 × 36-bit multiplication. Each DSP block supports eight possible 9 × 9-bit or smaller multipliers. There are four multiplier blocks available for multipliers larger than 9 × 9 bits but smaller than 18 × 18 bits. There is one multiplier block available for multipliers larger than 18 × 18 bits but smaller than or equal to 36 × 36 bits. The ability to have several small multipliers is useful in applications such as video processing. Large multipliers greater than 18 × 18 bits are useful for applications such as the mantissa multiplication of a single-precision floating-point number.

The multiplier operands can be signed or unsigned numbers, where the result is signed if either input is signed as shown in Table 4–14. The `sign_a` and `sign_b` signals provide dynamic control of each operand's representation: a logic 1 indicates the operand is a signed number, a logic 0 indicates the operand is an unsigned number. These sign signals affect all multipliers and adders within a single DSP block and you can register them to match the data path pipeline. The multipliers are full precision (that is, 18 bits for the 18-bit multiply, 36-bits for the 36-bit multiply, and so on), regardless of whether `sign_a` or `sign_b` set the operands as signed or unsigned numbers.

Table 4–14. Multiplier Signed Representation

Data A	Data B	Result
Unsigned	Unsigned	Unsigned
Unsigned	Signed	Signed
Signed	Unsigned	Signed
Signed	Signed	Signed

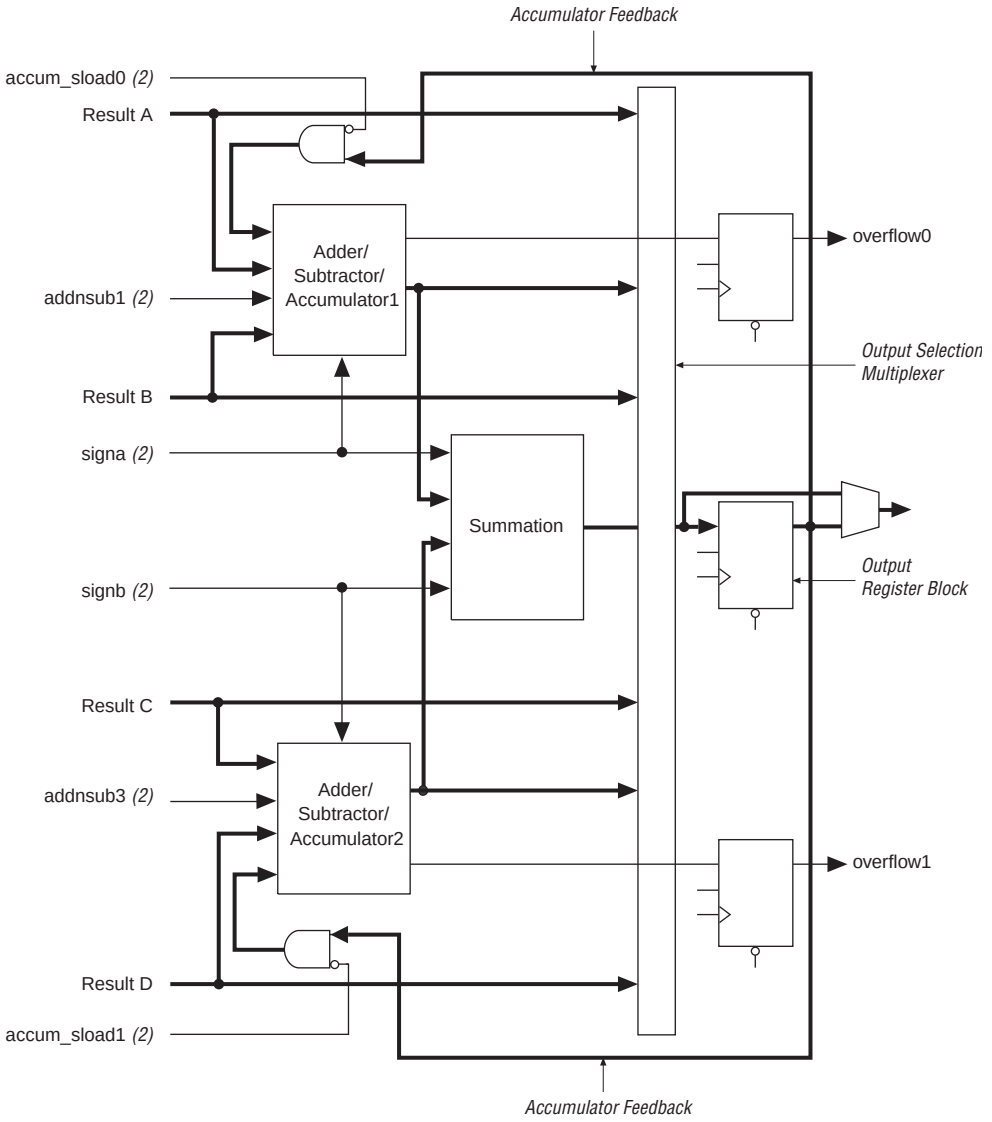
Pipeline/Post Multiply Register

The output of 9 × 9- or 18 × 18-bit multipliers can optionally feed a register to pipeline multiply-accumulate and multiply-add/subtract functions. For 36 × 36-bit multipliers, this register pipelines the multiplier function.

Adder/Output Blocks

The result of the multiplier sub-blocks are sent to the adder/output block which consist of an adder/subtractor/accumulator unit, summation unit, output select multiplexer, and output registers. The results are used to configure the adder/output block as a pure output, accumulator, a simple two-multiplier adder, four-multiplier adder, or final stage of the 36-bit multiplier. You can configure the adder/output block to use output registers in any mode, and must use output registers for the accumulator. The system cannot use adder/output blocks independently of the multiplier. Figure 4–33 shows the adder and output stages.

Figure 4–33. Adder/Output Blocks *Note (1)*



Notes to Figure 4–33:

(1) Adder/output block shown in Figure 4–33 is in 18 × 18-bit mode. In 9 × 9-bit mode, there are four adder/subtractor blocks and two summation blocks.

(2) These signals are either not registered, registered once, or registered twice to match the data path pipeline.

Adder/Subtractor/Accumulator

The adder/subtractor/accumulator is the first level of the adder/output block and can be used as an accumulator or as an adder/subtractor.

Adder/Subtractor
Each adder/subtractor/accumulator block can perform addition or subtraction using the `addnsub` independent control signal for each first-level adder in 18 × 18-bit mode. There are two `addnsub[1..0]` signals available in a DSP block for any configuration. For 9 × 9-bit mode, one `addnsub[1..0]` signal controls the top two one-level adders and another `addnsub[1..0]` signal controls the bottom two one-level adders. A high `addnsub` signal indicates addition, and a low signal indicates subtraction. The `addnsub` control signal can be unregistered or registered once or twice when feeding the adder blocks to match data path pipelines.

The `signa` and `signb` signals serve the same function as the multiplier block `signa` and `signb` signals. The only difference is that these signals can be registered up to two times. These signals are tied to the same `signa` and `signb` signals from the multiplier and must be connected to the same clocks and control signals.

Accumulator
When configured for accumulation, the adder/output block output feeds back to the accumulator as shown in Figure 4–33. The `accum_sload[1..0]` signal synchronously loads the multiplier result to the accumulator output. This signal can be unregistered or registered once or twice. Additionally, the `overflow` signal indicates the accumulator has overflowed or underflowed in accumulation mode. This signal is always registered and must be externally latched in LEs if the design requires a latched `overflow` signal.

Summation

The output of the adder/subtractor/accumulator block feeds to an optional summation block. This block sums the outputs of the DSP block multipliers. In 9 × 9-bit mode, there are two summation blocks providing the sums of two sets of four 9 × 9-bit multipliers. In 18 × 18-bit mode, there is one summation providing the sum of one set of four 18 × 18-bit multipliers.

Output Selection Multiplexer

The outputs from the various elements of the adder/output block are routed through an output selection multiplexer. Based on the DSP block operational mode and user settings, the multiplexer selects whether the output from the multiplier, the adder/subtractor/accumulator, or summation block feeds to the output.

Output Registers

Optional output registers for the DSP block outputs are controlled by four sets of control signals: `clock[3..0]`, `aclr[3..0]`, and `ena[3..0]`. Output registers can be used in any mode.

Modes of Operation

The adder, subtractor, and accumulate functions of a DSP block have four modes of operation:

- Simple multiplier
- Multiply-accumulator
- Two-multipliers adder
- Four-multipliers adder

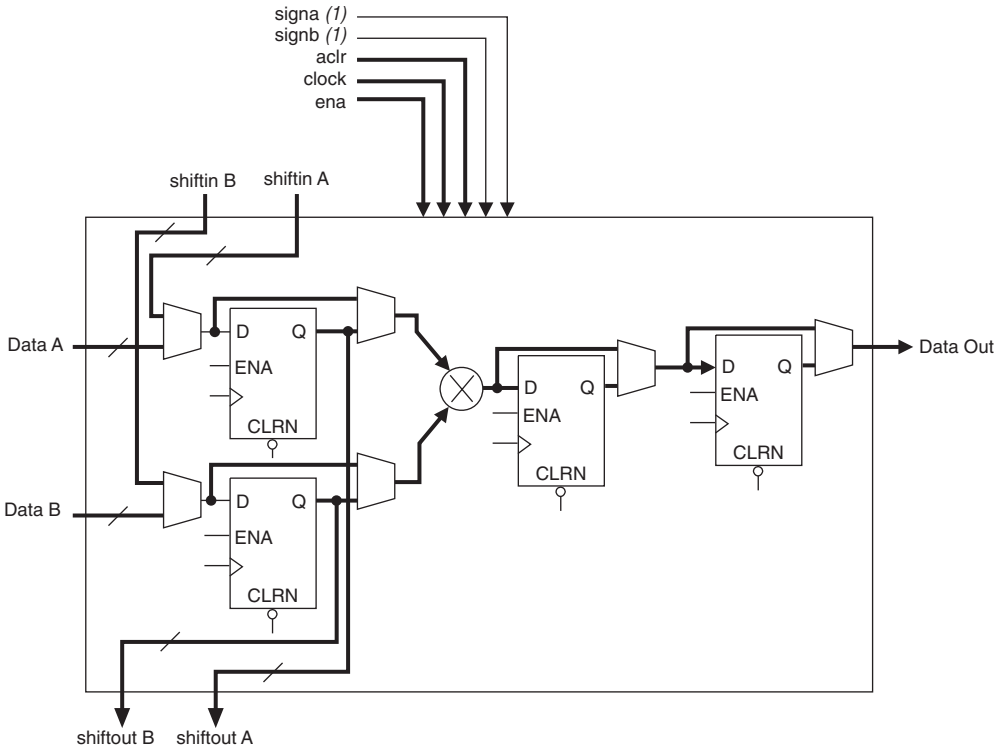


Each DSP block can only support one mode. Mixed modes in the same DSP block is not supported.

Simple Multiplier Mode

In simple multiplier mode, the DSP block drives the multiplier sub-block result directly to the output with or without an output register. Up to four 18 × 18-bit multipliers or eight 9 × 9-bit multipliers can drive their results directly out of one DSP block. See Figure 4–34.

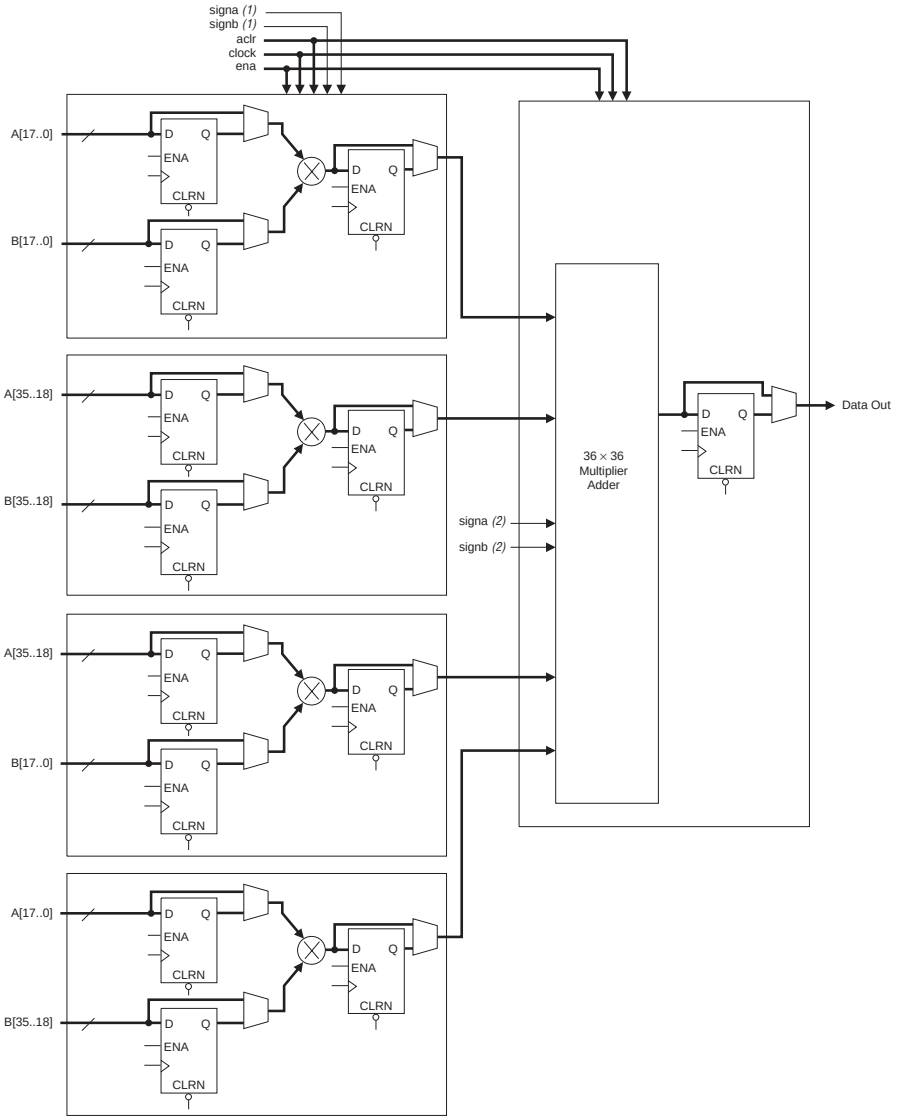
Figure 4–34. Simple Multiplier Mode



Note to Figure 4–34:
(1) These signals are not registered or registered once to match the data path pipeline.

DSP blocks can also implement one 36 × 36-bit multiplier in multiplier mode. DSP blocks use four 18 × 18-bit multipliers combined with dedicated adder and internal shift circuitry to achieve 36-bit multiplication. The input shift register feature is not available for the 36 × 36-bit multiplier. In 36 × 36-bit mode, the device can use the register that is normally a multiplier-result-output register as a pipeline stage for the 36 × 36-bit multiplier. Figure 4–35 shows the 36 × 36-bit multiply mode.

Figure 4–35. 36 × 36 Multiply Mode



Notes to Figure 4–35:

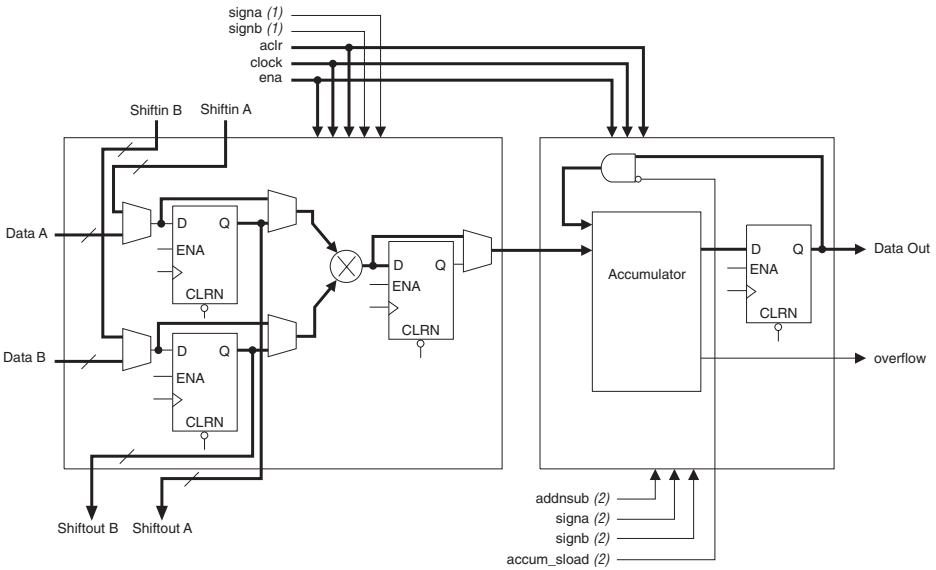
(1) These signals are not registered or registered once to match the pipeline.

(2) These signals are not registered, registered once, or registered twice for latency to match the pipeline.

Multiply-Accumulator Mode

In multiply-accumulator mode (see Figure 4–36), the DSP block drives multiplied results to the adder/subtractor/accumulator block configured as an accumulator. You can implement one or two multiply-accumulators up to 18 × 18 bits in one DSP block. The first and third multiplier sub-blocks are unused in this mode, since only one multiplier can feed one of two accumulators. The multiply-accumulator output can be up to 52 bits—a maximum of a 36-bit result with 16 bits of accumulation. The `accum_sload` and `overflow` signals are only available in this mode. The `addnsub` signal can set the accumulator for decimation and the `overflow` signal indicates underflow condition.

Figure 4–36. Multiply-Accumulate Mode



Notes to Figure 4–36:
(1) These signals are not registered or registered once to match the data path pipeline.
(2) These signals are not registered, registered once, or registered twice for latency to match the data path pipeline.

Two-Multipliers Adder Mode

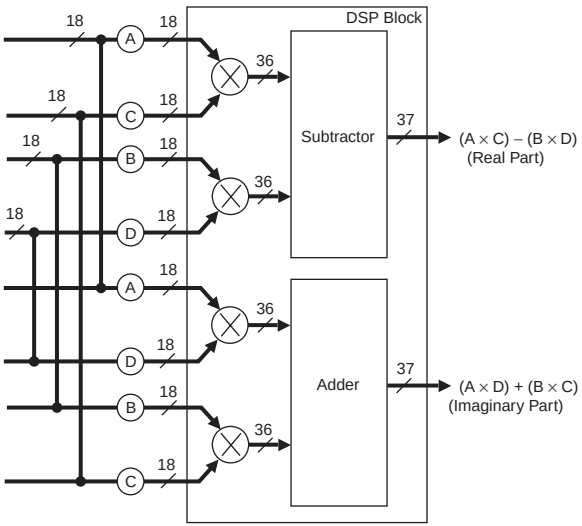
The two-multipliers adder mode uses the adder/subtractor/accumulator block to add or subtract the outputs of the multiplier block, which is useful for applications such as FFT functions and complex FIR filters. A single DSP block can implement two sums or differences from two 18 × 18-bit multipliers each or four sums or differences from two 9 × 9-bit multipliers each.

You can use the two-multipliers adder mode for complex multiplications, which are written as:

$$(a + jb) \times (c + jd) = [(a \times c) - (b \times d)] + j \times [(a \times d) + (b \times c)]$$

The two-multipliers adder mode allows a single DSP block to calculate the real part $[(a \times c) - (b \times d)]$ using one subtractor and the imaginary part $[(a \times d) + (b \times c)]$ using one adder, for data widths up to 18 bits. Two complex multiplications are possible for data widths up to 9 bits using four adder/subtractor/accumulator blocks. Figure 4–37 shows an 18-bit two-multipliers adder.

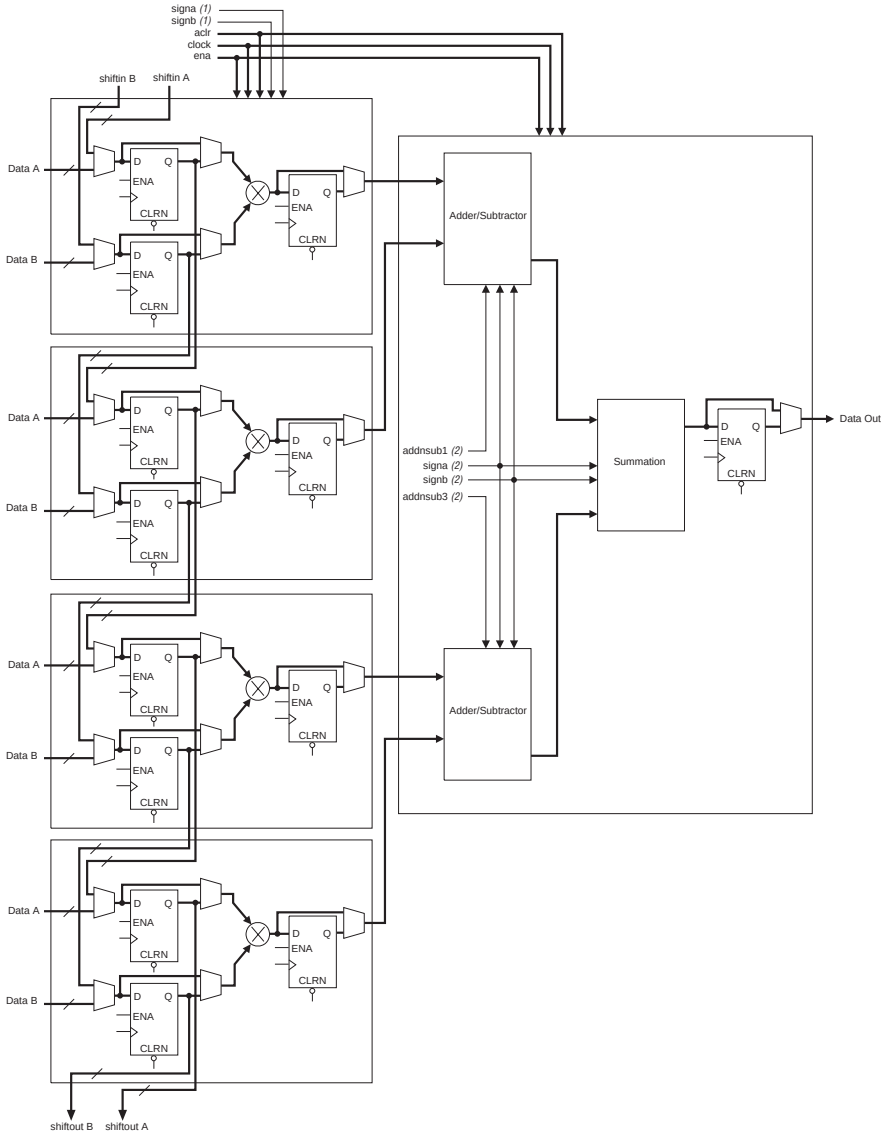
Figure 4–37. Two-Multipliers Adder Mode Implementing Complex Multiply



Four-Multipliers Adder Mode

In the four-multipliers adder mode, the DSP block adds the results of two first -stage adder/subtractor blocks. One sum of four 18 × 18-bit multipliers or two different sums of two sets of four 9 × 9-bit multipliers can be implemented in a single DSP block. The product width for each multiplier must be the same size. The four-multipliers adder mode is useful for FIR filter applications. Figure 4–38 shows the four multipliers adder mode.

Figure 4–38. Four-Multipliers Adder Mode



Notes to Figure 4–38:

(1) These signals are not registered or registered once to match the data path pipeline.

(2) These signals are not registered, registered once, or registered twice for latency to match the data path pipeline.

For FIR filters, the DSP block combines the four-multipliers adder mode with the shift register inputs. One set of shift inputs contains the filter data, while the other holds the coefficients loaded in serial or parallel. The input shift register eliminates the need for shift registers external to the DSP block (that is, implemented in LEs). This architecture simplifies filter design since the DSP block implements all of the filter circuitry.

One DSP block can implement an entire 18-bit FIR filter with up to four taps. For FIR filters larger than four taps, DSP blocks can be cascaded with additional adder stages implemented in LEs.

Table 4–15 shows the different number of multipliers possible in each DSP block mode according to size. These modes allow the DSP blocks to implement numerous applications for DSP including FFTs, complex FIR, FIR, and 2D FIR filters, equalizers, IIR, correlators, matrix multiplication and many other functions.

Table 4–15. Multiplier Size & Configurations per DSP block

DSP Block Mode	9 × 9	18 × 18	36 × 36 *(1)*
Multiplier	Eight multipliers with eight product outputs	Four multipliers with four product outputs	One multiplier with one product output
Multiply-accumulator	Two multiply and accumulate (52 bits)	Two multiply and accumulate (52 bits)	–
Two-multipliers adder	Four sums of two multiplier products each	Two sums of two multiplier products each	–
Four-multipliers adder	Two sums of four multiplier products each	One sum of four multiplier products each	–

Note to Table 4–15:
(1) The number of supported multiply functions shown is based on signed/signed or unsigned/unsigned implementations.

DSP Block Interface

Stratix GX device DSP block outputs can cascade down within the same DSP block column. Dedicated connections between DSP blocks provide fast connections between the shift register inputs to cascade the shift register chains. You can cascade DSP blocks for 9 × 9- or 18 × 18-bit FIR filters larger than four taps, with additional adder stages implemented in LEs. If the DSP block is configured as 36 × 36 bits, the adder, subtractor, or accumulator stages are implemented in LEs. Each DSP block can route the shift register chain out of the block to cascade two full columns of DSP blocks.

The DSP block is divided into eight block units that interface with eight LAB rows on the left and right. Each block unit can be considered half of an 18 × 18-bit multiplier sub-block with 18 inputs and 18 outputs. A local interconnect region is associated with each DSP block. Like an LAB, this interconnect region can be fed with 10 direct link interconnects from the LAB to the left or right of the DSP block in the same row. All row and column routing resources can access the DSP block's local interconnect region. The outputs also work similarly to LAB outputs as well. Nine outputs from the DSP block can drive to the left LAB through direct link interconnects and nine can drive to the right LAB though direct link interconnects. All 18 outputs can drive to all types of row and column routing. Outputs can drive right- or left-column routing. Figures 4–39 and 4–40 show the DSP block interfaces to LAB rows.

Figure 4–39. DSP Block Interconnect Interface

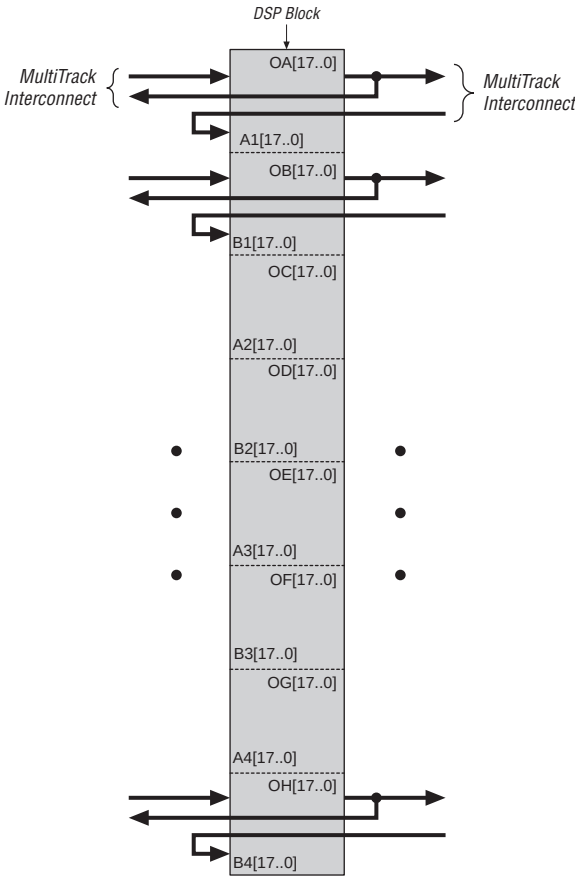
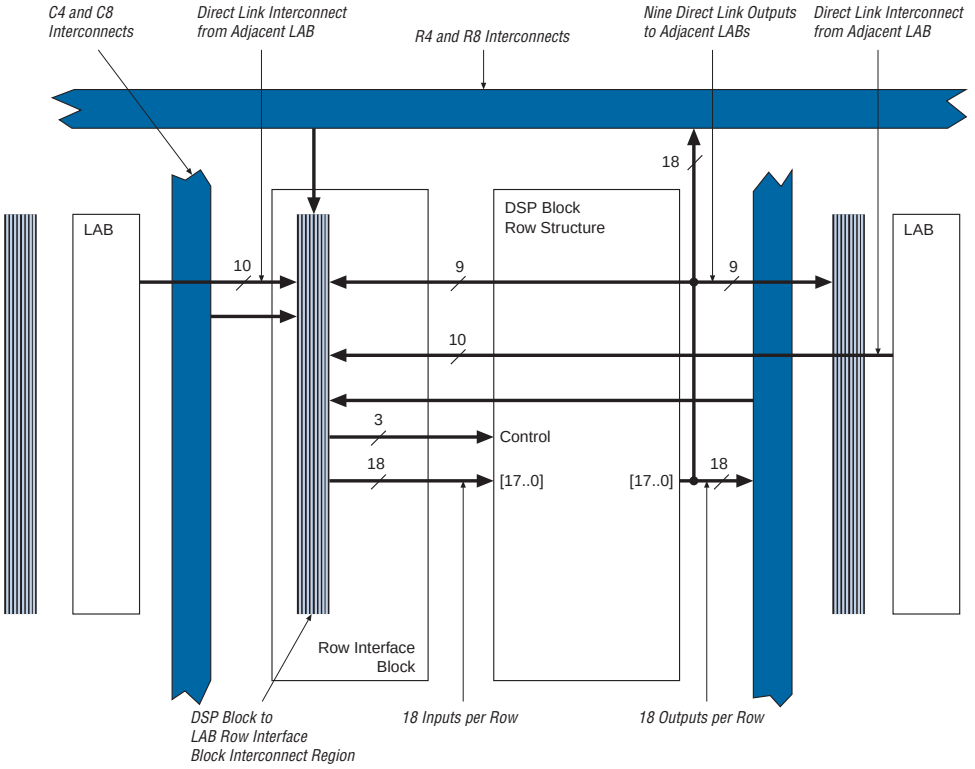


Figure 4–40. DSP Block Interface to Interconnect



A bus of 18 control signals feeds the entire DSP block. These signals include `clock[0..3]` clocks, `aclr[0..3]` asynchronous clears, `ena[1..4]` clock enables, `signa`, `signb` signed/unsigned control signals, `addnsub1` and `addnsub3` addition and subtraction control signals, and `accum_sload[0..1]` accumulator synchronous loads. The

clock signals are routed from LAB row clocks and are generated from specific LAB rows at the DSP block interface. The LAB row source for control signals, data inputs, and outputs is shown in Table 4–16.

Table 4–16. DSP Block Signal Sources & Destinations

LAB Row at Interface	Control Signals Generated	Data Inputs	Data Outputs
1	signa	A1[17..0]	OA[17..0]
2	aclr0 accum_sload0	B1[17..0]	OB[17..0]
3	addnsub1 clock0 ena0	A2[17..0]	OC[17..0]
4	aclr1 clock1 ena1	B2[17..0]	OD[17..0]
5	aclr2 clock2 ena2	A3[17..0]	OE[17..0]
6	sign_b clock3 ena3	B3[17..0]	OF[17..0]
7	clear3 accum_sload1	A4[17..0]	OG[17..0]
8	addnsub3	B4[17..0]	OH[17..0]

PLLs & Clock Networks

Stratix GX devices provide a hierarchical clock structure and multiple PLLs with advanced features. The large number of clocking resources in combination with the clock synthesis precision provided by enhanced and fast PLLs provides a complete clock management solution. Stratix GX devices contain up to four enhanced PLLs and up to four fast PLLs. In addition, there are four receiver PLLs and one transmitter PLL per transceiver block located on the right side of Stratix GX devices.

Global & Hierarchical Clocking

Stratix GX devices provide 16 dedicated global clock networks, 16 regional clock networks (four per device quadrant), 8 dedicated fast regional clock networks within EP1SGX10 and EP1SGX25, and 16 dedicated fast regional clock networks within EP1SGX40 devices.

These clocks are organized into a hierarchical clock structure that allows for up to 22 clocks per device region with low skew and delay. This hierarchical clocking scheme provides up to 40 unique clock domains within EP1SGX10 and EP1SGX25 devices, and 48 unique clock domains within EP1SGX40 devices.

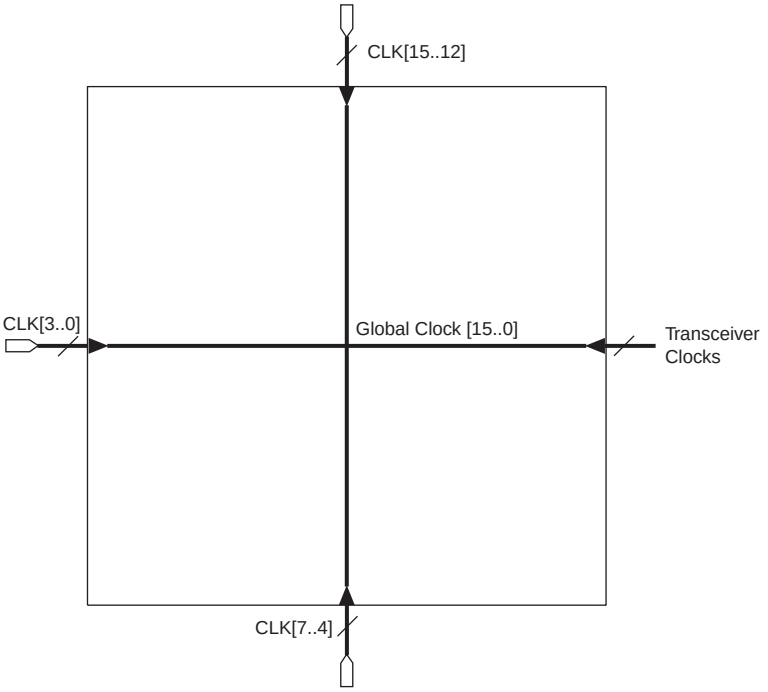
There are 12 dedicated clock pins (CLK[15..12], and CLK[7..0]) to drive either the global or regional clock networks. Three clock pins drive the top, bottom, and left side of the device. Enhanced and fast PLL outputs as well as an I/O interface can also drive these global and regional clock networks.

There are up to 20 recovered clocks (rxclkout[20..0]) and up to 5 transmitter clock outputs (coreclk_out) which can drive any of the global clock networks (CLK[15..0]), as shown in Figure 4–41.

Global Clock Network

These clocks drive throughout the entire device, feeding all device quadrants. The global clock networks can be used as clock sources for all resources within the device IOEs, LEs, DSP blocks, and all memory blocks. These resources can also be used for control signals, such as clock enables and synchronous or asynchronous clears fed from the external pin. The global clock networks can also be driven by internal logic for internally generated global clocks and asynchronous clears, clock enables, or other control signals with large fanout. Figure 4–41 shows the 12 dedicated CLK pins and the transceiver clocks driving global clock networks.

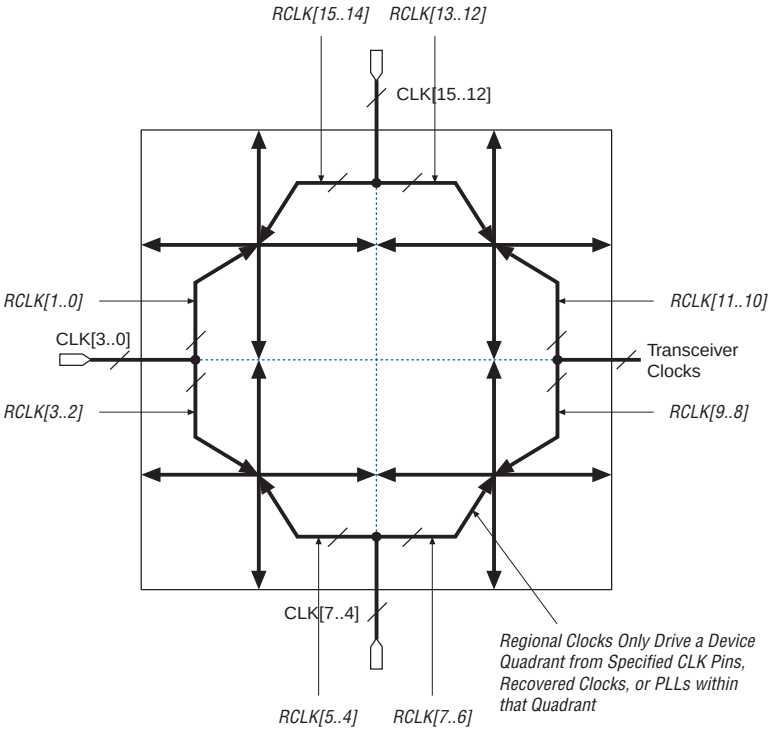
Figure 4–41. Global Clock Resources



Regional Clock Network

There are four regional clock networks RCLK[3..0] within each quadrant of the Stratix GX device that are driven by the same dedicated CLK[7..0] and CLK[15..12] input pins, PLL outputs, or transceiver clocks. The regional clock networks only pertain to the quadrant they drive into. The regional clock networks provide the lowest clock delay and skew for logic contained within a single quadrant. The CLK clock pins symmetrically drive the RCLK networks within a particular quadrant, as shown in Figure 4–42.

Figure 4–42. Regional Clocks



Fast Regional Clock Network

In EP1SGX25 and EP1SGX10 devices, there are two fast regional clock networks, FCLK[1..0], within each quadrant, fed by input pins (see Figure 4–43). In EP1SGX40 devices, there are two fast regional clock networks within each half-quadrant (see Figure 4–44). The FCLK[1..0] clocks can also be used for high fanout control signals, such as asynchronous clears, presets, clock enables, or protocol control signals such as TRDY and IRDY for PCI. Dual-purpose FCLK pins drive the fast clock networks. All devices have eight FCLK pins to drive fast regional clock networks. Any I/O pin can drive a clock or control signal onto any fast regional clock network with the addition of a delay. The I/O interconnect drives this signal.

Figure 4–43. EP1SGX25 & EP1SGX10 Device Fast Clock Pin Connections to Fast Regional Clocks

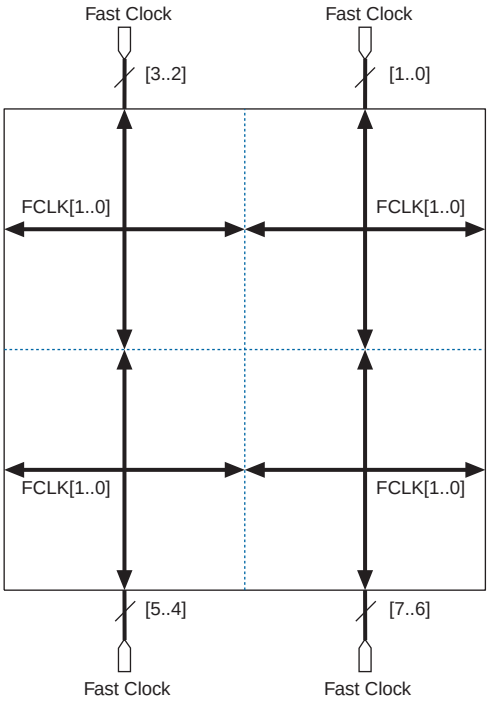
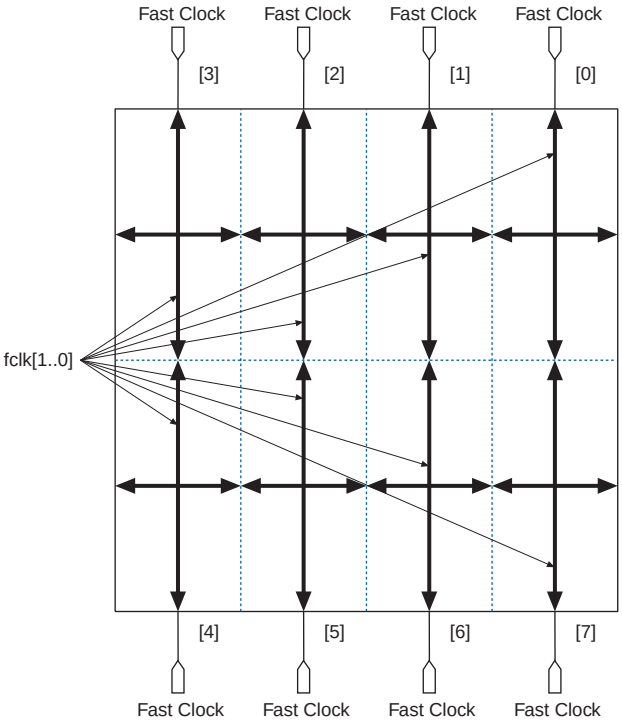


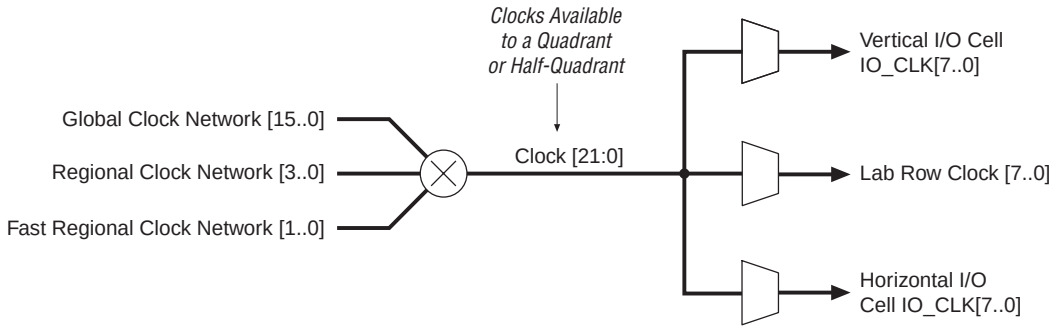
Figure 4–44. EP1SGX40 Device Fast Regional Clock Pin Connections to Fast Regional Clocks



Combined Resources

Within each region, there are 22 distinct dedicated clocking resources consisting of 16 global clock lines, 4 regional clock lines, and 2 fast regional clock lines. Multiplexers are used with these clocks to form 8-bit busses to drive LAB row clocks, column IOE clocks, or row IOE clocks. Another multiplexer at the LAB level selects two of the eight row clocks to feed the LE registers within the LAB. See Figure 4–45.

Figure 4–45. Regional Clock Bus



IOE clocks have horizontal and vertical block regions that are clocked by eight I/O clock signals chosen from the 22-quadrant or half-quadrant clock resources. Figures 4–46 and 4–47 show the quadrant and half-quadrant relationship to the I/O clock regions, respectively. The vertical regions (column pins) have less clock delay than the horizontal regions (row pins).

Figure 4–46. EP1SGX25 & EP1SGX10 Device I/O Clock Groups

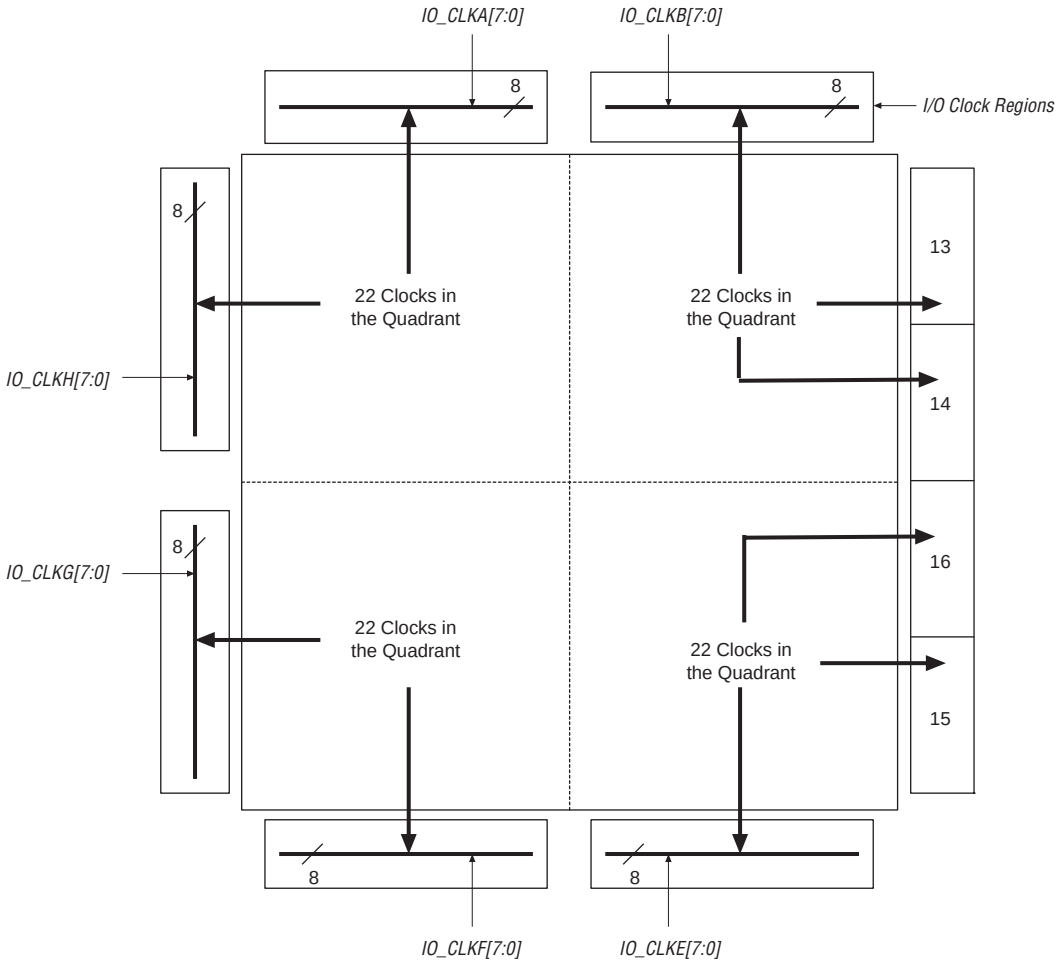
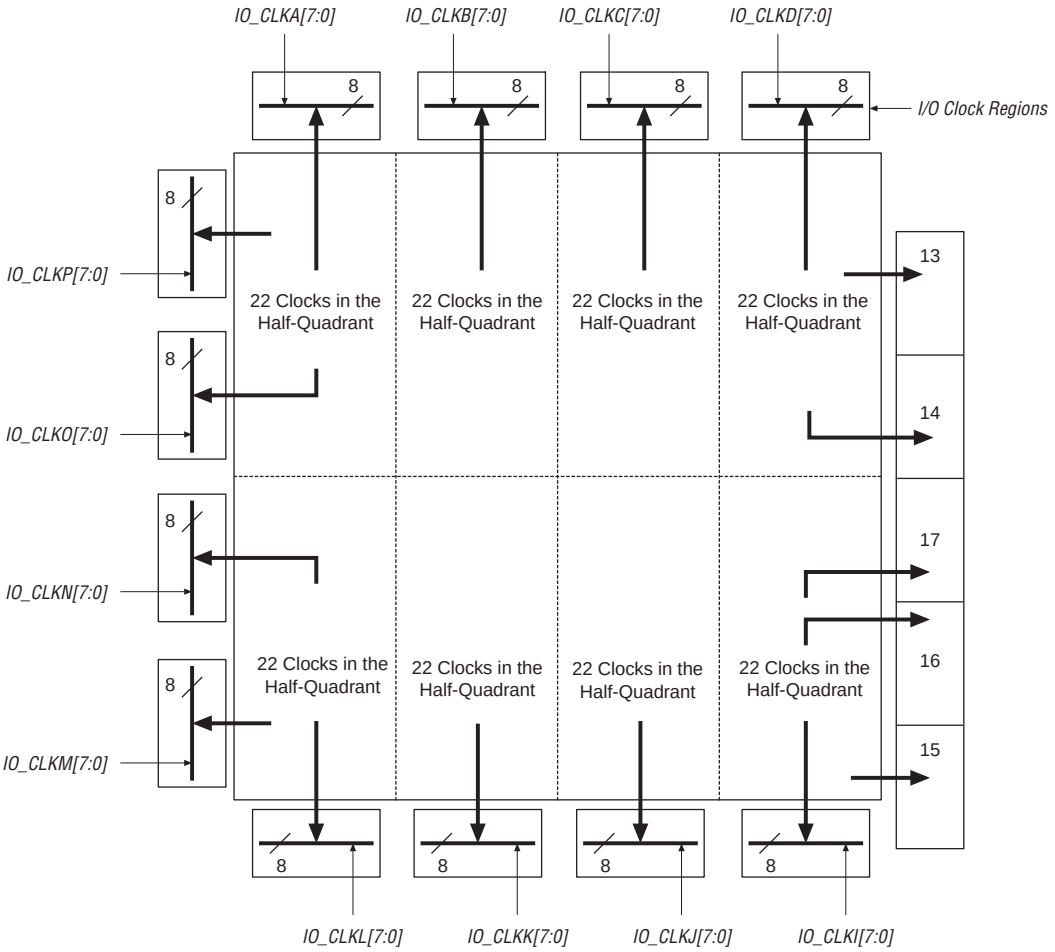


Figure 4–47. EP1SGX40 Device I/O Clock Groups



You can use the Quartus II software to control whether a clock input pin is either global, regional, or fast regional. The Quartus II software automatically selects the clocking resources if not specified.

Enhanced & Fast PLLs

Stratix GX devices provide robust clock management and synthesis using up to four enhanced PLLs and four fast PLLs. These PLLs increase performance and provide advanced clock interfacing and clock frequency synthesis. With features such as clock switchover, spread spectrum

clocking, programmable bandwidth, phase and delay control, and dynamic PLL reconfiguration, the Stratix GX device's enhanced PLLs provide you with complete control of your clocks and system timing. The fast PLLs provide general purpose clocking with multiplication and phase shifting as well as high-speed outputs for high-speed differential I/O support. Enhanced and fast PLLs work together with the Stratix GX high-speed I/O and advanced clock architecture to provide significant improvements in system performance and bandwidth.

The Quartus II software enables the PLLs and their features without requiring any external devices. Table 4–17 shows which PLLs are available for each Stratix GX device and their type. Table 4–18 shows the enhanced PLL and fast PLL features in Stratix GX devices.

Table 4–17. Stratix GX Device PLL Availability

Device	Fast PLLs								Enhanced PLLs			
	1	2	3 *(1)*	4 *(1)*	7	8	9 *(1)*	10 *(1)*	5 *(2)*	6 *(2)*	11 *(3)*	12 *(3)*
EP1SGX10	✓	✓							✓	✓		
EP1SGX25	✓	✓							✓	✓		
EP1SGX40	✓	✓			✓	✓			✓	✓	✓	✓

Notes to Table 4–17:

(1) PLLs 3, 4, 9, and 10 are not available in Stratix GX devices. However, these PLLs are listed in Table 4–17 because the Stratix GX PLL numbering scheme is consistent with Stratix devices.

(2) PLLs 5 and 6 each have eight single-ended outputs or four differential outputs.

(3) PLLs 11 and 12 each have one single-ended output.

Table 4–18. Stratix GX Enhanced PLL & Fast PLL Features (Part 1 of 2) *Notes (1)–(8)*

Feature	Enhanced PLL	Fast PLL
Clock multiplication and division	*m*/ (*n* × post-scale counter) *(1)*	*m*/(post-scale counter) *(2)*
Phase shift	Down to 156.25-ps increments *(3)*, *(4)*	Down to 125-ps increments *(3)*, *(4)*
Delay shift	250-ps increments for ±3 ns	
Clock switchover	✓	
PLL reconfiguration	✓	
Programmable bandwidth	✓	
Spread spectrum clocking	✓	
Programmable duty cycle	✓	✓
Number of internal clock outputs	6	3 *(5)*

Table 4–18. Stratix GX Enhanced PLL & Fast PLL Features (Part 2 of 2) Notes (1)–(8)		
Feature	**Enhanced PLL**	**Fast PLL**
Number of external clock outputs	Four differential/eight singled-ended or one single-ended *(6)*	*(7)*
Number of feedback clock inputs	4 *(8)*	

Notes to Table 4–18:

(1) The maximum count value is 1024, with a 50% duty cycle setting on the counter. The maximum count value for any other duty cycle setting is 512.
(2) For fast PLLs, *m* and post-scale counters range from 1 to 32.
(3) The smallest phase shift is determined by the VCO period divided by 8.
(4) For degree increments, Stratix GX devices can shift all output frequencies in increments of at least 45°. Smaller degree increments are possible depending on the frequency and divide parameters.
(5) PLLs 7 and 8 have two output ports per PLL. PLLs 1 and 2 have three output ports per PLL.
(6) Every Stratix GX device has two enhanced PLLs (PLLs 5 and 6) with eight single-ended or four differential outputs each. Two additional enhanced PLLs (PLLs 11 and 12) in EP1SGX40 devices each have one single-ended output.
(7) Fast PLLs can drive to any I/O pin as an external clock. For high-speed differential I/O pins, the device uses a data channel to generate `txclkout`.
(8) Every Stratix GX device has two enhanced PLLs with one single-ended or differential external feedback input per PLL.

Figure 4–48 shows a top-level diagram of the Stratix GX device and the PLL floorplan.

Figure 4–48. PLL Floorplan

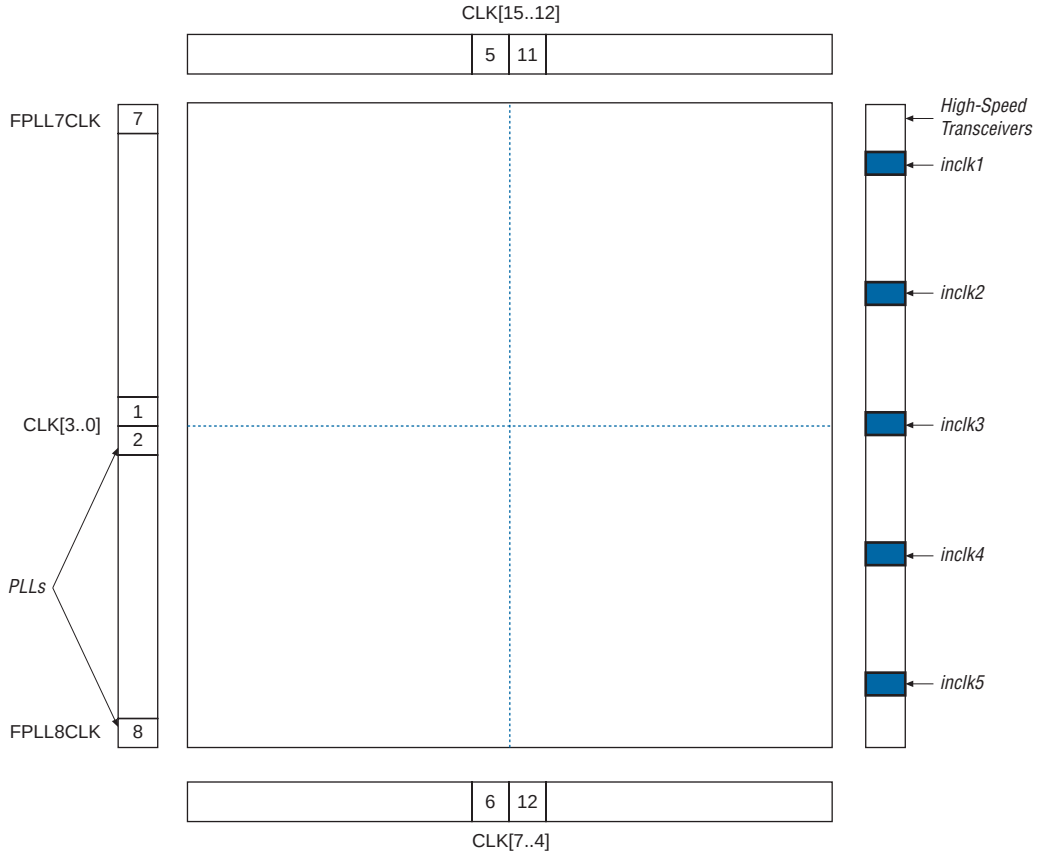
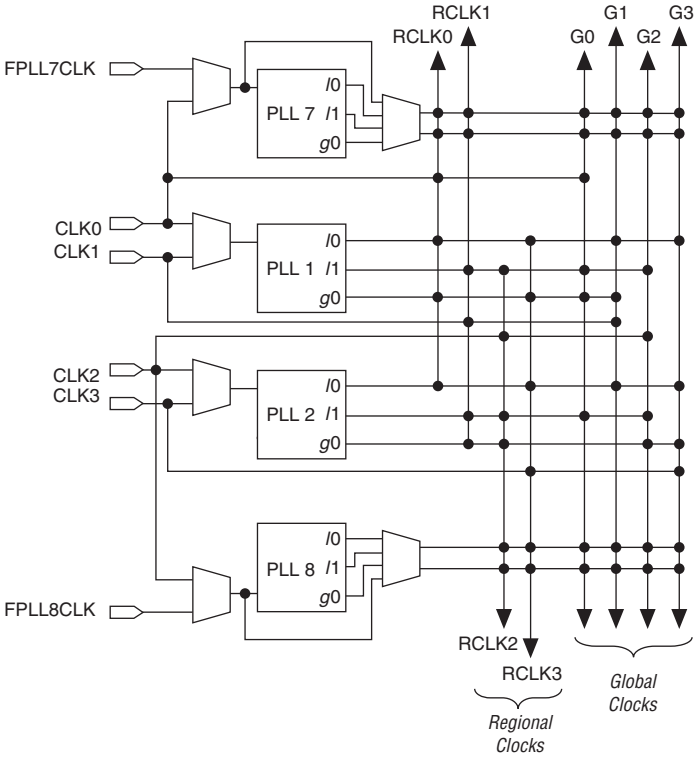


Figure 4–49 shows the global and regional clock connections from the PLL outputs and the CLK pins.

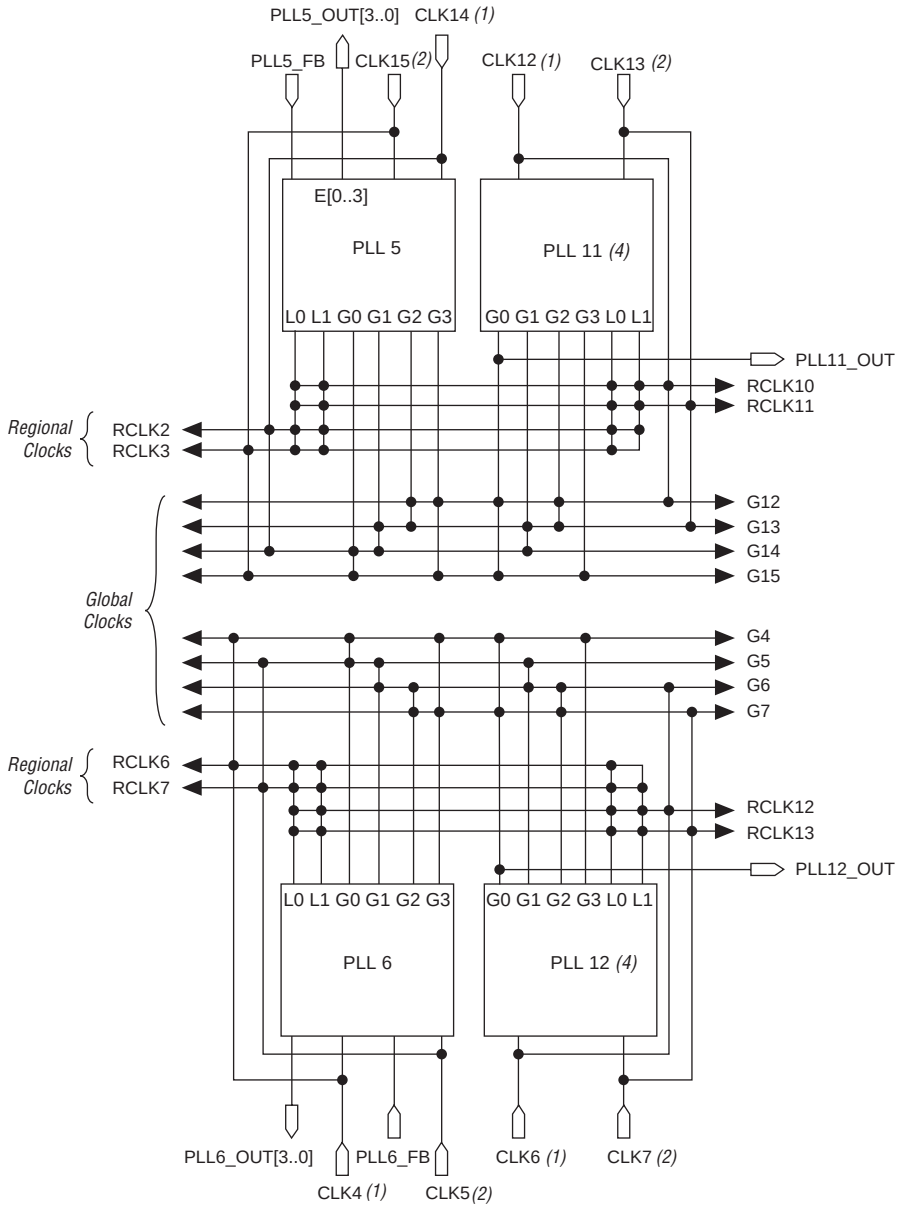
Figure 4–49. Global & Regional Clock Connections From Side Pins & Fast PLL Outputs *Note (1)*



Note to Figure 4–49:
(1) PLLs 1,2 7, and 8 are fast PLLs. PLLs 7 and 8 do not drive global clocks.

Figure 4–50 shows the global and regional clocking from enhanced PLL outputs and top CLK pins.

Figure 4–50. Global & Regional Clock Connections From Top Clock Pins & Enhanced PLL Outputs *Note (1)*

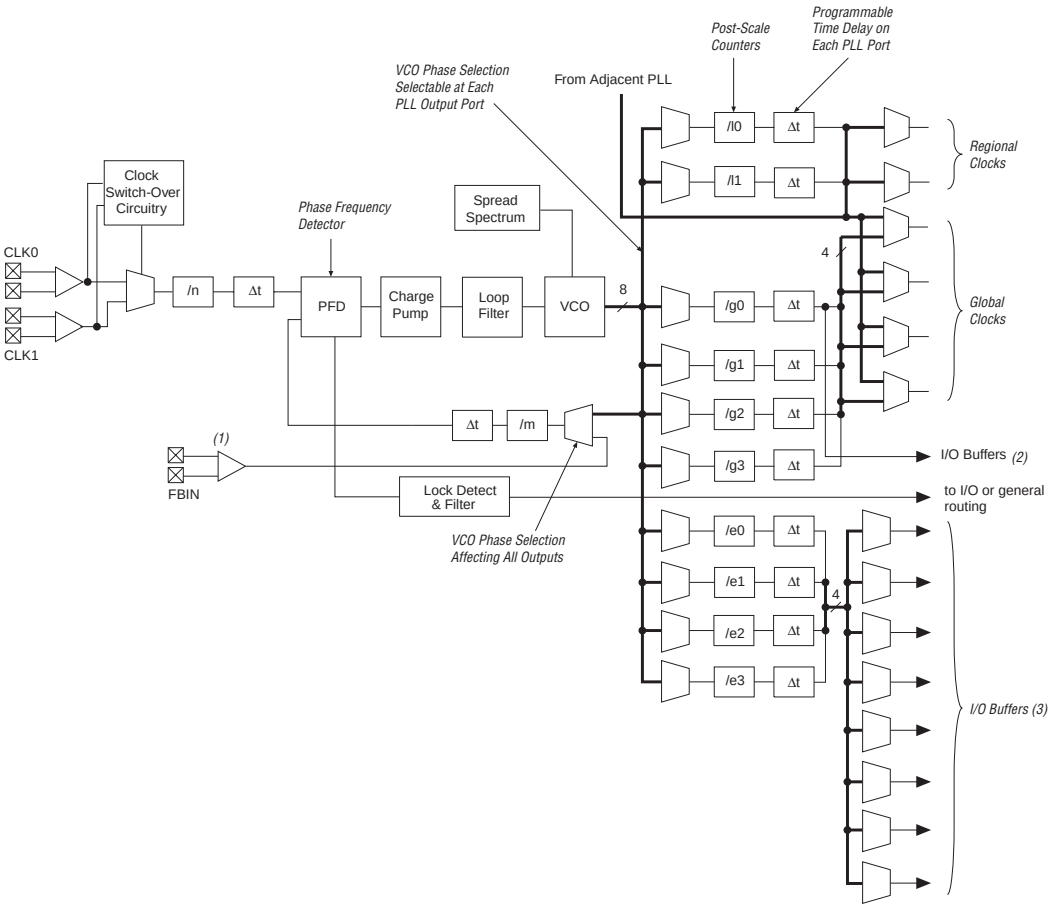


Note to Figure 4–50:
(1) PLLs 5, 6, 11, and 12 are enhanced PLLs.

Enhanced PLLs

Stratix GX devices contain up to four enhanced PLLs with advanced clock management features. Figure 4–51 shows a diagram of the enhanced PLL.

Figure 4–51. Stratix GX Enhanced PLL



Notes to Figure 4–51:

(1) External feedback is available in PLLs 5 and 6.

(2) This external output is available from the *g*0 counter for PLLs 11 and 12.

(3) These counters and external outputs are available in PLLs 5 and 6.

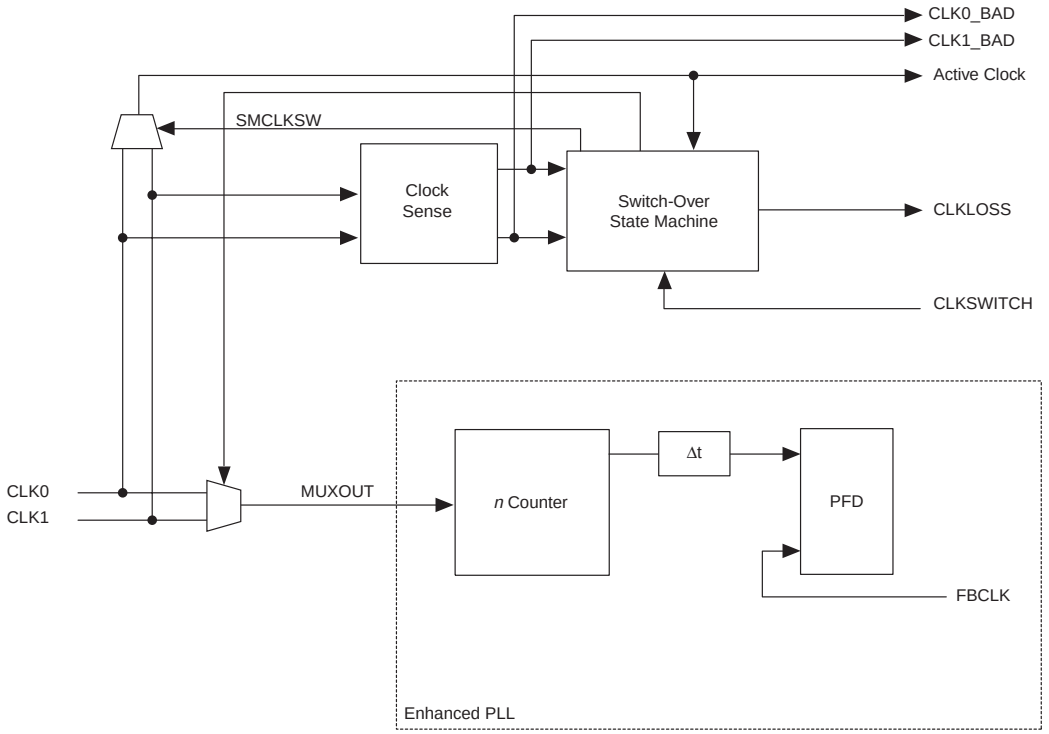
Clock Multiplication & Division

Each Stratix GX device enhanced PLL provides clock synthesis for PLL output ports using $m/(n \times$ post-scale counter) scaling factors. The input clock is divided by a pre-scale divider, n, and is then multiplied by the m feedback factor. The control loop drives the VCO to match $f_{IN} \times (m/n)$. Each output port has a unique post-scale counter that divides down the high-frequency VCO. For multiple PLL outputs with different frequencies, the VCO is set to the least common multiple of the output frequencies that meets its frequency specifications. Then, the post-scale dividers scale down the output frequency for each output port. For example, if output frequencies required from one PLL are 33 and 66 MHz, set the VCO to 330 MHz (the least common multiple in the VCO's range). There is one pre-scale divider, n, and one multiply divider, m, per PLL, with a range of 1 to 512 on each. There are two post-scale dividers (l) for regional clock output ports, four counters (g) for global clock output ports, and up to four counters (e) for external clock outputs, all ranging from 1 to 512. The Quartus II software automatically chooses the appropriate scaling factors according to the input frequency, multiplication, and division values entered.

Clock Switchover

To effectively develop high-reliability network systems, clocking schemes must support multiple clocks to provide redundancy. For this reason, Stratix GX device enhanced PLLs support a flexible clock switchover capability. Figure 4–52 shows a block diagram of the switchover circuit.The switchover circuit is configurable, so you can define how to implement it. Clock-sense circuitry automatically switches from the primary to secondary clock for PLL reference when the primary clock signal is not present.

Figure 4–52. Clock Switchover Circuitry



Note to Figure 4–52:
(1) PFD: phase frequency detector.

There are two possible ways to use the clock switchover feature.

- You can use automatic switchover circuitry for switching between inputs of the same frequency. For example, in applications that require a redundant clock with the same frequency as the primary clock, the switchover state machine generates a signal that controls the multiplexer select input on the bottom of Figure 4–52. In this case, the secondary clock becomes the reference clock for the PLL.
- You can use the `clkswitch` input for user- or system-controlled switch conditions. This is possible for same-frequency switchover or to switch between inputs of different frequencies. For example, if `inclk0` is 66 MHz and `inclk1` is 100 MHz, you must control the switchover because the automatic clock-sense circuitry cannot monitor primary and secondary clock frequencies with a frequency difference of more than ±20%. This feature is useful when clock sources can originate from multiple cards on the backplane,

requiring a system-controlled switchover between frequencies of operation. You can use `clkswitch` together with the lock signal to trigger the switch from a clock that is running but becomes unstable and cannot be locked onto.

During switchover, the PLL VCO continues to run and either slows down or speeds up, generating frequency drift on the PLL outputs. The clock switchover transitions without any glitches. After the switch, there is a finite resynchronization period to lock onto new clock as the VCO ramps up. The exact amount of time it takes for the PLL to relock relates to the PLL configuration and may be adjusted by using the programmable bandwidth feature of the PLL. The preliminary specification for the maximum time to relock is 100 µs.



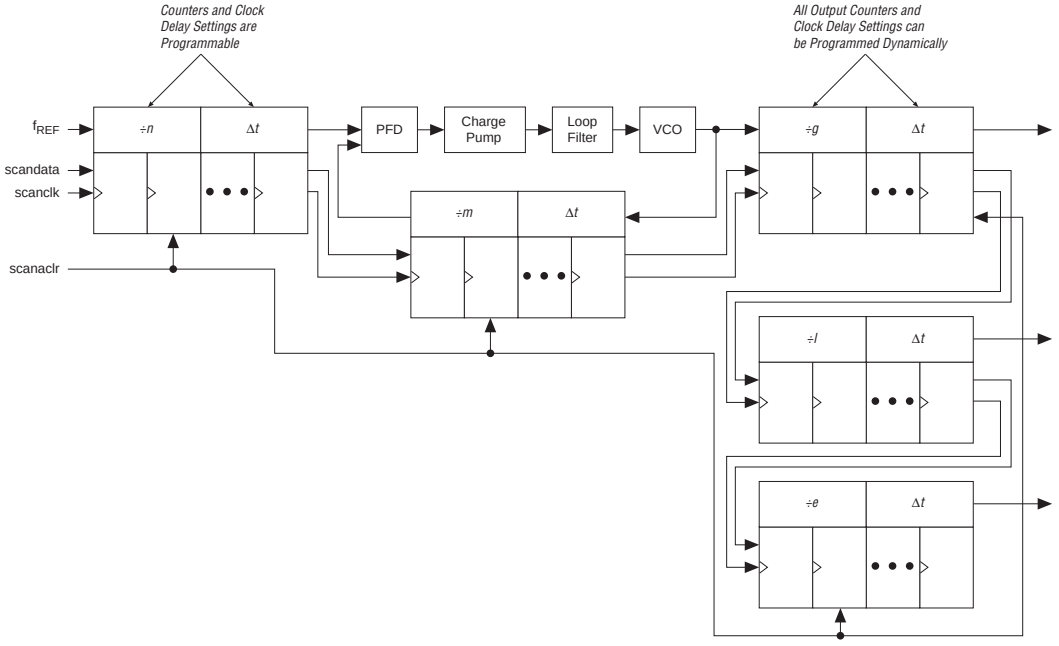
For more information on clock switchover, see *AN313: Implementing Clock Switchover in Stratix & Stratix GX Devices*.

PLL Reconfiguration

The PLL reconfiguration feature enables system logic to change Stratix GX device enhanced PLL counters and delay elements without reloading a Programmer Object File (**.pof**). This provides considerable flexibility for frequency synthesis, allowing real-time PLL frequency and output clock delay variation. You can sweep the PLL output frequencies and clock delay in prototype environments. The PLL reconfiguration feature can also dynamically or intelligently control system clock speeds or t_{CO} delays in end systems.

Clock delay elements at each PLL output port implement variable delay. Figure 4–53 shows a diagram of the overall dynamic PLL control feature for the counters and the clock delay elements. The configuration time is less than 20 µs for the enhanced PLL using a input shift clock rate of 25 MHz. The charge pump, loop filter components, and phase shifting using VCO phase taps cannot be dynamically adjusted.

Figure 4–53. Dynamically Programmable Counters & Delays in Stratix GX Device Enhanced PLLs



PLL reconfiguration data is shifted into serial registers from the logic array or external devices. The PLL input shift data uses a reference input shift clock. Once the last bit of the serial chain is clocked in, the register chain is synchronously loaded into the PLL configuration bits. The shift circuitry also provides an asynchronous clear for the serial registers.

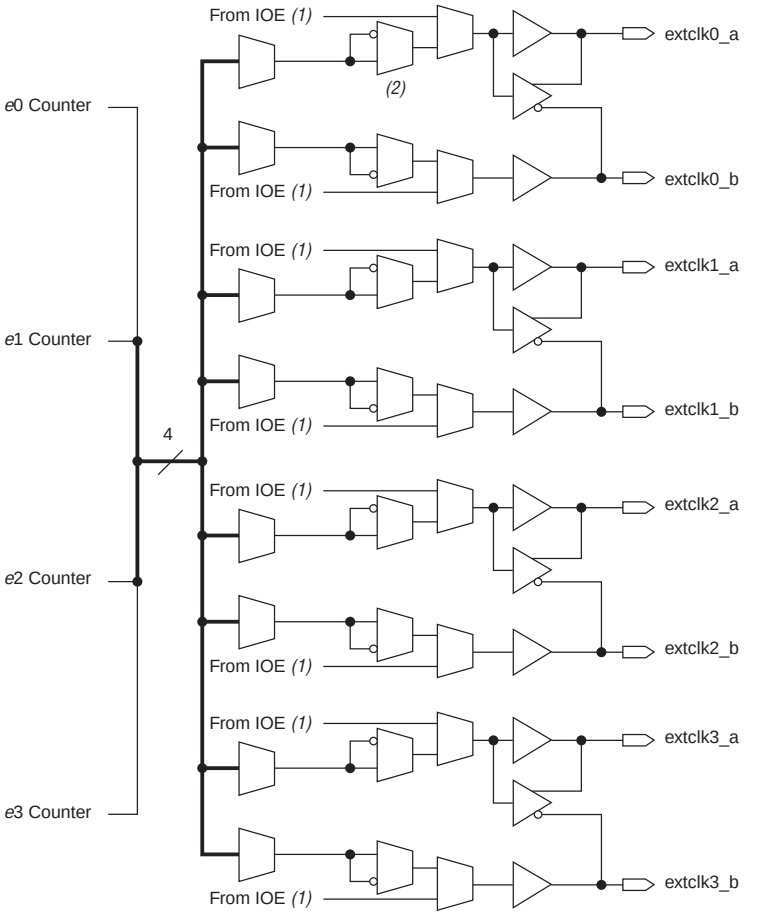
Programmable Bandwidth

You have advanced control of the PLL bandwidth using the programmable control of the PLL loop characteristics, including loop filter and charge pump. The PLL's bandwidth is a measure of its ability to track the input clock and jitter. A high-bandwidth PLL can quickly lock onto a reference clock and react to any changes in the clock. It also allows a wide band of input jitter spectrum to pass to the output. A low-bandwidth PLL takes longer to lock, but it attenuates all high-frequency jitter components. The Quartus II software can adjust PLL characteristics to achieve the desired bandwidth. The programmable bandwidth is tuned by varying the charge pump current, loop filter resistor value, high frequency capacitor value, and *m* counter value. You can manually adjust these values if desired. Bandwidth is programmable from 150 kHz to 2 MHz.

External Clock Outputs

Enhanced PLLs 5 and 6 each support up to eight single-ended clock outputs (or four differential pairs). See Figure 4–54.

Figure 4–54. External Clock Outputs for PLLs 5 & 6



Notes to Figure 4–54:

(1) Each external clock output pin can be used as a general purpose output pin from the logic array. These pins are multiplexed with IOE outputs.

(2) Two single-ended outputs are possible per output counter—either two outputs of the same frequency and phase or one shifted 180°.

Any of the four external output counters can drive the single-ended or differential clock outputs for PLLs 5 and 6. This means one counter or frequency can drive all output pins available from PLL 5 or PLL 6. Each

pair of output pins (four pins total) has dedicated VCC and GND pins to reduce the output clock's overall jitter by providing improved isolation from switching I/O pins.

For PLLs 5 and 6, each pin of a single-ended output pair can either be in phase or 180° out of phase. The clock output pin pairs support the same I/O standards as standard output pins (in the top and bottom banks) as well as LVDS, LVPECL, 3.3-V PCML, HyperTransport technology, differential HSTL, and differential SSTL. Table 4–19 shows which I/O standards the enhanced PLL clock pins support. When in single-ended or differential mode, the two outputs operate off the same power supply. Both outputs use the same standards in single-ended mode to maintain performance. You can also use the external clock output pins as user output pins if external enhanced PLL clocking is not needed.

Table 4–19. I/O Standards Supported for Enhanced PLL Pins (Part 1 of 2)

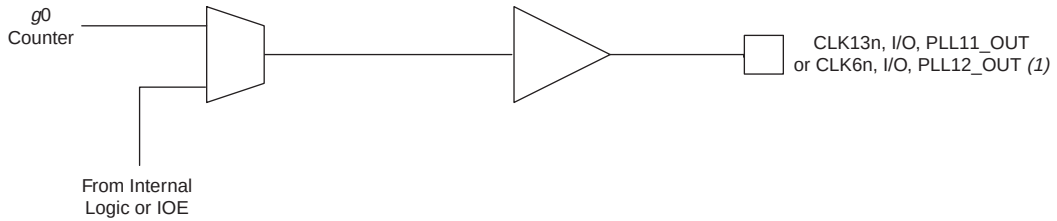
I/O Standard	Input			Output
	INCLK	FBIN	PLLENABLE	EXTCLK
LVTTL	✓	✓	✓	✓
LVCMOS	✓	✓	✓	✓
2.5 V	✓	✓		✓
1.8 V	✓	✓		✓
1.5 V	✓	✓		✓
3.3-V PCI	✓	✓		✓
3.3-V PCI-X	✓	✓		✓
LVPECL	✓	✓		✓
3.3-V PCML	✓	✓		✓
LVDS	✓	✓		✓
HyperTransport technology	✓	✓		✓
Differential HSTL	✓			✓
Differential SSTL				✓
3.3-V GTL	✓	✓		✓
3.3-V GTL+	✓	✓		✓
1.5-V HSTL class I	✓	✓		✓
1.5-V HSTL class II	✓	✓		✓
SSTL-18 class I	✓	✓		✓
SSTL-18 class II	✓	✓		✓
SSTL-2 class I	✓	✓		✓
SSTL-2 class II	✓	✓		✓

Table 4–19. I/O Standards Supported for Enhanced PLL Pins (Part 2 of 2)

I/O Standard	Input			Output
	INCLK	FBIN	PLLENABLE	EXTCLK
SSTL-3 class I	✓	✓		✓
SSTL-3 class II	✓	✓		✓
AGP (1× and 2×)	✓	✓		✓
CTT	✓	✓		✓

Enhanced PLLs 11 and 12 support one single-ended output each (see Figure 4–55). These outputs do not have their own VCC and GND signals. Therefore, to minimize jitter, do not place switching I/O pins next to this output pin.

Figure 4–55. External Clock Outputs for Enhanced PLLs 11 & 12



Note to Figure 4–55:
(1) For PLL 11, this pin is CLK13n; for PLL 12 this pin is CLK7n.

Stratix GX devices can drive any enhanced PLL driven through the global clock or regional clock network to any general I/O pin as an external output clock. The jitter on the output clock is not guaranteed for these cases.

Clock Feedback

The following four feedback modes in Stratix GX device enhanced PLLs allow multiplication and/or phase and delay shifting:

- Zero delay buffer: The external clock output pin is phase-aligned with the clock input pin for zero delay.
- External feedback: The external feedback input pin, FBIN, is phase-aligned with the clock input, CLK, pin. Aligning these clocks allows you to remove clock delay and skew between devices. This mode is only possible for PLLs 5 and 6. PLLs 5 and 6 each support

feedback for one of the dedicated external outputs, either one single-ended or one differential pair. In this mode, one *e* counter feeds back to the PLL `FBIN` input, becoming part of the feedback loop.

- Normal mode: If an internal clock is used in this mode, it is phase-aligned to the input clock pin. The external clock output pin has a phase delay relative to the clock input pin if connected in this mode. You define which internal clock output from the PLL should be phase-aligned to the internal clock pin.
- No compensation: In this mode, the PLL does not compensate for any clock networks or external clock outputs.

Phase & Delay Shifting

Stratix GX device enhanced PLLs provide advanced programmable phase and clock delay shifting. For phase shifting, you can specify a phase shift (in degrees or time units) for each PLL clock output port or for all outputs together in one shift. Phase-shifting values in time units are allowed with a resolution range of 160 to 420 ps. This resolution is a function of frequency input and the multiplication and division factors. In other words, it is a function of the VCO period equal to one-eighth of the VCO period. Each clock output counter can choose a different phase of the VCO period from up to eight taps. You can use this clock output counter along with an initial setting on the post-scale counter to achieve a phase-shift range for the entire period of the output clock. The phase tap feedback to the *m* counter can shift all outputs to a single phase or delay. The Quartus II software automatically sets the phase taps and counter settings according to the phase shift entered.

In addition to the phase-shift feature, the fine tune clock delay shift feature provides advanced time delay shift control on each of the four PLL outputs. Each PLL output shifts in 250-ps increments for a range of –3.0 ns to +3.0 ns between any two outputs using discrete delay elements. Total delay shift between any two PLL outputs must be less than 3 ns. For example, shifts on outputs of –1 and +2 ns is allowed, but not –1 and +2.5 ns. There is some delay variation due to process, voltage, and temperature. Only the clock delay shift blocks can be controlled during system operation for dynamic clock delay control.

Spread-Spectrum Clocking

The Stratix GX device's enhanced PLLs use spread-spectrum technology to reduce electromagnetic interference generation from a system by distributing the energy over a broader frequency range. The enhanced

PLL typically provides 0.5% down spread modulation using a triangular profile. The modulation frequency is programmable. Enabling spread spectrum for a PLL affects all of its outputs.

Lock Detect

The lock output indicates that there is a stable clock output signal in phase with the reference clock. Without any additional circuitry, the lock signal may toggle as the PLL begins tracking the reference clock. You may need to gate the lock signal for use as a system control. The lock signal from the locked port can drive the logic array or an output pin.

Whenever the PLL loses lock for any reason (be it excessive inclk jitter, clock switchover, PLL reconfiguration, power supply noise etc.), the PLL must be reset with the `areset` signal for correct phase shift operation. If the phase relationship between the input clock versus output clock, and between different output clocks from the PLL is not important in the design, then the PLL need not be reset.



See the *Stratix GX FPGA Errata Sheet* for more information on implementing the gated lock signal in the design.

Programmable Duty Cycle

The programmable duty cycle allows enhanced PLLs to generate clock outputs with a variable duty cycle. This feature is supported on each enhanced PLL post-scale counter (*g*0..*g*3, *l*0..*l*3, *e*0..*e*3). The duty cycle setting is achieved by a low and high time count setting for the post-scale dividers. The Quartus II software uses the frequency input and the required multiply or divide rate to determine the duty cycle choices.

Advanced Clear & Enable Control

There are several control signals for clearing and enabling PLLs and their outputs. You can use these signals to control PLL resynchronization and gate PLL output clocks for low-power applications.

The `pllenable` pin is a dedicated pin that enables/disables PLLs. When the `pllenable` pin is low, the clock output ports are driven by `GND` and all the PLLs go out of lock. When the `pllenable` pin goes high again, the PLLs relock and resynchronize to the input clocks. You can choose which PLLs are controlled by the `pllenable` signal by connecting the `pllenable` input port of the `altpll` megafunction to the common `pllenable` input pin.

The areset signals are reset/resynchronization inputs for each PLL. The areset signal should be asserted every time the PLL loses lock to guarantee correct phase relationship between the PLL output clocks. Users should include the areset signal in designs if any of the following conditions are true:

- PLL Reconfiguration or Clock switchover enables in the design.
- Phase relationships between output clocks need to be maintained after a loss of lock condition

The device input pins or logic elements (LEs) can drive these input signals. When driven high, the PLL counters resets, clearing the PLL output and placing the PLL out of lock. The VCO sets back to its nominal setting (~700 MHz). When driven low again, the PLL resynchronizes to its input as it relocks. If the target VCO frequency is below this nominal frequency, then the output frequency starts at a higher value than desired as the PLL locks. If the system cannot tolerate this, the clkena signal can disable the output clocks until the PLL locks.

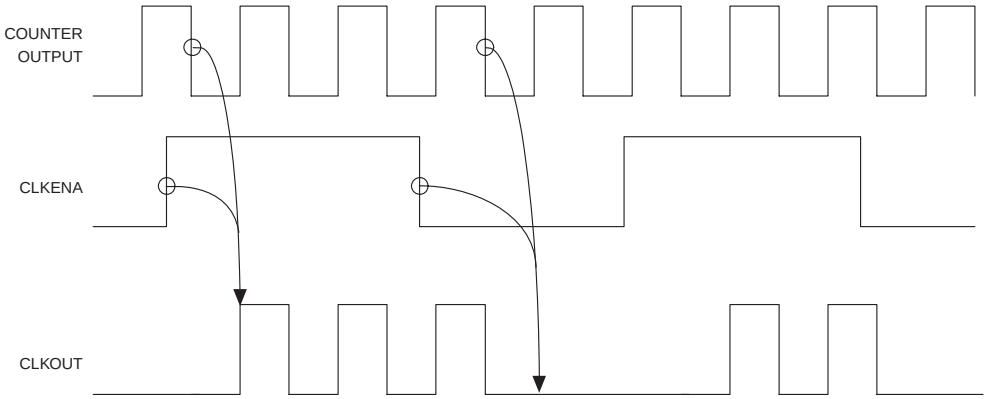
The pfdena signals control the phase frequency detector (PFD) output with a programmable gate. If you disable the PFD, the VCO operates at its last set value of control voltage and frequency with some long-term drift to a lower frequency. The system continues running when the PLL goes out of lock or the input clock is disabled. By maintaining the last locked frequency, the system has time to store its current settings before shutting down. You can either use your own control signal or a clkloss status signal to trigger pfdena.

The clkena signals control the enhanced PLL regional and global outputs. Each regional and global output port has its own clkena signal. The clkena signals synchronously disable or enable the clock at the PLL output port by gating the outputs of the *g* and *l* counters. The clkena signals are registered on the falling edge of the counter output clock to enable or disable the clock without glitches. Figure 4–56 shows the waveform example for a PLL clock port enable. The PLL can remain locked independent of the clkena signals since the loop-related counters are not affected. This feature is useful for applications that require a low power or sleep mode. Upon re-enabling, the PLL does not need a resynchronization or relock period. The clkena signal can also disable clock outputs if the system is not tolerant to frequency overshoot during resynchronization.

The extclkena signals work in the same way as the clkena signals, but they control the external clock output counters (*e*0, *e*1, *e*2, and *e*3). Upon re-enabling, the PLL does not need a resynchronization or relock period

unless the PLL is using external feedback mode. In order to lock in external feedback mode, the external output must drive the board trace back to the FBIN pin.

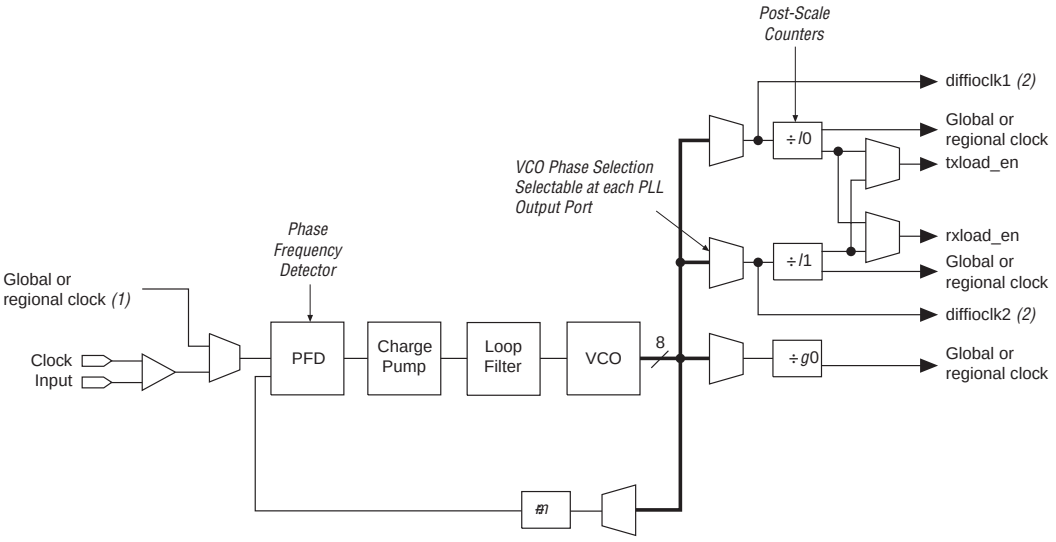
Figure 4–56. extclkena Signals



Fast PLLs

Stratix GX devices contain up to four fast PLLs with high-speed serial interfacing ability, along with general-purpose features. Figure 4–57 shows a diagram of the fast PLL.

Figure 4–57. Stratix GX Device Fast PLL



Notes to Figure 4–57:

(1) In high-speed differential I/O support mode, this high-speed PLL clock feeds the SERDES. Stratix GX devices only support one rate of data transfer per fast PLL in high-speed differential I/O support mode.

(2) This signal is a high-speed differential I/O support SERDES control signal.

Clock Multiplication & Division

The Stratix GX device's fast PLLs provide clock synthesis for PLL output ports using *m*/(post scaler) scaling factors. The input clock is multiplied by the *m* feedback factor. Each output port has a unique post scale counter to divide down the high-frequency VCO. There is one multiply divider, *m*, per fast PLL with a range of 1 to 32. There are two post scale L dividers for regional and/or LVDS interface clocks, and *g*0 counter for global clock output port; all range from 1 to 32.

In the case of a high-speed differential interface, you can set the output counter to 1 to allow the high-speed VCO frequency to drive the SERDES.

External Clock Outputs

Each fast PLL supports differential or single-ended outputs for source-synchronous transmitters or for general-purpose external clocks. There are no dedicated external clock output pins. Any I/O pin can be driven by the fast PLL global or regional outputs as an external output

pin. The I/O standards supported by any particular bank determines what standards are possible for an external clock output driven by the fast PLL in that bank.

Table 4–20 shows the I/O standards supported by fast PLL input pins.

Table 4–20. Fast PLL Port Input Pin I/O Standards

I/O Standard	Input	
	INCLK	PLLENABLE
LVTTL	✓	✓
LVCMOS	✓	✓
2.5 V	✓	
1.8 V	✓	
1.5 V	✓	
3.3-V PCI		
3.3-V PCI-X		
LVPECL	✓	
3.3-V PCML	✓	
LVDS	✓	
HyperTransport technology	✓	
Differential HSTL	✓	
Differential SSTL		
3.3-V GTL	✓	
3.3-V GTL+	✓	
1.5V HSTL class I	✓	
1.5V HSTL class II	✓	
SSTL-18 class I	✓	
SSTL-18 class II	✓	
SSTL-2 class I	✓	
SSTL-2 class II	✓	
SSTL-3 class I	✓	
SSTL-3 class II	✓	
AGP (1× and 2×)	✓	
CTT	✓	

Phase Shifting

Stratix GX device fast PLLs have advanced clock shift capability that enables programmable phase shifts. You can enter a phase shift (in degrees or time units) for each PLL clock output port or for all outputs together in one shift. You can perform phase shifting in time units with a resolution range of 150 to 400 ps. This resolution is a function of the VCO period.

Control Signals

The fast PLL has the same `lock` output, `pllenable` input, and `areset` input control signals as the enhanced PLL.

For more information on high-speed differential I/O support, see the *High-Speed Source-Synchronous Differential I/O Interfaces in Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2*.

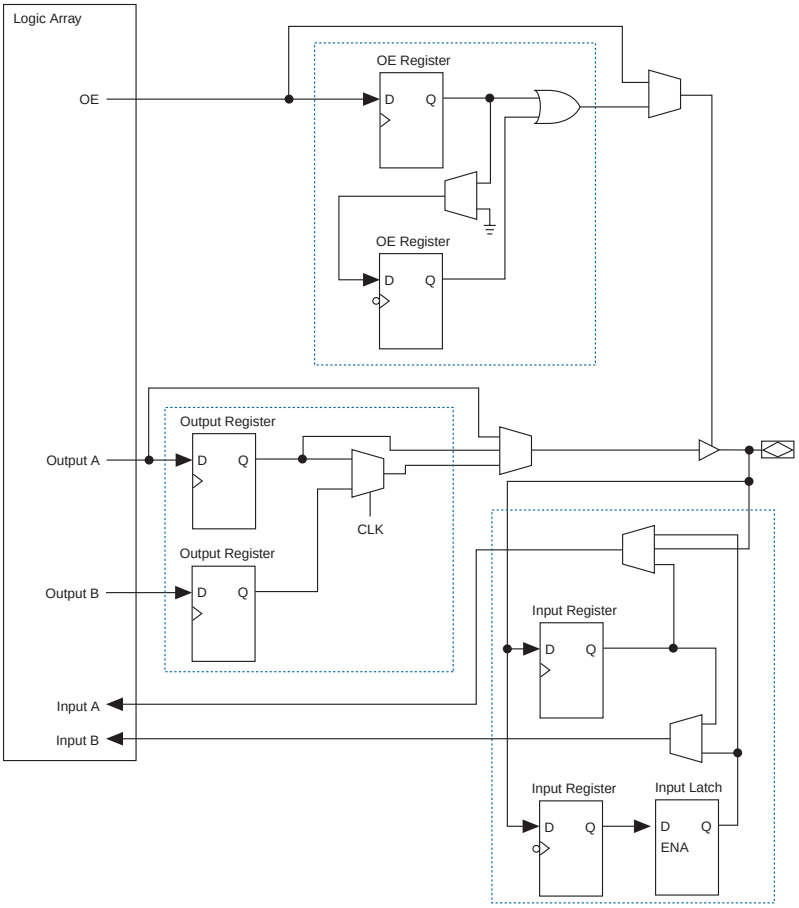
I/O Structure

IOEs provide many features, including:

- Dedicated differential and single-ended I/O buffers
- 3.3-V, 64-bit, 66-MHz PCI compliance
- 3.3-V, 64-bit, 133-MHz PCI-X 1.0 compliance
- Joint Test Action Group (JTAG) boundary-scan test (BST) support
- Differential on-chip termination for LVDS I/O standard
- Programmable pull-up during configuration
- Output drive strength control
- Slew-rate control
- Tri-state buffers
- Bus-hold circuitry
- Programmable pull-up resistors
- Programmable input and output delays
- Open-drain outputs
- DQ and DQS I/O pins
- Double-data rate (DDR) Registers

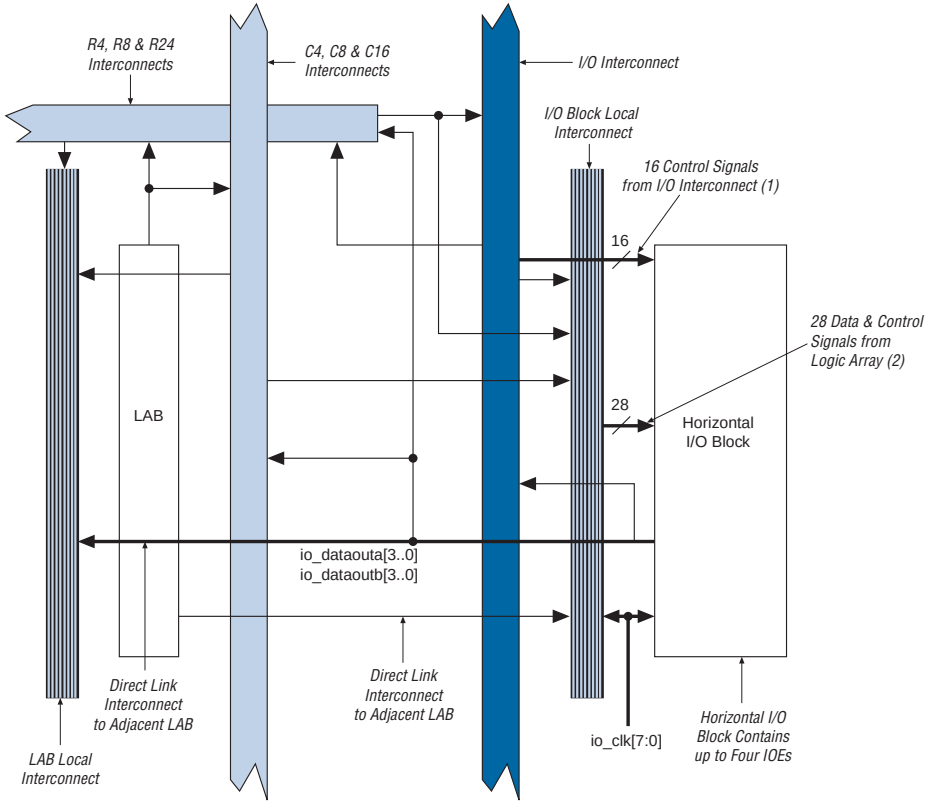
The IOE in Stratix GX devices contains a bidirectional I/O buffer, six registers, and a latch for a complete embedded bidirectional single data rate or DDR transfer. Figure 4–58 shows the Stratix GX IOE structure. The IOE contains two input registers (plus a latch), two output registers, and two output enable registers. The design can use both input registers and the latch to capture DDR input and both output registers to drive DDR outputs. Additionally, the design can use the output enable (OE) register for fast clock-to-output enable timing. The negative edge-clocked OE register is used for DDR SDRAM interfacing. The Quartus II software automatically duplicates a single OE register that controls multiple output or bidirectional pins.

Figure 4–58. Stratix GX IOE Structure



The IOEs are located in I/O blocks around the periphery of the Stratix GX device. There are up to four IOEs per row I/O block and six IOEs per column I/O block. The row I/O blocks drive row, column, or direct link interconnects. The column I/O blocks drive column interconnects. Figure 4–59 shows how a row I/O block connects to the logic array. Figure 4–60 shows how a column I/O block connects to the logic array.

Figure 4–59. Row I/O Block Connection to the Interconnect

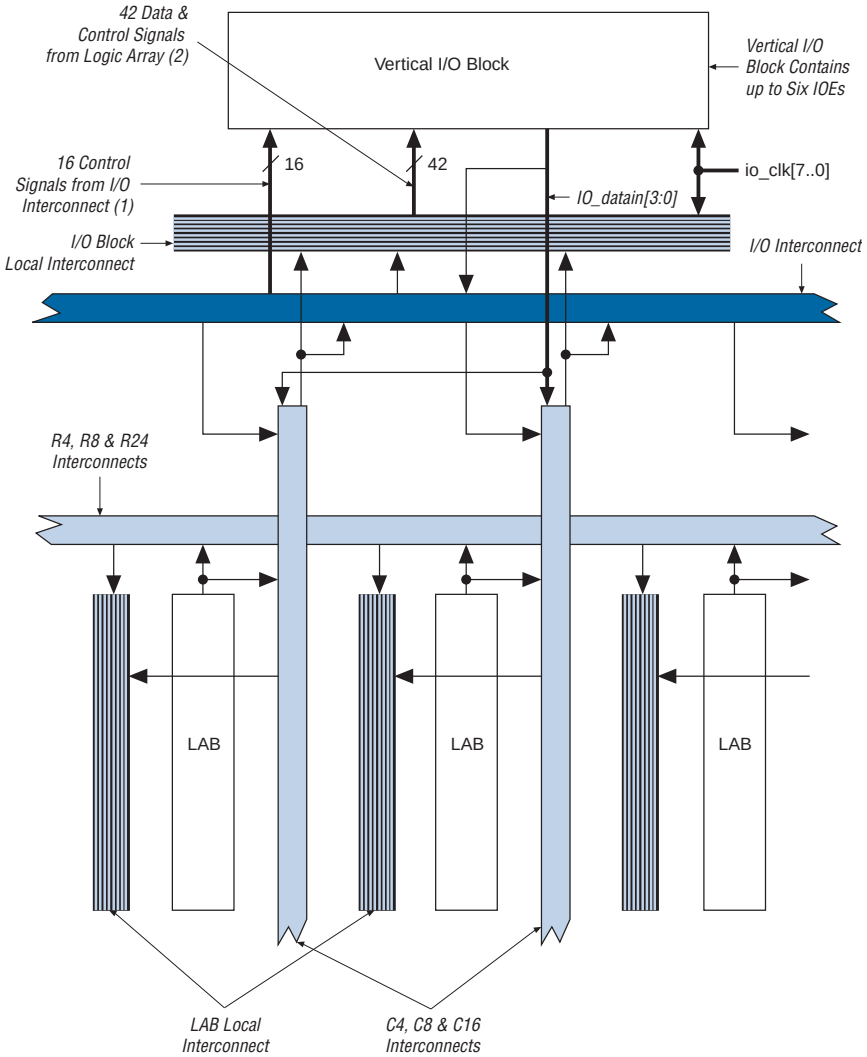


Notes to Figure 4–59:

(1) The 16 control signals are composed of four output enables `io_boe[3..0]`, four clock enables `io_bce[3..0]`, four clocks `io_clk[3..0]`, and four clear signals `io_bclr[3..0]`.

(2) The 28 data and control signals consist of eight data out lines: four lines each for DDR applications `io_dataouta[3..0]` and `io_dataoutb[3..0]`, four output enables `io_coe[3..0]`, four input clock enables `io_cce_in[3..0]`, four output clock enables `io_cce_out[3..0]`, four clocks `io_cclk[3..0]`, and four clear signals `io_cclr[3..0]`.

Figure 4–60. Column I/O Block Connection to the Interconnect



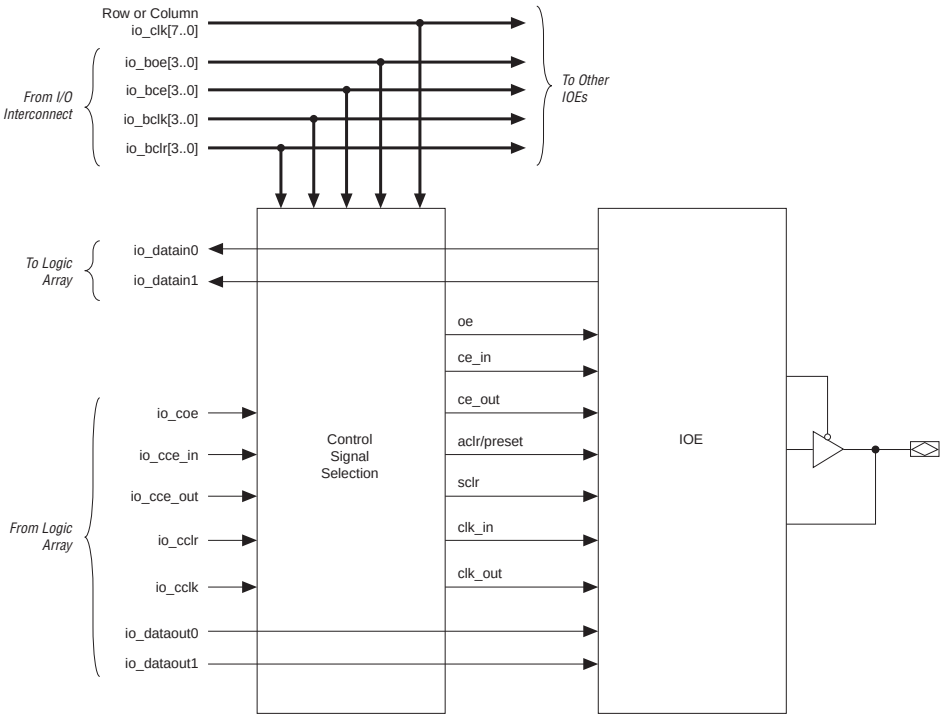
Notes to Figure 4–60:

(1) The 16 control signals are composed of four output enables `io_boe[3..0]`, four clock enables `io_bce[3..0]`, four clocks `io_bclk[3..0]`, and four clear signals `io_bclr[3..0]`.

(2) The 42 data and control signals consist of 12 data out lines; six lines each for DDR applications `io_dataouta[5..0]` and `io_dataoutb[5..0]`, six output enables `io_coe[5..0]`, six input clock enables `io_cce_in[5..0]`, six output clock enables `io_cce_out[5..0]`, six clocks `io_cclk[5..0]`, and six clear signals `io_cclr[5..0]`.

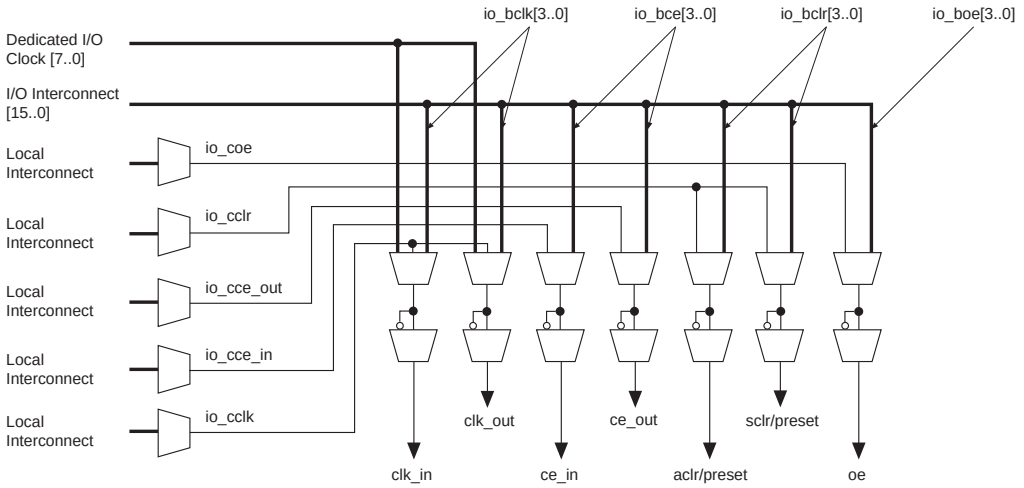
Stratix GX devices have an I/O interconnect similar to the R4 and C4 interconnect to drive high-fanout signals to and from the I/O blocks. There are 16 signals that drive into the I/O blocks composed of four output enables `io_boe[3..0]`, four clock enables `io_bce[3..0]`, four clocks `io_bclk[3..0]`, and four clear signals `io_bclr[3..0]`. The pin's `datain` signals can drive the IO interconnect, which in turn drives the logic array or other I/O blocks. In addition, the control and data signals can be driven from the logic array, providing a slower but more flexible routing resource. The row or column IOE clocks, `io_clk[7..0]`, provide a dedicated routing resource for low-skew, high-speed clocks. I/O clocks are generated from regional, global, or fast regional clocks (see "PLLs & Clock Networks" on page 4–68). Figure 4–61 illustrates the signal paths through the I/O block.

Figure 4–61. Signal Path Through the I/O Block



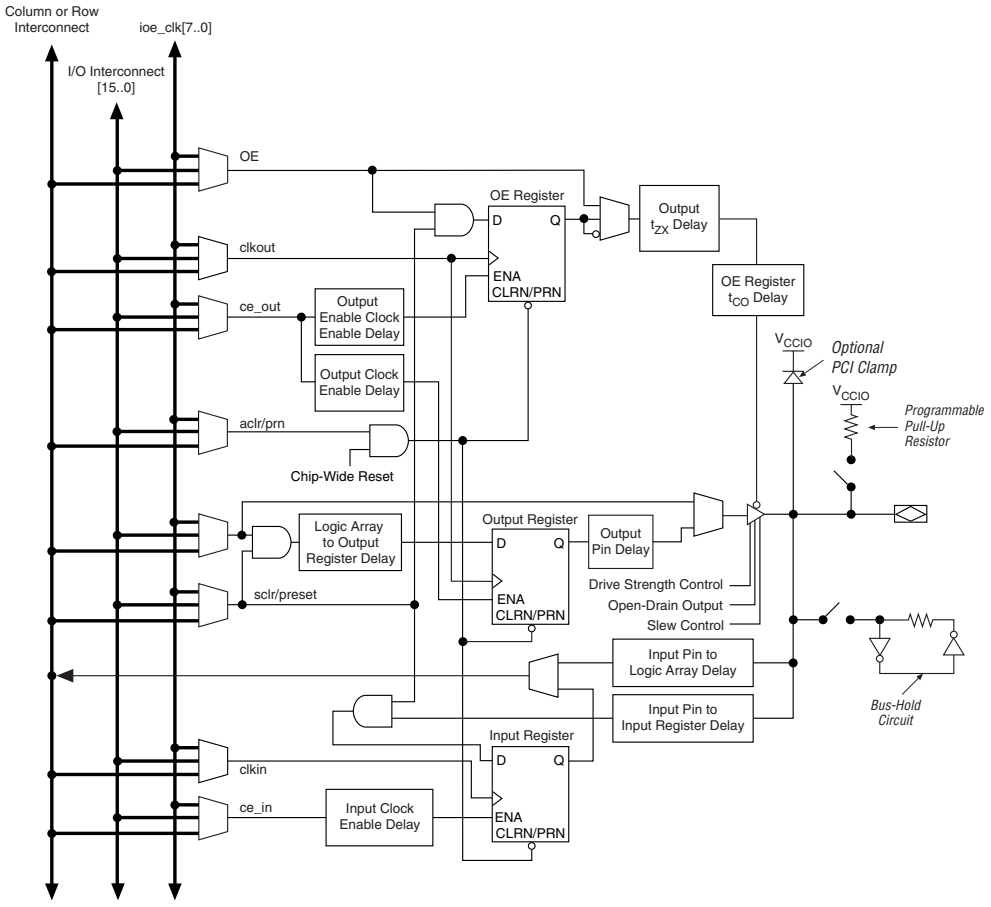
Each IOE contains its own control signal selection for the following control signals: `oe`, `ce_in`, `ce_out`, `aclr/preset`, `sclr/preset`, `clk_in`, and `clk_out`. Figure 4–62 illustrates the control signal selection.

Figure 4–62. Control Signal Selection per IOE



In normal bidirectional operation, the input register can be used for input data requiring fast setup times. The input register can have its own clock input and clock enable separate from the OE and output registers. The output register can be used for data requiring fast clock-to-output performance. The OE register can be used for fast clock-to-output enable timing. The OE and output register share the same clock source and the same clock enable source from local interconnect in the associated LAB, dedicated I/O clocks, and the column and row interconnects. Figure 4–63 shows the IOE in bidirectional configuration.

Figure 4–63. Stratix GX IOE in Bidirectional I/O Configuration *Note (1)*



Note to Figure 4–63:
(1) All input signals to the IOE can be inverted at the IOE.

The Stratix GX device IOE includes programmable delays that can be activated to ensure zero hold times, input IOE register-to-logic array register transfers, or logic array-to-output IOE register transfers.

A path in which a pin directly drives a register may require the delay to ensure zero hold time, whereas a path in which a pin drives a register through combinatorial logic may not require the delay. Programmable delays exist for decreasing input-pin-to-logic-array and IOE input register delays. The Quartus II Compiler can program these delays to automatically minimize setup time while providing a zero hold time.

Programmable delays can increase the register-to-pin delays for output and/or output enable registers. A programmable delay exists to increase the t_{ZX} delay to the output pin, which is required for ZBT interfaces. Table 4–21 shows the programmable delays for Stratix GX devices.

Table 4–21. Stratix GX Programmable Delay Chain

Programmable Delays	Quartus II Logic Option
Input pin to logic array delay	Decrease input delay to internal cells
Input pin to input register delay	Decrease input delay to input register
Output pin delay	Increase delay to output pin
Output enable register t_{CO} delay	Increase delay to output enable pin
Output t_{ZX} delay	Increase t_{ZX} delay to output pin
Output clock enable delay	Increase output clock enable delay
Input clock enable delay	Increase input clock enable delay
Logic array to output register delay	Decrease input delay to output register
Output enable clock enable delay	Increase output enable clock enable delay

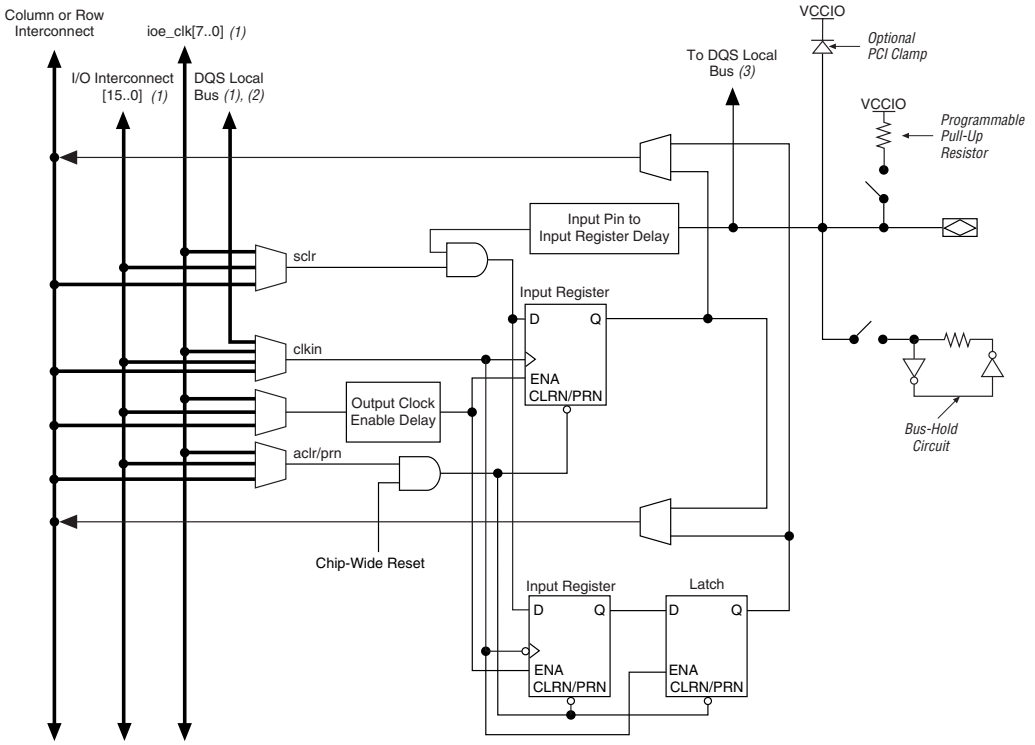
The IOE registers in Stratix GX devices share the same source for clear or preset. You can program preset or clear for each individual IOE. You can also program the registers to power up high or low after configuration is complete. If programmed to power up low, an asynchronous clear can control the registers. If programmed to power up high, an asynchronous preset can control the registers. This feature prevents the inadvertent activation of another device's active-low input upon power-up. If one register in an IOE uses a preset or clear signal then all registers in the IOE must use that same signal if they require preset or clear. Additionally, a synchronous reset signal is available for the IOE registers.

Double-Data Rate I/O Pins

Stratix GX devices have six registers in the IOE, which support DDR interfacing by clocking data on both positive and negative clock edges. The IOEs in Stratix GX devices support DDR inputs, DDR outputs, and bidirectional DDR modes.

When using the IOE for DDR inputs, the two input registers clock double rate input data on alternating edges. An input latch is also used within the IOE for DDR input acquisition. The latch holds the data that is present during the clock high times. This allows both bits of data to be synchronous with the same clock edge (either rising or falling). Figure 4–64 shows an IOE configured for DDR input. Figure 4–65 shows the DDR input timing diagram.

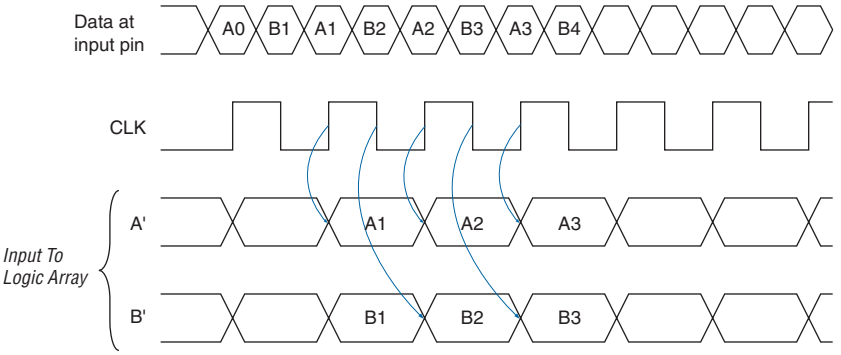
Figure 4–64. Stratix GX IOE in DDR Input I/O Configuration *Note (1)*



Notes to Figure 4–64:

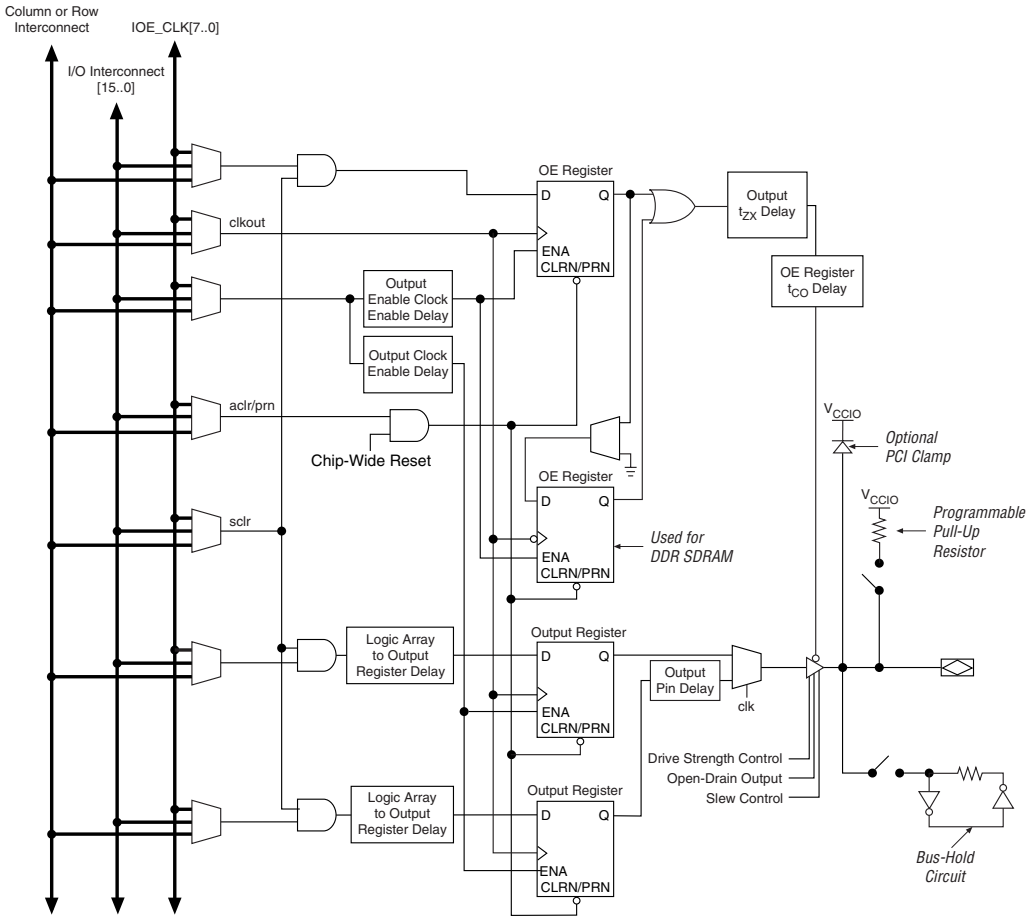
(1) All input signals to the IOE can be inverted at the IOE.
(2) This signal connection is only allowed on dedicated DQ function pins.
(3) This signal is for dedicated DQS function pins only.

Figure 4–65. Input Timing Diagram in DDR Mode



When using the IOE for DDR outputs, the two output registers are configured to clock two data paths from LEs on rising clock edges. These output registers are multiplexed by the clock to drive the output pin at a ×2 rate. One output register clocks the first bit out on the clock high time, while the other output register clocks the second bit out on the clock low time. Figure 4–66 shows the IOE configured for DDR output. Figure 4–67 shows the DDR output timing diagram.

Figure 4–66. Stratix GX IOE in DDR Output I/O Configuration *Notes (1), (2)*

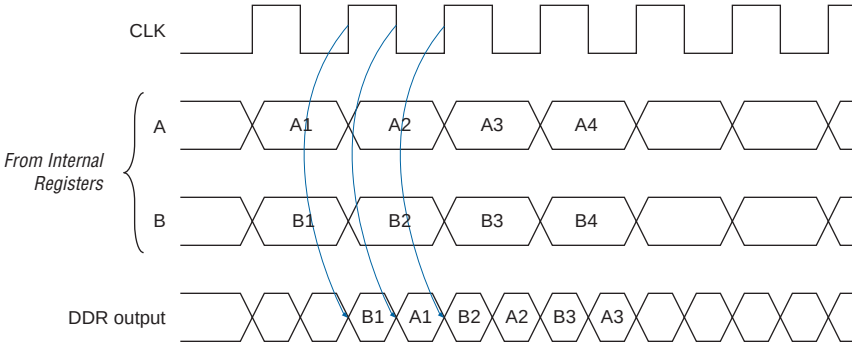


Notes to Figure 4–66:

(1) All input signals to the IOE can be inverted at the IOE.

(2) The tristate is by default active high. It can, however, be designed to be active low.

Figure 4–67. Output Timing Diagram in DDR Mode



The Stratix GX IOE operates in bidirectional DDR mode by combining the DDR input and DDR output configurations. Stratix GX device I/O pins transfer data on a DDR bidirectional bus to support DDR SDRAM. The negative-edge-clocked OE register holds the OE signal inactive until the falling edge of the clock. This is done to meet DDR SDRAM timing requirements.

External RAM Interfacing

Stratix GX devices support DDR SDRAM at up to 200 MHz (400-Mbps data rate) through dedicated phase-shift circuitry, QDR and QDRII SRAM interfaces up to 167 MHz, and ZBT SRAM interfaces up to 200 MHz. Stratix GX devices also provide preliminary support for reduced latency DRAM II (RLDRAM II) at rates up to 200 MHz through the dedicated phase-shift circuitry.

In addition to the required signals for external memory interfacing, Stratix GX devices offer the optional clock enable signal. By default the Quartus II software sets the clock enable signal high, which tells the output register to update with new values. The output registers hold their own values if the design sets the clock enable signal low. See Figure 4–63.

To find out more about the DDR SDRAM specification, see the JEDEC web site (**www.jedec.org**). For information on memory controller megafunctions for Stratix GX devices, see the Altera web site (**www.altera.com**). See *AN 342: Interfacing DDR SDRAM with Stratix & Stratix GX Devices* for more information on DDR SDRAM interface in Stratix GX. Also see *AN 349: QDR SRAM Controller Reference Design for Stratix & Stratix GX Devices* and *AN 329: ZBT SRAM Controller Reference Design for Stratix & Stratix GX Devices*.

Table 4–22 shows the performance specification for DDR SDRAM, RLDRAM II, QDR SRAM, QDRII SRAM, and ZBT SRAM interfaces in EP1SGX10 through EP1SGX40 devices. The DDR SDRAM and QDR SRAM numbers in Table 4–22 have been verified with hardware characterization with third-party DDR SDRAM and QDR SRAM devices over temperature and voltage extremes.

Table 4–22. External RAM Support in EP1SGX10 Through EP1SGX40 Devices

DDR Memory Type	I/O Standard	Maximum Clock Rate (MHz)		
		-5 Speed Grade	-6 Speed Grade	-7 Speed Grade
DDR SDRAM *(1)*, *(2)*	SSTL-2	200	167	133
DDR SDRAM - side banks *(2)*, *(3)*, *(4)*	SSTL-2	150	133	133
RLDRAM II *(4)*	1.8-V HSTL	200	*(5)*	*(5)*
QDR SRAM *(6)*	1.5-V HSTL	167	167	133
QDRII SRAM *(6)*	1.5-V HSTL	200	167	133
ZBT SRAM *(7)*	LVTTL	200	200	167

Notes to Table 4–22:

(1) These maximum clock rates apply if the Stratix GX device uses DQS phase-shift circuitry to interface with DDR SDRAM. DQS phase-shift circuitry is only available in the top and bottom I/O banks (I/O banks 3, 4, 7, and 8).
(2) For more information on DDR SDRAM, see *AN 342: Interfacing DDR SDRAM with Stratix & Stratix GX Devices.*
(3) DDR SDRAM is supported on the Stratix GX device side I/O banks (I/O banks 1, 2, 5, and 6) without dedicated DQS phase-shift circuitry. The read DQS signal is ignored in this mode.
(4) These performance specifications are preliminary.
(5) This device does not support RLDRAM II.
(6) For more information on QDR or QDRII SRAM, see *AN 349: QDR SRAM Controller Reference Design for Stratix & Stratix GX Devices*.
(7) For more information on ZBT SRAM, see *AN 329: ZBT SRAM Controller Reference Design for Stratix & Stratix GX Devices*.

In addition to six I/O registers and one input latch in the IOE for interfacing to these high-speed memory interfaces, Stratix GX devices also have dedicated circuitry for interfacing with DDR SDRAM. In every Stratix GX device, the I/O banks at the top (I/O banks 3 and 4) and bottom (I/O banks 7 and 8) of the device support DDR SDRAM up to 200 MHz. These pins support DQS signals with DQ bus modes of ×8, ×16, or ×32.

Table 4–23 shows the number of DQ and DQS buses that are supported per device.

Table 4–23. DQS & DQ Bus Mode Support *Note (1)*

Device	Package	Number of ×8 Groups	Number of ×16 Groups	Number of ×32 Groups
EP1SGX10	672-pin FineLine BGA	12 *(2)*	0	0
EP1SGX25	672-pin FineLine BGA	16 *(3)*	8	4
	1,020-pin FineLine BGA	20	8	4
EP1SGX40	1,020-pin FineLine BGA	20	8	4

Notes to Table 4–23:

(1) See the *Selectable I/O Standards in Stratix & Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2* for V_{REF} guidelines.

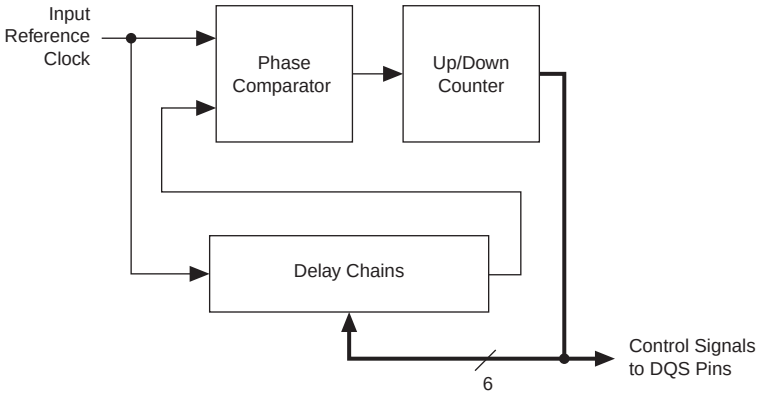
(2) These packages have six groups in I/O banks 3 and 4 and six groups in I/O banks 7 and 8.

(3) These packages have eight groups in I/O banks 3 and 4 and eight groups in I/O banks 7 and 8.

A compensated delay element on each DQS pin automatically aligns input DQS synchronization signals with the data window of their corresponding DQ data signals. The DQS signals drive a local DQS bus in the top and bottom I/O banks. This DQS bus is an additional resource to the I/O clocks and clocks DQ input registers with the DQS signal.

Two separate single phase-shifting reference circuits are located on the top and bottom of the Stratix GX device. Each circuit is driven by a system reference clock through the `CLK` pins that is the same frequency as the DQS signal. Clock pins `CLK[15..12]p` feed the phase-shift circuitry on the top of the device and clock pins `CLK[7..4]p` feed the phase-shift circuitry on the bottom of the device. The phase-shifting reference circuit on the top of the device controls the compensated delay elements for all 10 DQS pins located at the top of the device. The phase-shifting reference circuit on the bottom of the device controls the compensated delay elements for all 10 DQS pins located on the bottom of the device. All 10 delay elements (DQS signals) on either the top or bottom of the device shift by the same degree amount. For example, all 10 DQS pins on the top of the device can be shifted by 90° and all 10 DQS pins on the bottom of the device can be shifted by 72°. The reference circuits require a maximum of 256 system reference clock cycles to set the correct phase on the DQS delay elements. Figure 4–68 illustrates the phase-shift reference circuit control of each DQS delay shift on the top of the device. This same circuit is duplicated on the bottom of the device.

Figure 4–68. Simplified Diagram of the DQS Phase-Shift Circuitry



See the *External Memory Interfaces* chapter of the *Stratix GX Device Handbook, Volume 2* for more information on external memory interfaces.

Programmable Drive Strength

The output buffer for each Stratix GX device I/O pin has a programmable drive strength control for certain I/O standards. The LVTTL and LVCMOS standard has several levels of drive strength that the user can control. SSTL-3 class I and II, SSTL-2 class I and II, HSTL class I and II, and 3.3-V GTL+ support a minimum setting, the lowest drive strength that guarantees the I_{OH}/I_{OL} of the standard. Using minimum settings provides signal slew rate control to reduce system noise and signal overshoot.

Table 4–24 shows the possible settings for the I/O standards with drive strength control.

Table 4–24. Programmable Drive Strength

I/O Standard	I_{OH} / I_{OL} Current Strength Setting (mA)
3.3-V LVTTL	24 *(1)*, 16, 12, 8, 4
3.3-V LVCMOS	24 *(2)*, 12 *(1)*, 8, 4, 2
2.5-V LVTTL/LVCMOS	16 *(1)*, 12, 8, 2
1.8-V LVTTL/LVCMOS	12 *(1)*, 8, 2
1.5-V LVCMOS	8 *(1)*, 4, 2
GTL/GTL+ 1.5-V HSTL class I and II 1.8-V HSTL class I and II SSTL-3 class I and II SSTL-2 class I and II SSTL-18 class I and II	Support maximum and minimum strength

Notes to Table 4–24:
(1) This is the Quartus II software default current setting.
(2) I/O banks 1 and 2 do not support this setting.

The Quartus II software, beginning with version 4.2, reports current strength as "PCI Compliant" for 3.3-V PCI, 3.3-V PCI-X 1.0, and Compact PCI I/O standards.

Stratix GX devices support series on-chip termination (OCT) using programmable drive strength. For more information, contact your Altera Support Representative.

Open-Drain Output

Stratix GX devices provide an optional open-drain (equivalent to an open-collector) output for each I/O pin. This open-drain output enables the device to provide system-level control signals (that is, interrupt and write-enable signals) that can be asserted by any of several devices.

Slew-Rate Control

The output buffer for each Stratix GX device I/O pin has a programmable output slew-rate control that can be configured for low-noise or high-speed performance. A faster slew rate provides high-speed transitions for high-performance systems. However, these fast transitions may introduce noise transients into the system. A slow slew rate reduces system noise, but adds a nominal delay to rising and falling edges. Each I/O pin has an individual slew-rate control, allowing you to specify the slew rate on a pin-by-pin basis. The slew-rate control affects both the rising and falling edges.

Bus Hold

Each Stratix GX device I/O pin provides an optional bus-hold feature. The bus-hold circuitry can weakly hold the signal on an I/O pin at its last-driven state. Since the bus-hold feature holds the last-driven state of the pin until the next input signal is present, an external pull-up or pull-down resistor is not needed to hold a signal level when the bus is tri-stated.

Table 4–25 shows bus hold support for different pin types.

Table 4–25. Bus Hold Support

Pin Type	Bus Hold
I/O pins	✓
CLK[15..0]	
CLK[0,1,2,3,8,9,10,11]	
FCLK	✓
FPLL[7..10]CLK	

The bus-hold circuitry also pulls undriven pins away from the input threshold voltage where noise can cause unintended high-frequency switching. You can select this feature individually for each I/O pin. The bus-hold output drives no higher than V_{CCIO} to prevent overdriving signals. If the bus-hold feature is enabled, the programmable pull-up option cannot be used. Disable the bus-hold feature when using open-drain outputs with the GTL+ I/O standard or when the I/O pin has been configured for differential signals.

The bus-hold circuitry uses a resistor with a nominal resistance (R_{BH}) of approximately 7 kΩ to weakly pull the signal level to the last-driven state. The chapter *DC & Switching Characteristics* of the *Stratix GX Device Handbook, Volume 1* gives the specific sustaining current driven through this resistor and the overdrive current used to identify the next-driven input level. This information is provided for each V_{CCIO} voltage level.

The bus-hold circuitry is active only after configuration. When going into user mode, the bus-hold circuit captures the value on the pin present at the end of configuration.

Programmable Pull-Up Resistor

Each Stratix GX device I/O pin provides an optional programmable pull-up resistor during user mode. If this feature is enabled for an I/O pin, the pull-up resistor (typically 25 kΩ) weakly holds the output to the V_{CCIO} level of the output pin's bank. Table 4–26 shows which pin types support the weak pull-up resistor feature.

Table 4–26. Programmable Weak Pull-Up Resistor Support

Pin Type	Programmable Weak Pull-Up Resistor
I/O pins	✓
CLK[15..0]	
FCLK	✓
FPLL[7..10]CLK	
Configuration pins	
JTAG pins	✓ *(1)*

Note to Table 4–26:
(1) TDO pins do not support programmable weak pull-up resistors.

Advanced I/O Standard Support

Stratix GX device IOEs support the following I/O standards:

- LVTTL
- LVCMOS
- 1.5 V
- 1.8 V
- 2.5 V
- 3.3-V PCI
- 3.3-V PCI-X 1.0
- 3.3-V AGP (1× and 2×)

- LVDS
- LVPECL
- 3.3-V PCML
- HyperTransport
- Differential HSTL (on input/output clocks only)
- Differential SSTL (on output column clock pins only)
- GTL/GTL+
- 1.5-V HSTL class I and II
- 1.8-V HSTL Class I and II
- SSTL-3 class I and II
- SSTL-2 class I and II
- SSTL-18 class I and II
- CTT

Table 4–27 describes the I/O standards supported by Stratix GX devices.

Table 4–27. Stratix GX Supported I/O Standards (Part 1 of 2)

I/O Standard	Type	Input Reference Voltage (V_{REF}) (V)	Output Supply Voltage (V_{CCIO}) (V)	Board Termination Voltage (V_{TT}) (V)
LVTTL	Single-ended	N/A	3.3	N/A
LVCMOS	Single-ended	N/A	3.3	N/A
2.5 V	Single-ended	N/A	2.5	N/A
1.8 V	Single-ended	N/A	1.8	N/A
1.5 V	Single-ended	N/A	1.5	N/A
3.3-V PCI	Single-ended	N/A	3.3	N/A
3.3-V PCI-X 1.0	Single-ended	N/A	3.3	N/A
LVDS	Differential	N/A	3.3	N/A
LVPECL	Differential	N/A	3.3	N/A
3.3-V PCML	Differential	N/A	3.3	N/A
HyperTransport	Differential	N/A	2.5	N/A
Differential HSTL *(1)*	Differential	0.75	1.5	0.75
Differential SSTL *(2)*	Differential	1.25	2.5	1.25
GTL	Voltage-referenced	0.8	N/A	1.20
GTL+	Voltage-referenced	1.0	N/A	1.5
1.5-V HSTL class I and II	Voltage-referenced	0.75	1.5	0.75
1.8-V HSTL class I and II	Voltage-referenced	0.9	1.8	0.9
SSTL-18 class I and II	Voltage-referenced	0.90	1.8	0.90
SSTL-2 class I and II	Voltage-referenced	1.25	2.5	1.25

Table 4–27. Stratix GX Supported I/O Standards (Part 2 of 2)

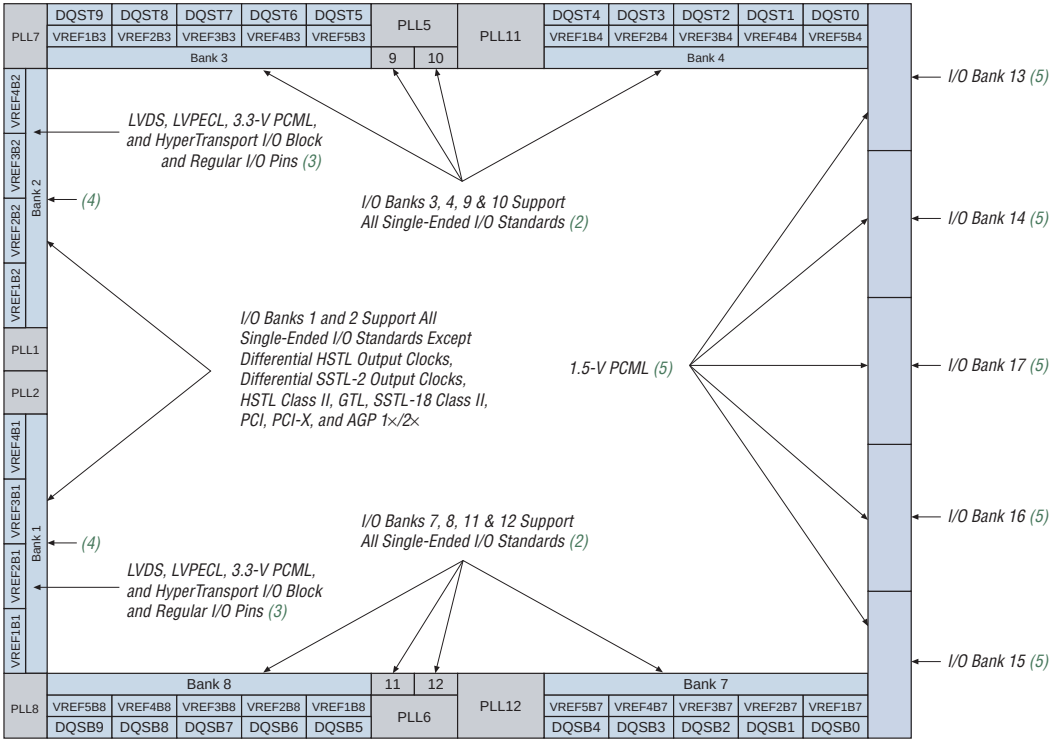
I/O Standard	Type	Input Reference Voltage (V_{REF}) (V)	Output Supply Voltage (V_{CCIO}) (V)	Board Termination Voltage (V_{TT}) (V)
SSTL-3 class I and II	Voltage-referenced	1.5	3.3	1.5
AGP (1× and 2×)	Voltage-referenced	1.32	3.3	N/A
CTT	Voltage-referenced	1.5	3.3	1.5

Notes to Table 4–27:
(1) This I/O standard is only available on input and output clock pins.
(2) This I/O standard is only available on output column clock pins.

For more information on I/O standards supported by Stratix GX devices, see the *Selectable I/O Standards in Stratix & Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2*.

Stratix GX devices contain eight I/O banks in addition to the four enhanced PLL external clock out banks, as shown in Figure 4–69. The four I/O banks on the right and left of the device contain circuitry to support high-speed differential I/O for LVDS, LVPECL, 3.3-V PCML, and HyperTransport inputs and outputs. These banks support all I/O standards listed in Table 4–27 except PCI I/O pins or PCI-X 1.0, GTL, SSTL-18 Class II, and HSTL Class II outputs. The top and bottom I/O banks support all single-ended I/O standards. Additionally, Stratix GX devices support four enhanced PLL external clock output banks, allowing clock output capabilities such as differential support for SSTL and HSTL. Table 4–28 shows I/O standard support for each I/O bank.

Figure 4–69. Stratix GX I/O Banks *Notes (1), (2), (3)*



Notes to Figure 4–69:

(1) Figure 4–69 is a top view of the Stratix GX silicon die.

(2) Banks 9 through 12 are enhanced PLL external clock output banks.

(3) If the high-speed differential I/O pins are not used for high-speed differential signaling, they can support all of the I/O standards except HSTL class I and II, GTL, SSTL-18 Class II, PCI, PCI-X, and AGP 1×/2×.

(4) For guidelines for placing single-ended I/O pads next to differential I/O pads, see the *Selectable I/O Standards in Stratix & Stratix GX Devices* chapter in the *Stratix GX Device Handbook, Volume 2*.

(5) These I/O banks in Stratix GX devices also support the LVDS, LVPECL, and 3.3-V PCML I/O standards on reference clocks and receiver input pins (AC coupled)

Table 4–28 shows I/O standard support for each I/O bank.

Table 4–28. I/O Support by Bank (Part 1 of 2)

I/O Standard	Top & Bottom Banks (3, 4, 7 & 8)	Left Banks (1 & 2)	Enhanced PLL External Clock Output Banks (9, 10, 11 & 12)
LVTTL	✓	✓	✓
LVCMOS	✓	✓	✓
2.5 V	✓	✓	✓
1.8 V	✓	✓	✓
1.5 V	✓	✓	✓
3.3-V PCI	✓		✓
3.3-V PCI-X 1.0	✓		✓
LVPECL		✓	✓
3.3-V PCML		✓	✓
LVDS		✓	✓
HyperTransport technology		✓	✓
Differential HSTL (clock inputs)	✓	✓	
Differential HSTL (clock outputs)			✓
Differential SSTL (clock outputs)			✓
3.3-V GTL	✓		✓
3.3-V GTL+	✓	✓	✓
1.5-V HSTL class I	✓	✓	✓
1.5-V HSTL class II	✓		✓
1.8-V HSTL class I	✓	✓	✓
1.8-V HSTL class II	✓		✓
SSTL-18 class I	✓	✓	✓
SSTL-18 class II	✓		✓
SSTL-2 class I	✓	✓	✓
SSTL-2 class II	✓	✓	✓
SSTL-3 class I	✓	✓	✓

Table 4–28. I/O Support by Bank (Part 2 of 2)

I/O Standard	Top & Bottom Banks (3, 4, 7 & 8)	Left Banks (1 & 2)	Enhanced PLL External Clock Output Banks (9, 10, 11 & 12)
SSTL-3 class II	✓	✓	✓
AGP (1× and 2×)	✓		✓
CTT	✓	✓	✓

Each I/O bank has its own VCCIO pins. A single device can support 1.5-, 1.8-, 2.5-, and 3.3-V interfaces; each bank can support a different standard independently. Each bank also has dedicated VREF pins to support any one of the voltage-referenced standards (such as SSTL-3) independently.

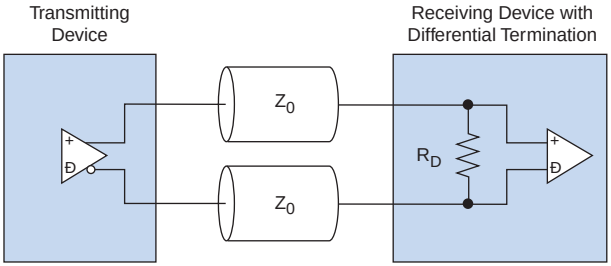
Each I/O bank can support multiple standards with the same V_{CCIO} for input and output pins. Each bank can support one voltage-referenced I/O standard. For example, when V_{CCIO} is 3.3 V, a bank can support LVTTL, LVCMOS, 3.3-V PCI, and SSTL-3 for inputs and outputs.

Differential On-Chip Termination

Stratix GX devices provide differential on-chip termination (LVDS I/O standard) to reduce reflections and maintain signal integrity. Differential on-chip termination simplifies board design by minimizing the number of external termination resistors required. Termination can be placed inside the package, eliminating small stubs that can still lead to reflections. The internal termination is designed using transistors in the linear region of operation.

Stratix GX devices support internal differential termination with a nominal resistance value of 137.5 Ω for LVDS input receiver buffers. LVPECL signals require an external termination resistor. Figure 4–70 shows the device with differential termination.

Figure 4–70. LVDS Input Differential On-Chip Termination



I/O banks on the left and right side of the device support LVDS receiver (far-end) differential termination.

Table 4–29 shows the Stratix GX device differential termination support.

Table 4–29. Differential Termination Supported by I/O Banks

Differential Termination Support	I/O Standard Support	Top & Bottom Banks (3, 4, 7 & 8)	Left Banks (1 & 2)
Differential termination *(1)*, *(2)*	LVDS		✓

Notes to Table 4–29:
(1) Clock pin CLK0, CLK2, CLK9, CLK11, and pins FPLL[7..10]CLK do not support differential termination.
(2) Differential termination is only supported for LVDS because of a 3.3-V V_{CCIO}.

Table 4–30 shows the termination support for different pin types.

Table 4–30. Differential Termination Support Across Pin Types

Pin Type	R_D
Top and bottom I/O banks (3, 4, 7, and 8)	
DIFFIO_RX[]	✓
CLK[0,2,9,11],CLK[4-7],CLK[12-15]	
CLK[1,3,8,10]	✓
FCLK	
FPLL[7..10]CLK	

The differential on-chip resistance at the receiver input buffer is 118 Ω ±20 %.

However, there is additional resistance present between the device ball and the input of the receiver buffer, as shown in Figure 4–71. This resistance is because of package trace resistance (which can be calculated as the resistance from the package ball to the pad) and the parasitic layout metal routing resistance (which is shown between the pad and the intersection of the on-chip termination and input buffer).

Figure 4–71. Differential Resistance of LVDS Differential Pin Pair (R_D)

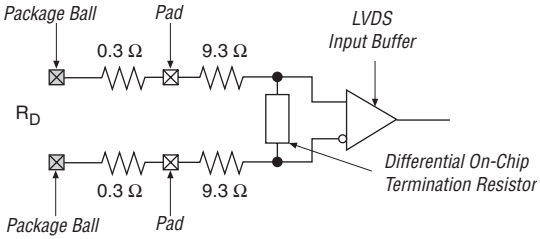


Table 4–31 defines the specification for internal termination resistance for commercial devices.

Table 4–31. Differential On-Chip Termination

Symbol	Description	Conditions	Resistance			Unit
			Min	Typ	Max	
R_D *(2)*	Internal differential termination for LVDS	Commercial *(1)*, *(3)*	110	135	165	Ω
		Industrial *(2)*, *(3)*	100	135	170	Ω

Notes to Table 4–31:

(1) Data measured over minimum conditions (T_j = 0 C, V_{CCIO} +5%) and maximum conditions (T_j = 85 C, V_{CCIO} = –5%).

(2) Data measured over minimum conditions (T_j = –40 C, V_{CCIO} +5%) and maximum conditions (T_j = 100 C, V_{CCIO} = –5%).

(3) LVDS data rate is supported for 840 Mbps using internal differential termination.

MultiVolt I/O Interface

The Stratix GX architecture supports the MultiVolt I/O interface feature, which allows Stratix GX devices in all packages to interface with systems of different supply voltages.

The Stratix GX VCCINT pins must always be connected to a 1.5-V power supply. With a 1.5-V V_{CCINT} level, input pins are 1.5-V, 1.8-V, 2.5-V, and 3.3-V tolerant. The VCCIO pins can be connected to either a 1.5-V, 1.8-V,

2.5-V, or 3.3-V power supply, depending on the output requirements. The output levels are compatible with systems of the same voltage as the power supply (for example, when VCCIO pins are connected to a 1.5-V power supply, the output levels are compatible with 1.5-V systems). When VCCIO pins are connected to a 3.3-V power supply, the output high is 3.3 V and is compatible with 3.3-V or 5.0-V systems.

Table 4–32 summarizes Stratix GX MultiVolt I/O support.

Table 4–32. Stratix GX MultiVolt I/O Support *Note (1)*

V_{CCIO} (V)	Input Signal *(5)*					Output Signal *(6)*				
	1.5 V	1.8 V	2.5 V	3.3 V	5.0 V	1.5 V	1.8 V	2.5 V	3.3 V	5.0 V
1.5	✓	✓	✓ *(2)*	✓ *(2)*		✓				
1.8	✓ *(2)*	✓	✓ *(2)*	✓ *(2)*		✓ *(3)*	✓			
2.5			✓	✓		✓ *(3)*	✓ *(3)*	✓		
3.3			✓ *(2)*	✓	✓ *(4)*	✓ *(3)*	✓ *(3)*	✓ *(3)*	✓	✓

Notes to Table 4–32:

(1) To drive inputs higher than V_{CCIO} but less than 4.1 V, disable the PCI clamping diode. However, to drive 5.0-V inputs to the device, enable the PCI clamping diode to prevent V_I from rising above 4.0 V.

(2) The input pin current may be slightly higher than the typical value.

(3) Although V_{CCIO} specifies the voltage necessary for the Stratix GX device to drive out, a receiving device powered at a different level can still interface with the Stratix GX device if it has inputs that tolerate the V_{CCIO} value.

(4) Stratix GX devices can be 5.0-V tolerant with the use of an external resistor and the internal PCI clamp diode.

(5) This is the external signal that is driving the Stratix GX device.

(6) This represents the system voltage that Stratix GX supports when a VCCIO pin is connected to a specific voltage level. For example, when VCCIO is 3.3 V and if the I/O standard is LVTTL/LVCMOS, the output high of the signal coming out from Stratix GX is 3.3 V and is compatible with 3.3-V or 5.0-V systems.

Power Sequencing & Hot Socketing

Because Stratix GX devices can be used in a mixed-voltage environment, they have been designed specifically to tolerate any possible power-up sequence. Therefore, the VCCIO and VCCINT power supplies may be powered in any order.

Signals can be driven into Stratix GX devices before and during power up without damaging the device. In addition, Stratix GX devices do not drive out during power up. Once operating conditions are reached and the device is configured, Stratix GX devices operate as specified by the user. For more information, see the *Selectable I/O Standards in Stratix & Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2*.

IEEE Std. 1149.1 (JTAG) Boundary-Scan Support

All Stratix GX devices provide JTAG BST circuitry that complies with the IEEE Std. 1149.1a-1990 specification. JTAG boundary-scan testing can be performed either before or after, but not during configuration. Stratix GX devices can also use the JTAG port for configuration together with either the Quartus II software or hardware using either Jam Files (**.jam**) or Jam Byte-Code Files (**.jbc**).

Stratix GX devices support IOE I/O standard setting reconfiguration through the JTAG BST chain. The JTAG chain can update the I/O standard for all input and output pins any time before or during user mode. You can use this ability for JTAG testing before configuration when some of the Stratix GX pins drive or receive from other devices on the board using voltage-referenced standards. Because the Stratix GX device may not be configured before JTAG testing, the I/O pins may not be configured for appropriate electrical standards for chip-to-chip communication. Programming those I/O standards via JTAG allows you to fully test I/O connection to other devices.

The enhanced PLL reconfiguration bits are part of the JTAG chain before configuration and after power-up. After device configuration, the PLL reconfiguration bits are not part of the JTAG chain.

Stratix GX devices also use the JTAG port to monitor the logic operation of the device with the SignalTap® embedded logic analyzer. Stratix GX devices support the JTAG instructions shown in Table 4–33.

Table 4–33. Stratix GX JTAG Instructions (Part 1 of 2)

JTAG Instruction	Description
SAMPLE/PRELOAD	Allows a snapshot of signals at the device pins to be captured and examined during normal device operation, and permits an initial data pattern to be output at the device pins. Also used by the SignalTap® embedded logic analyzer.
EXTEST *(1)*	Allows the external circuitry and board-level interconnects to be tested by forcing a test pattern at the output pins and capturing test results at the input pins.
BYPASS	Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through selected devices to adjacent devices during normal device operation.
USERCODE	Selects the 32-bit USERCODE register and places it between the TDI and TDO pins, allowing the USERCODE to be serially shifted out of TDO.
IDCODE	Selects the IDCODE register and places it between TDI and TDO, allowing the IDCODE to be serially shifted out of TDO.
HIGHZ *(1)*	Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through selected devices to adjacent devices during normal device operation, while tri-stating all of the I/O pins.

Table 4–33. Stratix GX JTAG Instructions (Part 2 of 2)

JTAG Instruction	Description
CLAMP *(1)*	Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through selected devices to adjacent devices during normal device operation while holding I/O pins to a state defined by the data in the boundary-scan register.
ICR instructions	Used when configuring a Stratix GX device through the JTAG port with a MasterBlaster™ or ByteBlasterMV™ download cable, or when using a .jam file or .jbc file with an embedded processor.
PULSE_NCONFIG	Emulates pulsing the nCONFIG pin low to trigger reconfiguration even though the physical pin is unaffected.
CONFIG_IO	Allows the IOE standards to be configured through the JTAG chain. Stops configuration if executed during configuration. Can be executed before or after configuration.
SignalTap instructions	Monitors internal device operation with the SignalTap embedded logic analyzer.

Note to Table 4–33:
(1) Bus hold and weak pull-up resistor features override the high-impedance state of HIGHZ, CLAMP, and EXTEST.

The Stratix GX device instruction register length is 10 bits, and the USERCODE register length is 32 bits. Tables 4–34 and 4–35 show the boundary-scan register length and IDCODE information for Stratix GX devices.

Table 4–34. Stratix GX Boundary-Scan Register Length

Device	Boundary-Scan Register Length
EP1SGX10	1,029
EP1SGX25	1,665
EP1SGX40	1,941

Table 4–35. 32-Bit Stratix GX Device IDCODE (Part 1 of 2)

Device	IDCODE (32 Bits) *(1)*			
	Version (4 Bits)	Part Number (16 Bits)	Manufacturer Identity (11 Bits)	LSB (1 Bit) *(2)*
EP1SGX10	0000	0010 0000 0100 0001	000 0110 1110	1
EP1SGX25	0000	0010 0000 0100 0011	000 0110 1110	1

Table 4–35. 32-Bit Stratix GX Device IDCODE (Part 2 of 2)

Device	IDCODE (32 Bits) *(1)*			
	Version (4 Bits)	Part Number (16 Bits)	Manufacturer Identity (11 Bits)	LSB (1 Bit) *(2)*
EP1SGX40	0000	0010 0000 0100 0101	000 0110 1110	1

Notes to Table 4–35:
(1) The most significant bit (MSB) is at the left end of the string.
(2) The IDCODE's least significant bit (LSB) is always 1.

Figure 4–72 shows the timing requirements for the JTAG signals.

Figure 4–72. Stratix GX JTAG Waveforms

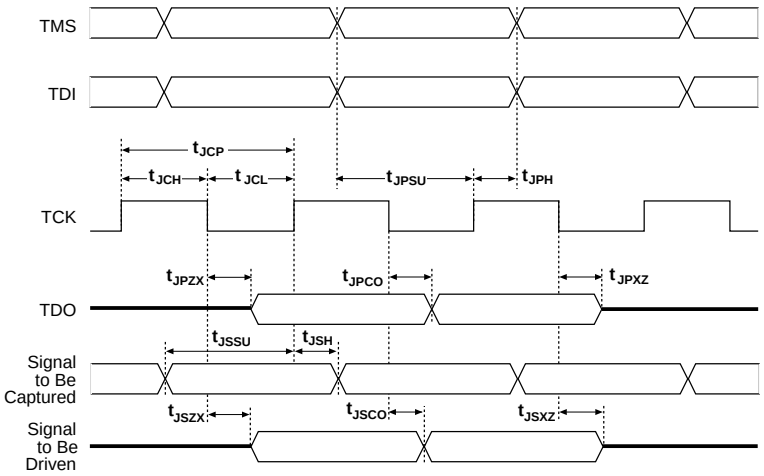


Table 4–36 shows the JTAG timing parameters and values for Stratix GX devices.

Table 4–36. Stratix GX JTAG Timing Parameters & Values (Part 1 of 2)

Symbol	Parameter	Min (ns)	Max (ns)
t_{JCP}	TCK clock period	100	
t_{JCH}	TCK clock high time	50	
t_{JCL}	TCK clock low time	50	
t_{JPSU}	JTAG port setup time	20	

Table 4–36. Stratix GX JTAG Timing Parameters & Values (Part 2 of 2)

Symbol	Parameter	Min (ns)	Max (ns)
t_{JPH}	JTAG port hold time	45	
t_{JPCO}	JTAG port clock to output		25
t_{JPZX}	JTAG port high impedance to valid output		25
t_{JPXZ}	JTAG port valid output to high impedance		25
t_{JSSU}	Capture register setup time	20	
t_{JSH}	Capture register hold time	45	
t_{JSCO}	Update register clock to output		35
t_{JSZX}	Update register high impedance to valid output		35
t_{JSXZ}	Update register valid output to high impedance		35

5. Configuration & Testing

SGX51005-1.0

SignalTap Embedded Logic Analyzer

Stratix® GX devices feature the SignalTap® embedded logic analyzer, which monitors design operation over a period of time through the IEEE Std. 1149.1 (JTAG) circuitry. You can analyze internal logic at speed without bringing internal signals to the I/O pins. This feature is particularly important for advanced packages, such as FineLine BGA® packages, because it can be difficult to add a connection to a pin during the debugging process after a board is designed and manufactured.

Configuration

The logic, circuitry, and interconnects in the Stratix GX architecture are configured with CMOS SRAM elements. Stratix GX devices are reconfigurable and are 100% tested prior to shipment. As a result, you do not have to generate test vectors for fault coverage purposes, and can instead focus on simulation and design verification. In addition, you do not need to manage inventories of different ASIC designs. Stratix GX devices can be configured on the board for the specific functionality required.

Stratix GX devices are configured at system power-up with data stored in an Altera serial configuration device or provided by a system controller. Altera offers in-system programmability (ISP)-capable configuration devices that configure Stratix GX devices via a serial data stream. Stratix GX devices can be configured in under 100 ms using 8-bit parallel data at 100 MHz. The Stratix GX device's optimized interface allows microprocessors to configure it serially or in parallel, and synchronously or asynchronously. The interface also enables microprocessors to treat Stratix GX devices as memory and configure them by writing to a virtual memory location, making reconfiguration easy. After a Stratix GX device has been configured, it can be reconfigured in-circuit by resetting the device and loading new data. Real-time changes can be made during system operation, enabling innovative reconfigurable computing applications.

Operating Modes

The Stratix GX architecture uses SRAM configuration elements that require configuration data to be loaded each time the circuit powers up. The process of physically loading the SRAM data into the device is called configuration. During initialization, which occurs immediately after configuration, the device resets registers, enables I/O pins, and begins to operate as a logic device. The I/O pins are tri-stated during power up,

and before and during configuration. Together, the configuration and initialization processes are called command mode. Normal device operation is called user mode.

A built-in weak pull-up resistor pulls all user I/O pins to V_{CCIO} before and during device configuration.

SRAM configuration elements allow Stratix GX devices to be reconfigured in-circuit by loading new configuration data into the device. With real-time reconfiguration, the device is forced into command mode with a device pin. The configuration process loads different configuration data, reinitializes the device, and resumes user-mode operation. You can perform in-field upgrades by distributing new configuration files either within the system or remotely.

Configuration Schemes

You can load the configuration data for a Stratix GX device with one of five configuration schemes (see Table 5–1), chosen on the basis of the target application. You can use a configuration device, intelligent controller, or the JTAG port to configure a Stratix GX device. A configuration device can automatically configure a Stratix GX device at system power-up.

You can configure multiple Stratix GX devices in any of five configuration schemes by connecting the configuration enable (nCE) and configuration enable output (nCEO) pins on each device.

Table 5–1. Data Sources for Configuration

Configuration Scheme	Data Source
Configuration device	Enhanced or EPC2 configuration device
Passive serial (PS)	ByteBlasterMV™ or MasterBlaster™ download cable or serial data source
Passive parallel asynchronous (PPA)	Parallel data source
Fast passive parallel	Parallel data source
JTAG	MasterBlaster or ByteBlasterMV download cable or a microprocessor with a Jam or JBC file (.jam or .jbc)

Partial Reconfiguration

The enhanced PLLs within the Stratix GX device family support partial reconfiguration of their multiply, divide, and time delay settings without reconfiguring the entire device. You can use either serial data from the logic array or regular I/O pins to program the PLL's counter settings in a serial chain. This option provides considerable flexibility for frequency synthesis, allowing real-time variation of the PLL frequency and delay. The rest of the device is functional while reconfiguring the PLL. See the *Stratix GX Architecture* chapter of the *Stratix GX Device Handbook, Volume 1* for more information on Stratix GX PLLs.

Remote Update Configuration Modes

Stratix GX devices also support remote configuration using an Altera enhanced configuration device (for example, EPC16, EPC8, and EPC4 devices) with page mode selection. Factory configuration data is stored in the default page of the configuration device. This is the default configuration which contains the design required to control remote updates and handle or recover from errors. You write the factory configuration once into the flash memory or configuration device. Remote update data can update any of the remaining pages of the configuration device. If there is an error or corruption in a remote update configuration, the configuration device reverts back to the factory configuration information.

There are two remote configuration modes: remote and local configuration. You can use the remote update configuration mode for all three configuration modes: serial, parallel synchronous, and parallel asynchronous. Configuration devices (for example, EPC16 devices) only support serial and parallel synchronous modes. Asynchronous parallel mode allows remote updates when an intelligent host is used to configure the Stratix GX device. This host must support page mode settings similar to an EPC16 device.

Remote Update Mode

When the Stratix GX device is first powered-up in remote update programming mode, it loads the configuration located at page address `000`. The factory configuration should always be located at page address `000`, and should never be remotely updated. The factory configuration contains the required logic to perform the following operations:

- Determine the page address/load location for the next application's configuration data
- Recover from a previous configuration error

- Receive new configuration data and write it into the configuration device

The factory configuration is the default and takes control if an error occurs while loading the application configuration.

While in the factory configuration, the factory-configuration logic performs the following operations:

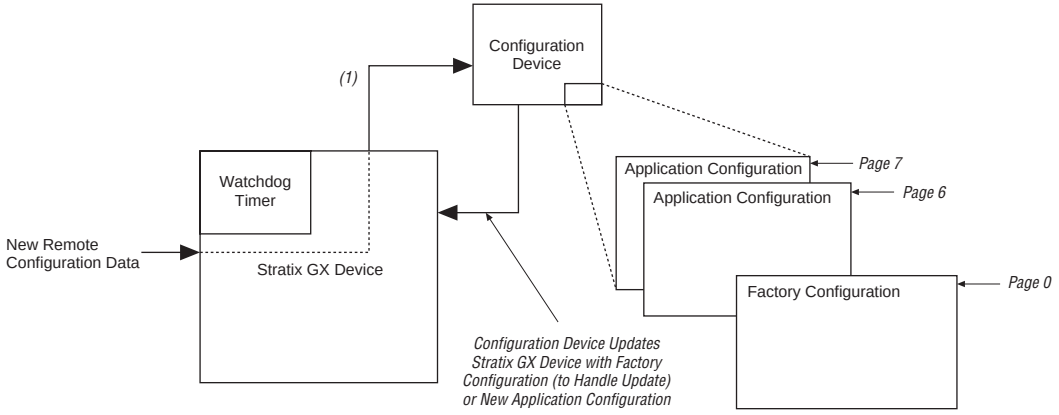
- Loads a remote update-control register to determine the page address of the new application configuration
- Determines whether to enable a user watchdog timer for the application configuration
- Determines what the watchdog timer setting should be if it is enabled

The user watchdog timer is a counter that must be continually reset within a specific amount of time in the user mode of an application configuration to ensure that valid configuration occurred during a remote update. Only valid application configurations designed for remote update can reset the user watchdog timer in user mode. If a valid application configuration does not reset the user watchdog timer in a specific amount of time, the timer updates a status register and loads the factory configuration. The user watchdog timer is automatically disabled for factory configurations.

If an error occurs in loading the application configuration, the configuration logic writes a status register to specify the cause of the reconfiguration. Once this occurs, the Stratix GX device automatically loads the factory configuration, which reads the status register and determines the reason for reconfiguration. Based on the reason, the factory configuration takes appropriate steps and writes the remote update control register to specify the next application configuration page to be loaded.

When the Stratix GX device successfully loads the application configuration, it enters into user mode. The Stratix GX device then executes the main application of the user. Intellectual property (IP), such as a Nios® embedded processor, can help the Stratix GX device determine when remote update is coming. The Nios embedded processor or user logic receives incoming data, writes it to the configuration device, and loads the factory configuration. The factory configuration reads the remote update status register and determine the valid application configuration to load. Figure 5–1 shows the Stratix GX remote update. Figure 5–2 shows the transition diagram for remote update mode.

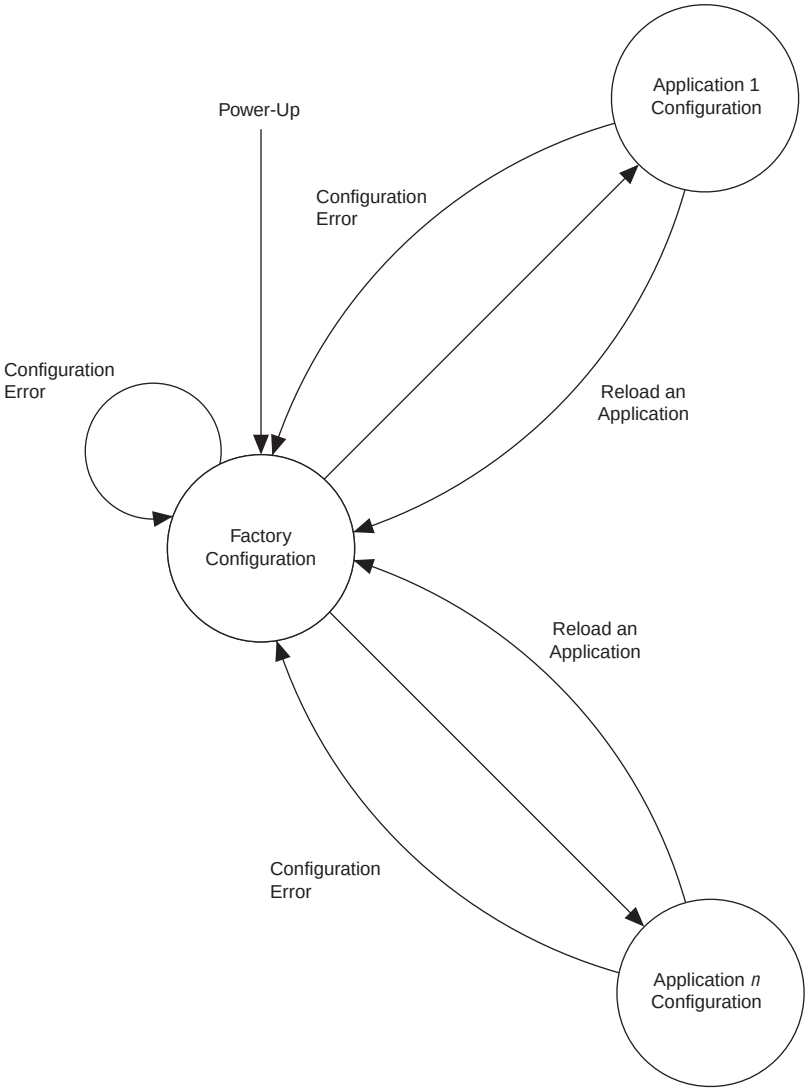
Figure 5–1. Stratix GX Device Remote Update



Note to Figure 5–1:

(1) When the Stratix GX device is configured with the factory configuration, it can handle update data from EPC16, EPC8, or EPC4 configuration device pages and point to the next page in the configuration device.

Figure 5–2. Remote Update Transition Diagram *Notes (1), (2)*



Notes to Figure 5–2:

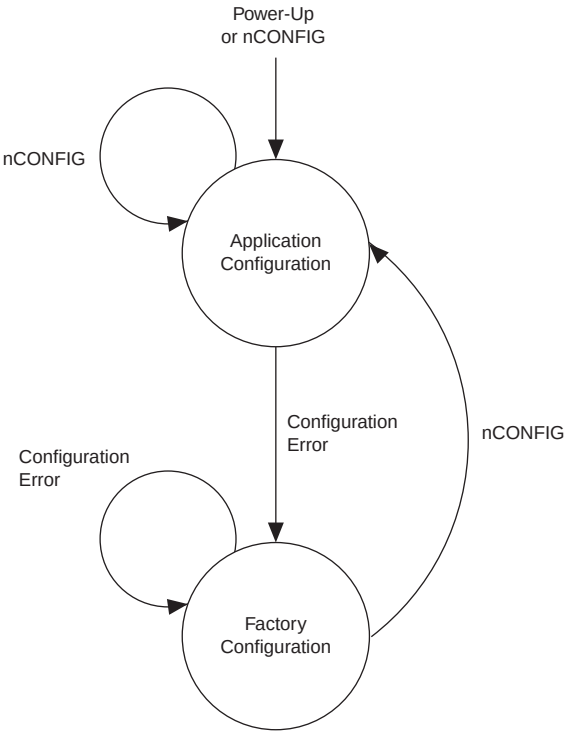
(1) Remote update of application configuration is controlled by a Nios embedded processor or user logic programmed in the factory or application configurations.

(2) Up to seven pages can be specified allowing up to seven different configuration applications.

Local Update Mode

Local update mode is a simplified version of the remote update. This feature is intended for simple systems that need to load a single application configuration immediately upon power-up without loading the factory configuration first. Local update designs have only one application configuration to load, so it does not require a factory configuration to determine which application configuration to use. Figure 5–3 shows the transition diagram for local update mode.

Figure 5–3. Local Update Transition Diagram



Stratix GX Automated Single Event Upset (SEU) Detection

Stratix GX devices offer on-chip circuitry for automated checking of single event upset (SEU) detection. Some applications that require the device to operate error free at high elevations or in close proximity to earth's North or South Pole require periodic checks to ensure continued data integrity. The error detection cyclic redundancy code (CRC) feature controlled by the **Device & Pin Options** dialog box in the Quartus II software uses a 32-bit CRC circuit to ensure data reliability and is one of the best options for mitigating SEU.

You can implement the error detection CRC feature with existing circuitry in Stratix GX devices, eliminating the need for external logic. For Stratix GX devices, the CRC is computed by Quartus II and downloaded into the device as a part of the configuration bit stream. The CRC_ERROR pin reports a soft error when configuration SRAM data is corrupted, triggering device reconfiguration.

Custom-Built Circuitry

Dedicated circuitry is built into Stratix GX devices to perform error detection automatically. This error detection circuitry constantly checks for errors in the configuration SRAM cells while the device is in user mode. You can monitor one external pin for the error and use it to trigger a reconfiguration cycle. You can select the desired time between checks by adjusting a built-in clock divider.

Software Interface

In the Quartus II software version 4.1 and later, you can turn on the automated error detection CRC feature in the **Device & Pin Options** dialog box. This dialog box allows you to enable the feature and set the internal frequency of the CRC between 400 kHz to 100 MHz. This controls the rate that the CRC circuitry verifies the internal configuration SRAM bits in the FPGA device.

For more information on CRC, refer to *AN 357: Error Detection Using CRC in Altera FPGA Devices*.

Temperature-Sensing Diode

Stratix GX devices include a diode-connected transistor for use as a temperature sensor in power management. This diode is used with an external digital thermometer device such as a MAX1617A or MAX1619 from MAXIM Integrated Products. These devices steer bias current through the Stratix GX diode, measuring forward voltage and converting this reading to temperature in the form of an 8-bit signed number (7 bits plus sign). The external device's output represents the package temperature of the Stratix GX device and can be used for intelligent power management.

The diode requires two pins (tempdiodep and tempdioden) on the Stratix GX device to connect to the external temperature-sensing device, as shown in Figure 5–4. The temperature-sensing diode is a passive element and therefore can be used before the Stratix GX device is powered.

Figure 5–4. External Temperature-Sensing Diode

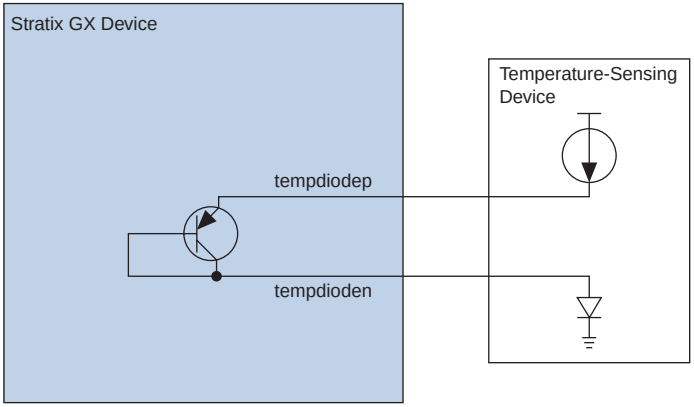


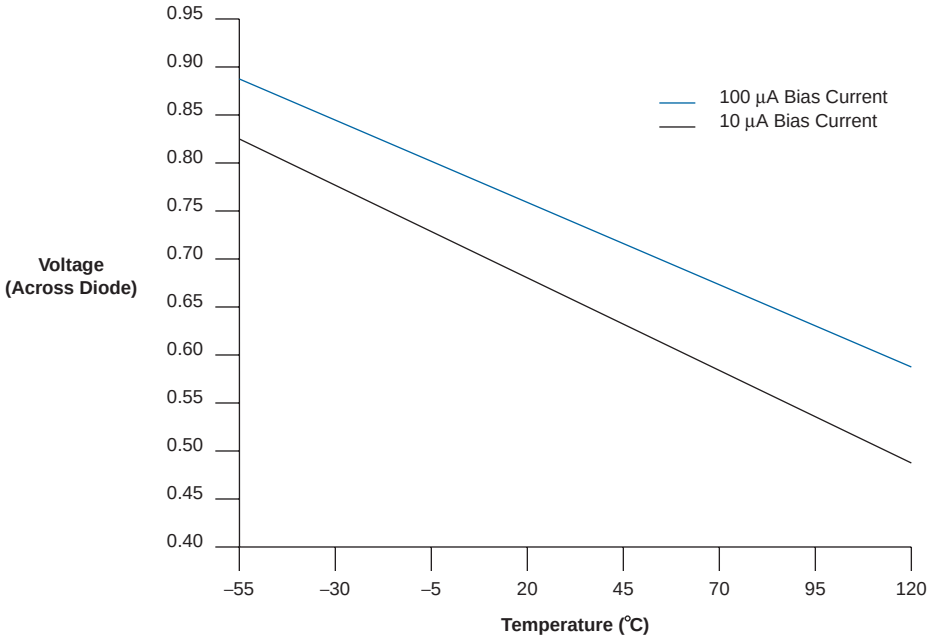
Table 5–2 shows the specifications for bias voltage and current of the Stratix GX temperature-sensing diode.

Table 5–2. Temperature-Sensing Diode Electrical Characteristics

Parameter	Minimum	Typical	Maximum	Units
I_{BIAS} high	80	100	120	µA
I_{BIAS} low	8	10	12	µA
$V_{BP} - V_{BN}$	0.3		0.9	V
V_{BN}		0.7		V
Series resistance			3	W

The temperature-sensing diode works for the entire operating range shown in Figure 5–5.

Figure 5–5. Temperature Versus Temperature-Sensing Diode Voltage



6. DC & Switching Characteristics

SGX51006-1.2

Operating Conditions

Stratix® GX devices are offered in both commercial and industrial grades. However, industrial-grade devices may have limited speed-grade availability.

Tables 6–1 through 6–12 provide information on absolute maximum ratings, recommended operating conditions, DC operating conditions, and transceiver block absolute maximum ratings. Notes for Tables 6–1 through 6–6 immediately follow Table 6–6, notes for Table 6–7 immediately follow that table, and notes for Tables 6–8 through 6–12 immediately follow Table 6–12.

Table 6–1. Stratix GX Device Absolute Maximum Ratings *Notes (1), (2)*

Symbol	Parameter	Conditions	Minimum	Maximum	Unit
V_{CCINT}	Supply voltage	With respect to ground *(3)*	–0.5	2.4	V
V_{CCIO}			–0.5	4.6	V
V_I	DC input voltage		–0.5	4.6	V
I_{OUT}	DC output current, per pin		–25	25	mA
T_{STG}	Storage temperature	No bias	–65	150	° C
T_{AMB}	Ambient temperature	Under bias	–65	135	° C
T_J	Junction temperature	BGA packages under bias		135	° C

Table 6–2. Stratix GX Device Recommended Operating Conditions (Part 1 of 2) *Note (7), (12), (13)*

Symbol	Parameter	Conditions	Minimum	Maximum	Unit
V_{CCINT}	Supply voltage for internal logic and input buffers	*(4)*	1.425	1.575	V
V_{CCIO}	Supply voltage for output buffers, 3.3-V operation	*(4), (5)*	3.00 (3.135)	3.60 (3.465)	V
	Supply voltage for output buffers, 2.5-V operation	*(4)*	2.375	2.625	V
	Supply voltage for output buffers, 1.8-V operation	*(4)*	1.71	1.89	V
	Supply voltage for output buffers, 1.5-V operation	*(4)*	1.4	1.6	V
V_I	Input voltage	*(3), (6)*	–0.5	4.1	V

Table 6–2. Stratix GX Device Recommended Operating Conditions (Part 2 of 2) *Note (7), (12), (13)*

Symbol	Parameter	Conditions	Minimum	Maximum	Unit
V_O	Output voltage		0	V_{CCIO}	V
T_J	Operating junction temperature	For commercial use	0	85	° C
		For industrial use	–40	100	° C

Table 6–3. Stratix GX Device DC Operating Conditions *Note (12)*

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
I_I	Input pin leakage current	$V_I = V_{CCIOmax}$ to 0 V *(8)*	–10		10	µA
I_{OZ}	Tri-stated I/O pin leakage current	$V_O = V_{CCIOmax}$ to 0 V *(8)*	–10		10	µA
R_{CONF}	Value of I/O pin pull-up resistor before and during configuration	V_{CCIO} = 3.0 V *(9)*	20		50	kΩ
		V_{CCIO} = 2.375 V *(9)*	30		80	kΩ
		V_{CCIO} = 1.71 V *(9)*	60		150	kΩ

Table 6–4. Stratix GX Transceiver Block Absolute Maximum Ratings

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{CCA}	Transceiver block supply voltage	Commercial and industrial	–0.5	4.6	V
V_{CCP}	Transceiver block supply voltage	Commercial and industrial	–0.5	2.4	V
V_{CCR}	Transceiver block supply Voltage	Commercial and industrial	–0.5	2.4	V
V_{CCT}	Transceiver block supply voltage	Commercial and industrial	–0.5	2.4	V
V_{CCG}	Transceiver block supply voltage	Commercial and industrial	–0.5	2.4	V
Receiver input voltage	V_{ICM} ±V_{ID} single / 2	Commercial and industrial		1.675 *(10)*, *(13)*	V
`refclkb` input voltage	V_{ICM} ±V_{ID} single / 2	Commercial and industrial		1.675 *(10)*, *(13)*	V

Table 6–5. Stratix GX Transceiver Block Operating Conditions

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCA}	Transceiver block supply voltage	Commercial and industrial	3.135	3.3	3.465	V
V_{CCP}	Transceiver block supply voltage	Commercial and industrial	1.425	1.5	1.575	V
V_{CCR}	Transceiver block supply voltage	Commercial and industrial	1.425	1.5	1.575	V
V_{CCT}	Transceiver block supply voltage	Commercial and industrial	1.425	1.5	1.575	V
V_{CCG}	Transceiver block supply voltage	Commercial and industrial	1.425	1.5	1.575	V
V_{ID} (differential p-p)	Receiver input differential voltage swing	Commercial and industrial	170		2,000	mV
	`refclkb` input differential voltage swing	Commercial and industrial	400		2,000	mV
V_{ICM}	Receiver input common mode voltage	Commercial and industrial	1,025	1,100	1,175	mV
V_{OD} (differential p-p)	Transmitter output differential voltage	Commercial and industrial	350		1,600	mV
V_{OCM}	Transmitter output common mode voltage	Commercial and industrial		750		mV
R_{REF} *(11)*	Reference resistor	Commercial and industrial	2K –1%	2K	2K +1%	Ω

Table 6–6. Stratix GX Transceiver Block On-Chip Termination (Part 1 of 2)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Rx	Receiver termination	Commercial and industrial, 100-Ω setting	103	108	113	Ω
		Commercial and industrial, 120-Ω setting	120	128	134	Ω
		Commercial and industrial, 150-Ω setting	149	158	167	Ω
Tx	Transmitter termination	Commercial and industrial, 100-Ω setting	103	108	113	Ω
		Commercial and industrial, 120-Ω setting	120	128	134	Ω
		Commercial and industrial, 150-Ω setting	149	158	167	Ω

Table 6–6. Stratix GX Transceiver Block On-Chip Termination (Part 2 of 2)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Refclkb	Dedicated transceiver clock termination	Commercial and industrial, 100-Ω setting	103	108	113	Ω
		Commercial and industrial, 120-Ω setting	120	128	134	Ω
		Commercial and industrial, 150-Ω setting	149	158	167	Ω

Notes to Tables 6–1 through 6–6:

(1) See the *Operating Requirements for Altera Devices Data Sheet*.
(2) Conditions beyond those listed in Table 6–1 may cause permanent damage to a device. Additionally, device operation at the absolute maximum ratings for extended periods of time may have adverse affects on the device.
(3) Minimum DC input is –0.5 V. During transitions, the inputs may undershoot to –2.0 V or overshoot to 4.6 V for input currents less than 100 mA and periods shorter than 20 ns. (The information in this note does not include the transceiver pins. See note 13 for information about the transient voltage on the transceiver pins.)
(4) Maximum V_{CC} rise time is 100 ms, and V_{CC} must rise monotonically.
(5) V_{CCIO} maximum and minimum conditions for LVPECL, LVDS, and 3.3-V PCML are shown in parentheses.
(6) All pins, including dedicated inputs, clock, I/O, and JTAG pins, may be driven before V_{CCINT} and V_{CCIO} are powered.
(7) Typical values are for T_A = 25° C, V_{CCINT} = 1.5 V, and V_{CCIO} = 1.5 V, 1.8 V, 2.5 V, and 3.3 V.
(8) This value is specified for normal device operation. The value may vary during power-up. This applies for all V_{CCIO} settings (3.3, 2.5, 1.8, and 1.5 V).
(9) Pin pull-up resistance values decrease if an external source drives the pin higher than V_{CCIO}.
(10) The device can tolerate prolonged operation at this absolute maximum, as long as the maximum specification is not violated.
(11) Each usable quad requires its own R_{REF} resistor path to ground. For example, the "D" in the EP1SGX25DC1020 device code means it has two usable quad so two different R_{REF} pins must be connected to a R_{REF} resistor each to ground. The DC signal on the R_{REF} pin must be as clean as possible. Ensure that no noise is coupled to this pin.
(12) The Stratix GX device's recommended operating conditions do not include the transceiver. Refer to Tables 6–4 to 6–7.
(13) Minimum DC input to the transceiver pins is –0.5 V. During transitions, the transceiver pins may undershoot to –0.5 V or overshoot to 3.5 V for input currents less than 100 mA and periods shorter than 20 ns.

Table 6–7. Stratix GX Transceiver Block AC Specification (Part 1 of 7)

Symbol / Description	Conditions	-5 Commercial Speed Grade *(1)*			-6 Commercial & Industrial Speed Grade *(1)*			-7 Commercial & Industrial Speed Grade *(1)*			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Power per quadrant (PCS + PMA)	3.125 Gbps, 400-mV V_{od} 0 pre-emphasis		450			450					mW

Table 6–7. Stratix GX Transceiver Block AC Specification (Part 2 of 7)

Symbol / Description	Conditions	-5 Commercial Speed Grade *(1)*			-6 Commercial & Industrial Speed Grade *(1)*			-7 Commercial & Industrial Speed Grade *(1)*			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Reference Clock											
Jitter tolerance (peak-to-peak)	Jitter components <20 MHz			20			20			20	ps
	Wideband			50			50			50	ps
Reference input clock frequency	Dedicated `refclkb` pins	25		650	25		650	25		312.5	MHz
	PLD clock resources	25		325	25		325	25		156.25	MHz
Receiver											
Serial data rate (general)	Commercial / industrial	614		3,187.5	614		3,187.5	614		2,500	Mbps
Serial data rate (8B/10B encoded)	Commercial / industrial	500		3,187.5	500		3,187.5	500		2,500	Mbps
Parallel transceiver/ logic array interface speed		20		398.4	20		375	20		312.5	MHz
Rate matching frequency tolerance	XAUI mode only			±100			±100			±100	ppm
8B/10B Custom Receiver Jitter Tolerance using Encoded CJPAT *Note (2)*											
Deterministic jitter	500 Mbps			0.45			0.45			0.45	UI
Total jitter	500 Mbps			0.71			0.71			0.71	UI
Fibre Channel Receiver Jitter Tolerance using 8B/10B Encoded CJTPAT *Note (2)*											
Deterministic jitter	1.0625 Gbps			0.37			0.37			0.37	UI
Total jitter	1.0625 Gbps			0.68			0.68			0.68	UI

Table 6–7. Stratix GX Transceiver Block AC Specification (Part 3 of 7)

Symbol / Description	Conditions	-5 Commercial Speed Grade *(1)*			-6 Commercial & Industrial Speed Grade *(1)*			-7 Commercial & Industrial Speed Grade *(1)*			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Sinusoidal jitter	f = 42.5 kHz at 1.0625 Gbps			1.5			1.5			1.5	UI
	f = 637 kHz at 1.0625 Gbps			0.1			0.1			0.1	UI
Deterministic jitter	2.125 Gbps			0.33			0.33			0.33	UI
Total jitter	2.125 Gbps			0.62			0.62			0.62	UI
Sinusoidal jitter	f = 85 kHz at 2.125 Gbps			1.5			1.5			1.5	UI
	f = 1,274 kHz at 2.125 Gbps			0.1			0.1			0.1	UI
Serial Rapid I/O Receiver Jitter Tolerance using 8B/10B Encoded CJPAT *Note (2)*											
Deterministic jitter	1.25 Gbps			0.45			0.45			0.45	UI
Total jitter	1.25 Gbps			0.71			0.71			0.71	UI
Deterministic jitter	2.5 Gbps			0.41			0.41			0.41	UI
Total jitter	2.5 Gbps			0.65			0.65			0.65	UI
Deterministic jitter	3.125 Gbps			0.36			0.36			N/A	UI
Total jitter	3.125 Gbps			0.60			0.60			N/A	UI
SONET Receiver Jitter Tolerance using PRBS23 *Note (2)*											
Sinusoidal jitter	f = 6 kHz at 2.48832 Gbps			1.5			1.5			1.5	UI
	f = 1 MHz at 2.48832 Gbps			0.15			0.15			0.15	UI
XAUI Receiver Jitter Tolerance using 8B/10B Encoded CJPAT *Note (2)*											
Deterministic jitter	3.125 Gbps			0.37			0.37			N/A	UI
Total jitter	3.125 Gbps			0.65			0.65			N/A	UI

Table 6–7. Stratix GX Transceiver Block AC Specification (Part 4 of 7)

Symbol / Description	Conditions	-5 Commercial Speed Grade *(1)*			-6 Commercial & Industrial Speed Grade *(1)*			-7 Commercial & Industrial Speed Grade *(1)*			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Sinusoidal jitter	f = 22.1 kHz at 3.125 Gbps			8.5			8.5			N/A	
	f = 1.875 MHz at 3.125 Gbps			0.1			0.1			N/A	
	f = 20 MHz at 3.125 Gbps			0.1			0.1			N/A	
BER *(12)*				10^{-12}			10^{-12}			10^{-12}	
Receive latency *(4)*	Single width	7		32	7		32	7		32	*(3)*
	Double width	5		19	5		19	5		19	*(3)*
Channel to channel bit skew tolerance *(5)*, *(6)*	XAUI mode / inter-quadrant only			40			40			N/A	UI *(7)*
Run-length	*(8)*			80			80			80	UI
Receive return loss (differential)	100 MHz to 2.5 Ghz			–10			–10			–10	dB
Receive return loss (common mode)	100 MHz to 2.5 Ghz			–6			–6			–6	dB
Transmitter											
Serial data rate	Commercial / industrial	500		3,187.5	500		3,187.5	500		2,500	Mbps
Parallel transceiver/ core interface speed		20		398.4	20		375	20		312.5	MHz
8B/10B Custom Transmitter Jitter using Encoded CRPAT *Note (9)*											
Deterministic jitter	500 Mbps Pre-emphasis = 1 V_{OD} = 1,400 mV			0.11			0.11			0.11	UI
Total jitter				0.18			0.18			0.18	UI

Table 6–7. Stratix GX Transceiver Block AC Specification (Part 5 of 7)

Symbol / Description	Conditions	-5 Commercial Speed Grade (1)			-6 Commercial & Industrial Speed Grade (1)			-7 Commercial & Industrial Speed Grade (1)			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Fibre Channel Transmitter Jitter using 8B/10B Encoded CRPAT *Note (9)*											
Deterministic jitter	1.0625 Gbps Pre-emphasis = 0 V_{OD} = 1,200 mV			0.09			0.09			0.09	UI
Total jitter				0.17			0.17			0.17	UI
Deterministic jitter	2.125 Gbps Pre-emphasis= 1 V_{OD} = 1,200 mV			0.16			0.16			0.16	UI
Total jitter				0.33			0.33			0.33	UI
Serial Rapid I/O Short Run Transmitter Jitter using 8B/10B Encoded CRPAT *Note (9)*											
Deterministic jitter	1.25 Gbps Pre-emphasis = 1 V_{OD} = 1,600 mV			0.09			0.09			0.09	UI
Total jitter				0.17			0.17			0.17	UI
Deterministic jitter	2.5 Gbps Pre-emphasis = 1 V_{OD} = 800 mV			0.15			0.15			0.15	UI
Total jitter				0.32			0.32			0.32	UI
Deterministic jitter	3.125 Gbps Pre-emphasis = 1 V_{OD} = 800 mV			0.15			0.15			N/A	UI
Total jitter				0.32			0.32			N/A	UI
Serial Rapid I/O Long Run Transmitter Jitter using 8B/10B Encoded CRPAT *Note (9)*											
Deterministic jitter	1.25 Gbps Pre-emphasis = 1 V_{OD} = 1,600 mV			0.09			0.09			0.09	UI
Total jitter				0.17			0.17			0.17	UI
Deterministic jitter	2.5 Gbps Pre-emphasis = 2 V_{OD} = 1,400 mV			0.18			0.18			0.18	UI
Total jitter				0.35			0.35			0.35	UI
Deterministic jitter	3.125 Gbps Pre-emphasis = 2 V_{OD} = 1,400 mV			0.20			0.20			N/A	UI
Total jitter				0.37			0.37			N/A	UI
SONET Transmitter Jitter PRBS23 *Note (9)*											
Total jitter	2.48832 Gbps Pre-emphasis = 1 V_{OD} = 800 mV			0.20			0.20			0.20	UI

Table 6–7. Stratix GX Transceiver Block AC Specification (Part 6 of 7)

Symbol / Description	Conditions	-5 Commercial Speed Grade *(1)*			-6 Commercial & Industrial Speed Grade *(1)*			-7 Commercial & Industrial Speed Grade *(1)*			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
XAUI Transmitter Jitter using 8B/10B Encoded CJPAT *Note (9)*											
Deterministic jitter	3.125 Gbps Pre-emphasis = 0 V_{OD} = 1,200 mV			0.15			0.15			N/A	UI
Total jitter				0.32			0.32			N/A	UI
Jitter transfer bandwidth *(10)*	Low bandwidth setting at 3.125 Gbps		3			3				N/A	MHz
	High bandwidth setting at 3.125 Gbps		4.7			4.7				N/A	MHz
	Low bandwidth setting at 2.5 Gbps		3.2			3.2				3.2	MHz
	High bandwidth setting at 2.5 Gbps		4.3			4.3				4.3	MHz
Output t_{RISE}	20% to 80%	60		130	60		130	60		130	ps
Output t_{FALL}	80% to 20%	60		130	60		130	60		130	ps
Transmit latency *(11)*	Single width	3		8	3		8	3		8	*(3)*
	Double width	3		7	3		7	3		7	*(3)*
Intra differential pair skew				10			10			10	ps
Channel to channel skew	Within a single quadrant			50			50			50	ps

Table 6–7. Stratix GX Transceiver Block AC Specification (Part 7 of 7)

Symbol / Description	Conditions	-5 Commercial Speed Grade (1)			-6 Commercial & Industrial Speed Grade (1)			-7 Commercial & Industrial Speed Grade (1)			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Output return loss	100 MHz to 2.5 GHz	–10			–10			–10			dB

Notes to Table 6–7:

(1) All numbers for the -6 and -7 speed grades are for both commercial and industrial unless specified otherwise in the Conditions column. Speed grade -5 is available only for commercial specifications.
(2) Not all V_{ID} and equalizer values will get the same results. The condition for the specification was that the V_{ID} before jitter was added is 1,000 mV and the equalizer was set to the maximum condition of 111 (equalizer control setting = 4 in the MegaWizard Plug-In Manager).
(3) Number of parallel clocks.
(4) Receive latency delay from serial receiver indata to parallel receiver data.
(5) Per IEEE Standard 802.3ae @ 3.125 for –5 and –6.
(6) The specification is for channel aligner tolerance.
(7) UI = Unit Interval.
(8) Run-length conditions are true for all data rates, but the average transition density must be enough to keep the receiver phase aligned and the overall data must be DC balanced.
(9) Not all combinations of V_{OD} and pre-emphasis will get the same results.
(10) The numbers are for 3.125-Gbps data rate for –5 and –6 devices and 2.5 Gbps for –7 devices.
(11) Transmitter latency delay from parallel transceiver data to serial transceiver out data.
(12) The receiver operates with a BER of better than 10^{-12} in the presence of an input signal as defined in the XAUI driver template for 3.125 Gbps and in the PCI Exp transmitter eye mask for 2.5 Gbps.

Table 6–8. LVTTL Specifications

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{CCIO}	Output supply voltage		3.0	3.6	V
V_{IH}	High-level input voltage		1.7	4.1	V
V_{IL}	Low-level input voltage		–0.5	0.7	V
V_{OH}	High-level output voltage	I_{OH} = –4 to –24 mA (1)	2.4		V
V_{OL}	Low-level output voltage	I_{OL} = 4 to 24 mA (1)		0.45	V

Table 6–9. LVCMOS Specifications

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{CCIO}	Output supply voltage		3.0	3.6	V
V_{IH}	High-level input voltage		1.7	4.1	V
V_{IL}	Low-level input voltage		–0.5	0.7	V

Table 6–9. LVCMOS Specifications

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{OH}	High-level output voltage	V_{CCIO} = 3.0, I_{OH} = –0.1 mA	V_{CCIO} – 0.2		V
V_{OL}	Low-level output voltage	V_{CCIO} = 3.0, I_{OL} = 0.1 mA		0.2	V

Table 6–10. 2.5-V I/O Specifications *Note (1)*

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{CCIO}	Output supply voltage		2.375	2.625	V
V_{IH}	High-level input voltage		1.7	4.1	V
V_{IL}	Low-level input voltage		–0.5	0.7	V
V_{OH}	High-level output voltage	I_{OH} = –0.1 mA	2.1		V
		I_{OH} = –1 mA	2.0		V
		I_{OH} = –2 to –16 mA *(1)*	1.7		V
V_{OL}	Low-level output voltage	I_{OL} = 0.1 mA		0.2	V
		I_{OH} = 1 mA		0.4	V
		I_{OH} = 2 to 16 mA *(1)*		0.7	V

Table 6–11. 1.8-V I/O Specifications

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{CCIO}	Output supply voltage		1.65	1.95	V
V_{IH}	High-level input voltage		0.65 × V_{CCIO}	2.25	V
V_{IL}	Low-level input voltage		–0.3	0.35 × V_{CCIO}	V
V_{OH}	High-level output voltage	I_{OH} = –2 to –8 mA *(1)*	V_{CCIO} – 0.45		V
V_{OL}	Low-level output voltage	I_{OL} = 2 to 8 mA *(1)*		0.45	V

Table 6–12. 1.5-V I/O Specifications (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{CCIO}	Output supply voltage		1.4	1.6	V
V_{IH}	High-level input voltage		0.65 × V_{CCIO}	V_{CCIO} + 0.3	V
V_{IL}	Low-level input voltage		–0.3	0.35 × V_{CCIO}	V
V_{OH}	High-level output voltage	I_{OH} = –2 mA *(1)*	0.75 × V_{CCIO}		V

Table 6–12. 1.5-V I/O Specifications (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{OL}	Low-level output voltage	I_{OL} = 2 mA *(1)*		0.25 × V_{CCIO}	V

Note to Tables 6–8 through 6–12:

(1) Drive strength is programmable according to values in found in the *Stratix GX Architecture* chapter of the *Stratix GX Device Handbook, Volume 1*.

Figures 6–1 through 6–3 show receiver input and transmitter output waveforms, respectively, for all differential I/O standards (LVDS, 3.3-V PCML, LVPECL, and HyperTransport technology).

Figure 6–1. Receiver Input Waveforms for Differential I/O Standards

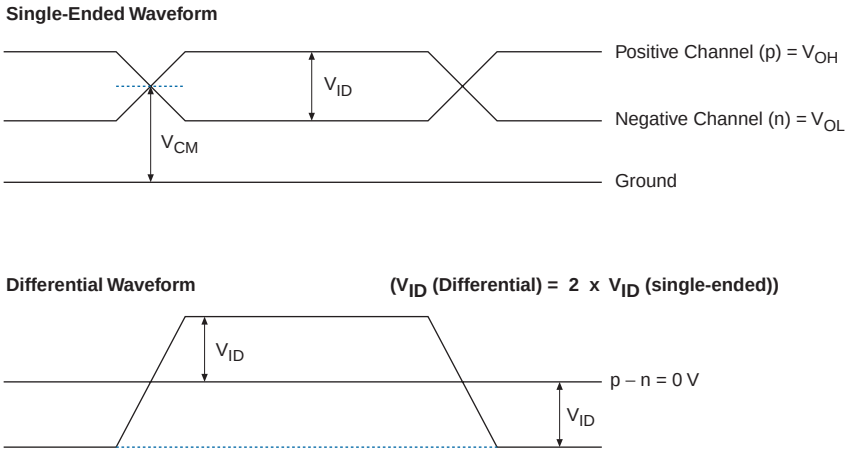
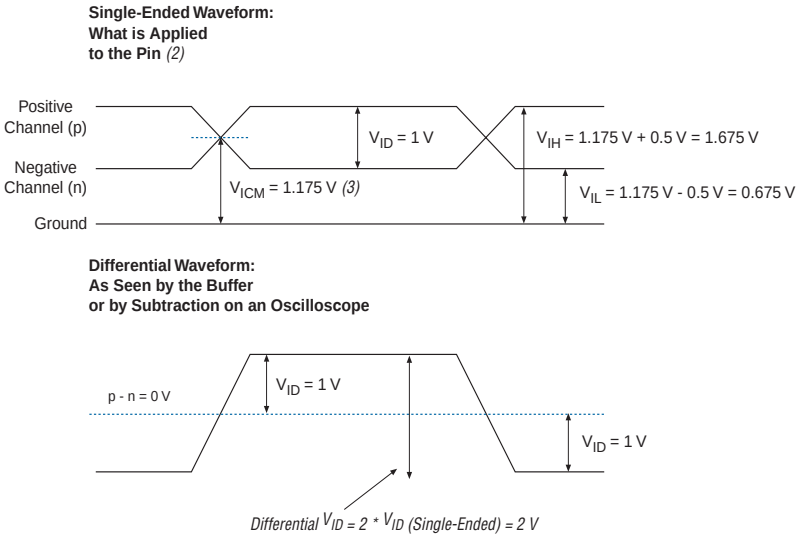


Figure 6–2. Receiver Input Waveform Example with Values



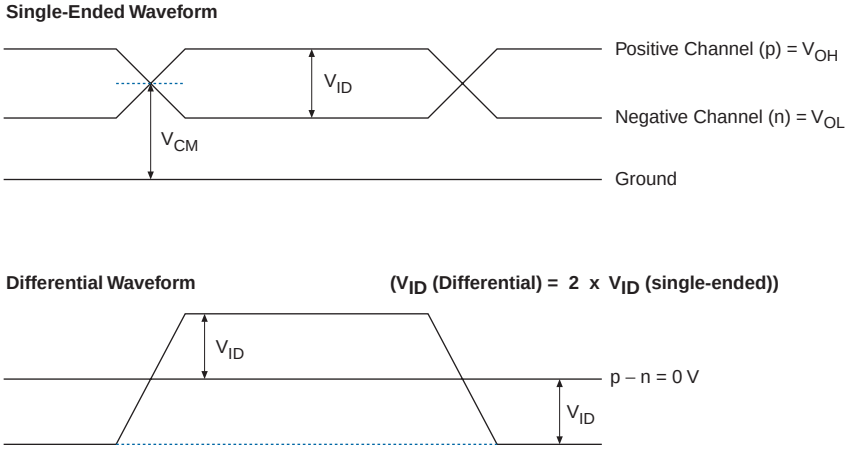
Notes to Figure 6–2:

(1) The values in this figure are for example only.

(2) These values must meet the voltages specified in the section "Operating Conditions" on page 6–1.

(3) If internal termination is used, the common mode is generated after the pins.

Figure 6–3. Transmitter Output Waveforms for Differential I/O Standards



Tables 6–13 through 6–33 provide information about specifications and bus hold parameters for 1.5-V Stratix GX devices. Notes for Tables 6–14 through 6–33 immediately follow Table 6–33.

Table 6–13. 3.3-V LVDS I/O Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	I/O supply voltage		3.135	3.3	3.465	V
V_{ID} *(1)*	Input differential voltage swing (single-ended)	0.1 V < V_{CM} < 1.1 V *W* = 1 through 10	300		1,000	mV
		1.1 V < V_{CM} < 1.6 V *W* = 1	200		1,000	mV
		1.1 V < V_{CM} < 1.6 V *W* = 2 through 10	100		1,000	mV
		1.6 V < V_{CM} < 1.8 V *W* = 1 through 10	300		1,000	mV
V_{ICM} *(1)*	Input common-mode voltage	LVDS 0.3 V < V_{ID} < 1.0 V *W* = 1 through 10	100		1,100	mV
		LVDS 0.3 V < V_{ID} < 1.0 V *W* = 1 through 10	1,600		1,800	mV
		LVDS 0.2 V < V_{ID} < 1.0 V *W* = 1	1,100		1,600	mV
		LVDS 0.1 V < V_{ID} < 1.0 V *W* = 2 through 10	1,100		1,600	mV
V_{OD}	Differential output voltage (single ended)	R_L = 100 Ω	250	375	550	mV
ΔV_{OD}	Change in V_{OD} between high and low	R_L = 100 Ω			50	mV
V_{OCM}	Output common-mode voltage	R_L = 100 Ω	1,125	1,200	1,375	mV
ΔV_{OCM}	Change in V_{OCM} between high and low	R_L = 100 Ω			50	mV
R_L	Receiver differential input resistor, external		90	100	110	Ω

Note to Table 6–13:
(1) For up to 1 Gbps in DPA mode and 840 Mbps in non-DPA mode

Table 6–14. 3.3-V PCML Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	I/O supply voltage		3.135	3.3	3.465	V
V_{ID}	Input differential voltage swing (single-ended)		300		600	mV
V_{ICM}	Input common mode voltage		1.5		3.465	V
V_{OD}	Output differential voltage (single-ended)		300	370	500	mV
ΔV_{OD}	Change in V_{OD} between high and low				50	mV
V_{OCM}	Output common mode voltage		2.5	2.85	3.3	V
ΔV_{OCM}	Change in V_{OCM} between high and low				50	mV
V_T	Output termination voltage			V_{CCIO}		V
R_1	Output external pull-up resistors		45	50	55	Ω
R_2	Output external pull-up resistors		45	50	55	Ω

Table 6–15. LVPECL Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	I/O supply voltage		3.135	3.3	3.465	V
V_{ID}	Input differential voltage swing (single-ended)		300		1,000	mV
V_{ICM}	Input common mode voltage		1		2	V
V_{OD}	Differential output voltage (single ended)	R_L = 100 Ω	525	700	970	mV
V_{OCM}	Output common mode voltage	R_L = 100 Ω	1.5	1.7	1.9	V
R_L	Receiver differential input resistor, external		90	100	110	Ω

Table 6–16. HyperTransport Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	I/O supply voltage		2.375	2.5	2.625	V
V_{OD}	Differential output voltage (single ended)	R_L = 100 Ω	380	485	820	mV
ΔV_{OD}	Change in between high and low	R_L = 100 Ω			50	mV
V_{OCM}	Output common mode voltage	R_L = 100 Ω	440	650	780	mV
ΔV_{OCM}	Change in between high and low	R_L = 100 Ω			50	mV
V_{ID}	Differential input voltage swing (single-ended)		300		900	mV
V_{ICM}	Input common mode voltage		300		900	mV
R_L	Receiver differential input resistor, external		90	100	110	Ω

Table 6–17. 3.3-V PCI Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		3.0	3.3	3.6	V
V_{IH}	High-level input voltage		0.5 × V_{CCIO}		V_{CCIO} + 0.5	V
V_{IL}	Low-level input voltage		–0.5		0.3 × V_{CCIO}	V
V_{OH}	High-level output voltage	I_{OUT} = –500 µA	0.9 × V_{CCIO}			V
V_{OL}	Low-level output voltage	I_{OUT} = 1,500 µA			0.1 × V_{CCIO}	V

Table 6–18. PCI-X Specifications (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		3.0		3.6	V
V_{IH}	High-level input voltage		0.5 × V_{CCIO}		V_{CCIO} + 0.5	V
V_{IL}	Low-level input voltage		–0.5		0.35 × V_{CCIO}	V

Table 6–18. PCI-X Specifications (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{IPU}	Input pull-up voltage		0.7 × V_{CCIO}			V
V_{OH}	High-level output voltage	I_{OUT} = –500 µA	0.9 × V_{CCIO}			V
V_{OL}	Low-level output voltage	I_{OUT} = 1,500 µA			0.1 × V_{CCIO}	V

Table 6–19. GTL+ I/O Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{TT}	Termination voltage		1.35	1.5	1.65	V
V_{REF}	Reference voltage		0.88	1.0	1.12	V
V_{IH}	High-level input voltage		V_{REF} + 0.1			V
V_{IL}	Low-level input voltage				V_{REF} – 0.1	V
V_{OL}	Low-level output voltage	I_{OL} = 36 mA *(1)*			0.65	V

Table 6–20. GTL I/O Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{TT}	Termination voltage		1.14	1.2	1.26	V
V_{REF}	Reference voltage		0.74	0.8	0.86	V
V_{IH}	High-level input voltage		V_{REF} + 0.05			V
V_{IL}	Low-level input voltage				V_{REF} – 0.05	V
V_{OL}	Low-level output voltage	I_{OL} = 40 mA *(1)*			0.4	V

Table 6–21. SSTL-18 Class I Specifications (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		1.65	1.8	1.95	V
V_{REF}	Reference voltage		0.8	0.9	1.0	V
V_{TT}	Termination voltage		V_{REF} – 0.04	V_{REF}	V_{REF} + 0.04	V
$V_{IH(DC)}$	High-level DC input voltage		V_{REF} + 0.125			V

Table 6–21. SSTL-18 Class I Specifications (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
$V_{IL(DC)}$	Low-level DC input voltage				V_{REF} – 0.125	V
$V_{IH(AC)}$	High-level AC input voltage		V_{REF} + 0.275			V
$V_{IL(AC)}$	Low-level AC input voltage				V_{REF} – 0.275	V
V_{OH}	High-level output voltage	I_{OH} = –6.7 mA *(1)*	V_{TT} + 0.475			V
V_{OL}	Low-level output voltage	I_{OL} = 6.7 mA *(1)*			V_{TT} – 0.475	V

Table 6–22. SSTL-18 Class II Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		1.65	1.8	1.95	V
V_{REF}	Reference voltage		0.8	0.9	1.0	V
V_{TT}	Termination voltage		V_{REF} – 0.04	V_{REF}	V_{REF} + 0.04	V
$V_{IH(DC)}$	High-level DC input voltage		V_{REF} + 0.125			V
$V_{IL(DC)}$	Low-level DC input voltage				V_{REF} – 0.125	V
$V_{IH(AC)}$	High-level AC input voltage		V_{REF} + 0.275			V
$V_{IL(AC)}$	Low-level AC input voltage				V_{REF} – 0.275	V
V_{OH}	High-level output voltage	I_{OH} = –13.4 mA *(1)*	V_{TT} + 0.630			V
V_{OL}	Low-level output voltage	I_{OL} = 13.4 mA *(1)*			V_{TT} – 0.630	V

Table 6–23. SSTL-2 Class I Specifications (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		2.375	2.5	2.625	V
V_{TT}	Termination voltage		V_{REF} – 0.04	V_{REF}	V_{REF} + 0.04	V
V_{REF}	Reference voltage		1.15	1.25	1.35	V
V_{IH}	High-level input voltage		V_{REF} + 0.18		3.0	V
V_{IL}	Low-level input voltage		–0.3		V_{REF} – 0.18	V

Table 6–23. SSTL-2 Class I Specifications (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{OH}	High-level output voltage	I_{OH} = –8.1 mA *(1)*	V_{TT} + 0.57			V
V_{OL}	Low-level output voltage	I_{OL} = 8.1 mA *(1)*			V_{TT} – 0.57	V

Table 6–24. SSTL-2 Class II Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		2.3	2.5	2.7	V
V_{TT}	Termination voltage		V_{REF} – 0.04	V_{REF}	V_{REF} + 0.04	V
V_{REF}	Reference voltage		1.15	1.25	1.35	V
V_{IH}	High-level input voltage		V_{REF} + 0.18		V_{CCIO} + 0.3	V
V_{IL}	Low-level input voltage		–0.3		V_{REF} – 0.18	V
V_{OH}	High-level output voltage	I_{OH} = –16.4 mA *(1)*	V_{TT} + 0.76			V
V_{OL}	Low-level output voltage	I_{OL} = 16.4 mA *(1)*			V_{TT} – 0.76	V

Table 6–25. SSTL-3 Class I Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		3.0	3.3	3.6	V
V_{TT}	Termination voltage		V_{REF} – 0.05	V_{REF}	V_{REF} + 0.05	V
V_{REF}	Reference voltage		1.3	1.5	1.7	V
V_{IH}	High-level input voltage		V_{REF} + 0.2		V_{CCIO} + 0.3	V
V_{IL}	Low-level input voltage		–0.3		V_{REF} – 0.2	V
V_{OH}	High-level output voltage	I_{OH} = –8 mA *(1)*	V_{TT} + 0.6			V
V_{OL}	Low-level output voltage	I_{OL} = 8 mA *(1)*			V_{TT} – 0.6	V

Table 6–26. SSTL-3 Class II Specifications (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		3.0	3.3	3.6	V
V_{TT}	Termination voltage		V_{REF} – 0.05	V_{REF}	V_{REF} + 0.05	V
V_{REF}	Reference voltage		1.3	1.5	1.7	V
V_{IH}	High-level input voltage		V_{REF} + 0.2		V_{CCIO} + 0.3	V

Table 6–26. SSTL-3 Class II Specifications (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{IL}	Low-level input voltage		–0.3		V_{REF} – 0.2	V
V_{OH}	High-level output voltage	I_{OH} = –16 mA *(1)*	V_{TT} + 0.8			V
V_{OL}	Low-level output voltage	I_{OL} = 16 mA *(1)*			V_{TT} – 0.8	V

Table 6–27. 3.3-V AGP 2× Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		3.15	3.3	3.45	V
V_{REF}	Reference voltage		0.39 × V_{CCIO}		0.41 × V_{CCIO}	V
V_{IH}	High-level input voltage *(2)*		0.5 × V_{CCIO}		V_{CCIO} + 0.5	V
V_{IL}	Low-level input voltage *(2)*				0.3 × V_{CCIO}	V
V_{OH}	High-level output voltage	I_{OUT} = –0.5 mA	0.9 × V_{CCIO}		3.6	V
V_{OL}	Low-level output voltage	I_{OUT} = 1.5 mA			0.1 × V_{CCIO}	V

Table 6–28. 3.3-V AGP 1× Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		3.15	3.3	3.45	V
V_{IH}	High-level input voltage *(2)*		0.5 × V_{CCIO}		V_{CCIO} + 0.5	V
V_{IL}	Low-level input voltage *(2)*				0.3 × V_{CCIO}	V
V_{OH}	High-level output voltage	I_{OUT} = –0.5 mA	0.9 × V_{CCIO}		3.6	V
V_{OL}	Low-level output voltage	I_{OUT} = 1.5 mA			0.1 × V_{CCIO}	V

Table 6–29. 1.5-V HSTL Class I Specifications (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		1.4	1.5	1.6	V
V_{REF}	Input reference voltage		0.68	0.75	0.9	V
V_{TT}	Termination voltage		0.7	0.75	0.8	V
V_{IH} (DC)	DC high-level input voltage		V_{REF} + 0.1			V
V_{IL} (DC)	DC low-level input voltage		–0.3		V_{REF} – 0.1	V
V_{IH} (AC)	AC high-level input voltage		V_{REF} + 0.2			V
V_{IL} (AC)	AC low-level input voltage				V_{REF} – 0.2	V

Table 6–29. 1.5-V HSTL Class I Specifications (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{OH}	High-level output voltage	I_{OH} = 8 mA *(1)*	V_{CCIO} – 0.4			V
V_{OL}	Low-level output voltage	I_{OH} = –8 mA *(1)*			0.4	V

Table 6–30. 1.5-V HSTL Class II Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		1.4	1.5	1.6	V
V_{REF}	Input reference voltage		0.68	0.75	0.9	V
V_{TT}	Termination voltage		0.7	0.75	0.8	V
V_{IH} (DC)	DC high-level input voltage		V_{REF} + 0.1			V
V_{IL} (DC)	DC low-level input voltage		–0.3		V_{REF} – 0.1	V
V_{IH} (AC)	AC high-level input voltage		V_{REF} + 0.2			V
V_{IL} (AC)	AC low-level input voltage				V_{REF} – 0.2	V
V_{OH}	High-level output voltage	I_{OH} = 16 mA *(1)*	V_{CCIO} – 0.4			V
V_{OL}	Low-level output voltage	I_{OH} = –16 mA *(1)*			0.4	V

Table 6–31. 1.5-V Differential HSTL Specifications

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	I/O supply voltage		1.4	1.5	1.6	V
V_{DIF} (DC)	DC input differential voltage		0.2			V
V_{CM} (DC)	DC common mode input voltage		0.68		0.9	V
V_{DIF} (AC)	AC differential input voltage		0.4			V

Table 6–32. CTT I/O Specifications (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{CCIO}	Output supply voltage		3.0	3.3	3.6	V
V_{TT}/V_{REF}	Termination and input reference voltage		1.35	1.5	1.65	V
V_{IH}	High-level input voltage		V_{REF} + 0.2			V
V_{IL}	Low-level input voltage				V_{REF} – 0.2	V

Table 6–32. CTT I/O Specifications (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Units
V_{OH}	High-level output voltage	I_{OH} = –8 mA	V_{REF} + 0.4			V
V_{OL}	Low-level output voltage	I_{OL} = 8 mA			V_{REF} – 0.4	V
I_O	Output leakage current (when output is high *Z*)	GND ≤ V_{OUT} ≤ V_{CCIO}	–10		10	µA

Table 6–33. Bus Hold Parameters

Parameter	Conditions	V_{CCIO} Level								Units
		1.5 V		1.8 V		2.5 V		3.3 V		
		Min	Max	Min	Max	Min	Max	Min	Max	
Low sustaining current	V_{IN} > V_{IL} (maximum)	25		30		50		70		µA
High sustaining current	V_{IN} < V_{IH} (minimum)	–25		–30		–50		–70		µA
Low overdrive current	0 V < V_{IN} < V_{CCIO}		160		200		300		500	µA
High overdrive current	0 V < V_{IN} < V_{CCIO}		–160		–200		–300		–500	µA
Bus-hold trip point		0.5	1.0	0.68	1.07	0.7	1.7	0.8	2.0	V

Notes to Tables 6–14 through 6–33:

(1) Drive strength is programmable according to values in the *Stratix GX Architecture* chapter of the *Stratix GX Device Handbook, Volume 1*.

(2) V_{REF} specifies the center point of the switching range.

Power Consumption

Detailed power consumption information for Stratix GX devices will be released when available.

Timing Model

The DirectDrive™ technology and MultiTrack™ interconnect ensure predictable performance, accurate simulation, and accurate timing analysis across all Stratix GX device densities and speed grades. This section describes and specifies the performance, internal, external, and PLL timing specifications.

All specifications are representative of worst-case supply voltage and junction temperature conditions.

Preliminary & Final Timing

Timing models can have either preliminary or final status. The Quartus® II software displays an informational message during the design compilation if the timing models are preliminary. Table 6–34 shows the status of the Stratix GX device timing models.

Preliminary status means the timing model is subject to change. Initially, timing numbers are created using simulation results, process data, and other known parameters. These tests are used to make the preliminary numbers as close to the actual timing parameters as possible.

Final timing numbers are based on actual device operation and testing. These numbers reflect the actual performance of the device under worst-case voltage and junction temperature conditions.

Table 6–34. Stratix GX Device Timing Model Status

Device	Preliminary	Final
EP1SGX10	—	✓
EP1SGX25	—	✓
EP1SGX40	—	✓

Performance

Table 6–35 shows Stratix GX device performance for some common designs. All performance values were obtained with Quartus II software compilation of LPM, or MegaCore® functions for the FIR and FFT designs.

Table 6–35. Stratix GX Device Performance (Part 1 of 3) *Notes (1), (2)*

Applications		Resources Used			Performance			
		LEs	TriMatrix Memory Blocks	DSP Blocks	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Units
LE	16-to-1 multiplexer *(1)*	22	0	0	407.83	324.56	288.68	MHz
	32-to-1 multiplexer *(3)*	46	0	0	318.26	255.29	242.89	MHz
	16-bit counter	16	0	0	422.11	422.11	390.01	MHz
	64-bit counter	64	0	0	321.85	290.52	261.23	MHz

Table 6–35. Stratix GX Device Performance (Part 2 of 3) *Notes (1), (2)*

Applications		Resources Used			Performance			
		LEs	TriMatrix Memory Blocks	DSP Blocks	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Units
TriMatrix memory M512 block	Simple dual-port RAM 32 × 18 bit	0	1	0	317.76	277.62	241.48	MHz
	FIFO 32 × 18 bit	30	1	0	319.18	278.86	242.54	MHz
TriMatrix memory M4K block	Simple dual-port RAM 128 × 36 bit	0	1	0	290.86	255.55	222.27	MHz
	True dual-port RAM 128 × 18 bit	0	1	0	290.86	255.55	222.27	MHz
	FIFO 128 × 36 bit	34	1	0	290.86	255.55	222.27	MHz
TriMatrix memory M-RAM block	Single port RAM 4K × 144 bit	1	1	0	255.95	223.06	194.06	MHz
	Simple dual-port RAM 4K × 144 bit	0	1	0	255.95	233.06	194.06	MHz
	True dual-port RAM 4K × 144 bit	0	1	0	255.95	233.06	194.06	MHz
	Single port RAM 8K × 72 bit	0	1	0	278.94	243.19	211.59	MHz
	Simple dual-port RAM 8K × 72 bit	0	1	0	255.95	223.06	194.06	MHz
	True dual-port RAM 8K × 72 bit	0	1	0	255.95	223.06	194.06	MHz
	Single port RAM 16K × 36 bit	0	1	0	280.66	254.32	221.28	MHz
	Simple dual-port RAM 16K × 36 bit	0	1	0	269.83	237.69	206.82	MHz

Table 6–35. Stratix GX Device Performance (Part 3 of 3) *Notes (1), (2)*

Applications		Resources Used			Performance			
		LEs	TriMatrix Memory Blocks	DSP Blocks	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Units
TriMatrix memory M-RAM block	True dual-port RAM 16K × 36 bit	0	1	0	269.83	237.69	206.82	MHz
	Single port RAM 32K × 18 bit	0	1	0	275.86	244.55	212.76	MHz
	Simple dual-port RAM 32K × 18 bit	0	1	0	275.86	244.55	212.76	MHz
	True dual-port RAM 32K × 18 bit	0	1	0	275.86	244.55	212.76	MHz
	Single port RAM 64K × 9 bit	0	1	0	287.85	253.29	220.36	MHz
	Simple dual-port RAM 64K × 9 bit	0	1	0	287.85	253.29	220.36	MHz
	True dual-port RAM 64K × 9 bit	0	1	0	287.85	253.29	220.36	MHz
DSP block	9 × 9-bit multiplier *(3)*	0	0	1	335.0	293.94	255.68	MHz
	18 × 18-bit multiplier *(4)*	0	0	1	278.78	237.41	206.52	MHz
	36 × 36-bit multiplier *(4)*	0	0	1	148.25	134.71	117.16	MHz
	36 × 36-bit multiplier *(5)*	0	0	1	278.78	237.41	206.52	MHz
	18-bit, 4-tap FIR filter	0	0	1	278.78	237.41	206.52	MHz
Larger Designs	8-bit, 16-tap parallel FIR filter	58	0	4	141.26	133.49	114.88	MHz
	8-bit, 1,024-point FFT function	870	5	1	261.09	235.51	205.21	MHz

Notes to Table 6–35:

(1) These design performance numbers were obtained using the Quartus II software.
(2) Numbers not listed will be included in a future version of the data sheet.
(3) This application uses registered inputs and outputs.
(4) This application uses registered multiplier input and output stages within the DSP block.
(5) This application uses registered multiplier input, pipeline, and output stages within the DSP block.

Internal Timing Parameters

Internal timing parameters are specified on a speed grade basis independent of device density. Tables 6–36 through 6–42 describe the Stratix GX device internal timing microparameters for LEs, IOEs, TriMatrix™ memory structures, DSP blocks, and MultiTrack interconnects.

Table 6–36. LE Internal Timing Microparameter Descriptions

Symbol	Parameter
t_{SU}	LE register setup time before clock
t_{H}	LE register hold time after clock
t_{CO}	LE register clock-to-output delay
t_{LUT}	LE combinational LUT delay for data-in to data-out
t_{CLR}	Minimum clear pulse width
t_{PRE}	Minimum preset pulse width
t_{CLKHL}	Minimum clock high or low time

Table 6–37. IOE Internal Timing Microparameter Descriptions

Symbol	Parameter
t_{SU}	IOE input and output register setup time before clock
t_{H}	IOE input and output register hold time after clock
t_{CO}	IOE input and output register clock-to-output delay
$t_{PIN2COMBOUT_R}$	Row input pin to IOE combinational output
$t_{PIN2COMBOUT_C}$	Column input pin to IOE combinational output
$t_{COMBIN2PIN_R}$	Row IOE data input to combinational output pin
$t_{COMBIN2PIN_C}$	Column IOE data input to combinational output pin
t_{CLR}	Minimum clear pulse width
t_{PRE}	Minimum preset pulse width
t_{CLKHL}	Minimum clock high or low time

Table 6–38. DSP Block Internal Timing Microparameter Descriptions	
Symbol	**Parameter**
t_{SU}	Input, pipeline, and output register setup time before clock
t_{H}	Input, pipeline, and output register hold time after clock
t_{CO}	Input, pipeline, and output register clock-to-output delay
$t_{INREG2PIPE9}$	Input register to DSP block pipeline register in 9 × 9-bit mode
$t_{INREG2PIPE18}$	Input register to DSP block pipeline register in 18 × 18-bit mode
$t_{PIPE2OUTREG2ADD}$	DSP block pipeline register to output register delay in two-multipliers adder mode
$t_{PIPE2OUTREG4ADD}$	DSP Block Pipeline Register to output register delay in four-multipliers adder mode
t_{PD9}	Combinational input to output delay for 9 × 9-bit mode
t_{PD18}	Combinational input to output delay for 18 × 18-bit mode
t_{PD36}	Combinational input to output delay for 36 × 36-bit mode
t_{CLR}	Minimum clear pulse width
t_{CLKHL}	Minimum clock high or low time

Table 6–39. M512 Block Internal Timing Microparameter Descriptions	
Symbol	**Parameter**
t_{M512RC}	Synchronous read cycle time
t_{M512WC}	Synchronous write cycle time
$t_{M512WERESU}$	Write or read enable setup time before clock
$t_{M512WEREH}$	Write or read enable hold time after clock
$t_{M512DATASU}$	Data setup time before clock
$t_{M512DATAH}$	Data hold time after clock
$t_{M512WADDRSU}$	Write address setup time before clock
$t_{M512WADDRH}$	Write address hold time after clock
$t_{M512RADDRSU}$	Read address setup time before clock
$t_{M512RADDRH}$	Read address hold time after clock
$t_{M512DATACO1}$	Clock-to-output delay when using output registers
$t_{M512DATACO2}$	Clock-to-output delay without output registers
$t_{M512CLKHL}$	Minimum clock high or low time
$t_{M512CLR}$	Minimum clear pulse width

Table 6–40. M4K Block Internal Timing Microparameter Descriptions	
Symbol	**Parameter**
t_{M4KRC}	Synchronous read cycle time
t_{M4KWC}	Synchronous write cycle time
$t_{M4KWERESU}$	Write or read enable setup time before clock
$t_{M4KWEREH}$	Write or read enable hold time after clock
$t_{M4KBESU}$	Byte enable setup time before clock
t_{M4KBEH}	Byte enable hold time after clock
$t_{M4KDATAASU}$	A port data setup time before clock
$t_{M4KDATAAH}$	A port data hold time after clock
$t_{M4KADDRASU}$	A port address setup time before clock
$t_{M4KADDRAH}$	A port address hold time after clock
$t_{M4KDATABSU}$	B port data setup time before clock
$t_{M4KDATABH}$	B port data hold time after clock
$t_{M4KADDRBSU}$	B port address setup time before clock
$t_{M4KADDRBH}$	B port address hold time after clock
$t_{M4KDATACO1}$	Clock-to-output delay when using output registers
$t_{M4KDATACO2}$	Clock-to-output delay without output registers
$t_{M4KCLKHL}$	Minimum clock high or low time
t_{M4KCLR}	Minimum clear pulse width

Table 6–41. M-RAM Block Internal Timing Microparameter Descriptions (Part 1 of 2)	
Symbol	**Parameter**
t_{MRAMRC}	Synchronous read cycle time
t_{MRAMWC}	Synchronous write cycle time
$t_{MRAMWERESU}$	Write or read enable setup time before clock
$t_{MRAMWEREH}$	Write or read enable hold time after clock
$t_{MRAMBESU}$	Byte enable setup time before clock
$t_{MRAMBEH}$	Byte enable hold time after clock
$t_{MRAMDATAASU}$	A port data setup time before clock
$t_{MRAMDATAAH}$	A port data hold time after clock
$t_{MRAMADDRASU}$	A port address setup time before clock
$t_{MRAMADDRAH}$	A port address hold time after clock

Table 6–41. M-RAM Block Internal Timing Microparameter Descriptions (Part 2 of 2)

Symbol	Parameter
$t_{MRAMDATABSU}$	B port setup time before clock
$t_{MRAMDATABH}$	B port hold time after clock
$t_{MRAMADDRBSU}$	B port address setup time before clock
$t_{MRAMADDRBH}$	B port address hold time after clock
$t_{MRAMDATACO1}$	Clock-to-output delay when using output registers
$t_{MRAMDATACO2}$	Clock-to-output delay without output registers
$t_{MRAMCLKHL}$	Minimum clock high or low time
$t_{MRAMCLR}$	Minimum clear pulse width

Table 6–42. Routing Delay Internal Timing Microparameter Descriptions

Symbol	Parameter
t_{R4}	Delay for an R4 line with average loading; covers a distance of four LAB columns
t_{R8}	Delay for an R8 line with average loading; covers a distance of eight LAB columns
t_{R24}	Delay for an R24 line with average loading; covers a distance of 24 LAB columns
t_{C4}	Delay for an C4 line with average loading; covers a distance of four LAB rows
t_{C8}	Delay for an C8 line with average loading; covers a distance of eight LAB rows
t_{C16}	Delay for an C16 line with average loading; covers a distance of 16 LAB rows
t_{LOCAL}	Local interconnect delay

Table 6–43. Stratix GX Reset & PLL Lock Time Parameter Descriptions (Part 1 of 2)

Symbol	Parameter
$t_{ANALOGRESETPW}$	Pulse width to power down analog circuits.
$t_{DIGITALRESETPW}$	Pulse width to reset digital circuits
$t_{TX_PLL_LOCK}$	The time it takes the `tx_pll` to lock to the reference clock.

Table 6–43. Stratix GX Reset & PLL Lock Time Parameter Descriptions (Part 2 of 2)	
Symbol	**Parameter**
$t_{RX_FREQLOCK}$	The time until the clock recovery unit (CRU) switches to data mode from lock to reference mode.
$t_{RX_FREQLOCK2PHASELOCK}$	The time until CRU phase locks to data after switching from lock to data mode.

Figure 6–4 shows the TriMatrix memory waveforms for the M512, M4K, and M-RAM timing parameters shown in Tables 6–39 through 6–41.

Figure 6–4. Dual-Port RAM Timing Microparameter Waveform

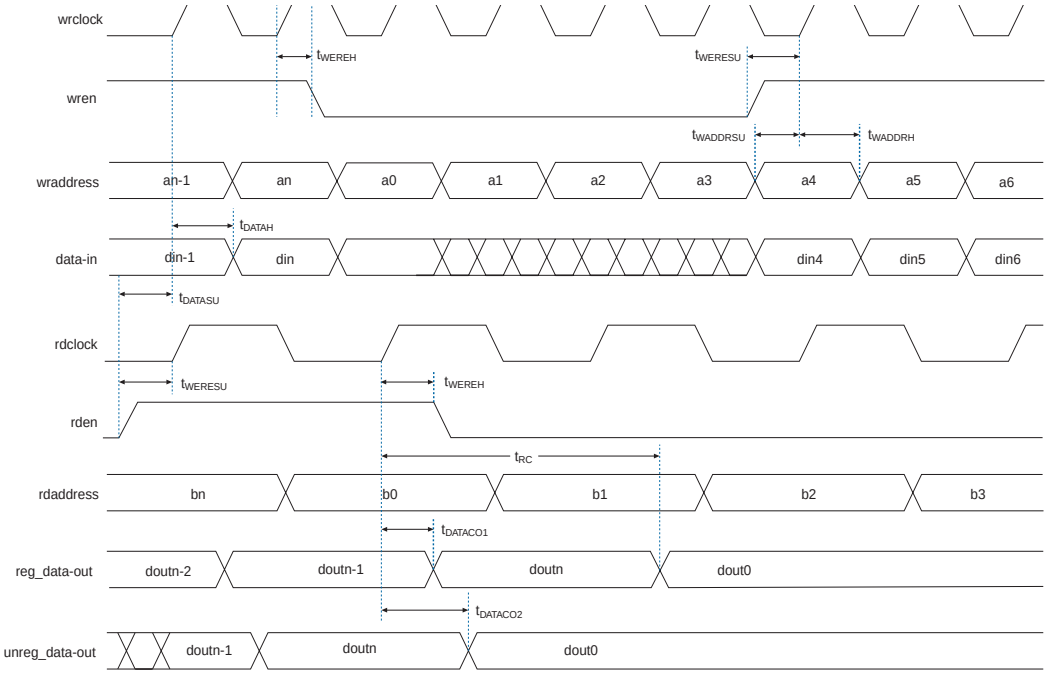
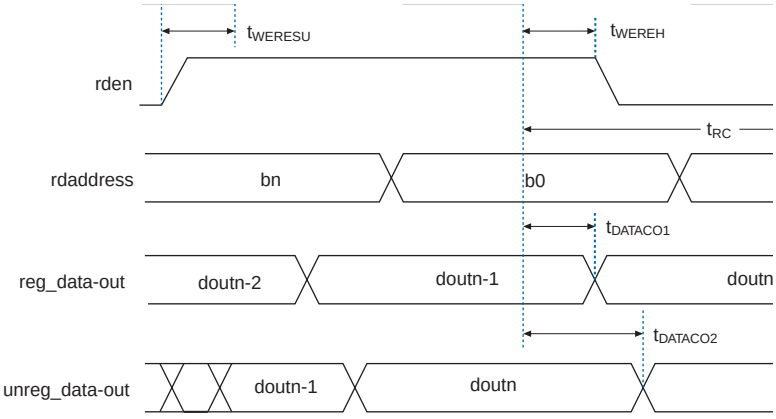


Figure 6–5. Stratix GX Transceiver Reset & PLL Lock Time Waveform *Note (1)*



Note to Figure 6–5:

(1) Waveforms are for minimum pulse width timing and output timing only. Please refer to the *Stratix GX Transceiver User Guide* for the complete reset sequence.

Tables 6–44 through 6–50 show the internal timing microparameters for all Stratix GX devices.

Table 6–44. LE Internal Timing Microparameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{SU}	10		10		11		ps
t_{H}	100		100		114		ps
t_{CO}		156		176		202	ps
t_{LUT}		366		459		527	ps
t_{CLR}	100		100		114		ps
t_{PRE}	100		100		114		ps
t_{CLKHL}	100		100		114		ps

Table 6–45. IOE Internal Timing Microparameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{SU}	64		68		68		ps
t_{H}	76		80		80		ps
t_{CO}		162		171		171	ps
$t_{PIN2COMBOUT_R}$		1,038		1,093		1,256	ps
$t_{PIN2COMBOUT_C}$		927		976		1,122	ps
$t_{COMBIN2PIN_R}$		2,944		3,099		3,563	ps
$t_{COMBIN2PIN_C}$		3,189		3,357		3,860	ps
t_{CLR}	262		276		317		ps
t_{PRE}	262		276		317		ps
t_{CLKHL}	90		95		109		ps

Table 6–46. DSP Block Internal Timing Microparameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{SU}	0		0		0		ps
t_{H}	67		75		86		ps
t_{CO}		142		158		181	ps
$t_{INREG2PIPE18}$		2,613		2,982		3,429	ps
$t_{INREG2PIPE9}$		3,390		3,993		4,591	ps
$t_{PIPE2OUTREG2ADD}$		2,002		2,203		2,533	ps
$t_{PIPE2OUTREG4ADD}$		2,899		3,189		3,667	ps
t_{PD9}		3,709		4,081		4,692	ps
t_{PD18}		4,795		5,275		6,065	ps
t_{PD36}		7,495		8,245		9,481	ps
t_{CLR}	450		500		575		ps
t_{CLKHL}	1,350		1,500		1,724		ps

Table 6–47. M512 Block Internal Timing Microparameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{M512RC}		3,340		3,816		4,387	ps
t_{M512WC}		3,318		3,590		4,128	ps
$t_{M512WERESU}$	110		123		141		ps
$t_{M512WERH}$	34		38		43		ps
$t_{M512DATASU}$	110		123		141		ps
$t_{M512DATAH}$	34		38		43		ps
$t_{M512WADDRASU}$	110		123		141		ps
$t_{M512WADDRH}$	34		38		43		ps
$t_{M512DATACO1}$		424		472		541	ps
$t_{M512DATACO2}$		3,366		3,846		4,421	ps
$t_{M512CLKHL}$	150		167		192		ps
$t_{M512CLR}$	170		189		217		ps

Table 6–48. M4K Block Internal Timing Microparameters (Part 1 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{M4KRC}		3,807		4,320		4,967	ps
t_{M4KWC}		2,556		2,840		3,265	ps
$t_{M4KWERESU}$	131		149		171		ps
$t_{M4KWERH}$	34		38		43		ps
$t_{M4KDATASU}$	131		149		171		ps
$t_{M4KDATAH}$	34		38		43		ps
$t_{M4KWADDRASU}$	131		149		171		ps
$t_{M4KWADDRH}$	34		38		43		ps
$t_{M4KRADDRASU}$	131		149		171		ps
$t_{M4KRADDRH}$	34		38		43		ps
$t_{M4KDATABSU}$	131		149		171		ps
$t_{M4KDATABH}$	34		38		43		ps
$t_{M4KADDRBSU}$	131		149		171		ps
$t_{M4KADDRBH}$	34		38		43		ps

Table 6–48. M4K Block Internal Timing Microparameters (Part 2 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
$t_{M4KDATACO1}$		571		635		729	ps
$t_{M4KDATACO2}$		3,984		4,507		5,182	ps
$t_{M4KCLKHL}$	150		167		192		ps
t_{M4KCLR}	170		189		255		ps

Table 6–49. M-RAM Block Internal Timing Microparameters

Symbol	-5		-6		-7		Unit
	Min	Max	Min	Max	Min	Max	
t_{MRAMRC}		4,364		4,838		5,562	ps
t_{MRAMWC}		3,654		4,127		4,746	ps
$t_{MRAMWERESU}$	25		25		28		ps
$t_{MRAMWERH}$	18		20		23		ps
$t_{MRAMDATASU}$	25		25		28		ps
$t_{MRAMDATAH}$	18		20		23		ps
$t_{MRAMWADDRASU}$	25		25		28		ps
$t_{MRAMWADDRH}$	18		20		23		ps
$t_{MRAMRADDRASU}$	25		25		28		ps
$t_{MRAMRADDRH}$	18		20		23		ps
$t_{MRAMDATABSU}$	25		25		28		ps
$t_{MRAMDATABH}$	18		20		23		ps
$t_{MRAMADDRBSU}$	25		25		28		ps
$t_{MRAMADDRBH}$	18		20		23		ps
$t_{MRAMDATACO1}$		1,038		1,053		1,210	ps
$t_{MRAMDATACO2}$		4,362		4,939		5,678	ps
$t_{MRAMCLKHL}$	270		300		345		ps
$t_{MRAMCLR}$	135		150		172		ps

Table 6–50. Stratix GX Transceiver Reset & PLL Lock Time Parameters

Symbol	Min	Typ	Max	Units
$t_{ANALOGRESETPW}$ (5)	1			mS
$t_{DIGITALRESETPW}$ (5)	4			Parallel clock cycle
$t_{TX_PLL_LOCK}$ (3)			10	µS
$t_{RX_FREQLOCK}$ (4)			5	mS
$t_{RX_FREQLOCK2PHASELOCK}$ (2)			5	µS

Notes to Table 6–50:

(1) The minimum pulse width specified is associated with the power-down of circuits.

(2) The clock recovery unit (CRU) phase locked-to-data time is based on a data rate of 500 Mbps and 8B/10B encoded data.

(3) After `#pll_areset`, `pll_enable`, or PLL power-up, the time required for the transceiver PLL to lock to the reference clock.

(4) After `#rx_analogreset`, the time for the CRU to switch to lock-to-data mode.

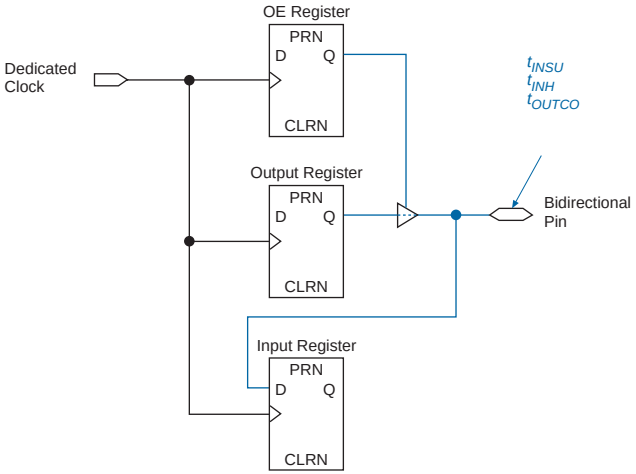
(5) There is no maximum pulse width specification. The GXB can be held in reset indefinitely.

Routing delays vary depending on the load on a specific routing line. The Quartus II software reports the routing delay information when running the timing analysis for a design. Contact Altera Applications Engineering for more details.

External Timing Parameters

External timing parameters are specified by device density and speed grade. Figure 6–6 shows the timing model for bidirectional IOE pin timing. All registers are within the IOE.

Figure 6–6. External Timing in Stratix GX Devices



All external I/O timing parameters shown are for 3.3-V LVTTL or LVCMOS I/O standards with the maximum current strength. For external I/O timing using standards other than LVTTL or LVCMOS use the I/O standard input and output delay adders in Tables 6–72 through 6–76.

Table 6–51 shows the external I/O timing parameters when using fast regional clock networks.

Table 6–51. Stratix GX Fast Regional Clock External I/O Timing Parameters *Notes (1), (2)*

Symbol	Parameter	Conditions
t_{INSU}	Setup time for input or bidirectional pin using column IOE input register with fast regional clock fed by FCLK pin	
t_{INH}	Hold time for input or bidirectional pin using column IOE input register with fast regional clock fed by FCLK pin	
t_{OUTCO}	Clock-to-output delay output or bidirectional pin using column IOE output register with fast regional clock fed by FCLK pin	C_{LOAD} = 10 pF

Notes to Table 6–51:
(1) These timing parameters are sample-tested only.
(2) These timing parameters are for column IOE pins. Row IOE pins are 100- to 250-ps slower depending on device and speed grade and whether it is t_{CO} or t_{SU}. You should use the Quartus II software to verify the external timing for any pin.

Table 6–52 shows the external I/O timing parameters when using regional clock networks.

Table 6–52. Stratix GX Regional Clock External I/O Timing Parameters *Notes (1), (2)*		
Symbol	**Parameter**	**Conditions**
t_{INSU}	Setup time for input or bidirectional pin using column IOE input register with regional clock fed by CLK pin	
t_{INH}	Hold time for input or bidirectional pin using column IOE input register with regional clock fed by CLK pin	
t_{OUTCO}	Clock-to-output delay output or bidirectional pin using column IOE output register with regional clock fed by CLK pin	C_{LOAD} = 10 pF
$t_{INSUPLL}$	Setup time for input or bidirectional pin using column IOE input register with regional clock fed by Enhanced PLL with default phase setting	
t_{INHPLL}	Hold time for input or bidirectional pin using column IOE input register with regional clock fed by Enhanced PLL with default phase setting	
$t_{OUTCOPLL}$	Clock-to-output delay output or bidirectional pin using column IOE output register with regional clock Enhanced PLL with default phase setting	C_{LOAD} = 10 pF

Notes to Table 6–52:
(1) These timing parameters are sample-tested only.
(2) These timing parameters are for column IOE pins. Row IOE pins are 100- to 250-ps slower depending on device, speed grade, and the specific parameter in question. You should use the Quartus II software to verify the external timing for any pin.

Table 6–53 shows the external I/O timing parameters when using global clock networks.

Table 6–53. Stratix GX Global Clock External I/O Timing Parameters (Part 1 of 2) *Notes (1), (2)*		
Symbol	**Parameter**	**Conditions**
t_{INSU}	Setup time for input or bidirectional pin using column IOE input register with global clock fed by CLK pin	
t_{INH}	Hold time for input or bidirectional pin using column IOE input register with global clock fed by CLK pin	
t_{OUTCO}	Clock-to-output delay output or bidirectional pin using column IOE output register with global clock fed by CLK pin	C_{LOAD} = 10 pF
$t_{INSUPLL}$	Setup time for input or bidirectional pin using column IOE input register with global clock fed by Enhanced PLL with default phase setting	

Table 6–53. Stratix GX Global Clock External I/O Timing Parameters (Part 2 of 2) Notes (1), (2)		
Symbol	**Parameter**	**Conditions**
t_{INHPLL}	Hold time for input or bidirectional pin using column IOE input register with global clock fed by enhanced PLL with default phase setting	
$t_{OUTCOPLL}$	Clock-to-output delay output or bidirectional pin using column IOE output register with global clock enhanced PLL with default phase setting	C_{LOAD} = 10 pF

Notes to Table 6–53:
(1) These timing parameters are sample-tested only.
(2) These timing parameters are for column IOE pins. Row IOE pins are 100- to 250-ps slower depending on device, speed grade, and the specific parameter in question. You should use the Quartus II software to verify the external timing for any pin.

Tables 6–54 through 6–59 show the external timing parameters on column and row pins for EP1SGX10 devices.

Table 6–54. EP1SGX10 Column Pin Fast Regional Clock External I/O Timing Parameters							
Symbol	**-5 Speed Grade**		**-6 Speed Grade**		**-7 Speed Grade**		**Unit**
	Min	**Max**	**Min**	**Max**	**Min**	**Max**	
t_{INSU}	2.245		2.332		2.666		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.597	2.000	4.920	2.000	5.635	ns

Table 6–55. EP1SGX10 Column Pin Regional Clock External I/O Timing Parameters							
Symbol	**-5 Speed Grade**		**-6 Speed Grade**		**-7 Speed Grade**		**Unit**
	Min	**Max**	**Min**	**Max**	**Min**	**Max**	
t_{INSU}	2.114		2.218		2.348		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.728	2.000	5.078	2.000	6.004	ns
$t_{INSUPLL}$	1.035		0.941		1.070		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.629	0.500	2.769	0.500	3.158	ns

Table 6–56. EP1SGX10 Column Pin Global Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	1.785		1.814		2.087		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	5.057	2.000	5.438	2.000	6.214	ns
$t_{INSUPLL}$	0.988		0.936		1.066		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.634	0.500	2.774	0.500	3.162	ns

Table 6–57. EP1SGX10 Row Pin Fast Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.194		2.384		2.727		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.956	2.000	4.971	2.000	5.463	ns

Table 6–58. EP1SGX10 Row Pin Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.244		2.413		2.574		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.906	2.000	4.942	2.000	5.616	ns
$t_{INSUPLL}$	1.126		1.186		1.352		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.804	0.500	2.627	0.500	2.765	ns

Table 6–59. EP1SGX10 Row Pin Global Clock External I/O Timing Parameters (Part 1 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	1.919		2.062		2.368		ns
t_{INH}	0.000		0.000		0.000		ns

Table 6–59. EP1SGX10 Row Pin Global Clock External I/O Timing Parameters (Part 2 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{OUTCO}	2.000	5.231	2.000	5.293	2.000	5.822	ns
$t_{INSUPLL}$	1.126		1.186		1.352		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.804	0.500	2.627	0.500	2.765	ns

Tables 6–60 through 6–65 show the external timing parameters on column and row pins for EP1SGX25 devices.

Table 6–60. EP1SGX25 Column Pin Fast Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.418		2.618		3.014		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.524	2.000	4.834	2.000	5.538	ns

Table 6–61. EP1SGX25 Column Pin Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	1.713		1.838		2.069		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	5.229	2.000	5.614	2.000	6.432	ns
$t_{INSUPLL}$	1.061		1.155		1.284		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.661	0.500	2.799	0.500	3.195	ns

Table 6–62. EP1SGX25 Column Pin Global Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	1.790		1.883		2.120		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	5.194	2.000	5.569	2.000	6.381	ns
$t_{INSUPLL}$	1.046		1.141		1.220		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.676	0.500	2.813	0.500	3.208	ns

Table 6–63. EP1SGX25 Row Pin Fast Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.394		2.594		2.936		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.456	2.000	4.761	2.000	5.454	ns

Table 6–64. EP1SGX25 Row Pin Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	1.970		2.109		2.377		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.880	2.000	5.246	2.000	6.013	ns
$t_{INSUPLL}$	1.326		1.386		1.552		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.304	0.500	2.427	0.500	2.765	ns

Table 6–65. EP1SGX25 Row Pin Global Clock External I/O Timing Parameters (Part 1 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	1.963		2.108		2.379		ns
t_{INH}	0.000		0.000		0.000		ns

Table 6–65. EP1SGX25 Row Pin Global Clock External I/O Timing Parameters (Part 2 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{OUTCO}	2.000	4.887	2.000	5.247	2.000	6.011	ns
$t_{INSUPLL}$	1.326		1.386		1.552		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.304	0.500	2.427	0.500	2.765	ns

Tables 6–66 through 6–71 show the external timing parameters on column and row pins for EP1SGX40 devices.

Table 6–66. EP1SGX40 Column Pin Fast Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.704		2.912		3.235		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	5.060	2.000	5.432	2.000	6.226	ns

Table 6–67. EP1SGX40 Column Pin Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.467		2.671		3.011		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	5.255	2.000	5.673	2.000	6.501	ns
$t_{INSUPLL}$	1.254		1.259		1.445		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.610	0.500	2.751	0.500	3.134	ns

Table 6–68. EP1SGX40 Column Pin Global Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.033		2.184		2.451		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	5.689	2.000	6.116	2.000	7.010	ns
$t_{INSUPLL}$	1.228		1.278		1.415		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.594	0.500	2.732	0.500	3.113	ns

Table 6–69. EP1SGX40 Row Pin Fast Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.450		2.662		3.046		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.880	2.000	5.241	2.000	6.004	ns

Table 6–70. EP1SGX40 Row Pin Regional Clock External I/O Timing Parameters

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	2.398		2.567		2.938		ns
t_{INH}	0.000		0.000		0.000		ns
t_{OUTCO}	2.000	4.932	2.000	5.336	2.000	6.112	ns
$t_{INSUPLL}$	1.126		1.186		1.352		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.304	0.500	2.427	0.500	2.765	ns

Table 6–71. EP1SGX40 Row Pin Global Clock External I/O Timing Parameters (Part 1 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{INSU}	1.965		2.128		2.429		ns
t_{INH}	0.000		0.000		0.000		ns

Table 6–71. EP1SGX40 Row Pin Global Clock External I/O Timing Parameters (Part 2 of 2)

Symbol	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{OUTCO}	2.000	5.365	2.000	5.775	2.000	6.621	ns
$t_{INSUPLL}$	1.126		1.186		1.352		ns
t_{INHPLL}	0.000		0.000		0.000		ns
$t_{OUTCOPLL}$	0.500	2.304	0.500	2.427	0.500	2.765	ns

External I/O Delay Parameters

External I/O delay timing parameters, both for I/O standard input and output adders and programmable input and output delays, are specified by speed grade, independent of device density.

Tables 6–72 through 6–77 show the adder delays associated with column and row I/O pins. If an I/O standard is selected other than LVTTL 24 mA with a fast slew rate, add the selected delay to the external t_{CO} and t_{SU} I/O parameters.

Table 6–72. Stratix GX I/O Standard Column Pin Input Delay Adders (Part 1 of 2)

I/O Standard	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
LVCMOS		0		0		0	ps
3.3-V LVTTL		0		0		0	ps
2.5-V LVTTL		30		31		35	ps
1.8-V LVTTL		150		157		180	ps
1.5-V LVTTL		210		220		252	ps
GTL		220		231		265	ps
GTL+		220		231		265	ps
3.3-V PCI		0		0		0	ps
3.3-V PCI-X 1.0		0		0		0	ps
Compact PCI		0		0		0	ps
AGP 1×		0		0		0	ps
AGP 2×		0		0		0	ps
CTT		120		126		144	ps
SSTL-3 class I		–30		–32		–37	ps
SSTL-3 class II		–30		–32		–37	ps

Table 6–72. Stratix GX I/O Standard Column Pin Input Delay Adders (Part 2 of 2)

I/O Standard	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
SSTL-2 class I		–70		–74		–86	ps
SSTL-2 class II		–70		–74		–86	ps
SSTL-18 class I		180		189		217	ps
SSTL-18 class II		180		189		217	ps
1.5-V HSTL class I		120		126		144	ps
1.5-V HSTL class II		120		126		144	ps
1.8-V HSTL class I		70		73		83	ps
1.8-V HSTL class II		70		73		83	ps

Table 6–73. Stratix GX I/O Standard Row Pin Input Delay Adders (Part 1 of 2)

I/O Standard	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
LVCMOS		0		0		0	ps
3.3-V LVTTL		0		0		0	ps
2.5-V LVTTL		30		31		35	ps
1.8-V LVTTL		150		157		180	ps
1.5-V LVTTL		210		220		252	ps
GTL		0		0		0	ps
GTL+		220		231		265	ps
3.3-V PCI		0		0		0	ps
3.3-V PCI-X 1.0		0		0		0	ps
Compact PCI		0		0		0	ps
AGP 1×		0		0		0	ps
AGP 2×		0		0		0	ps
CTT		80		84		96	ps
SSTL-3 class I		–30		–32		–37	ps
SSTL-3 class II		–30		–32		–37	ps
SSTL-2 class I		–70		–74		–86	ps
SSTL-2 class II		–70		–74		–86	ps
SSTL-18 class I		180		189		217	ps
SSTL-18 class II		0		0		0	ps
1.5-V HSTL class I		130		136		156	ps

Table 6–73. Stratix GX I/O Standard Row Pin Input Delay Adders (Part 2 of 2)

I/O Standard	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
1.5-V HSTL class II		0		0		0	ps
1.8-V HSTL class I		70		73		83	ps
1.8-V HSTL class II		70		73		83	ps
LVDS *(1)*		40		42		48	ps
LVPECL *(1)*		–50		–53		–61	ps
3.3-V PCML *(1)*		330		346		397	ps
HyperTransport *(1)*		80		84		96	ps

Table 6–74. Stratix GX I/O Standard Output Delay Adders for Fast Slew Rate on Column Pins (Part 1 of 2)

Standard		-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
LVCMOS	2 mA		570		599		689	ps
	4 mA		570		599		689	ps
	8 mA		350		368		423	ps
	12 mA		130		137		157	ps
	24 mA		0		0		0	ps
3.3-V LVTTL	4 mA		570		599		689	ps
	8 mA		350		368		423	ps
	12 mA		130		137		157	ps
	16 mA		70		74		85	ps
	24 mA		0		0		0	ps
2.5-V LVTTL	2 mA		830		872		1,002	ps
	8 mA		250		263		302	ps
	12 mA		140		147		169	ps
	16 mA		100		105		120	ps
1.8-V LVTTL	2 mA		420		441		507	ps
	8 mA		350		368		423	ps
	12 mA		350		368		423	ps
1.5-V LVTTL	2 mA		1,740		1,827		2,101	ps
	4 mA		1,160		1,218		1,400	ps
	8 mA		690		725		833	ps
GTL			–150		–157		–181	ps

Table 6–74. Stratix GX I/O Standard Output Delay Adders for Fast Slew Rate on Column Pins (Part 2 of 2)

Standard	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
GTL+		–110		–115		–133	ps
3.3-V PCI		–230		–241		–277	ps
3.3-V PCI-X 1.0		–230		–241		–277	ps
Compact PCI		–230		–241		–277	ps
AGP 1×		–30		–31		–36	ps
AGP 2×		–30		–31		–36	ps
CTT		50		53		61	ps
SSTL-3 class I		90		95		109	ps
SSTL-3 class II		–50		–52		–60	ps
SSTL-2 class I		100		105		120	ps
SSTL-2 class II		20		21		24	ps
SSTL-18 class I		230		242		278	ps
SSTL-18 class II		0		0		0	ps
1.5-V HSTL class I		380		399		459	ps
1.5-V HSTL class II		190		200		230	ps
1.8-V HSTL class I		380		399		459	ps
1.8-V HSTL class II		390		410		471	ps

Table 6–75. Stratix GX I/O Standard Output Delay Adders for Fast Slew Rate on Row Pins (Part 1 of 2)

Standard		-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
LVCMOS	2 mA		570		599		689	ps
	4 mA		570		599		689	ps
	8 mA		350		368		423	ps
	12 mA		130		137		157	ps
	24 mA		0		0		0	ps
3.3-V LVTTL	4 mA		570		599		689	ps
	8 mA		350		368		423	ps
	12 mA		130		137		157	ps
	16 mA		70		74		85	ps
	24 mA		0		0		0	ps

Table 6–75. Stratix GX I/O Standard Output Delay Adders for Fast Slew Rate on Row Pins (Part 2 of 2)

Standard		-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
2.5-V LVTTL	2 mA		830		872		1,002	ps
	8 mA		250		263		302	ps
	12 mA		140		147		169	ps
	16 mA		100		105		120	ps
1.8-V LVTTL	2 mA		1,510		1,586		1,824	ps
	8 mA		420		441		507	ps
	12 mA		350		368		423	ps
1.5-V LVTTL	2 mA		1,740		1,827		2,101	ps
	4 mA		1,160		1,218		1,400	ps
	8 mA		690		725		833	ps
CTT			50		53		61	ps
SSTL-3 class I			90		95		109	ps
SSTL-3 class II			–50		–52		–60	ps
SSTL-2 class I			100		105		120	ps
SSTL-2 class II			20		21		24	ps
LVDS *(1)*			–20		–21		–24	ps
LVPECL *(1)*			40		42		48	ps
PCML *(1)*			–60		–63		–73	ps
HyperTransport Technology *(1)*			70		74		85	ps

Table 6–76. Stratix GX I/O Standard Output Delay Adders for Slow Slew Rate on Column Pins (Part 1 of 2)

I/O Standard		-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
LVCMOS	2 mA		1,911		2,011		2,312	ps
	4 mA		1,911		2,011		2,312	ps
	8 mA		1,691		1,780		2,046	ps
	12 mA		1,471		1,549		1,780	ps
	24 mA		1,341		1,412		1,623	ps

Table 6–76. Stratix GX I/O Standard Output Delay Adders for Slow Slew Rate on Column Pins (Part 2 of 2)

I/O Standard		-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
3.3-V LVTTL	4 mA		1,993		2,097		2,411	ps
	8 mA		1,773		1,866		2,145	ps
	12 mA		1,553		1,635		1,879	ps
	16 mA		1,493		1,572		1,807	ps
	24 mA		1,423		1,498		1,722	ps
2.5-V LVTTL	2 mA		2,631		2,768		3,182	ps
	8 mA		2,051		2,159		2,482	ps
	12 mA		1,941		2,043		2,349	ps
	16 mA		1,901		2,001		2,300	ps
1.8-V LVTTL	2 mA		4,632		4,873		5,604	ps
	8 mA		3,542		3,728		4,287	ps
	12 mA		3,472		3,655		4,203	ps
1.5-V LVTTL	2 mA		6,620		6,964		8,008	ps
	4 mA		6,040		6,355		7,307	ps
	8 mA		5,570		5,862		6,740	ps
GTL			1,191		1,255		1,442	ps
GTL+			1,231		1,297		1,90	ps
3.3-V PCI			1,111		1,171		1,346	ps
3.3-V PCI-X 1.0			1,111		1,171		1,346	ps
Compact PCI			1,111		1,171		1,346	ps
AGP 1×			1,311		1,381		1,587	ps
AGP 2×			1,311		1,381		1,587	ps
CTT			1,391		1,465		1,684	ps
SSTL-3 class I			1,431		1,507		1,732	ps
SSTL-3 class II			1,291		1,360		1,563	ps
SSTL-2 class I			1,912		2,013		2,314	ps
SSTL-2 class II			1,832		1,929		2,218	ps
SSTL-18 class I			3,097		3,260		3,748	ps
SSTL-18 class II			2,867		3,018		3,470	ps
1.5-V HSTL class I			4,916		5,174		5,950	ps
1.5-V HSTL class II			4,726		4,975		5,721	ps
1.8-V HSTL class I			3,247		3,417		3,929	ps
1.8-V HSTL class II			3,257		3,428		3,941	ps

Table 6–77. Stratix GX I/O Standard Output Delay Adders for Slow Slew Rate on Row Pins

I/O Standard		-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
LVCMOS	2 mA		1,930		2,031		2,335	ps
	4 mA		1,930		2,031		2,335	ps
	8 mA		1,710		1,800		2,069	ps
	12 mA		1,490		1,569		1,803	ps
3.3-V LVTTL	4 mA		1,953		2,055		2,363	ps
	8 mA		1,733		1,824		2,097	ps
	12 mA		1,513		1,593		1,831	ps
	16 mA		1,453		1,530		1,759	ps
2.5-V LVTTL	2 mA		2,632		2,769		3,183	ps
	8 mA		2,052		2,160		2,483	ps
	12 mA		1,942		2,044		2,350	ps
	16 mA		1,902		2,002		2,301	ps
1.8-V LVTTL	2 mA		4,537		4,773		5,489	ps
	8 mA		3,447		3,628		4,172	ps
	12 mA		3,377		3,555		4,088	ps
1.5-V LVTTL	2 mA		6,575		6,917		7,954	ps
	4 mA		5,995		6,308		7,253	ps
	8 mA		5,525		5,815		6,686	ps
CTT			1,410		1,485		1,707	ps
SSTL-3 class I			1,450		1,527		1,755	ps
SSTL-3 class II			1,310		1,380		1,586	ps
SSTL-2 class I			1,797		1,892		2,175	ps
SSTL-2 class II			1,717		1,808		2,079	ps
LVDS *(1)*			1,340		1,411		1,622	ps
LVPECL *(1)*			1,400		1,474		1,694	ps
3.3-V PCML *(1)*			1,300		1,369		1,573	ps
HyperTransport technology *(1)*			1,430		1,506		1,731	ps

Note to Tables 6–72 through 6–77:
(1) These parameters are only available on the left side row I/O pins.

Tables 6–78 and 6–79 show the adder delays for the column and row IOE programmable delays, respectively. These delays are controlled with the Quartus II software logic options listed in the Parameter column.

Table 6–78. Stratix GX IOE Programmable Delays on Column Pins

Parameter	Setting	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
Decrease input delay to internal cells	Off		3,970		4,367		5,022	ps
	On		3,390		3,729		4,288	ps
	Small		2,810		3,091		3,554	ps
	Medium		212		224		257	ps
	Large		212		224		257	ps
Decrease input delay to input register	Off		3900		4,290		4,933	ps
	On		0		0		0	ps
Decrease input delay to output register	Off		1,240		1,364		1,568	ps
	On		0		0		0	ps
Increase delay to output pin	Off		0		0		0	ps
	On		377		397		456	ps
Increase delay to output enable pin	Off		0		0		0	ps
	On		338		372		427	ps
Increase output clock enable delay	Off		0		0		0	ps
	On		540		594		683	ps
	Small		1,016		1,118		1,285	ps
	Large		1,016		1,118		1,285	ps
Increase input clock enable delay	Off		0		0		0	ps
	On		540		594		683	ps
	Small		1,016		1,118		1,285	ps
	Large		1,016		1,118		1,285	ps
Increase output enable clock enable delay	Off		0		0		0	ps
	On		540		594		683	ps
	Small		1,016		1,118		1,285	ps
	Large		1,016		1,118		1,285	ps

Table 6–79. Stratix GX IOE Programmable Delays on Row Pins

Parameter	Setting	-5 Speed Grade		-6 Speed Grade		-7 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
Decrease input delay to internal cells	Off		3,970		4,367		5,022	ps
	On		3,390		3,729		4,288	ps
	Small		2,810		3,091		3,554	ps
	Medium		164		173		198	ps
	Large		164		173		198	ps
Decrease input delay to input register	Off		3,900		4,290		4,933	ps
	On		0		0		0	ps
Decrease input delay to output register	Off		1,240		1,364		1,568	ps
	On		0		0		0	ps
Increase delay to output pin	Off		0		0		0	ps
	On		377		397		456	ps
Increase delay to output enable pin	Off		0		0		0	ps
	On		348		383		441	ps
Increase output clock enable delay	Off		0		0		0	ps
	On		180		198		227	ps
	Small		260		286		328	ps
	Large		260		286		328	ps
Increase input clock enable delay	Off		0		0		0	ps
	On		180		198		227	ps
	Small		260		286		328	ps
	Large		260		286		328	ps
Increase output enable clock enable delay	Off		0		0		0	ps
	On		540		594		683	ps
	Small		1,016		1,118		1,285	ps
	Large		1,016		1,118		1,285	ps

The scaling factors for output pin timing in Table 6–80 are shown in units of time per pF unit of capacitance (ps/pF). Add this delay to the combinational timing path for output or bidirectional pins in addition to the "I/O Adder" delays shown in Tables 6–72 through 6–77 and the "IOE Programmable Delays" in Tables 6–78 and 6–79.

Table 6–80. Output Delay Adder for Loading on LVTTL/LVCMOS Output Buffers

LVTTL/LVCMOS Standards						
Conditions		**Output Pin Adder Delay (ps/pF)**				
Parameter	**Value**	3.3-V LVTTL	2.5-V LVTTL	1.8-V LVTTL	1.5-V LVTTL	LVCMOS
Drive Strength	24 mA	15	–	–	-	8
	16 mA	25	18	–	–	–
	12 mA	30	25	25	–	15
	8 mA	50	35	40	35	20
	4 mA	60	–	–	80	30
	2 mA	–	75	120	160	60
SSTL/HSTL Standards						
Conditions		**Output Pin Adder Delay (ps/pF)**				
		SSTL-3	SSTL-2	SSTL-1.8	1.5-V HSTL	1.8-V HSTL
Class I		25	25	25	25	25
Class II		25	20	25	20	20
GTL+/GTL/CTT/PCI Standards						
Conditions		**Output Pin Adder Delay (ps/pF)**				
Parameter	**Value**	GTL+	GTL	CTT	PCI	AGP
V_{CCIO} voltage level	3.3 V	18	18	25	20	20
	2.5 V	15	18	-	-	-

Maximum Input & Output Clock Rates

Tables 6–81 through 6–83 show the maximum input clock rate for column and row pins in Stratix GX devices.

Table 6–81. Stratix GX Maximum Input Clock Rate for CLK[7..4] & CLK[15..12] Pins

I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
LVTTL	422	422	390	MHz
2.5 V	422	422	390	MHz
1.8 V	422	422	390	MHz
1.5 V	422	422	390	MHz
LVCMOS	422	422	390	MHz
GTL	300	250	200	MHz
GTL+	300	250	200	MHz
SSTL-3 class I	400	350	300	MHz
SSTL-3 class II	400	350	300	MHz
SSTL-2 class I	400	350	300	MHz
SSTL-2 class II	400	350	300	MHz
SSTL-18 class I	400	350	300	MHz
SSTL-18 class II	400	350	300	MHz
1.5-V HSTL class I	400	350	300	MHz
1.5-V HSTL class II	400	350	300	MHz
1.8-V HSTL class I	400	350	300	MHz
1.8-V HSTL class II	400	350	300	MHz
3.3-V PCI	422	422	390	MHz
3.3-V PCI-X 1.0	422	422	390	MHz
Compact PCI	422	422	390	MHz
AGP 1×	422	422	390	MHz
AGP 2×	422	422	390	MHz
CTT	300	250	200	MHz
Differential HSTL	400	350	300	MHz
LVDS	645	645	622	MHz
LVPECL	645	645	622	MHz
PCML	300	275	275	MHz
HyperTransport technology	500	500	450	MHz

Table 6–82. Stratix GX Maximum Input Clock Rate for CLK[0, 2, 9, 11] Pins & FPLL[8..7]CLK Pins

I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
LVTTL	422	422	390	MHz
2.5 V	422	422	390	MHz
1.8 V	422	422	390	MHz
1.5 V	422	422	390	MHz
LVCMOS	422	422	390	MHz
GTL	300	250	200	MHz
GTL+	300	250	200	MHz
SSTL-3 class I	400	350	300	MHz
SSTL-3 class II	400	350	300	MHz
SSTL-2 class I	400	350	300	MHz
SSTL-2 class II	400	350	300	MHz
SSTL-18 class I	400	350	300	MHz
SSTL-18 class II	400	350	300	MHz
1.5-V HSTL class I	400	350	300	MHz
1.5-V HSTL class II	400	350	300	MHz
1.8-V HSTL class I	400	350	300	MHz
1.8-V HSTL class II	400	350	300	MHz
3.3-V PCI	422	422	390	MHz
3.3-V PCI-X 1.0	422	422	390	MHz
Compact PCI	422	422	390	MHz
AGP 1×	422	422	390	MHz
AGP 2×	422	422	390	MHz
CTT	300	250	200	MHz
Differential HSTL	400	350	300	MHz
LVDS	717	717	640	MHz
LVPECL	717	717	640	MHz
PCML	400	375	350	MHz
HyperTransport technology	717	717	640	MHz

Table 6–83. Stratix GX Maximum Input Clock Rate for CLK[1, 3, 8, 10] Pins (Part 1 of 2)

I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
LVTTL	422	422	390	MHz
2.5 V	422	422	390	MHz

Table 6–83. Stratix GX Maximum Input Clock Rate for CLK[1, 3, 8, 10] Pins (Part 2 of 2)				
I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
1.8 V	422	422	390	MHz
1.5 V	422	422	390	MHz
LVCMOS	422	422	390	MHz
GTL	300	250	200	MHz
GTL+	300	250	200	MHz
SSTL-3 class I	400	350	300	MHz
SSTL-3 class II	400	350	300	MHz
SSTL-2 class I	400	350	300	MHz
SSTL-2 class II	400	350	300	MHz
SSTL-18 class I	400	350	300	MHz
SSTL-18 class II	400	350	300	MHz
1.5-V HSTL class I	400	350	300	MHz
1.5-V HSTL class II	400	350	300	MHz
1.8-V HSTL class I	400	350	300	MHz
1.8-V HSTL class II	400	350	300	MHz
3.3-V PCI	422	422	390	MHz
3.3-V PCI-X 1.0	422	422	390	MHz
Compact PCI	422	422	390	MHz
AGP 1×	422	422	390	MHz
AGP 2×	422	422	390	MHz
CTT	300	250	200	MHz
Differential HSTL	400	350	300	MHz
LVDS	645	645	640	MHz
LVPECL	645	645	640	MHz
PCML	300	275	275	MHz
HyperTransport technology	645	645	640	MHz

Tables 6–84 and 6–85 show the maximum output clock rate for column and row pins in Stratix GX devices.

Table 6–84. Stratix GX Maximum Output Clock Rate for PLL[5, 6, 11, 12] Pins (Part 1 of 2)				
I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
LVTTL	350	300	250	MHz
2.5 V	350	300	300	MHz

Table 6–84. Stratix GX Maximum Output Clock Rate for PLL[5, 6, 11, 12] Pins (Part 2 of 2)

I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
1.8 V	250	250	250	MHz
1.5 V	225	200	200	MHz
LVCMOS	350	300	250	MHz
GTL	200	167	125	MHz
GTL+	200	167	125	MHz
SSTL-3 class I	167	150	133	MHz
SSTL-3 class II	167	150	133	MHz
SSTL-2 class I	200	200	167	MHz
SSTL-2 class II	200	200	167	MHz
SSTL-18 class I	150	133	133	MHz
SSTL-18 class II	150	133	133	MHz
1.5-V HSTL class I	250	225	200	MHz
1.5-V HSTL class II	225	200	200	MHz
1.8-V HSTL class I	250	225	200	MHz
1.8-V HSTL class II	225	200	200	MHz
3.3-V PCI	350	300	250	MHz
3.3-V PCI-X 1.0	350	300	250	MHz
Compact PCI	350	300	250	MHz
AGP 1×	350	300	250	MHz
AGP 2×	350	300	250	MHz
CTT	200	200	200	MHz
Differential HSTL	225	200	200	MHz
Differential SSTL-2	200	200	167	MHz
LVDS	500	500	500	MHz
LVPECL	500	500	500	MHz
PCML	350	350	350	MHz
HyperTransport technology	350	350	350	MHz

Table 6–85. Stratix GX Maximum Output Clock Rate (Using I/O Pins) for PLL[1, 2] Pins (Part 1 of 2)

I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
LVTTL	400	350	300	MHz
2.5 V	400	350	300	MHz
1.8 V	400	350	300	MHz

Table 6–85. Stratix GX Maximum Output Clock Rate (Using I/O Pins) for PLL[1, 2] Pins (Part 2 of 2)

I/O Standard	-5 Speed Grade	-6 Speed Grade	-7 Speed Grade	Unit
1.5 V	350	300	300	MHz
LVCMOS	400	350	300	MHz
GTL	200	167	125	MHz
GTL+	200	167	125	MHz
SSTL-3 class I	167	150	133	MHz
SSTL-3 class II	167	150	133	MHz
SSTL-2 class I	150	133	133	MHz
SSTL-2 class II	150	133	133	MHz
SSTL-18 class I	150	133	133	MHz
SSTL-18 class II	150	133	133	MHz
HSTL class I	250	225	200	MHz
HSTL class II	225	225	200	MHz
3.3-V PCI	250	225	200	MHz
3.3-V PCI-X 1.0	225	225	200	MHz
Compact PCI	400	350	300	MHz
AGP 1×	400	350	300	MHz
AGP 2×	400	350	300	MHz
CTT	300	250	200	MHz
Differential HSTL	225	225	200	MHz
LVDS	717	717	500	MHz
LVPECL	717	717	500	MHz
PCML	420	420	420	MHz
HyperTransport technology	420	420	420	MHz

High-Speed I/O Specification

Table 6–86 provides high-speed timing specifications definitions.

Table 6–86. High-Speed Timing Specifications & Definitions (Part 1 of 2)

High-Speed Timing Specification	Definitions
t_C	High-speed receiver/transmitter input and output clock period.
f_{HSCLK}	High-speed receiver/transmitter input and output clock frequency.
t_{RISE}	Low-to-high transmission time.

Table 6–86. High-Speed Timing Specifications & Definitions (Part 2 of 2)	
High-Speed Timing Specification	**Definitions**
t_{FALL}	High-to-low transmission time.
Timing unit interval (TUI)	The timing budget allowed for skew, propagation delays, and data sampling window. (TUI = 1/(Receiver Input Clock Frequency × Multiplication Factor) = t_C/w).
f_{HSDR}	Maximum/minimum LVDS data transfer rate (f_{HSDR} = 1/TUI), non-DPA.
$f_{HSDRDPA}$	Maximum/minimum LVDS data transfer rate ($f_{HSDRDPA}$ = 1/TUI), DPA.
Channel-to-channel skew (TCCS)	The timing difference between the fastest and slowest output edges, including t_{CO} variation and clock skew. The clock is included in the TCCS measurement.
Sampling window (SW)	The period of time during which the data must be valid in order to capture it correctly. The setup and hold times determine the ideal strobe position within the sampling window. SW = t_{SW} (max) – t_{SW} (min).
Input jitter (peak-to-peak)	Peak-to-peak input jitter on high-speed PLLs.
Output jitter (peak-to-peak)	Peak-to-peak output jitter on high-speed PLLs.
t_{DUTY}	Duty cycle on high-speed transmitter output clock.
t_{LOCK}	Lock time for high-speed transmitter and receiver PLLs.

Table 6–87 shows the high-speed I/O timing specifications for Stratix GX devices.

Table 6–87. High-Speed I/O Specifications (Part 1 of 4) Notes (1), (2)											
Symbol	**Conditions**	**-5 Speed Grade**			**-6 Speed Grade**			**-7 Speed Grade**			**Unit**
		Min	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	
f_{HSCLK} (Clock frequency) (LVDS, LVPECL, HyperTransport technology) f_{HSCLK} = f_{HSDR} / W	*W* = 1 to 30 for ≤717 Mbps W = 2 to 30 for > 717 Mbps	10		717	10		717	10		624	MHz
f_{HSCLK_DPA}		74		717	74		717	74		717	MHz

Table 6–87. High-Speed I/O Specifications (Part 2 of 4) *Notes (1), (2)*

Symbol	Conditions	-5 Speed Grade			-6 Speed Grade			-7 Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
f_{HSDR} Device operation (LVDS, LVPECL, HyperTransport technology)	*J* = 10	300		840	300		840	300		840	Mbps
	J = 8	300		840	300		840	300		840	Mbps
	J = 7	300		840	300		840	300		840	Mbps
	J = 4	300		840	300		840	300		840	Mbps
	J = 2	100		624	100		624	100		462	Mbps
	J = 1 (LVDS and LVPECL only)	100		462	100		462	100		462	Mbps
$f_{HSDRDPA}$ (LVDS, LVPECL)	J=10	300		1000	300		840	300		840	Mbps
	J=8	300		1000	300		840	300		840	Mbps
f_{HSCLK} (Clock frequency) (PCML) f_{HSCLK} = f_{HSDR} / W	*W* = 1 to 30	10		400	10		400	10		311	MHz
f_{HSDR} Device operation (PCML)	*J* = 10	300		400	300		400	300		311	Mbps
	J = 8	300		400	300		400	300		311	Mbps
	J = 7	300		400	300		400	300		311	Mbps
	J = 4	300		400	300		400	300		311	Mbps
	J = 2	100		400	100		400	100		300	Mbps
	J = 1	100		250	100		250	100		200	Mbps
DPA Run Length				6400			6400			6400	UI
DPA Jitter Tolerance$_{(p-p)}$	all data rates			0.44			0.44			0.44	UI
DPA Minimum Eye opening (p-p)		0.56			0.56			0.56			UI
DPA Receiver Latency		5		9	5		9	5		9	*(3)*

Table 6–87. High-Speed I/O Specifications (Part 3 of 4) *Notes (1), (2)*

Symbol	Conditions			-5 Speed Grade			-6 Speed Grade			-7 Speed Grade			Unit
				Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
DPA Lock Time	Standard	Training Pattern	Transition Density										
	SPI-4, CSIX	0000 0000 0011 1111 1111	10%	256			256			256			*(4)*
	Rapid IO	0000 1111	25%	256			256			256			*(4)*
		1001 0000	50%	256			256			256			*(4)*
	Misc	1010 1010	100 %	256			256			256			*(4)*
		0101 0101		256			256			256			*(4)*
TCCS	All					200			200			300	ps
SW	PCML (J = 4, 7, 8, 10)			750			750			800			ps
	PCML (J = 2)			900			900			1,200			ps
	PCML (J = 1)			1,500			1,500			1,700			ps
	LVDS and LVPECL (J = 1)			500			500			550			ps
	LVDS, LVPECL, HyperTransport technology (J = 2 through 10)			440			440			500			ps
Input jitter tolerance (peak-to-peak)	All					250			250			250	ps
Output jitter (peak-to-peak)	All					160			160			200	ps
Output t_{RISE}	LVDS			80	110	120	80	110	120	80	110	120	ps
	HyperTransport technology			110	170	200	110	170	200	120	170	200	ps
	LVPECL			90	130	150	90	130	150	100	135	150	ps
	PCML			80	110	135	80	110	135	80	110	135	ps

Table 6–87. High-Speed I/O Specifications (Part 4 of 4) *Notes (1), (2)*

Symbol	Conditions	-5 Speed Grade			-6 Speed Grade			-7 Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Output t_{FALL}	LVDS	80	110	120	80	110	120	80	110	120	ps
	HyperTransport technology	110	170	200	110	170	200	110	170	200	ps
	LVPECL	90	130	160	90	130	160	100	135	160	ps
	PCML	105	140	175	105	140	175	110	145	175	ps
t_{DUTY}	LVDS (*J* = 2 through 10)	47.5	50	52.5	47.5	50	52.5	47.5	50	52.5	%
	LVDS (*J* =1) and LVPECL, PCML, HyperTransport technology	45	50	55	45	50	55	45	50	55	%
t_{LOCK}	All			100			100			100	µs

Notes to Table 6–87:
(1) When J = 4, 7, 8, and 10, the SERDES block is used.
(2) When J = 2 or J = 1, the SERDES is bypassed.
(3) Number of parallel CLK cycles.
(4) Number of repetitions.

PLL Timing

Tables 6–88 through 6–90 describe the Stratix GX device enhanced PLL specifications.

Table 6–88. Enhanced PLL Specifications for -5 Speed Grades (Part 1 of 2)

Symbol	Parameter	Min	Typ	Max	Unit
f_{IN}	Input clock frequency	3 *(1)*		684	MHz
f_{INDUTY}	Input clock duty cycle	40		60	%
$f_{EINDUTY}$	External feedback clock input duty cycle	40		60	%
$t_{INJITTER}$	Input clock period jitter			±200 *(2)*	ps
$t_{EINJITTER}$	External feedback clock period jitter			±200 *(2)*	ps
t_{FCOMP}	External feedback clock compensation time *(3)*			6	ns
f_{OUT}	Output frequency for internal global or regional clock	0.3		500	MHz
f_{OUT_EXT}	Output frequency for external clock *(2)*	0.3		526	MHz

Table 6–88. Enhanced PLL Specifications for -5 Speed Grades (Part 2 of 2)

Symbol	Parameter	Min	Typ	Max	Unit
$t_{OUTDUTY}$	Duty cycle for external clock output (when set to 50%)	45		55	%
t_{JITTER}	Period jitter for external clock output *(5)*			±100 ps for >200 MHz `outclk` ±20 mUI for <200 MHz `outclk`	ps or mUI
$t_{CONFIG5,6}$	Time required to reconfigure the scan chains for PLLs 5 and 6			$289/f_{SCANCLK}$	
$t_{CONFIG11,12}$	Time required to reconfigure the scan chains for PLLs 11 and 12			$193/f_{SCANCLK}$	
$t_{SCANCLK}$	`scanclk` frequency *(4)*			22	MHz
t_{DLOCK}	Time required to lock dynamically (after switchover or reconfiguring any non-post-scale counters/delays) *(6)*			100	µs
t_{LOCK}	Time required to lock from end of device configuration	10		400	µs
f_{VCO}	PLL internal VCO operating range	300		800 *(7)*	MHz
t_{LSKEW}	Clock skew between two external clock outputs driven by the same counter		±50		ps
t_{SKEW}	Clock skew between two external clock outputs driven by the different counters with the same settings		±75		ps
f_{SS}	Spread spectrum modulation frequency	30		150	kHz
% spread	Percentage spread for spread spectrum frequency *(9)*	0.4	0.5	0.6	%
t_{ARESET}	Minimum pulse width on `areset` signal	10			ns

Table 6–89. Enhanced PLL Specifications for -6 Speed Grades (Part 1 of 2)

Symbol	Parameter	Min	Typ	Max	Unit
f_{IN}	Input clock frequency	3 *(1)*		650	MHz
f_{INDUTY}	Input clock duty cycle	40		60	%
$f_{EINDUTY}$	External feedback clock input duty cycle	40		60	%
$t_{INJITTER}$	Input clock period jitter			±200 *(2)*	ps
$t_{EINJITTER}$	External feedback clock period jitter			±200 *(2)*	ps
t_{FCOMP}	External feedback clock compensation time *(3)*			6	ns

Table 6–89. Enhanced PLL Specifications for -6 Speed Grades (Part 2 of 2)

Symbol	Parameter	Min	Typ	Max	Unit
f_{OUT}	Output frequency for internal global or regional clock	0.3		450	MHz
f_{OUT_EXT}	Output frequency for external clock *(2)*	0.3		500	MHz
$t_{OUTDUTY}$	Duty cycle for external clock output (when set to 50%)	45		55	%
t_{JITTER}	Period jitter for external clock output *(5)*			±100 ps for >200 MHz `outclk` ±20 mUI for <200 MHz `outclk`	ps or mUI
$t_{CONFIG5,6}$	Time required to reconfigure the scan chains for PLLs 5 and 6			$289/f_{SCANCLK}$	
$t_{CONFIG11,12}$	Time required to reconfigure the scan chains for PLLs 11 and 12			$193/f_{SCANCLK}$	
$t_{SCANCLK}$	`scanclk` frequency *(4)*			22	MHz
t_{DLOCK}	Time required to lock dynamically (after switchover or reconfiguring any non-post-scale counters/delays) *(6)* *(10)*	*(8)*		100	µs
t_{LOCK}	Time required to lock from end of device configuration *(10)*	10		400	µs
f_{VCO}	PLL internal VCO operating range	300		800 *(7)*	MHz
t_{LSKEW}	Clock skew between two external clock outputs driven by the same counter		±50		ps
t_{SKEW}	Clock skew between two external clock outputs driven by the different counters with the same settings		±75		ps
f_{SS}	Spread spectrum modulation frequency	30		150	kHz
% spread	Percentage spread for spread spectrum frequency *(9)*	0.4	0.5	0.6	%
t_{ARESET}	Minimum pulse width on `areset` signal	10			ns

Table 6–90. Enhanced PLL Specifications for -7 Speed Grade (Part 1 of 3)

Symbol	Parameter	Min	Typ	Max	Unit
f_{IN}	Input clock frequency	3 *(1)*		565	MHz
f_{INDUTY}	Input clock duty cycle	40		60	%
$f_{EINDUTY}$	External feedback clock input duty cycle	40		60	%
$t_{INJITTER}$	Input clock period jitter			±200 *(2)*	ps
$t_{EINJITTER}$	External feedback clock period jitter			±200 *(2)*	ps

Table 6–90. Enhanced PLL Specifications for -7 Speed Grade (Part 2 of 3)

Symbol	Parameter	Min	Typ	Max	Unit
t_{FCOMP}	External feedback clock compensation time *(3)*			6	ns
f_{OUT}	Output frequency for internal global or regional clock	0.3		420	MHz
f_{OUT_EXT}	Output frequency for external clock *(2)*	0.3		434	MHz
$t_{OUTDUTY}$	Duty cycle for external clock output (when set to 50%)	45		55	%
t_{JITTER}	Period jitter for external clock output *(5)*			±100 ps for >200 MHz `outclk` ±20 mUI for <200 MHz `outclk`	ps or mUI
$t_{CONFIG5,6}$	Time required to reconfigure the scan chains for PLLs 5 and 6			$289/f_{SCANCLK}$	
$t_{CONFIG11,12}$	Time required to reconfigure the scan chains for PLLs 11 and 12			$193/f_{SCANCLK}$	
$t_{SCANCLK}$	`scanclk` frequency *(4)*			22	MHz
t_{DLOCK}	Time required to lock dynamically (after switchover or reconfiguring any non-post-scale counters/delays) *(6)* *(10)*	*(8)*		100	µs
t_{LOCK}	Time required to lock from end of device configuration *(10)*	10		400	µs
f_{VCO}	PLL internal VCO operating range	300		600 *(7)*	MHz

Table 6–90. Enhanced PLL Specifications for -7 Speed Grade (Part 3 of 3)

Symbol	Parameter	Min	Typ	Max	Unit
t_{LSKEW}	Clock skew between two external clock outputs driven by the same counter		±50		ps
t_{SKEW}	Clock skew between two external clock outputs driven by the different counters with the same settings		±75		ps
f_{SS}	Spread spectrum modulation frequency	30		150	kHz
% spread	Percentage spread for spread spectrum frequency *(9)*	0.5		0.6	%
t_{ARESET}	Minimum pulse width on `areset` signal	10			ns

Notes to Tables 6–88 through 6–90:

(1) The minimum input clock frequency to the PFD (f_{IN}/N) must be at least 3 MHz for Stratix device enhanced PLLs.
(2) See "Maximum Input & Output Clock Rates" on page 6–54.
(3) t_{FCOMP} can also equal 50% of the input clock period multiplied by the pre-scale divider *n* (whichever is less).
(4) This parameter is timing analyzed by the Quartus II software because the `scanclk` and `scandata` ports can be driven by the logic array.
(5) Actual jitter performance may vary based on the system configuration.
(6) Total required time to reconfigure and lock is equal to $t_{DLOCK} + t_{CONFIG}$. If only post-scale counters and delays are changed, then t_{DLOCK} is equal to 0.
(7) The VCO range is limited to 500 to 800 MHz when the spread spectrum feature is selected.
(8) Lock time is a function of PLL configuration and may be significantly faster depending on bandwidth settings or feedback counter change increment.
(9) Exact, user-controllable value depends on the PLL settings.
(10) The LOCK circuit on Stratix PLLs does not work for industrial devices below -20C unless the PFD frequency > 200 MHz. See the *Stratix FPGA Errata Sheet* for more information on the PLL.

Table 6–91 describes the Stratix GX device fast PLL specifications.

Table 6–91. Fast PLL Specifications for -5 & -6 Speed Grade Devices

Symbol	Parameter	Min	Max	Unit
f_{IN}	CLKIN frequency (for *m* = 1) *(1)*	300	717	MHz
	CLKIN frequency (for *m* = 2 to 19)	300/*m*	1,000/*m*	MHz
	CLKIN frequency (for *m* = 20 to 32)	10	1,000/*m*	MHz
f_{OUT}	Output frequency for internal global or regional clock *(2)*	9.4	420	MHz
f_{OUT_EXT}	Output frequency for external clock	9.375	717	MHz
f_{VCO}	VCO operating frequency	300	1,000	MHz
t_{INDUTY}	CLKIN duty cycle	40	60	%
$t_{INJITTER}$	Period jitter for CLKIN pin		±200	ps
t_{DUTY}	Duty cycle for DFFIO 1× CLKOUT pin *(3)*	45	55	%
t_{JITTER}	Period jitter for DIFFIO clock out *(3)*		±80	ps
	Period jitter for internal global or regional clock		±100 ps for >200-MHz outclk ±20 mUI for <200-MHz outclk	ps or mUI
t_{LOCK}	Time required for PLL to acquire lock	10	100	µs
m	Multiplication factors for *m* counter *(3)*	1	32	Integer
*l*0, *l*1, *g*0	Multiplication factors for *l*0, *l*1, and *g*0 counter *(4)*, *(5)*	1	32	Integer
t_{ARESET}	Minimum pulse width on areset signal	10		ns

Table 6–92. Fast PLL Specifications for -7 & -8 Speed Grades (Part 1 of 2)

Symbol	Parameter	Min	Max	Unit
f_{IN}	CLKIN frequency (for *m* = 1) *(1)*,	300	640	MHz
	CLKIN frequency (for *m* = 2 to 19)	300/*m*	700/*m*	MHz
	CLKIN frequency (for *m* = 20 to 32)	10	700/*m*	MHz
f_{OUT}	Output frequency for internal global or regional clock *(2)*	9.375	420	MHz
f_{OUT_EXT}	Output frequency for external clock	9.4	500	MHz
f_{VCO}	VCO operating frequency	300	700	MHz
t_{INDUTY}	CLKIN duty cycle	40	60	%
$t_{INJITTER}$	Period jitter for CLKIN pin		±200	ps

Table 6–92. Fast PLL Specifications for -7 & -8 Speed Grades (Part 2 of 2)

Symbol	Parameter	Min	Max	Unit
t_{DUTY}	Duty cycle for DFFIO 1× CLKOUT pin *(3)*	45	55	%
t_{JITTER}	Period jitter for DIFFIO clock out *(3)*		±80	ps
	Period jitter for internal global or regional clock		±100 ps for >200 MHz outclk ±20 mUI for <200 MHz outclk	ps or mUI
t_{LOCK}	Time required for PLL to acquire lock	10	100	µs
m	Multiplication factors for *m* counter *(4)*	1	32	Integer
*l*0, *l*1, *g*0	Multiplication factors for *l*0, *l*1, and *g*0 counter *(4)*, *(5)*	1	32	Integer
t_{ARESET}	Minimum pulse width on areset signal	10		ns

Notes to Tables 6–91 & 6–92:

(1) See "Maximum Input & Output Clock Rates" on page 6–54.
(2) When using the SERDES, high-speed differential I/O mode supports a maximum output frequency of 210 MHz to the global or regional clocks (that is, the maximum data rate 840 Mbps divided by the smallest SERDES J factor of 4).
(3) This parameter is for high-speed differential I/O mode only.
(4) These counters have a maximum of 32 if programmed for 50/50 duty cycle. Otherwise, they have a maximum of 16.
(5) High-speed differential I/O mode supports $W = 1$ to 16 and $J = 4, 7, 8,$ or 10.

DLL Jitter

Table 6–93 reports the jitter for the DLL in the DQS phase-shift reference circuit.

Table 6–93. DLL Jitter for DQS Phase Shift Reference Circuit

Frequency (MHz)	DLL Jitter (ps)
197 to 200	± 100
160 to 196	± 300
100 to 159	± 500

7. Reference & Ordering Information

SGX51007-1.0

Software

Stratix® GX devices are supported by the Altera® Quartus® II design software, which provides a comprehensive environment for system-on-a-programmable-chip (SOPC) design. The Quartus II software includes hardware description language and schematic design entry, compilation and logic synthesis, full simulation and advanced timing analysis, SignalTap® logic analysis, and device configuration. See the *Design Software Selector Guide* for more details on the Quartus II software features.

The Quartus II software supports the Windows 2000/NT/98, Sun Solaris, Linux Red Hat v6.2 and HP-UX operating systems. It also supports seamless integration with industry-leading EDA tools through the NativeLink® interface.

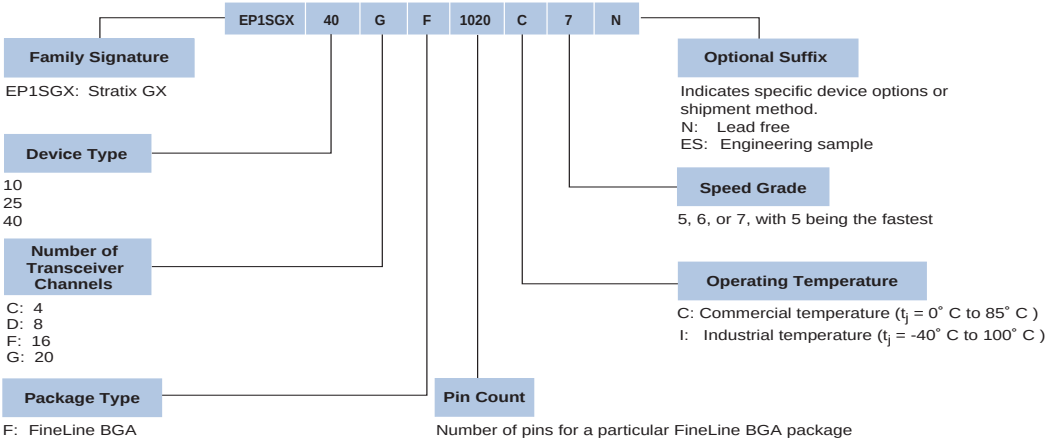
Device Pin-Outs

Device pin-outs for Stratix GX devices will be released on the Altera web site (**www.altera.com**).

Ordering Information

Figure 7–1 describes the ordering codes for Stratix GX devices.

Figure 7–1. Stratix GX Device Packaging Ordering Information





Stratix GX Device Handbook, Volume 2



101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com

SGX5V2-2.0

Copyright © 2006 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Altera Lead-Free

I.S. EN ISO 9001

Contents

Chapter Revision Dates ..... **xiii**

About This Handbook ..... **xv**
How to Contact Altera xv
Typographic Conventions xv

Section I. Stratix GX Transceiver User Guide
Revision History ..... Section I–1

Chapter 1. Introduction
Gigabit Transceiver Block Highlights 1–1
Transceiver Block Architecture 1–2
Analog Section Overview 1–2
Digital Overview 1–3
Modes of Operation 1–5
Custom Mode 1–5
SONET Mode 1–6
XAUI Mode 1–7
GIGE Mode 1–8
Loopback 1–9
Built-In Self Test 1–9

Chapter 2. Stratix GX Analog Description
Introduction 2–1
Transmitter Analog 2–2
Transmitter Buffer 2–2
Transmitter PLL 2–5
Serializer (Parallel-to-Serial Converter) 2–8
Receiver Analog 2–9
Receiver Input Buffer 2–9
Receiver PLL 2–12
Clock Recovery Unit 2–16
Deserializer (Serial-to-Parallel Converter) 2–19
MegaWizard Plug-In Manager Analog Features 2–20
MegaWizard Plug-In Manager Analog Feature Considerations 2–20
Stratix GX Transceiver Merging 2–30

Chapter 3. Custom Mode

Introduction 3–1
Custom Mode Receiver Architecture 3–2
Word Aligner 3–2
8B/10B Decoder 3–9
Byte Deserializer 3–13
Receiver Phase Compensation FIFO Buffer 3–15
Custom Mode Transmitter Architecture 3–16
Transmitter Phase Compensation FIFO Buffer 3–16
Byte Serializer 3–17
8B/10B Encoder 3–17
Custom Mode Clocking 3–20
Custom Mode Channel Clocking 3–20
Custom Mode Inter-Transceiver Block Clocking 3–24
Custom Mode MegaWizard Plug-In Manager 3–29
Custom Mode MegaWizard Plug-In Manager Considerations 3–30
Custom Mode altgxb MegaWizard Plug-In Manager Options 3–30
Stratix GX Transceiver Merging 3–43

Chapter 4. SONET Mode

Introduction 4–1
SONET Mode Receiver Architecture 4–2
Word Aligner 4–2
Byte Deserializer 4–8
Receiver Phase Compensation FIFO Module 4–10
SONET Mode Transmitter Architecture 4–11
Transmitter Phase Compensation FIFO Buffer 4–11
Byte Serializer 4–12
SONET Mode Clocking 4–12
SONET Mode Channel Clocking 4–12
SONET Mode Inter-Transceiver Block Clocking 4–17
SONET Mode MegaWizard Plug-In Manager 4–23
SONET Mode MegaWizard Plug-In Manager Considerations 4–23
SONET Mode altgxb MegaWizard Plug-In Manager Options 4–23
Stratix GX Transceiver Merging 4–36

Chapter 5. XAUI Mode

Introduction 5–1
XAUI Mode Receiver Architecture 5–5
Word Aligner 5–6
Channel Aligner 5–9
Rate Matcher 5–11
8B/10B Decoder 5–11
PCS - XGMII Code Conversion 5–15
Byte Deserializer 5–15
Receiver Phase Compensation FIFO Module 5–18
XAUI Mode Transmitter Architecture 5–18

Transmitter Phase Compensation FIFO Module 5–18
Byte Serializer 5–19
XGMII Character to PCS Code-Group Mapping 5–20
8B/10B Encoder 5–22
XAUI Mode Clocking 5–25
XAUI Mode Channel Clocking 5–25
XAUI Inter-Transceiver Block Clocking 5–29
XAUI Mode MegaWizard Plug-In Manager 5–36
XAUI Mode MegaWizard Plug-In Manager Considerations 5–36
XAUI Mode altgxb MegaWizard Plug-In Manager Options 5–36
Stratix GX Transceiver Merging 5–47

Chapter 6. GIGE Mode

Introduction 6–1
Word Aligner 6–4
Rate Matcher 6–9
8B/10B Decoder 6–11
Receiver Phase Compensation FIFO Buffer 6–14
GIGE Mode Transmitter Architecture 6–14
Transmitter Phase Compensation FIFO Buffer 6–15
GIGE Transmitter Synchronization 6–16
Idle Generation 6–16
8B/10B Encoder 6–17
GIGE Mode Clocking 6–20
GIGE Mode Channel Clocking 6–20
GIGE Mode Inter-Transceiver Clocking 6–25
GIGE Mode MegaWizard Plug-In Manager 6–31
GIGE Mode MegaWizard Plug-In Manager Considerations 6–31
GIGE Mode altgxb MegaWizard Plug-In Manager Options 6–31
Stratix GX Transceiver Merging 6–43
Design Example 6–45
Design Description 6–45
Simulation Waveform & Hardware Verification Results 6–51

Chapter 7. Loopback Modes

Introduction 7–1
Serial Loopback 7–1
Parallel Loopback 7–2
Reverse Serial Loopback 7–3

Chapter 8. Stratix GX Built-In Self Test (BIST)

Introduction 8–1
Pattern Generator 8–2
PRBS Mode Generator 8–2
Incremental Mode Generator 8–3
High-Frequency Mode Generator 8–3
Low-Frequency Mode Generator 8–4

Mix-Frequency Mode Generator 8–5
Pattern Verifier 8–5
PRBS Mode Verifier 8–5
Incremental Mode Verifier 8–6
Design Examples 8–7
Design 1: PRBS BIST Generator & Verification Design 8–7
Design 2: Incremental BIST Generator & Verification Design 8–11
Design 3: High-Frequency Transmitter Generator Design 8–16
Design 4: Low-Frequency Transmitter Generator Design 8–18
Design 5: Mix-Frequency Transmitter Generator Design 8–20

Chapter 9. Reset Control & Power Down

Introduction 9–1
Power On Reset (POR) 9–1
USER Reset & Enable Signals 9–1
Recommended Resets 9–4
Receiver & Transmitter Reset 9–4
Receiver Reset 9–32
Transmitter Reset 9–52
Power Down 9–57

Chapter 10. Data & Control Codes

8B/10B Code 10–1
Code Notation 10–1
Disparity Calculation 10–1
Supported Codes 10–3

Chapter 11. Ports & Parameters

Input Ports 11–1
Output Ports 11–5
Parameter Descriptions 11–9

Chapter 12. REFCLKB Pin Constraints

Known Issues 12–1
Quartus II Software Messages 12–3
Recommendations 12–5

Section II. Clock Management

Revision History ..... Section II–1

Chapter 13. General-Purpose PLLs in Stratix & Stratix GX Devices

Introduction 13–1
Enhanced PLLs 13–5

Clock Multiplication & Division 13–9
External Clock Outputs 13–10
Clock Feedback 13–14
Phase Shifting 13–14
Lock Detect 13–15
Programmable Duty Cycle 13–16
General Advanced Clear & Enable Control 13–16
Programmable Bandwidth 13–18
Clock Switchover 13–25
Spread-Spectrum Clocking 13–25
PLL Reconfiguration 13–30
Enhanced PLL Pins 13–30
Fast PLLs 13–31
Clock Multiplication & Division 13–34
External Clock Outputs 13–34
Phase Shifting 13–35
Programmable Duty Cycle 13–36
Control Signals 13–36
Pins 13–37
Clocking 13–39
Global & Hierarchical Clocking 13–39
Clock Input Connections 13–41
Clock Output Connections 13–43
Board Layout 13–50
VCCA & GNDA 13–50
VCCG & GNDG 13–52
External Clock Output Power 13–53
Guidelines 13–56
Conclusion 13–56

Section III. Memory

Revision History ..... Section III–1

Chapter 14. TriMatrix Embedded Memory Blocks in Stratix & Stratix GX Devices

Introduction 14–1
TriMatrix Memory 14–1
Clear Signals 14–3
Parity Bit Support 14–3
Byte Enable Support 14–4
Using TriMatrix Memory 14–7
Implementing Single-Port Mode 14–7
Implementing Simple Dual-Port Mode 14–8
Implementing True Dual-Port Mode 14–11
Implementing Shift-Register Mode 14–14
Implementing ROM Mode 14–15

Implementing FIFO Buffers 14–16
Clock Modes 14–16
Independent Clock Mode 14–16
Input/Output Clock Mode 14–18
Read/Write Clock Mode 14–21
Single-Port Mode 14–23
Designing With TriMatrix Memory 14–23
Selecting TriMatrix Memory Blocks 14–24
Pipeline & Flow-Through Modes 14–24
Power-up Conditions & Memory Initialization 14–25
Read-During-Write Operation at the Same Address 14–25
Same-Port Read-During-Write Mode 14–25
Mixed-Port Read-During-Write Mode 14–26
Conclusion 14–27

Chapter 15. External Memory Interfaces in Stratix & Stratix GX Devices

Introduction 15–1
External Memory Standards 15–1
DDR SDRAM 15–1
RLDRAM II 15–4
QDR & QDRII SRAM 15–6
ZBT SRAM 15–8
DDR Memory Support Overview 15–10
DDR Memory Interface Pins 15–11
DQS Phase-Shift Circuitry 15–15
DDR Registers 15–20
PLL 15–27
Conclusion 15–27

Section IV. I/O Standards

Revision History ..... Section IV–1

Chapter 16. Selectable I/O Standards in Stratix & Stratix GX Devices

Introduction 16–1
Stratix & Stratix GX I/O Standards 16–1
3.3-V Low Voltage Transistor-Transistor Logic (LVTTL) - EIA/JEDEC Standard JESD8-B 16–2
3.3-V LVCMOS - EIA/JEDEC Standard JESD8-B 16–3
2.5-V LVTTL Normal Voltage Range - EIA/JEDEC Standard EIA/JESD8-5 16–3
2.5-V LVCMOS Normal Voltage Range - EIA/JEDEC Standard EIA/JESD8-5 16–3
1.8-V LVTTL Normal Voltage Range - EIA/JEDEC Standard EIA/JESD8-7 16–4
1.8-V LVCMOS Normal Voltage Range - EIA/JEDEC Standard EIA/JESD8-7 16–4
1.5-V LVCMOS Normal Voltage Range - EIA/JEDEC Standard JESD8-11 16–4
1.5-V HSTL Class I & II - EIA/JEDEC Standard EIA/JESD8-6 16–5
1.5-V Differential HSTL - EIA/JEDEC Standard EIA/JESD8-6 16–6

3.3-V PCI Local Bus - PCI Special Interest Group PCI Local Bus Specification Rev. 2.3 16–6
3.3-V PCI-X 1.0 Local Bus - PCI-SIG PCI-X Local Bus Specification Revision 1.0a 16–7
3.3-V Compact PCI Bus - PCI SIG PCI Local Bus Specification Revision 2.3 16–7
3.3-V 1× AGP - Intel Corporation Accelerated Graphics Port Interface Specification 2.0 ... 16–7
3.3-V 2× AGP - Intel Corporation Accelerated Graphics Port Interface Specification 2.0 ... 16–8
GTL - EIA/JEDEC Standard EIA/JESD8-3 16–8
GTL+ 16–8
CTT - EIA/JEDEC Standard JESD8-4 16–9
SSTL-3 Class I & II - EIA/JEDEC Standard JESD8-8 16–9
SSTL-2 Class I & II - EIA/JEDEC Standard JESD8-9A 16–10
SSTL-18 Class I & II - EIA/JEDEC Preliminary Standard JC42.3 16–11
Differential SSTL-2 - EIA/JEDEC Standard JESD8-9A 16–11
LVDS - ANSI/TIA/EIA Standard ANSI/TIA/EIA-644 16–12
LVPECL 16–13
Pseudo Current Mode Logic (PCML) 16–13
HyperTransport Technology - HyperTransport Consortium 16–14
High-Speed Interfaces 16–15
OIF-SPI4.2 16–15
OIF-SFI4.1 16–15
10 Gigabit Ethernet Sixteen Bit Interface (XSBI) - IEEE Draft Standard P802.3ae/D2.0 16–16
RapidIO Interconnect Specification Revision 1.1 16–16
HyperTransport Technology - HyperTransport Consortium 16–17
UTOPIA Level 4 – ATM Forum Technical Committee Standard AF-PHY-0144.001 16–17
Stratix & Stratix GX I/O Banks 16–17
Non-Voltage-Referenced Standards 16–24
Voltage-Referenced Standards 16–24
Mixing Voltage Referenced & Non-Voltage Referenced Standards 16–25
Drive Strength 16–26
Standard Current Drive Strength 16–26
Programmable Current Drive Strength 16–27
Hot Socketing 16–27
DC Hot Socketing Specification 16–28
AC Hot Socketing Specification 16–28
I/O Termination 16–28
Voltage-Referenced I/O Standards 16–28
Differential I/O Standards 16–29
Differential Termination (RD) 16–29
Transceiver Termination 16–30
I/O Pad Placement Guidelines 16–30
Differential Pad Placement Guidelines 16–30
VREF Pad Placement Guidelines 16–31
Output Enable Group Logic Option in Quartus II 16–34
Toggle Rate Logic Option in Quartus II 16–35
DC Guidelines 16–35
Power Source of Various I/O Standards 16–38
Quartus II Software Support 16–38
Compiler Settings 16–38

Device & Pin Options 16–39
Assign Pins 16–39
Programmable Drive Strength Settings 16–40
I/O Banks in the Floorplan View 16–40
Auto Placement & Verification of Selectable I/O Standards 16–41
Conclusion 16–42
More Information 16–42
References 16–42

Chapter 17. High-Speed Source-Synchronous Differential I/O Interfaces in Stratix GX Devices

Introduction 17–1
Skew & Dynamic Phase Alignment 17–1
Stratix GX I/O Banks 17–2
Dedicated Source-Synchronous Circuitry 17–3
DPA Block Overview 17–5
DPA Input Support 17–6
Interface & Fast PLL 17–7
DPA Operation 17–10
Protocols, Training Pattern & DPA Lock Time 17–11
Phase Synchronizer 17–12
Receiver Data Realignment In DPA Mode 17–12
Source-Synchronous Circuitry with DPA vs. CDR 17–16
Software Support 17–16
Stratix GX Transmitter 17–18
Stratix GX Receiver Without DPA 17–20
Stratix GX Receiver with DPA 17–23
Summary 17–28

Section V. Digital Signal Processing

Revision History ..... Section V–1

Chapter 18. DSP Blocks in Stratix & Stratix GX Devices

Introduction 18–1
DSP Block Overview 18–2
Architecture 18–5
Multiplier Block 18–5
Adder/Output Block 18–9
Routing Structure & Control Signals 18–12
Operational Modes 18–18
Simple Multiplier Mode 18–18
Multiply Accumulator Mode 18–22
Two-Multiplier Adder Mode 18–23
Four-Multiplier Adder Mode 18–24
Software Support 18–28

Conclusion 18–28

Chapter 19. Implementing High Performance DSP Functions in Stratix & Stratix GX Devices

Introduction 19–1
Stratix & Stratix GX DSP Block Overview 19–1
TriMatrix Memory Overview 19–4
DSP Function Overview 19–5
Finite Impulse Response (FIR) Filters 19–5
FIR Filter Background 19–6
Basic FIR Filter 19–7
Time-Domain Multiplexed FIR Filters 19–13
Polyphase FIR Interpolation Filters 19–17
Polyphase FIR Decimation Filters 19–24
Complex FIR Filter 19–31
Infinite Impulse Response (IIR) Filters 19–34
IIR Filter Background 19–34
Basic IIR Filters 19–36
Butterworth IIR Filters 19–39
Matrix Manipulation 19–45
Background on Matrix Manipulation 19–45
Two-Dimensional Filtering & Video Imaging 19–46
Discrete Cosine Transform (DCT) 19–52
DCT Background 19–52
2-D DCT Algorithm 19–53
Arithmetic Functions 19–59
Background 19–59
Arithmetic Function Implementation 19–60
Arithmetic Function Implementation Results 19–62
Arithmetic Function Design Example 19–62
Conclusion 19–62
References 19–63

Chapter Revision Dates

The chapters in this book, *Stratix GX Device Handbook, Volume 2*, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1. Introduction
Revised: *June 2006*
Part number: *SGX52001-1.2*

Chapter 2. Stratix GX Analog Description
Revised: *June 2006*
Part number: *SGX52002-1.2*

Chapter 3. Custom Mode
Revised: *June 2006*
Part number: *SGX52003-1.2*

Chapter 4. SONET Mode
Revised: *June 2006*
Part number: *SGX52004-1.2*

Chapter 5. XAUI Mode
Revised: *June 2006*
Part number: *SGX52005-1.2*

Chapter 6. GIGE Mode
Revised: *June 2006*
Part number: *SGX52006-1.2*

Chapter 7. Loopback Modes
Revised: *June 2006*
Part number: *SGX52007-1.2*

Chapter 8. Stratix GX Built-In Self Test (BIST)
Revised: *August 2005*
Part number: *SGX52008-1.1*

Chapter 9. Reset Control & Power Down
Revised: *February 2005*
Part number: *SGX52009-1.0*

Chapter 10. Data & Control Codes
Revised: *February 2005*
Part number: *SGX52010-1.0*

Chapter 11. Ports & Parameters
Revised: *June 2006*
Part number: *SGX52011-1.1*

Chapter 12. REFCLKB Pin Constraints
Revised: *February 2005*
Part number: *SGX52012-1.0*

Chapter 13. General-Purpose PLLs in Stratix & Stratix GX Devices
Revised: *July 2005*
Part number: *S52001-3.2*

Chapter 14. TriMatrix Embedded Memory Blocks in Stratix & Stratix GX Devices
Revised: *July 2005*
Part number: *S52003-3.3*

Chapter 15. External Memory Interfaces in Stratix & Stratix GX Devices
Revised: *June 2006*
Part number: *SII52003-3.3*

Chapter 16. Selectable I/O Standards in Stratix & Stratix GX Devices
Revised: *June 2006*
Part number: *S52004-3.4*

Chapter 17. High-Speed Source-Synchronous Differential I/O Interfaces in Stratix GX Devices
Revised: *June 2006*
Part number: *SGX52013-1.2*

Chapter 18. DSP Blocks in Stratix & Stratix GX Devices
Revised: *July 2005*
Part number: *S52006-2.2*

Chapter 19. Implementing High Performance DSP Functions in Stratix & Stratix GX Devices
Revised: *September 2004*
Part number: *S52007-1.1*

About This Handbook

This handbook provides comprehensive information about the Altera® Stratix® GX family of devices.

How to Contact Altera

For the most up-to-date information about Altera products, go to the Altera world-wide web site at www.altera.com. For technical support on this product, go to www.altera.com/mysupport. For additional information about Altera products, consult the sources shown below.

Information Type	USA & Canada	All Other Locations
Technical support	www.altera.com/mysupport/	www.altera.com/mysupport/
	(800) 800-EPLD (3753) (7:00 a.m. to 5:00 p.m. Pacific Time)	+1 408-544-8767 7:00 a.m. to 5:00 p.m. (GMT -8:00) Pacific Time
Product literature	www.altera.com	www.altera.com
Altera literature services	literature@altera.com	literature@altera.com
Non-technical customer service	(800) 767-3753	+ 1 408-544-7000 7:00 a.m. to 5:00 p.m. (GMT -8:00) Pacific Time
FTP site	ftp.altera.com	ftp.altera.com

Typographic Conventions

This document uses the typographic conventions shown below.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: **Save As** dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: $\mathbf{f_{MAX}}$, **\qdesigns** directory, **d:** drive, **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Document titles are shown in italic type with initial capital letters. Example: *AN 75: High-Speed Board Design.*

Visual Cue	Meaning
Italic type	Internal timing parameters and variables are shown in italic type. Examples: t_{PIA}, $n + 1$. Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: *<file name>*, *<project name>*.**pof** file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
"Subheading Title"	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: "Typographic Conventions."
`Courier type`	Signal and port names are shown in lowercase Courier type. Examples: `data1`, `tdi`, `input.` Active-low signals are denoted by suffix `n`, e.g., `resetn`. Anything that must be typed exactly as it appears is shown in Courier type. For example: `c:\qdesigns\tutorial\chiptrip.gdf`. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword `SUBDESIGN`), as well as logic function names (e.g., `TRI`) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ● •	Bullets are used in a list of items when the sequence of the items is not important.
✓	The checkmark indicates a procedure that consists of one step only.
☞	The hand points to information that requires special attention.
CAUTION	The caution indicates required information that needs special consideration and understanding and should be read prior to starting or continuing with the procedure or process.
	The warning indicates information that should be read prior to starting or continuing the procedure or processes
↵	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information on a particular topic.

Section I. Stratix GX Transceiver User Guide

This section provides information on the configuration modes for Stratix GX devices. It also includes information on testing, Stratix GX port and parameter information, and pin constraint information.

This section includes the following chapters:

- Chapter 1, Introduction
- Chapter 2, Stratix GX Analog Description
- Chapter 3, Custom Mode
- Chapter 4, SONET Mode
- Chapter 5, XAUI Mode
- Chapter 6, GIGE Mode
- Chapter 7, Loopback Modes
- Chapter 8, Stratix GX Built-In Self Test (BIST)
- Chapter 9, Reset Control & Power Down
- Chapter 10, Data & Control Codes
- Chapter 11, Ports & Parameters
- Chapter 12, REFCLKB Pin Constraints

Revision History

The table below shows the revision history for Chapters 1 through 12.

Chapter(s)	Date / Version	Changes Made	Comments
1	June 2006, v1.2	Minor text edit.	Changed Basic mode to Custom mode.
	August 2005, v1.1	Text edit to the *Byte Serializer/Deserializer* section.	

Chapter(s)	Date / Version	Changes Made	Comments
2	June 2006, v1.2	• Updated the "MegaWizard Plug-In Manager Analog Features" section, including updating the figures to match the Quartus II software GUI. • Added "Stratix GX Transceiver Merging" section.	
	August 2005, v1.1	Text edit: changed 8'h6a to 8'h56 and rx_analogreset to rxanalogreset.	
3	June 2006, v1.2	• Changed chapter name from Basic Mode to Custom Mode to reflect updated GUI. • Updated "Custom Mode Clocking" section. • Updated Figures 3–23 and 3–24. • Updated the "Custom Mode MegaWizard Plug-In Manager" section, including updating the figures to match the Quartus II software GUI. • Added "Stratix GX Transceiver Merging" section.	
	August 2005, v1.1	Text edit to the *Byte Deserializer* section. Changed `patterndetect [1]` to `rx_patterndetect [1]`. Updated Figures 3-11 through 3-13.	
4	June 2006, v1.2	• Updated Figures 4–15 and 4–16. • Updated the "SONET Mode MegaWizard Plug-In Manager" section, including updating the figures to match the Quartus II software GUI. • Added "Stratix GX Transceiver Merging" section.	
	August 2005, v1.1	Changed `patterndetect [1]` to `rx_patterndetect [1]`. Updated Figures 4-6, 4-7, 4-8, 4-10, 4-12, and 4-13.	

Chapter(s)	Date / Version	Changes Made	Comments
5	June 2006, v1.2	• Updated Figures 5–26 and 5–27. • Updated the "XAUI Mode MegaWizard Plug-In Manager" section, including updating the figures to match the Quartus II software GUI. • Added "Stratix GX Transceiver Merging" section.	
	August 2005, v1.1	Updated Table 5-1. Updated Figures 5-3, 5-4, 5-12, and 5-14. Added a paragraph to the *Disparity Error Detector* section. Updated Table 5-4.	
6	June 2006, v1.2	• Updated Figures 6–25 and 6–26. • Updated the "GIGE Mode MegaWizard Plug-In Manager" section, including updating the figures to match the Quartus II software GUI. • Updated "Design Description" section. • Added "Stratix GX Transceiver Merging" section.	
	August 2005, v1.1	Updated Figures 6-14 and 6-15.	
7	June 2006, v1.2	Updated "Serial Loopback" section.	
	August 2005, v1.1	Text edit to the *Serial Loopback* and *Reverse Serial Loopback* sections.	
8	August 2005, v1.1	Changed `rx_apllreset` to `rxanalogreset`.	
9	February 2005 v1.0	Initial Release.	
10	February 2005 v1.0	Initial Release.	
11	June 2006, v1.1	Updated Table 11–1. Added description for `tx_forcedisparity[]`.	
	February 2005 v1.0	Initial Release.	
12	February 2005 v1.0	Initial Release.	

1. Introduction

SGX52001-1.2

Introduction

Stratix® GX devices combine highly advanced 3.1875-gigabit-per-second (Gbps) four-channel gigabit transceiver blocks with one of the industry's most advanced FPGA architectures. Stratix GX devices are manufactured on a 1.5-V, 0.13-µm, all-layer copper CMOS process technology with 1.5-V PCML I/O standard support.

Historically, designers have used high-speed transceivers in strictly structured, line-side applications. Now, with the new gigabit transceiver blocks embedded in FPGAs, you can use transceivers in a host of new systems that require flexibility, increased time-to-market, high performance, and top-of-the-line features.

Gigabit Transceiver Block Highlights

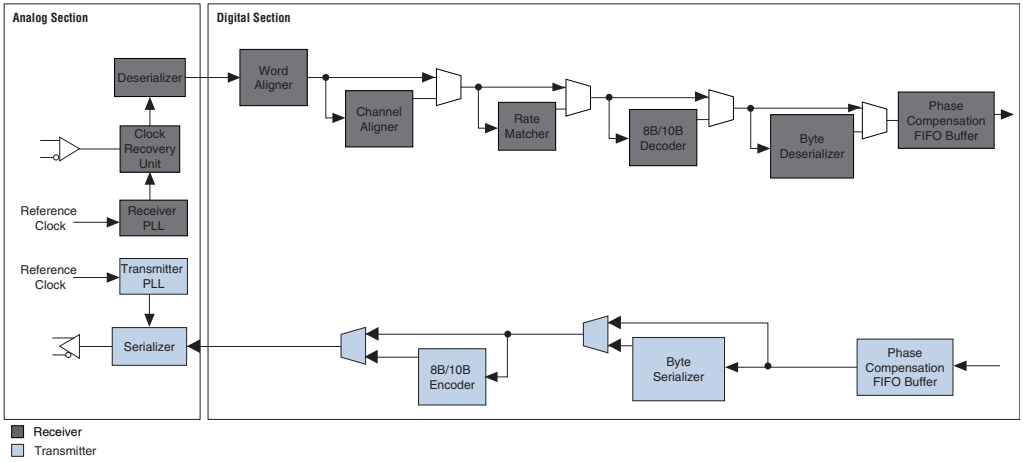
Stratix GX devices are organized into four-channel blocks with four 3.1875 Gbps full-duplex channels per block and up to 20 channels (in five blocks) per device. Each self-contained Stratix GX gigabit transceiver block supports a variety of embedded functions and does the following:

- Supports frequencies from 500 megabits per second (Mbps) to 3.1875 Gbps
- Integrates serializer/deserializer (SERDES), clock data recovery (CDR), word aligner, channel aligner, rate matcher, 8B/10B encoder/decoder, byte serializer/deserializer, and phase compensation first-in first-out (FIFO) modules
- Supports flexible reference clock generation capabilities, including a dedicated transmitter phase-locked loop (PLL) and four receiver PLLs per gigabit transceiver block
- Supports programmable pre-emphasis, equalization, and programmable V_{OD} settings in I/O buffers, and dynamic reprogrammability for each of these features
- Implements XAUI physical media attachment (PMA) and physical coding sublayer (PCS) functionality for 10GBASE-X systems
- Provides built-in Gigabit Ethernet (GigE) physical coding sublayer functionality
- Provides individual transmitter and receiver power-down capability for reduced power consumption during non-operation
- Includes built-in self test (BIST) capability, including embedded Pseudo Random Binary Sequence (PRBS) pattern generation and verification
- Includes three independent loopback paths for system verification

Transceiver Block Architecture

Figure 1–1 shows a block diagram of the gigabit transceiver block (GXB). You can bypass various modules if desired. Refer to"Modes of Operation" on page 1–5 for a description of the supported features in each mode. You can divide the transceiver block into an analog section and a digital section, as shown in Figure 1–1.

Figure 1–1. Block Diagram of a Stratix GX Gigabit Transceiver Block



Analog Section Overview

This section describes the various components within the analog section of the transceiver block.

Transmitter Differential I/O Buffers

The gigabit transmitter block differential I/O buffers support the 1.5-V PCML I/O standard, and contain features that improve system signal integrity. These features include programmable pre-emphasis, which helps compensate for high frequency losses, and a variety of programmable V_{OD} settings that support noise margin tuning.

Receiver Differential I/O Buffers

The gigabit transceiver block differential I/O buffers support the 1.5-V PCML I/O standard, and contain a variety of features that improve system signal integrity. Programmable equalization capabilities are used to compensate for signal degradation across transmission mediums.

Transmitter & Receiver PLLs

Each gigabit transceiver block contains one dedicated transmitter PLL and four dedicated receiver PLLs. These PLLs provide clocking flexibility and support a range of incoming data streams. For data transmission and recovery, these PLLs generate the required clock frequencies based upon the synthesis of an input reference clock. Each transmitter PLL supports multiplication factors of 2, 4, 5, 8, 10, 16, or 20. Either external reference clocks or a variety of clock sources within the Stratix GX device drive the PLLs.

Clock Recovery Unit

The gigabit transceiver block clock recovery unit (CRU) performs analog Clock Data Recovery (CDR). The CRU uses an external reference clock to extract a recovered clock that is frequency and phase aligned with the incoming data, thereby eliminating any clock-to-data skew. This recovered clock then clocks the data through the rest of the gigabit transceiver block.

Serializer Deserializer (SERDES)

The transmitter serializer converts the incoming lower speed parallel signal to a high-speed serial signal on the transmit side. The SERDES supports a variety of conversion factors, ensuring implementation flexibility. For example, the SERDES supports 10- and 20-bit serialization factors, typically required for 8B/10B encoded data, as well as 8- and 16-bit factors.

The receiver deserializer converts the incoming data stream from a high-speed serial signal to a lower-speed parallel signal that can be processed in the FPGA logic array on the receive side. The SERDES supports a variety of conversion factors, ensuring implementation flexibility. For example, the SERDES supports both 10-bit and 8-bit serialization and deserialization factors.

Digital Overview

This section describes the various components in the digital section of the transceiver block.

Transmitter & Receiver Phase Compensation FIFO Buffer

The transmitter and receiver data path has a dedicated phase compensation FIFO buffer that decouples phase variations between the FPGA and transceiver clock domains. These FIFO buffers ensure a consistent, reliable interface to the logic array and simplify system design and timing analysis.

Byte Serializer/Deserializer

The byte serializer converts a 16- or 20-bit data bus into two 8- or 10-bit data buses, respectively, at double the data rate.

The byte deserializer converts an 8- or 10-bit data bus into 16- or 20-bit data buses, allowing maximum throughput of the transceiver without burdening the FPGA logic array.

8B/10B Encoder/Decoder

8B/10B encoding/decoding is the backbone of many transceiver protocols, and it is often used in proprietary implementations. The gigabit transceiver block has dedicated circuitry to perform 8B/10B encoding in the transmitter and decoding in the receiver. This coding technique ensures sufficient data transitions and a DC balanced stream in the data signal for successful data recovery at the receiver.

Word Aligner

The word aligner module contains a fully programmable pattern detector to identify specific patterns within the incoming data stream. The pattern detector includes recognition support /K28.5/ comma characters for 8B/10B encoded data and A1 or A2 frame alignment patterns for scrambled signals. Additionally, you can specify a custom alignment pattern in lieu of the /K28.5/ comma.

The word aligner in the gigabit transceiver block also creates words from the incoming serial data stream by realigning the data based on identified byte boundaries. The realignment function uses a barrel shifter and works with the pattern detector. Additionally, the word aligner has a manual data realignment mode that lets you control the data realignment in user mode without consistent alignment characters.

Channel Aligner

An embedded channel aligner aligns byte boundaries across multiple channels and synchronizes the data entering the logic array from the gigabit transceiver block's four channels. The Stratix GX channel aligner is optimized for a 10-Gigabit Ethernet XAUI 4-channel implementation. The channel aligner includes the control circuitry and channel alignment character detection defined by the XAUI protocol. The channel aligner is only available in XAUI mode.

Rate Matcher

In multi-crystal environments, the clock frequencies of the transmitting and receiving devices do not match. This mismatch can cause the data to transmit at a rate slightly faster or slower than the receiving device can interpret. The Stratix GX rate matcher resolves the frequency differences between the recovered clock and the FPGA logic array clock by inserting or deleting removable characters from the data stream, as defined by the transmission protocol, without compromising transmitted data. If the functional mode is XAUI, the rate matcher is based on the 10-Gigabit Ethernet protocol. If the functional mode is GIGE, the rate matcher is based on the Gigabit Ethernet protocol.

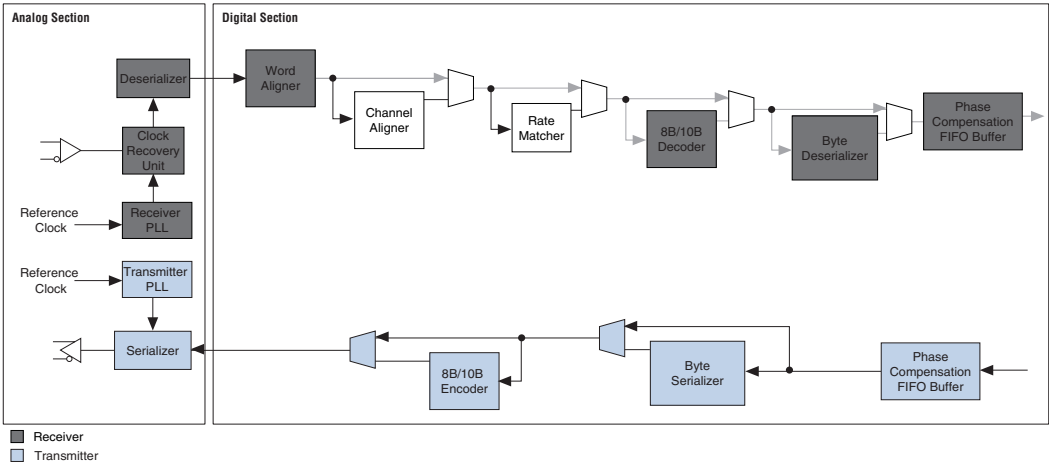
Modes of Operation

You can bypass various modules of the gigabit transceiver block based on the configured mode of operation. Stratix GX transceivers currently support Custom mode, SONET mode, and XAUI mode. This section provides an overview of each supported mode of operation.

Custom Mode

Custom mode enables a subset of the transceiver blocks so you can perform customizable configuration. Channel aligner and the rate matcher features are not available in this mode. Refer to the *Custom Mode* chapter for more details on the configurability of this mode. Figure 1–2 shows a block diagram of a duplex channel configured in Custom mode.

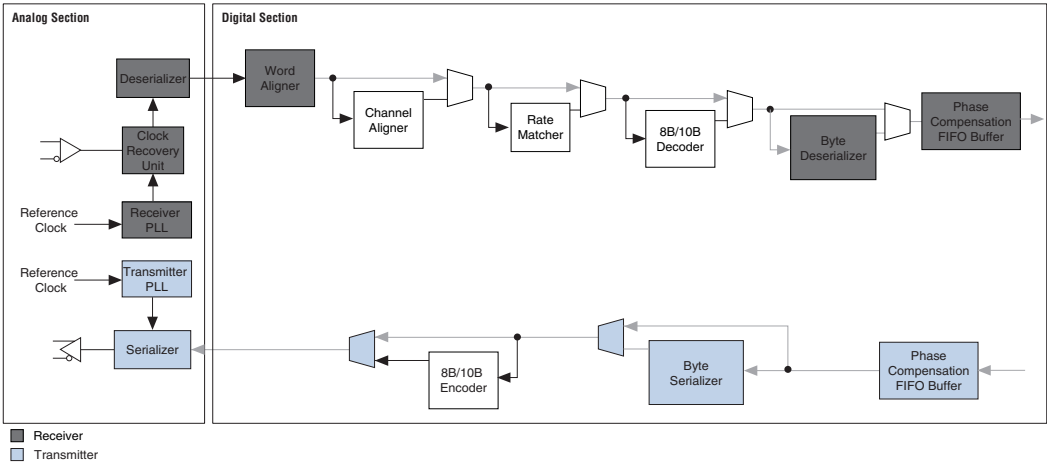
Figure 1–2. Block Diagram of a Duplex Channel Configured in Custom Mode



SONET Mode

SONET mode lets you to select a subset of the transceiver blocks to perform SONET-like configuration. SONET-like implies that the data width can either be 8 or 16 bits and that the 8B/10B encoder/decoder, channel aligner, and the rate matcher features are not available. Refer to the *SONET Mode* chapter for more details on the configurability of this mode. Figure 1–3 shows a block diagram of a duplex channel configured in SONET mode.

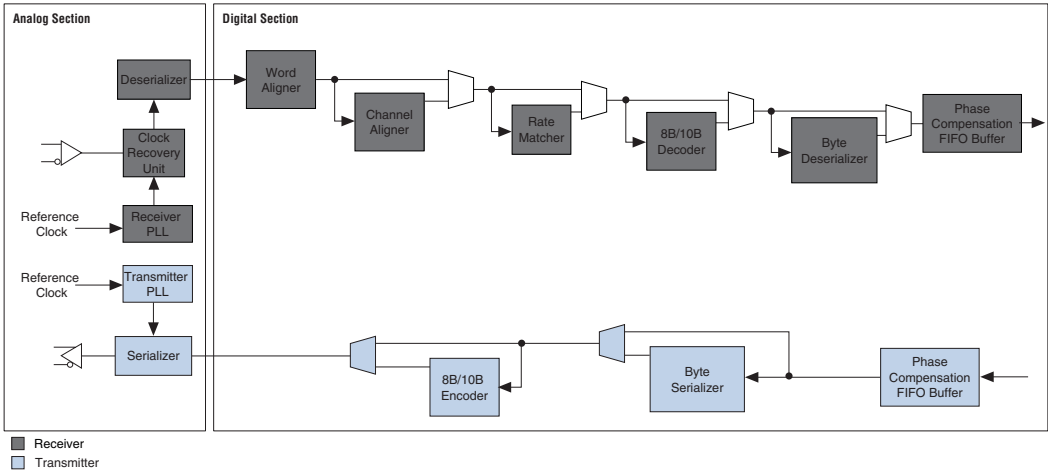
Figure 1–3. Block Diagram of a Duplex Channel Configured in SONET Mode



XAUI Mode

Stratix GX transceivers contain embedded macros dedicated to supporting the XAUI protocol, specified in clause 47 of the IEEE 802.3ae specification. These macros includes synchronization, channel deskew, rate matching, XGXS to XGMII, and XGMII to XGXS code-group conversion. Refer to the *XAUI Mode* chapter for more details on the configurability of this mode. Figure 1–4 shows a block diagram of a duplex channel configured in XAUI mode.

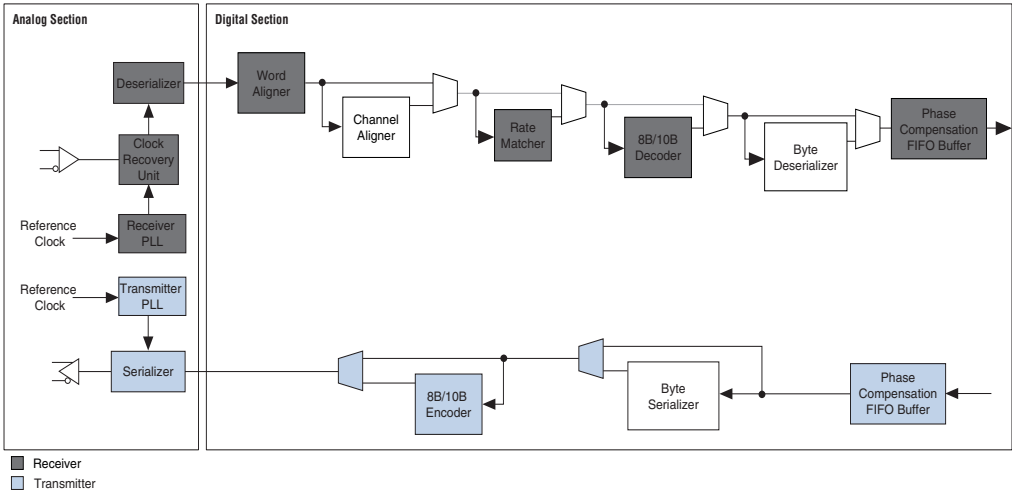
Figure 1–4. Block Diagram of a Duplex Channel Configured in XAUI Mode



GIGE Mode

Stratix GX devices in GIGE mode can use the 8B/10B encoder/decoder, rate matcher, synchronizer, and byte serializer/deserializer built-in hard macros. Refer to the *GIGE Mode* chapter of the *Stratix GX Device Handbook, Volume 2* for more information about this mode. The rate matcher and word aligner each have a dedicated state machine governing their functions. These state machines are active only in GIGE mode. Figure 1–5 shows a block diagram of a duplex channel configured in GIGE mode.

Figure 1–5. Block Diagram of a Duplex Channel Configured in GIGE Mode



Loopback

There are three different loopback modes to use in the gigabit transceiver block to allow for a complete method of in-system verification. The loopback modes are versatile and robust enough to accommodate all protocols and let you to choose whether to retime the data.

Built-In Self Test

The gigabit transceiver block contains several features that simplify design verification. An embedded PRBS pattern generator provides a bitstream pattern that you can use to test the device and board connections. The PRBS pattern generator works with a PRBS receiver to implement a full self-test path. Additionally, serial and parallel loopback paths let you test the FPGA logic without monitoring external signals. The reverse loopback path enables external system testing with minimal device interaction.

2. Stratix GX Analog Description

SGX52002-1.2

Introduction

This chapter describes how to serialize the parallel data for transmission and convert received data into parallel data. Data transmission and reception is performed by pseudo current mode logic (PCML) buffers. These transceiver buffers support programmable pre-emphasis, equalization, and programmable V_{OD} settings in I/O buffers.

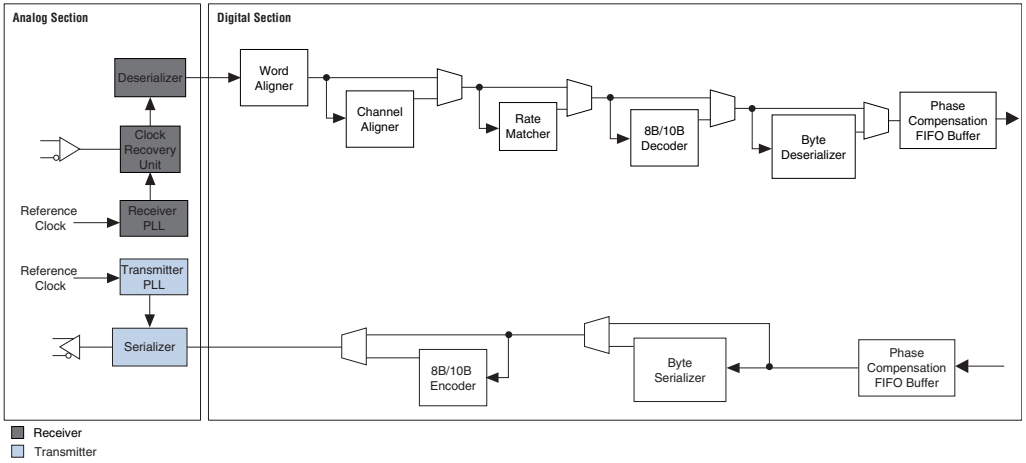
The programmable pre-emphasis setting is available on transmit buffers to maximize the eye opening on the far-end receiver by boosting the high-frequency component of the data signal. Similarly, programmable equalization is available for receive buffers to reduce the high-frequency losses and inter-symbol interference. These features are useful in lossy transmission lines. Transceivers also support flexible reference clock generation capabilities, including a dedicated transmitter phase-locked loop (PLL) and four receiver PLLs per transceiver block.

The clock recovery unit (CRU) is the main part of each receive analog section; it recovers the clock from the serial data stream (refer to Figure 2–1).

You can set the CRU to automatically or manually alter the receiver PLL phase and frequency to match the bit transition on the incoming data stream. This is to eliminate any clock-to-data skew or to keep the receiver PLL locked to the reference clock (lock-to-data or lock-to-reference mode).

During the clock recovery phase, the receiver PLL initially locks to the reference clock and then attempts to lock on to the incoming data by first recovering the clock from the incoming serial data.

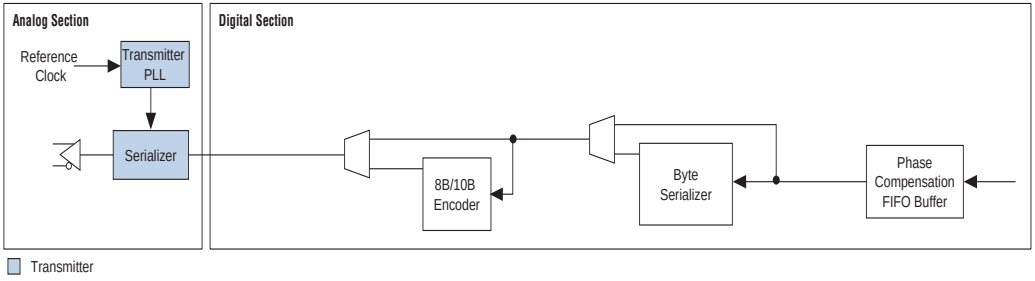
Figure 2–1. Block Diagram Analog Components



Transmitter Analog

This section describes the transmitter buffer, the transmitter PLL, and the serializer. Figure 2–2 shows the transmitter analog components.

Figure 2–2. Transmitter Analog Components



Transmitter Buffer

The Stratix® GX transceiver buffers support the 1.5-V PCML standard at speeds up to 3.1875 gigabits per second (Gbps) and are capable of driving 40 inches of FR4 trace across two connectors. In addition, the buffer contains programmable output voltage, programmable pre-emphasis circuitry, and internal termination circuitry.

Programmable Voltage Output Differential (V_{OD})

Stratix GX transceivers let you customize the differential output voltage (V_{OD}) to handle different length, backplane, and receiver requirements (refer to Figure 2–3). You can select the V_{OD} (differential) from a range of 400 to 1,600 mV, as shown in Table 2–1.

Figure 2–3. V_{OD} (Differential) Signal Level

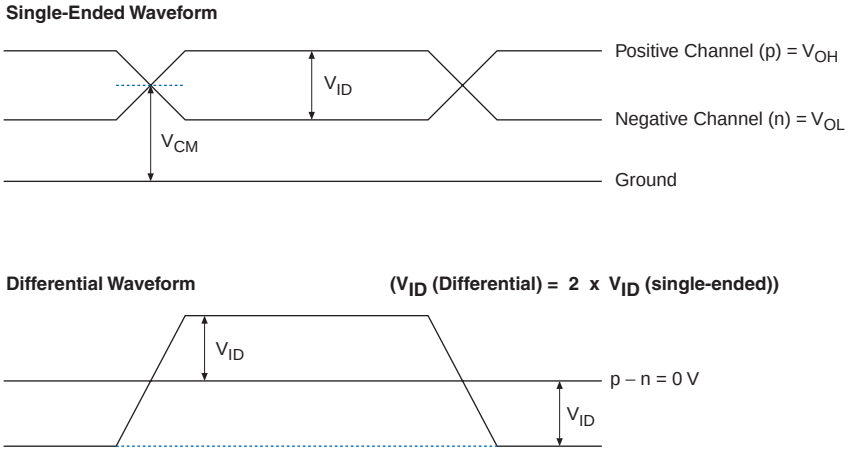


Table 2–1 shows the differential output voltage (V_{OD}) setting per current level for each of the on-chip transmitter programmable termination values.

Table 2–1. Programmable V_{OD} (Differential)

100 Ω (mV)	120 Ω (mV)	150 Ω (mV)
400	480	600
800	960	1,200
1,000	1,200	1,500
1,200	1,440	
1,400		
1,600		

You can set the differential V_{OD} values statically during configuration or dynamically adjust them in user mode. You select the static V_{OD} value through a list in the `altgxb` MegaWizard® Plug-In Manager, which sets the appropriate V_{OD} setting in the configuration file. The disadvantage of the static mode setting is that the V_{OD} is set on a per transceiver block basis and cannot be changed unless you regenerate another programming file.

Alternatively, if you enable dynamic adjustment in the `altgxb` MegaWizard Plug-In Manager, you can dynamically configure the V_{OD} setting by the device during user mode. This configuration is done by asserting encoded values on the `tx_vodctrl` bus, which is instantiated in the `altgxb` module when you select the dynamic adjustment option. This option lets you make quick performance evaluations of the various settings without having to recompile and regenerate multiple configuration files. Another advantage of this option is that it allows the V_{OD} of each channel to be configured independently. Refer to the section "MegaWizard Plug-In Manager Analog Features" on page 2–20 for further details.

Programmable Pre-Emphasis

The programmable pre-emphasis module in each transmit buffer boosts the high frequencies in the transmit data signal, which may be attenuated in the transmission media. This maximizes the data eye opening at the far-end receiver. Pre-emphasis is particularly useful in lossy transmission mediums.

The transfer function of a transmission line can be represented in the frequency domain as a low-pass filter. Any frequency components below the –3 dB frequency pass through with minimal losses. Frequency components that are greater than the –3-dB frequency are attenuated. This variation in frequency response yields data-dependant jitter and other ISI effects. By applying pre-emphasis, the high frequency components are boosted, or in other words, pre-emphasized. This pre-emphasis equalizes the frequency response as seen at the receiver so that the delta between the low-frequency and high-frequency components is reduced, which in return minimizes the ISI effects from the transmission medium.

In Stratix GX transceivers, the programmable pre-emphasis settings can have one of six values (0 to 5). You should experiment with the pre-emphasis values to determine the optimal setting based on your system variables.

As with the V_{OD} settings, you can set the pre-emphasis settings statically during configuration or adjust them dynamically in user mode. You can set the static pre-emphasis value through a drop-down menu in the `altgxb` MegaWizard Plug-In Manager, which sets the appropriate pre-emphasis setting in the configuration file. The disadvantage of the static mode setting is that the pre-emphasis is set on a per-transceiver-block basis and cannot be changed without regenerating another programming file.

On the other hand, if you select dynamic adjustment in the `altgxb` MegaWizard Plug-In Manager, the pre-emphasis setting can be configured dynamically by the device during user mode. This configuration is done by asserting encoded values on the `tx_preemphasisctrl` bus, which is instantiated in the `altgxb` module when you select the dynamic adjustment option. This option lets you make quick performance evaluations of the various settings without having to recompile and regenerate multiple configuration files. Another advantage of this option is that it allows the pre-emphasis of each channel to be configured independently. For further details, refer to "MegaWizard Plug-In Manager Analog Features" on page 2–20.

Avoid pre-emphasis and V_{OD} settings that yield a value greater than 1,600 mV. Settings beyond this value do not damage the buffer, but they prevent accurate device operation. Verify that the combination of V_{OD} and pre-emphasis settings do not exceed the 1,600-mV limit.

Programmable Transmitter Termination

The Stratix GX transmitter buffer includes a 100-, 120-, or 150-Ω programmable on-chip differential termination resistor. The Stratix GX transmitter buffers are current-mode drivers, so the resultant V_{OD} is a function of the transmitter termination value. For more information on resultant V_{OD} values, refer to "Programmable Voltage Output Differential (V_{OD})" on page 2–3.

Transmitter PLL

Each transceiver block contains a transmitter PLL and a slow-speed reference clock. The transmitter PLL receives the reference clock and generates the high-speed serial clock used by the serializer. The slow-speed reference clock is used for the transceiver logic. Figure 2–4 shows the transmitter PLL's block diagram. The `pll_locked` signal indicates when the transmitter PLL is locked to the reference clock. A high signal indicates that the PLL is locked to the reference clock; a low signal indicates that the PLL is not locked to the reference clock.

Figure 2–4. Transmitter PLL Block Diagram

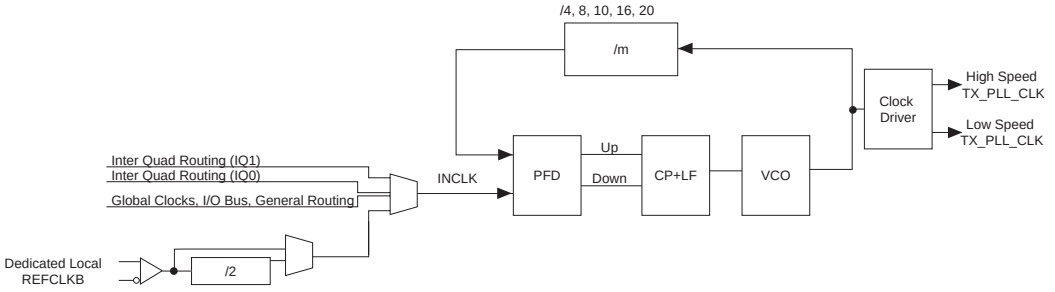


Table 2–2 lists some of the transmitter PLL specifications.

Table 2–2. Transmitter PLL Specifications

Parameter	Specifications
Input reference frequency range	25 MHz to 650 MHz
Data rate support	500 Mbps to 3.1875 Gbps
Multiplication factor (W)	2, 4, 5, 8, 10, 16, or 20 *(1)*

Note to Table 2–2:

(1) Multiplication factors 2 and 5 can only be achieved with the use of the pre-divider on the REFCLKB.

Clock Synthesis

The maximum input frequency of the phase frequency detector (PFD) is 325 MHz. To achieve reference clock frequency above this limitation, the /2 pre-divider on the dedicated local REFCLKB path can be enabled automatically by the Quartus® II software. The /2 pre-divider divides the reference clock frequency by a factor of 2 and then the /m factor compensates the frequency difference. An example would be a data rate of 2,488 Mbps with a 622-MHz reference clock. In this scenario, the reference clock must be assigned to the REFCLKB port where the 622-MHz reference clock is divided by 2, yielding a 311-MHz clock at the PFD. This 311-MHz reference clock is then multiplied by a factor of 8 to achieve the 2,488-MHz clock at the VCO.

If the reference clock exceeds 325 MHz, the clock must be fed by the dedicated local reference clock pin, REFCLKB. By default, the Quartus II software assigns pins to be LVTTL, so you must assign the 1.5-V PCML I/O standard to the I/O pins to select the REFCLKB port as the reference source. The Quartus II software prompts a fitter error if the reference clock exceeds 325 MHz and the reference clock source is not on the REFCLKB port.

You can also use the pre-divider on the REFCLKB path to support additional multiplication factors. The block diagram in Figure 2–4 shows that /m can only support multiplication factors of 4, 8, 10, 16, and 20, but Table 2–3 shows that the additional multiplication factors of 2 and 5 are also achievable. You can achieve these multiplication factors by using the pre-divider. A multiplication factor of 2 is achieved by pre-dividing the reference clock by 2 and then multiplying the resultant frequency by 4, which yields a multiplication factor of 2. A multiplication factor of 5 is achieved in the same manner by pre-dividing the reference clock by 2 and then multiplying the resultant frequency by 10, which yields a multiplication factor of 5.

Table 2–3 lists the possible multiplication values as a function of the source to the transmitter PLL. Table 2–3 assumes that the reference clock is directly fed from the source listed and does not factor any pre-clock synthesis (that is, the Stratix GX PLL driving a global clock that is used for the transmitter PLL reference clock source).

Table 2–3. Multiplication Values as a Function of the Reference Clock Source to the Transmitter PLL

Transmitter PLL Reference Clock Source	Multiplication Factors
Global clock, I/O bus, general routing	4, 8, 10, 16, 20
Inter-transceiver routing	2, 4, 5, 8, 10, 16, 20
Dedicated local REFCLKB	2, 4, 5, 8, 10, 16, 20

You must specify the data rate of the channel and input clock period of the reference clock. The data rate divided by the input clock period must equal one of the multiplication factors listed in Table 2–3.

Transmitter PLL Bandwidth Setting

The Stratix GX transmitter PLL in the transceiver block offers a programmable bandwidth setting. The PLL bandwidth is the measure of its ability to track the input clock and jitter. The bandwidth is determined by the –3-dB frequency of the closed-loop gain of the PLL.

A high-bandwidth setting provides a faster lock time and tracks more jitter on the input clock source which passes it through the PLL. This helps reject noise from the VCO and power supplies. A low-bandwidth setting, on the other hand, filters out more high frequency input clock jitter, but increases lock time.

You can set the bandwidth for Stratix GX devices to either low or high. The –3-dB frequencies for these settings can vary due to the non-linear nature and frequency dependencies of the circuit. As a result, you can vary the bandwidth to customize the performance on specific systems.

Serializer (Parallel-to-Serial Converter)

The serializer converts parallel data to serial data at the transmitter output buffer. The serializer can support 8- or 10-bit words when used with the transmitter multiplexer. The 8-bit serializer drives the serial data to the output buffer, as shown in Figure 2–5. The serializer can drive the serial bit-stream at a data rate range of 500 Mbps to 3.1875 Gbps. The serializer outputs the least significant bit (LSB) of the word first.

Figure 2–5. Example of 8-Bit Serialization

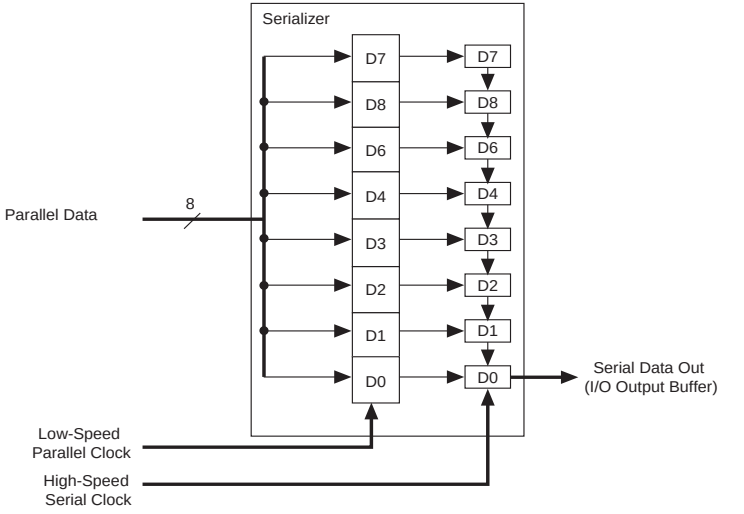
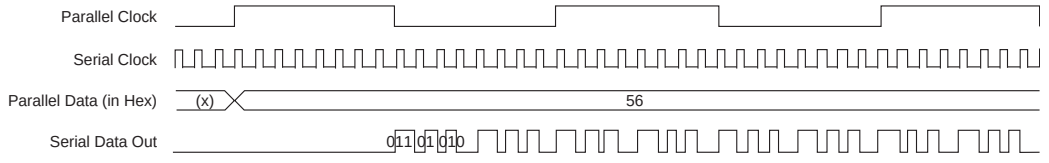


Figure 2–6 shows the serial bit order of the serializer output. In this example, a constant `8'h56` (`01010110`) value is serialized.

The serial data is transmitted from LSB to most significant bit (MSB).

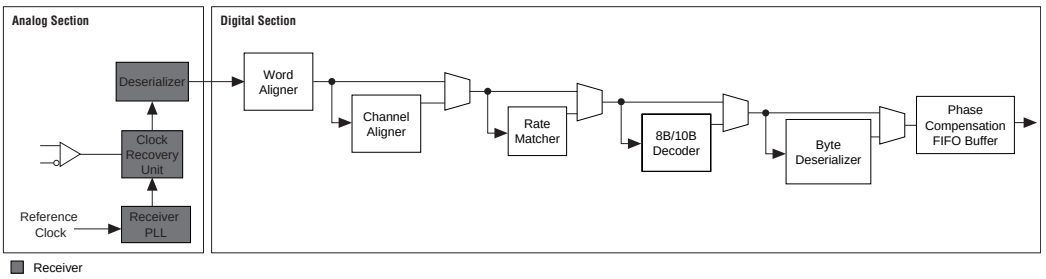
Figure 2–6. Serializer Bit Order



Receiver Analog

This section describes the receiver input buffer, the receiver PLL, the clock recovery unit, and the deserializer. Figure 2–7 shows the receiver analog components.

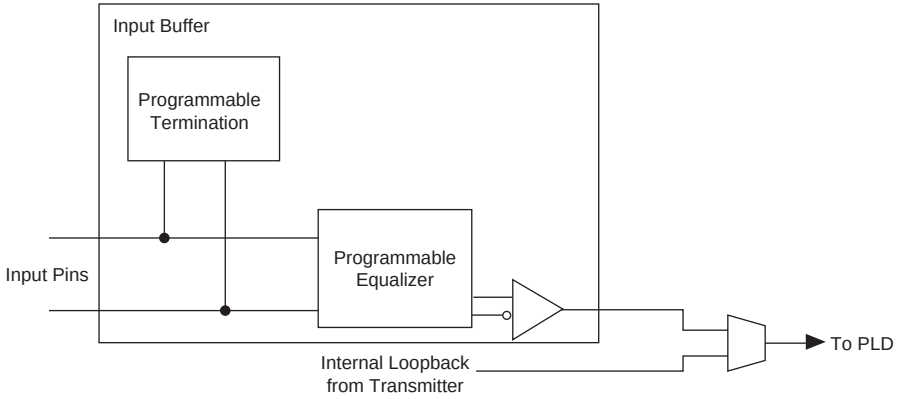
Figure 2–7. Highlighted Block Diagram of the Receiver Analog Components



Receiver Input Buffer

The receiver input buffer contains internal termination and internal equalization. Figure 2–8 shows the structure of the input buffer. The input buffer has programmable equalization that you can apply to increase the signal integrity of the transmission line. The internal termination in the receiver buffer can support AC and DC coupling with programmable differential termination settings of 100, 120, or 150 Ω

Figure 2–8. Receiver Input Buffer



Programmable Receiver Termination

The Stratix GX receiver buffer includes programmable on-chip differential termination of 100, 120, or 150 Ω.

This assignment must be made per pin through the Assignment Editor in the Quartus II software. Select **Assignment Organizer > Options for Individual Nodes Only > Stratix GX Termination Value** (Assignments menu).



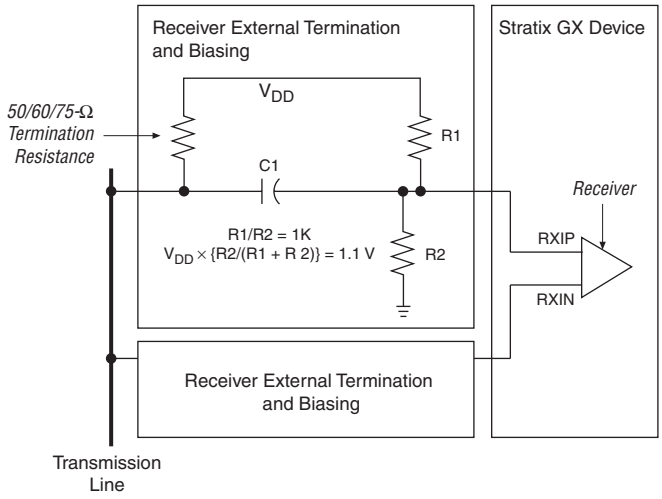
The proper termination settings should be selected and verified accordingly before compilation.

The transmitter PLL input signal (`inclk`) drives the termination resistance calibration circuit. The Quartus II software allows receiver-only configurations in Stratix GX devices. However, if you use the Quartus II software to remove the transmitter PLL in a receiver-only configuration, you will see an incorrect value or unpredictable behavior with the receiver input pin termination. If the `rx_cruclk` signal is globally routed, the Quartus II software handles this automatically. If the `rx_cruclk` signal is not globally routed or routed using the inter-quadrant line (IQ2), the Quartus II software returns a no-fit. In this situation, you must add a transmitter PLL to your design.

If the `pll_areset` (analog reset) signal goes high, the `RX_Vcm` value is less than the 1.1 V. This value varies unpredictably because the circuit is tristated. `RX_Vcm` is referenced from the Stratix GX receiver analog power supply.

If external termination is used, the receiver must be externally terminated and biased to 1.1 V. Figure 2–9 shows an example of an external termination and biasing circuit.

Figure 2–9. External Termination & Biasing Circuit



Enable Stratix GX-to-Stratix GX Receiver DC Coupling

You can configure the Stratix GX receiver buffers so that DC-coupled Stratix GX-to-Stratix GX communication is possible. The Stratix GX transmitter's common-mode is typically around 750 mV, while the receiver common mode by default is approximately 1.1 V. However, by enabling DC coupling, the receiver common mode is biased to allow interoperability with the Stratix GX transmitter.

Equalizer Mode

Stratix GX transceivers offer an equalization circuit in each receiver channel to increase noise margins and help reduce the effects of high frequency losses. The programmable equalizer compensates for inter-symbol interference (ISI) and high frequency losses that distort the signal and reduce the noise margin of the transmission medium by equalizing the frequency response.

The transfer function of a transmission line can be represented in the frequency domain as a low-pass filter. Any frequency components below the –3-dB frequency pass through with minimal losses. Frequency components that are greater than the –3-dB frequency are attenuated.

This variation in frequency response yields data-dependant jitter and other ISI effects. By applying equalization, the low frequency components are attenuated. This equalizes the frequency response so that the delta between the low frequency and high frequency components are reduced, which minimizes the ISI effects from the transmission medium.

In Stratix GX transceivers, the programmable equalizer settings can have one of five values (0 through four). You should experiment with the equalization values to determine the optimal setting based on your system variables.

As with the V_{OD} settings, you can set the equalization settings statically during configuration or adjust them dynamically in user mode. You can select the static equalization value through a drop-down menu in the `altgxb` MegaWizard Plug-In Manager. This action sets the appropriate equalization setting in the configuration file. The disadvantage of this mode is that the equalization is set on a per-transceiver block basis and cannot be changed without regenerating another programming file.

On the other hand, if you select the dynamic adjustment in the `altgxb` MegaWizard Plug-In Manager, the equalization setting can be configured dynamically by the device during user mode. This configuration is accomplished by asserting encoded values on the `rx_equalizerctrl` signal, which is instantiated in the `altgxb` module when this option is selected. This feature lets you make quick performance evaluations of the various settings without having to recompile and regenerate multiple configuration files. Another advantage is that this option allows the equalization of each channel to be configured independently. Refer to "MegaWizard Plug-In Manager Analog Features" on page 2–20 for more details.

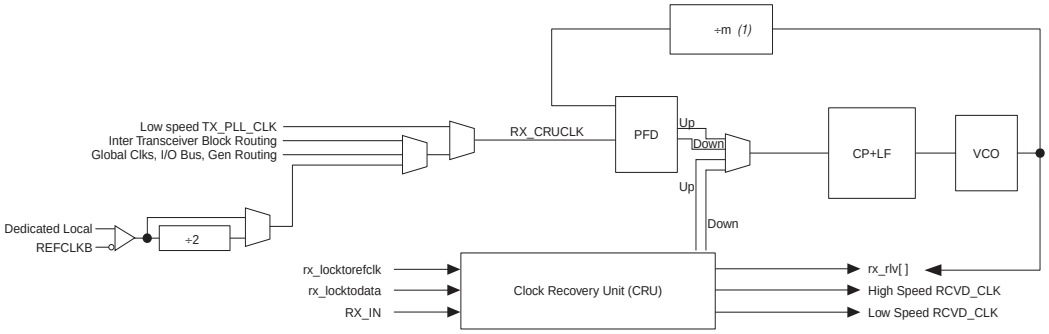
Receiver PLL

Each transceiver block contains four receiver PLLs and a slow-speed reference clock. The receiver PLLs receive the reference clock and generate the high-speed serial clock used by the CR. The slow-speed reference clock is used for the transceiver logic. Figure 2–10 shows the block diagram for the lock-to-reference portion of the receiver PLL.

This section focuses on the receiver PLL in Lock-to-Reference mode. The lock-to-data circuit has been omitted. Refer to "Lock-to-Reference Mode & Lock-to-Data Mode" on page 2–16 for more information on the operation between the two modes.

The receiver PLL contains an optional loss-of-lock indicator signal (rx_locked) that indicates when the receiver PLL is not locked to the reference clock. The rx_locked signal is active low. A low signal indicates that the PLL is locked to a reference clock; a high signal indicates that the PLL is not locked to the reference clock.

Figure 2–10. Receiver PLL Block Diagram



Note to Figure 2–10:
(1) *m* = 8, 10, 16, or 20.

Table 2–4 lists some of the clock recovery unit specifications.

Table 2–4. Clock Recovery Unit Specifications

Parameter	Specification
Input reference frequency range	25 MHz to 650 MHz
Data rate support	500 Mbps to 3.1875 Gbps
Multiplication factor (*W*)	2, 4, 5, 8, 10, 16, or 20 *(1)*

Note to Table 2–4:
(1) Multiplication factors 2, 4, and 5 can only be achieved with the use of the pre-divider on the REFCLKB port or if the CRU is trained with the low speed clock from the transmitter PLL.

Clock Synthesis

The maximum input frequency of the PFD of the receiver PLL is 325 MHz. To achieve reference clock frequency above this limit, the Quartus II software enables the divide by 2 pre-divider on the dedicated local REFCLKB path. This divides the reference clock frequency by a factor of 2 and then the /m factor compensates the frequency difference. For example, given a data rate of 2,488 Mbps with a reference clock of 622 MHz, the reference clock must be assigned to the REFCLKB port,

where the reference clock signal is divided by 2, yielding a 311 MHz clock at the PFD. This 311-MHz reference clock is then multiplied by a factor of 8 to achieve the 2,488-MHz clock at the VCO.

If the reference clock (RX_CRUCLK) exceeds 325 MHz, the clock must be fed by the dedicated local reference clock pin, REFCLKB. By default, the Quartus II software assigns pins to be LVTTL, so a 1.5-V PCML I/O standard assignment is required to select the REFCLKB port as the reference source. The Quartus II software prompts a fitter error if the reference clock exceeds 325 MHz and the reference clock source is not on the REFCLKB port.

The pre-divider on the REFCLKB path is also used to support additional multiplication factors. The block diagram in Figure 2–10 on page 2–13 shows that /m supports only multiplication factors of 8, 10, 16, and 20, but Table 2–4 states that the additional multiplication factors of 2, 4, and 5 can also be achieved.

Without using the transmitter PLL, the pre-divider achieves the multiplication factors of 4 and 5. A multiplication factor of 4 is achieved by pre-dividing the reference clock by 2 and then multiplying the resulting frequency by 8, which yields a multiplication factor of 4. A multiplication factor of 5 is achieved in the same manner by pre-dividing the reference clock by 2 and then multiplying the resulting frequency by 10, which yields a multiplication factor of 5.

The MegaWizard Plug-In Manager altgxb option enables the transmitter PLL in receiver mode. There is also an option to train the receiver CRU with the output of the low-speed transmitter PLL clock. If you select this option, all the multiplication factors that are supported in the transmitter PLL are also supported in the receiver CRU PLL, including the multiplication factor of 2. This option selects the low-speed transmitter PLL clock as the reference source. The low speed transmitter PLL clock is either divided by a SERDES factor of 8 or 10. The receiver PLL then multiplies this reference clock by a factor of 8 or 10 to achieve the same multiplication factor as the transmitter PLL.

For example, a multiplication factor of 2 is achieved on the transmitter PLL by pre-dividing the reference clock by 2 and then multiplying the resultant frequency by 4, which yields a multiplication factor of 2. However, on the low-speed clock output, this frequency is divided by a factor of 8 or 10, depending on the deserialization factor. The low-speed clock feeds the reference of the receiver PLL where the clock is multiplied back up by a factor of 8 or 10, which results in total multiplication factor of 2.

Table 2–5 lists the possible multiplication values as a function of the reference clock source to the receiver PLL. Table 2–5 assumes that the reference clock (RX_CRUCLK) is directly fed from the source listed and does not factor any pre-clock synthesis (that is, a Stratix GX PLL driving a global clock used for the receiver PLL reference clock source).

Table 2–5. Multiplication Values as a Function of the Reference Clock Source to the Receiver PLL

Receiver PLL Reference Clock Source	Multiplication Factors
Global clock, IO bus, general routing	8, 10, 16, 20
Inter-transceiver routing	4, 5, 8, 10, 16, 20
Dedicated local REFCLKB	4, 5, 8, 10, 16, 20
Low-speed transmitter PLL clock (train CRU with transmitter PLL option)	2, 4, 5, 8, 10, 16, 20

You specify the data rate of the channel and receiver CRU clock period of the receiver reference clock. The data rate divided by the input clock period must equal one of the multiplication factors listed in Table 2–5.

PPM Frequency Threshold Detector

The PPM frequency threshold detector senses whether the incoming reference clock to the CRU and the PLL VCO of the CRU are within a prescribed PPM tolerance range. Valid parameters are 125, 250, 500, or 1,000 PPM. The default parameter, if no assignments are made, is 1,000 PPM. The output of the PPM frequency threshold detector is one of the variables that asserts the rx_freqlocked signal. Refer to "Clock Recovery Unit" on page 2–16 for more detail regarding the rx_freqlocked signal.

Receiver Bandwidth Type

The Stratix GX receiver PLL in the CRU offers a programmable bandwidth setting. The bandwidth of a data recovery PLL is the measure of its ability to track the input data and jitter. The bandwidth is determined by the –3-dB frequency of the closed-loop gain of the PLL.

A higher bandwidth setting provides a faster lock time and tracks greater jitter on the input data source, rx_in[], which passes it through the PLL. This helps reject noise from the VCO and power supplies. A low-bandwidth setting, on the other hand, filters out more high-frequency data input jitter, but increases lock time.

Valid receiver bandwidth settings are low, medium, and high. The –3-dB frequencies for these settings vary due to the non-linear nature and data dependencies of the circuit. You vary the bandwidth to customize the performance on specific systems.

Clock Recovery Unit

The CRU in each Stratix GX receiver channel recovers the clock from the serial data stream on RX_IN. You can set the CRU to automatically or manually alter the receiver PLL phase and frequency to match the bit transition on the incoming data stream. This is to eliminate any clock-to-data skew or to keep the receiver PLL locked to the reference clock (lock-to-data or lock-to-reference mode). The CRU generates two clocks, a high-speed RCVD_CLK to feed the deserializer and a low-speed RCVD_CLK to feed transceiver logic. You can set the CRU to optionally detect run-length violations in the incoming data stream and generate an error whenever the preset run length is exceeded (run-length violation detection circuit).

Lock-to-Reference Mode & Lock-to-Data Mode

The Stratix GX device offers both automatic and manual locking options, as described in the following sections.

Automatic Lock Mode

By default, the CRU initially locks to the CRU reference clock RX_CRUCLK (lock-to-reference mode) until conditions warrant the switchover to the incoming data (lock-to-data mode). The device switches to the lock-to-data mode when the rx_freqlocked signal goes high. After switching to lock-to-data mode, the CRU requires more time to lock to the incoming serial data.



For information about the CRU to serial data lock time, which includes frequency lock (during lock-to-reference mode) and phase lock (during lock-to-data mode), refer to the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1*. Also refer to the *Reset Control & Power Down* chapter for the recommendations on resets.

To automatically transition from the lock-to-reference mode to the lock-to-data mode, the following conditions must be met:

- The CRU PLL is within the prescribed PPM frequency threshold setting (125, 250, 500, or 1,000 PPM) of the CRU reference clock.
- Reference clock and CRU PLL output are phase matched (phases are within 0.08 UI).

During the lock-to-reference mode, the frequency detector determines whether the reference clock to the receiver PLL and the VCO output are within the prescribed PPM setting.

The phase lock happens when the phase-frequency detector up/down transitions are relatively few and, the pulse widths are sufficiently narrow. These conditions show that the PLL is close to absolute phase lock to the reference clock. This ensures that when actual data signals are sampled, the receiver PLL locks to the fundamental REFCLK frequency and does not drift off to any sub-harmonic.

In lock-to-data mode, the PLL uses a phase detector to keep the recovered clock aligned properly with the data. If the PLL does not stay locked to data because of problems such as frequency drift or severe amplitude attenuation, the receiver PLL locks back to the reference clock of the CRU to train the VCO. When the device is in lock-to-data mode, the CRU tries to align itself with incoming data and there is no phase relationship with the reference clock.

In lock-to-data mode, the rx_freqlocked signal is asserted, and the rx_locked signal looses its significance. The rx_locked signal signifies that the CRU has locked to the reference clock. When the CRU is in lock-to-data mode, the rx_locked signal behavior is not predictable.

In automatic lock mode, CRU is forced out of lock-to-data mode if the CRU PLL is not within the recommended PPM frequency threshold setting (125 PPM, 250 PPM, 500 PPM, 1000 PPM) of the CRU reference clock.

When the CRU goes out of lock-to-data mode, the rx_freqlocked signal goes low. The rx_freqlocked signal also goes low when either the rxanalogreset or pll_areset signal goes high. The rxanalogreset signal powers down the receiver and the pll_areset signal powers down the entire transceiver block (four channels).

Manual Lock Options

Two optional input pins, rx_locktorefclk[] and rx_locktodata[], are available that let you control whether the CRU PLL automatically or manually switches between lock-to-reference clock and lock-to-data modes. This lets you bypass the default automatic switchover circuitry if either the rx_locktorefclk[] or rx_locktodata[] signal is instantiated.

When the rx_locktorefclk[] signal is asserted, it forces the CRU PLL to lock to the reference clock (RX_CRUCLK). Asserting the rx_locktodata[] signal forces the CRU PLL to lock to data, whether

or not the CRU is ready. When both signals are asserted, the rx_locktodata[] signal takes precedence over the rx_locktorefclk[] signal.

You might want to have control over both rx_locktorefclk[] and rx_locktodata[] signals to potentially reduce the CRU lock times. The PPM threshold frequency detector and phase relationship detector require additional latencies to ensure that the CRU is ready to lock to data. These extra latencies are potentially reduced by manually controlling the CRU train signals. You assert the rx_locktorefclk[] signal to initially train the CRU and, after some delta time, assert the rx_locktodata[] signal.

You configure the controller that controls the signals based on your system. You do this by experimenting because many variables must be considered, such as temperature, transition densities, and data rates. However, by doing so, you are not subjected to the CRU lock times required to verify if the two conditions to switch from lock-to-reference mode to lock-to-data mode in the defaulted automatic mode are met. When the rx_locktorefclk goes high, the rx_freqlocked signal is ignored and does not toggle. The rx_freqlocked signal always goes high if lock-to-data mode is asserted. If you want to transition from lock-to-data mode to automatic mode, the transition should be followed by rxanalogreset to send the rx_freqlocked signal low. The CRU does not often transition from manual mode to automatic mode during system operation.



The rxanalogreset signal functions like a power down signal as opposed to a digital reset. For more information on various reset signals, refer to the *Reset Control & Power Down* chapter of the *Stratix GX Device Handbook, Volume 2*.

Run Length Violation Detection Circuit

The programmable run length violation (RLV) circuit is in the CRU and detects consecutive ones or zeros in the data. If the data stream exceeds the preset maximum number of consecutive ones or zeros, the violation is signified by the assertion of the rx_rlv signal.

The rx_rlv signal is not synchronized to the parallel data, and as a result appears in the logic array earlier than the run-length violation data. To ensure that the FPGA latches this signal in systems where there are frequency variations between the recovered clock and the PLD logic array clock, the rx_rlv signal is asserted for more than two clock cycles in 8- or 10-bit data modes and three clock cycles in 16- or 20-bit data modes.

If the data width is 8 or 16, set the legal run length threshold values within the range of 4 to 128 UI in multiples of four. If the data width is 10 or 20, or if using 8b10b, set the legal run length threshold values within the range of 5 to 160 UI in multiples of five.



Refer to the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1* to verify the guaranteed maximum run length.

Deserializer (Serial-to-Parallel Converter)

The deserializer converts incoming high-speed serial data streams to either 8- or 10-bit-wide parallel data synchronized to the recovered clock of the CRU. The deserializer drives the parallel data to the pattern detector and word aligner, as shown in Figure 2–11. The data rate of the deserializer output bus is the input data rate divided by the width of the output data bus. For example, for a 10-bit bus and a serial input data rate of 2.5 Gbps, the parallel data rate is 2500/10 or 250 MHz. The first bit into the deserializer is the LSB of the data bus out of the deserializer.

Figure 2–11. Deserializer Block Diagram

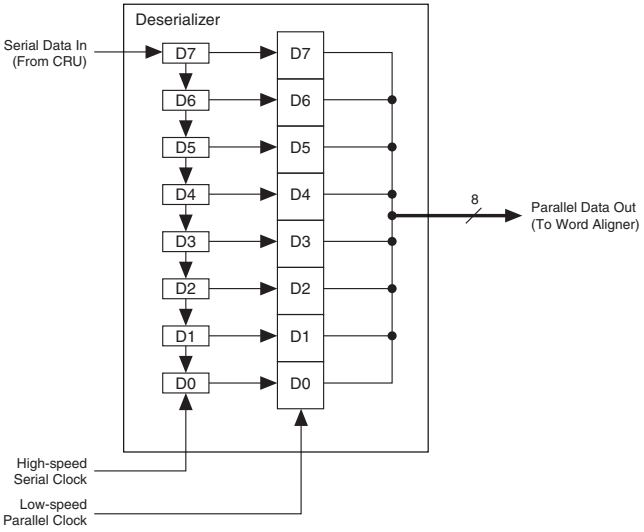
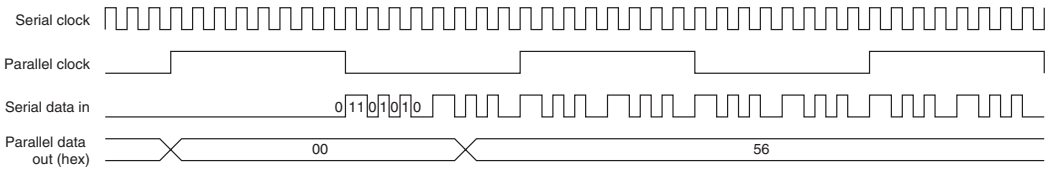


Figure 2–12 shows the serial bit order of the deserializer input and the parallel data out of the deserializer.



The serial data is received LSB to MSB.

Figure 2–12. Deserializer Bit Order



MegaWizard Plug-In Manager Analog Features

This section describes the analog options for the instantiation of the altgxb megafunction in the Quartus II MegaWizard® Plug-In Manager. Altera® recommends that the Stratix GX transceiver block be instantiated and parameterized through the MegaWizard Plug-In Manager. The MegaWizard Plug-In Manager offers a graphical user interface (GUI) that organizes the altgxb options in easy-to-use sections. The wizard also sets the proper ports and parameters automatically, based on the options and parameters you select. Invalid settings are automatically flagged to avoid illegal configurations.

Although you can instantiate the Stratix GX block directly by calling out the altgxb megafunction, Altera recommends using the MegaWizard Plug-In Manager to instantiate your altgxb megafunction, reducing the likelihood of invalid settings.

MegaWizard Plug-In Manager Analog Feature Considerations

Each altgxb MegaWizard Plug-In Manager instantiation uses one or more transceiver blocks based on the number of channels you select. There are four channels per transceiver block. If a MegaWizard Plug-In Manager instantiation uses fewer than four channels, the remaining channels in that transceiver block are not available for use.

Each MegaWizard Plug-In Manager instantiation must have similar functionality and data rates. If you want transceiver blocks that differ in functionality and data rates, create a separate MegaWizard Plug-In Manager instantiation for each transceiver block.

As mentioned in the clocking section, the MegaWizard Plug-In Manager also displays the configuration of the altgxb megafunction. Figures 2–13 through 2–19 change dynamically based on the selected mode, options, and clocking schemes.

Figure 2–13 shows the analog options on page 3 of the altgxb MegaWizard Plug-In Manager.

Figure 2–13. MegaWizard Plug-In Manager - altgxb (Page 3)

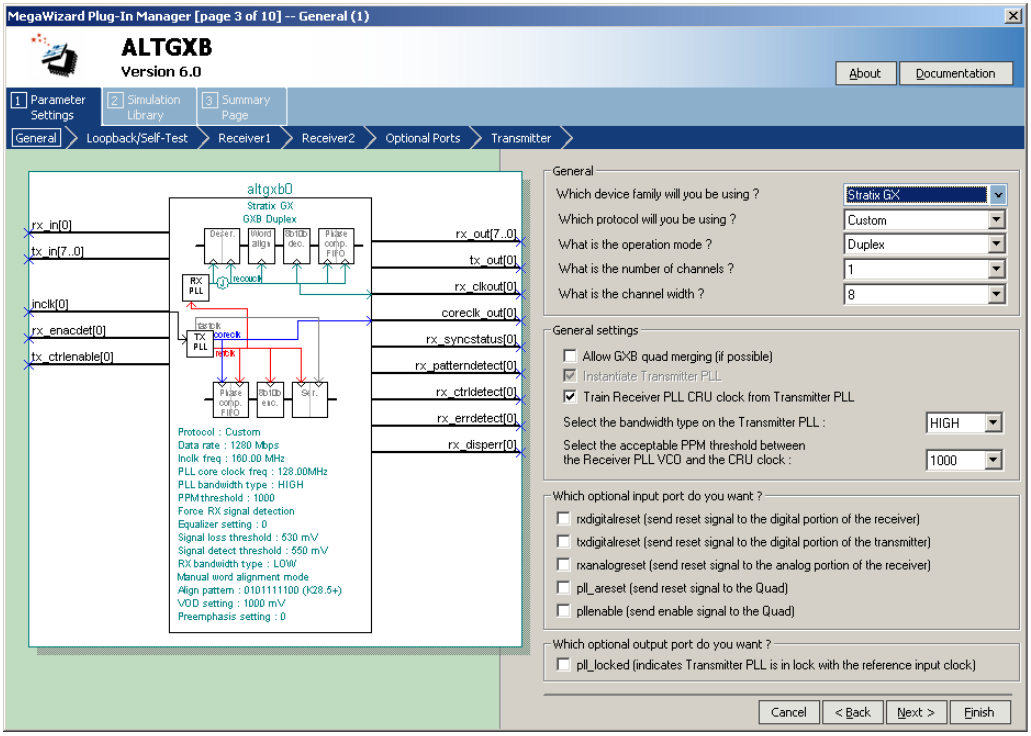


Table 2–6 describes the available options on page 3 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 2–6. MegaWizard Plug-In Manager Options (Page 3 for Analog) (Part 1 of 2)

altgxb Setting	Description
Instantiate Transmitter PLL	This option is available in receiver-only mode. It supports use of the transmitter PLL even when the transmit channel is disabled. Provides a non-recovered clock output for the logic array.
Train Receiver PLL CRU clock from Transmitter PLL	This option enables the transmitter PLL to train the receiver PLL. Use this option to support additional multiplication factors for the receiver PLL. This option also supports the separation of receiver and transmitter reference clocks. An additional input receiver reference clock (`rx_cruclk`) is available when this option is turned off. The first option that is enabled is needed for non-encoded 16-bit modes with a line rate of 2,600 Mbps or greater. For more details about this feature, refer to "Clock Synthesis" on page 2–6.

Table 2–6. MegaWizard Plug-In Manager Options (Page 3 for Analog) (Part 2 of 2)

altgxb Setting	Description
Select the bandwidth type on the Transmitter PLL	High bandwidth supports faster lock times. It also tracks higher frequency jitter (based on the –3-db frequency of the PLL gain plot) on the input clock. Low bandwidth has a smaller pass band to filter out more high-frequency jitter, but has a slower lock time.
Select the acceptable PPM threshold between the Receiver PLL VCO and the CRU clock	Selectable PPM difference tolerance {125, 250, 500, 1000} between the Receiver PLL VCO and the CRU clock. This is one of three parameters that affect the `rx_freqlocked` signal. If an out-of-tolerance event occurs, `rx_freqlocked` goes low.
`rxanalogreset` (send reset signal to the analog portion of the receiver)	The `rxanalogreset` port resets the receiver's analog circuits, including the receiver PLL. Each active receiver channel has its own analog reset.
`pll_areset` (send reset signal to the Quad)	The `pll_areset` port resets the entire transceiver block (all receiver and transmitter digital and analog circuits, including receiver and transmitter PLLs).
`pllenable` (send enable signal to the Quad)	The `pllenable` port enables the entire transceiver block; if deasserted, the entire transceiver block is held in the reset condition.
`pll_locked` (indicates Transmitter PLL is in lock with the reference input clock)	For more information, refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 2–14 shows the analog options on page 4 of the `altgxb` MegaWizard Plug-In Manager.

Figure 2–14. MegaWizard Plug-In Manager - altgxb (Page 4)

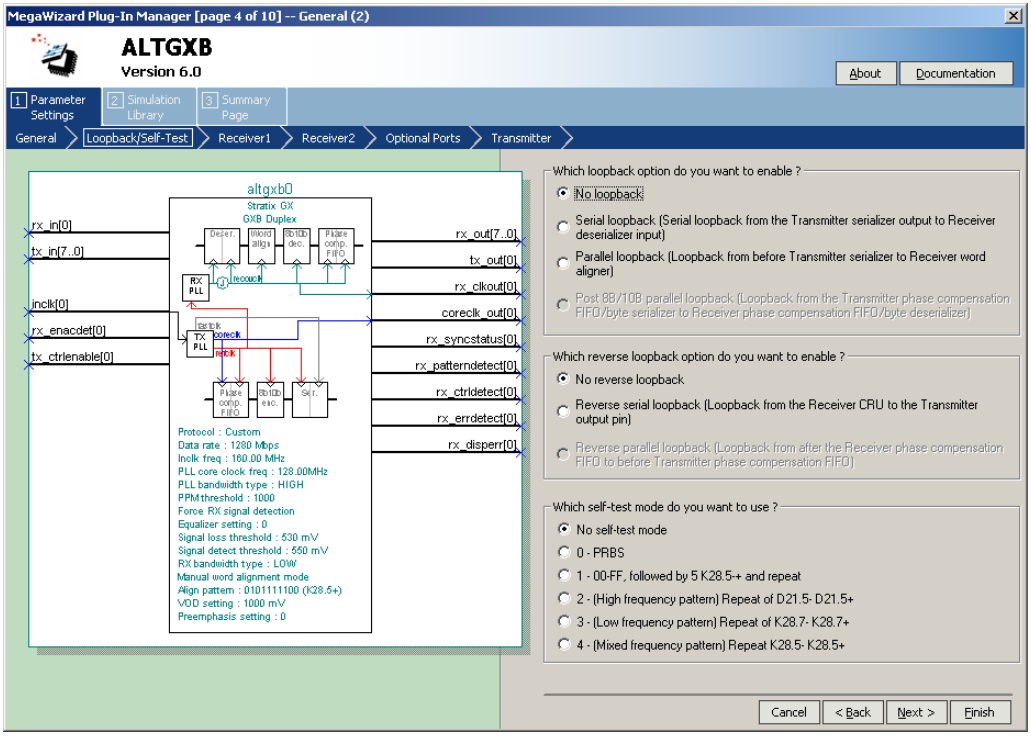


Figure 2–15 shows the analog options on page 5 of the `altgxb` MegaWizard Plug-In Manager.

Figure 2–15. MegaWizard Plug-In Manager - altgxb (Page 5)

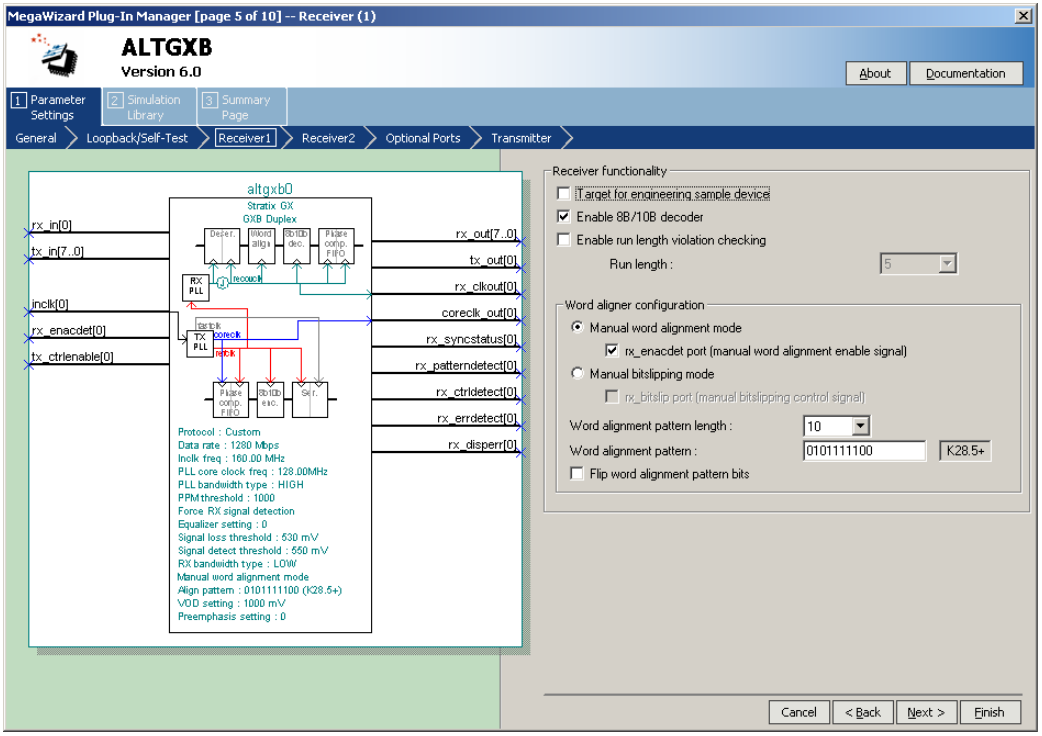


Table 2–7 describes the available options on page 5 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 2–7. MegaWizard Plug-In Manager Options (Page 5 for Analog)

altgxb Setting	Description
Enable run-length violation checking	Enable run-length violation circuit. If enabled, the optional output pin `rx_rlv` is available and pulses high when the specified run length is violated. In 8- or 16-bit mode, set the run length threshold from 4 to 124 in steps of 4. In 10 and 20-bit mode, or if using 8B/10B, set the run-length threshold from 5 to 160 in steps of 5.

Figure 2–16 shows the analog options on page 6 of the `altgxb` MegaWizard Plug-In Manager.

Figure 2–16. MegaWizard Plug-In Manager - altgxb (Page 6)

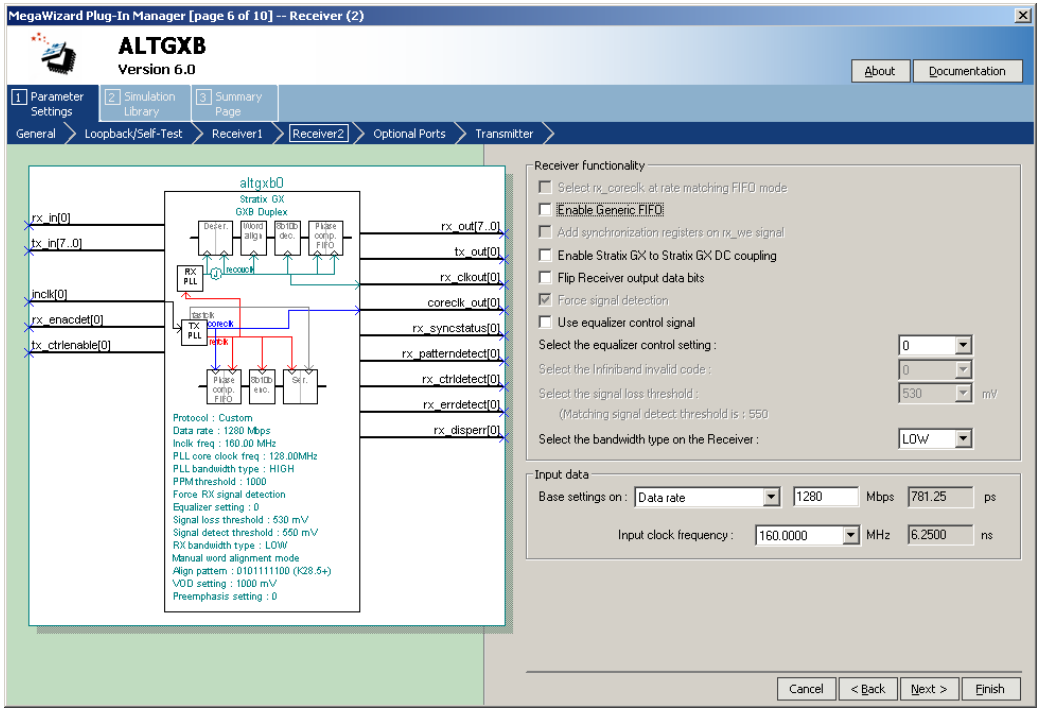


Table 2–8 describes the available options on page 6 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 2–8. MegaWizard Plug-In Manager Options (Page 6 for Analog) (Part 1 of 2)

altgxb Setting	Description
Enable Stratix GX to Stratix GX DC coupling	Stratix GX to Stratix GX DC coupling only. Lets the receiver accept a 1.5-V PCML signal from a Stratix GX transmitter buffer.
Force signal detection	This option forces `rx_signaldetect` to be set HIGH. This option is always enabled in the current version of the software.
Use equalizer control signal	The **Use equalizer control signal** option enables dynamic equalization via the optional `rx_equalizerctrl` input port.
Select the equalizer control setting	If this control signal is not used, you can set equalization in the MegaWizard Plug-In Manager via the **Select the equalizer control setting**. The valid values are 0 through 4, with 0 being off and 4 being the largest gain setting.

Table 2–8. MegaWizard Plug-In Manager Options (Page 6 for Analog) (Part 2 of 2)	
altgxb Setting	**Description**
Select the signal loss threshold	This option is unavailable in the current version of the software. Receiver signal detection is always forced.
Select the bandwidth type on the Receiver	Available settings are High, Medium, and Low. High bandwidth allows for faster lock times and tracks higher frequency jitter (based on the -3 db frequency of the PLL gain plot) on the input clock. Low bandwidth contains a smaller pass band to filter out more high-frequency jitter, but has slower lock times.

Figure 2–17 shows the analog options on page 7 of the `altgxb` MegaWizard Plug-In Manager.

Figure 2–17. MegaWizard Plug-In Manager - altgxb (Page 7)

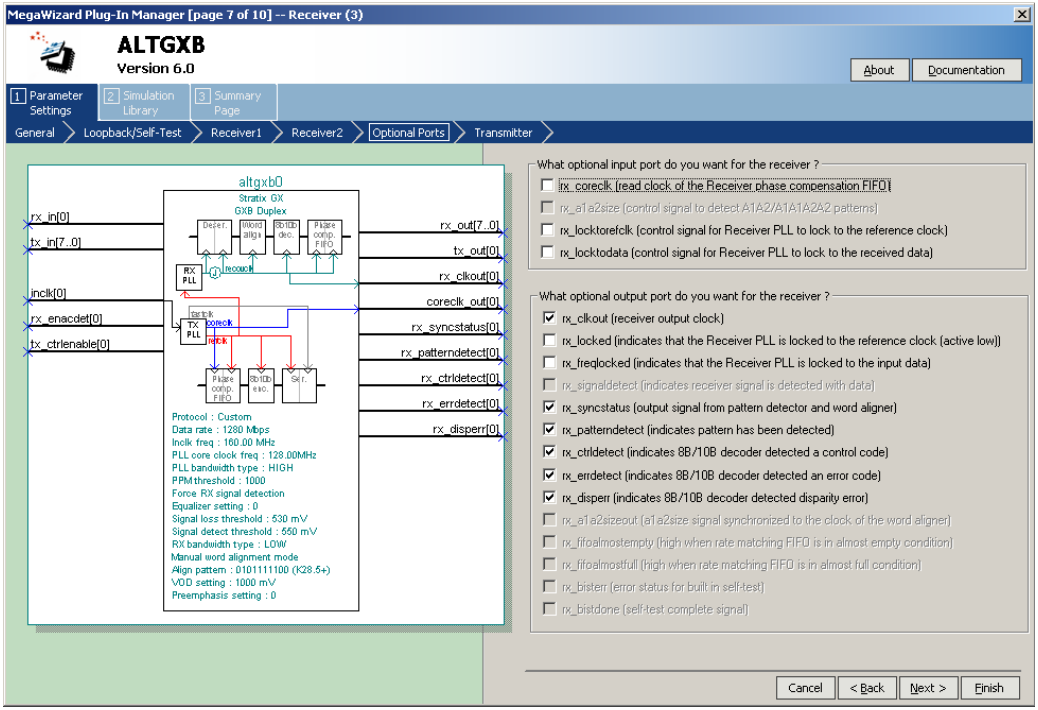


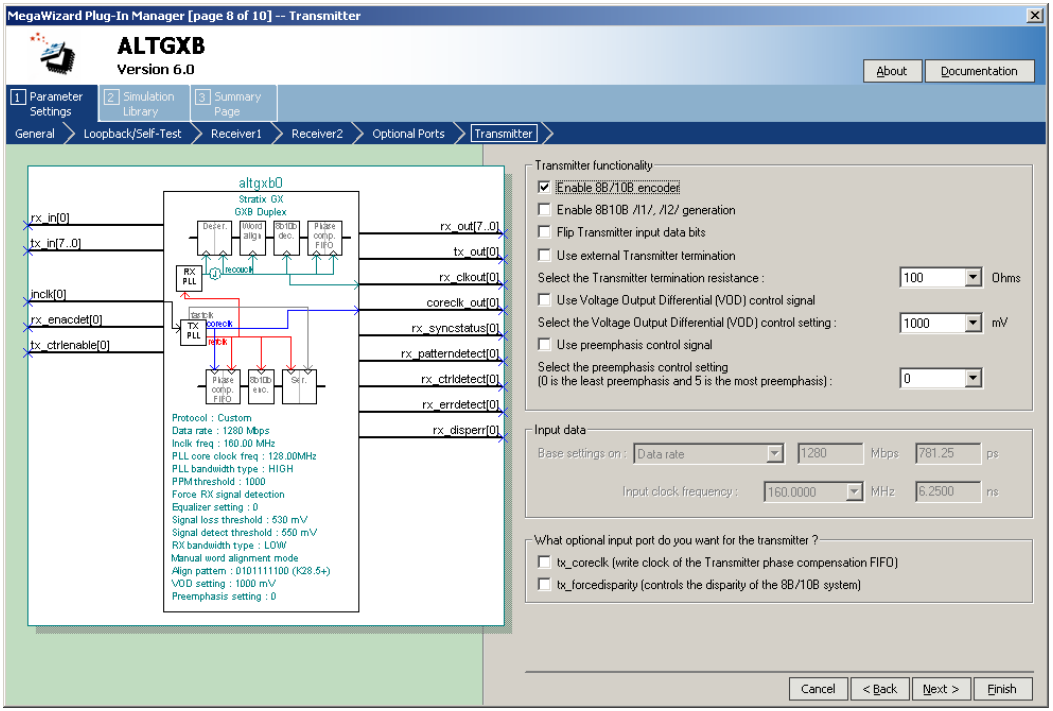
Table 2–9 describes the available options on page 7 of the MegaWizard Plug-In Manager for your altgxb custom megafunction variation.

Table 2–9. MegaWizard Plug-In Manager Options (Page 7 for Analog)

altgxb Setting	Description
rx_locktorefclk (control signal for Receiver PLL to lock to the reference clock)	Optional input signal that forces the CRU to lock to the reference clock. This disables the auto switch- over mode that switches the CRU to lock-to-data mode. If both rx_locktorefclk and rx_locktodata are asserted, then rx_locktodata takes precedence.
rx_locktodata (control signal for Receiver PLL to lock to the received data)	Optional input signal that forces the CRU to lock to the incoming data. If both rx_locktorefclk and rx_locktodata are asserted, rx_locktodata takes precedence.
rx_clkout (receiver input clock)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_locked (indicates that the Receiver PLL is locked to the reference clock (active low))	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_freqlocked (indicates that the Receiver PLL is locked to the input data)	Optional output signal that indicates when the CRU is locked to the incoming data stream. The lock indication is based on the following conditions: • The CRU PLL is within the prescribed PPM frequency threshold setting (125 PPM, 250 PPM, 500 PPM, 1,000 PPM) of the CRU reference clock. • The reference clock and CRU PLL output are phase matched (~ phases are within 0.08 UI).
rx_signaldetect (indicates receiver signal is detected with data)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_syncstatus (output signal from pattern detector and word aligner)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 2–18 shows the analog options on page 8 of the altgxb MegaWizard Plug-In Manager.

Figure 2–18. MegaWizard Plug-In Manager - altgxb (Page 8)



Notes to Figure 2–18:

(1) The **Use V_{OD} control signal** option enables dynamic V_{OD} adjustment via the optional `tx_vodctrl` input port. If this control signal is not used, set the V_{OD} in the MegaWizard Plug-In Manager via the **Select the V_{OD} control setting** option. The valid values are based on your transmitter termination value and range from 400 to 1,600 mV.

(2) The **Use Preemphasis control signal** option enables dynamic pre-emphasis control using the optional `tx_preemphasisctrl` input port. If this control signal is not used, set the pre-emphasis in the MegaWizard Plug-In Manager using the **Select the preemphasis control setting**. The valid values are 1 through 5, where 1 is the smallest pre-emphasis value and 5 is the largest. The amount of pre-emphasis is based on your V_{OD} values.

Table 2–10 describes the available options on page 8 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 2–10. MegaWizard Plug-In Manager Options (Page 8 for Analog) (Part 1 of 2)

altgxb Setting	Description
Use external Transmitter termination	This option allows you to disable the on-chip termination on `tx_out` and instead use external termination.
Use Voltage Output Differential (VOD) control signal	The **Use V_{OD} control signal** option enables dynamic V_{OD} adjustment via the optional `tx_vodctrl` input port.

Table 2–10. MegaWizard Plug-In Manager Options (Page 8 for Analog) (Part 2 of 2)	
altgxb Setting	**Description**
Select the Voltage Output Differential (VOD) control setting	If this control signal is not used, set the V_{OD} in the MegaWizard Plug-In Manager via the **Select the V_{OD} control setting** option. The valid values are based on your transmitter termination value and range from 400 to 1,600 mV.
Use preemphasis control signal	The Use Preemphasis control signal option enables dynamic pre-emphasis control using the optional `tx_preemphasisctrl` input port.
Select the preemphasis control setting (0 is the least preemphasis and 5 is the most preemphasis)	If this control signal is not used, set the pre-emphasis in the MegaWizard Plug-In Manager using the Select the preemphasis control setting. The valid values are 1 through 5, where 1 is the smallest pre-emphasis value and 5 is the largest. The amount of pre-emphasis is based on your VOD values.

Figure 2–19 shows page 9, the Simulation Libraries page, of the MegaWizard Plug-In Manager for the analog feature.

Figure 2–19. MegaWizard Plug-In Manager - altgxb (Page 9)

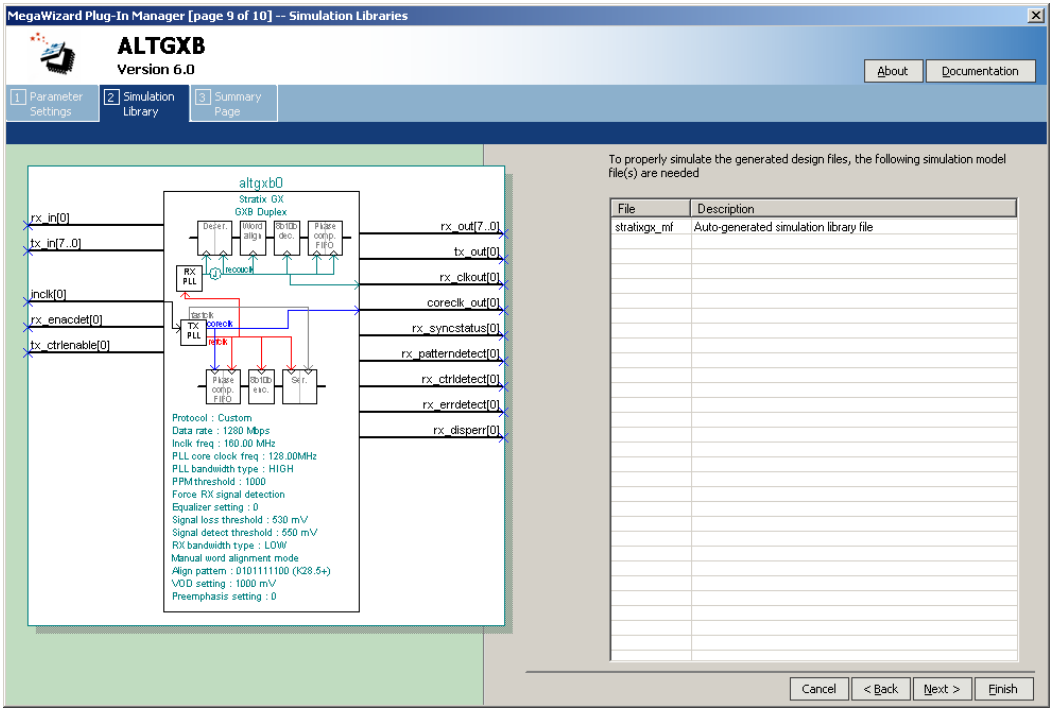
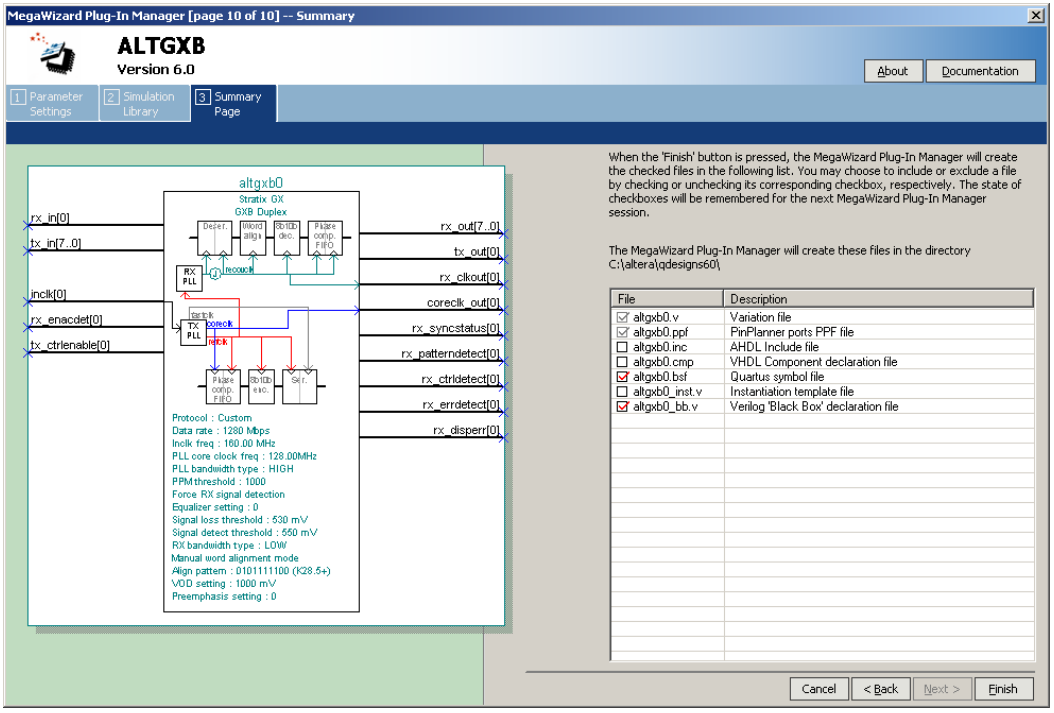


Figure 2–20 shows page 10 of the MegaWizard Plug-In Manager for the analog feature. You can select optional files on this page. After you make your selections, click **Finish** to generate the files.

Figure 2–20. MegaWizard Plug-In Manager - altgxb (Page 10)



Stratix GX Transceiver Merging

A transceiver block contains four transceivers. In a design, an `altgxb` instantiation is placed in one or more transceiver blocks and potentially leaves unused transceivers in a block. For example, a six transceiver instantiation completely fills one transceiver block and half fills a second, taking up two full transceiver blocks. If another instantiation is in the design, it is placed the same way. For example, an instantiation of two transmitters takes up a third transceiver block. Merging two of the partially filled transceiver blocks into one transceiver block reduces the resources used and allows a design to fit into a device with fewer transceiver blocks.

The altgxb MegaWizard Plug-In Manager in the Quartus II software has a feature that allows merging of similar quads (transceiver blocks). With a few exceptions, transceiver blocks can be merged if the options chosen in the wizard are the same. The Quartus II software merges the transceiver blocks automatically if you turn on the merging option for each instantiation. Table 2–11 shows the considerations for merging.

Table 2–11. Stratix GX Merging Considerations

Merging Rules	Required to Match Between Transceiver Blocks	Not Required to Match Between Transceiver Blocks
MegaWizard Plug-In Manager options	USE_8B_10_MODE USE_DOUBLE_DATA_MODE CHANNEL_WIDTH SYNC_MODE DATA_RATE TRANSMIT_PROTOCOL	VOD Pre-emphasis Equalization Number of transmitters Number of receivers
Input control signals must be shared across transceiver blocks and must be from the same source	Clock CRU_CLOCK PLL_RESET PLL_ENABLE	
The merging partial transceiver blocks must reduce the number of transceiver block when merged	The total number of receivers and transmitters must be tx ≤4 x *n* and rx ≤4 x *n*, where *n* is the reduced number of transceiver blocks remaining after merging.	

The Quartus II software does not merge two transceiver blocks if they won't completely fit into one. It can, however, merge three transceiver blocks into two. Figure 2–21 shows a configuration with three transceiver blocks that can potentially be merged.

Figure 2–21. Three Transceiver Configuration



In Figure 2–21, two transceiver blocks cannot merge because there would be extra channels remaining. But because there are three transceiver blocks, the Quartus II software can merge them into two with no remaining channels.

If you turn on the merging option, the Quartus II software automatically merges transceiver blocks if possible or provides a warning during compilation if merging is not possible. The merging feature can reduce the transceiver block resources used in your design.

3. Custom Mode

SGX52003-1.2

Introduction

The Custom mode of the Stratix® GX device includes the following features:

- Serial data rate range from 500 Mbps to 3.1875 Gbps
- Input reference clock range from 25 to 650 MHz
- Parallel interface width of 8, 10, 16, or 20-bit support
- 8B/10B encoder/decoder can be enabled or bypassed
- Word aligner supports 7-bit, 10-bit, 16-bit, or bit slip mode

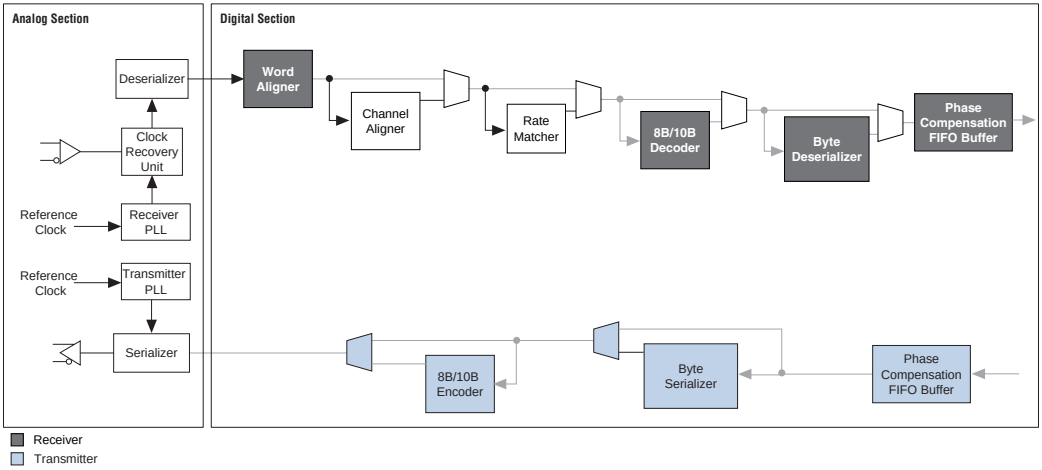
Applications like packet or streaming data applications, chip-to-chip connectivity, backplanes, or board-to-board connectivity, which do not have a defined protocol overhead or a custom protocol to transfer data serially over a medium, can use the Custom mode offered by Stratix GX devices. The Custom mode includes SERDES and parallel interconnect functionality. In this mode, the transceiver performs serialization and de-serialization with an optional 8B/10B coding scheme. Custom mode is not aware of the system level protocol wrapped on top of it.

Custom mode enables a subset of the transceiver blocks for customizable configuration. The channel aligner and the rate matcher features are not available in Custom mode.

This chapter details the supported digital architecture, clocking schemes, and software implementation for Custom mode. Figure 3–1 shows a block diagram of a duplex channel configured in Custom mode.

The digital section starts at the word aligner of the receiver channel and propagates up to the device logic array.

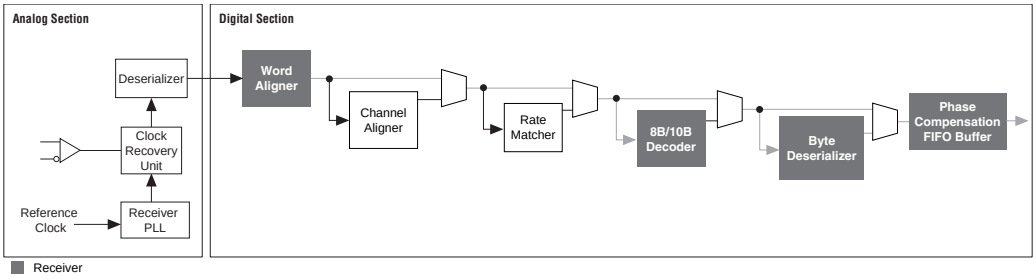
Figure 3–1. Block Diagram of a Duplex Channel Configured in Custom Mode



Custom Mode Receiver Architecture

Figure 3–2 shows a block diagram of the digital components of the receiver in Custom mode.

Figure 3–2. Block Diagram of the Receiver Digital Components in Custom Mode



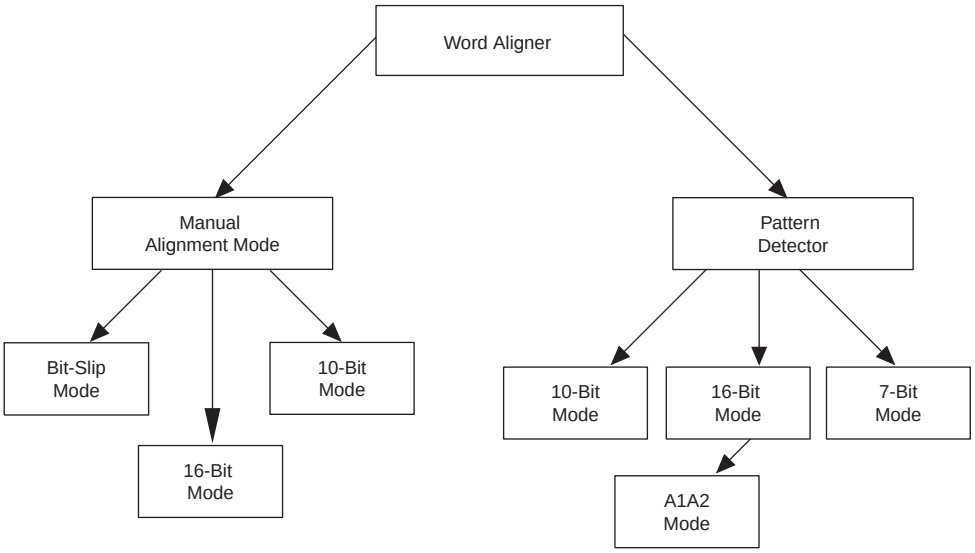
Word Aligner

For embedded clocking schemes, the clock is recovered from the incoming data stream based on transition density of the data. This feature eliminates the need to factor in receiver skew margins between the clock and data. However, with this clocking methodology, the word boundary of the re-timed data can be altered. Stratix GX devices offer an embedded

word alignment circuit that is used in conjunction with the pattern detector to align the word boundary of the re-timed data to a specified comma. In Custom mode, this embedded circuit is configured to manual alignment mode, which consists of 10-bit, 16-bit, and bit-slip modes.

The word aligner is composed of a pattern detector, manual alignment controller, bit-slipper circuitry, and synchronization state machines. Depending on the configuration, these components work in conjunction or independently of one another. The word aligner cannot be bypassed, but if the `rx_enacdet` signal is not used, the word aligner does not alter the data. Figure 3–3 shows the various components of the word aligner in Custom mode. The functionality is described in the following sections.

Figure 3–3. Components in the Stratix GX Word Aligner



Pattern Detector Module

The pattern detector matches a predefined comma to the current byte boundary. If the comma is found, the optional `rx_patterndetect` signal is asserted for the duration of one clock cycle to signify that the comma exists in the current word boundary. The pattern detector module only indicates that the signal exists and does not modify the word boundary. Modification of the word boundary is discussed in the word alignment and synchronization sections.

A 10-bit pattern, 7-bit pattern, or 16-bit pattern can be programmed for the pattern detector to recognize. Refer to the section "Custom Mode MegaWizard Plug-In Manager" on page 3–29 for more details.

10-Bit Pattern Mode

When the word alignment pattern length parameter in the MegaWizard® Plug-In Manager is set to 10, the module matches the 10-bit comma with the data and its complement in the current word boundary. Both positive and negative disparities are checked in this mode. For example, if a `/K28.5/ (b'0011111010)` pattern is specified as the comma, the `rx_patterndetect` is asserted if `b'0011111010` or `b'1100000101` is detected in the incoming data.

7-Bit Pattern Mode

When the word alignment pattern length parameter in the MegaWizard Plug-In Manager is set to 7, the module matches the 7-bit comma specified in the wizard field parameter with the seven least significant bits (LSB) of the data and its complement in the current word boundary. Both positive and negative disparities are also checked in this mode.

The 7-bit pattern mode is useful because it can mask out the three most significant bits of the data. This lets the pattern detector recognize multiple commas. For example, in the 8b/10b encoded data, a `/K28.5/ (b'0011111010),/K28.1/ (b'0011111001),` and `/K28.7/ (b'0011111000)` shares seven common LSBs, so masking the three MSBs lets the pattern detector resolve all three commas.

16-Bit Pattern Mode

The two consecutive 8-bit characters (`A1A2`) are used as the comma in 16-bit pattern mode.

`A1` represents the least significant byte, which consists of bits `[7..0]`, and `A2` represents the most significant byte, consisting of bits `[15..8]`. Therefore, the comma must be specified as `[A2,A1]` in the MegaWizard Plug-In Manager word alignment comma section. Only the positive disparity of the comma is detected in the mode. The A1A1A2A2 mode is only available when SONET is specified as the protocol.

Table 3–1. Pattern Detector Comma Patterns in Custom Mode

Pattern Detect Mode	Data Width	Disparity
10-bit	10-bits, 20-bits	±
7-bit	10-bits, 20-bits	±
Two consecutive 8-bit characters	8-bits, 16-bits	+

Manual Alignment Modes

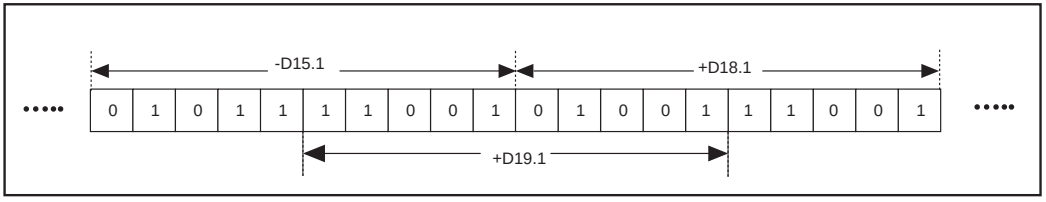
The Stratix GX device supports manual alignment in 10-bit, 16-bit, and bit-slipping modes.

Manual 10-Bit Alignment Mode

You can configure the word aligner to align to a 10-bit word boundary if you use 8B/10B encoding or if you specify the data width to be either 10- or 20-bits wide. In this mode, the internal word alignment circuitry barrel shifts the correct word boundary if the comma specified in the pattern detector is detected in the data stream.

When `rx_enacdet[]` is high, the word aligner detects the specified comma and re-aligns the byte boundary, if needed. The `rx_syncstatus[]` signal is asserted for one clock cycle to signify that the word boundary has been synchronized. The `rx_enacdet[]` signal can be held high if the comma is known to be unique and does not also appear across the byte boundaries of other data. For example, if the design uses an encoding scheme such as 8B/10B to guarantee that the /K28.5/ code group is a unique pattern in the data stream, the `rx_enacdet` is held high. In situations where the comma exists between word boundaries, `rx_enacdet` must be controlled to avoid false word alignment. For example, suppose that you use 8B/10B encoding and specify a /+D19.1/ (b'110010 1001) character as the comma. In this case, a false word boundary is detected if a /-D15.1/ (b'010111 1001) is followed by a /+D18.1/ (b'010011 1001). (Refer to Figure 3–4.)

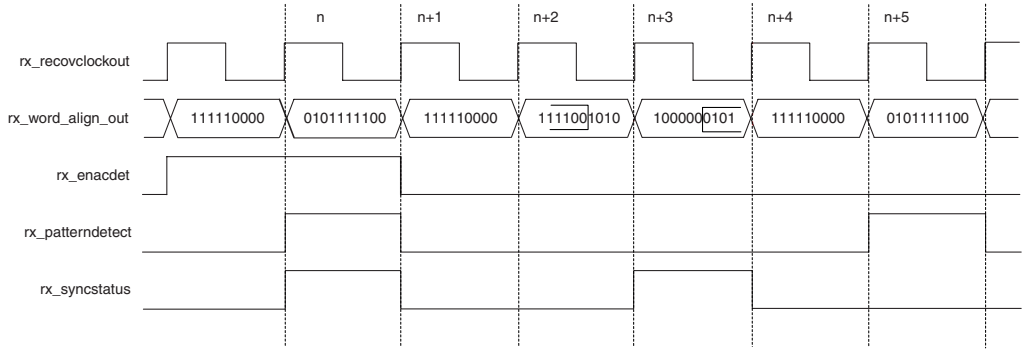
Figure 3–4. False Word Boundary Alignment if the Comma Exists Across Word Boundaries



The `rx_enacdet` signal must be deasserted after the initial word alignment is found, so as to prevent false word boundary alignment. When `rx_enacdet` is deasserted, the current word boundary is locked even if the comma is detected across different boundaries. In this case, `rx_syncstatus[]` acts as a re-synchronization signal to signify that the comma was detected, but the boundary is different than the current boundary. For best results, monitor this signal and reassert `rx_enacdet[]`, if re-alignment is desired.

Figure 3–5 shows an example of how the word aligner signals interact in 10-bit alignment mode. For this example, a `/K28.5/ (10'b0011111010)` is specified as the comma. Because `rx_enacdet` is held high at time *n*, alignment occurs whenever a comma exists in the pattern. The `rx_patterndetect` signal is asserted for one clock cycle to signify that the pattern exists on the re-aligned boundary. The `rx_syncstatus` signal is also asserted for one clock cycle to signify that the boundary has been synchronized.

Figure 3–5. Example of How the Word Aligner Symbols Interact in 10-Bit Manual Alignment Mode



At time *n*+1 the `rx_enacdet` signal is deasserted, which instructs the word aligner to lock the current word boundary. The comma is detected at time *n*+2, but it exists on a different boundary than the current locked boundary. Because the bit orientation of the Stratix GX device is LSB to MSB, it follows, from the waveform, that the comma exists across time *n*+2 and *n*+3. In this condition, the `rx_patterndetect` signal remains low because the comma does not exist on the current word boundary, but the `rx_syncstatus` signal is asserted for one clock cycle to signify a resynchronization condition. This means that the comma has been detected, but across another word boundary. The logic of the design determines whether to assert the `rx_enacdet` signal to re-initiate the word alignment process. At time *n*+5 the `rx_patterndetect` signal is asserted for one clock cycle to signify that the comma has been detected on the current word boundary.

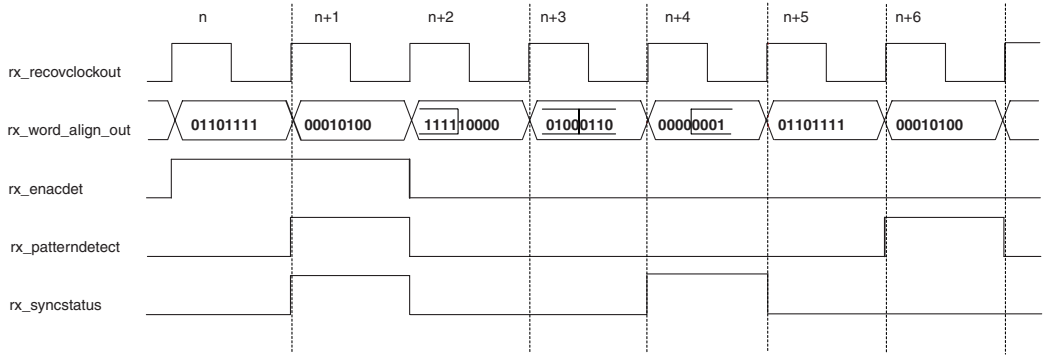
Manual 16-bit Alignment Mode

You can enable the 16-bit alignment mode if the data widths are 8-bits or 16-bits. This mode is similar to the manual A1A2 SONET alignment mode, except that the `rx_a1a2size[]` and `rx_a1a2sizeout[]` signals are not available.

The byte boundary is locked after the first comma is detected and aligned after the rising edge of the rx_enacdet[] signal. If the byte boundary changes, the rx_enacdet[] signal must be deasserted and reasserted to reset the alignment circuit. On the rising edge of the rx_enacdet[], the word aligner locks onto the first comma detected. In this scenario, the rx_patterndetect[] signal is asserted to signify that the comma has been aligned. Also, the rx_syncstatus[] signal is asserted for a clock cycle to signify that the word boundary has been synchronized. After the word boundary has been locked, regardless of whether the rx_enacdet[] is high or low, the rx_syncstatus[] signal asserts itself for one clock cycle whenever a comma is detected across a different byte boundary. The rx_syncstatus[] signal operates in this re-synchronization state until a rising edge is detected on rx_enacdet[].

Figure 3–6 shows how the word aligner signals interact in 16-bit alignment mode for an A1A2 pattern.

Figure 3–6. Example of How the Word Aligner Signals Interact in SONET A1A2 Manual Alignment Mode



In Figure 3–6, the rx_enacdet signal is toggled high at time *n*, at which point the aligner locks to the boundary of the next present comma. The A1 comma also appears on the rx_word_align_out port during this period. At time *n*+1 the A2 comma appears on the rx_word_align_out port. Because the comma exists, the rx_patterndetect and rx_syncstatus signals are asserted for one clock cycle to signify that the A1A2 comma has been detected and the word boundary has been locked. The A1A2 comma appears again across word boundaries during periods *n*+2, *n*+3, and *n*+4. The rx_enacdet signal is held high, but the word aligner does not re-align the byte boundary as it would in 10-bit manual alignment mode. Instead, the rx_syncstatus signal is asserted for one clock cycle to signify a re-synchronization condition. You must deassert and reassert the rx_enacdet signal to retrigger the word aligner. The next transition occurs at time *n*+5, where rx_enacdet is

deasserted and the A1 pattern is present on the rx_word_align_out port. At time *n*+6, the A2 pattern is present on the rx_word_align_out port. The word aligner then asserts the rx_patterndetect signal for one clock cycle to flag the detection of the comma on the current word boundary.

Manual Bit-Slipping Alignment Mode

You can also achieve word alignment by enabling the manual bit-slip option. With this option enabled, the transceiver has the ability to shift the word boundary one-bit every parallel clock cycle. Bits are shifted from the MSB to LSB direction. Shifting occurs every time the bit-slipping circuitry detects a rising edge of the rx_bitslip[] signal. Each time a bit is slipped, the bit that arrived at the receiver earlier is skipped. When the word boundary matches what is specified as the comma, the rx_patterndetect[] signal is asserted for one clock cycle. For best results, implement the logic in the device logic array to control the bit-slip circuitry.

This scheme is useful if the comma changes dynamically when the Stratix GX device is in user mode. Because the controller is implemented in the logic array, a custom controller can be built to dynamically change the comma without needing to reprogram the Stratix GX device.

Figure 3–7 shows an example of how the word aligner signals interact in the manual bitslip alignment mode. For this example, 8'b00111100 is specified as the comma, and an 8'b11110000 value is held at the rx_in port. Every rising edge on the rx_bitslip port causes the rx_word_align_out data to shift a bit from the MSB to the LSB. This shifting is shown at time *n*+2, where the 8'b11110000 data is shifted to a value of 8'b01111000. At this state, the rx_patterndetect is held low, because the specified comma does not exist in the current word boundary. The rx_bitslip is disabled at time *n*+3 and re-enabled at time *n*+4. The output of the rx_word_align_out now matches the specified comma, so the rx_patterndetect is asserted for one clock cycle. At time *n*+5 the rx_patterndetect is still asserted since the comma still exists in the current word boundary. Finally, at time *n*+6 the rx_word_align_out boundary is shifted again, and the rx_patterndetect signal is deasserted to signify that the word boundary does not contain the comma.

Figure 3–7. Example of How the Word Aligner Symbols Interact in Manual Bitslip Mode

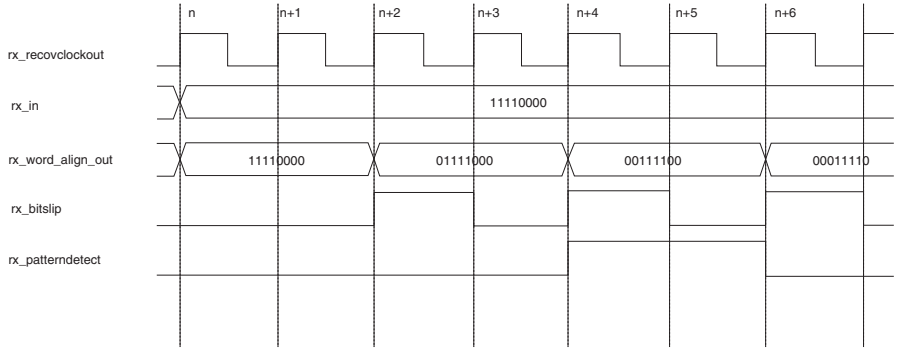


Table 3–2. Word Alignment Support for Custom Mode

Word Alignment Mode	Effective Mode	Control Signals	Status Signals
Manual 10-bit alignment Mode	Alignment to detected pattern when allowed by the `rx_enacdet` signal	`rx_enacdet`	`rx_syncstatus` `rx_patterndetect`
Manual 16-bit alignment Mode	Alignment to detected pattern when allowed by the `rx_enacdet` signal	`rx_enacdet`	`rx_syncstatus` `rx_patterndetect`
Manual bit-slipping alignment mode	Manual bit slip controlled by the device logic array	`rx_bitslip`	`rx_patterndetect`

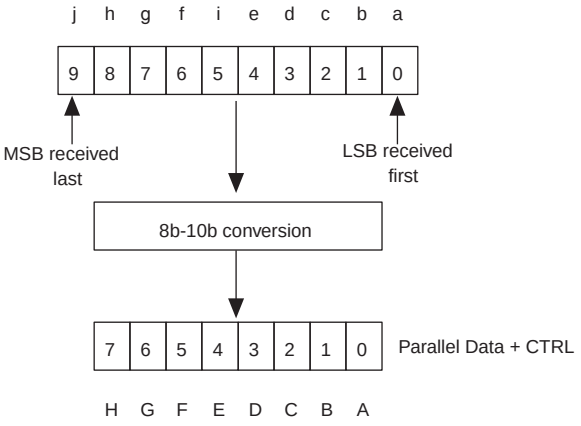
8B/10B Decoder

The 8B/10B decoder is part of the Stratix GX transceiver block. The 8B/10B decoder restores the 8-bit data + 1-bit control identifier from the 10-bit code.

10-bit Decoding

The 8B/10B decoder translates the 10-bit encoded code into the 8-bit equivalent data or control code. The 10-bit encoded code is received LSB to MSB. The data received must come from the supported Dx.y or Kx.y list. All 8B/10B control signals (Disparity error, control detect, and code error) are pipelined with the data in the Stratix GX receiver block and are edge-aligned with the data. Figure 3–8 diagrams the 10-bit to 8-bit conversion.

Figure 3–8. 10-Bit to 8-Bit Conversion



Reset

The `rxdigitalreset` signal governs the reset condition of the 8B/10B decoder. In reset, the disparity registers are cleared. Upon exiting reset, the 8B/10B decoder can start with either a positive or negative disparity. The decoder calculates the initial running disparity based on the first valid code received.

The receiver block must be word aligned after reset before the 8B/10B decoder can decode valid data or control codes.

Code Error Detect

The `rx_errdetect` signal indicates when the code that is received contains an error. This port is optional and if not in use, there is no way to determine whether a code that is received is valid. The `rx_errdetect` signal goes high if a code received is an invalid code or if it contains a disparity error. If a code is received that is not part of the valid Dx.y or Kx.y list, the `rx_errdetect` signal goes high. This signal is aligned with the invalid code word that is received at the device logic array and/or the code word that triggered the disparity error.

Disparity Error Detector

The 8B/10B decoder can detect disparity errors based on which 10-bit code it received. The disparity error is indicated at the optional `rx_disperr` port. The current running disparity is based on the

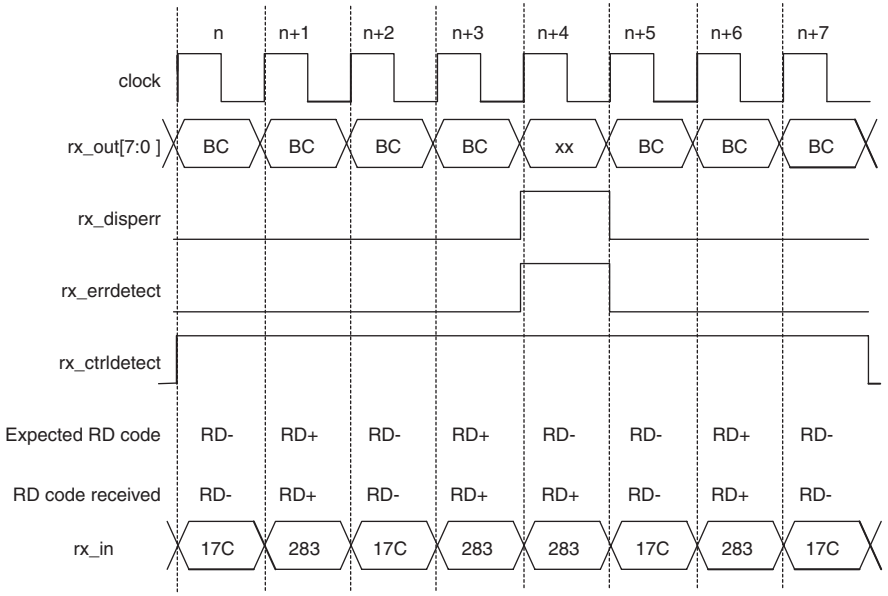
disparity calculation of the last code it received. The disparity calculation is described in the *Data & Control Codes* chapter of the *Stratix GX Device Handbook, Volume 2*.

If negative disparity is calculated for the last 10-bit code, a neutral or positive disparity 10-bit code is expected. If the decoder does not receive a neutral or positive disparity 10-bit code as the next code word, the `rx_disperr` signal goes high, indicating that the code that was received contained a disparity error.

If a positive disparity is calculated, the next code-group must be a neutral or negative disparity 10-bit code. The `rx_disperr` signal goes high if the code that is received is not as expected. When the `rx_disperr` signal transitions high, `rx_errdetect` also transitions high.

Figure 3–9 shows a case where the disparity is violated. A K28.5 code has an 8-bit value of 8'hbc and a 10-bit value (`jhgfiedcba`). The 10-bit value is 10'b0011111010 (10'h17c) for RD- or 10'b1100000101 (10'h283) for RD+. Assume that the running disparity at time n-1 is negative, so the expected code at time *n* is taken from the RD- column. Because a K28.5 does not have a balanced 10-bit code (equal number of 1's and 0's), the expected RD code toggles back and forth between RD- and RD+. At time *n*+3, the 8B/10B decoder received a RD+ K28.5 code (10'h283), which would make the current running disparity negative. At time *n*+4, because the current disparity is negative, a K28.5 from the RD- column is expected, but a K28.5 code from the RD+ is received instead. This code prompts `rx_disperr` to go high during time *n*+4 to indicate that this particular K28.5 code contained a disparity error. The current running disparity at the end of time *n*+4 is negative because a K28.5 from the RD+ column was received. Based on the current running disparity at the end of time *n*+5, a positive disparity K28.5 code (from the RD-) column is expected at time *n*+5.

Figure 3–9. Disparity Error

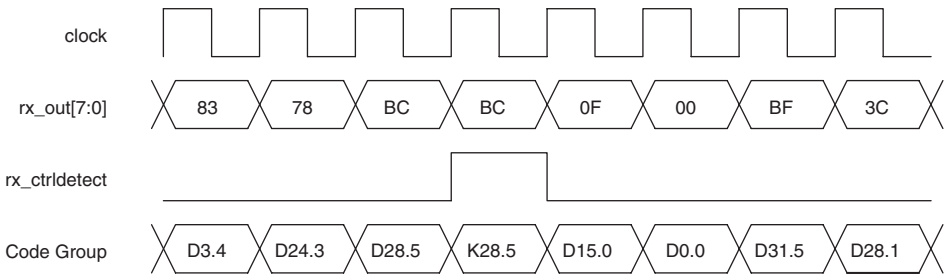


Control Detect

The 8B/10B can differentiate between data and control codes via the `rx_ctrldetect` port. This port is optional, and if it is not in use, there is no way of differentiating a Dx.y from a Kx.y.

Figure 3–10 shows an example waveform demonstrating the receipt of a K28.5 code (BC + ctrl). The `rx_ctrldetect=1'b1` is aligned with 8'hbc, indicating that it is a control code.

Figure 3–10. Control Code Detection



Byte Deserializer

The byte deserializer module further reduces the speed at which the FPGA logic array must run in order to meet performance. If the input is 10 bits of data, the output to the FPGA logic array is deserialized to 20 bits. If the input is 8 bits of data, the output to the FPGA logic array is deserialized to 16 bits. The byte deserializer does not process the data and as such, the control signals that are fed to the module are only processed to match the latency to the data.

The byte deserializer in the receiver block takes in a maximum of 13 bits. It is possible to feed the following to the byte deserializer:

- 8 bits of data and up to two control signals (`rx_patterndetect` and `rx_syncstatus`)
- 8 bits of data and up to five control signals (`rx_patterndetect`, `rx_syncstatus`, `rx_disperr`, `rx_ctrldetect`, and `rx_errdetect`)
- 10 bits of data and up to two control signals (`rx_patterndetect` and `rx_syncstatus`)

The byte deserializer outputs up to 26 bits, depending on the number of bits that was passed to it. When the input includes data and control signals, the data and the control signals are deserialized to include double the data bits and two bits of each control signal, one for the MSB and one for the LSB. The aggregate bandwidth does not change by using the byte deserializer, because the logic array data width is doubled.

Figure 3–11 demonstrates input and output signals of the byte deserializer when deserializing a 10-bit data input to 20 bits. In this case, the alignment pattern A (1010100000) is located in the MSB of the 20-bit output, and this is reflected with `rx_patterndetect [1]` going high. The output of the byte deserializer is AX, CB, ED, and so on.

Figure 3–11. Receiver Byte Deserialzer in 10/20-Bit Mode With Alignment Pattern in MSB

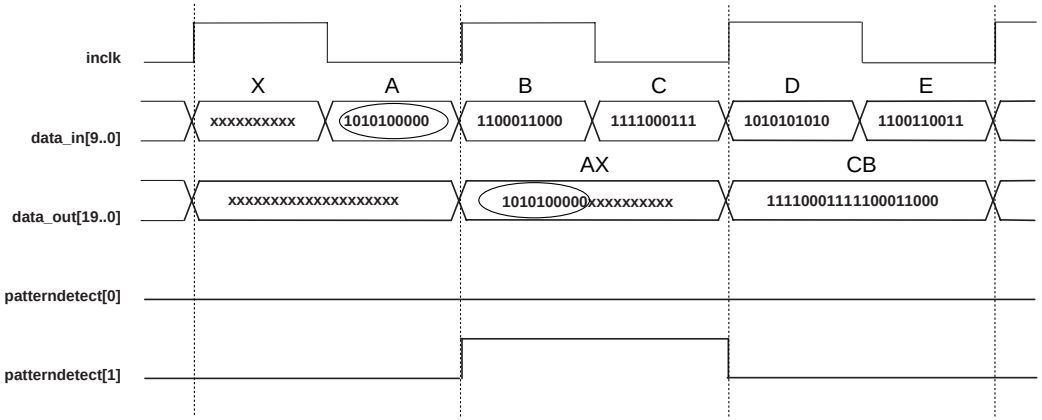
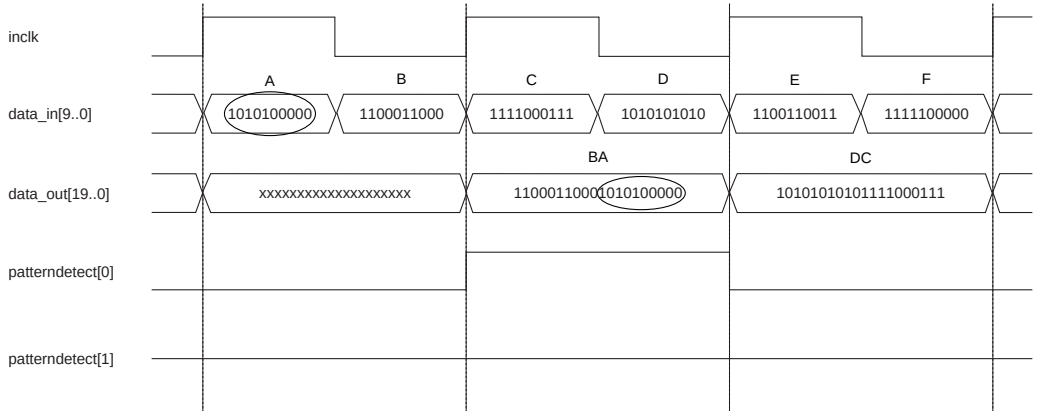


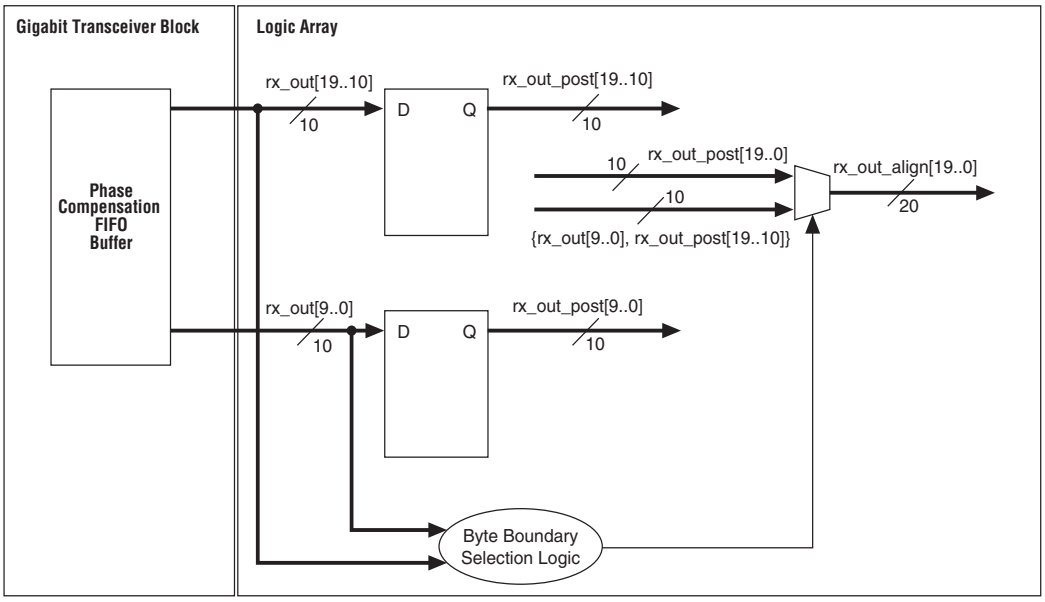
Figure 3–12 demonstrates the alternate case of the alignment pattern found in the LSB of the 20-bit output. Correspondingly, `rx_patterndetect[0]` goes high. In this case, the output is BA, DC, FE, and so on.

Figure 3–12. Receiver Byte Deserialzer in 10/20-Bit Mode With Alignment Pattern in LSB



You must implement logic for byte position alignment, if necessary, once data enters the logic array, as seen in Figure 3–13. In this example, the byte position selection logic determines the proper byte position based on the pattern detect signal.

Figure 3–13. Receiver Byte Deserialzer Data Recovery in Logic Array



Receiver Phase Compensation FIFO Buffer

The receiver phase compensation FIFO buffer is located at the FPGA logic array interface in the receiver block and is four words deep. The buffer compensates for the phase difference between the clock in the FPGA and the operating clocks in the transceiver block.

In Custom mode, the write port is clocked by the recovered clock from the CRU. This clock is half the original rate if the byte deserializer is used. The read clock is clocked by `rx_coreclk` or `rx_clkout`.

You can select `rx_coreclk` as an optional receiver input port. It can also accept a clock supply. The clock that feeds `rx_coreclk` must be derived from the `rx_clkout` port of its associated receiver channel. The receiver phase compensation FIFO buffer can only account for phase differences.

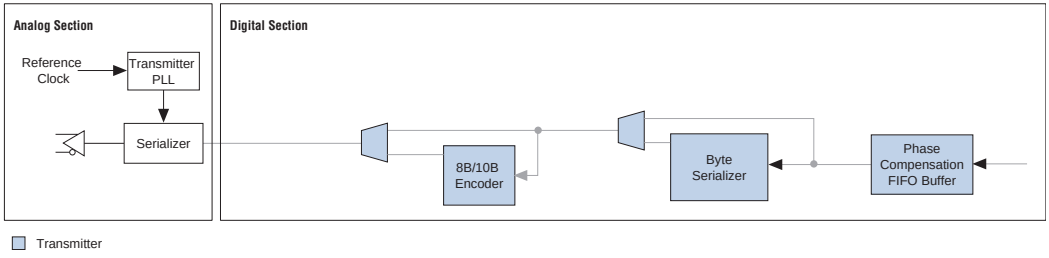
In Custom mode, if the rx_clkout port is not selected for use, the read clock is clocked by rx_coreclk, which is fed by rx_clkout. An FPGA global clock, regional clock, or fast regional clock resource is required to make the connection for the read clock. Refer to the section "Custom Mode Channel Clocking" on page 3–20 or the block diagram in the MegaWizard Plug-In Manager for more information on the clock structure in a particular mode.

The receiver phase compensation FIFO buffer is always used and cannot be bypassed.

Custom Mode Transmitter Architecture

Figure 3–14 shows the components of the transmitter block that are used in the Custom mode of operation.

Figure 3–14. Block diagram of the Transmitter Digital Components in Custom Mode



Transmitter Phase Compensation FIFO Buffer

The transmitter phase compensation FIFO buffer is located at the FPGA logic array interface in the transmitter block and is four words deep. The phase compensation FIFO buffer compensates for the phase difference between the clock in the FPGA and the operating clocks in the transceiver block.

The read port of the phase compensation FIFO module is clocked by the transmitter PLL clock. The write clock is clocked by TX_CORECLK. You can select the TX_CORECLK as an optional transmitter input port in which to supply a clock. In this case, you must ensure that there is no frequency difference between the TX_CORECLK and the Transmitter PLL clock. The transmitter phase compensation FIFO buffer can only account for phase differences.

If the TX_CORECLK is not selected as an optional input transmitter port, TX_CORECLK is fed by CORECLK_OUT. This connection occurs using the logic array routing. In this situation, the software defaults to using an FPGA global clock, a regional clock, or a fast regional clock resource.

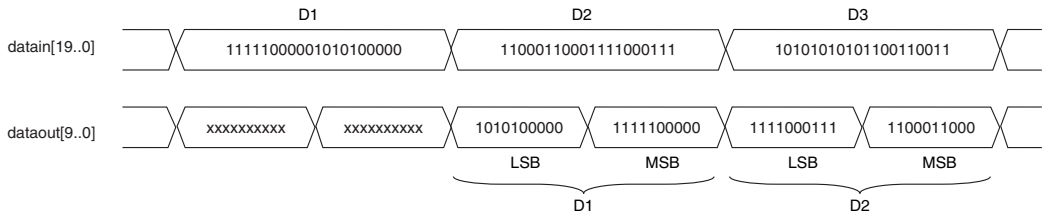
The transmitter phase compensation FIFO buffer is always used and cannot be bypassed. The input to the transmitter phase compensation FIFO module is the data from the device logic array. If they are used, the tx_ctrlenable and tx_forcedisparity signals are also passed through the FIFO module to ensure that they are synchronized with the data when they feed the 8B/10B encoder module.

Byte Serializer

The byte serializer in the transmitter block takes in a 20- or 16-bit input from the phase compensation FIFO module and serializes it to 10 or 8 bits. It transmits the least significant byte to the most significant byte. Altera® recommends using the transmitter digital reset to reset the byte serializer FIFO module pointers whenever an unknown state is encountered: for example, periods when the transmitter PLL unlocks. Refer to the *Reset Control & Power Down* chapter for further details on the reset sequence.

Figure 3–15 demonstrates input and output signals of the byte serializer when serializing a 20-bit input to 10 bits. The tx_in[] signal is the input from the FPGA logic array that has already passed through the transmitter phase compensation FIFO buffer.

Figure 3–15. Transmitter Byte Serializer in 20- to 10-Bit Mode



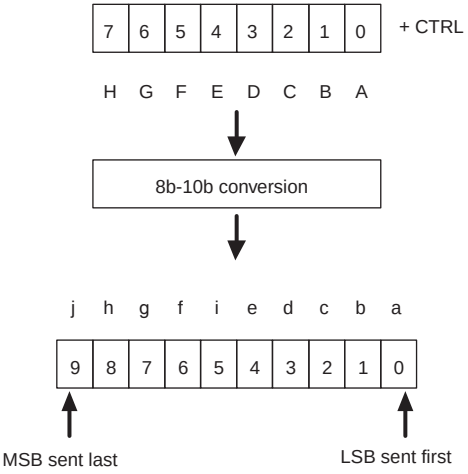
The LSB is transmitted before the MSB in the Transmitter byte serializer. For the input of D1, the output is D1LSB and then D1MSB.

8B/10B Encoder

The 8B/10B encoder is part of the Stratix GX transceiver block. The purpose of the 8B/10B encoder is to translate 8-bit data and a 1-bit control identifier (via tx_ctrlenable) into a 10-bit DC-balanced data stream.

For additional information regarding the 8B/10B code itself, refer to the *Data & Control Codes* chapter of the *Stratix GX Device Handbook, Volume 2*. The 8B/10B encoder translates the 8-bit data or 8-bit control character to its 10-bit equivalent. The conversion format is shown in Figure 3–16. The 10-bit resultant data is transmitted LSB first by the serializer.

Figure 3–16. 8B/10B Conversion Format



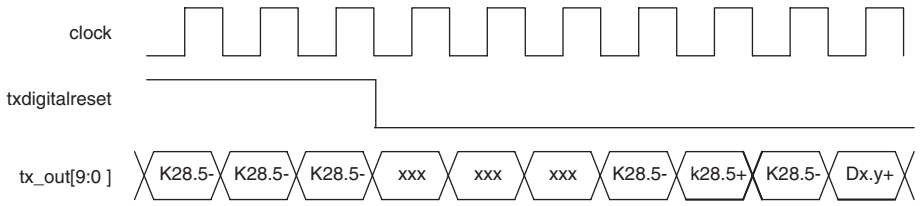
8B/10B Reset Condition

The `txdigitalreset` controls the reset of the 8B/10B encoder. To reset the 8B/10B encoder, `txdigitalreset` must be high. During reset, the running disparity registers are cleared, as are the data registers. Also, the 8B/10B encoder outputs a K28.5 pattern from the RD- column continuously until `txdigitalreset` is low. The `tx_in[]` and `tx_ctrlenable[]` are ignored during the reset state. Once out of reset, the 8B/10B encoder starts with a bias towards negative disparity (RD-) and transmits three K28.5 code for synchronizing before it starts encoding and transmitting the data on `tx_in[]`.

If the reset for the 8B/10B encoder is asserted, the 8B/10B decoder receiving the data might receive an invalid code error, sync error, control detect, and/or disparity error while `txdigitalreset` is high.

Figure 3–17 shows the reset behavior of the 8B/10B encoder. When in reset (txdigitalreset is high), a K28.5- (K28.5 10-bit code from the RD- column) is sent continuously until txdigitalreset is low. Because of the pipelining of the transmitter channel, there are some don't-care values (10'hxxx) until the first of three K28.5 is sent (Figure 3–17 shows three don't-cares). Normal user data follows the third K28.5.

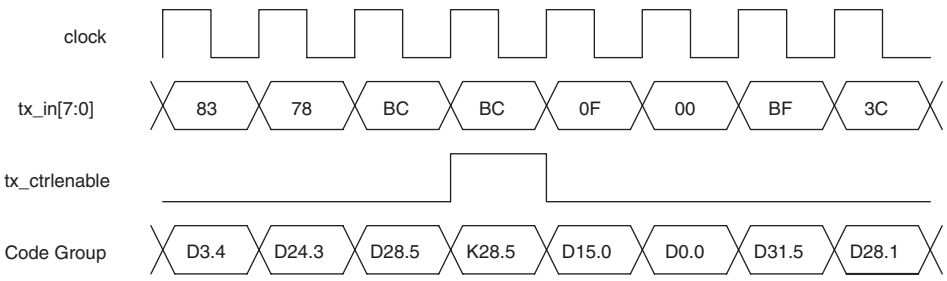
Figure 3–17. Transmitter Output During Reset Conditions



Control Code Encoding

The tx_ctrlenable[] controls when a control code is to be inserted in the encoded data flow. When tx_ctrlenable[] is low, the byte at tx_in[] is encoded as data. When tx_ctrlenable[] is high, tx_in[] is encoded as a control word. Figure 3–18 shows that the second 0xBC is encoded as a control code. The others are encoded as data.

Figure 3–18. Control Word Identification Waveform



The 8B/10B encoder does not check whether the control code word entered is one of the 12 valid control code-groups. If an invalid control code is entered, the resulting 10-bit code might also be invalid (might not map to a valid Dx.y or Kx.y code), depending on the value entered.

An example would be the invalid code encoding of a K24.1 (`data = 8'h38 + tx_ctrlenable = 1'b1`). Depending on the current running disparity, the K24.1 can be encoded to be 10'b0110001100 (0x18C), which is equivalent to a D24.6+ (0xD8 from the RD+ column). An 8B/10B decoder would decode this value incorrectly.

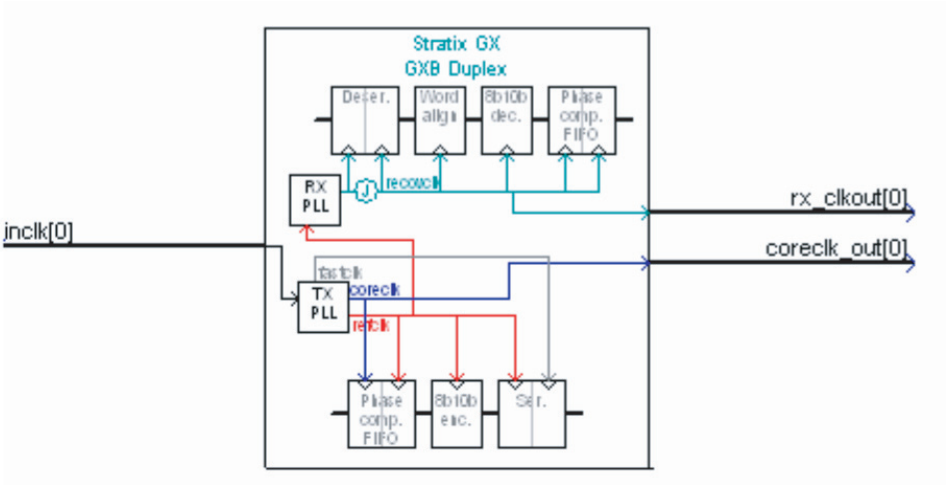
Custom Mode Clocking

Two types of clocking are available in Custom mode: channel clocking and inter-transceiver clocking.

Custom Mode Channel Clocking

This section describes internal clocking and the external clocks of the transceiver in Custom mode. By default, the MegaWizard Plug-In Manager parameterizes the `altgxb` megafunction with the clock configuration shown in Figure 3–19.

Figure 3–19. Default Configuration of the altgxb Megafunction in Custom Mode



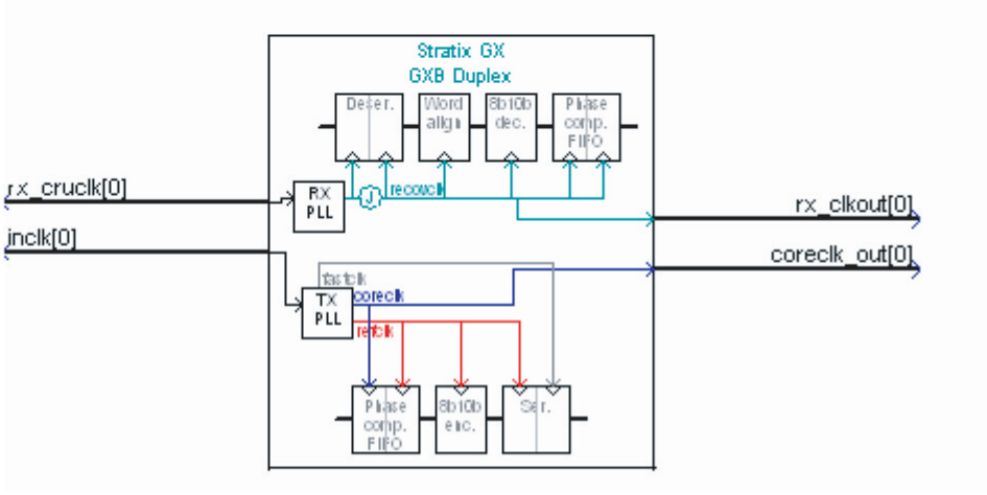
The `altgxb` megafunction (shown in Custom mode in Figure 3–19) is configured such that the train receiver PLL with transmitter PLL is enabled. The transmitter PLL is fed from an `inclk` port, which itself can be fed from a dedicated `REFCLKB`, nondedicated `REFCLKB` through an IQ line, global clock, regional clock, or fast regional clock source. The receiver logic is clocked by the recovered clock from the CRU, which is `rx_clkout`. This recovered clock is also fed into the device so that in a multi-crystal environment, some level of clock domain decoupling can be implemented to interface with a system clock. On the transmitter channel

the output of the transmitter PLL, coreclk_out, is sent into the logic array and also loops back to clock the write side of the transmit phase compensation FIFO buffer.

You can disable the trained receiver PLL CRU clock from the transmitter PLL feature in the MegaWizard Plug-In Manager. Deselecting this option adds an additional RX_CRUCLK input reference clock port for the receiver PLL. This feature supports additional multiplication factors for the receiver PLL and also supports the separation of receiver and transmitter reference clocks. This separation is required if the output reference clock frequency from the transmitter PLL exceeds the 325-MHz phase frequency detector of the receiver PLL. For more information on this feature, refer to the *Stratix GX Analog Description* chapter of the *Stratix GX Device Handbook, Volume 2*. This configuration is shown in Figure 3–20.

If double width is used (16-bit bus) and the data rate is above 2,600 Mbps, the train receiver PLL clock from the transmitter PLL must be turned off, because the output clock from the transmitter PLL exceeds the 325-MHz limit on the receiver PLL input clock, if the input clock is fed by any non-REFCLKB pins. REFCLKB input pins have a 650-MHz limit.

Figure 3–20. altgxb Megafunction in Custom Mode with the Train Receiver CRU From Transmitter PLL Disabled



This configuration has an independent rx_cruclk that feeds the receiver PLL reference clock. This input clock port is only available when the receiver PLL is not trained by the transmitter PLL. There is one rx_cruclk associated with a channel. If four channels are active, there are four rx_cruclk signals.

The rx_clkout is the recovered clock from the associated receiver channel. One rx_clkout is available for each receiver channel that is used. You can use this clock to clock the rate-matching FIFO buffer write port in the device. The read port of the FIFO buffer can be clocked by the coreclk_out signal or device clock.

The coreclk_out port is the output from the transmitter PLL. A coreclk_out port is available for each transceiver block used. You should use the coreclk_out clock to clock the transmitter input.

The receiver phase compensation FIFO buffer read clock and the transmitter phase compensation FIFO buffer write clock can be optionally enabled to manually feed in a clock from the device buffer write block. You can use these options to optimize the global clock usage. For example, if all transmitter channels between transceiver blocks are from a common clock domain, the transceiver instantiations use only one global resource clock instead of one global per transceiver block, if the tx_coreclk option is disabled.

The situation is similar for the receiver channels in a single-crystal synchronous system with rx_coreclk. During initialization or long run lengths, the recovered clock becomes asynchronous with the system clock. As a result, the pointers of the receiver phase compensation FIFO buffer might overlap and fail to function correctly. In these situations, the receiver phase compensation FIFO buffers must be reset by the rxdigitalreset signal.

In multi-crystal environments, individual recovered clocks must drive the read clock of the phase compensation FIFO buffer. The Quartus® II software does so by default and you do not need to manually make this connection. The rx_coreclk and tx_coreclk must be frequency matched with their respective read and write ports. Figure 3–21 shows the clock configuration with these optional input ports enabled.

Figure 3–21. altgxb in Custom Mode With rx_coreclk & tx_coreclk Enabled

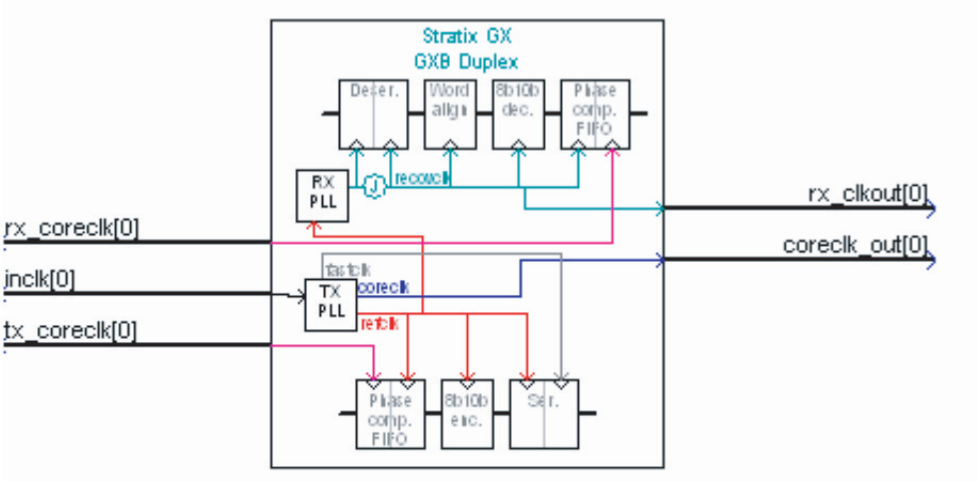


Table 3–3 displays a list of the input and output clock ports available in Custom mode.

Table 3–3. Input & Output Ports Available in Custom Mode (Part 1 of 2)

Clock	Port	Description
rx_cruclk	Input	Input to CRU available as a port when CRU is not trained by the transmitter PLL.
inclk	Input	Input to transmitter PLL available as a port when the transmitter PLL is instantiated.
coreclk_out	Output	Output clock from transmitter PLL equivalent to tx_pll_clk. Available as port if transmitter PLL is used.
rx_clkout	Output	Output clock from transceiver. In this mode, rx_clkout is the recovered clock of the respective channel.

Table 3–3. Input & Output Ports Available in Custom Mode (Part 2 of 2)

Clock	Port	Description
tx_coreclk	Input	Clocks write port of transmitter phase compensation FIFO module. Available as an optional port in the Quartus II MegaWizard® Plug-In Manager. Must be frequency matched to tx_pll_clk. If not available as a port, this is fed by coreclk_out through logic array routing.
rx_coreclk	Input	Clocks read port of Receiver phase compensation FIFO module. Available as optional port in the Quartus II MegaWizard Plug-In Manager. If not available as a port, this is fed by rx_clkout through logic array routing.

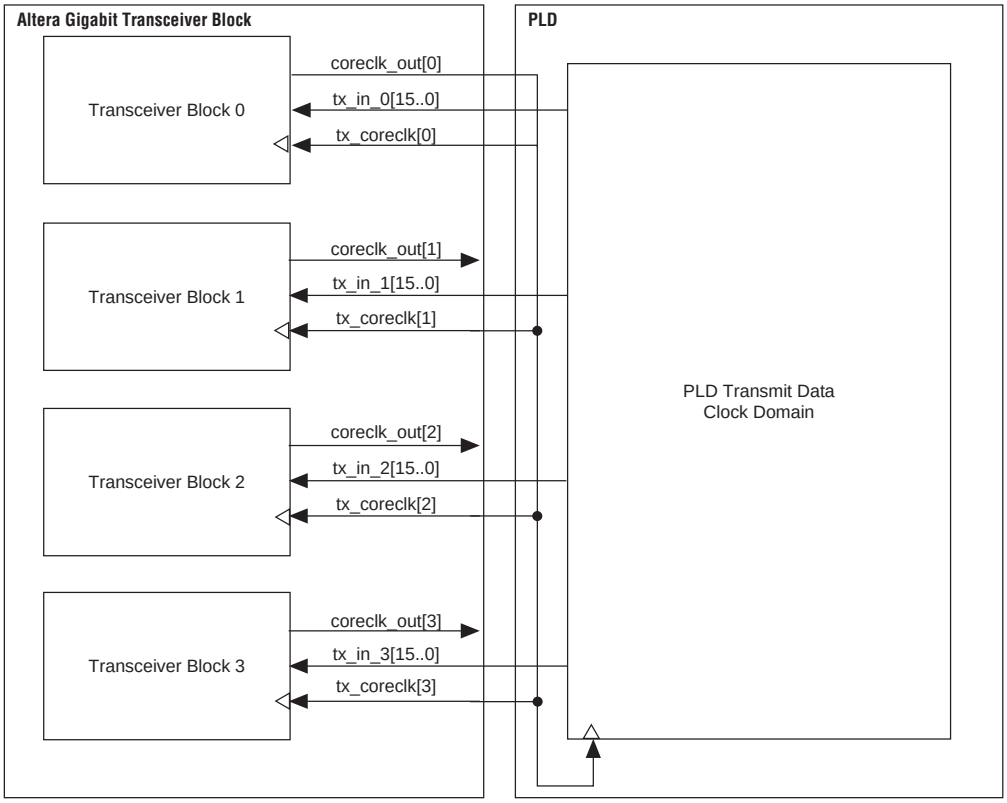
Custom Mode Inter-Transceiver Block Clocking

This section describes guidelines for using transceiver interface clocking between the device logic array and transceiver channels when multiple transceiver blocks are active. Depending on the mode supported by the Stratix GX devices, each transceiver block has a different transceiver-to-device-interface clocking. Different input and output clocks are available based on the options provided by the MegaWizard Plug-In Manager's built-in functions. Support for the number of channels offered varies depending on which Stratix GX device is selected. Because of the various configurations of input and output clocks, you must carefully consider the clocking schemes between transceiver blocks to prevent pitfalls later in the design cycle.

One of the clocking interfaces to consider while designing with Stratix GX devices is the transceiver-to-FPGA interface. This clocking scheme is further classified as the FPGA to transmitter channel and the receiver channel to the FPGA.

In Custom mode, the write port of the transmitter phase compensation FIFO module is either clocked by the CORECLK_OUT or the TX_CORECLK signal. The constraint on using TX_CORECLK is that the clock must be frequency locked to the read clock of the transmitter phase compensation FIFO module. Synchronous data transfers for a multi-transceiver block configuration can be accomplished by using the TX_CORECLK port. The TX_CORECLK of multi-transceiver blocks is connected to a common clock domain either from a single CORECLK_OUT signal or from a device system clock domain. This scheme is shown in Figure 3–22.

Figure 3–22. Example of a Multi-Transceiver Block Device to Transmitter Interface Clocking Scheme in Custom Mode



When TX_CORECLK is not enabled, the Quartus II software automatically routes the CORECLK_OUT signal to the write clock of the phase compensation FIFO module using a global, regional, or fast regional resource. In a multi transceiver block configuration, this routing can lead to timing violations because the coreclk_out per transceiver block cannot guarantee phase relationship. Therefore, the TX_CORECLK with a common clock is recommended for synchronous transmission.

Another inter-transceiver block consideration is the selection of the dedicated REFCLKB pin. Stratix GX channels are arranged in banks of four, or transceiver blocks. Each transceiver block is able to share a common reference clock through the inter-transceiver lines. You can reduce the Stratix GX logic array clock usage by using the inter-transceiver lines. The inter-transceiver lines are used when a REFCLKB input port from one transceiver block or channel drives any other transceiver blocks or channels. The Quartus II software automatically determines the inter-transceiver line usage.

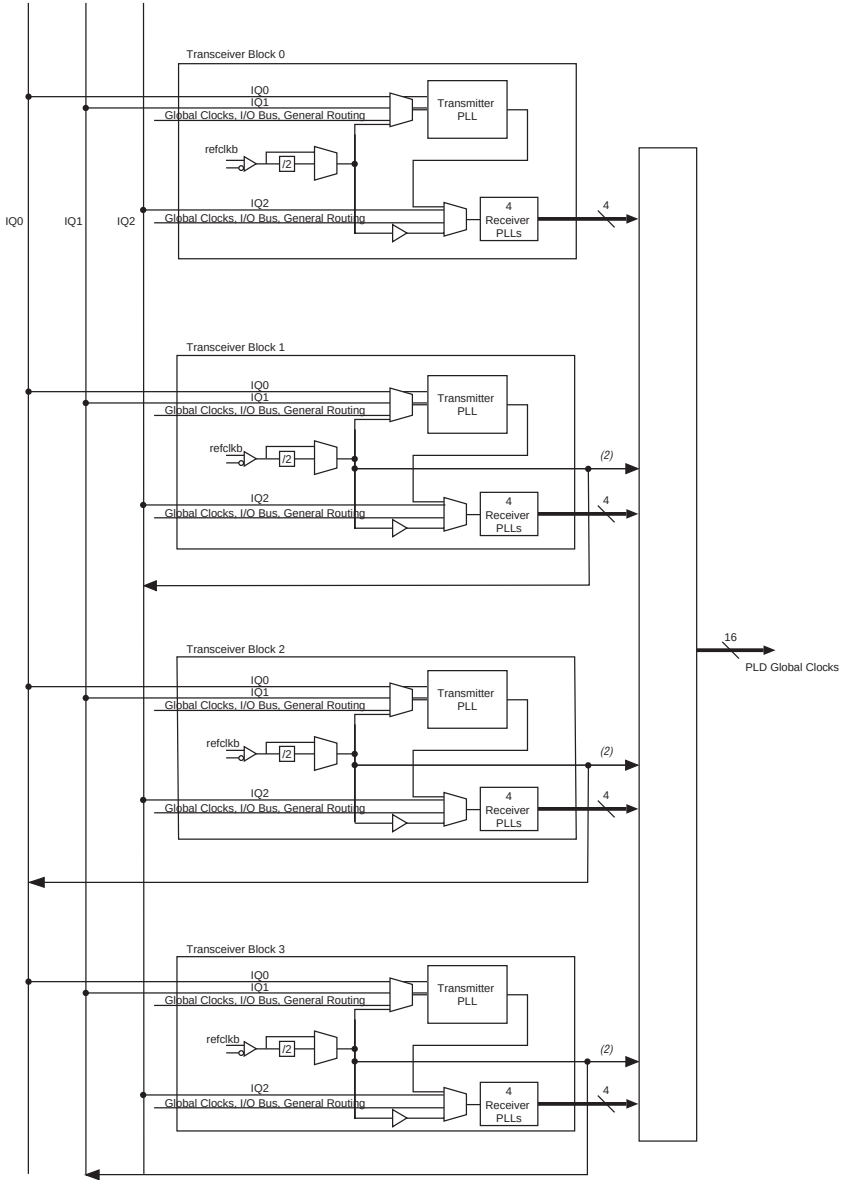
When determining the location of REFCLKB pins, you should consider what is fed by the chosen pin. Table 3–4 shows the available inter-transceiver lines, along with the transceiver block that drives them. This information is based on the number of transceiver channels in the Stratix GX device.

Table 3–4. REFCLKB to IQ Line Connections

Channel Density	REFCLKB in Transceiver Block Number	Channels in Transceiver Block	IQ Line Driven by REFCLKB
8 channels (EP1SGX10)	0	[3:0]	IQ2
	1	[7:4]	IQ0
16 channels (EP1SGX25)	0	[3:0]	N/A
	1	[7:4]	IQ2
	2	[11:8]	IQ0
	3	[15:12]	IQ1
20 channels (EP1SGX40)	0	[3:0]	N/A
	1	[7:4]	IQ2
	2	[11:8]	IQ0
	3	[15:12]	IQ1
	4	[19:16	N/A

Figure 3–23 shows the transceiver routing with respect to inter-transceiver lines for the EP1SGX25 device. It is important to use this information when placing REFCLKB pins. For example, if a REFCLKB pin is used and is required to feed a transmitter PLL using an inter-transceiver line, the REFCLKB pin cannot be in transceiver block 1, because IQ2 feeds only the receiver PLLs.

Figure 3–23. Inter-Transceiver Line Connections for EP1SGX25 Device *Note (1)*



Note to Figure 3–23:

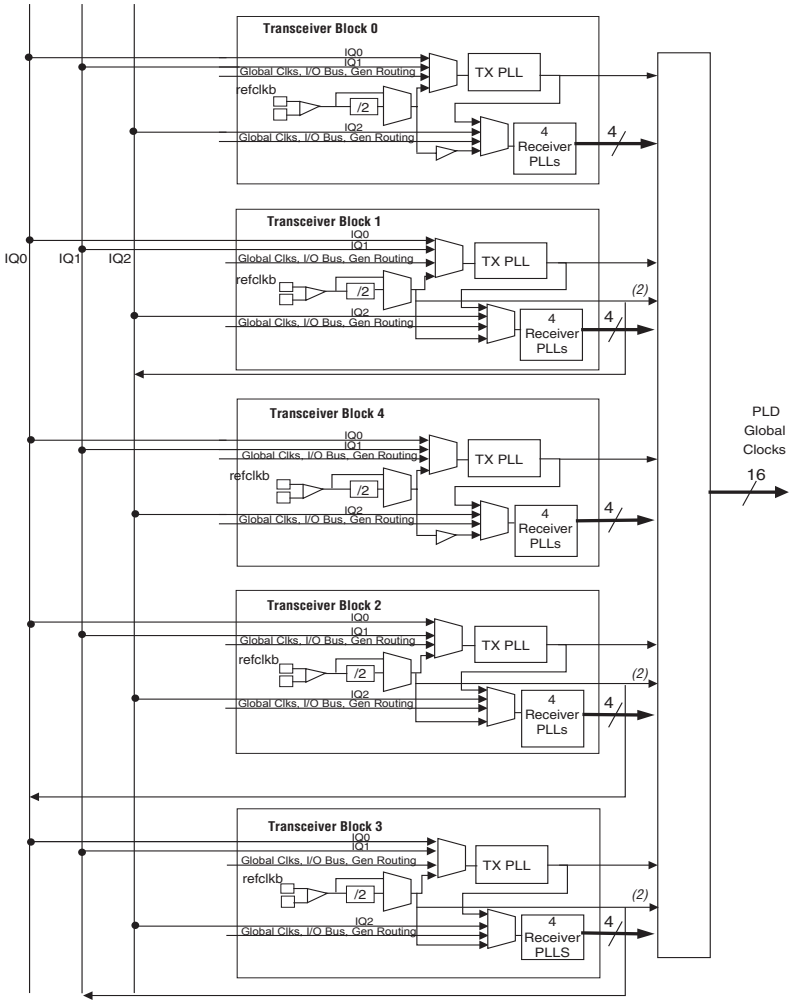
(1) IQ lines are inter-transceiver block lines.

(2) If the /2 pre-divider is used, the path to drive the PLD logic array, local, or global clocks is not allowed.

(3) There are four receiver PLLs in each transceiver block.

Figure 3–24 shows the transceiver routing with respect to inter-transceiver lines for the EP1SGX40G Device. This device has an extra transceiver block (4), which is in the middle of the row of transceiver blocks. This information is important when placing REFCLKB pins. For example, if a REFCLKB pin must feed a transmitter PLL using an inter-transceiver line, the REFCLKB pin cannot be in transceiver block 1, because IQ2 feeds only the receiver PLLs. For connecting to the transmitter PLL, choose REFCLKB in transceiver block 2 and transceiver block 3.

Figure 3–24. Inter-Transceiver Line Connections for the EP1SGX40G Device *Note (1)*



Note to Figure 3–24:

(1) IQ lines are inter-transceiver block lines.

(2) If the /2 pre-divider is used, the path to drive the PLD logic array, local, or global clocks is not allowed.

(3) There are four receiver PLLs in each transceiver block.

Custom Mode MegaWizard Plug-In Manager

Altera recommends that the Stratix GX transceiver block be instantiated and parameterized through the `altgxb` megafunction in the MegaWizard Plug-In Manager. The MegaWizard Plug-In Manager offers a graphical user interface (GUI) that organizes the `altgxb` options into

easy-to-use sections. The wizard also sets the proper ports and parameters automatically, based on the options and parameters you select. Invalid settings are automatically flagged in the wizard to help prevent illegal configurations. The MegaWizard Plug-In Manager does not provide access to any options that do not apply to Custom mode.

Custom Mode MegaWizard Plug-In Manager Considerations

Each `altgxb` megafunction instantiation uses one or more transceiver blocks, based on the number of channels that you select. There are four channels per transceiver block. If a MegaWizard Plug-In Manager instantiation uses fewer than four channels, the remaining channels in that transceiver block are not available for use.

Each MegaWizard Plug-In Manager instantiation must have similar functionality and data rates. If transceiver blocks that differ in functionality and/or data rates are required, you can create separate instantiations for each transceiver block.

As mentioned in the section "Custom Mode Clocking" on page 3–20, the MegaWizard Plug-In Manager displays the configuration of the `altgxb` megafunction, as shown in Figure 3–19 on page 3–20. This diagram changes dynamically based on the selected mode, options, and clocking schemes.

Custom Mode altgxb MegaWizard Plug-In Manager Options

This section shows the MegaWizard Plug-In Manager pages where you select the options for a Custom mode configuration.

Figure 3–25 shows page 3 of the `altgxb` MegaWizard Plug-In Manager in Custom mode.

Figure 3–25. MegaWizard Plug-In Manager - altgxb (Page 3)

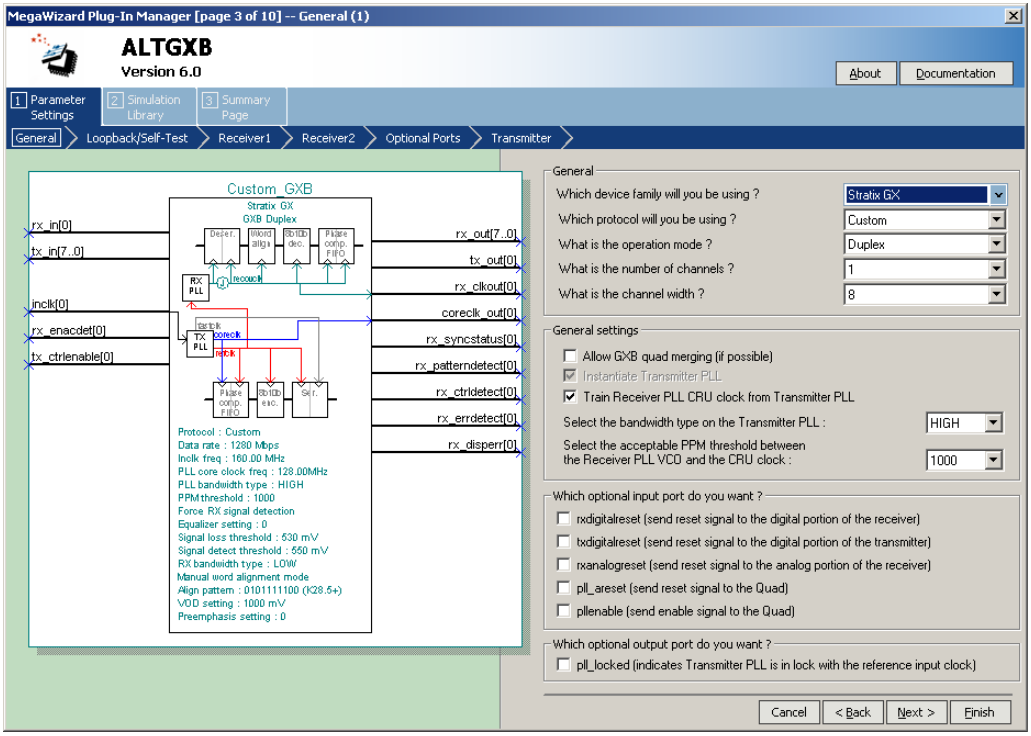


Table 3–5 describes the available options on page 3 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 3–5. MegaWizard Plug-In Manager Options (Page 3 for Custom Mode) (Part 1 of 2)

altgxb Setting	Description
Which device family will you be using?	Stratix GX is the only option available.
Which protocol will you be using?	For the Custom mode, you must select the Custom protocol.
What is the operation mode?	Custom protocol mode supports duplex, receiver- only, or transmitter-only operation modes.
What is the number of channels?	This value can be from 1 to the maximum number of channels available on the device.

Table 3–5. MegaWizard Plug-In Manager Options (Page 3 for Custom Mode) (Part 2 of 2)

altgxb Setting	Description
What is the channel width?	The correct channel width setting depends on whether you are using 8B/10B decoding. With 8B/10B, 8 bits is single width and 16 bits is double width. Without 8B/10B, 8 bits is single width, 16 bits is double width, 10 bits is single width, and 20 bits is double width.
Add GXB quad merging (if possible)	For information about this option, refer to the section "Stratix GX Transceiver Merging" on page 3–43.
Instantiate Transmitter PLL	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Train Receiver PLL CRU clock from Transmitter PLL	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the bandwidth type on the Transmitter PLL	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the acceptable PPM threshold between the Receiver PLL VCO and the CRU clock	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
`rxdigitalreset` (send reset signal to the digital portion of the receiver)	The `rxdigitalreset` port resets the digital blocks in the receiver channel. Each active receiver channel has its own digital reset.
`txdigitalreset` (send reset signal to the digital portion of the transmitter)	The `txdigitalreset` port resets the digital blocks of the transmitter channel. Each active transmitter channel has its own digital reset.
`rxanalogreset` (send reset signal to the analog portion of the receiver)	The `rxanalogreset` port resets the receiver's analog circuits, including the receiver PLL. Each active receiver channel has its own analog reset.
`pll_areset` (send reset signal to the Quad)	The `pll_areset` port resets the entire transceiver block (all receiver and transmitter digital and analog circuits, including receiver and transmitter PLLs).
`pllenable` (send enable signal to the Quad)	The `pllenable` port enables the entire transceiver block; if deasserted, the entire transceiver block is held in the reset condition.
`pll_locked` (indicates Transmitter PLL is in lock with the reference input clock)	For more information, refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 3–26 shows page 4 of the `altgxb` MegaWizard Plug-In Manager in Custom mode.

Figure 3–26. MegaWizard Plug-In Manager - altgxb (Page 4)

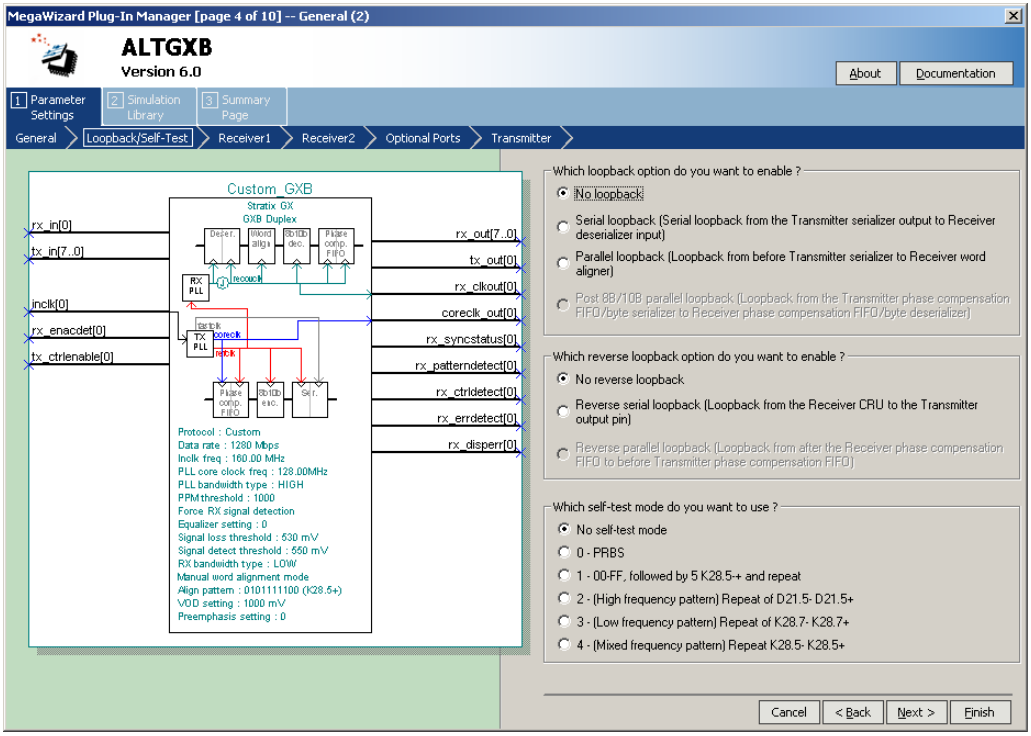


Table 3–6 describes the available options on page 4 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 3–6. MegaWizard Plug-In Manager Options (Page 4 for Custom Mode)

altgxb Setting	Description
Which loopback option do you want to enable?	For more information, refer to the *Loopback Modes* chapter in volume 2 of the *Stratix GX Device Handbook*.
Which reverse loopback option do you want to enable?	For more information, refer to the *Loopback Modes* chapter in volume 2 of the *Stratix GX Device Handbook*.
Which self-test mode do you want to use?	For more information, refer to the *Stratix GX Built-In Self Test (BIST)* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 3–27 shows page 5 of the `altgxb` MegaWizard Plug-In Manager in Custom mode.

Figure 3–27. MegaWizard Plug-In Manager - altgxb (Page 5)

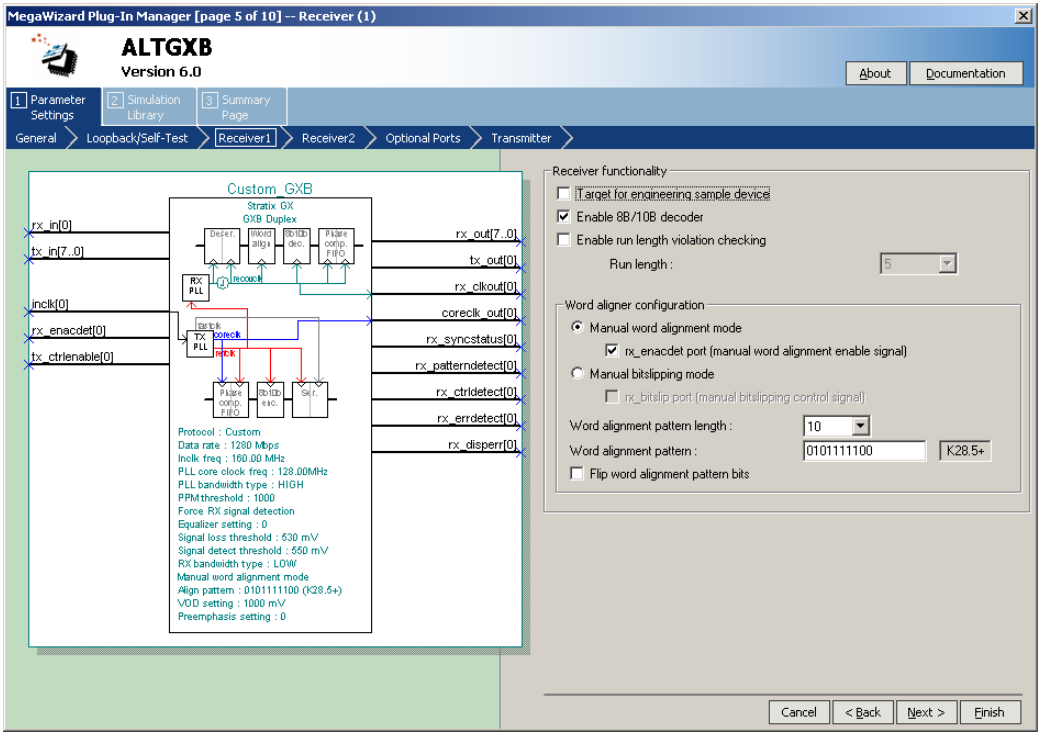


Table 3–7 describes the available options on page 5 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 3–7. MegaWizard Plug-In Manager Options (Page 5 for Custom Mode) (Part 1 of 2)

altgxb Setting	Description
Target for engineering sample device	You must select this option if the design is targeted for an engineering sample (ES) device.
Enable 8B/10B decoder	You can enable or disable 8B/10B. With 8B/10B active, the data width must be 8-bits or 16-bits.
Enable run-length violation checking	Select this option to enable the run-length violation circuitry. When enabled, the run-length violation status is provided on the `rx_rlv` signal.
Manual word alignment mode	Use this option to configure the word aligner in manual alignment mode. Refer to the "Manual Alignment Modes" on page 3–5 section for more information.

Table 3–7. MegaWizard Plug-In Manager Options (Page 5 for Custom Mode) (Part 2 of 2)

altgxb Setting	Description
rx_enacdet port (manual word alignment enable signal)	The rx_enacdet port supports the word aligner to byte align to the word alignment pattern. When rx_enacdet is held high and the comma is detected, the word aligner aligns to the byte boundary. If this option is not turned on, the word aligner is not active, but the pattern detect signal is still functional. Refer to the section "Word Aligner" on page 3–2 for further details.
Manual bitslipping mode	Manual bit-slipping mode lets you control the word aligner's shift register directly via the rx_bitslip port.
rx_bitslip port (manual bitslipping control signal)	A low to high transition on the rx_bitslip port enables the word aligner's shift register to slip one bit. For example, if a 3-bit shift is required to align the incoming byte, rx_bitslip must be toggled low, high, low, high, low, high. The rx_bitslip port can be left in the high or low position after the above sequence.
Word alignment pattern length	Set the word alignment pattern length to 16-bits if using an 8- or 16-bit data bus size with 8B/10B turned off. With 8B/10B turned on and a data width of 8 or 16 bits, set the pattern size to 7 or 10 bits. The 7-bit mode is for the pattern detect module. Word alignment is still done on the 10-bit pattern, even in a 7-bit mode.
Word alignment pattern	Enter the word alignment pattern here in binary format. The number of bits in the pattern must be equal to the word alignment pattern length selected.
Flip word alignment pattern bits	Flips the word alignment bit order. If this option is turned on, the right-most bit is the MSB, otherwise the right-most bit is the LSB. This option is used in conjunction with the receiver and transmitter bit-flip options to ensure that the MSB is transmitted and received first in the serial stream.

Figure 3–28 shows page 6 of the altgxb MegaWizard Plug-In Manager in Custom mode.

Figure 3–28. MegaWizard Plug-In Manager - altgxb (Page 6)

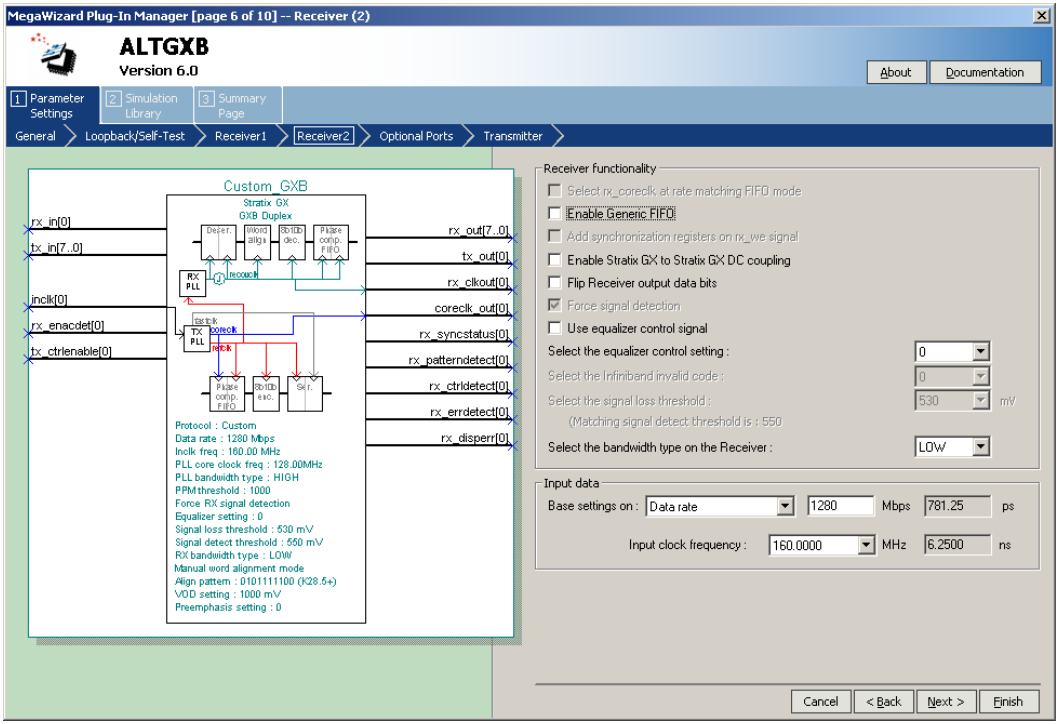


Table 3–8 describes the available options on page 6 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 3–8. MegaWizard Plug-In Manager Options (Page 6 for Custom Mode) (Part 1 of 2)

altgxb Setting	Description
Select `rx_coreclk` at rate matching FIFO mode	This option is not available in Custom mode.
Enable Generic FIFO	Select this option to include a generic FIFO in the receiver data path between the word aligner and the 8B/10B decoder (if enabled). This FIFO can be used to decouple between the recovered clock and the local `rx_coreclk`.
Enable Stratix GX to Stratix GX DC coupling	You must enable this option if a Stratix GX transmitter is DC coupled to a Stratix GX receiver. This option biases the receiver buffer appropriately to inter-operate with a DC-coupled Stratix GX transmitter.

Table 3–8. MegaWizard Plug-In Manager Options (Page 6 for Custom Mode) (Part 2 of 2)

altgxb Setting	Description
Flip Receiver output data bits	This option flips the received data bit order on the `rx_out` port. This option is used in conjunction with the transmitter and the word alignment bit-flip options to ensure that MSB is transmitted and received first in the serial steam.
Force signal detection	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Use equalizer control signal	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the equalizer control setting	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the Infiniband invalid code	This option is not available in Custom mode.
Select the signal loss threshold	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the bandwidth type on the Receiver	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Base settings on	Data rate versus input clock frequency must adhere to the set multiplication factors of 2, 4, 5, 8, 10, 16, and 20 of the input clock. Multiplication factors of 2, 4, and 5 must use the dedicated `refclkb` pins. The multiplication factor of 2 also requires that the receiver PLL be trained by the transmitter PLL.

Figure 3–29 shows page 7 of the `altgxb` MegaWizard Plug-In Manager in Custom mode.

Figure 3–29. MegaWizard Plug-In Manager - altgxb (Page 7)

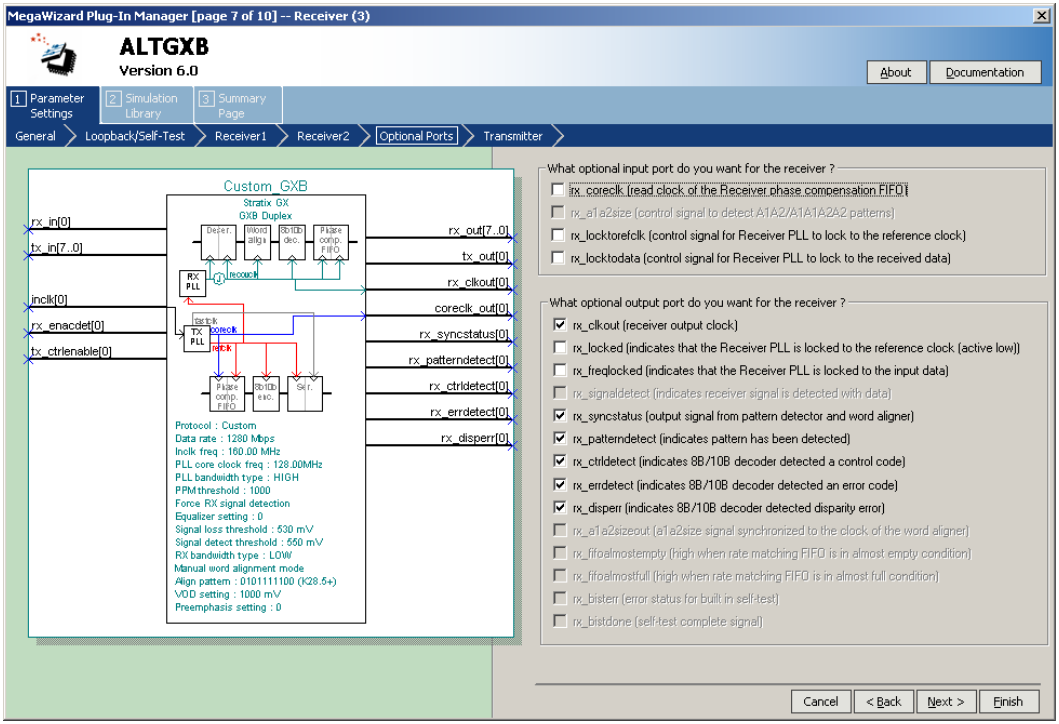


Table 3–9 describes the available options on page 7 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 3–9. MegaWizard Plug-In Manager Options (Page 7 for Custom Mode) (Part 1 of 2)

altgxb Setting	Description
`rx_coreclk` (read clock of the Receiver phase compensation FIFO)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
`rx_a1a2size` (control logic signal to detect A1A2/A1A1A2A2 patterns)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
`rx_locktorefclk` (control signal for Receiver PLL to lock to the reference clock)	You can force the receiver PLL to lock to the reference clock by setting this signal in manual lock mode. For more information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.

Table 3–9. MegaWizard Plug-In Manager Options (Page 7 for Custom Mode) (Part 2 of 2)

altgxb Setting	Description
rx_locktodata (control signal for Receiver PLL to lock to the received data)	Set this signal in manual lock mode to force the receiver PLL to lock to the serial data stream. For more information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_clkout (receiver input clock)	The rx_clkout signal is a recovered clock output from individual receiver channels. One rx_clkout signal is available per channel.
rx_locked (indicates that the Receiver PLL is locked to the reference clock (active low))	The rx_locked signal is an active low signal that indicates that the receiver PLL is phase locked to the reference clock. In data mode, this signal might be deasserted because the phase is being locked to the data and not the reference clock.
rx_freqlocked (indicates that the Receiver PLL is locked to the input data)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_signaldetect (indicates receiver signal is detected with data)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_syncstatus (output signal from pattern detector and word aligner)	The rx_syncstatus signal indicates the status of the word aligner. Refer to the section "Word Aligner" on page 3–2 for more information.
rx_patterndetect (indicates pattern has been detected)	The rx_patterndetect signal is an active high signal that signifies that the comma appears in the current byte boundary of the incoming data stream.
rx_ctrldetect (indicates 8B/10B decoder detected a control code)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_errdetect (indicates 8B/10B decoder detected an error code	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_disperr (indicates 8B/10B decoder detected disparity error)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_a1a2sizeout (a1a2size signal synchronized to the clock of the word aligner)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_fifoalmostempty (high when rate matching FIFO is in almost empty condition)	If Generic FIFO is enabled to perform rate matching between the recovered clock and local receiver clock, this signal indicates an almost empty condition when driven HIGH.
rx_fifoalmostfull (high when rate matching FIFO is in almost full condition)	If Generic FIFO is enabled to perform rate matching between the recovered clock and local receiver clock, this signal indicates an almost full condition when driven HIGH.
rx_bisterr (error status for built-in self-test)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_bistdone (self-test complete signal)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 3–30 shows page 8 of the altgxb MegaWizard Plug-In Manager in Custom mode.

Figure 3–30. MegaWizard Plug-In Manager - altgxb (Page 8)

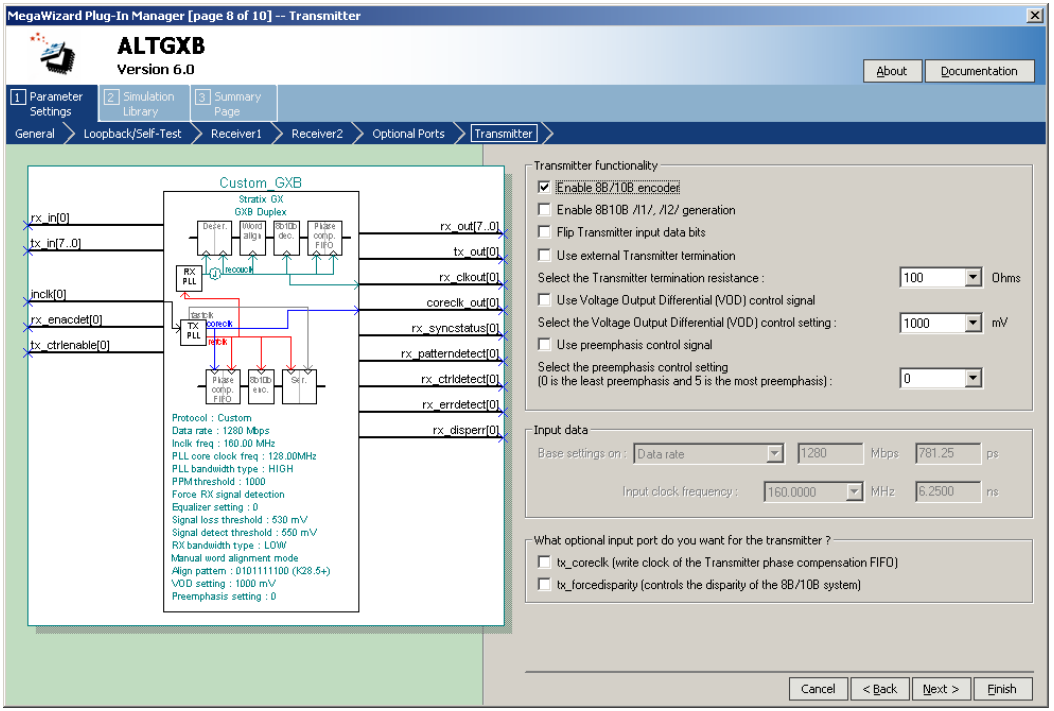


Table 3–10 describes the available options on page 8 of the MegaWizard Plug-In Manager for your altgxb custom megafunction variation.

Table 3–10. MegaWizard Plug-In Manager Options (Page 8 for Custom Mode) (Part 1 of 2)

altgxb Setting	Description
Enable 8B/10B encoder	Use this option to enable or disable the 8B/10B encoder in the transmitter data path.
Enable 8B/10B /I1/, /I2/ generation	This option enables the transmitter to replace any /Dx.y/ following a /K28.5/ with either /D5.6/ or /D16.2/, depending on the running disparity before /K28.5/. Refer to the information on idle generation in the chapter *GIGE Mode* in volume 2 of the *Stratix GX Device Handbook*.

Table 3–10. MegaWizard Plug-In Manager Options (Page 8 for Custom Mode) (Part 2 of 2)

altgxb Setting	Description
Flip Transmitter input data bits	This option flips the transmitter data bit order on the PLD interface. This option is used in conjunction with the receiver and the word alignment bit-flip options to ensure that MSB is transmitted and received first in the serial steam.
Use external Transmitter termination	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Use Voltage Output Differential (VOD) control signal	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the Voltage Output Differential (VOD) control setting	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Use preemphasis control signal	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the preemphasis control setting (0 is the least preemphasis and 5 is the most preemphasis)	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
`tx_coreclk` (write clock of the Transmitter phase compensation FIFO buffer	Enable this option to select `tx_coreclk` to clock the write side of the transmitter phase compensation FIFO. If enabled, you must ensure that there is no ppm difference between the `tx_coreclk` and the transmitter PLL input clock.
`tx_forcedisparity` (controls the disparity of the 8B/10B system)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 3–31 shows page 9, the Simulation Libraries page, of the MegaWizard Plug-In Manager for the Custom protocol set up.

Figure 3–31. MegaWizard Plug-In Manager - altgxb (Page 9)

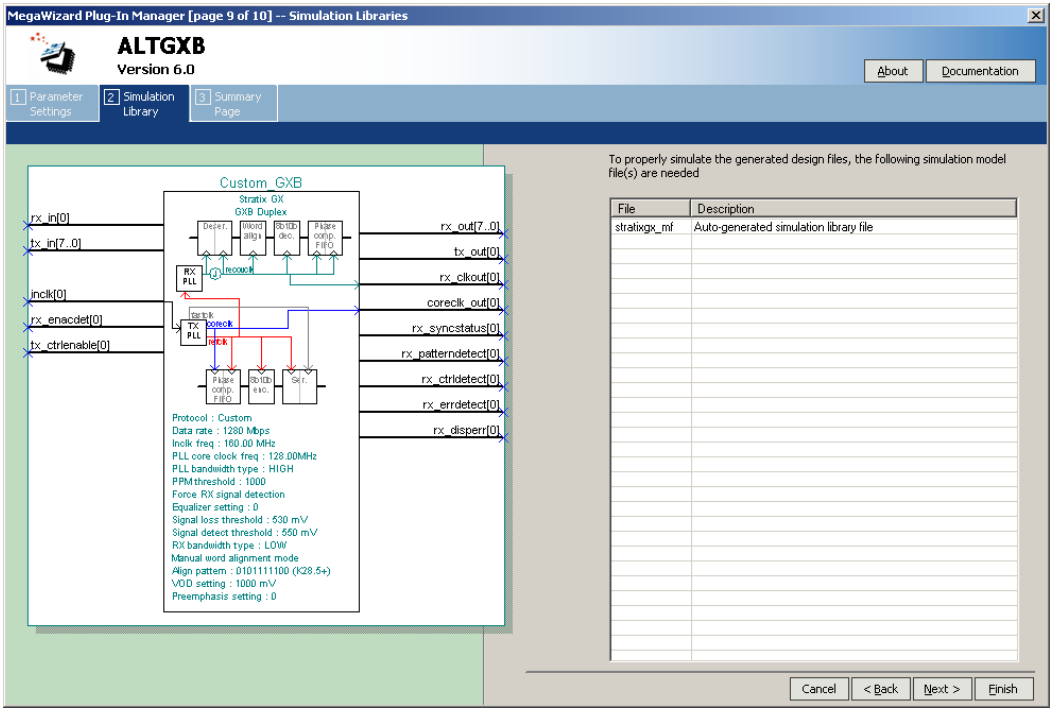
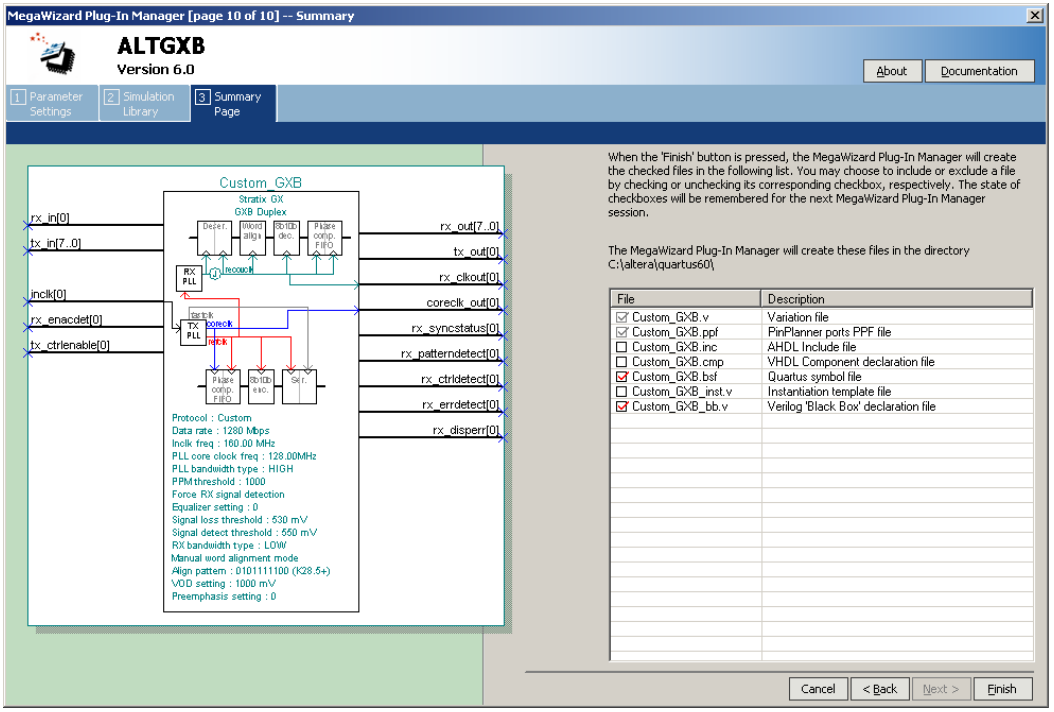


Figure 3–32 shows page 10 of the MegaWizard Plug-In Manager for the Custom protocol set up. You can select optional files on this page. After you make your selections, click **Finish** to generate the files.

Figure 3–32. MegaWizard Plug-In Manager - altgxb (Page 10)



Stratix GX Transceiver Merging

A transceiver block contains four transceivers. In a design, an `altgxb` instantiation is placed in one or more transceiver blocks and potentially leaves unused transceivers in a block. For example, a six transceiver instantiation completely fills one transceiver block and half fills a second, taking up two full transceiver blocks. If another instantiation is in the design, it is placed the same way. For example, an instantiation of two transmitters takes up a third transceiver block. Merging two of the partially filled transceiver blocks into one transceiver block reduces the resources used and allows a design to fit into a device with fewer transceiver blocks.

The `altgxb` MegaWizard Plug-In Manager in the Quartus II software has a feature that allows merging of similar quads (transceiver blocks). With a few exceptions, transceiver blocks can be merged if the options

chosen in the wizard are the same. The Quartus II software merges the transceiver blocks automatically if you turn on the merging option for each instantiation. Table 3–11 shows the considerations for merging.

Table 3–11. Stratix GX Merging Considerations

Merging Rules	Required to Match Between Transceiver Blocks	Not Required to Match Between Transceiver Blocks

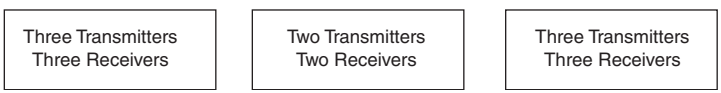
MegaWizard Plug-In Manager options	`USE_8B_10_MODE`
	`USE_DOUBLE_DATA_MODE`
	`CHANNEL_WIDTH`
	`SYNC_MODE`
	`DATA_RATE`
	`TRANSMIT_PROTOCOL`
		VOD
		Pre-emphasis
		Equalization
		Number of transmitters
		Number of receivers

Input control signals must be shared across transceiver blocks and must be from the same source	Clock
	`CRU_CLOCK`
	`PLL_RESET`
	`PLL_ENABLE`	

The merging partial transceiver blocks must reduce the number of transceiver block when merged	The total number of receivers and transmitters must be tx ≤4 x *n* and rx ≤4 x *n*, where *n* is the reduced number of transceiver blocks remaining after merging.	

The Quartus II software does not merge two transceiver blocks if they won't completely fit into one. It can, however, merge three transceiver blocks into two. Figure 3–33 shows a configuration with three transceiver blocks that can potentially be merged.

Figure 3–33. Three Transceiver Configuration



In Figure 3–33, two transceiver blocks cannot merge because there would be extra channels remaining. But because there are three transceiver blocks, the Quartus II software can merge them into two with no remaining channels.

If you turn on the merging option, the Quartus II software automatically merges transceiver blocks if possible or provides a warning during compilation if merging is not possible. The merging feature can reduce the transceiver block resources used in your design.

4. SONET Mode

SGX52004-1.2

Introduction

One of the most common serial backplanes in the communications or telecom area is the SONET/SDH interface. For SONET/SDH applications the synchronous transport signal STS-48 and Synchronous Transport Module -16 (STM -16) are becoming popular SONET backplanes.

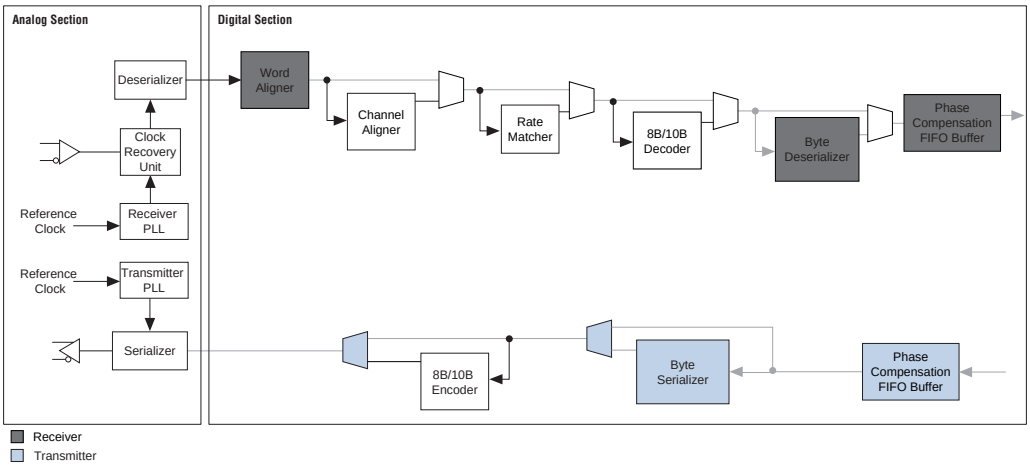
Transceiver blocks provide an implementation of SONET/SDH backplanes. The serial data range over 40" of FR4 printed circuit board support a STS-12/STS-48 and STS-192 standards data range. You can implement many functions associated with SONET/SDH processing. SONET/SDH backplanes are not designed to a specific standard because different telecom manufacturers have developed their own proprietary buses. The backplane transceiver in a SONET/SDH application requires two types of features: protocol-specific functions and electrical features. Transceiver blocks provide both of these features to a limited extent. One example is the protocol feature using A1A2 or A1A1A2A2 for word alignment.

SONET mode supports a subset of the transceiver blocks to allow for customizable configuration. The channel aligner, rate matcher, and the 8B/10B encoder/decoder features are not available in this mode. This chapter describes the supported digital architecture, clocking schemes, and software implementation in SONET mode. Figure 4–1 shows a block diagram of a transceiver channel configured in SONET mode.

Stratix® GX devices offer the following SONET/SDH features:

- Serial data rate range from 614 Mbps to 3.1875 Gbps (non-encoded)
- Input reference clock range from 38.375 to 650 MHz
- Supports parallel interface width of 8 or 16 bits
- Word aligner supports 16-bit or bit-slip mode

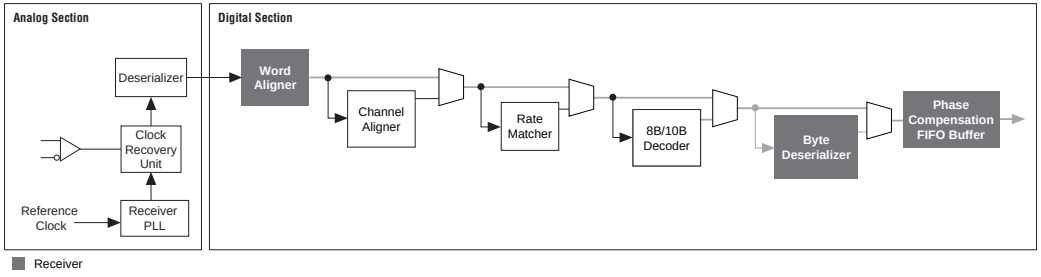
Figure 4–1. Block Diagram of Transceiver Channel Configured in SONET Mode



SONET Mode Receiver Architecture

Figure 4–2 shows the digital components of the Stratix GX receiver that are active in SONET mode.

Figure 4–2. Block Diagram of Receiver Digital Components in SONET Mode



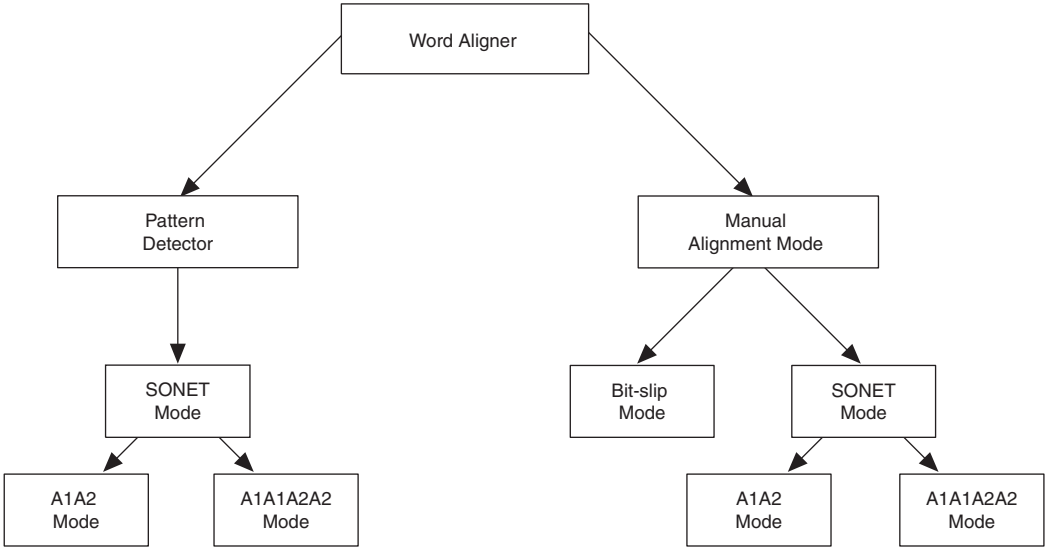
Word Aligner

For embedded clocking schemes, the clock is recovered from the incoming data stream based on transition density of the data. This feature eliminates the need to factor in receiver skew margins between the clock and data. However, with this clocking methodology, the word boundary of the re-timed data can be altered. Stratix GX transceivers offer an

embedded word alignment circuit to use in conjunction with the pattern detector to align the word boundary of the re-timed data to a specified comma. In SONET mode, this embedded circuit is configured to manual alignment mode consisting of 16-bit and bit-slip modes.

The word aligner is composed of a pattern detector, manual alignment controller, bit-slipper circuitry, and synchronization state machines. Depending on the configuration, these components work in conjunction with or independently of one another. The word aligner cannot be bypassed, but if the `rx_enacdet` signal is held low, the word aligner does not alter the word boundary. Figure 4–3 shows the various components of the word aligner in SONET mode. The functionality is described in the following sections.

Figure 4–3. Stratix GX Word Aligner Components



Pattern Detector Module

The pattern detector matches the comma to the current byte-boundary, as specified in the MegaWizard® Plug-In Manager. If the comma is found, the optional `rx_patterndetect` signal is asserted for the duration of one clock cycle to signify that the comma exists in the current word boundary. The pattern detector module only indicates that the signal exists and does not modify the word boundary. Modification of the word boundary is discussed later in the word alignment and synchronization sections.

Manual SONET Alignment Mode (2 Consecutive 8-bit Characters (A1A2) or 4 Consecutive 8-bit Characters (A1A1A2A2)
The 2 consecutive 8-bit characters, A1A2 SONET Section Overhead Framing Bytes, are used as the comma in 16-bit pattern mode.

The 16-bit comma is specified in the MegaWizard Plug-In Manager. The comma has the bit orientation of [MSB..LSB]. A1 represents the least significant byte, which consist of bits `[7..0]`, and A2 represents the most significant byte consisting of bits `[15..8]`. The comma, or alignment pattern, must be specified as [A2,A1] in the MegaWizard Plug-In Manager. If "Flip Word Alignment bits" is selected, the ordering of the alignment pattern is [LSB..MSB] for the bit ordering and [A1, A2] for the byte ordering. Only the positive disparity of the comma is detected in the mode. Table 4–1 shows several word alignment patterns based on different bit transmission orders and whether the receiver word alignment bit flip option is checked. The bit transmission order assumes that if double width mode is used, the LSB is transmitted first, followed by the MSB.

Table 4–1. Word Alignment Patterns for SONET Mode

Bit Transmission Order (at the Source)	Word Alignment Bit Flip	Word Alignment Pattern
MSB to LSB	On	1111011000101000 (hex F628)
MSB to LSB	Off	0001010001101111 (hex 146F)
LSB to MSB	Off	0010100011110110 (hex 28F6)

In SONET mode, the word aligner either aligns to two consecutive 8-bit characters (A1A2) or four consecutive 8-bit characters (A1A1A2A2). The `rx_a1a2size[]` signal differentiates between the 2 and 4 consecutive modes. The word aligner aligns to the A1A2 pattern when the `rx_a1a2size[]` is held low, or to the A1A1A2A2 when `rx_a1a2size[]` is high. If the optional `rx_a1a2size` signal is not selected, the word aligner defaults to the A1A2 mode. An optional signal, `rx_a1a2sizeout[]`, can also be enabled to send the state of the `rx_a1a2[]` signal as seen by the word aligner into the device logic array. The value of the signal is forwarded to the device, along with the byte that was in the word aligner when the `rx_a1a2size[]` signal was sampled.

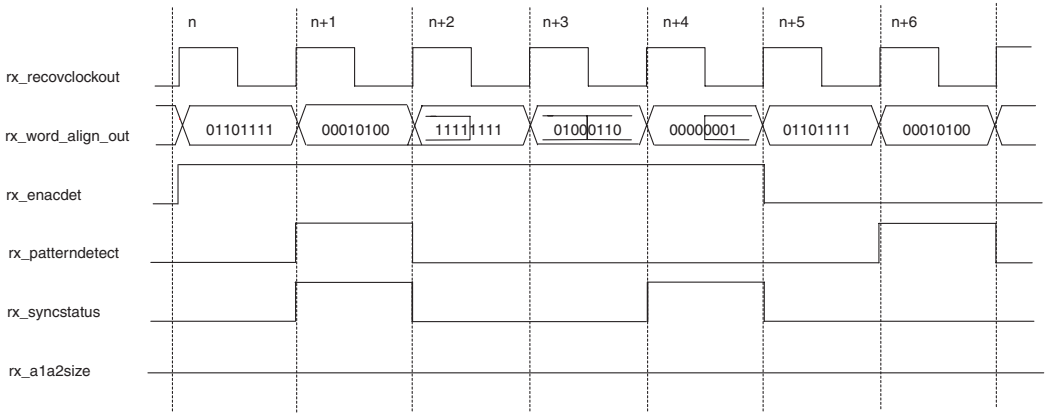
In SONET mode, the byte boundary is locked after the first comma is detected, and the boundary is aligned after the rising edge of the rx_enacdet[] signal. If the byte boundary changes the rx_enacdet[] signal must be deasserted and reasserted to reset the alignment circuit. This feature is valuable in SONET because the data is scrambled and not encoded. The comma can exist across byte boundaries and can trigger a false re-alignment. In SONET, the byte boundary must be aligned and locked at the beginning of a SONET frame, because the A1A2 comma resides in the framing section at the beginning of the transport overhead.

Because the SONET frame is a set size, the occurrence of the A1A2 framing bytes is anticipated. The actual A1A2 framing bytes are checked with a counter (A1A2 framing bytes occur every 125 µs based on an STS-1 Frame and a rate of 51.84 Mbps).

As stated earlier, at the rising edge of the rx_enacdet[], the word aligner locks onto the first comma detected. In this scenario, the rx_patterndetect[] is asserted for one clock cycle to signify that the comma has been aligned. Also, the rx_syncstatus[] signal is asserted for a clock cycle to signify that the word boundary has been synchronized. After the word boundary has been locked, regardless of whether the rx_enacdet[] is held high or low, the rx_syncstatus[] signal asserts itself for one clock cycle whenever the comma is detected across a different byte boundary. The rx_syncstatus[] operates in this re-synchronization state until a rising edge is detected on the rx_enacdet[].

Figure 4–4 shows an example of how the word aligner signals interact in SONET alignment mode for an A1A2 pattern. In this example, a SONET A1A2 Framing pattern is used (16'b0001010001101111). In this case, the A1 is represented by 8'b01101111, and A2 is represented by 8'b00010100.

Figure 4–4. Word Aligner Symbols Interacting in SONET A1A2 Manual Alignment Mode



The rx_a1a2size signal is held low. This low signal sets the SONET alignment mode to A1A2. Because rx_enacdet is toggled high at time *n*, the aligner locks to the boundary of the next present comma. Additionally, the A1 comma appears on the rx_word_align_out port during this period. At time n+1, the A2 comma appears on the rx_word_align_out port. Because the comma exists, the rx_patterndetect and rx_syncstatus signals are asserted for one clock cycle to signify that the A1A2 comma has been detected and that the word boundary has been locked. The A1A2 comma appears again across word boundaries during periods n+2, n+3, and n+4. The rx_enacdet signal is held high, but the word aligner does not re-align the byte boundary. Instead, the rx_syncstatus signal is asserted for one clock cycle to signify a re-synchronization condition. You must deassert and reassert the rx_enacdet signal to re-trigger the word aligner. The next transition occurs at time n+5, where rx_enacdet is deasserted and the A1 pattern is present on the rx_word_align_out port. At time n+6, the A2 pattern is present on the rx_word_align_out port. The word aligner then asserts the rx_patterndetect signal for one clock cycle to flag the detection of the comma on the current word boundary.

Manual Bit-Slipping Alignment Mode

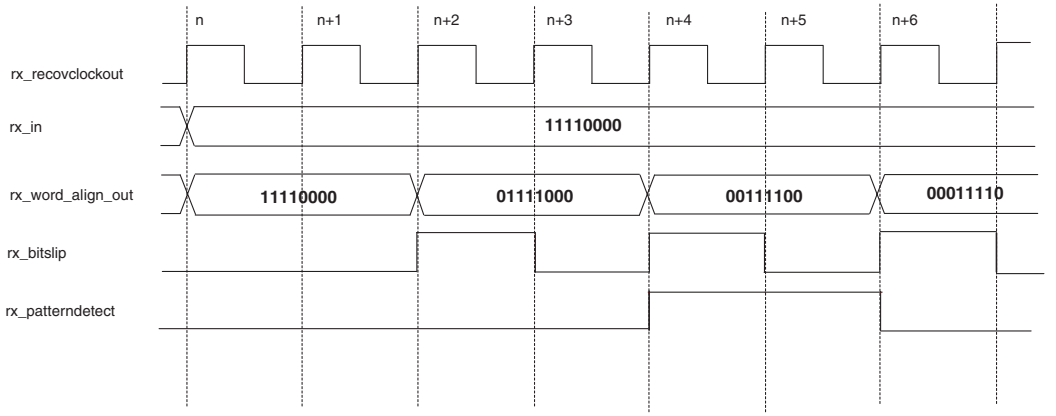
Word alignment is achieved by enabling the manual bit-slip option in the MegaWizard Plug-In Manager. With this option enabled, the transceiver can shift the word boundary by one bit in every parallel clock cycle. Bits are shifted from the MSB to LSB direction. This shift occurs every time the bit-slipping circuitry detects a rising edge of the rx_bitslip[] signal. Each time a bit is slipped, the bit that arrived at the receiver earlier is skipped. When the word boundary matches what is specified as the

comma, the rx_patterndetect[] signal is asserted for one clock cycle. You must implement the logic in the device logic array to control the bit-slip circuitry.

This scheme is useful if the comma changes dynamically when the Stratix GX device is in user mode. Because the controller is implemented in the logic array, a custom controller can be built to dynamically change the comma without needing to reprogram the Stratix GX device. The pattern detect circuitry matches only the pattern that is specified in the MegaWizard Plug-In Manager and is not dynamically adjustable.

Figure 4–5 shows an example of how the word aligner signals interact in the manual bit-slip alignment mode. In this example, 8'b00111100 is specified as the comma, and an 8'b11110000 value is held at the rx_in port. Every rising edge on the rx_bitslip port causes the rx_word_align_out data to shift one bit from the MSB to the LSB. At time n+2, the 8'b11110000 data is shifted to a value of 8'b01111000. At this state the rx_patterndetect is held low, because the specified comma does not exist in the current word boundary. The rx_bitslip is disabled at time n+3 and re-enabled at time n+4. The output of the rx_word_align_out now matches the specified comma, so the rx_patterndetect is asserted for one clock cycle. At time n+5, the rx_patterndetect is still asserted because the comma still exists in the current word boundary. Finally, at time n+6, the rx_word_align_out boundary is shifted again and the rx_patterndetect signal is deasserted to signify that the word boundary does not contain the comma.

Figure 4–5. Word Aligner Symbols Interacting in Manual Bit-Slip Mode



Byte Deserializer

The byte deserializer module further reduces the speed that the FPGA logic array must achieve in order to meet performance. The possible division factors are 8 and 16. This requirement results in a byte or double byte data width in the PLD logic array.

In SONET mode, the maximum output bus width is 22 bits. If the input includes data and control signals, the data and the control signals are deserialized to include double the data bits and 2 bits of each control signal, one for the MSB and one for the LSB. This case is shown when in SONET mode where the inputs to the Byte Deserializer are `datain[7..0]`, `rx_syncstatus`, `rx_patterndetect`, and `rx_a1a2sizeout`. These total 11 input signals feeding the byte deserializer and 22 output signals are fed to the FPGA logic array. The signals are sent into the logic array as two 11-bit buses. The aggregate bandwidth does not change by use of the Byte Deserializer because the logic array data width is doubled.

Figure 4–6 demonstrates input and output signals of the byte deserializer when deserializing an 8-bit data input to 16-bits. In this case, the finishing alignment pattern A2 (00010100) shown as 'B' is located in the MSB of the 16-bit output and this is reflected with `rx_patterndetect[1]` going high. The output of the byte deserializer is BA, DC, FE, and so on. This example assumes that the word alignment bit-flip option is unchecked (OFF), and that the transmitter and receiver bit-flip option is checked (ON) to adhere to the MSB transmitted first option.

Figure 4–6. Receiver Byte Deserializer in 8/16-Bit Mode With Finishing Alignment Pattern in MSB

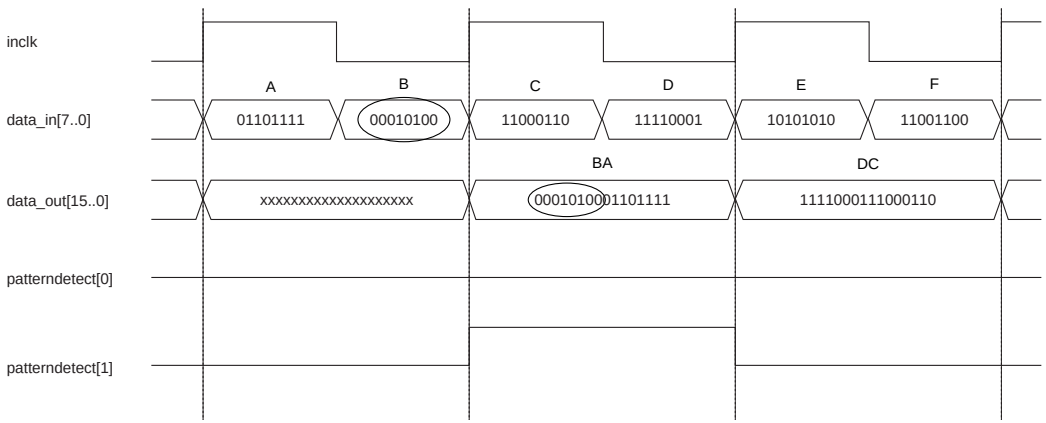
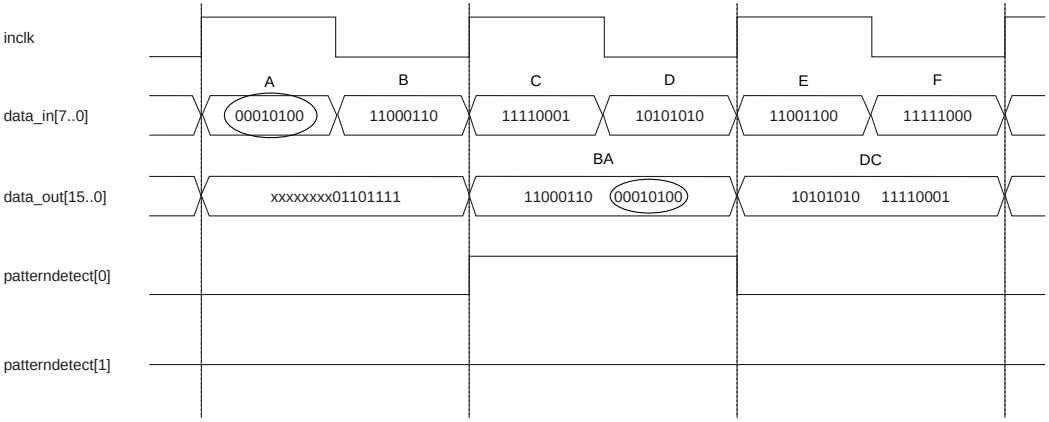


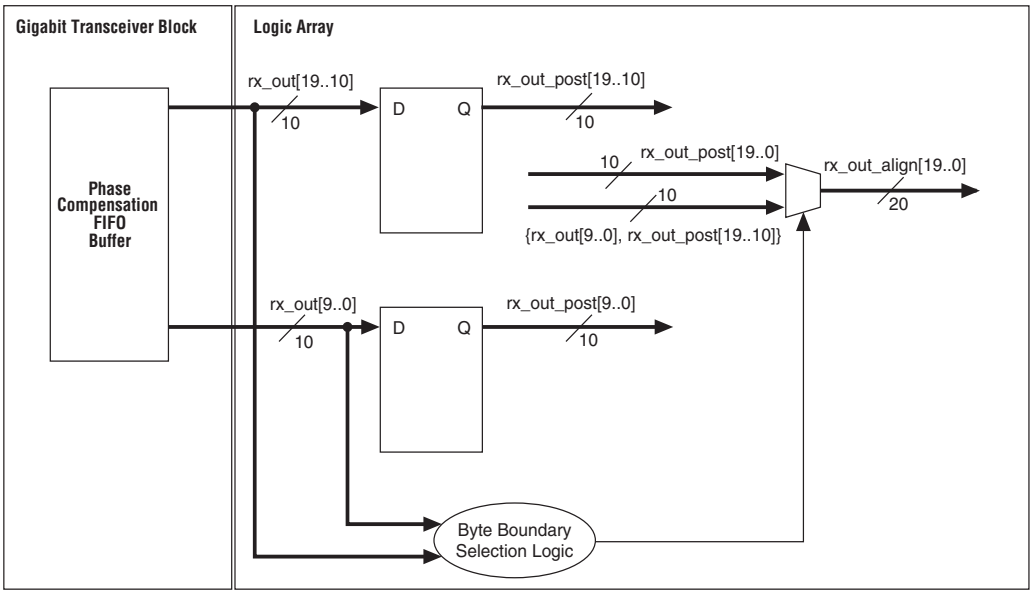
Figure 4–7 demonstrates the alternate case of the finishing alignment pattern found in the LSB of the 16-bit output. Correspondingly `rx_patterndetect[0]` goes high. In this case, the output is BA, DC, FE, and so on.

Figure 4–7. Receiver Byte Deserializer in 8/16-Bit Mode with Finishing Alignment Pattern in LSB



If necessary, you might implement logic to perform byte position alignment once data enters the logic array, as seen in Figure 4–8. In this example, the byte position selection logic determines the proper byte position based on the pattern detect signal.

Figure 4–8. Receiver Byte Deserializer Data Recovery in Logic Array



Receiver Phase Compensation FIFO Module

The receiver phase compensation FIFO module is located at the FPGA logic array interface in the receiver block and is four words deep. The FIFO module compensates for the phase difference between the clock in the FPGA and the operating clocks in the transceiver block.

In SONET mode, the write port is clocked by the recovered clock from the CRU. The rate of this clock is reduced by half if the byte deserializer is used. The read clock can be clocked by `rx_coreclk` or `rx_clkout`.

You can select `rx_coreclk` as an optional receiver input port that can also accept a clock supply for the read side of the receiver phase compensation FIFO. The receiver phase compensation FIFO buffer can only account for phase differences.

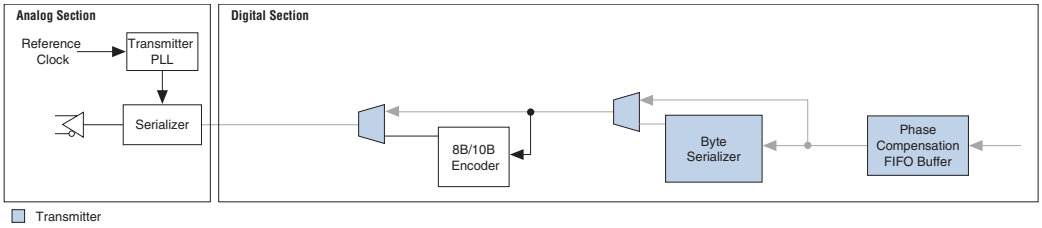
In SONET mode, if you do not select the `rx_clkout` port, the read clock of the receiver phase compensation FIFO module, clocked by `rx_coreclk`. An FPGA global clock, regional clock, or fast regional clock resource is required to make the connection for the read clock. Refer to "SONET Mode Channel Clocking" on page 4–12 or the block diagram in the MegaWizard Plug-In Manager for more information on the clock structure in a particular mode.

The Receiver Phase Compensation FIFO module is always used and cannot be bypassed.

SONET Mode Transmitter Architecture

Figure 4–9 shows a diagram of the digital components of the transmitter. The rest of this section describes the active components of the transmitter, which are the phase compensation FIFO buffer and the byte serializer. The 8B/10B decoder is not active during SONET mode.

Figure 4–9. Block Diagram of the Transmitter Digital Components in SONET Mode



Transmitter Phase Compensation FIFO Buffer

The transmitter phase compensation FIFO buffer is located at the FPGA logic array interface in the transmitter block and is four words deep. The phase compensation FIFO module compensates for the phase difference between the clock in the FPGA and the operating clocks in the transceiver block.

The read port of the phase compensation FIFO buffer is clocked by the transmitter PLL clock. The write clock is clocked by `tx_coreclk`. You can select the `tx_coreclk` as an optional transmitter input port to supply a clock to. In this case, you must ensure that there is no frequency difference between the `tx_coreclk` and the transmitter PLL clock. The transmitter phase compensation FIFO module can only account for phase differences.

If the `tx_coreclk` is not selected as an optional input transmitter port, `tx_coreclk` is fed by `coreclk_out`. This connection occurs using the logic array routing. In this case, the software defaults to using an FPGA global clock, regional clock, or fast regional clock resource.

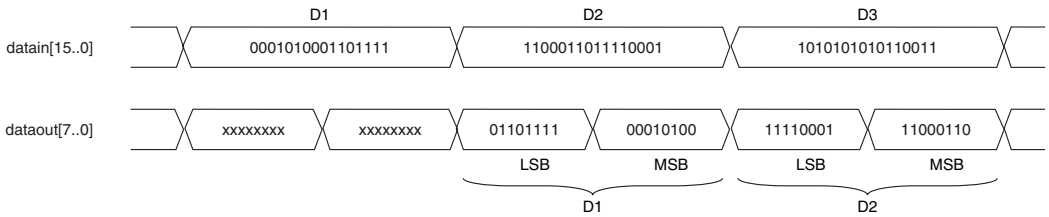
The Transmitter Phase Compensation FIFO module is always used and cannot be bypassed. The input to the Transmitter Phase Compensation FIFO module is the data from the FPGA logic array.

Byte Serializer

In SONET mode, the Byte Serializer in the transmitter block takes in a 16-bit input from the phase compensation FIFO module and serializes it to 8 bits. It transmits the least significant byte to the most significant byte. The transmitter digital reset must always be used to reset the Byte Serializer FIFO module pointers whenever an unknown state is encountered, for example, during periods when the transmitter PLL loses lock. Refer to Chapter 8, Reset Control and Power Down, for further details on the reset sequence.

Figure 4–10 demonstrates input and output signals of the byte serializer when serializing a 16 bit input to 8 bits. The `tx_in[] signal` is the input from the FPGA logic array that has already passed through the Transmitter Phase Compensation FIFO module.

Figure 4–10. Transmitter Byte Serializer in 8- to 16-Bit Mode



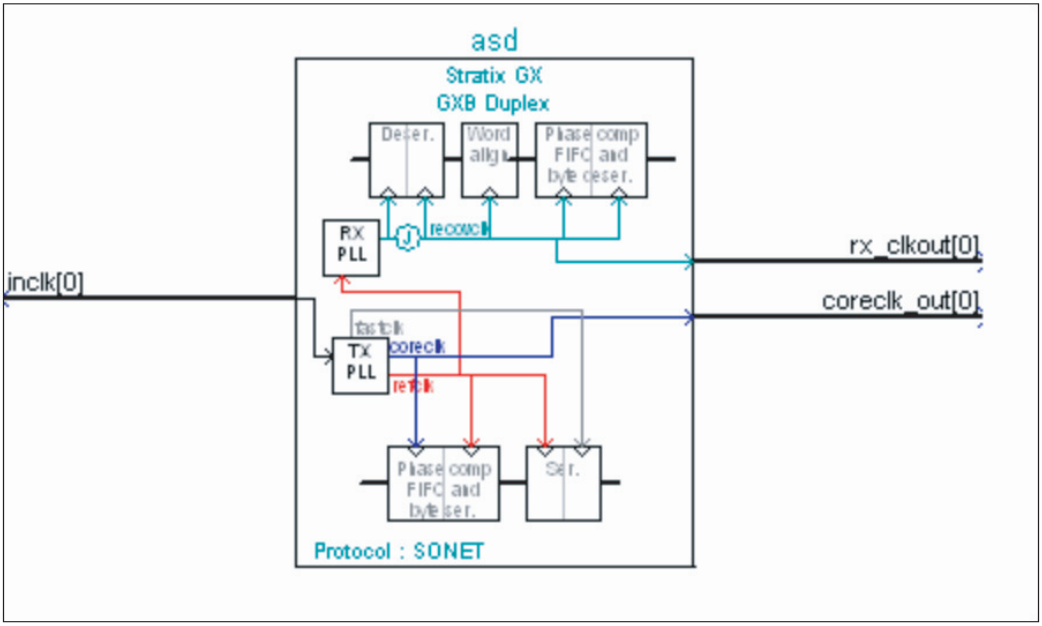
The LSB is transmitted before the MSB in the transmitter byte serializer. Figure 4–10 shows the order of data transmitted. For the input of D1, the output is D1LSB and then D1MSB. The byte serializer is selected in the MegaWizard Plug-In Manager when a 16-bit channel width is selected.

SONET Mode Clocking

SONET Mode Channel Clocking

This section covers describes the internal clocking and the external clocks of the transceiver in SONET mode. By default, the MegaWizard Plug-In Manager parameterizes the `altgxb` megafunction with the clock configuration shown in Figure 4–11.

Figure 4–11. Default Configuration of altgxb in SONET Mode



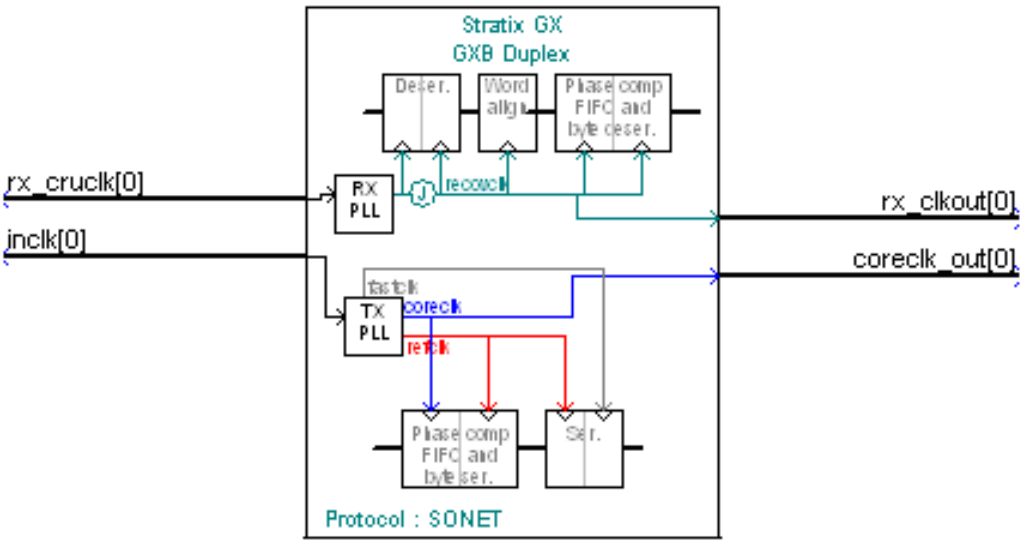
In Figure 4–11, the altgxb megafunction is configured so that the train receiver PLL with transmitter PLL is enabled. The transmitter PLL is fed from an inclk port, which can be fed from a dedicated REFCLKB, Global clock, Regional clock, or Fast Regional clock source. The receiver logic is clocked by the recovered clock from the clock recovery unit, rx_clkout. This recovered clock is also fed into the FPGA so that, in a multi-crystal environment, some level of clock domain decoupling can be implemented to interface with a system clock. On the transmitter channel, the output of the transmitter PLL, coreclk_out, is sent into the logic array and also loops back to clock the write side of the transmit phase compensation FIFO module.

The train receiver PLL CRU clock from the transmitter PLL feature can be disabled in the altgxb MegaWizard Plug-In Manager. Deselecting this option enables an additional rx_cruclk input reference clock port for the receiver PLL. This feature supports additional multiplication factors for the receiver PLL and allows for the separation of receiver and transmitter reference clocks. This separation is required if the output reference clock frequency from the transmitter PLL exceeds the 325 MHz phase frequency detector of the receiver PLL. For more information on

this feature, refer to the *Stratix GX Analog Description* chapter of the *Stratix GX Device Handbook, Volume 2*. This configuration is shown in Figure 4–12.

If double width is used (16-bit bus) and the data rate is above 2,600 Mbps, the trained receiver PLL clock from the transmitter PLL must be turned off, because the output clock from the transmitter PLL exceeds the 325-MHz limit on the receiver PLL input clock, if the input clock is fed from any non-REFCLKB pin. REFCLKB pins have a 650-MHz limit.

Figure 4–12. altgxb Megafunction in SONET Mode With Train Receiver CRU From Transmitter PLL Disabled



This configuration contains an independent rx_cruclk, which feeds the receiver PLL reference clock. This input clock port is only available when the receiver PLL is not trained by the transmitter PLL. One rx_cruclk is associated with a channel. If four channels are active, there are four rx_cruclks.

The rx_clkout is the recovered clock from the associated receiver channel. An rx_clkout is available for each receiver channel that is used. This clock is used to clock the write port of a rate matching FIFO module. The read port of the FIFO module is clocked by the coreclk_out or PLD clock.

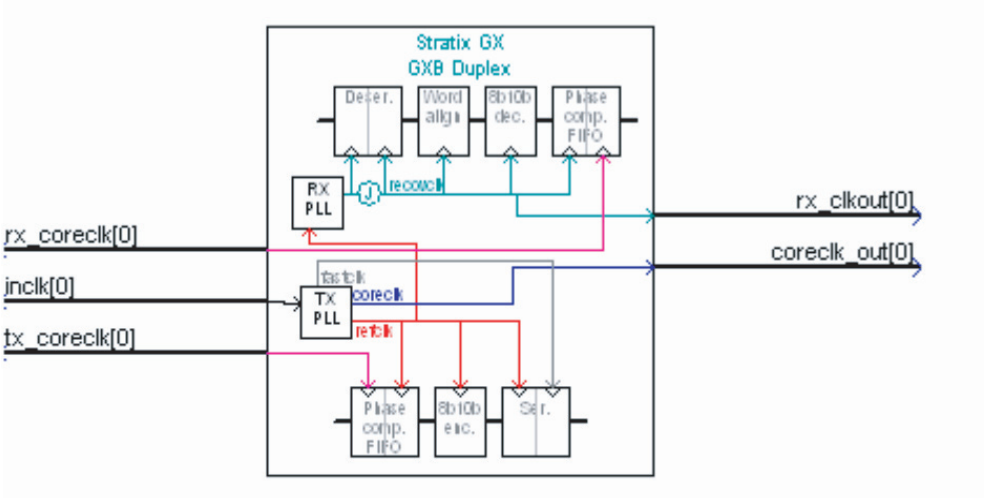
The coreclk_out is the output from the transmitter PLL. A coreclk_out is available for each transceiver block that is used. Altera® recommends clocking the logic that is feeding the transmitter with this clock.

The read clock of the receiver phase compensation FIFO module and the write clock of the transmitter phase compensation FIFO module are optionally enabled to manually feed in a clock from the FPGA logic array. You use these options to optimize the global clock usage. For instance, if all transmitter channels between transceiver blocks are from a common clock domain, the transceiver instantiations use a total of one global resource clock versus one global per transceiver block, if the tx_coreclk option is not enabled.

The same situation can be optimized for the receiver channels in a single crystal synchronous system with the rx_coreclk. Even in a system that is based on a single crystal, the recovered clock can still become asynchronous to the system clock during initialization or long run lengths. As a result, the pointers of the Receiver Phase Compensation FIFO module might overlap and fail to function correctly. In situations where there are long run lengths or no data transmissions, these FIFO modules must be reset by the rxdigitalreset signal.

In multi-crystal environments, individual recovered clocks must drive the read clock of the phase compensation FIFO module. The Quartus® II software does this by default, and you do not have to manually make this connection. The rx_coreclk and tx_coreclk must be frequency matched with their respective read and write ports. The phase compensation FIFO module can only correct for phase, not frequency differences. Figure 4–13 shows the clock configuration with these optional input ports enabled.

Figure 4–13. altgxb in SONET Mode With rx_coreclk & tx_coreclk Enabled



For reference, the various input and output clock ports are listed in Table 4–2.

Table 4–2. List of Clocking Input & Output Ports Available in SONET Mode (Part 1 of 2)

Clock	Port	Description
rx_cruclk	Input	Input to CRU available as a port when CRU is not trained by the transmitter PLL.
inclk	Input	Input to the transmitter PLL, available as a port when the transmitter PLL is instantiated.
coreclk_out	Output	Output clock from the transmitter PLL equivalent to TX_PLL_CLK. Available as a port if the transmitter PLL is used.

Table 4–2. List of Clocking Input & Output Ports Available in SONET Mode (Part 2 of 2)

Clock	Port	Description
rx_clkout	Output	Output clock from transceiver. In this mode, rx_clkout is the recovered clock of the respective channel.
tx_coreclk	Input	Clocks the write port of transmitter phase compensation FIFO module. Available as an optional port in the Quartus II MegaWizard® Plug-In Manager. Must be frequency matched to tx_pll_clk. If not available as a port, this is fed by coreclk_out through logic array routing.
rx_coreclk	Input	Clocks read port of receiver phase compensation FIFO module. Available as an optional port in the Quartus II MegaWizard Plug-In Manager. If not available as a port, this is fed by rx_clkout through logic array routing.

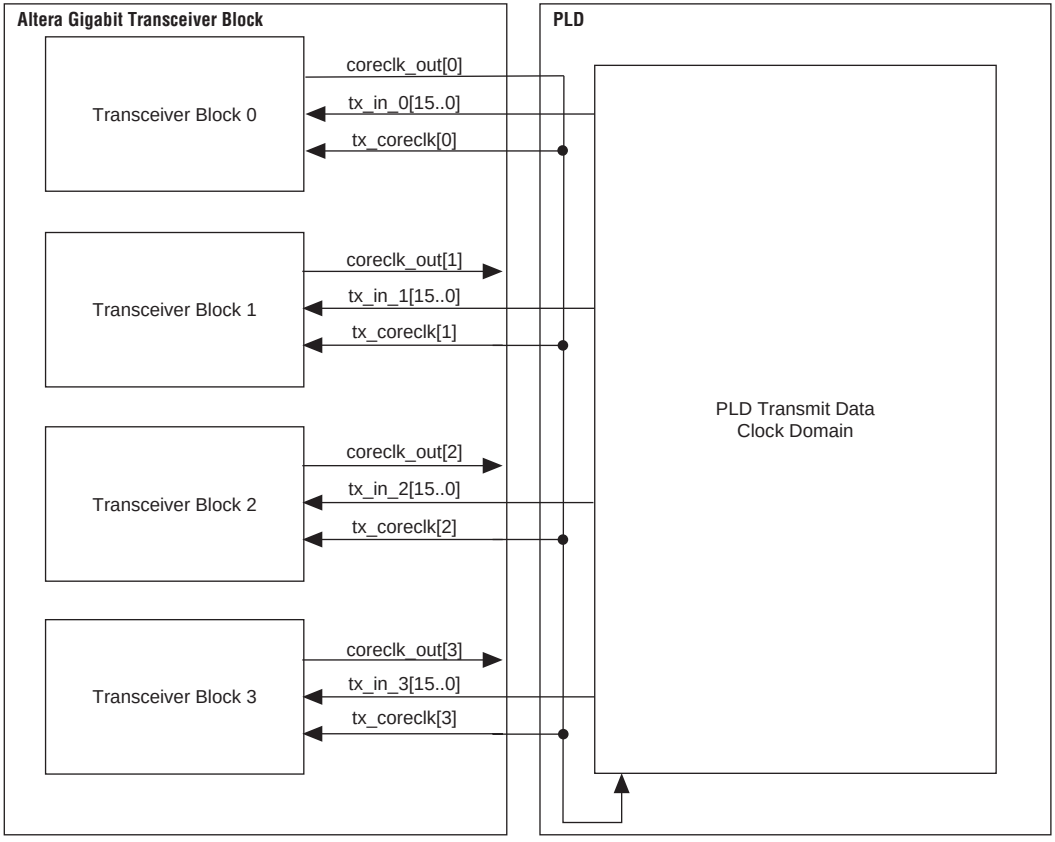
SONET Mode Inter-Transceiver Block Clocking

This section provides guidelines for using transceiver interface clocking between the FPGA logic array and transceiver channels when multiple transceiver blocks are active. Depending on each mode supported by Stratix GX devices, each transceiver block contains different transceiver-to-FPGA interface clocking. Different input and output clocks are available based on the options provided by the Quartus II MegaWizard Plug-In Manager's built-in functions. The number of supported channels varies based on which Stratix GX device you select. Because of the various configurations of input and output clocks, consider the clocking schemes between inter-transceiver blocks carefully to prevent problems later in the design cycle.

One of the clocking interfaces to consider while designing with Stratix GX devices is the transceiver-to-FPGA interface. This clocking scheme is further classified as the FPGA-to-transmitter channel and the FPGA-to-receiver channel to the PLD.

In SONET mode, the write port of the transmitter phase compensation FIFO module is either clocked by the `coreclk_out` or by the `tx_coreclk` signal. The constraint on using `tx_coreclk` is that the clock must be frequency locked to the read clock of the transmitter phase compensation FIFO module. Synchronous data transfers for a multi-transceiver block configuration are accomplished by using the `tx_coreclk` port. The `tx_coreclk` of multi-transceiver blocks are connected to a common clock domain either from a single `coreclk_out` signal or from an FPGA system clock domain. This scheme is shown in Figure 4–14.

Figure 4–14. Example of a Multi-Transceiver Block FPGA to Transmitter Interface Clocking Scheme in SONET Mode



When `tx_coreclk` is not enabled, the Quartus II software automatically routes the `coreclk_out` signal to the write clock of the phase compensation FIFO module via a global, regional, or fast regional resource. In multi-transceiver block configuration, this routing might lead to timing violations because the `coreclk_out` per transceiver block cannot guarantee phase relationship. For this reason, Altera recommends clocking the `tx_coreclk` with a common clock for synchronous transmission.

Another inter-transceiver block consideration is the selection of the dedicated `refclkb` pin. Stratix GX channels are arranged in banks of four, or transceiver blocks. Each transceiver block has the ability to share a common reference clock through the inter-transceiver (IQ) lines. The

Stratix GX logic array clock usage can be reduced by using the IQ lines. The IQ lines are used when a refclkb input port from one transceiver block or channel drives any other transceiver blocks or channels. The Quartus II software automatically determines the IQ line usage.

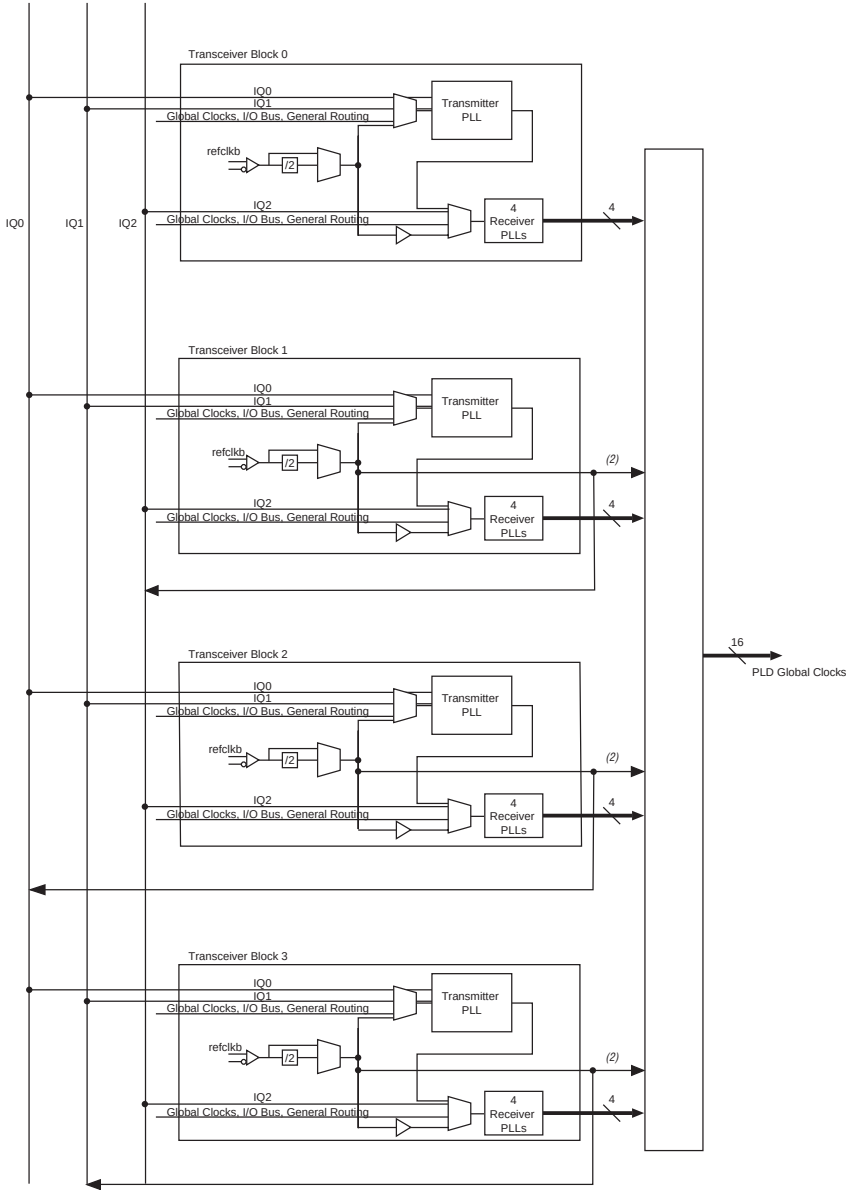
When determining the location of refclkb pins, be sure to take into consideration what is fed by the pin you choose. Table 4–3 shows the available IQ lines and which transceiver block refclkb drives them. This capability is based on the number of transceiver channels in the Stratix GX device.

Table 4–3. REFCLKB to Inter-Transceiver Line Connections

Channel Density	REFCLKB in Transceiver Block Number	Channels in Transceiver Block	IQ Line Driven by REFCLKB
8 channels (EP1S10)	0	[3:0]	IQ2
	1	[7:4]	IQ0
16 channels (EP1S25)	0	[3:0]	N/A
	1	[7:4]	IQ2
	2	[11:8]	IQ0
	3	[15:12]	IQ1
20 channels (EP1S40)	0	[3:0]	N/A
	1	[7:4]	IQ2
	2	[11:8]	IQ0
	3	[15:12]	IQ1
	4	[19:16]	N/A

Figure 4–15 shows the transceiver routing with respect to inter-transceiver lines. It is important to use this information when placing REFCLKB pins. For example, if a REFCLKB pin is required to feed a transmitter PLL using an IQ line, the REFCLKB pin cannot be in transceiver block 1, because IQ2 only feeds the receiver PLLs.

Figure 4–15. Inter-Transceiver Line Connections for EP1SGX25 Device *Note (1)*



Notes to Figure 4–15:

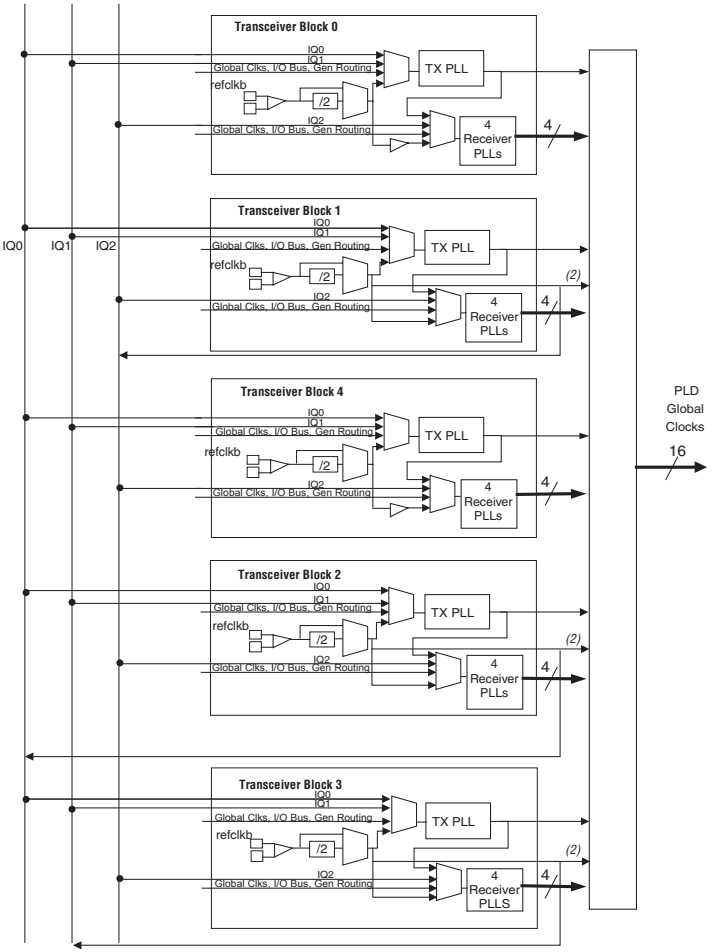
(1) IQ lines are inter-transceiver block lines.

(2) If the /2 pre-divider is used, the path to drive the PLD logic array, local, or global clocks is not allowed.

(3) There are four receiver PLLs in each transceiver block.

Figure 4–16 shows the transceiver routing with respect to IQ lines for the EP1SGX40G device. This device has an extra transceiver block (transceiver block 4), in the middle of the other transceiver blocks, as shown. Again, this information is important when determining where to place REFCLKB pins. For example, if a REFCLKB pin is needed to feed to a transmitter PLL using an IQ line, the pin cannot be in transceiver block 1 because IQ2 feeds only the receiver PLLs.

Figure 4–16. IQ Line Connections for EP1SGX40G Device *Note (1)*



Notes to Figure 4–16:

(1) IQ lines are inter-transceiver block lines.

(2) If the /2 pre-divider is used, the path to drive the PLD logic array, local, or global clocks is not allowed.

(3) There are four receiver PLLs in each transceiver block.

SONET Mode MegaWizard Plug-In Manager

This section describes the altgxb megafunction options in the MegaWizard Plug-In Manager for SONET mode. Altera recommends that the Stratix GX transceiver block be instantiated and parameterized through the altgxb megafunction in the MegaWizard Plug-In Manager. The Quartus II MegaWizard Plug-In Manager offers a Graphical User Interface (GUI) that organizes the altgxb options in easy to use sections. The MegaWizard Plug-In Manager also sets the proper ports and parameters automatically based on the selected options and parameters. Invalid settings are automatically flagged in the MegaWizard Plug-In Manager to help prevent illegal configurations. The MegaWizard Plug-In Manager also grays out any options that do not apply to SONET mode.

Although you can instantiate the Stratix GX block directly by calling out the altgxb megafunction, Altera recommends using the MegaWizard Plug-In Manager to instantiate the altgxb megafunction to reduce the chance of invalid settings.

SONET Mode MegaWizard Plug-In Manager Considerations

Each altgxb megafunction instantiation uses one or more transceiver blocks based on the number of channels that you select. There are four channels per transceiver block. If a MegaWizard Plug-In Manager instantiation uses fewer than four channels, the remaining channels in that transceiver block are not available for use.

Each MegaWizard Plug-In Manager instantiation must have similar functionality and data rates. To have transceiver blocks that differ in functionality and/or data rates, you can create a separate instantiation for each transceiver block.

As mentioned in the "SONET Mode Clocking" on page 4–12, the MegaWizard Plug-In Manager displays the configuration of the altgxb megafunction, as shown in Figure 4–11 on page 4–13. This diagram changes dynamically based on the selected mode, options, and clocking schemes.

SONET Mode altgxb MegaWizard Plug-In Manager Options

This section shows the MegaWizard Plug-In Manager pages where you select the options for a SONET mode configuration.

Figure 4–17 shows page 3 of the altgxb MegaWizard Plug-In Manager in SONET mode.

Figure 4–17. MegaWizard Plug-In Manager - altgxb (Page 3)

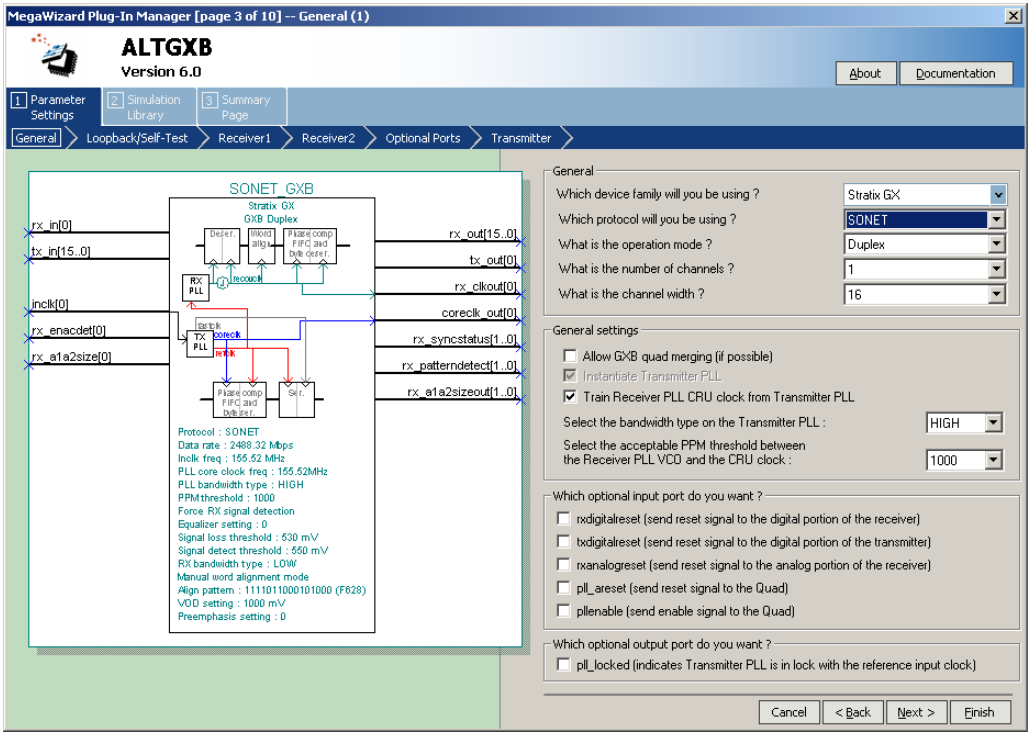


Table 4–4 describes the available options on page 3 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 4–4. MegaWizard Plug-In Manager Options (Page 3 for SONET Mode) (Part 1 of 2)

altgxb Setting	Description
Which device family will you be using?	Stratix GX is the only option available.
Which protocol will you be using?	For the SONET mode, you must select the SONET protocol.
What is the operation mode?	SONET protocol mode supports duplex, receiver- only, or transmitter-only operation modes.
What is the number of channels?	This value can be from 1 to the maximum number of channels available on the device. The Quartus II software automatically assigns the channels to a transceiver block unless input and output pin assignments are made to the channel's HSSIO input and output pins.

Table 4–4. MegaWizard Plug-In Manager Options (Page 3 for SONET Mode) (Part 2 of 2)

altgxb Setting	Description
Allow GXB quad merging (if possible)	For information about this option, refer to the section "Stratix GX Transceiver Merging" on page 4–36.
What is the channel width?	8 bits is single width and 16 bits is double width.
Instantiate Transmitter PLL	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Train Receiver PLL CRU clock from Transmitter PLL	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the bandwidth type on the Transmitter PLL	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the acceptable PPM threshold between the Receiver PLL VCO and the CRU clock	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
`rxdigitalreset` (send reset signal to the digital portion of the receiver)	The `rxdigitalreset` port resets the digital blocks in the receiver channel. Each active receiver channel has its own digital reset.
`txdigitalreset` (send reset signal to the digital portion of the transmitter)	The `txdigitalreset` port resets the digital blocks of the transmitter channel. Each active transmitter channel has its own digital reset.
`rxanalogreset` (send reset signal to the analog portion of the receiver)	The `rxanalogreset` port resets the receiver's analog circuits, including the receiver PLL. Each active receiver channel has its own analog reset.
`pll_areset` (send reset signal to the Quad)	The `pll_areset` port resets the entire transceiver block (all receiver and transmitter digital and analog circuits, including receiver and transmitter PLLs).
`pllenable` (send enable signal to the Quad)	The `pllenable` port enables the entire transceiver block; if deasserted, the entire transceiver block is held in the reset condition.
`pll_locked` (indicates Transmitter PLL is in lock with the reference input clock)	For more information, refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 4–18 shows page 4 of the `altgxb` MegaWizard Plug-In Manager in SONET mode.

Figure 4–18. MegaWizard Plug-In Manager - altgxb (Page 4)

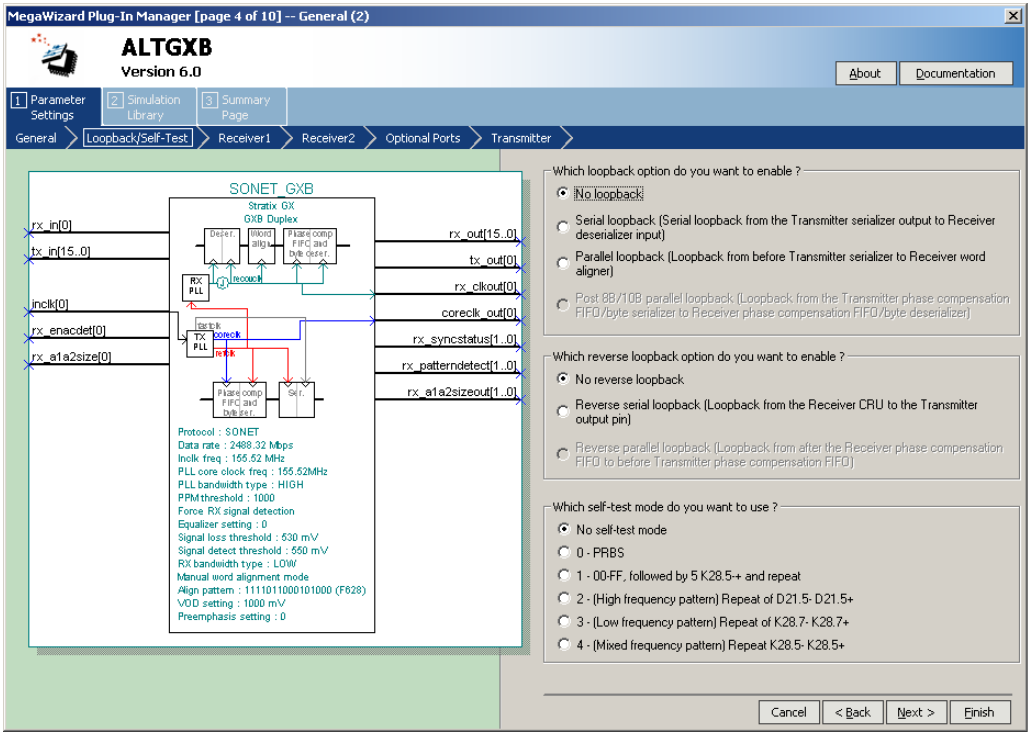


Table 4–5 describes the available options on page 4 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 4–5. MegaWizard Plug-In Manager Options (Page 4 for SONET Mode)

altgxb Setting	Description
Which loopback option do you want to enable?	For more information, refer to the *Loopback Modes* chapter in volume 2 of the *Stratix GX Device Handbook*.
Which reverse loopback option do you want to enable?	For more information, refer to the *Loopback Modes* chapter in volume 2 of the *Stratix GX Device Handbook*.
Which self-test mode do you want to use?	For more information, refer to the *Stratix GX Built-In Self Test (BIST)* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 4–19 shows page 5 of the `altgxb` MegaWizard Plug-In Manager in SONET mode.

Figure 4–19. MegaWizard Plug-In Manager - altgxb (Page 5)

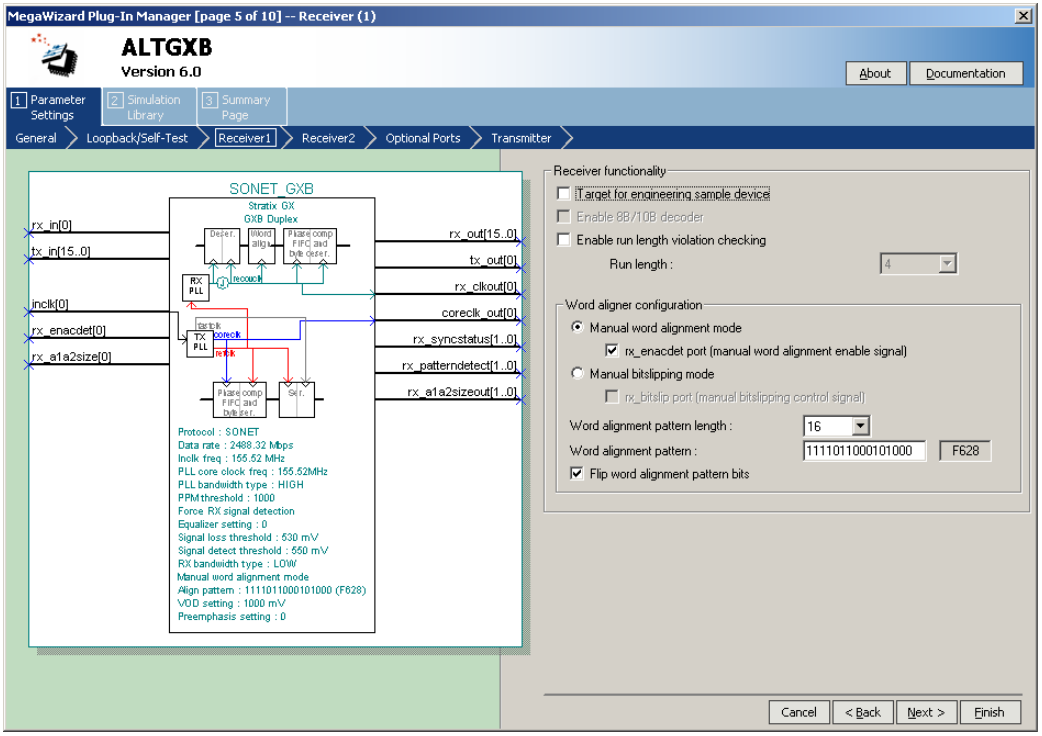


Table 4–6 describes the available options on page 5 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 4–6. MegaWizard Plug-In Manager Options (Page 5 for SONET Mode) (Part 1 of 2)

altgxb Setting	Description
Target for engineering sample device	You must select this option if the design is targeted for an engineering sample (ES) device.
Enable 8B/10B decoder	This option is not available in SONET mode.
Enable run-length violation checking	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Manual word alignment mode	Use this option to configure the word aligner in manual alignment mode. Refer to the "Word Aligner" on page 4–2 section for more information.

Table 4–6. MegaWizard Plug-In Manager Options (Page 5 for SONET Mode) (Part 2 of 2)

altgxb Setting	Description
rx_enacdet port (manual word alignment enable signal)	The rx_enacdet port supports the word aligner to byte align to the word alignment pattern (active high synchronous signal). The signal must go low, then high to trigger word realignment. If this option is not turned on, the word aligner is not active, but the pattern detect signal is still functional.
Manual bitslipping mode	Manual bit-slipping mode lets you control the word aligner's shift register directly via the rx_bitslip port. A low to high transition on the rx_bitslip port enables the word aligner's shift register to slip one bit. For example, if a 3-bit shift is required to align the incoming byte, rx_bitslip must be toggled low, high, low, high, low, high. The rx_bitslip port can be left in the high or low position after the above sequence.
rx_bitslip port (manual bitslipping control signal)	A low to high transition on the rx_bitslip port enables the word aligner's shift register to slip one bit. For example, if a 3-bit shift is required to align the incoming byte, rx_bitslip must be toggled low, high, low, high, low, high. The rx_bitslip port can be left in the high or low position after the above sequence.
Word alignment pattern length	Only 16-bit word alignment pattern length is allowed in SONET mode.
Word alignment pattern	The word alignment pattern size in SONET mode is always set to 16-bits. The pattern must be set to 1111011000101000 (F628) if "flip word alignment pattern bits" is turned on. Otherwise, set it to 0001010001101111 (146F), regardless of whether the incoming pattern is an A1/A2 or A1/A1/A2/A2.
Flip word alignment pattern bits	Flips the word alignment bit order. If this option is turned on, the right-most bit is the MSB, otherwise the right-most bit is the LSB. This option is used in conjunction with the receiver and transmitter bit-flip options to ensure that the MSB is transmitted and received first in the serial stream.

Figure 4–20 shows page 6 of the altgxb MegaWizard Plug-In Manager in SONET mode.

Figure 4–20. MegaWizard Plug-In Manager - altgxb (Page 6)

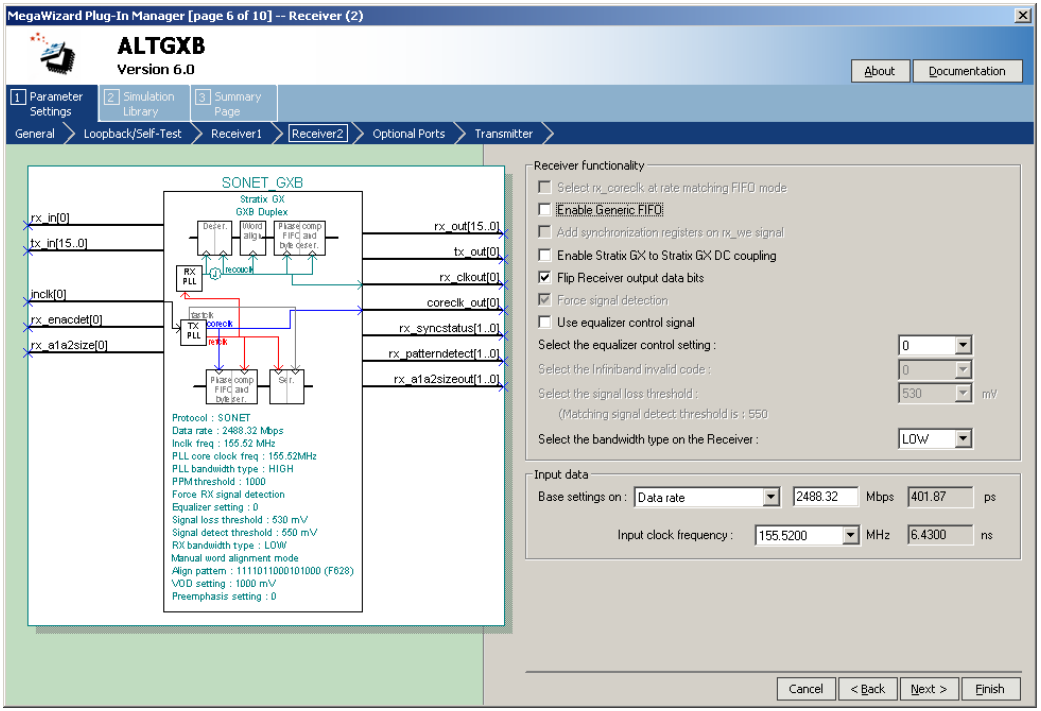


Table 4–7 describes the available options on page 6 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 4–7. MegaWizard Plug-In Manager Options (Page 6 for SONET Mode) (Part 1 of 2)

altgxb Setting	Description
Select `rx_coreclk` at rate matching FIFO mode	This option is not available in SONET mode.
Enable Generic FIFO	Select this option to include a generic FIFO in the receiver data path between the word aligner and the 8B/10B decoder (if enabled). This FIFO can be used to decouple between the recovered clock and the local `rx_coreclk`.
Enable Stratix GX to Stratix GX DC coupling	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Flip Receiver output data bits	Flips the bit ordering at the receiver output to the FPGA. The bit flip operates in a by-byte mode only. The low byte and high byte are flipped separately. The low byte is still transmitted first. This option is used in conjunction with transmitter and word aligner bit flip in SONET mode.

Table 4–7. MegaWizard Plug-In Manager Options (Page 6 for SONET Mode) (Part 2 of 2)

altgxb Setting	Description
Force signal detection	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Use equalizer control signal	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the equalizer control setting	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the Infiniband invalid code	This option is not available in SONET mode.
Select the signal loss threshold	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the bandwidth type on the Receiver	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Base settings on	By default, the SONET data rate is set at 2488.32 Mbps. Other data rates are possible, but they must adhere to the set multiplication factors of 2, 4, 5, 8, 10, 16, and 20 of the input clock. Multiplication factors of 2, 4, and 5 must use the dedicated `refclkb` pins. A multiplication factor of 2 also requires that the receiver PLL be trained by the transmitter PLL.

Figure 4–21 shows page 7 of the `altgxb` MegaWizard Plug-In Manager in SONET mode.

Figure 4–21. MegaWizard Plug-In Manager - altgxb (Page 7)

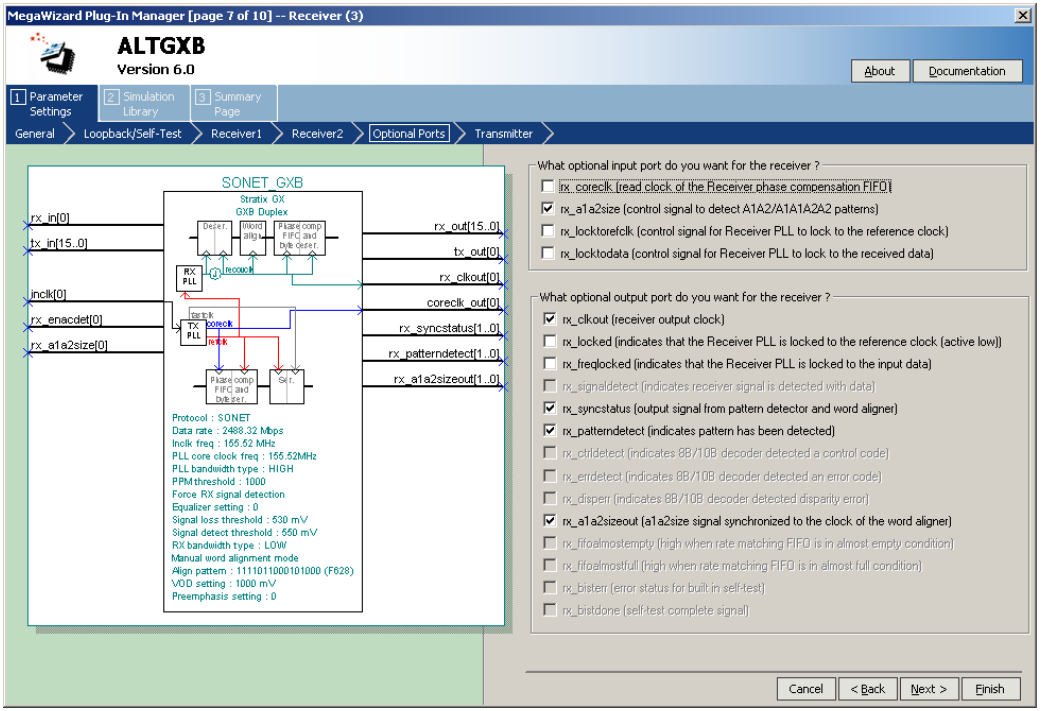


Table 4–8 describes the available options on page 7 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 4–8. MegaWizard Plug-In Manager Options (Page 7 for SONET Mode) (Part 1 of 2)

altgxb Setting	Description
`rx_coreclk` (read clock of the Receiver phase compensation FIFO)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
`rx_a1a2size` (control logic signal to detect A1A2/A1A1A2A2 patterns)	Indicates to the word aligner to align to an A1A2 or A1A1A2A2 pattern. Low = A1A2 High = A1A1A2A2
`rx_locktorefclk` (control signal for Receiver PLL to lock to the reference clock)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.

Table 4–8. MegaWizard Plug-In Manager Options (Page 7 for SONET Mode) (Part 2 of 2)

altgxb Setting	Description
rx_locktodata (control signal for Receiver PLL to lock to the received data)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_clkout (receiver input clock)	The rx_clkout signal is a recovered clock output from individual receiver channels. One rx_clkout signal is available per channel.
rx_locked (indicates that the Receiver PLL is locked to the reference clock (active low))	Transmitter PLL and receiver PLL lock indicator. For pll_locked, High = transmitter PLL is locked to the reference clock. For rx_locked, Low = receiver PLL is locked to the reference clock.
rx_freqlocked (indicates that the Receiver PLL is locked to the input data)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_signaldetect (indicates receiver signal is detected with data)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_syncstatus (output signal from pattern detector and word aligner)	Indicates when the word aligner has aligned to the byte boundary. The rx_syncstatus signal goes high for one rx_clkout period when the word aligner aligns to the new byte boundary. In 16-bit mode, each high and low byte has a separate rx_syncstatus signal.
rx_patterndetect (indicates pattern has been detected)	Similar to rx_syncstatus, except that rx_patterndetect asserts only when the word alignment pattern appears in the data stream within the synchronized byte boundary.
rx_ctrldetect (indicates 8B/10B decoder detected a control code)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_errdetect (indicates 8B/10B decoder detected an error code	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_disperr (indicates 8B/10B decoder detected disparity error)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_a1a2sizeout (a1a2size signal synchronized to the clock of the word aligner)	A loopback of the rx_a1a2size signal that is synchronized with the current byte from the word aligner.
rx_fifoalmostempty (high when rate matching FIFO is in almost empty condition)	If Generic FIFO is enabled to perform rate matching between the recovered clock and local receiver clock, this signal indicates an almost empty condition when driven HIGH.
rx_fifoalmostfull (high when rate matching FIFO is in almost full condition)	If Generic FIFO is enabled to perform rate matching between the recovered clock and local receiver clock, this signal indicates an almost full condition when driven HIGH.
rx_bisterr (error status for built-in self-test)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_bistdone (self-test complete signal)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 4–22 shows page 8 of the altgxb MegaWizard Plug-In Manager in SONET mode.

Figure 4–22. MegaWizard Plug-In Manager - altgxb (Page 8)

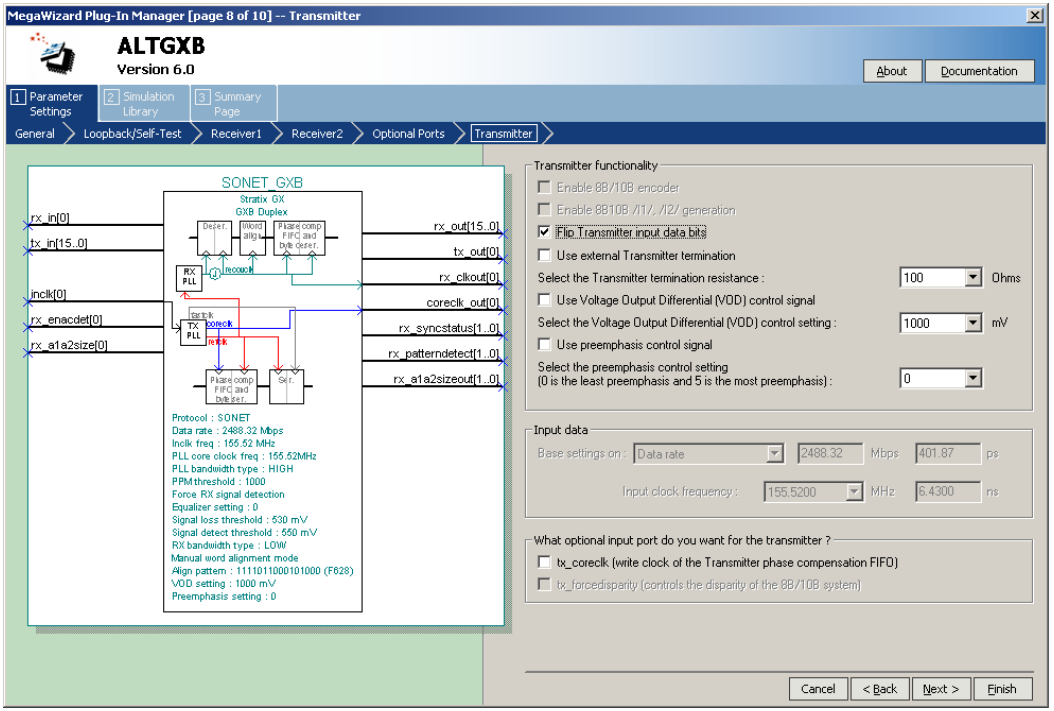


Table 4–9 describes the available options on page 8 of the MegaWizard Plug-In Manager for your altgxb custom megafunction variation.

Table 4–9. MegaWizard Plug-In Manager Options (Page 8 for SONET Mode) (Part 1 of 2)

altgxb Setting	Description
Enable 8B/10B encoder	This option is not available in SONET mode.
Enable 8B/10B /I1/, /I2/ generation	This option is not available in SONET mode.
Flip Transmitter input data bits	Flips the bit ordering from the FPGA to the transmitter input. Bit-flip operates in a by-byte mode only. The low byte and high byte are flipped separately, and the low byte is still transmitted first. This option is used in conjunction with the receiver and word aligner bit-flip in SONET mode.
Use external Transmitter termination	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.

Table 4–9. MegaWizard Plug-In Manager Options (Page 8 for SONET Mode) (Part 2 of 2)

altgxb Setting	Description
Use Voltage Output Differential (VOD) control signal	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the Voltage Output Differential (VOD) control setting	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Use preemphasis control signal	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the preemphasis control setting (0 is the least preemphasis and 5 is the most preemphasis)	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
`tx_coreclk` (write clock of the Transmitter phase compensation FIFO buffer	For information about this option, refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
`tx_forcedisparity` (controls the disparity of the 8B/10B system)	This option is not available in SONET mode.

Figure 4–23 shows page 9, the Simulation Libraries page, of the MegaWizard Plug-In Manager for the SONET protocol set up.

Figure 4–23. MegaWizard Plug-In Manager - altgxb (Page 9)

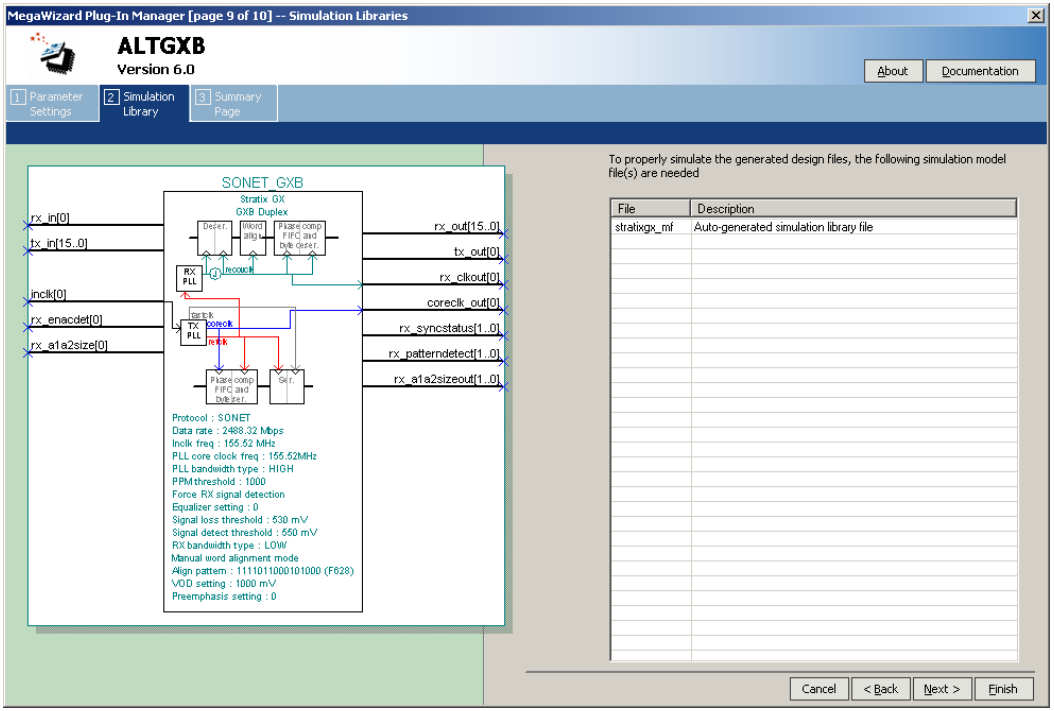
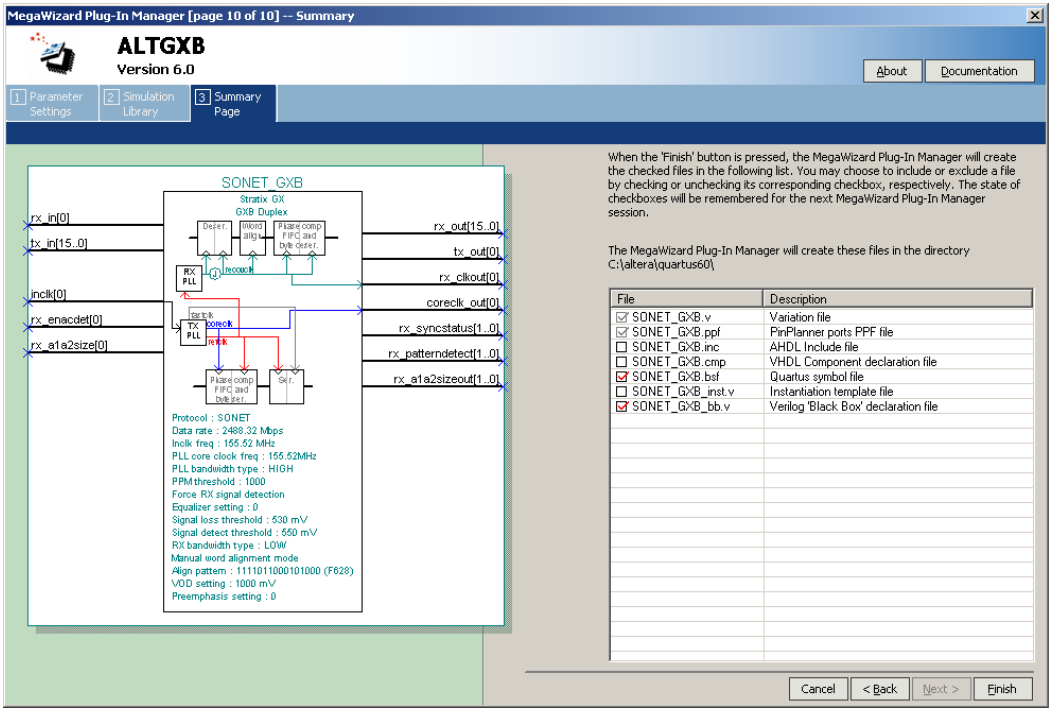


Figure 4–24 shows page 10 of the MegaWizard Plug-In Manager for the SONET protocol set up. You can select optional files on this page. After you make your selections, click **Finish** to generate the files.

Figure 4–24. MegaWizard Plug-In Manager - altgxb (Page 10)



Stratix GX Transceiver Merging

A transceiver block contains four transceivers. In a design, an `altgxb` instantiation is placed in one or more transceiver blocks and potentially leaves unused transceivers in a block. For example, a six transceiver instantiation completely fills one transceiver block and half fills a second, taking up two full transceiver blocks. If another instantiation is in the design, it is placed the same way. For example, an instantiation of two transmitters takes up a third transceiver block. Merging two of the partially filled transceiver blocks into one transceiver block reduces the resources used and allows a design to fit into a device with fewer transceiver blocks.

The `altgxb` MegaWizard Plug-In Manager in the Quartus II software has a feature that allows merging of similar quads (transceiver blocks). With a few exceptions, transceiver blocks can be merged if the options

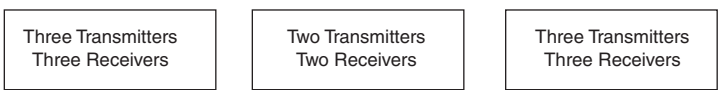
chosen in the wizard are the same. The Quartus II software merges the transceiver blocks automatically if you turn on the merging option for each instantiation. Table 4–10 shows the considerations for merging.

Table 4–10. Stratix GX Merging Considerations

Merging Rules	Required to Match Between Transceiver Blocks	Not Required to Match Between Transceiver Blocks
MegaWizard Plug-In Manager options	USE_8B_10_MODE USE_DOUBLE_DATA_MODE CHANNEL_WIDTH SYNC_MODE DATA_RATE TRANSMIT_PROTOCOL	VOD Pre-emphasis Equalization Number of transmitters Number of receivers
Input control signals must be shared across transceiver blocks and must be from the same source	Clock CRU_CLOCK PLL_RESET PLL_ENABLE	
The merging partial transceiver blocks must reduce the number of transceiver block when merged	The total number of receivers and transmitters must be tx ≤4 x *n* and rx ≤4 x *n*, where *n* is the reduced number of transceiver blocks remaining after merging.	

The Quartus II software does not merge two transceiver blocks if they won't completely fit into one. It can, however, merge three transceiver blocks into two. Figure 4–25 shows a configuration with three transceiver blocks that can potentially be merged.

Figure 4–25. Three Transceiver Configuration



In Figure 4–25, two transceiver blocks cannot merge because there would be extra channels remaining. But because there are three transceiver blocks, the Quartus II software can merge them into two with no remaining channels.

If you turn on the merging option, the Quartus II software automatically merges transceiver blocks if possible or provides a warning during compilation if merging is not possible. The merging feature can reduce the transceiver block resources used in your design.

5. XAUI Mode

SGX52005-1.2

Introduction

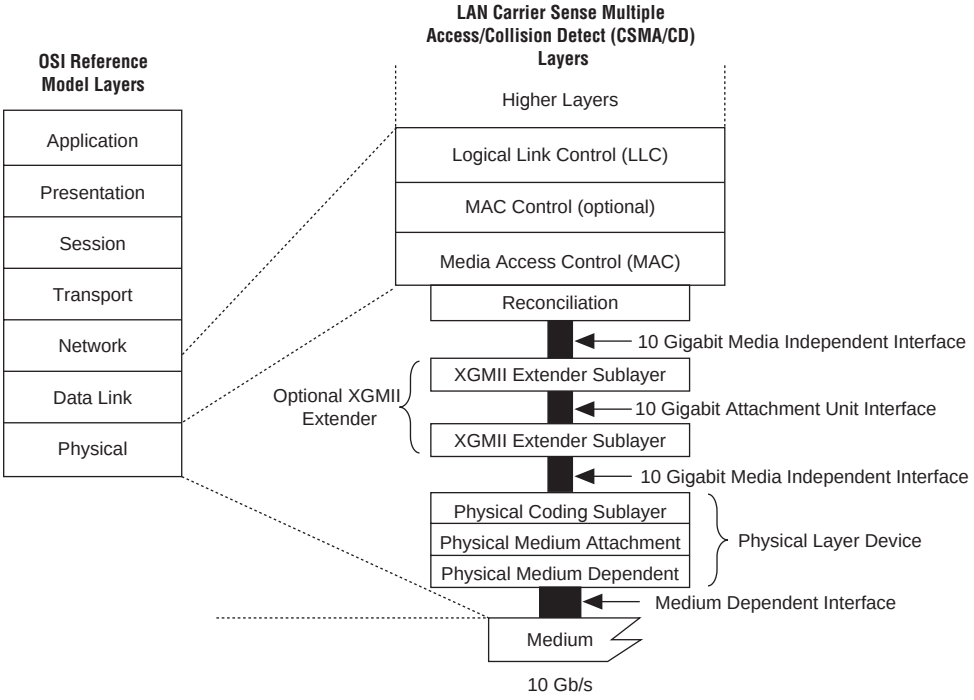
The 10 Gigabit Attachment Unit Interface (XAUI) is an optional, self-managed interface that can be inserted between the reconciliation sublayer and the PHY layer to transparently extend the physical reach of the 10 Gigabit Media Independent Interface (XGMII).

XAUI addresses several physical limitations of the XGMII. XGMII signaling is based on the HSTL class 1 single-ended I/O standard, which has an electrical distance limitation of approximately 7 cm. Because XAUI uses low voltage differential signaling method, the electrical limitation is increased to approximately 50 cm. Another advantage of XAUI is simplification of backplane and board trace routing. XGMII is composed of 32 transmit channels, 32 receive channels, 1 transmit clock, 1 receive clock, 4 transmitter control characters, and 4 receive control characters for a 74-pin wide interface in total. XAUI, on the other hand, only consists of 4 differential transmitter channels and 4 differential receiver channels for a 16-pin wide interface in total. This reduction in pin count significantly simplifies the routing process in the layout design. Figure 5–1 shows the relationships between the XGMII and XAUI layers.

Stratix® GX devices offer the following XAUI features:

- Serial data rate range from 500 Mbps to 3.1875 Gbps
- Input reference clock range from 25 to 637.5 MHz
- Parallel interface width of 16 bits
- 8B/10B encoder and decoder
- Word aligner supports 10-bit code-group
- Channel deskew
- Rate compensation or elastic buffer
- XGMII-to -PCS code conversion on transmit
- PCS-to -XGMII code conversion on receive
- Byte deserializer

Figure 5–1. XGMII & XAUI Relationship to ISO/IEC Open Systems Interconnection (OSI) Reference Model & IEEE 802.3 CSMA/CD LAN Model



As noted earlier, the XGMII interface consists of 4 lanes of 8 bits. At the transmit side of the XAUI interface, the data and control characters are converted within the XGXS into an 8B/10B encoded data stream. Each data stream is then transmitted across a single differential pair running at 3.125 Gbps. At the XAUI receiver, the incoming data is decoded and mapped back to the 32 bit XGMII format. This process provides a transparent extension of the physical reach of the XGMII and also reduces the interface pin count.

XAUI functions as a self-managed interface because code-group synchronization, channel deskew, and clock domain decoupling are handled with no upper layer support requirements. These features are accomplished based on Physical Coding Sublayer (PCS) code-groups that are used during the Inter-Packet Gap (IPG) time and during idle periods. PCS code-groups are mapped by the XGMII Extender Sublayer (XGXS) to XGMII characters, as specified in Table 5–1.

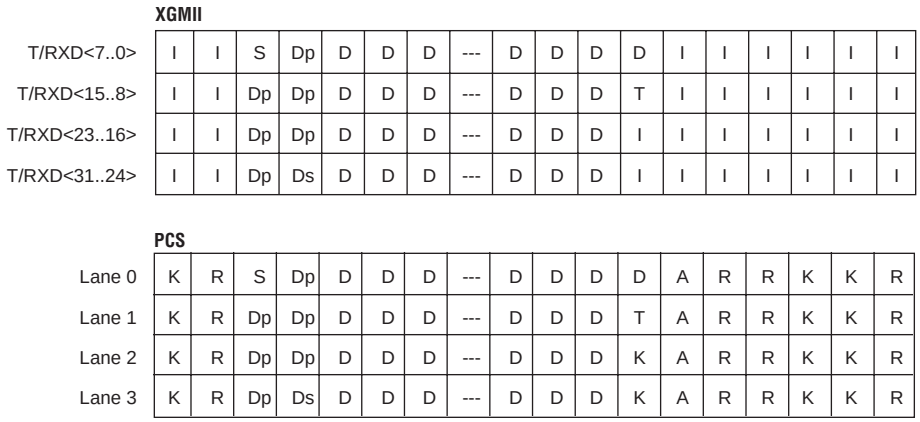
Table 5–1. XGMII Character to PCS Code-Group Mapping *Note (1)*

XGMII TXC	XGMII TXD	PCD Code-Group	Description
0	00 through FF	Dxx,y	Normal data transmission
1	07	K28.0 or K28.3 or K28.5	Idle in ||**I**||
1	07	K28.5	Idle in ||**T**||
1	9C	K28.4	Sequence
1	FB	K27.7	Start
1	FD	K29.7	Terminate
1	FE	K30.7	Error
1	Any other value	K30.7	Invalid XGMII character

Note to Table 5–1:
(1) Values in TXD column are in hexadecimal.

Figure 5–2 shows an example of the mapping between XGMII characters to the PCS code-groups used in XAUI. The idle characters are mapped to a pseudorandom sequence of ||A||, ||R||, and ||K|| code-groups.

Figure 5–2. Example of Mapping XGMII Characters to PCS Code-Groups



The PCS code-groups are sent via PCS ordered sets. PCS ordered sets consist of combinations of special and data code-groups defined as a column of code-groups. These ordered sets are composed of four code-groups beginning in lane 0. Table 5–2 lists the defined idle ordered sets (||I||) that are used for the self managed properties of XAUI.

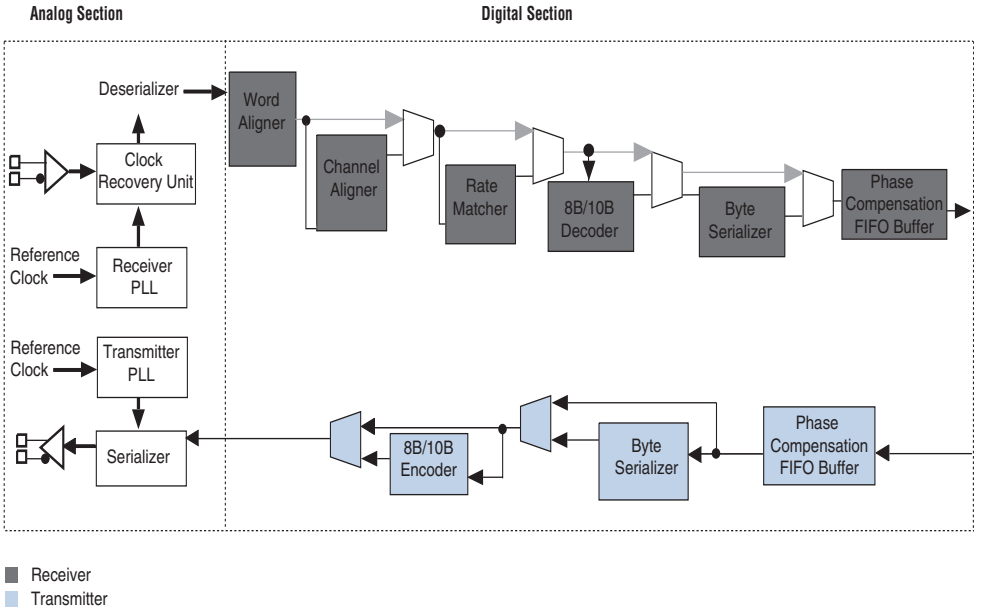
Table 5–2. Defined Idle Ordered Set

Code	Ordered_Set	Number of Code_Groups	Encoding
||I||	Idle		Substitute for XGMII Idle
||K||	Sync column	4	/K28.5/K28.5/K28.5/K28.5/
||R||	Skip column	4	/K28.0/K28.0/K28.0/K28.0/
||A||	Align column	4	/K28.3/K28.3/K28.3/K28.3/

This section briefly introduces XAUI, along with the code-groups and ordered sets associated with this self-managed interface. For full details on the XAUI standard, refer to clauses 47–48 in the 10 Gigabit Ethernet standard (IEEE 802.3ae).

XAUI mode enables the configuring of transceiver blocks to support XAUI synchronization, channel alignment, rate compensation, XGMII to PCS code-group conversion, and PCS code-group-to-XGMII conversion. This section describes the supported digital architecture, clocking schemes, and software implementation of the XAUI mode. Figure 5–3 shows a block diagram of a duplex channel configured in XAUI mode.

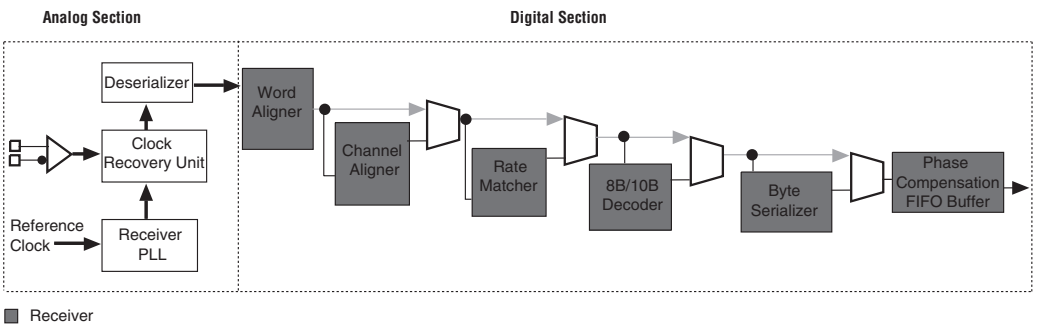
Figure 5–3. Block Diagram of a Duplex Channel Configured in XAUI Mode



XAUI Mode Receiver Architecture

Figure 5–4 diagrams the digital components of the receiver in XAUI mode.

Figure 5–4. Block diagram of Receiver Digital Components in XAUI Mode

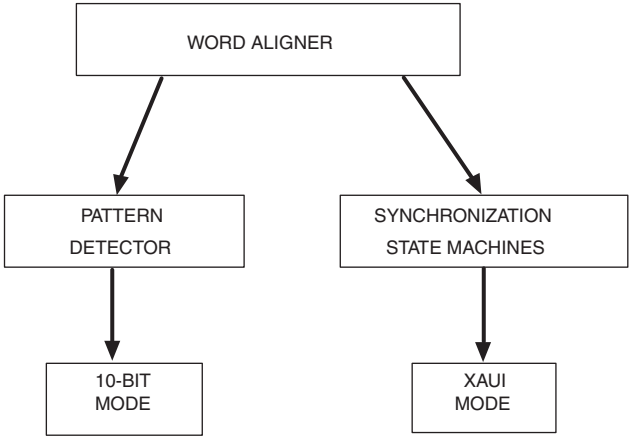


Word Aligner

For embedded clocking schemes, the clock is recovered from the incoming data stream based on the transition density of the data. This feature eliminates the need for you to factor in receiver skew margins between the clock and data. However, with this clocking methodology, the word boundary of the re-timed data can be altered. Stratix GX transceivers offer an embedded word alignment circuit that can be used in conjunction with the pattern detector to align the word boundary of the re-timed data to a specified comma. You can configure this embedded circuit to synchronize with the XAUI protocols.

The word aligner is composed of a pattern detector and synchronization state machines. The word aligner cannot be bypassed, but if the `rx_enacdet` signal is not used, the word aligner does not alter the data. Figure 5–5 shows the various components of the word aligner. The functionality of each component is described in the following sections.

Figure 5–5. Stratix GX Word Aligner Components



Pattern Detector Module

The pattern detector matches a pre-defined comma to the current byte boundary. If the comma is found, the optional `rx_patterndetect` signal is asserted for the duration of one clock cycle to signify that the comma exists in the current word boundary. The pattern detector module indicates only that the signal exists and does not modify the word boundary. Modification of the word boundary is discussed in the word alignment and synchronization sections. You can program a 10-bit pattern for the pattern detector to recognize.

This module matches the 10-bit comma specified in the MegaWizard® Plug-In Manager with the data and its complement in the current word boundary. Both positive and negative disparities are checked in this mode. For example, if a /K28.5/ (b'0011111010) pattern is specified as the comma, the rx_patterndetect signal is asserted if b'0011111010 or b'1100000101 is detected in the incoming data.

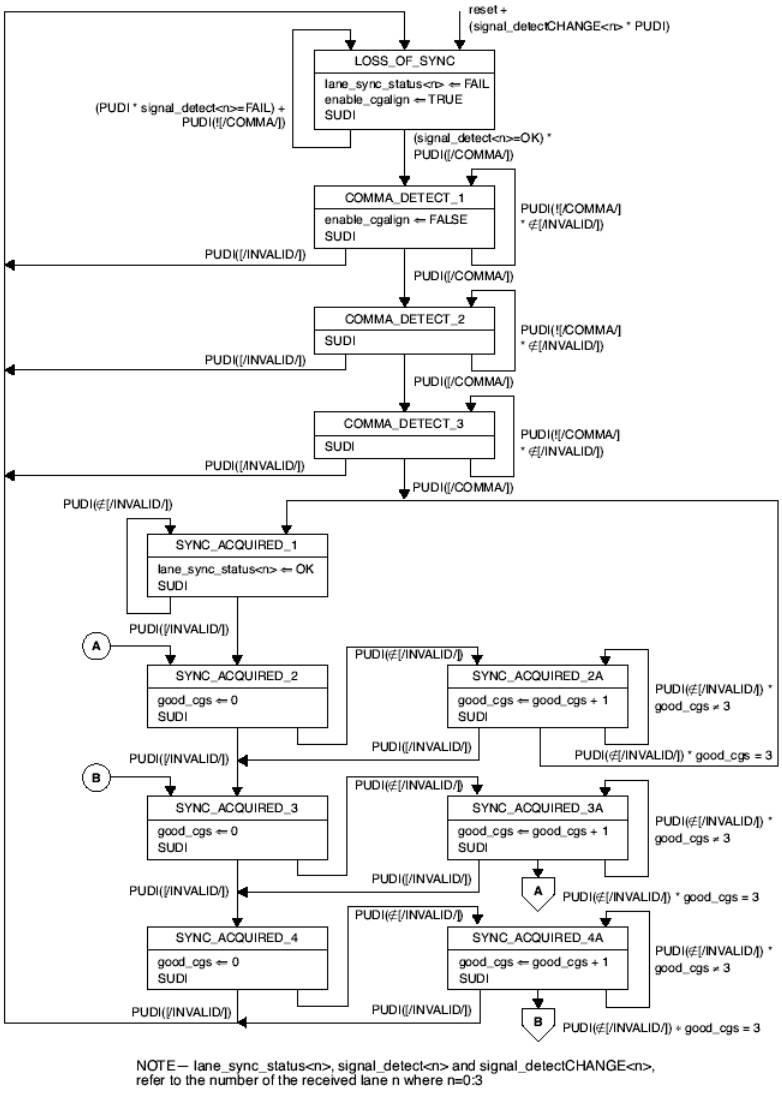
XAUI uses an embedded clocking scheme that re-times the data, which can also alter the code-group boundary. The boundaries of the code-groups are realigned through a synchronization process specified in clause 48 of the IEEE 802.3ae standard, which states that synchronization is achieved upon the reception of four /K28.5/ commas. Each comma can be followed by any number of valid code-groups. Invalid code-groups are not supported during the synchronization stage.

XAUI Synchronization Mode

When a Stratix GX transceiver is configured to the XAUI protocol, the built-in pattern detector, word aligner, and XAUI state machines adhere to the PCS synchronization specification. The code-group synchronization is achieved upon the reception of four /K28.5/ commas. Each comma is followed by any number of valid code-groups. Invalid code-groups are not supported during the synchronization stage. When code-group synchronization is achieved, the optional rx_syncstatus[] signal is asserted. Refer to clause 47-48 of the IEEE P802.3ae standard for more information regarding the operation of the synchronization phase. If the rx_sigdet signal is valid and the reset is deasserted, the synchronization state machine begins the synchronization process. If either of the two signals are not valid, the state machine falls out of the synchronization process and waits for the valid rx_sigdet signal and reset.

For reference, the PCS synchronization state diagram specified in clause 48 of the IEEE P802.3ae is shown in Figure 5–6.

Figure 5–6. IEEE 802.3ae PCS Synchronization State Diagram



Note to Figure 5–6:

(1) `lane_sync_status<n>`, `signal_detect<n>`, and `signal_detectCHANGE<n>` refer to the number of the received lane n where n = 0 to 3.

Channel Aligner

You use the channel aligner when implementing the XAUI protocol, to ensure that the channels are aligned. The channel aligner uses a 16-word-deep FIFO module.

Ordered sets can be misaligned with respect to one another because of board skew or differences between the independent clock recoveries per serial lane. Channel alignment re-aligns the ordered sets. This process is commonly referred to as deskew or channel bonding. Channel alignment is accomplished by using the alignment code-group, referred to as /A/. The /A/ code-group is transmitted simultaneously on all four lanes, constituting an ||A|| ordered set, during idles or inter packet gaps (IPG). XAUI receivers use these code-groups to resolve any lane to lane skew. Skew between the lanes can be up to 40 UI (12.8ns) as specified in the standard, which relaxes the board design constraints. Figure 5–7 shows lane skew at the receiver input, and how the deskew circuitry uses the /A/ code-group to deskew the channels.

Figure 5–7. Example of Lane Skew at Receiver Input

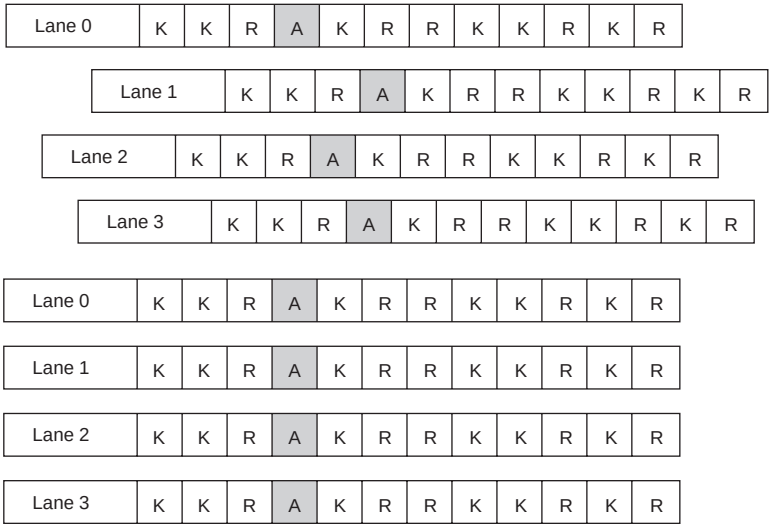
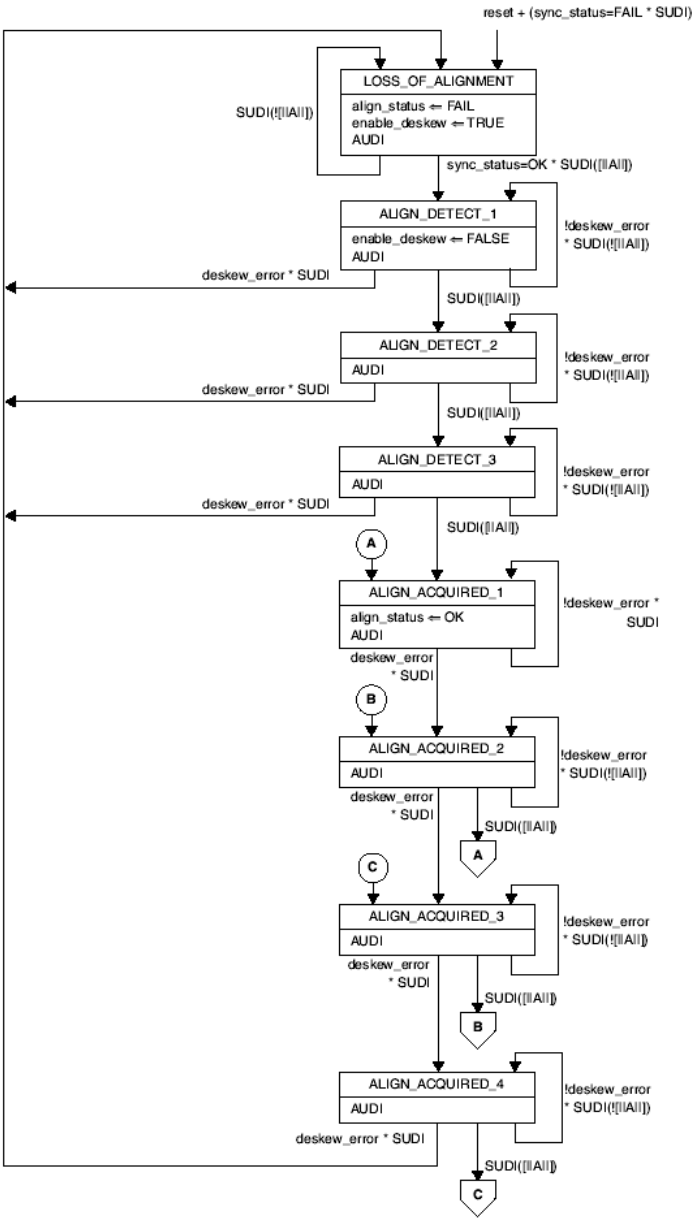


Figure 5–8. IEEE802.3ae PCS Deskew State Diagram



Stratix GX transceivers handle XAUI channel alignment with a dedicated deskew macro consisting of a 16-word-deep FIFO module that is controlled by a XAUI deskew state machine. The XAUI deskew state machine first looks for the /A/ code-group within each channel. When /A/ is detected in each channel, the deskew FIFO module is enabled. The deskew state machine then monitors the reception of /A/ code-groups. When four aligned /A/ code-groups are received, the `rx_channelaligned[]` signal is asserted. The deskew state machine continues to monitor the reception of /A/ code-groups and deasserts the `rx_channelaligned[]` signal if alignment conditions are lost. This built-in deskew macro is only enabled for the XAUI protocol. For reference, the PCS deskew state diagram specified in clause 48 of the IEEE P802.3ae is shown in Figure 5–8.

Rate Matcher

XAUI can operate in multi-crystal environments, which can tolerate a frequency variation of ±100 ppm between crystals. Stratix GX transceivers have embedded circuitry to perform clock rate compensation. This is achieved by the insertion or removal of the PCS SKIP code-group (/R/) from the inter packet gap (IPG) or idle stream. This process is called *rate matching* and is sometimes referred to as clock rate compensation.

The rate matcher in Stratix GX transceivers consists of a 12-word-deep FIFO module along with control logic. In XAUI mode, the controller begins to write data into the FIFO module whenever the `rx_channelaligned` signal is asserted. Within the control logic, there is a FIFO module counter that keeps track of the read and write executions. When the FIFO module is near an overflow condition, the receivers delete the /R/ code-group simultaneously across all channels during IPG or idle conditions. If the FIFO counter is near an underflow condition, the receivers insert the /R/ code-group simultaneously across all channels during IPG or idle conditions. This circuitry compensates for ±100 ppm frequency variations.

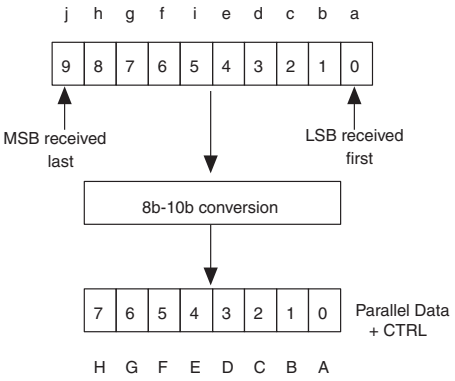
8B/10B Decoder

The 8B/10B decoder is part of the Stratix GX transceiver blocks. The purpose of the 8B/10B decoder is to restore the 8-bit data + 1-bit control identifier from the 10-bit code.

10-Bit Decoding

The 8B/10B decoder translates the 10-bit encode code into the 8-bit equivalent data or control code. The 10-bit encoded code is received LSB to MSB. The data that is received must be from the supported Dx.y or Kx.y list. All 8B/10B control signals (disparity error, control detect, and code error) are pipelined with the data in the Stratix GX receiver block and are edge-aligned with the data. Figure 5–9 diagrams the 10- to 8-bit conversion.

Figure 5–9. 10-Bit to 8-Bit Conversion



Reset

The `rxdigitalreset` signal governs the reset condition of the 8B/10B decoder. In reset, the disparity registers are cleared. Upon exiting reset, the 8B/10B decoder can start with either a positive or negative disparity. The decoder calculates the initial running disparity based on the first valid code received.

The receiver block must be word-aligned after reset before the 8B/10B decoder can decode valid data or control codes.

Code Error Detect

The `rx_errdetect` signal indicates when the code received contains an error. This port is optional and if not in use, there is no way to detect whether a code received is valid or not. The `rx_errdetect` goes high if code that is received is invalid or if it contains a disparity error. If code that is received is not part of the valid Dx.y or Kx.y list, the `rx_errdetect` signal goes high. This signal is aligned to the invalid code word that is received at the FPGA logic array.

Disparity Error Detector

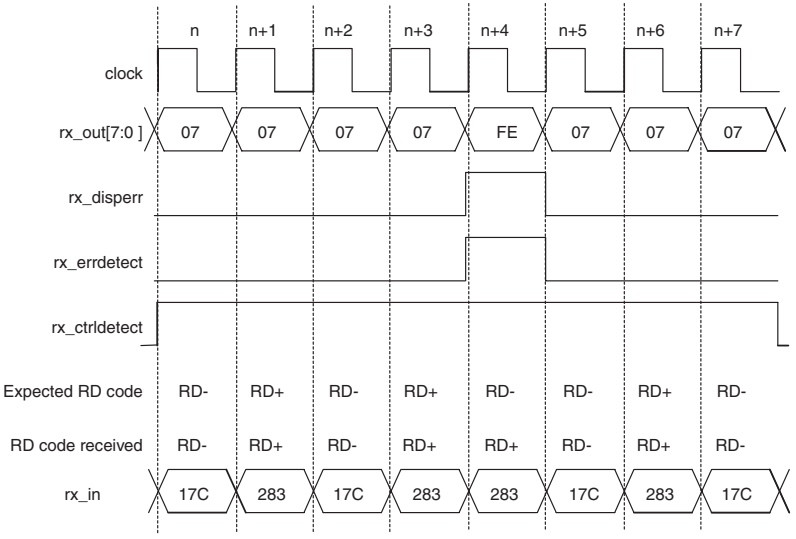
The 8B/10B decoder detects disparity errors based on which 10-bit code it received. The disparity error is indicated at the optional `rx_disperr` port. The current running disparity is based on the disparity calculation of the last code received. The disparity calculation is described in Appendix A, Data & Control Codes.

If negative disparity is calculated for the last 10-bit code, a neutral or positive disparity 10-bit code is expected. If the decoder does not receive a neutral or positive disparity 10-bit code, the `rx_disperr` signal goes high, which indicates that the code received had a disparity error.

If a positive disparity is calculated, a neutral or negative disparity 10-bit code is expected. `Rx_disperr` goes high if the code received is not as expected. When the `rx_disperr` signal is high, the `rx_errdetect` signal also transitions high.

In XAUI mode, idle character 8'h07 is converted to 8'hbc, 8'h1c, or 8'h7c. Figure 5–10 shows 8'hbc encoded into RD+ and RD-, and also shows a case where the disparity is violated. A K28.5 code has an 8-bit value of `8'hbc` and a 10-bit value (`jhgfiedcba`). The 10-bit value is 10'b0011111010 (10'h17c) for RD- or 10'b1100000101 (10'h283) for RD+. Assume that the running disparity at time n-1 is negative, so the expected code at time *n* is from the RD- column. Since a K28.5 does not have a balanced 10-bit code (equal number of 1's and 0's), the expected RD code toggles back and forth between RD- and RD+. At time n+3, the 8B/10B decoder received a RD+ K28.5 code (10'h283), which makes the current running disparity negative. At time n+4, because the current disparity is negative, a K28.5 from the RD- column is expected, but a K28.5 code from the RD+ is received instead. This code prompts `rx_disperr` to go high during time n+4 to indicate that the K28.5 code had a disparity error. The current running disparity at the end of time n+4 is negative, because a K28.5 from the RD+ column was received. Based on the current running disparity at the end of time n+5, a positive disparity K28.5 code (from the RD-) column is expected at time n+5.

Figure 5–10. Disparity Error

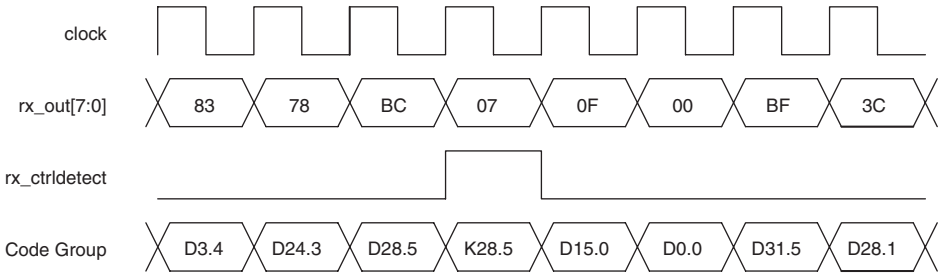


Control Detect

The 8B/10B has the ability to differentiate between data and control codes via the `rx_ctrldetect` port. If this port is not in use, there is no way to differentiate Dx.y from Kx.y.

Figure 5–11 shows an example waveform demonstrating the detection of a K28.5 code (BC + ctrl). The `rx_ctrldetect`=1'b1 is aligned with `8'hbc`, which indicates that this code is a control code. The reset of the code received is data.

Figure 5–11. Control Code Detection



PCS - XGMII Code Conversion

In XAUI mode, the 8b/10b decoder in Stratix GX transceivers is controlled by a global receiver state machine that maps various PCS code-groups into specific 8-bit XGMII codes. Table 5–3 lists the PCS code group to XGMII character mapping.

Table 5–3. PCS Code-Group to XGMII Character Mapping *Note (1)*

XGMII RXC	XGMII RXD	PCS Code Group	Description
0	00 through FF	Dxx.y	Normal data reception
1	07	K28.0 or K28.3 or K28.5	Idle in ||**I**||
1	07	K28.5	Idle in ||**T**||
1	9C	K28.4	Sequence
1	FB	K27.7	Start
1	FD	K29.7	Terminate
1	FE	K30.7	Error
1	FE	Invalid code-group	Received code-group

Note to Table 5–3:
(1) Values in RXD column are in hexadecimal.

Byte Deserializer

The byte deserializer module reduces the speed at which the FPGA logic array must run to meet performance specifications. It is possible to bring the data rate down from the line rate by a factor of 20.

The input to this module is 8 bits of data; the output to the FPGA logic array is deserialized to 16 bits. The byte deserializer does not process the data, and therefore, the control signals fed to the module are simply processed to match the latency to the data.

The byte deserializer in the receiver block takes in up to 13 bits. It is possible to feed the following to the byte deserialzer:

- 8 bits of data and up to 2 control signals (`rx_patterndetect`, `rx_syncstatus`)
- 8 bits of data and up to 5 control signals (`rx_patterndetect`, `rx_syncstatus`, `rx_disperr`, `rx_ctrldetect`, and `rx_errdetect`)

The byte deserializer outputs up to 26 bits, depending on the number of bits that was passed to it. When the input includes data and control signals, the data and the control signals are deserialized to include double the data bits and 2 bits of each control signal, one for the MSB and one for the LSB. This case is shown in the XAUI mode where the inputs to the Byte Deserializer are `datain[7..0]`, `patterndetect`, `syncstatus`, `disperr`, `ctrldet`, and `errdet`. These 13 input signals feeding the byte deserializer and 26 output signals are fed to the FPGA logic array. The signals are sent into the logic array as two 13-bit buses. The aggregate bandwidth does not change by using the byte deserializer because the logic array data width is doubled.

Figure 5–12 demonstrates input and output signals of the byte deserializer when deserializing an 8-bit data input to 16 bits. In this case, the alignment pattern B (10111100) is located in the MSB of the 16-bit output, and this is reflected with `patterndetect[1]` going high. The output of the byte deserializer is BA, DC, FE, and so on.

Figure 5–12. Receiver Byte Deserializer in 8/16-Bit Mode with Alignment Pattern in MSB

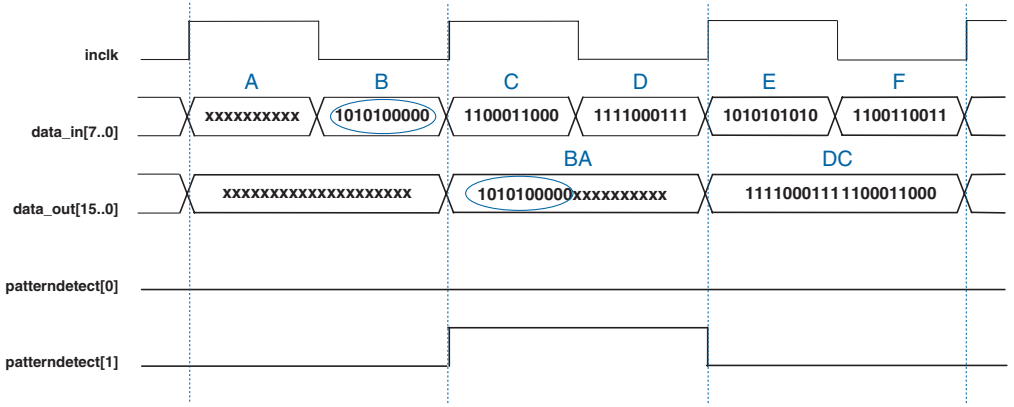
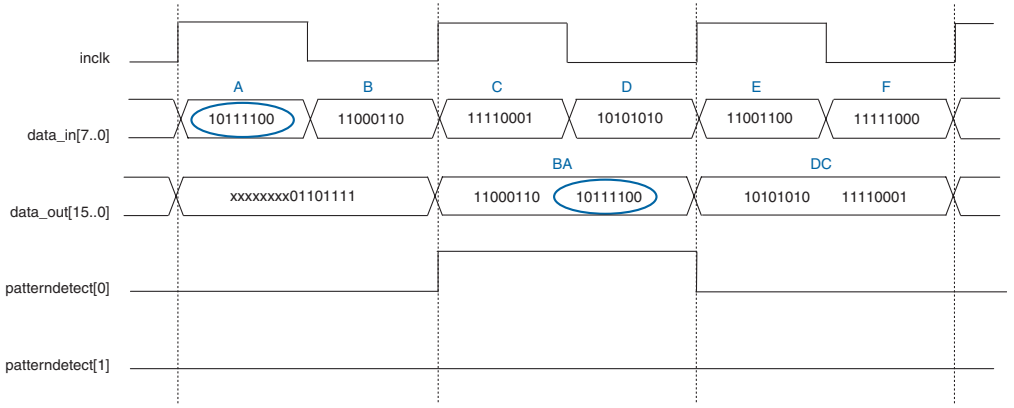


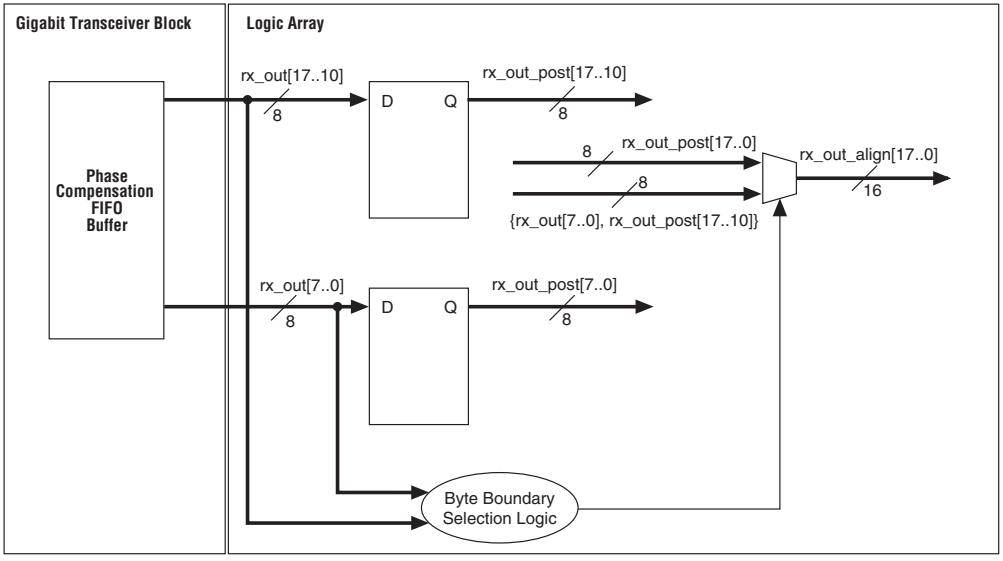
Figure 5–13 demonstrates the alternate case of the alignment pattern found in the LSB of the 16-bit output. Correspondingly, `patterndetect[0]` goes high. In this case, the output is BA, DC, FE, and so on.

Figure 5–13. Receiver Byte Deserializer in 8/16-Bit Mode With Alignment Pattern in LSB



The logic array must include logic to perform byte position alignment when the data enters the logic array, as seen in Figure 5–14. In this example, the byte position selection logic determines the proper byte position based on the pattern detect signal.

Figure 5–14. Receiver Byte Deserializer Data Recovery in Logic Array



Receiver Phase Compensation FIFO Module

The receiver phase compensation FIFO module is located at the FPGA logic array interface in the receiver block and is four words deep. The FIFO module compensates for the phase difference between the clock in the FPGA and the operating clocks in the transceiver block.

In XAUI mode, the write port is clocked by the `refclk` from the transmitter PLL. This clock is half the rate if the byte deserializer is used. The read clock is clocked by `coreclk` (output from the transmitter PLL).

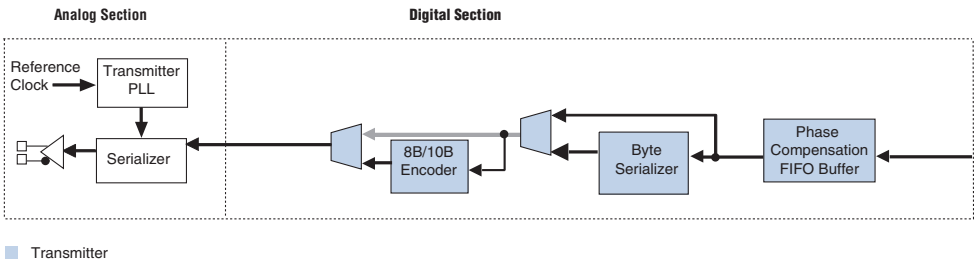
The receiver phase compensation FIFO module only accounts for phase differences.

The receiver phase compensation FIFO module is always used and cannot be bypassed.

XAUI Mode Transmitter Architecture

Figure 5–15 diagrams the transmitter digital components in XAUI mode.

Figure 5–15. Block Diagram of Transmitter Digital Components in XAUI Mode



Transmitter Phase Compensation FIFO Module

The Transmitter Phase Compensation FIFO module is located at the FPGA logic array interface in the transmitter block and is four words deep. The FIFO module compensates for the phase difference between the clock in the FPGA and the operating clocks in the transceiver block.

The read port of the phase compensation FIFO module is clocked by the transmitter PLL clock. The write clock is clocked by `tx_coreclk`. You can select the `tx_coreclk` as an optional transmitter input port to

receive a clock supply. In this case, there must be no frequency difference between the tx_coreclk and the transmitter PLL clock. The transmitter Phase Compensation FIFO module can only account for phase differences.

If the tx_coreclk is not selected as an optional input transmitter port, tx_coreclk is fed by coreclk_out. This connection occurs using the logic array routing. As such, the software defaults to using an FPGA global clock, regional clock, or fast regional clock resource.

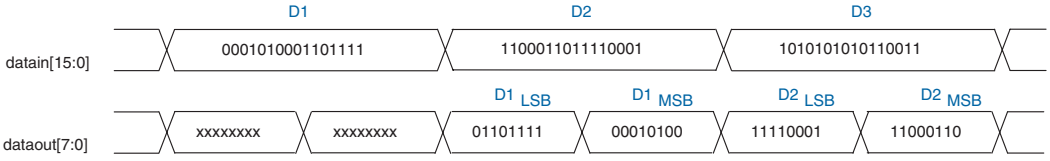
The transmitter phase compensation FIFO module is always used and cannot be bypassed. The input to this FIFO module is the data from the FPGA logic array. If they are used, the tx_ctrlenable and tx_forcedisparity signals are also passed through the FIFO module to ensure that they are synchronized with the data when they feed to the subsequent module.

Byte Serializer

The byte serializer in the transmitter block takes a 16-bit input from the FPGA logic array and serializes it to 8 bits. It transmits from the least significant byte to the most significant byte. The transmitter digital reset must always be used to reset the FIFO module pointers whenever an unknown state is encountered, such as when the transmitter PLL loses lock. Refer to the chapter *Reset Control & Power Down* for further details on the reset sequence.

Figure 5–16 demonstrates input and output signals of the byte serializer when serializing a 16-bit input to 8 bits. The tx_in[] signal is the input that has already passed from the FPGA logic array through the transmitter phase compensation FIFO module.

Figure 5–16. Transmitter Byte Serializer in 16- to 8-Bit Mode



The LSB is transmitted before the MSB in the transmitter byte serializer. Figure 5–16 shows the order of data transmitted. For the input of D1, the output is D1LSB and then D1MSB.

XGMII Character to PCS Code-Group Mapping

In XAUI mode, the 8b/10b encoder in the Stratix GX transceiver is controlled by a global transmitter state machine that maps various 8-bit XGMII codes to 10-bit PCS code-groups. This state machine complies with the IEEE 802.3ae PCS transmit specification. For reference, the PCS transmit source state diagram, specified in clause 48 of the IEEE P802.3ae specification, is shown in Figure 5–17.

Figure 5–17. IEEE 802.3ae PCS Transmit Source State Diagram

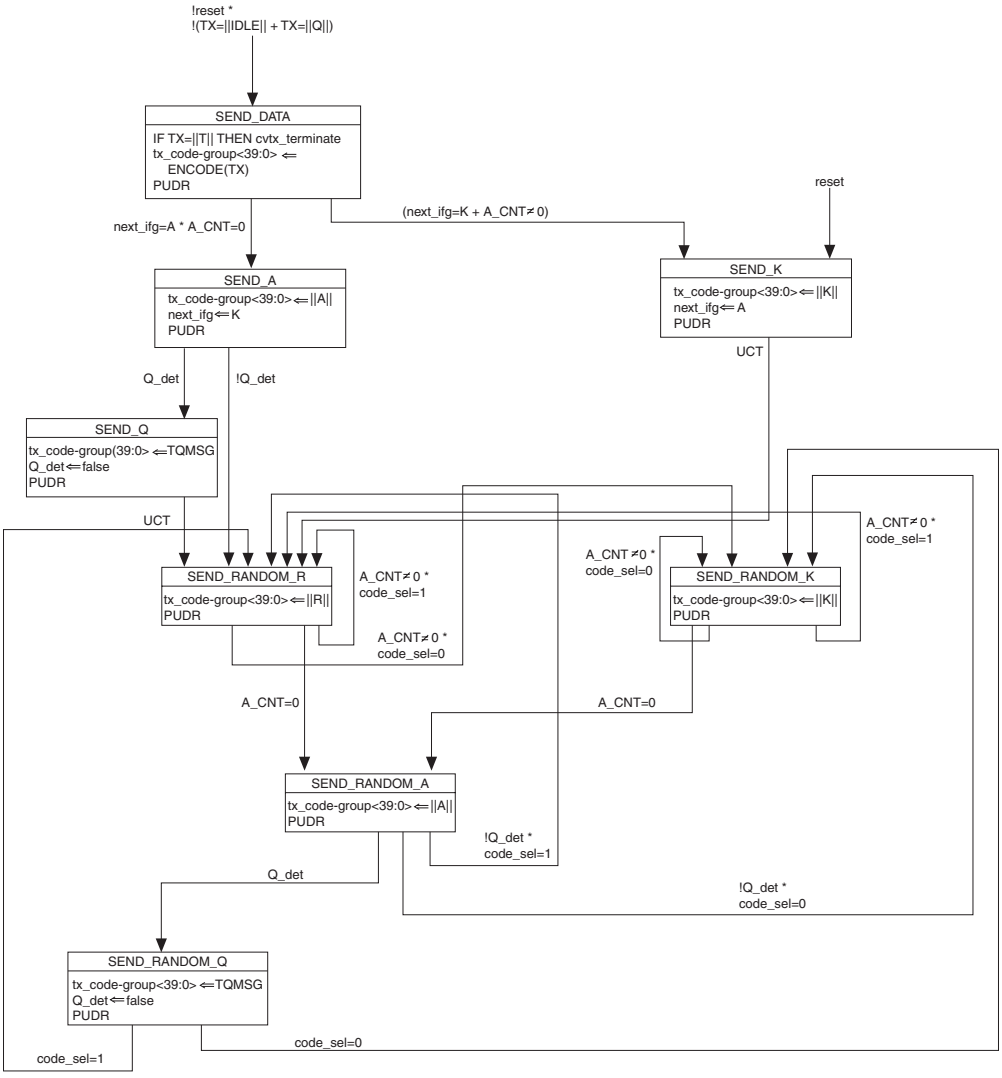


Table 5–4 lists the XGMII character-to -PCS code-group mapping.

Table 5–4. XGMII Character to PCS Code-Group Mapping *Note (1)*

XGMII TXC	XGMII TXD	PCS Code-Group	Description
0	00 though FF	Dxx,y	Normal data transmission
1	07	K28.0 or K28.3 or K28.5	Idle in ||**I**||
1	07	K28.5	Idle in ||**T**||
1	9C	K28.4	Sequence
1	FB	K27.7	Start
1	FD	K29.7	Terminate
1	FE	K30.7	Error
1	Any other value	K30.7	Invalid XGMII character

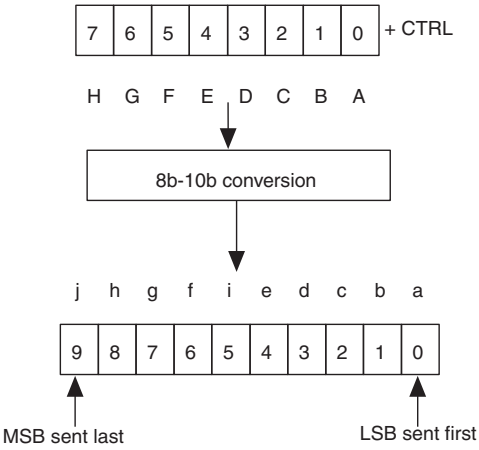
Note to Table 5–4:
(1) Values in TXD column are in hexadecimal.

8B/10B Encoder

The 8B/10B encoder is part of the Stratix GX transceiver blocks. The purpose of the 8B/10B encoder is to translate 8-bit data and a 1-bit control identifier (via `tx_ctrlenable`) into a 10-bit DC balanced data stream.

For additional information about the 8B/10B code itself, refer to the *Data & Control Codes* chapter of the *Stratix GX Device Handbook, Volume 2*. The 8B/10B encoder translates the 8-bit data or 8-bit control character to its 10-bit equivalent. The conversion format is shown in Figure 5–18. The10-bit resultant data is transmitted LSB first by the serializer.

Figure 5–18. 8B/10B Conversion Format



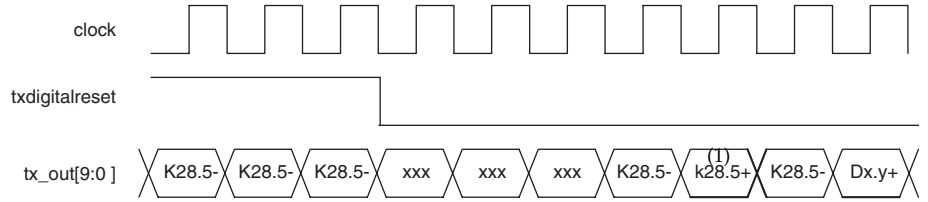
8B/10B Reset Condition

The `txdigitalreset` controls the reset of the 8B/10B encoder. To reset the 8B/10B encoder, `txdigitalreset` must be high. During reset, the running disparity registers are cleared, along with the data registers. Also, the 8B/10B encoder outputs a K28.5 pattern from the RD- column continuously until `txdigitalreset` goes low. The `tx_in[]` and `tx_ctrlenable[]` are ignored during the reset state. Once out of reset, the 8B/10B encoder starts with a bias towards negative disparity (RD-) and transmits three K28.5 codes for synchronizing before it starts encoding and transmitting the data on `tx_in[]`.

If the reset for the 8B/10B encoder is asserted, the 8B/10B decoder receiving the data may receive an invalid code error, sync error, control detect, and/or disparity error while `txdigitalreset` is high.

Figure 5–19 shows the reset behavior of the 8B/10B encoder. When in reset (`txdigitalreset` is high) a K28.5- (K28.5 10-bit code from the RD- column) is sent continuously until `txdigitalreset` goes low. Because of pipelining of the transmitter channel, there are several don't-care values (10'hxxx) until the first of three K28.5 is sent (Figure 5–19 shows three don't-cares). Normal user data follows the third K28.5.

Figure 5–19. Transmitter Output During Reset Conditions

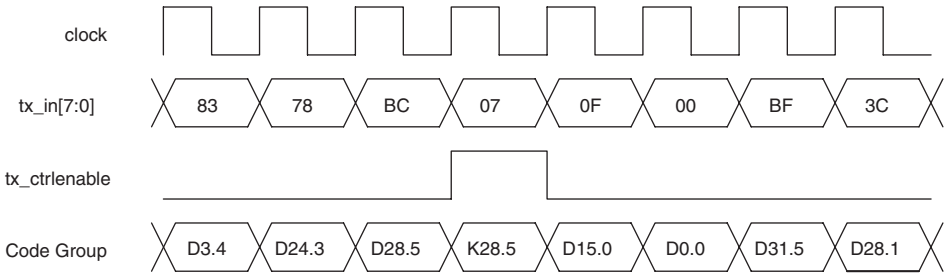


Note to Figure 5–19:
(1) K28.5 is an example, but an 07 control generates an idle sequence based on the 802.3 specification.

Control Code Encoding

The `tx_ctrlenable[]` signal dictates when a control code is to be inserted in the encoded data flow. When `tx_ctrlenable[]` is low, the byte at `tx_in[]` is encoded as data. When `tx_ctrlenable[]` is high, `tx_in[]` is encoded as a control word. The waveform in Figure 5–20 shows that 0x07 is encoded as a control code. The other values of `tx_in[]` are encoded as data.

Figure 5–20. Control Word Identification Waveform



The 8B/10B encoder does not check to see if the code word that is entered is one of the 12 valid codes. If an invalid control code is entered, the resulting 10-bit code might be encoded as an invalid code, which does not map to a valid Dx.y or Kx.y code), or a valid Dx.y code, depending on the value entered.

An example is the invalid encoding of a K24.1 (data = 8'h38 + tx_ctrlenable = 1'b1). Depending on the current running disparity, you can encode the K24.1 to be 10'b0110001100 (0x18C), which is equivalent to a D24.6+ (0xD8 from the RD+ column). An 8B/10B decoder decodes this value incorrectly (based on the 8B/10B Fibre Channel specification).

XAUI Mode Clocking

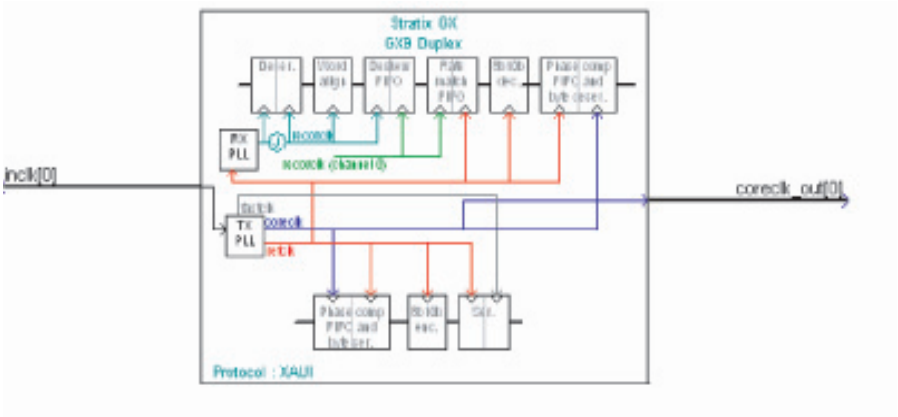
This section describes the clocking supported by the Stratix GX device in XAUI mode.

XAUI Mode Channel Clocking

This section describes clocking of the transceiver, internal clocking details, and external clock ports in XAUI mode. Each block diagram shows the input and output port clocks. Most of the settings are based on per transceiver block (4 channels) basis. By default, the MegaWizard Plug-In Manager selects a set of clocks for transmitters and receivers in a transceiver block when XAUI mode is selected. The MegaWizard Plug-In Manager also offers clock options other than the default selection, which facilitates the clocking scheme.

Figure 5–21 shows that the altgxb megafunction is configured such that the train receiver PLL with transmitter PLL is enabled. The transmitter PLL is fed from an inclk port that can itself be fed from a dedicated REFCLKB, global clock, regional clock, or fast regional clock source. The receiver logic is clocked by the recovered clock from the clock recovery unit up to a deskew FIFO module in the data path. Rate matching is done between recovered clock of channel 0 and refclk from the transmitter PLL. The data from the receive parallel interface, which is also from the phase compensation FIFO module, is clocked by coreclk_out from the transmitter PLL. On the transmitter channel, the output of the transmitter PLL, coreclk_out, is sent out of the logic array as an output and also loops back to clock the write side of the transmit phase compensation FIFO module and the read side of the receive phase compensation FIFO module.

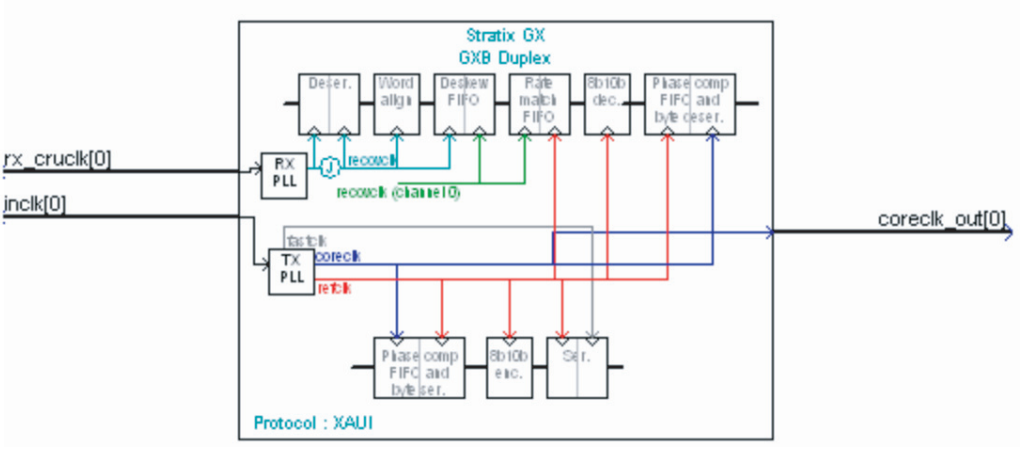
Figure 5–21. Default Configuration of altgxb Megafunction in XAUI Mode



You can disable the train receiver PLL CRU clock from transmitter PLL feature in the `altgxb` MegaWizard Plug-In Manager. Deselecting this option enables an additional `rx_cruclk` input reference clock port for the receiver PLL. You can use this feature to support additional multiplication factors for the receiver PLL, because it supports the separation of receiver and transmitter reference clocks. This separation is necessary if the output reference clock frequency from the transmitter PLL exceeds the 325 MHz phase frequency detector of the receiver PLL (for more information, refer to the *Stratix GX Analog Description* chapter of the *Stratix GX Device Handbook, Volume 2*). This configuration is shown in Figure 5–22.

Refer to the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1* for information on parallel interface speeds for other device speed grades.

Figure 5–22. Train Receiver PLL CRU Clock From Transmitter PLL Feature Is Disabled With Added Port RX_CRUCLK



If `tx_coreclk` is enabled and the train receiver CRU clock from transmitter PLL is disabled, and if other default options are also enabled, this configuration has an independent `rx_cruclk` that feeds the receiver PLL reference clock. This input clock port is only available when the receiver PLL is not trained by the transmitter PLL.

You can enable the write clock of the transmitter phase compensation FIFO module to manually feed in a clock from the FPGA logic array. You can use this option to optimize the global clock usage. For instance, if all transmitter channels between transceiver blocks are from a common clock domain, the transceiver instantiations use a total of one global resource instead of one global per transceiver block if the `tx_coreclk` option is not enabled. On the transmitter functionality screen under the optional port of transmitter section, if `tx_coreclk` is selected as an input port, the default clocking scheme changes by using `tx_coreclk` as the write clock for the phase compensation FIFO module.

There are two ways to connect `tx_coreclk`. To use `coreclk_out`, connect `coreclk_out` to `tx_coreclk` by using either `gclk/rclk/fclk` or logic array routing. Alternatively, `tx_coreclk` can be supplied from a crystal or any other clock source, as long as `tx_coreclk` is frequency-locked to the read side of the phase compensation FIFO module on the transmit side.

The tx_coreclk must be frequency matched with its respective read ports. The phase compensation FIFO module can only correct for phase, not for frequency differences. The receiver parallel interface clocks the data to the FPGA based on coreclk_out, which is the default option in the MegaWizard Plug-In Manager.

Figure 5–23 shows the clock configuration with these optional input ports enabled.

Figure 5–23. TX_CORECLK Enabled With RX_CRUCLK Port

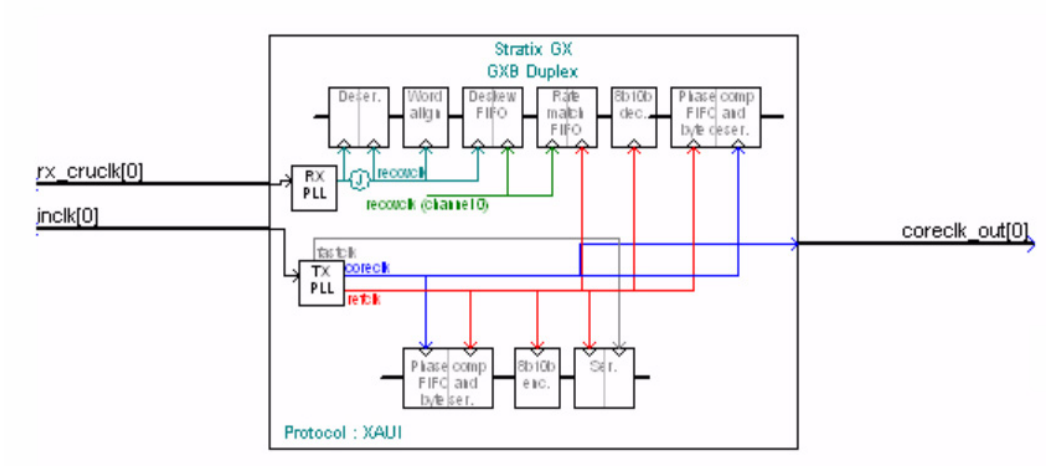


Table 5–5 describes the input and output ports shown in Figure 5–23.

Table 5–5. List of Input & Output Ports Available in XAUI Mode

Clock	Port	Description
inclk	Input	Input to the transmitter PLL. Available as a port when the transmitter PLL is instantiated.
rx_cruclk	Input	Input to CRU. Available as a port when CRU is not trained by the transmitter PLL.
tx_coreclk	Input	Clocks write port of the transmitter phase compensation FIFO module. Optional port in the Quartus® II software. Must be frequency matched to tx_pll_clk. If not available as a port, is fed by coreclk_out through logic array routing.
coreclk_out	Output	Output clock from the transmitter PLL equivalent to tx_pll_clk. Available as port if the transmitter PLL is used.

XAUI Inter-Transceiver Block Clocking

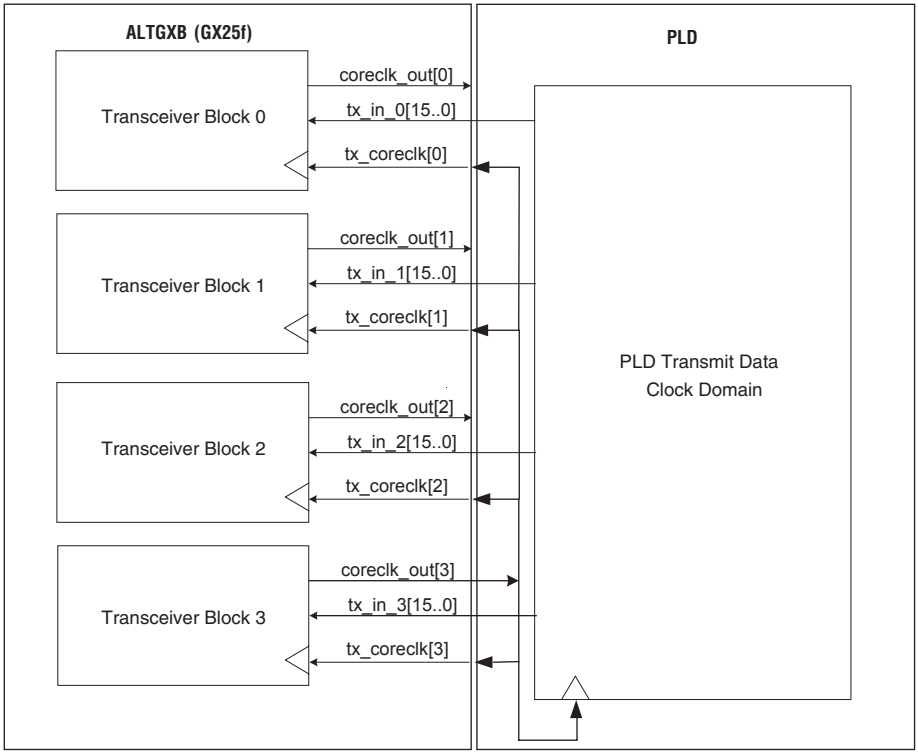
This section describes guidelines for the transceiver interface clocking that is used inside the FPGA logic array when multiple transceiver blocks are active. The transceiver blocks for each mode are supported by transceiver-to-FPGA interface clocking, unique to the Stratix GX transceiver. Different input and output clocks are available based on the options provided by the Quartus II MegaWizard Plug-In Manager's built-in functions. The number of supported channels varies based on the type of Stratix GX device you select (for example, EP1SGX40G, EP1SGX25F, and so on). Consider the clocking schemes at a system level with multiple lanes carefully to prevent pitfalls later in the design cycle. XAUI mode is transceiver-block-based and can only support lanes in multiples of four.

One of the clocking interfaces in the Stratix GX device is the interface between the transceiver and the FPGA, which can be further divided into FPGA-transmit of a transceiver and FPGA-receive of a transceiver. In XAUI mode, depending on the options set in the MegaWizard Plug-In Manager, you can use either the `coreclk_out` or `tx_coreclk` clock to send the data into the transmit of the transceiver. However, the `tx_coreclk` must be frequency locked with the transmit system clock of each transceiver block. (In each transceiver block, one transmitter PLL is shared among four transmitters.)

In a multi-transceiver block scenario, if there are synchronous data transfers based on transmit clocks when `tx_coreclk` is enabled for each channel, each enabled transceiver block must connect to one of the `coreclk_out` outputs. When `tx_coreclk` is not enabled, the Quartus II software automatically routes the `coreclk_out` signal to write the clock of the phase compensation FIFO module using a global, regional, or fast regional resource. In a multi-transceiver block configuration, this feature can lead to timing violations because the `coreclk_out` per transceiver block cannot guarantee a phase relationship. For this reason, Altera recommends clocking the `tx_coreclk` with a common clock for synchronous transmission.

In the multi-transceiver block case, use the transmit clock by enabling `tx_coreclk` and connecting one of the `coreclk_out` clock signals output from one of the transceiver blocks that is active. An illustration of this scheme is shown in Figure 5–24.

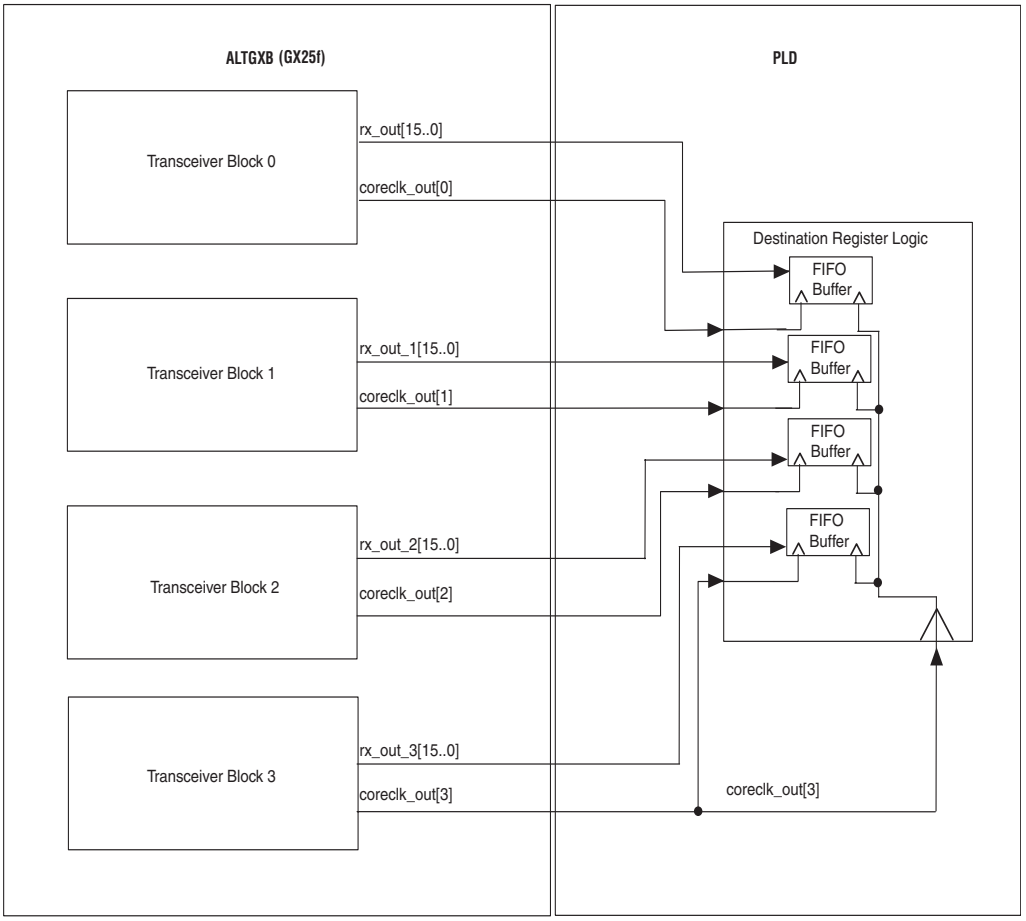
Figure 5–24. PLD to Transmit Interface Clocking Scheme in a Multi-Channel Application



At the FPGA-receive interface, there is no receive parallel interface clock option in the MegaWizard Plug-In Manager; the default is the transmitter PLL output clock, which is a transceiver internal clock.

Altera recommends implementing channel bonding across the transceiver blocks used in Stratix GX devices to ensure that there is no skew between the transceiver blocks (if each transceiver is operating, no channel bonding is required and the data can simply go to destination registers, as shown in Figure 5–25). Also, all traces in your design should match.

Figure 5–25. Clocking Scheme in Multi-Channel, Only CORECLK_OUT Is Enabled



XAUI mode applications are typically transceiver block-based. The previous recommendations are valid in a multi-transceiver block situation. In a multi-transceiver block situation, data striping across the channels is common. Skew introduced between transceiver blocks by passive and active elements of the link must be de-skewed in the PLD core (channel alignment) to ensure error-free data.

Another multi-transceiver block issue is the selection of the dedicated refclkb pin. Stratix GX channels are arranged in banks of four, which are called transceiver blocks. Each transceiver block has the ability to share a common reference clock through the Inter-Transceiver (IQ) lines. You can reduce the Stratix GX logic array clock usage by using the IQ lines. The IQ lines are used when a refclkb input port from one transceiver block or channel drives any other transceiver blocks or channels. The IQ line usage is determined automatically by the Quartus II software.

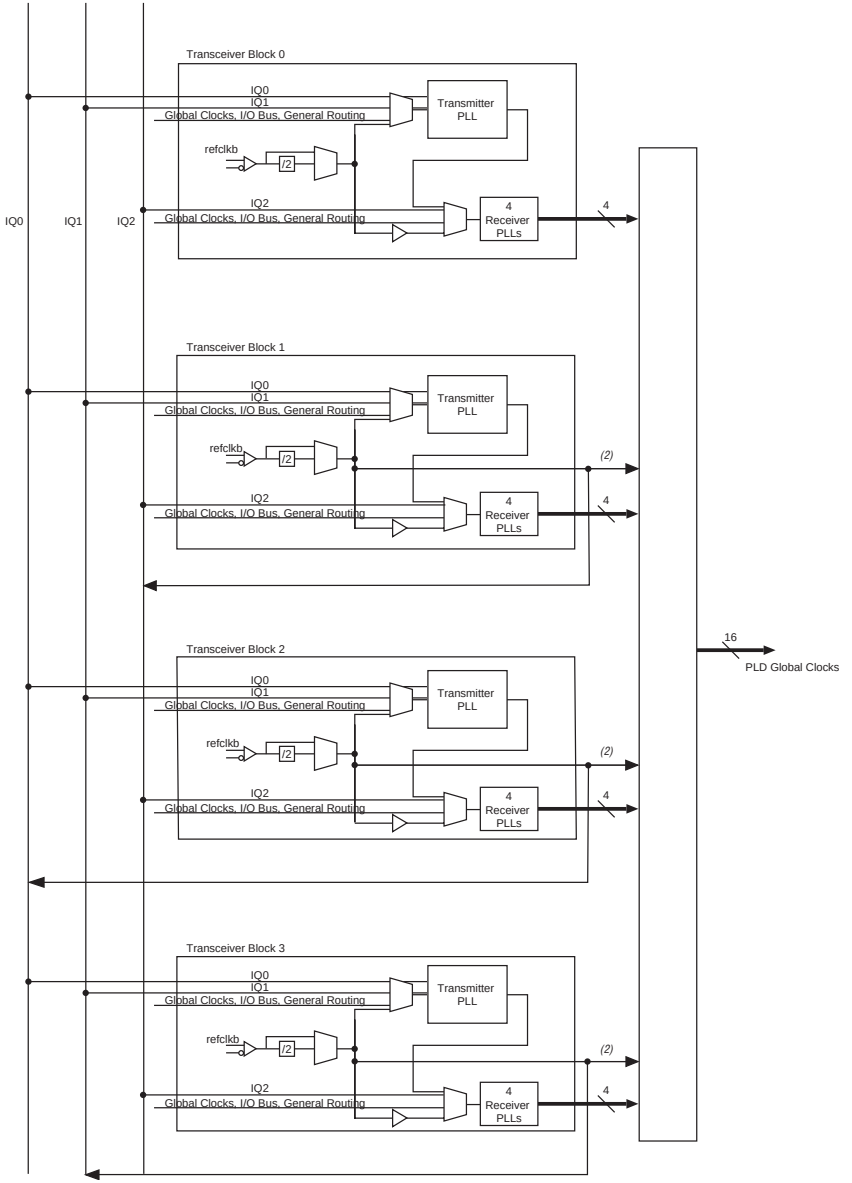
When determining the location of refclkb pins, consider what is fed by the pin you select. Table 5–6 shows the available IQ lines and which transceiver blocks are driven by refclkb. This information is based on the number of transceiver channels in the Stratix GX device.

Table 5–6. REFCLKB to Inter-Transceiver Line connections

Channel Density	REFCLKB in Transceiver Block Number	Channels in Transceiver Block	IQ Line Driven by REFCLKB
8 channels (EP1SGX10)	0	[3:0]	IQ2
	1	[7:4]	IQ0
16 channels (EP1SGX25)	0	[3:0]	N/A
	1	[7:4]	IQ2
	2	[11:8]	IQ0
	3	[15:12]	IQ1
20 channels (EP1SGX40)	0	[3:0]	N/A
	1	[7:4]	IQ2
	2	[11:8]	IQ0
	3	[15:12]	IQ1
	4	[19:16]	N/A

Figure 5–26 shows the transceiver routing with respect to Inter-Transceiver lines. This information is vital when placing refclkb pins. (When placing refclkb pins, refer to the *REFCLKB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* for information about analog reads and refclkb pin usage constraints.) For example, if a refclkb pin is required to feed a transmitter PLL using an IQ line, the refclkb pin cannot be in transceiver block 1, because IQ2 only feeds the receiver PLLs.

Figure 5–26. IQ Line Connections for EP1SGX25 Device *Note (1)*



Notes to Figure 5–26:

(1) IQ lines are inter-transceiver block lines.

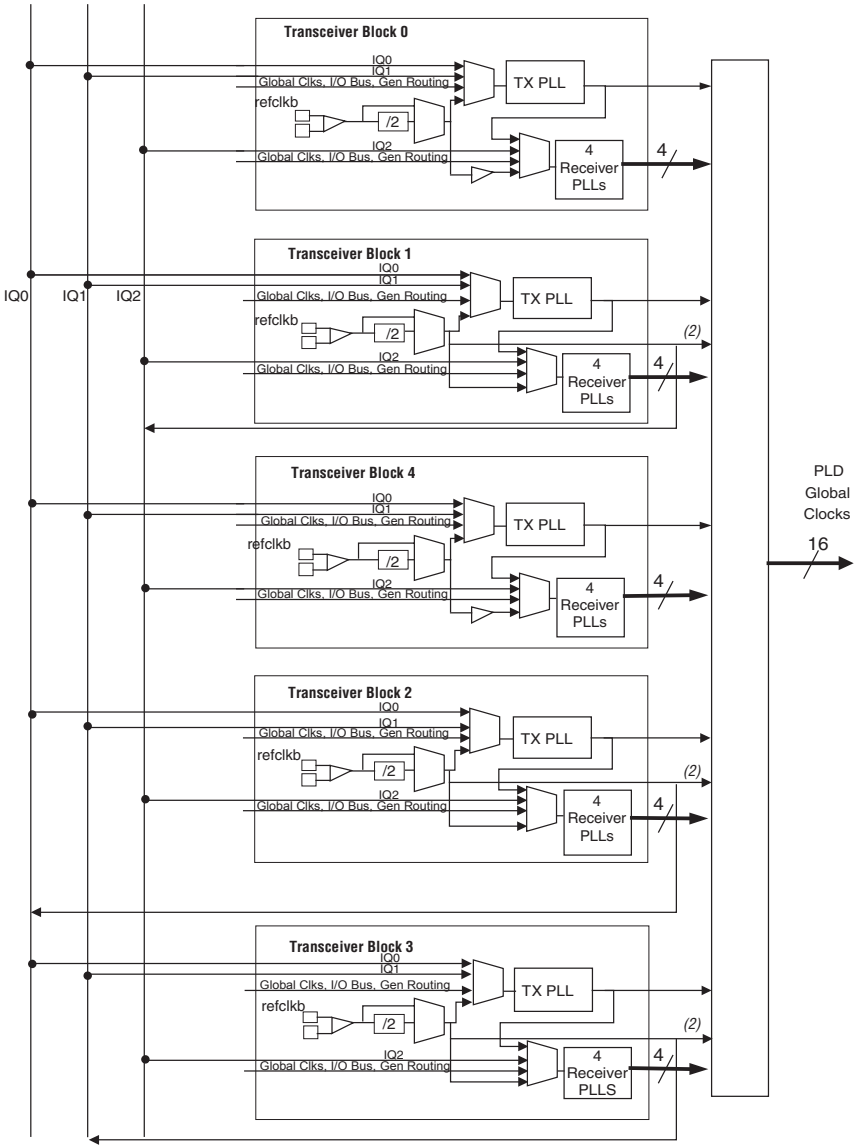
(2) If the /2 pre-divider is used, the path to drive the PLD logic array, local, or global clocks is not allowed.

(3) There are four receiver PLLs in each transceiver block.

Figure 5–27 shows the transceiver routing with respect to IQ lines for the EP1SGX40G device. This device has an extra transceiver block (transceiver block 4), which is in the middle of the other transceiver blocks, as shown. It is important to use this information when placing refclkb pins. (When placing refclkb pins, refer to the *REFCLKB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* for information about analog reads and refclkb pin usage constraints.)

For example, if a refclkb pin is required to feed a transmitter PLL using an IQ line, the refclkb pin cannot be in transceiver block 1, because IQ2 only feeds the receiver PLLs.

Figure 5–27. IQ Line Connections for EP1SGX40G *Note (1)*



Notes to Figure 5–27:

(1) IQ lines are inter-transceiver block lines.

(2) If the /2 pre-divider is used, the path to drive the PLD logic array, local, or global clocks is not allowed.

(3) There are four receiver PLLs in each transceiver block.

XAUI Mode MegaWizard Plug-In Manager

This section describes the altgxb megafunction MegaWizard® Plug-In Manager options in XAUI mode. Altera recommends that the Stratix GX transceiver block be instantiated and parameterized through the MegaWizard Plug-In Manager in the Quartus II software. The Quartus II MegaWizard Plug-In Manager offers a graphical user interface (GUI) that organizes the altgxb options in easy-to-use sections. The MegaWizard Plug-In Manager also sets the proper ports and parameters automatically, based on the selected options and parameters. Invalid settings are automatically flagged to help prevent illegal configurations. The MegaWizard Plug-In Manager grays out any options that do not apply to XAUI mode.

Although it is possible to instantiate the Stratix GX block directly by calling out the altgxb megafunction, Altera recommends using the MegaWizard Plug-In Manager to instantiate the altgxb megafunction to reduce the chance of invalid settings.

XAUI Mode MegaWizard Plug-In Manager Considerations

Each altgxb MegaWizard Plug-In Manager instantiation can use one or more transceiver blocks based on the number of channels you select. There are four channels per transceiver block. In XAUI mode, the number of channels are in multiples of four or transceiver block based.

Each MegaWizard Plug-In Manager instantiation must have similar functionality and data rates. To use transceiver blocks that differ in functionality and/or data rates, create a separate MegaWizard Plug-In Manager instantiation for each transceiver block.

As mentioned in the clocking section, the MegaWizard Plug-In Manager displays the configuration of the altgxb megafunction. This diagram changes dynamically based on the selected mode, options, and clocking schemes.

XAUI Mode altgxb MegaWizard Plug-In Manager Options

This section shows the MegaWizard Plug-In Manager pages where you select the options for a XAUI mode configuration.

Figure 5–28 shows page 3 of the altgxb MegaWizard Plug-In Manager in XAUI mode.

Figure 5–28. MegaWizard Plug-In Manager - altgxb (Page 3)

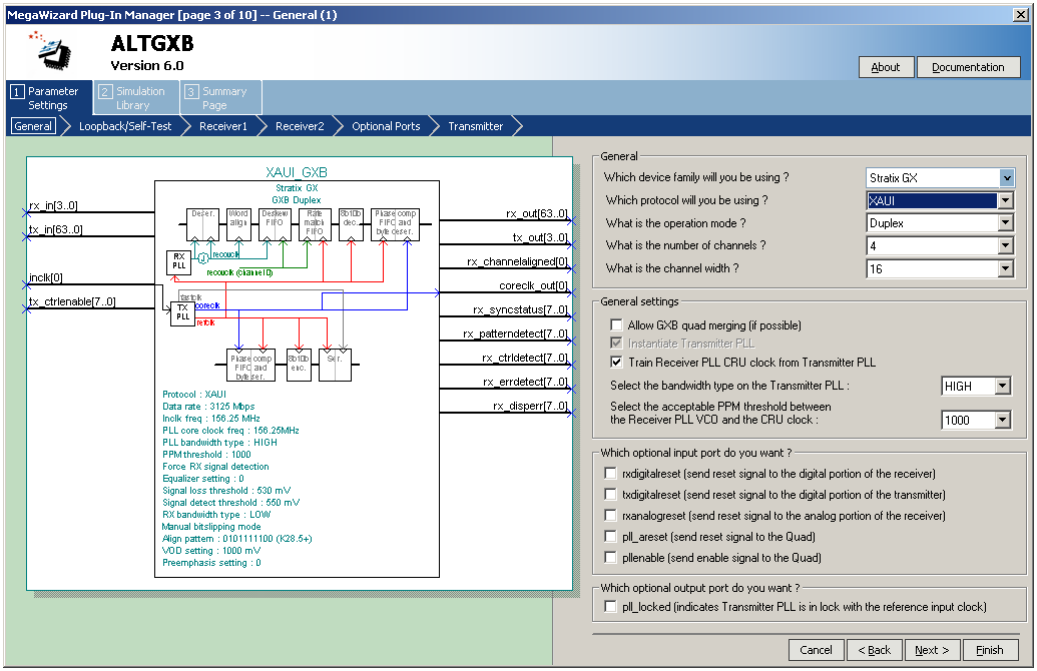


Table 5–7 describes the available options on page 3 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 5–7. MegaWizard Plug-In Manager Options (Page 3 for XAUI Mode) (Part 1 of 2)

altgxb Setting	Description
Which device family will you be using?	Stratix GX is the only option available.
Which protocol will you be using?	For the XAUI mode, you must select the XAUI protocol.
What is the operation mode?	XAUI protocol mode supports duplex only mode.
What is the number of channels?	This value can be from 4 to the maximum number of channels available on the device in increments of 4.
What is the channel width?	16 bits is double width.
Allow GXB quad merging (if possible)	For information about this option, refer to the section "Stratix GX Transceiver Merging" on page 5–47.
Instantiate Transmitter PLL	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.

Table 5–7. MegaWizard Plug-In Manager Options (Page 3 for XAUI Mode) (Part 2 of 2)

altgxb Setting	Description
Train Receiver PLL CRU clock from Transmitter PLL	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the bandwidth type on the Transmitter PLL	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the acceptable PPM threshold between the Receiver PLL VCO and the CRU clock	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
`rxdigitalreset` (send reset signal to the digital portion of the receiver)	The `rxdigitalreset` port resets the digital blocks in the receiver channel. Each active receiver channel has its own digital reset.
`txdigitalreset` (send reset signal to the digital portion of the transmitter)	The `txdigitalreset` port resets the digital blocks of the transmitter channel. Each active transmitter channel has its own digital reset.
`rxanalogreset` (send reset signal to the analog portion of the receiver)	The `rxanalogreset` port resets the receiver's analog circuits, including the receiver PLL. Each active receiver channel has its own analog reset.
`pll_areset` (send reset signal to the Quad)	The `pll_areset` port resets the entire transceiver block (all receiver and transmitter digital and analog circuits, including receiver and transmitter PLLs).
`pllenable` (send enable signal to the Quad)	The `pllenable` port enables the entire transceiver block; if deasserted, the entire transceiver block is held in the reset condition.
`pll_locked` (indicates Transmitter PLL is in lock with the reference input clock)	For more information, refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 5–29 shows page 4 of the `altgxb` MegaWizard Plug-In Manager in XAUI mode.

Figure 5–29. MegaWizard Plug-In Manager - altgxb (Page 4)

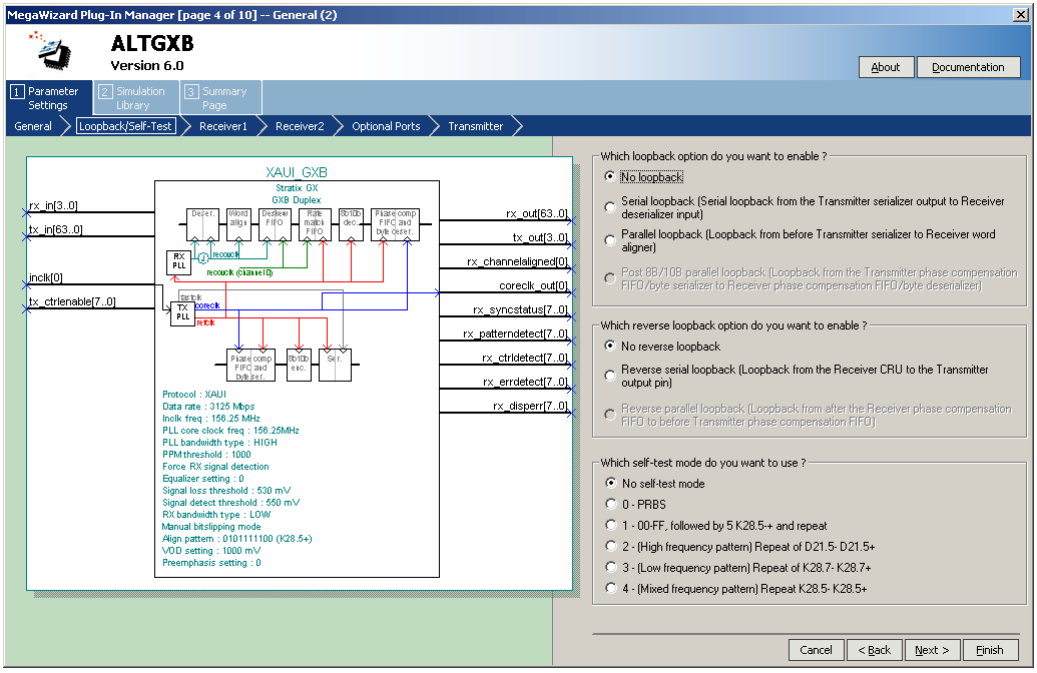


Table 5–8 describes the available options on page 4 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 5–8. MegaWizard Plug-In Manager Options (Page 4 for XAUI Mode)

altgxb Setting	Description
Which loopback option do you want to enable?	For more information, refer to the *Loopback Modes* chapter in volume 2 of the *Stratix GX Device Handbook*.
Which reverse loopback option do you want to enable?	For more information, refer to the *Loopback Modes* chapter in volume 2 of the *Stratix GX Device Handbook*.
Which self-test mode do you want to use?	For more information, refer to the *Stratix GX Built-In Self Test (BIST)* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 5–30 shows page 5 of the `altgxb` MegaWizard Plug-In Manager in XAUI mode.

Figure 5–30. MegaWizard Plug-In Manager - altgxb (Page 5)

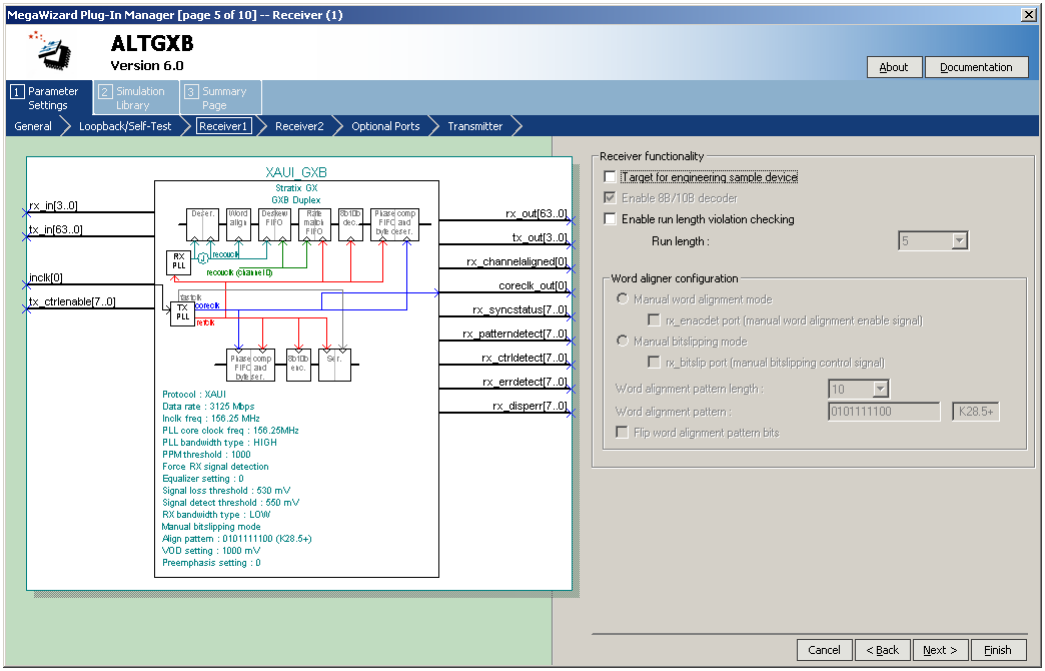


Table 5–9 describes the available options on page 5 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 5–9. MegaWizard Plug-In Manager Options (Page 5 for XAUI Mode) (Part 1 of 2)

altgxb Setting	Description
Target for engineering sample device	You must select this option if the design is targeted for an engineering sample (ES) device.
Enable 8B/10B decoder	In XAUI mode, this option is always enabled because 8B/10B data is always encoded.
Enable run-length violation checking	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Manual word alignment mode	This option is not available in XAUI mode.
`rx_enacdet` port (manual word alignment enable signal)	This option is not available in XAUI mode.
Manual bitslipping mode	This option is not available in XAUI mode.
`rx_bitslip` port (manual bitslipping control signal)	This option is not available in XAUI mode.

Table 5–9. MegaWizard Plug-In Manager Options (Page 5 for XAUI Mode) (Part 2 of 2)	
altgxb Setting	**Description**
Word alignment pattern length	This option is not available in XAUI mode because the word alignment pattern is always set to a 10-bit K28.5 pattern.
Word alignment pattern	The word aligner in XAUI mode is always set as a 10-bit K28.5 pattern. Both positive and negative disparities are checked.
Flip word alignment pattern bits	This option is not available in XAUI mode.

Figure 5–31 shows page 6 of the `altgxb` MegaWizard Plug-In Manager in XAUI mode.

Figure 5–31. MegaWizard Plug-In Manager - altgxb (Page 6)

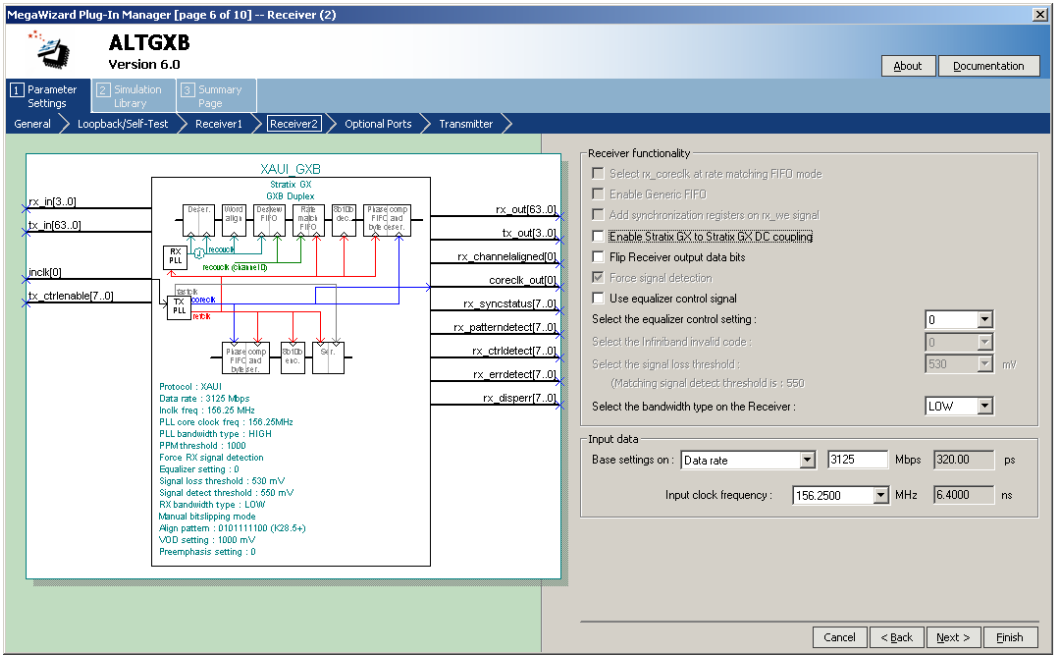


Table 5–10 describes the available options on page 6 of the MegaWizard Plug-In Manager for your altgxb custom megafunction variation.

Table 5–10. MegaWizard Plug-In Manager Options (Page 6 for XAUI Mode)

altgxb Setting	Description
Select rx_coreclk at rate matching FIFO mode	This option is not available in XAUI mode.
Enable Generic FIFO	This option is not available in XAUI mode, because a dedicated rate matching FIFO is included in the receiver data path by default.
Add synchronization registers on rx_we signal	This option is not available in XAUI mode.
Enable Stratix GX to Stratix GX DC coupling	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Force signal detection	The Force Signal Detect option is always on and cannot be turned off. The signal detect circuitry is always forced, so the rx_signaldetect is always set in XAUI mode.
Use equalizer control signal	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the equalizer control setting	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the Infiniband invalid code	This option is not available in XAUI mode.
Select the signal loss threshold	This option is not available in XAUI mode.
Select the bandwidth type on the Receiver	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Base settings on	By default, the data rate for XAUI is set to 3125 Mbps. Other data rates are possible, but they must adhere to the set multiplication factor of 2, 4, 5, 8, 10, 16, 20 of the input clock. Multiplication factors of 2, 4, 5 must use the refclkb pins. A multiplication factor of 2 also requires that the receiver PLL be trained by the transmitter PLL.

Figure 5–32 shows page 7 of the altgxb MegaWizard Plug-In Manager in XAUI mode.

Figure 5–32. MegaWizard Plug-In Manager - altgxb (Page 7)

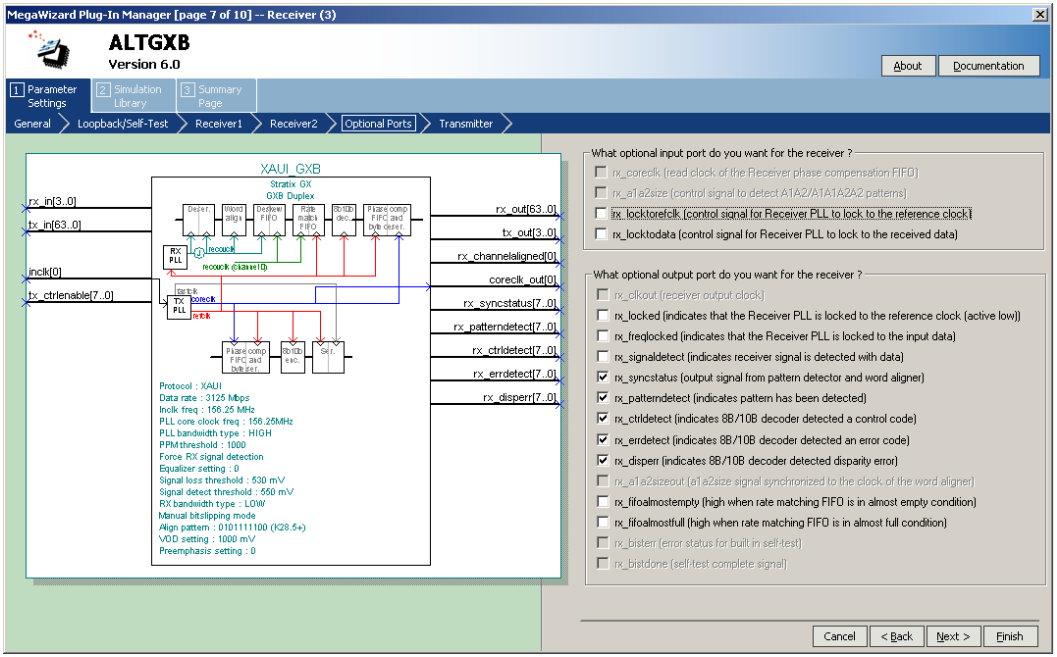


Table 5–11 describes the available options on page 7 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 5–11. MegaWizard Plug-In Manager Options (Page 7 for XAUI Mode) (Part 1 of 2)

altgxb Setting	Description
`rx_coreclk` (read clock of the Receiver phase compensation FIFO)	This option is not available in XAUI mode.
`rx_a1a2size` (control logic signal to detect A1A2/A1A1A2A2 patterns)	This option is not available in XAUI mode.
`rx_locktorefclk` (control signal for Receiver PLL to lock to the reference clock)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
`rx_locktodata` (control signal for Receiver PLL to lock to the received data)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
`rx_clkout` (receiver input clock)	This option is not available in XAUI mode.

Table 5–11. MegaWizard Plug-In Manager Options (Page 7 for XAUI Mode) (Part 2 of 2)

altgxb Setting	Description
rx_locked (indicates that the Receiver PLL is locked to the reference clock (active low))	Receiver PLL lock indicator. For rx_locked, Low = receiver PLL is locked to the reference clock.
rx_freqlocked (indicates that the Receiver PLL is locked to the input data)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_signaldetect (indicates receiver signal is detected with data)	rx_signaldetect is only available in XAUI or GIGE mode. The signal detect circuitry is always forced, so the rx_signaldetect signal is always set in XAUI and GIGE modes, supporting backward compatibility with existing designs. Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_syncstatus (output signal from pattern detector and word aligner)	Indicates when the word aligner has aligned to the byte boundary. The rx_syncstatus signal goes high for one rx_clkout period when the word aligner aligns to the new byte boundary. In 16-bit mode, each high and low byte has a separate rx_syncstatus signal.
rx_patterndetect (indicates pattern has been detected)	Similar to rx_syncstatus, except that rx_patterndetect asserts only when the word alignment pattern appears in the data stream within the synchronized byte boundary.
rx_ctrldetect (indicates 8B/10B decoder detected a control code)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_errdetect (indicates 8B/10B decoder detected an error code	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_disperr (indicates 8B/10B decoder detected disparity error)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_a1a2sizeout (a1a2size signal synchronized to the clock of the word aligner)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_fifoalmostempty (high when rate matching FIFO is in almost empty condition)	You can include this rate matching FIFO status signal by enabling this option. When driven HIGH, this signal indicates an almost empty condition for the rate matching FIFO.
rx_fifoalmostfull (high when rate matching FIFO is in almost full condition)	You can include this rate matching FIFO status signal by enabling this option. When driven HIGH, this signal indicates an almost full condition for the rate matching FIFO.
rx_bisterr (error status for built-in self-test)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_bistdone (self-test complete signal)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 5–33 shows page 8 of the altgxb MegaWizard Plug-In Manager in XAUI mode.

Figure 5–33. MegaWizard Plug-In Manager - altgxb (Page 8)

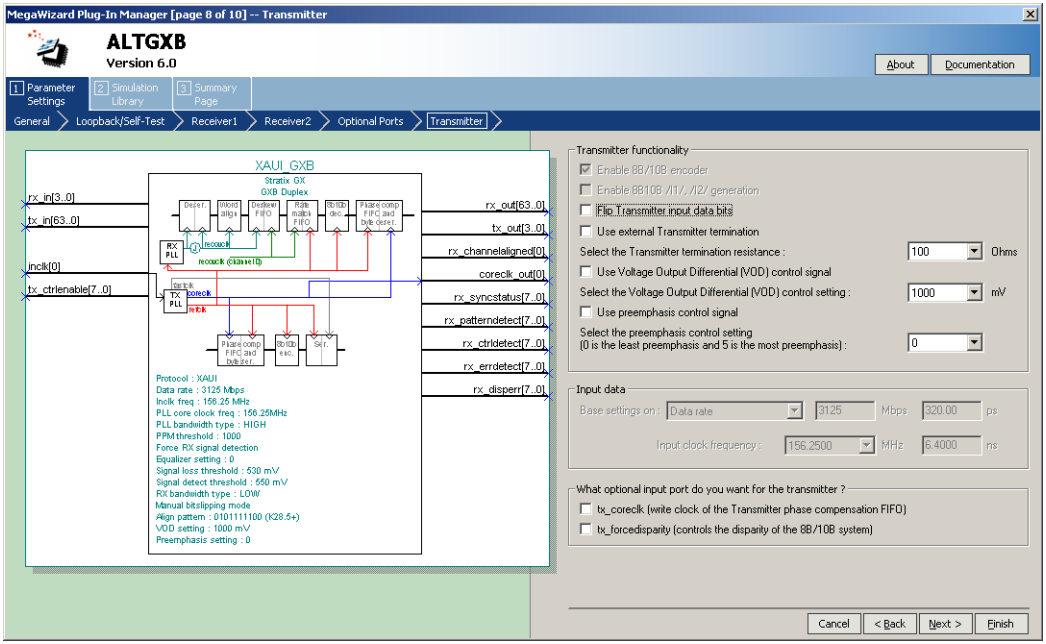


Table 5–12 describes the available options on page 8 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 5–12. MegaWizard Plug-In Manager Options (Page 8 for XAUI Mode) (Part 1 of 2)

altgxb Setting	Description
Enable 8B/10B encoder	In XAUI mode, this option is always enabled because data is always 8B/10B encoded.
Enable 8B/10B /I1/, /I2/ generation	This option is not available in XAUI mode.
Use external Transmitter termination	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Use Voltage Output Differential (VOD) control signal	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the Voltage Output Differential (VOD) control setting	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Use preemphasis control signal	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.

Table 5–12. MegaWizard Plug-In Manager Options (Page 8 for XAUI Mode) (Part 2 of 2)	
altgxb Setting	**Description**
Select the preemphasis control setting (0 is the least preemphasis and 5 is the most preemphasis)	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
`tx_coreclk` (write clock of the Transmitter phase compensation FIFO buffer	You can optionally choose the write clock of the transmitter phase compensation FIFO buffer. This clock should be frequency locked with the internal reference clock because the phase compensation FIFO buffer cannot tolerate frequency variations and contains no error flags.
`tx_forcedisparity` (controls the disparity of the 8B/10B system)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 5–34 shows page 9, the Simulation Libraries page, of the MegaWizard Plug-In Manager for the XAUI protocol set up.

Figure 5–34. MegaWizard Plug-In Manager - altgxb (Page 9)

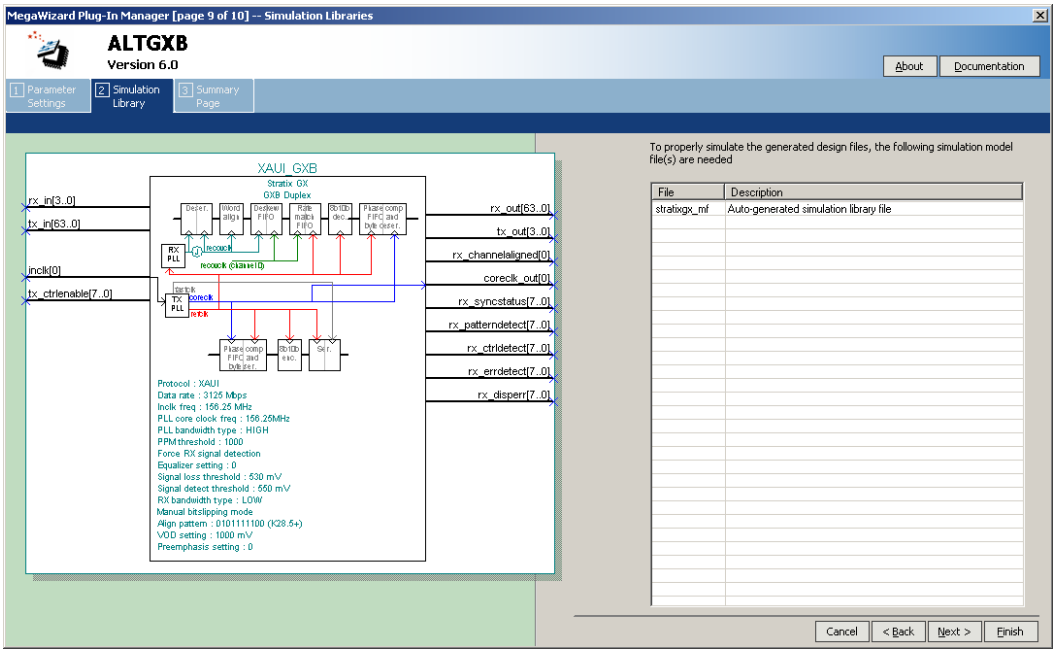
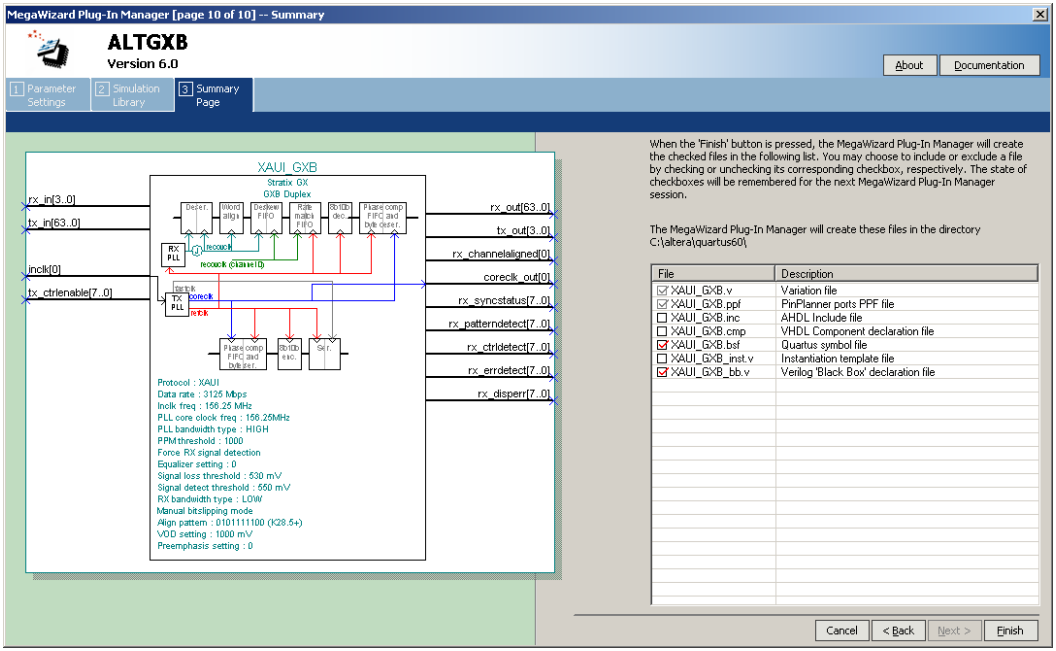


Figure 5–35 shows page 10 of the MegaWizard Plug-In Manager for the XAUI protocol set up. You can select optional files on this page. After you make your selections, click **Finish** to generate the files.

Figure 5–35. MegaWizard Plug-In Manager - altgxb (Page 10)



Stratix GX Transceiver Merging

A transceiver block contains four transceivers. In a design, an `altgxb` instantiation is placed in one or more transceiver blocks and potentially leaves unused transceivers in a block. For example, a six transceiver instantiation completely fills one transceiver block and half fills a second, taking up two full transceiver blocks. If another instantiation is in the design, it is placed the same way. For example, an instantiation of two transmitters takes up a third transceiver block. Merging two of the partially filled transceiver blocks into one transceiver block reduces the resources used and allows a design to fit into a device with fewer transceiver blocks.

The `altgxb` MegaWizard Plug-In Manager in the Quartus II software has a feature that allows merging of similar quads (transceiver blocks). With a few exceptions, transceiver blocks can be merged if the options

chosen in the wizard are the same. The Quartus II software merges the transceiver blocks automatically if you turn on the merging option for each instantiation. Table 5–13 shows the considerations for merging.

Table 5–13. Stratix GX Merging Considerations

Merging Rules	Required to Match Between Transceiver Blocks	Not Required to Match Between Transceiver Blocks

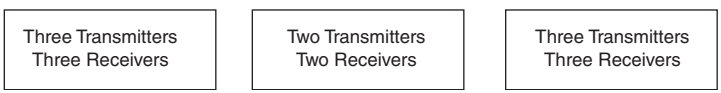
MegaWizard Plug-In Manager options	`USE_8B_10_MODE`
	`USE_DOUBLE_DATA_MODE`
	`CHANNEL_WIDTH`
	`SYNC_MODE`
	`DATA_RATE`
	`TRANSMIT_PROTOCOL`
		VOD
		Pre-emphasis
		Equalization
		Number of transmitters
		Number of receivers

Input control signals must be shared across transceiver blocks and must be from the same source	Clock
	`CRU_CLOCK`
	`PLL_RESET`
	`PLL_ENABLE`	

The merging partial transceiver blocks must reduce the number of transceiver block when merged	The total number of receivers and transmitters must be tx ≤4 x *n* and rx ≤4 x *n*, where *n* is the reduced number of transceiver blocks remaining after merging.	

The Quartus II software does not merge two transceiver blocks if they won't completely fit into one. It can, however, merge three transceiver blocks into two. Figure 5–36 shows a configuration with three transceiver blocks that can potentially be merged.

Figure 5–36. Three Transceiver Configuration



In Figure 5–36, two transceiver blocks cannot merge because there would be extra channels remaining. But because there are three transceiver blocks, the Quartus II software can merge them into two with no remaining channels.

If you turn on the merging option, the Quartus II software automatically merges transceiver blocks if possible or provides a warning during compilation if merging is not possible. The merging feature can reduce the transceiver block resources used in your design.

6. GIGE Mode

SGX52006-1.2

Introduction

The Gigabit Ethernet (GIGE) mode in Stratix® GX devices supports a subset of the IEEE GIGE standard. Stratix GX devices have Physical Coding Sub-layer (PCS) functions and Physical Medium Attachment (PMA) functions as Hard Intellectual Property (IP).

Stratix GX devices provide the following GIGE features:

- Serial data rate of 1.25 Gigabits per second
- Input clock reference range of 62.5 to 625 MHz (these values are the minimum and maximum for an input reference clock with a data rate of 1.25 Gbps and an 8-bit data width)
- Parallel interface width of 8 bits
- 8B/10B encoding decoder
- Word aligner supports 10-bit code groups
- Rate compensation or elastic buffer
- Gigabit Media Independent Interface (GMII) to PCS code conversion on transmit

The GMII is an intermediate or parallel interface that connects the PCS sub-layer with the media access control (MAC) in a system that supports GIGE mode. The GIGE physical layer is divided into three sub-layers: the PCS, the PMA, and the physical medium dependent (PMD) layers. If you implement a GMII-compliant interface, that interface offers data rates up to 1,000 Mbps at either half- or full-duplex modes.

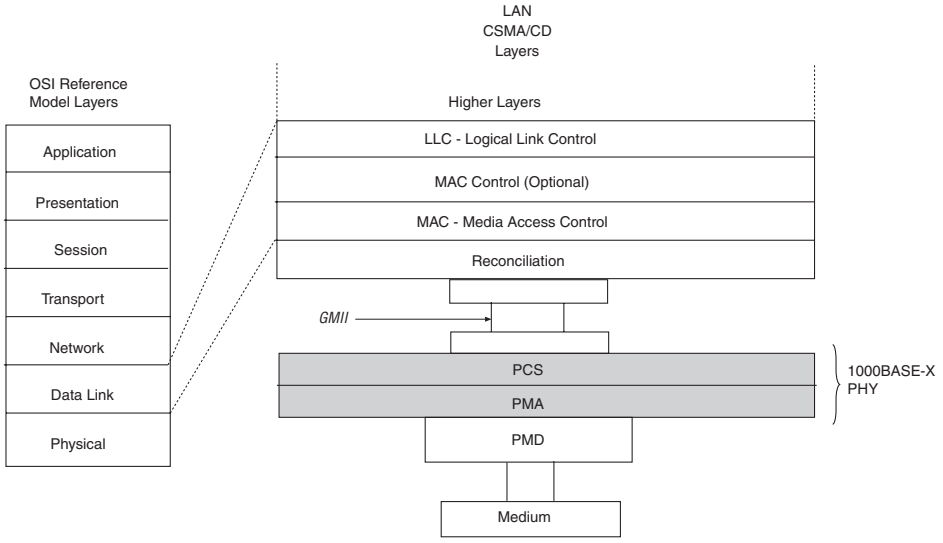
The PCS provides synchronization, encoding, decoding, and rate matching services to the MAC. The PCS also provides autonegotiation to the network to negotiate speeds, carrier-detect signals, and collision-detect signals.

The PMA sublayer provides the PCS with a media-independent interface that a variety of serial physical media can connect to. This sublayer handles the serialization and deserialization of the data.

The PMD sublayer defines the physical attachments, such as connectors for different media types.

Figure 6–1 shows the positioning of these layers.

Figure 6–1. GMII Position Relative to OSI Reference Model



Stratix GX devices are used for the PCS and the PMA layers of the GIGE physical layer. Stratix GX devices in GIGE mode use built-in hard macros for the 8B/10B encoder/decoder, rate matcher, synchronizer, or the byte serializer/deserializer. Figure 6–2 shows these components. The rate matcher and the word aligner contain a dedicated state machine governing their functions, which is active only in GIGE mode. GIGE mode enables transceivers to support GMII-to-PCS code group conversion and idle generation. Table 6–1 shows the GIGE code groups for the reference of idle ordered sets and configuration ordered sets, as explained in the "Idle Generation" section. For full details on the GIGE standard and code-group functionality, refer to clause 36 in the Gigabit Ethernet standard (IEEE 802.3).

The remaining functions of the PCS—auto negotiation, collision detect, and carrier detect—must be implemented in user logic or external circuits if these functions are needed.

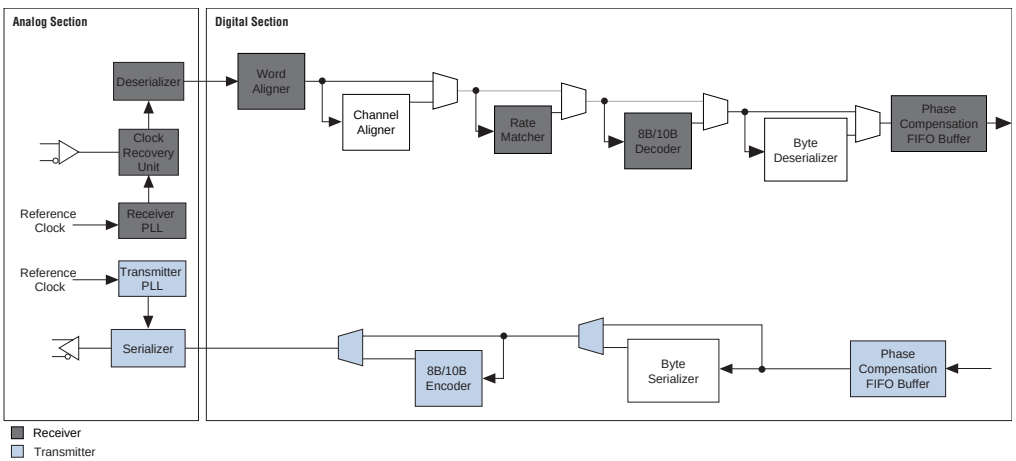
Table 6–1. GIGE Code Groups (Part 1 of 2) *Note (1)*

Code	Ordered Set	Number of Code Groups	Encoding
/C/	Configuration		Alternating /C1/ and /C2/ code groups
/C1/	Configuration 1	4	/K28.5/D21.5/Config_Reg *(1)*

Table 6–1. GIGE Code Groups (Part 2 of 2) Note (1)			
Code	**Ordered Set**	**Number of Code Groups**	**Encoding**
/C2/	Configuration 2	4	/K28.5/D2.2/Config_Reg (1)
/I/	IDLE		/I1/ is correcting; /I2/ is preserving
/I1/	IDLE 1	2	/K28.5/D5.6/
/I2/	IDLE 2	2	/K28.5/D16.2/
	Encapsulation		
/R/	Carrier_Extend	1	/K23.7/
/S/	Start_of_Packet	1	/K27.7/
/T/	End_of_Packet	1	/K29.7/
/V/	Error_Propagation	1	/K30.7/

Note to Table 6–1:
(1) Two data code groups represent the Config_Reg value.

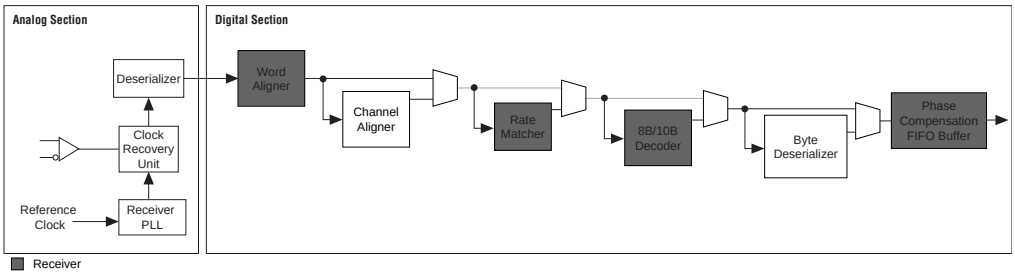
Figure 6–2. Block Diagram of a Duplex Channel Configured in GIGE Mode



GIGE Mode Receiver Architecture

Figure 6–3 shows the digital components of the Stratix GX receiver that are active in GIGE mode.

Figure 6–3. Block Diagram of the Stratix GX Receiver Digital Components in GIGE Mode



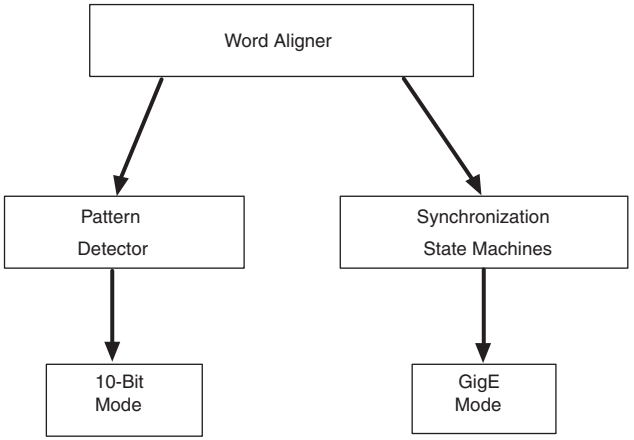
The GIGE mode receiver architecture includes:

- Word aligner
- Rate matcher
- 8B/10B decoder
- Receiver phase compensation FIFO buffer

Word Aligner

The word aligner is composed of a pattern detector and synchronization state machines. The word aligner cannot be bypassed, but if the application is not using the `rx_enacdet` signal, the word aligner does not alter the data. Figure 6–4 shows the various components of the word aligner. The "Pattern Detector Module" and "Synchronization State Machines" sections describe the functionality of the main components.

Figure 6–4. Components in Stratix GX Word Aligner



For embedded clocking schemes, the clock is recovered from the incoming data stream based on the data transition density. Therefore, you do not need to factor in receiver skew margins between the clock and data. However, with this clocking methodology, the word boundary of the re-timed data might be altered. Stratix GX devices offer an embedded word alignment circuit that uses synchronization state machines in conjunction with the pattern detector to align the word boundary of the re-timed data to a specified comma. This embedded circuit can be configured to synchronize to the GIGE protocols.

GIGE mode requires synchronization to align the byte boundary of the receiver after incoming serial data is de-serialized. This step is necessary because the Stratix GX block uses a non-source-synchronous serial stream. To correctly align the byte boundary at the receiver, the Stratix GX device sends a unique synchronization pattern to the receiver that does not occur between any `Dx.y` or `Kx.y` code combinations, namely, a `/K28.5/` 10-bit comma.

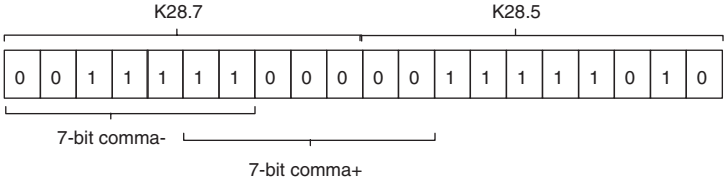
Pattern Detector Module

The pattern detector matches a predefined comma to the current byte-boundary. If the comma is present, the optional `rx_patterndetect` signal asserts for one clock cycle to signify that the comma exists in the current word boundary. The pattern detector module only indicates that the signal exists and does not modify the word boundary. A 10-bit pattern can be programmed for the pattern detector to recognize.

In GIGE mode, the MegaWizard® Plug-In Manager defaults to the 10-bit /K28.5/ code as the comma character. The Quartus® II software automatically sets the options related to the word aligner, and you cannot change these options in GIGE mode. This module matches the 10-bit comma with the data and its complement in the current word boundary. Both positive and negative disparities are checked in this mode. For example, if you specify a /K28.5/ (b'0011111010) pattern as the comma, the rx_patterndetect signal asserts if either the b'0011111010 or b'1100000101 pattern is present in the incoming data.

To use transceiver parameters to set the functional mode, you must preconfigure the receiver with a K28.5 (10'b0101111100 or 10'b1010000011) word align pattern (ALIGN_PATTERN = 0101111100 or ALIGN_PATTERN = 1010000011). Set the ALIGN_PATTERN_LENGTH to 10, even though a 7-bit comma string (7'b0011111 as a comma- or 7'b1100000 as a comma+) is allowed, as stated in the IEEE 802.3 specification. This 7-bit comma is part of the /K28.1/, /K28.5/, and /K28.7/ code-groups. Use a 10-bit /K28.5/ code group to prevent a 7-bit comma from being detected across boundaries when a /K28.7/ code is followed by a /K28.x/, /D3.x/, /D11.x/, /D12.x/, /D19.x/, /D20.x/, or /D28.x/ code group, where *x* is a value from 0 to 7. Figure 6–5 shows this situation.

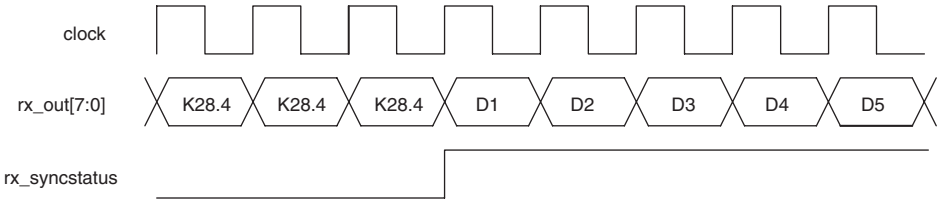
Figure 6–5. A Cross-boundary 7-bit Comma When a /K28.7/ Code is Followed by a /K28.5/ Code



The receiver sends a K28.4 (8'h9c + rx_ctrldetect) code from the rx_out[] port and deasserts the rx_syncstatus (1'b0) signal when the receiver is not synchronized. When synchronized, the receiver asserts the rx_syncstatus (1'b1) signal. This signal is aligned with the first valid data received from the rx_out[] port.

Figure 6–6 shows the waveforms related to receiver synchronization. The rx_syncstatus signal goes high when synchronization is complete, indicating that the data is valid. In the example, D1 is the first valid data.

Figure 6–6. Example of Completed Synchronization



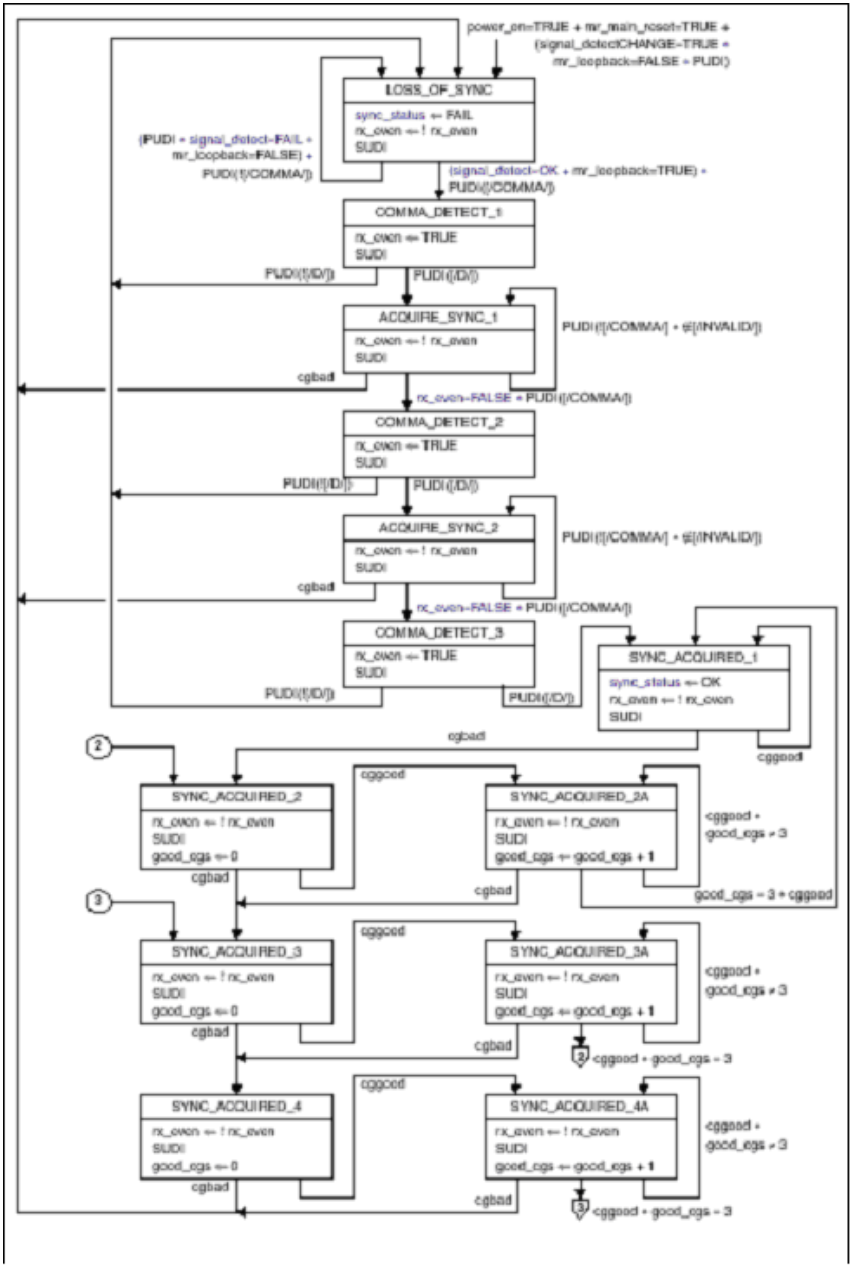
The receiver remains synchronized until it detects a series of bad code groups or is reset. The IEEE 802.3 standard defines the bad code group as four invalid code groups separated by fewer than three valid code groups. If the receiver detects the bad code group or is reset, the `rx_syncstatus` signal goes low, and a `/K28.4/` code appears on the `rx_out[]` port. GIGE mode uses an embedded clocking scheme that retimes all data that can potentially alter the code-group boundary. The boundaries of the code-groups are re-aligned through a synchronization process specified in the IEEE 802.3 standard.

Synchronization State Machines

Synchronization occurs when the receiver sees three consecutive ordered sets. An ordered set defined for synchronization is a `/K28.5/` comma followed by any odd number of valid data code groups (`/Dx.y/`). Although you can have a number of sync patterns based on the synchronization rule, three sets of {`/K28.5/ /Dx.y/`} code groups are the fastest way to achieve synchronization.

GIGE mode requires a special synchronization sequence that follows the IEEE 802.3 GMII PCS synchronization specification, as shown in Figure 6–7.

Figure 6–7. Synchronization Diagram State Machine



Rate Matcher

The GIGE mode operates in multi-crystal environments, which can tolerate a frequency variation of ±100 ppm between crystals. Stratix GX devices have embedded circuitry to perform clock rate compensation by inserting or removing the /I2/ code group from the interpacket gap (IPG) or idle stream. This process is called "rate matching" or "clock rate compensation."

The IEEE 802.3 standard, clause 36, specifies two idle order sets (/I1/ and /I2/) for the transmitter. The /I1/ ordered set consists of a negative disparity /K28.5/ (10'h283) followed by a /D5.6/ code group. (A /D5.6/ has the same value, 10'h1A5, for the positive and negative disparity versions and has a balanced 10-bit code.) The /I1/ ordered set should be transmitted only once if the running disparity before the idle is positive.

The /I2/ ordered set consists of a positive disparity /K28.5/ (10'h17C) followed by a negative disparity /D16.2/ (10'h289) code group. The /I2/ ordered set can start the idle sequence if the disparity before the idle sequence is negative. Otherwise, /I2/ follows an /I1/ ordered set and is continually transmitted, maintaining a negative running disparity until the end of the IPG.

Figure 6–8 shows a case in which the idle stream starts with an /I1/ followed by /I2/ ordered sets. The running disparity before the idle state is positive, as indicated by the positive disparity /D30.1/.

Figure 6–8. Idle Generation With /I1/ Ordered Set

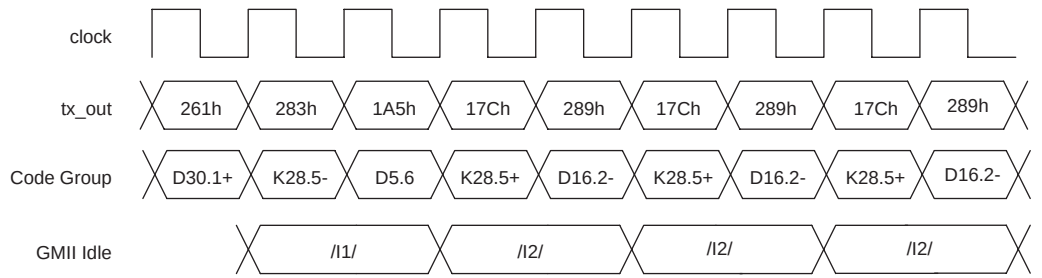
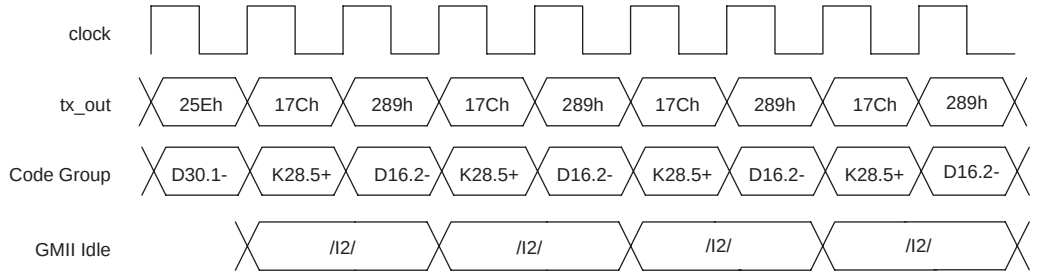


Figure 6–9 shows cases in which only /I2/ ordered sets are generated. The running disparity is negative before the start of the idle generation, as indicated by the negative disparity /D30.1/. The /D30.1/ code group in Figure 6–8 and Figure 6–9 is intended only for illustrating disparity and is not intended to signify an end of frame (EOF), nor is it required prior to idle generation.

Figure 6–9. Idle Generation Without /I1/ Ordered Set



Stratix GX devices have a built-in rate matcher that is 12 words deep, which is a FIFO buffer with control logic. Stratix GX devices implement rate matching in GIGE mode by adding or removing /I2/ ordered sets. The /I1/ ordered set is not added or removed.

If the rate matching FIFO buffer encounters an almost full condition, an /I2/ ordered set is deleted, as shown in Figure 6–10. If the rate matching FIFO buffer encounters an almost empty condition, an /I2/ ordered set will be added, as shown in Figure 6–11. The position of the /I2/ ordered set that is added to or deleted from the idle stream varies, depending on when the rate matcher encounters the almost full or almost empty condition.

Figure 6–10. Detection of an /I2/ Ordered Set During an Almost Full Condition

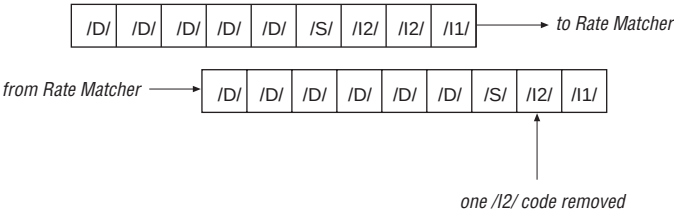
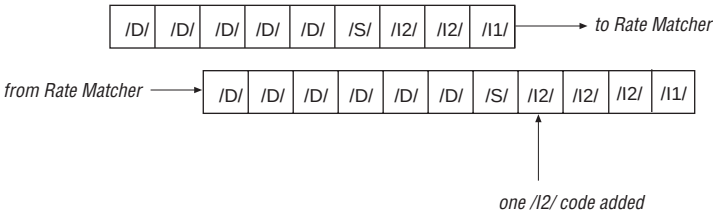


Figure 6–11. Addition of an /I2/ Ordered Set During an Almost Empty Condition



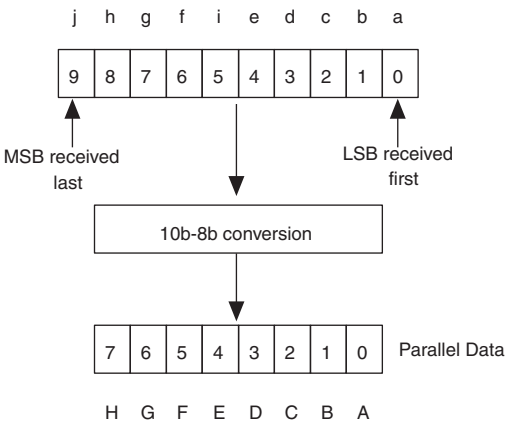
8B/10B Decoder

The 8B/10B decoder is part of the Stratix GX transceiver block. The purpose of the 8B/10B Decoder is to restore the 8-bit data plus 1-bit control identifier from the 10-bit code.

10-Bit Decoding

The 8B/10B decoder translates the 10-bit encoded data into the 8-bit equivalent data or control code. The byte deserializer receives the least significant bit (LSB) of the 10-bit encoded code first, and the most significant bit (MSB) last. The data received must be from the supported `Dx.y` or `Kx.y` list. All 8B/10B control signals (disparity error, control detect, and code error) are pipelined with the data in the Stratix GX receiver block and are edge-aligned with the data. Figure 6–12 is a diagram of the 10-bit to 8-bit conversion.

Figure 6–12. 10-Bit to 8-Bit Conversion



Reset

The rxdigitalreset signal governs the reset condition of the 8B/10B decoder. In reset, the disparity registers are cleared. Upon exiting reset, the 8B/10B decoder starts with either a positive or negative disparity. The decoder calculates the initial running disparity based on the first valid code that is received.

The receiver block must be word-aligned after reset before the 8B/10B decoder can decode valid data or control codes.

Code Error Detect

The rx_errdetect signal indicates when the code received contains an error. This port is optional and, if not in use, there is no way to detect whether a code received is valid. The rx_errdetect goes high if a code received is an invalid code, or if it has a disparity error. If a code is received that is not part of the valid Dx.y or Kx.y list, the rx_errdetect signal goes high. This signal is aligned to the invalid code word received at the PLD logic array.

Disparity Error Detector

The 8B/10B decoder detects disparity errors based on which 10-bit code it received. The disparity error is indicated at the optional rx_disperr port. The current running disparity is based on the disparity calculation of the last code received. The disparity calculation is described in the 8B/10B code section in the *Data & Control Codes* chapter of the *Stratix GX Device Handbook, Volume 2*.

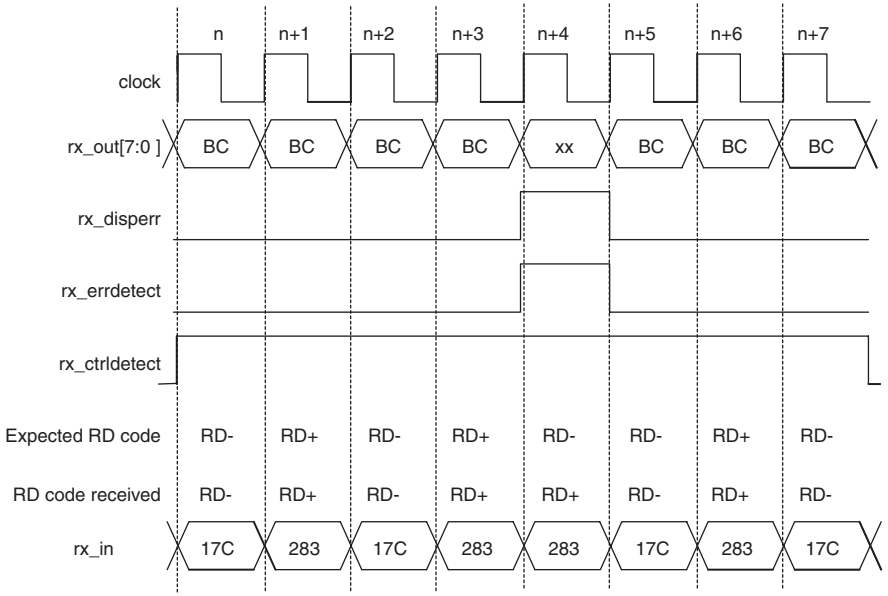
If negative disparity is calculated for the last 10-bit code, a neutral or positive disparity 10-bit code is expected. If the decoder does not receive a neutral or positive disparity 10-bit code, the rx_disperr signal goes high.

If a positive disparity is calculated, a neutral or negative disparity 10-bit code is expected. In this situation, the rx_disperr signal goes high if the code received is not as expected. When the rx_disperr signal is high, the rx_errdetect signal also goes high.

Figure 6–13 shows a case where the disparity is violated. A K28.5 code has an 8-bit value (8'hbc) and a 10-bit value (jhgfiedcba). The 10-bit value is 10'b0011111010 (10'h17c) for RD- or 10'b1100000101 (10'h283) for RD+. If the running disparity at time *n – 1* is negative the expected code at time *n* must be from the RD- column. Because a K28.5 does not have a balanced 10-bit code (having an equal number of 1's and 0's), the expected RD code must toggle back and forth between RD- and RD+. At time *n + 3*, the 8B/10B decoder received an RD+ K28.5 code

(10'h283), which would make the current running disparity negative. At time *n* + *4*, because the current disparity is negative, a K28.5 from the RD- column is expected, but a K28.5 code from the RD+ is received instead. This disparity prompts the rx_disperr signal to go high during time *n* + *4* to indicate that this particular K28.5 code contained a disparity error. The current running disparity at the end of time *n* + 4 is negative because a K28.5 code from the RD+ column was received. Based on the current running disparity at the end of time *n* + *5*, a positive disparity K28.5 code (from the RD-) column is expected at time *n* + *5*.

Figure 6–13. Disparity Error

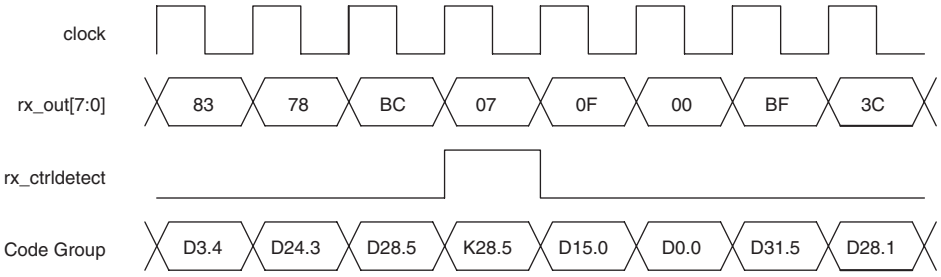


Control Detect

The 8B/10B decoder differentiates between data and control codes using the rx_ctrldetect port. Although this port is optional, there is no way of differentiating a Dx.y code group from a Kx.y code group if the port is unused.

Figure 6–14 shows an example waveform demonstrating the receipt of a K28.5 code (BC + ctrl). The rx_ctrldetect=1'b1 port is aligned with 8'hbc, indicating that it is a control code. The rest of the code received is data.

Figure 6–14. Control Code Detection



Receiver Phase Compensation FIFO Buffer

The receiver phase compensation FIFO buffer is located at the FPGA logic array interface in the receiver block and is four words deep. This FIFO buffer compensates for the phase difference between the clock in the FPGA and the operating clocks in the transceiver block.

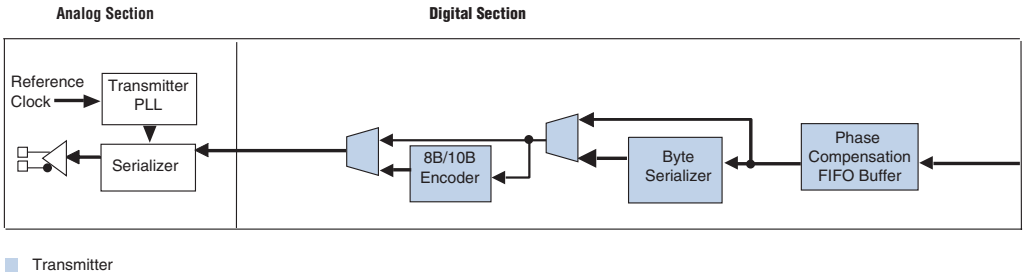
In GIGE mode, the write port is clocked by the `refclk` from the transmitter phase-locked loop (PLL). The read clock is clocked by `CORECLK` (output from the transmitter PLL). The receiver phase compensation FIFO buffer can only account for phase differences and must be derived from the recovered clock of its associated channel.

The receiver phase compensation FIFO buffer is always used, and you cannot bypass it.

GIGE Mode Transmitter Architecture

Figure 6–15 shows the digital components of the Stratix GX transmitter that are active in GIGE mode.

Figure 6–15. Block Diagram of Transmitter Components Configured in GIGE Mode



The transmitter architecture includes:

- Transmitter phase compensation FIFO buffer
- GIGE transmitter synchronization
- Idle generation
- 8B/10B encoder

Transmitter Phase Compensation FIFO Buffer

The transmitter phase compensation FIFO buffer is located at the FPGA logic array interface in the transmitter block and is four words deep. The phase compensation FIFO buffer compensates for the phase difference between the clock in the FPGA and the operating clocks in the transceiver block.

The transmitter PLL output clock (`refclk`) clocks the read port of the phase compensation FIFO buffer. The `TX_CORECLK` port clocks the write clock. You can select the `TX_CORECLK` port as an optional transmitter input port to use as a write-side clock of the FIFO buffer. Make sure that there is no frequency difference between the `TX_CORECLK` port and the transmitter PLL clock. The transmitter phase compensation FIFO only accounts for phase differences.

If you do not select the `TX_CORECLK` port as an optional input transmitter port, the `CORECLK_OUT` port feeds the `TX_CORECLK` port. This connection occurs using the logic array routing. As a result, the software defaults to using an FPGA global clock, regional clock, or fast regional clock resource.

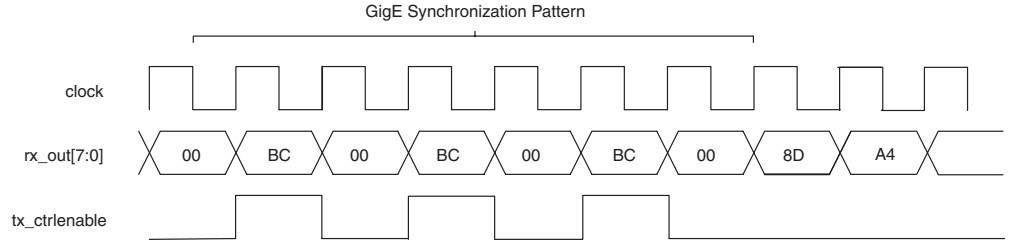
The transmitter phase compensation FIFO buffer is always used, and you cannot bypass it. The input to the transmitter phase compensation FIFO buffer is the data from the PLD logic array. The tx_ctrlenable and tx_forcedisparity signals are also passed through the FIFO buffer to ensure that they are synchronized with the data when they feed to the subsequent module.

GIGE Transmitter Synchronization

The transmitter must send out the GIGE synchronization sequence to synchronize the target receiver. Stratix GX devices do not have a built-in macro that performs this function on power-up or txdigitalreset. This function must be implemented in user logic to send out a /K28.5/, /Dx.y/, /K28.5/, /Dx.y/, /K28.5/, /Dx.y/ sequence.

Figure 6–16 shows an example of the GIGE synchronization pattern. Although the example shows one D0.0 (8'h00) as the /Dx.y/ code, any /Dx.y/ and any odd number of /Dx.y/ can be used.

Figure 6–16. Example of a GIGE Synchronization Transmit Pattern



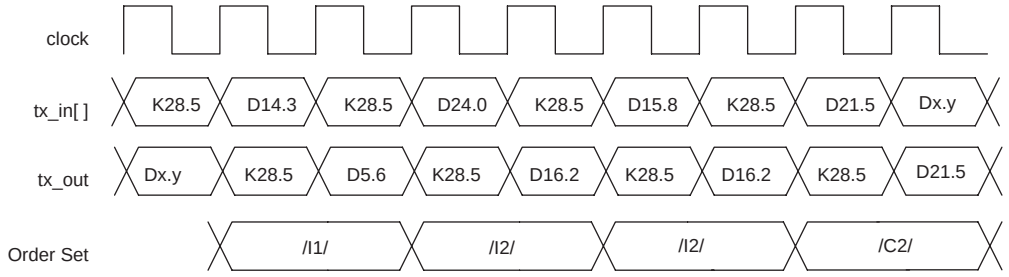
Idle Generation

In GIGE mode, the transmitter replaces any /Dx.y/ code group following a /K28.5/ comma with either a /D5.6/ (8'hc5) or a /D16.2/ (8'h50), depending on the current running disparity, except when the data following the /K28.5/ is /D21.5/ (8'hb5) or /D2.2/ (8'h42). This replacement is to ensure the generation of /I1/ (/K28.5/, /D5.6/) and /I2/ (/K28.5/, /D16.2/) ordered sets and to let the configuration ordered sets /C1/ (/K28.5/, /D21.5/) and /C2/ (/K28.5/, /D2.2/) be received. If the running disparity before the idle ordered set is positive, an /I1/ is chosen. If the running disparity is negative, an /I2/ is chosen. The disparity at the end of an /I1/ is the opposite of the disparity at the beginning of the /I1/. However, the disparity at the end of an /I2/ is the same as the beginning running

disparity (right before the idle code). This rule ensures a negative running disparity at the end of an idle ordered set. A /Kx.y/ following a /K28.5/ is not replaced.

Figure 6–17 shows the input data codes versus the output data codes. The /D14.3/, /D24.0/, and /D15.8/ code groups were replaced by /D5.6/ or /D16.2/ (for /I1/ and /I2/ ordered sets), and /D21.5/ (part of the /C2/ ordered set) was not replaced.

Figure 6–17. Input Data Codes vs. Output Data Codes

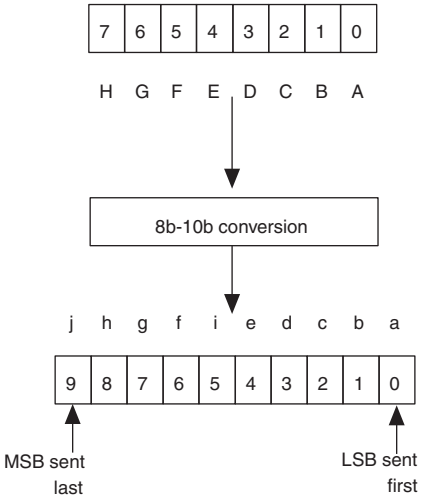


8B/10B Encoder

The 8B/10B encoder is part of the Stratix GX transceiver block. The 8B/10B encoder translates 8-bit data and a 1-bit control identifier (by using the tx_ctrlenable signal) into a 10-bit, DC-balanced data stream.

For more information about the 8B/10B code, refer to the 8B/10B Code section in the *Data & Control Codes* chapter of the *Stratix GX Device Handbook, Volume 2*. The 8B/10B encoder translates the 8-bit data or 8-bit control character to its 10-bit equivalent. Figure 6–18 shows the conversion format. The serializer sends the 10-bit data in order from LSB to MSB.

Figure 6–18. 8B/10B Conversion Format



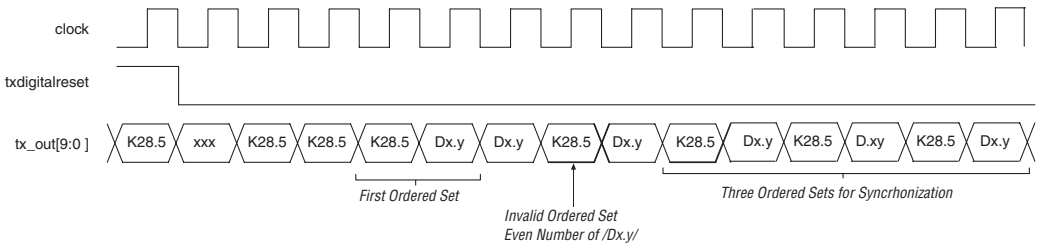
Reset

After power up or reset, the 8B/10B encoder in GIGE mode sends three /K28.5/ commas before user data can be sent. These commas affect the synchronization-ordered set transmission.

After reset (txdigitalreset), three /K28.5/ commas are sent automatically by the 8B/10B encoder. Depending on when you start outputting the synchronization sequence, there are an even or odd number of /Dx.y/ code groups sent by the transmitter before the synchronization sequence. The last of three automatically sent /K28.5/ commas and the first user-sent /Dx.y/ code groups are considered as one idle ordered set. This fact can be a problem if there are even numbers of /Dx.y/ code groups transmitted before the start of the synchronization sequence.

Figure 6–19 shows an example of an even number of /Dx.y/ code groups between the last automatically sent /K28.5/ comma and the first user sent /K28.5/. The first user-sent ordered set is ignored, so three additional ordered sets are required for proper synchronization. Although one set of invalid data is shown between the txdigitalreset signal going low and the first of three automatic K28.5, there can be more than one invalid data set.

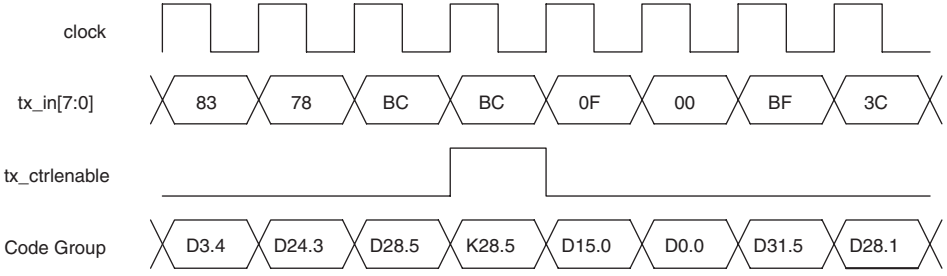
Figure 6–19. Even Number of /Dx.y/ Between Last Automatically Sent /K28.5/ & the First User-Sent /K28.5/



Control Code Encoding

The `tx_ctrlenable[]` signal determines when a control code must be inserted in the encoded data flow. When the `tx_ctrlenable[]` signal is low, the byte at `tx_in[]` is encoded as data. When the `tx_ctrlenable[]` signal is high, `tx_in[]` is encoded as a control word. The waveform in Figure 6–20 shows that the second 0xBC is encoded as a control code. The rest are encoded as data.

Figure 6–20. Control Word Identification Waveform



The 8B/10B encoder does not check that the code word you entered is one of the 12 valid codes. If an invalid control code is entered, the resulting 10-bit code is encoded as either invalid code (that does not map to a valid `/Dx.y/` or `/Kx.y/` code), or valid `/Dx.y/` code, depending on the value entered.

An example is the invalid encoding of a `/K24.1/` (data = `8'h38` + `tx_ctrlenable` = `1'b1`). Depending on the current running disparity, the `/K24.1/` can be encoded to be `10'b0110001100` (0x18C), which is equivalent to a `/D24.6/+` (0xD8 from the `RD+` column). An 8B/10B decoder decodes this incorrectly (based on the 8B/10B Fibre Channel specification).

GIGE Mode Clocking

GIGE Mode Channel Clocking

This section describes the details of clocking the transceiver, the internal clocking details, and the external clock ports in GIGE mode. Each block diagram shows the input and output port clocks. The MegaWizard Plug-In Manager by default selects a set of clocks for transmitters and receivers in a transceiver when GIGE mode is selected. The wizard also offers clock options, other than default, to facilitate your clocking schemes.

Figure 6–21. Default Configuration of altgxb Megafunction in GIGE Mode

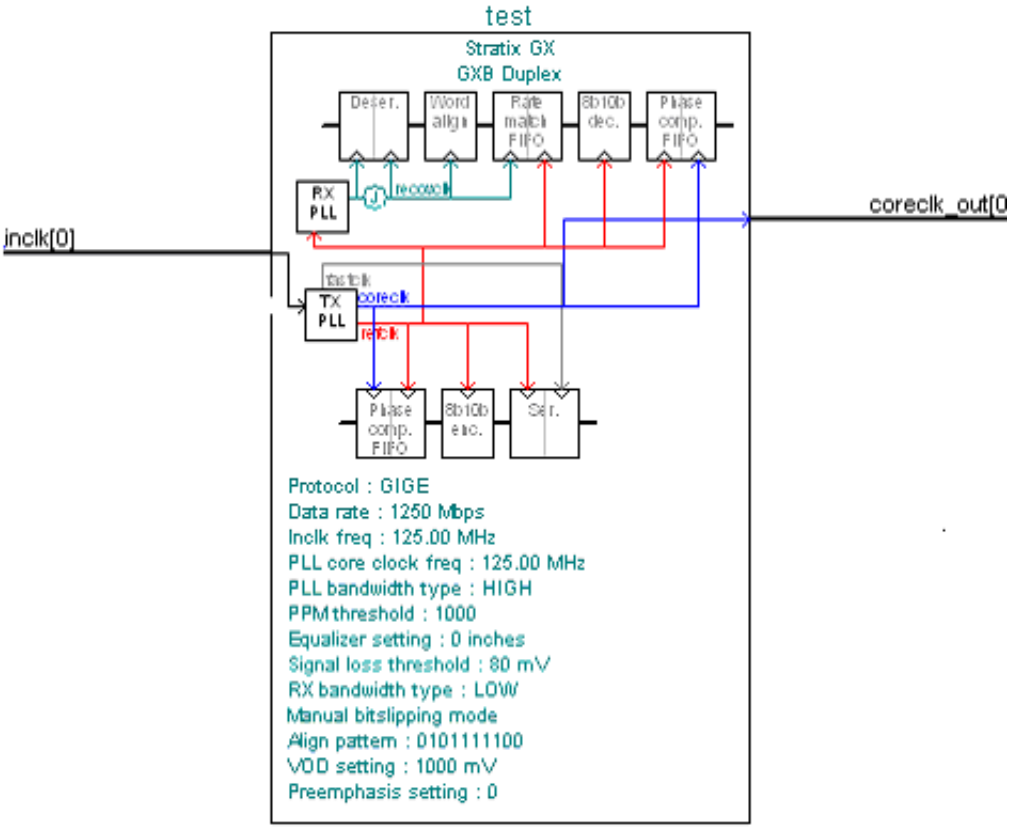


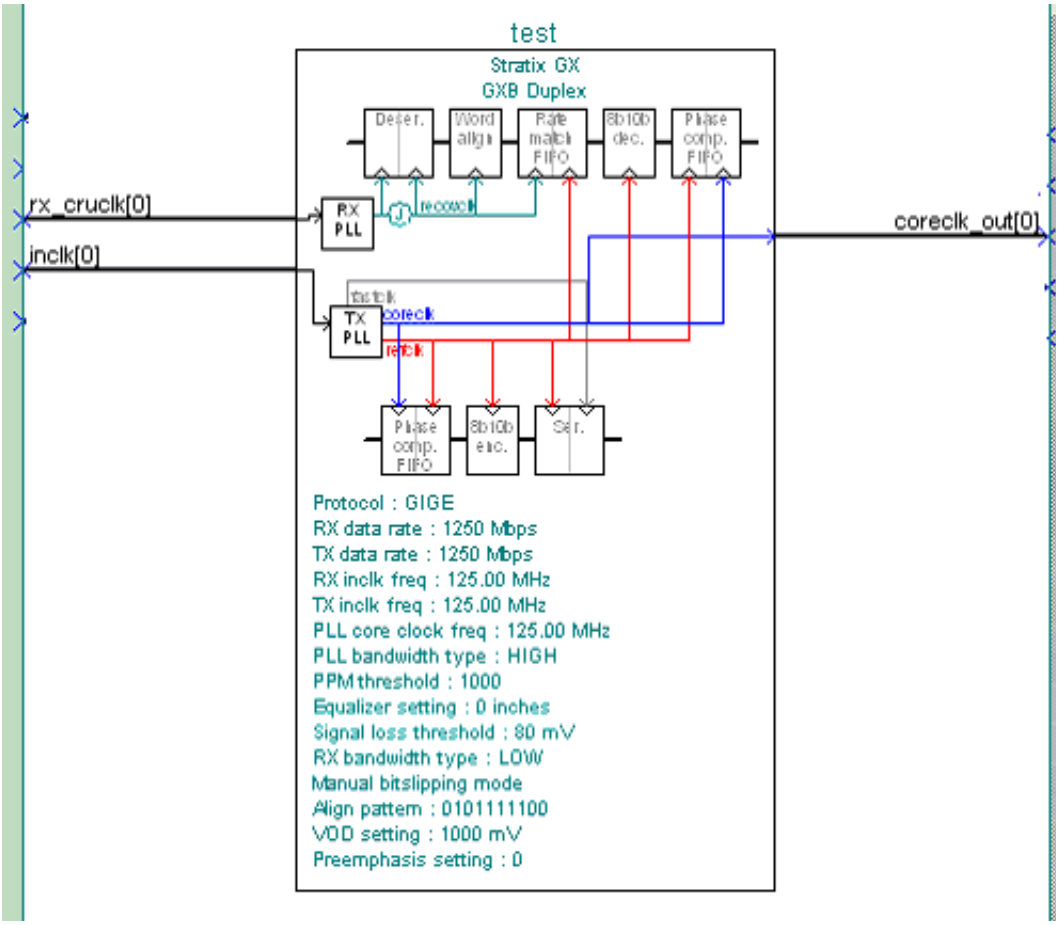
Figure 6–21 shows the altgxb megafunction configured so that the training receiver PLL with the transmitter PLL is enabled. The transmitter PLL is fed from an inclk port that can, in turn, be fed from a dedicated REFCLKB, global clock, regional clock, or fast regional clock source. The receiver logic is clocked by the recovered clock from the clock recovery unit up to the deskew FIFO buffer in the data path. Rate matching occurs between the recovered clock of the channel and refclk from the transmitter PLL. The data from the receiver's parallel interface is clocked by coreclk_out from the transmitter PLL. On the transmitter channel, the output of the transmitter PLL, coreclk_out, is sent from the logic array as an output and also loops back to clock the write side of the transmit phase compensation FIFO buffer (in this case, software automatically routes the connection) and the read side of the receive phase compensation FIFO buffer.

The training receiver PLL clock recovery unit (CRU) clock from the transmitter PLL can be disabled in the altgxb MegaWizard Plug-In Manager. Deselecting this option adds an additional RX_CRUCLK input reference clock port for the receiver PLL. This feature supports additional multiplication factors for the receiver PLL and also enables the separation of receiver and transmitter reference clocks. This configuration is shown in Figure 6–22.



For more information on parallel interface speeds, refer to the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1*.

Figure 6–22. Receiver PLL CRU Clock From Transmitter PLL is Disabled by Adding RX_CRUCLK



If the TX_CORECLK is enabled and the training receiver CRU clock from transmitter PLL is not enabled, and other default options are also enabled, this configuration has an independent rx_cruclk port that feeds the receiver PLL reference clock. This input clock port is available only when the receiver PLL is not trained by the transmitter PLL.

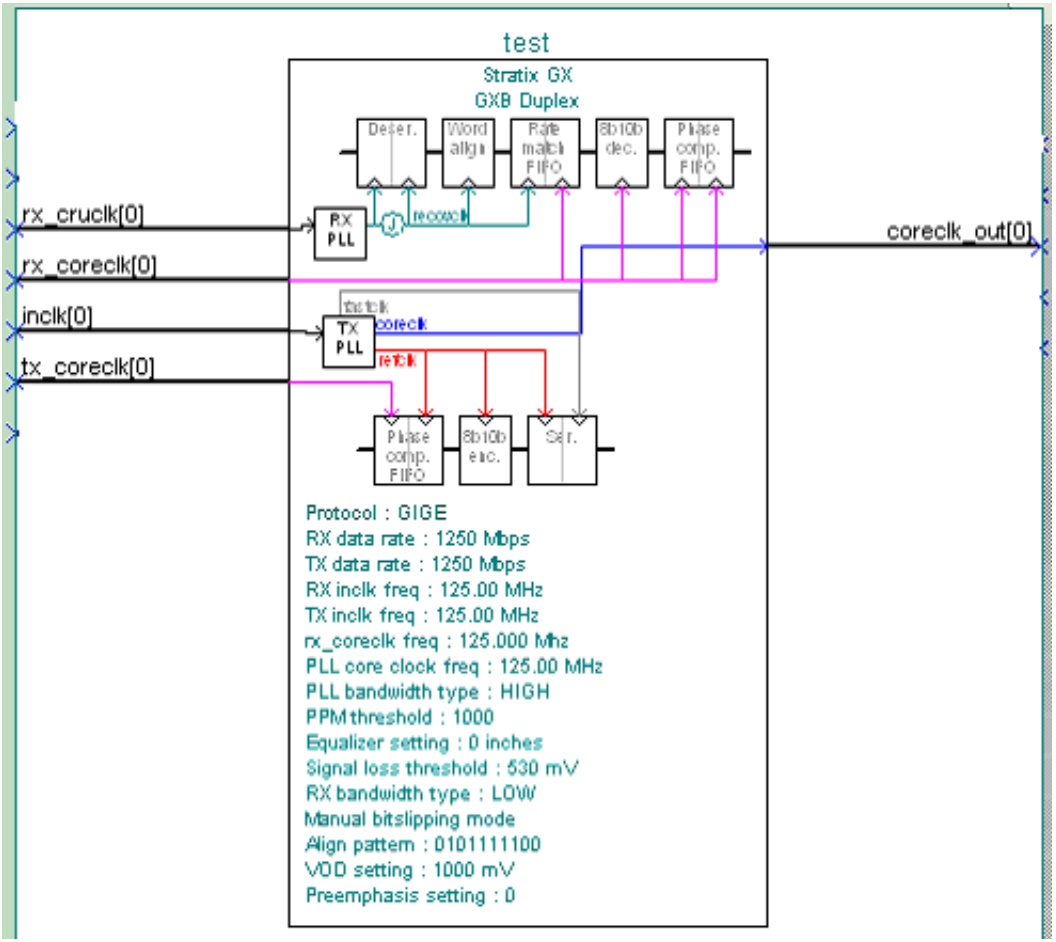
You can optionally enable the write clock of the transmitter phase compensation FIFO buffer to feed in a clock from the PLD logic array.

For example, if all the transmitter channels between transceiver blocks are from a common clock domain, the transceiver instantiations can use a total of one global resource versus one global per transceiver block if the TX_CORECLK option is not enabled. On the transmitter functionality screen and the "optional port of transmitter" section, if TX_CORECLK is selected as an input port, the default clocking scheme changes by using TX_CORECLK as the write clock for the phase compensation FIFO buffer. The user needs to connect CORECLK_OUT to TX_CORECLK using either gclk/rclk/fclk or logic array routing if CORECLK_OUT must be used. Alternatively, TX_CORECLK is supplied from a crystal or any other clock source, as long as it is frequency-locked to the read side of the phase compensation FIFO buffer on the transmit side.

In multicrystal environments, individual recovered clocks need to drive the read clock of the phase compensation FIFO. The Quartus® II software does this by default; you are not required to manually make the connection. tx_coreclk must be frequency matched with its respective read ports. The phase compensation FIFO buffer can only correct for phase, not for frequency differences. The receive parallel interface clocks the data to PLD based on CORECLK_OUT (the default option in the MegaWizard Plug-In Manager). RX_CORECLK is used as the read clock for the rate matching FIFO buffer. Before you can enable this feature, you must set the receiver to 8-bit mode.

Figure 6–23 shows the clock configuration with these optional input ports enabled.

Figure 6–23. TX_CORECLK & RX_CORECLK Enabled With RX_CRUCLK Port *Note (1)*



Note to Figure 6–23:
(1) The RX_CORECLK port is enabled for the rate-matching FIFO buffer.

Table 6–2 summarizes the clocks that are used in GIGE mode.

Table 6–2. Clocks in GIGE Mode (Part 1 of 2)

Clock	Port	Description
INCLK	Input	Input to transmitter PLL. Available as a port when transmitter PLL is instantiated.
RX_CRUCLK	Input	Input to CRU. Available as a port when CRU is not trained by transmitter PLL.

Table 6–2. Clocks in GIGE Mode (Part 2 of 2)

Clock	Port	Description
TX_CORECLK	Input	Clocks the write port of transmitter phase compensation FIFO buffer. Optional port in Quartus II software. Must be frequency matched to TX_PLL_CLK. If not available as a port, is fed by CORECLK_OUT through logic array routing.
RX_CORECLK	Input	Clocks the read port of receiver phase compensation FIFO buffer. Optional port in Quartus II software. If not available as a port, is fed by CORECLK_OUT through logic array routing.
CORECLK_OUT	Output	Output clock from transmitter PLL equivalent to TX_PLL_CLK. Available as a port if transmitter PLL is used.

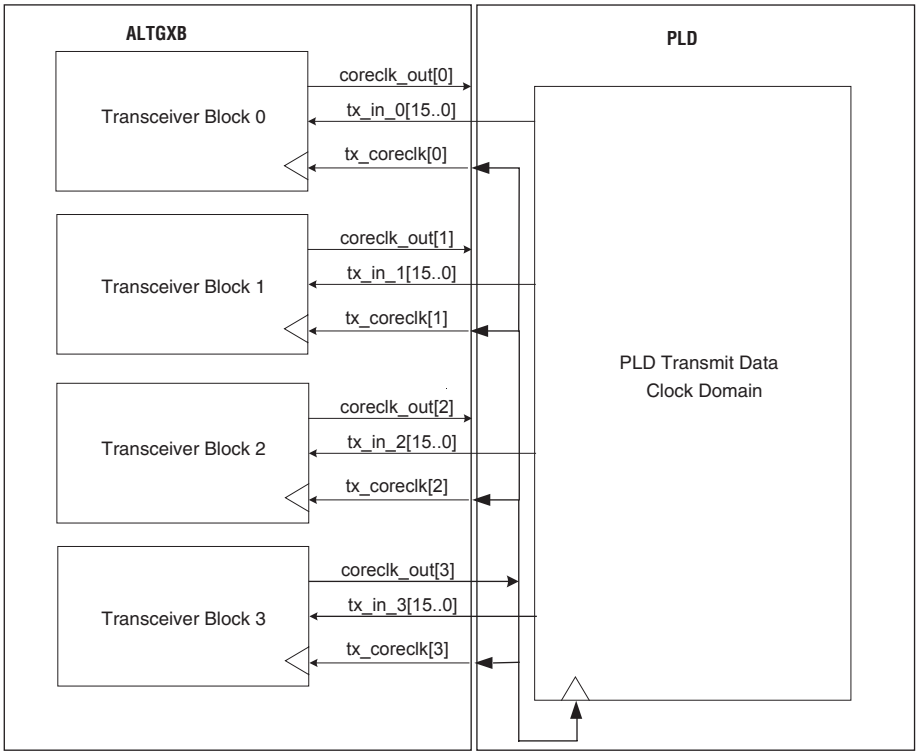
GIGE Mode Inter-Transceiver Clocking

This section provides guidelines for using transceiver interface clocking between the PLD logic array and transceiver channels when multiple transceiver blocks are active. Depending on which mode is supported by Stratix GX devices, each transceiver block has different transceiver-to-PLD interface clocking. Different input and output clocks are available based on the options provided by Quartus II MegaWizard Plug-In Manager's built-in functions. The number of supported channels varies based on the type of Stratix GX device you select. Because of the various configurations of the input and output clocks, consider the clocking schemes between transceiver blocks carefully to avoid future problems in the design cycle.

One of the clocking interfaces to consider while designing with Stratix GX is the transceiver-to-PLD interface. This clocking scheme can be further classified as the PLD-to-transmitter channel and receiver channel to the PLD.

In GIGE mode, the read port of the transmitter phase compensation FIFO buffer can either be clocked by the CORECLK_OUT or the TX_CORECLK port. The constraint on using TX_CORECLK port is that the clock must be frequency locked to the read port of the transmitter phase compensation FIFO buffer. Synchronous data transfers for a multi-transceiver configuration are accomplished with the TX_CORECLK port. The TX_CORECLK of multiple transceivers can be connected to a common clock domain, either from a single CORECLK_OUT signal or from a PLD system clock domain. This scheme is shown in Figure 6–24.

Figure 6–24. Example of a Multi-Transceiver PLD to Transmitter Interface Clocking Scheme



When TX_CORECLK is not enabled, the Quartus II software automatically routes the signal from the CORECLK_OUT port to the write clock of the phase compensation FIFO buffer using a global, regional, or fast regional resource. In a multi-transceiver configuration, this routing can lead to timing violations because the coreclk_out per transceiver block cannot guarantee a phase relationship. Therefore, clocking the TX_CORECLK with a common clock is recommended for synchronous transmission.

Another inter-transceiver consideration is the selection of the dedicated REFCLKB pin. Stratix GX channels are arranged in banks of four (called transceiver blocks). Each transceiver block has the ability to share a common reference clock through the inter-transceiver lines (IQ lines). The Stratix GX logic array clock usage can be reduced by using the IQ lines. The IQ lines are used when a REFCLKB input port from one transceiver block or channel drives other transceiver blocks or channels. The Quartus I software automatically determines the IQ line usage.

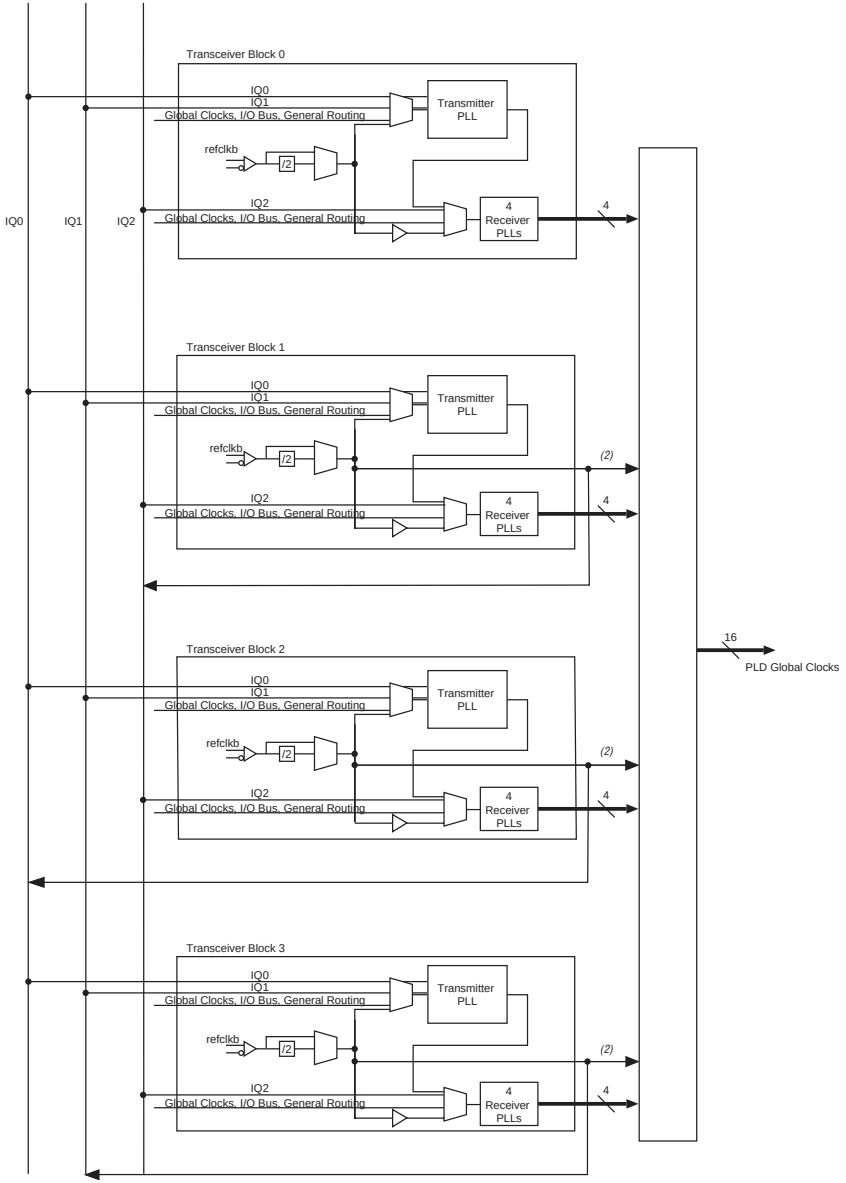
When determining the location of REFCLKB pins, consider what can be fed by the pin you choose. Table 6–3 shows the available IQ lines and which transceiver block REFCLKB drives the REFCLKB pin. This data is based on the number of transceiver channels in the Stratix GX device.

Table 6–3. REFCLKB Pin to Inter-Transceiver Line Connections

Channel Density	REFCLKB Pin in Transceiver Block Number	Channels in Transceiver Block	Inter-Transceiver Line Driven by REFCLKB
8 channels (EP1SGX10)	0	[3:0]	IQ2
	1	[7:4]	IQ0
16 channels (EP1SGX25)	0	[3:0]	—
	1	[7:4]	IQ2
	2	[11:8]	IQ0
	3	[15:12]	IQ1
20 channels (EP1SGX40)	0	[3:0]	—
	1	[7:4]	IQ2
	2	[11:8]	IQ0
	3	[15:12]	IQ1
	4	[19:16]	—

Figure 6–25 shows the transceiver routing with respect to inter-transceiver lines for the EP1SGX25F device. Be sure to use this information when placing REFCLKB pins. (When placing refclkb pins, refer to the *REFCLKB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* for information about analog reads and refclkb pin usage constraints.) For example, if a REFCLKB pin is required to feed a transmitter PLL using an inter-transceiver line, the REFCLKB pin cannot be in transceiver block 1, because IQ2 only feeds the receiver PLLs.

Figure 6–25. Inter-Transceiver Line Connections for EP1SGX25F Device *Note (1)*



Notes to Figure 6–25:

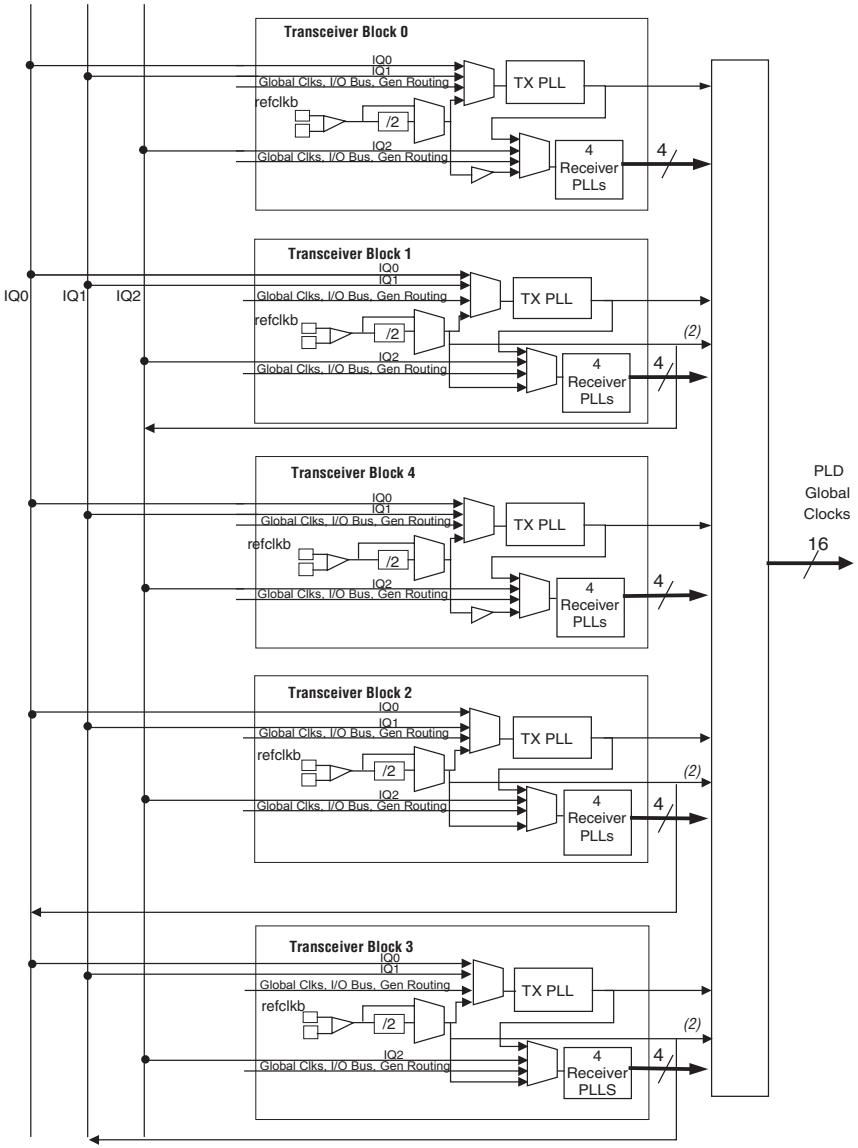
(1) IQ lines are inter-transceiver block lines.

(2) If the /2 pre-divider is used, the path to drive the PLD logic array, local, or global clocks is not allowed.

(3) There are four receiver PLLs in each transceiver block.

Figure 6–26 shows the transceiver routing with respect to inter-transceiver lines for the EP1SGX40G device. This device has an extra transceiver block (number 4), which is in the middle of all the transceiver blocks, as illustrated. Be sure to use this information when placing REFCLKB pins. (When placing refclkb pins, refer to the *REFCLKB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* for information about analog reads and refclkb pin usage constraints.) For example, if a REFCLKB pin is required to feed a transmitter PLL using an inter-transceiver line, the REFCLKB pin cannot be in transceiver block 1, because IQ2 only feeds the receiver PLLs.

Figure 6–26. Inter-Transceiver Line Connections for EP1SGX40G Device *Note (1)*



Notes to Figure 6–26:

(1) IQ lines are inter-transceiver block lines.

(2) If the /2 pre-divider is used, the path to drive the PLD logic array, local, or global clocks is not allowed.

(3) There are four receiver PLLs in each transceiver block.

GIGE Mode MegaWizard Plug-In Manager

This section describes the altgxb megafunction options for GIGE mode. Altera® recommends that the Stratix GX transceiver block be instantiated and parameterized through the altgxb MegaWizard Plug-In Manager in the Quartus II software. The Quartus II MegaWizard Plug-In Manager altgxb-In offers a graphical user interface (GUI) that organizes the altgxb options in easy-to-use sections. The MegaWizard Plug-In Manager also sets the correct ports and parameters automatically, based on the options and parameters you select. Invalid settings are automatically flagged in the wizard to avoid illegal configurations. The MegaWizard Plug-In Manager also disables any options that do not apply to GIGE mode.

Although you can instantiate the Stratix GX block directly by calling out the altgxb megafunction, Altera recommends that you use the MegaWizard Plug-In Manager to instantiate your altgxb megafunction to reduce the chance of invalid settings.

GIGE Mode MegaWizard Plug-In Manager Considerations

Each altgxb MegaWizard Plug-In Manager instantiation can use one or more transceiver blocks, based on the number of channels you select. There are four channels per transceiver block. If a MegaWizard Plug-In Manager instantiation uses fewer than four channels, the remaining channels in that transceiver block are no longer available for use.

Each instantiation must have similar functionality and data rates. If you wish to have transceiver blocks that differ in functionality or data rates, you can create a separate instantiation for each transceiver block.

Also, as mentioned in the clocking section, the wizard displays the configuration of the altgxb megafunction. This diagram changes dynamically based on the selected mode, options, and clocking schemes.

GIGE Mode altgxb MegaWizard Plug-In Manager Options

This section shows the MegaWizard Plug-In Manager pages where you select the options for a GIGE mode configuration.

Figure 6–27 shows page 3 of the altgxb MegaWizard Plug-In Manager in GIGE mode.

Figure 6–27. MegaWizard Plug-In Manager - altgxb (Page 3)

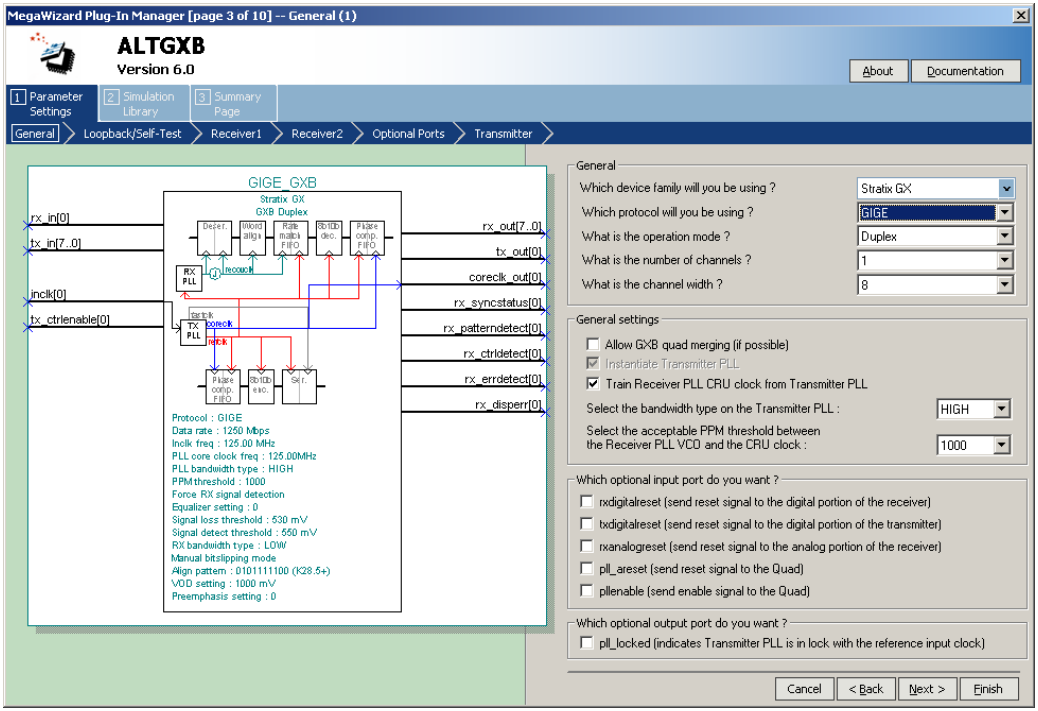


Table 6–4 describes the available options on page 3 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 6–4. MegaWizard Plug-In Manager Options (Page 3 for GIGE Mode) (Part 1 of 2)

altgxb Setting	Description
Which device family will you be using?	Stratix GX is the only option available.
Which protocol will you be using?	For the GIGE mode, you must select the GIGE protocol.
What is the operation mode?	GIGE protocol mode supports duplex, receiver-only, or transmitter-only operation modes.
What is the number of channels?	This value can be from 1 to the maximum number of channels available on the device. The Quartus II software automatically assigns the channels to a transceiver block unless input and output pins assignments are made to the channel's transceiver input and output pins.
What is the channel width?	8 bits is single width.

Table 6–4. MegaWizard Plug-In Manager Options (Page 3 for GIGE Mode) (Part 2 of 2)

altgxb Setting	Description
Allow GXB quad merging (if possible)	For information about this option, refer to the section "Stratix GX Transceiver Merging" on page 6–43.
Instantiate Transmitter PLL	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Train Receiver PLL CRU clock from Transmitter PLL	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the bandwidth type on the Transmitter PLL	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the acceptable PPM threshold between the Receiver PLL VCO and the CRU clock	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
`rxdigitalreset` (send reset signal to the digital portion of the receiver)	The `rxdigitalreset` port resets the digital blocks in the receiver channel. Each active receiver channel has its own digital reset.
`txdigitalreset` (send reset signal to the digital portion of the transmitter)	The `txdigitalreset` port resets the digital blocks of the transmitter channel. Each active transmitter channel has its own digital reset.
`rxanalogreset` (send reset signal to the analog portion of the receiver)	The `rxanalogreset` port resets the receiver's analog circuits, including the receiver PLL. Each active receiver channel has its own analog reset.
`pll_areset` (send reset signal to the Quad)	The `pll_areset` port resets the entire transceiver block (all receiver and transmitter digital and analog circuits, including receiver and transmitter PLLs).
`pllenable` (send enable signal to the Quad)	The `pllenable` port enables the entire transceiver block; if deasserted, the entire transceiver block is held in the reset condition.
`pll_locked` (indicates Transmitter PLL is in lock with the reference input clock)	For more information, refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 6–28 shows page 4 of the `altgxb` MegaWizard Plug-In Manager in GIGE mode.

Figure 6–28. MegaWizard Plug-In - altgxb (Page 4)

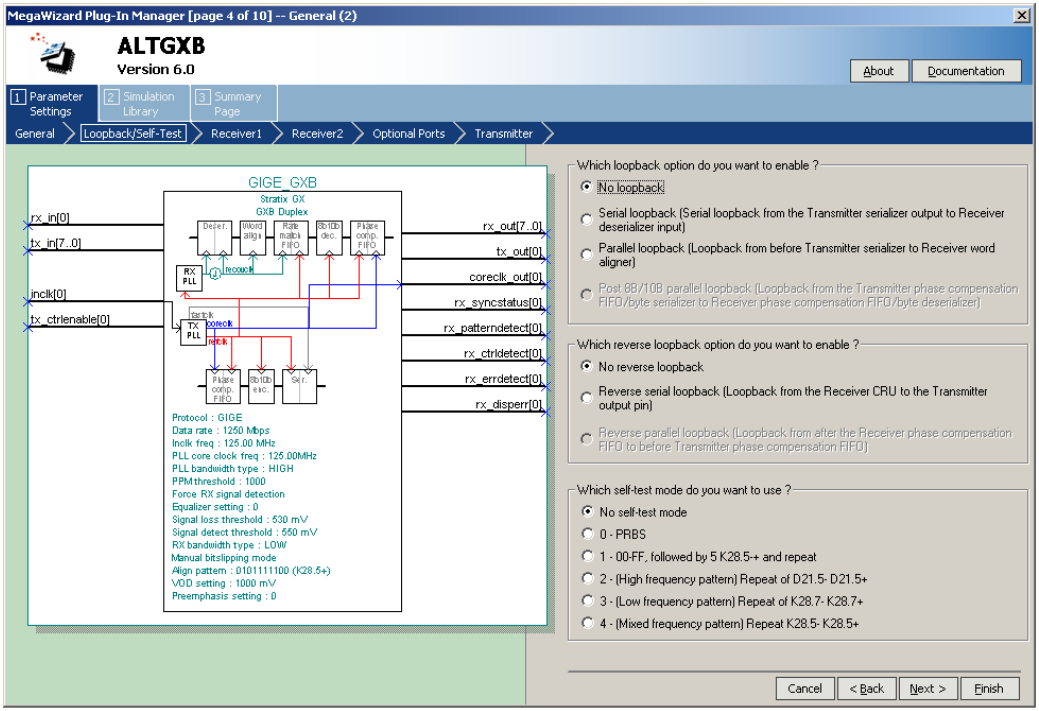


Table 6–5 describes the available options on page 4 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 6–5. MegaWizard Plug-In Manager Options (Page 4 for GIGE Mode)

altgxb Setting	Description
Which loopback option do you want to enable?	For more information, refer to the *Loopback Modes* chapter in volume 2 of the *Stratix GX Device Handbook*.
Which reverse loopback option do you want to enable?	For more information, refer to the *Loopback Modes* chapter in volume 2 of the *Stratix GX Device Handbook*.
Which self-test mode do you want to use?	For more information, refer to the *Stratix GX Built-In Self Test (BIST)* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 6–29 shows page 5 of the `altgxb` MegaWizard Plug-In Manager in GIGE mode.

Figure 6–29. MegaWizard Plug-In Manager - altgxb (Page 5)

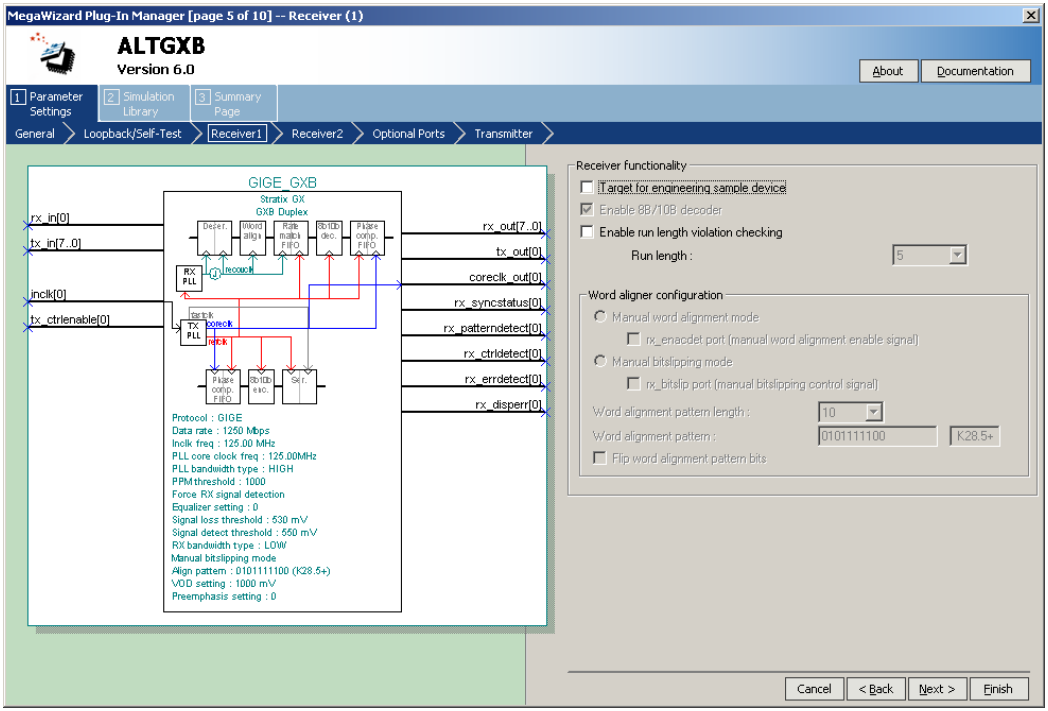


Table 6–6 describes the available options on page 5 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 6–6. MegaWizard Plug-In Manager Options (Page 5 for GIGE Mode) (Part 1 of 2)

altgxb Setting	Description
Target for engineering sample device	You must select this option if the design is targeted for an engineering sample (ES) device.
Enable 8B/10B decoder	In GIGE mode, this option is always enabled because data is always 8B/10B encoded.
Enable run-length violation checking	For more information, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Manual word alignment mode	This option is not available in GIGE mode.
`rx_enacdet` port (manual word alignment enable signal)	This option is not available in GIGE mode.
Manual bitslipping mode	This option is not available in GIGE mode.

Table 6–6. MegaWizard Plug-In Manager Options (Page 5 for GIGE Mode) (Part 2 of 2)

altgxb Setting	Description
rx_bitslip port (manual bitslipping control signal)	This option is not available in GIGE mode.
Word alignment pattern length	This option is not available in GIGE mode.
Word alignment pattern	The word aligner in GIGE mode is always set as a 10-bit K28.5 pattern. Both positive and negative disparities are checked.
Flip word alignment pattern bits	This option is not available in GIGE mode.

Figure 6–30 shows page 6 of the altgxb MegaWizard Plug-In Manager in GIGE mode.

Figure 6–30. MegaWizard Plug-In Manager - altgxb (Page 6)

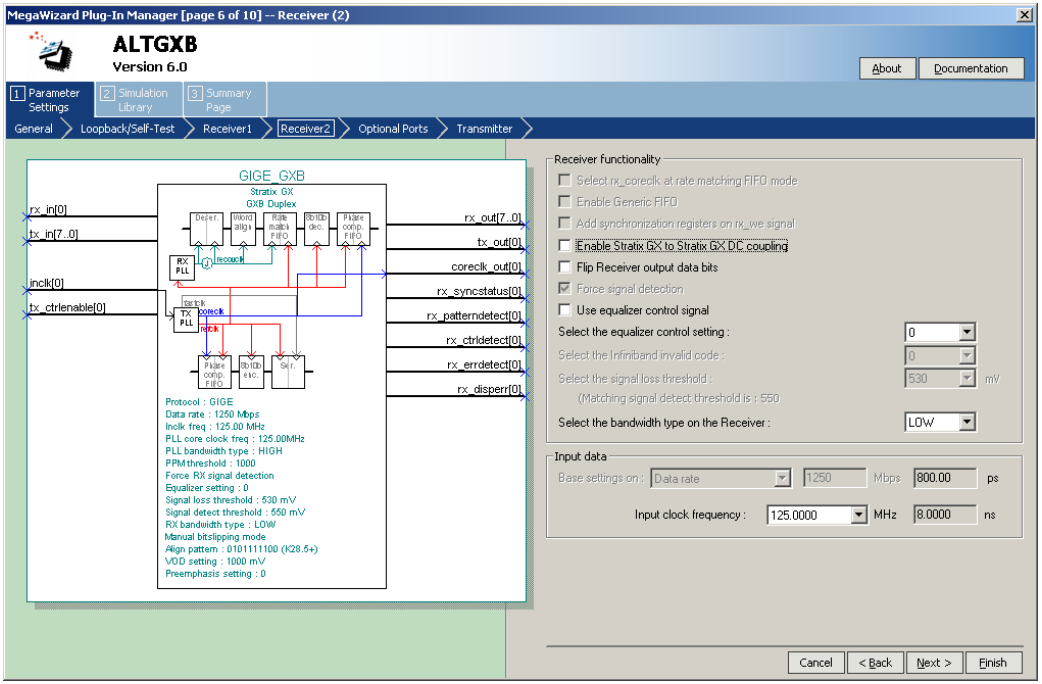


Table 6–7 describes the available options on page 6 of the MegaWizard Plug-In Manager for your altgxb custom megafunction variation.

Table 6–7. MegaWizard Plug-In Manager Options (Page 6 for GIGE Mode)

altgxb Setting	Description
Select rx_coreclk at rate matching FIFO mode	This option is not available in GIGE mode.
Enable Generic FIFO	This option is not available in GIGE mode, because a dedicated rate matching FIFO is included in the receiver data path by default.
Add synchronization registers on rx_we signal	This option is not available in GIGE mode.
Enable Stratix GX to Stratix GX DC coupling	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Force signal detection	The Force Signal Detect option is always on and cannot be turned off. The signal detect circuitry is always forced, so the rx_signaldetect is always set in GIGE mode.
Use equalizer control signal	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the equalizer control setting	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the Infiniband invalid code	This option is not available in GIGE mode.
Select the signal loss threshold	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the bandwidth type on the Receiver	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Base settings on	By default, the GIGE data rate is set to 1250 Mbps. Possible multiplication factors of the input clock are 2, 4, 5, 8, 10, 16, and 20. Multiplication factors of 2, 4, and 5 must use the refclkb pins. A multiplication factor of 2 also requires that the receiver PLL be trained by the transmitter PLL.

Figure 6–31 shows page 7 of the altgxb MegaWizard Plug-In Manager in GIGE mode.

Figure 6–31. MegaWizard Plug-In Manager - altgxb (Page 7)

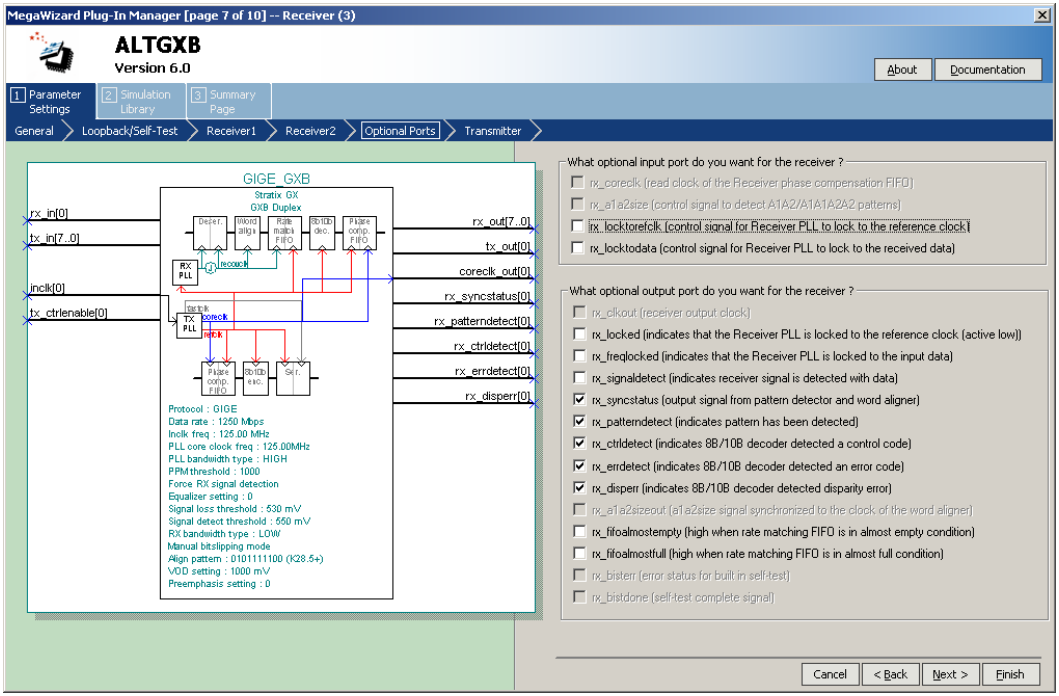


Table 6–8 describes the available options on page 7 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 6–8. MegaWizard Plug-In Manager Options (Page 7 for GIGE Mode) (Part 1 of 2)

altgxb Setting	Description
`rx_coreclk` (read clock of the Receiver phase compensation FIFO)	This option is not available in GIGE mode.
`rx_a1a2size` (control logic signal to detect A1A2/A1A1A2A2 patterns)	This option is not available in GIGE mode.
`rx_locktorefclk` (control signal for Receiver PLL to lock to the reference clock)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
`rx_locktodata` (control signal for Receiver PLL to lock to the received data)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
`rx_clkout` (receiver input clock)	This option is not available in GIGE mode.

Table 6–8. MegaWizard Plug-In Manager Options (Page 7 for GIGE Mode) (Part 2 of 2)

altgxb Setting	Description
rx_locked (indicates that the Receiver PLL is locked to the reference clock (active low))	Receiver PLL lock indicator. For rx_locked, Low = receiver PLL is locked to the reference clock.
rx_freqlocked (indicates that the Receiver PLL is locked to the input data)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_signaldetect (indicates receiver signal is detected with data)	rx_signaldetect is only available in XAUI or GIGE mode. Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook* for additional information.
rx_syncstatus (output signal from pattern detector and word aligner)	Indicates when the word aligner has aligned to the byte boundary. The rx_syncstatus signal goes high for one rx_clkout period when the word aligner aligns to the new byte boundary.
rx_patterndetect (indicates pattern has been detected)	Similar to rx_syncstatus, except that rx_patterndetect asserts only when the word alignment pattern appears in the data stream within the synchronized byte boundary.
rx_ctrldetect (indicates 8B/10B decoder detected a control code)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_errdetect (indicates 8B/10B decoder detected an error code	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_disperr (indicates 8B/10B decoder detected disparity error)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_a1a2sizeout (a1a2size signal synchronized to the clock of the word aligner)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_fifoalmostempty (high when rate matching FIFO is in almost empty condition)	You can include this rate matching FIFO status signal by enabling this option. When driven HIGH, this signal indicates an almost empty condition for the rate matching FIFO.
rx_fifoalmostfull (high when rate matching FIFO is in almost full condition)	You can include this rate matching FIFO status signal by enabling this option. When driven HIGH, this signal indicates an almost full condition for the rate matching FIFO.
rx_bisterr (error status for built-in self-test)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.
rx_bistdone (self-test complete signal)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 6–32 shows page 8 of the altgxb MegaWizard Plug-In Manager in GIGE mode.

Figure 6–32. MegaWizard Plug-In Manager - altgxb (Page 8)

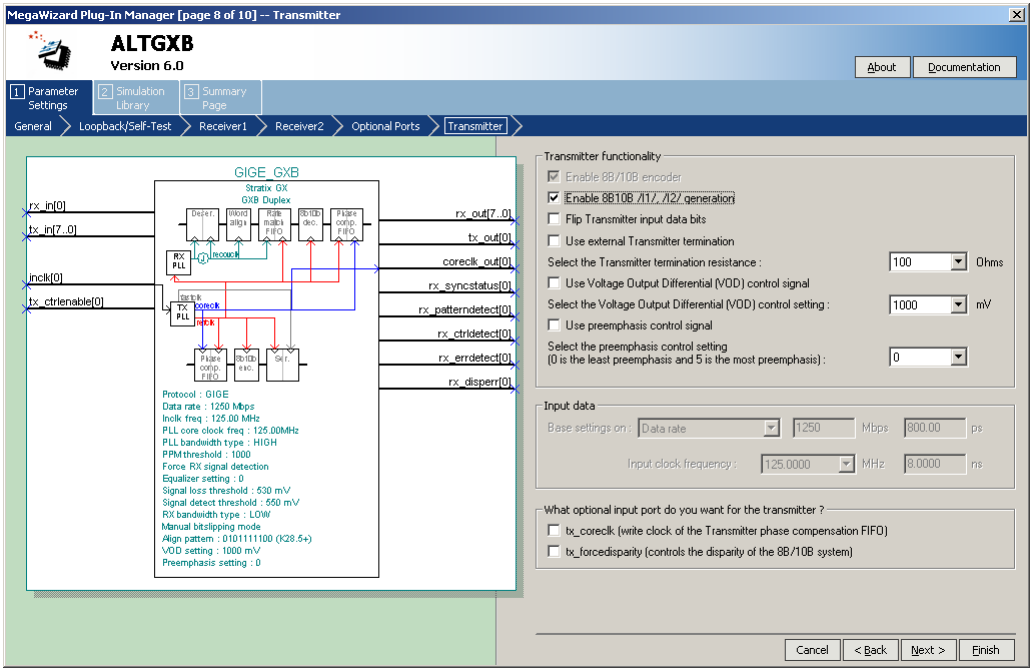


Table 6–9 describes the available options on page 8 of the MegaWizard Plug-In Manager for your `altgxb` custom megafunction variation.

Table 6–9. MegaWizard Plug-In Manager Options (Page 8 for GIGE Mode) (Part 1 of 2)

altgxb Setting	Description
Enable 8B/10B encoder	In GIGE mode, this option is always enabled because data is always 8B/10B encoded.
Enable 8B/10B /I1/, /I2/ generation	Selecting this option enables the transmitter to replace any /Dx.y/ following a /K28.5/ with either /D5.6/ or /D16.2/ depending on the running disparity before /K28.5/. Refer to the section "Idle Generation" on page 6–16 for more information.
Use external Transmitter termination	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Use Voltage Output Differential (VOD) control signal	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the Voltage Output Differential (VOD) control setting	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.

Table 6–9. MegaWizard Plug-In Manager Options (Page 8 for GIGE Mode) (Part 2 of 2)

altgxb Setting	Description
Use preemphasis control signal	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Select the preemphasis control setting (0 is the least preemphasis and 5 is the most preemphasis)	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Base settings on	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
Input clock frequency	For information about this option, refer to the *Stratix GX Analog Description* chapter in volume 2 of the *Stratix GX Device Handbook*.
`tx_coreclk` (write clock of the Transmitter phase compensation FIFO buffer	You can optionally choose the write clock of the transmitter phase compensation FIFO buffer. This clock should be frequency locked with the internal reference clock because the phase compensation FIFO buffer cannot tolerate frequency variations and contains no error flags.
`tx_forcedisparity` (controls the disparity of the 8B/10B system)	Refer to the *Ports & Parameters* chapter in volume 2 of the *Stratix GX Device Handbook*.

Figure 6–33 shows page 9, the Simulation Libraries page, of the MegaWizard Plug-In Manager for the GIGE protocol set up.

Figure 6–33. MegaWizard Plug-In Manager - altgxb (Page 9)

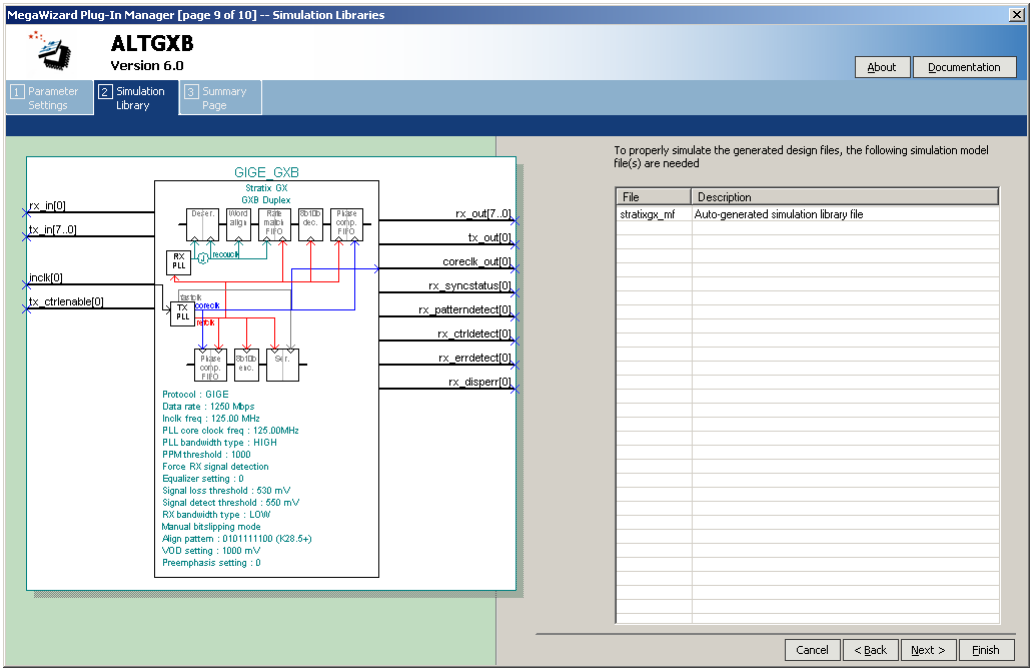
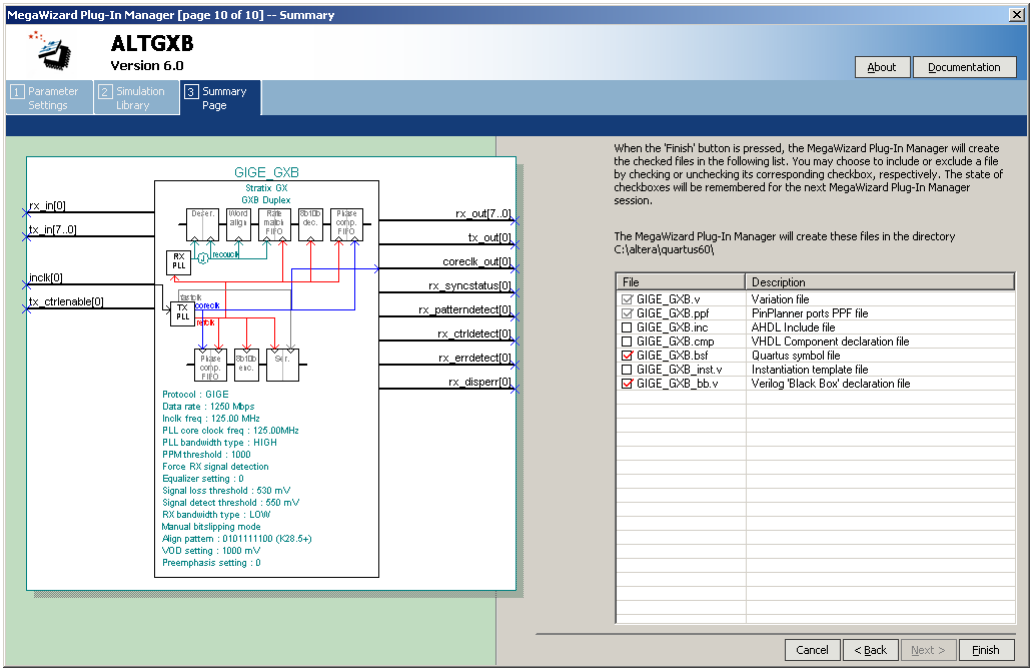


Figure 6–34 shows page 10 of the MegaWizard Plug-In Manager for the GIGE protocol set up. You can select optional files on this page. After you make your selections, click **Finish** to generate the files.

Figure 6–34. MegaWizard Plug-In Manager - altgxb (Page 10)



Stratix GX Transceiver Merging

A transceiver block contains four transceivers. In a design, an `altgxb` instantiation is placed in one or more transceiver blocks and potentially leaves unused transceivers in a block. For example, a six transceiver instantiation completely fills one transceiver block and half fills a second, taking up two full transceiver blocks. If another instantiation is in the design, it is placed the same way. For example, an instantiation of two transmitters takes up a third transceiver block. Merging two of the partially filled transceiver blocks into one transceiver block reduces the resources used and allows a design to fit into a device with fewer transceiver blocks.

The `altgxb` MegaWizard Plug-In Manager in the Quartus II software has a feature that allows merging of similar quads (transceiver blocks). With a few exceptions, transceiver blocks can be merged if the options

chosen in the wizard are the same. The Quartus II software merges the transceiver blocks automatically if you turn on the merging option for each instantiation. Table 6–10 shows the considerations for merging.

Table 6–10. Stratix GX Merging Considerations

Merging Rules	Required to Match Between Transceiver Blocks	Not Required to Match Between Transceiver Blocks

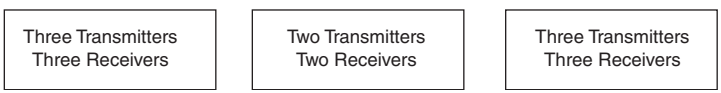
MegaWizard Plug-In Manager options	`USE_8B_10_MODE`
	`USE_DOUBLE_DATA_MODE`
	`CHANNEL_WIDTH`
	`SYNC_MODE`
	`DATA_RATE`
	`TRANSMIT_PROTOCOL`
		VOD
		Pre-emphasis
		Equalization
		Number of transmitters
		Number of receivers

Input control signals must be shared across transceiver blocks and must be from the same source	Clock
	`CRU_CLOCK`
	`PLL_RESET`
	`PLL_ENABLE`	

The merging partial transceiver blocks must reduce the number of transceiver block when merged	The total number of receivers and transmitters must be tx ≤4 x *n* and rx ≤4 x *n*, where *n* is the reduced number of transceiver blocks remaining after merging.	

The Quartus II software does not merge two transceiver blocks if they won't completely fit into one. It can, however, merge three transceiver blocks into two. Figure 6–35 shows a configuration with three transceiver blocks that can potentially be merged.

Figure 6–35. Three Transceiver Configuration



In Figure 6–35, two transceiver blocks cannot merge because there would be extra channels remaining. But because there are three transceiver blocks, the Quartus II software can merge them into two with no remaining channels.

If you turn on the merging option, the Quartus II software automatically merges transceiver blocks if possible or provides a warning during compilation if merging is not possible. The merging feature can reduce the transceiver block resources used in your design.

Design Example

The design example shows the GIGE synchronization sequence and illustrates what happens when the receiver loses synchronization, as described in clause 36 of the IEEE 802.3 specification. To simplify the documentation process, the design is implemented in Verilog hardware description language (HDL).

Design Description

When the protocol is specified as GIGE, synchronization is achieved on receiving three (/K28.5/, /Dx.y/) ordered sets. Each /K28.5/ is separated by any odd number of /Dx.y/ code groups. Invalid code groups are not supported during the synchronization stage. If at any time four invalid code groups are received separated by fewer than three valid code groups, synchronization is lost. This design example shows both the transmission of the synchronization sequence and the transmission of the invalid error codes that cause the loss of synchronization.

```
`define reset 3'd0
`define donothing 3'd1
`define sync 3'd2
`define tx_err 3'd3
`define count 3'd4
`define txk 3'd5
module gige8b10btest(
clk,
rx_in,
patterndetect,
ctrldetect,
errdetect,
syncstatus,
disperr,
rxout,
txout);
input clk, rx_in;
output patterndetect, ctrldetect,
errdetect,syncstatus,disperr;
output [7:0] rxout;
output txout;
reg [3:0] curst,nextst;
reg reset, txctrl;
reg [6:0] globalcntr;
reg [3:0] kcntr;
reg [7:0] datacntr, kdata,txdata;
reg tff;
wire [7:0] rxout;
wire coreclk, rxclk, rx_in;
wire patterndetect, ctrldetect,
errdetect,syncstatus,disperr;
//GXB instantiation
gige8b10bgxb gige8b10bgxb_inst(
```

```
.pll_areset(1'b0),
.pllenable(1'b1),
.inclk(clk),
.rx_in(rx_in),
.rx_slpbk(1'b1),
.rxanalogreset(1'b0),
.tx_in(txdata),
.tx_ctrlenable(txctrl),
.rxdigitalreset(reset),
.txdigitalreset(1'b0),
.rx_disperr(disperr),
.rx_patterndetect(patterndetect),
.rx_ctrldetect(ctrldetect),
.tx_out(txout),
.rx_errdetect(errdetect),
.coreclk_out(coreclk),
.rx_out(rxout),
.rx_syncstatus(syncstatus));
//governing counter
always@(posedge clk)
    globalcntr <= globalcntr +1;
//control character counter
always@(posedge clk)
    if(kcntr==4'd11 || reset==1'b1)
        kcntr<=4'b0;
    else
        kcntr<=kcntr+1;
//data counter
always@(posedge clk or posedge reset)
    if(reset==1'b1)
        datacntr<=1'b0;
    else
        datacntr<=datacntr+1;
//control character decode
always@(kcntr)
case (kcntr)
    0: kdata=8'h1c;  //k28.0
    1: kdata=8'h3c;  //k28.1
    2: kdata=8'h5c;  //k28.2
    3: kdata=8'h7c;  //k28.3
    4: kdata=8'h9c;  //k28.4
    5: kdata=8'hbc;  //k28.5
    6: kdata=8'hdc;  //k28.6
    7: kdata=8'hfc;  //k28.7
    8: kdata=8'hf7;  //k23.7
    9: kdata=8'hfb;  //k27.7
    10:kdata=8'hfd;  //k29.7
    11:kdata=8'hfe;  //k30.7
//  12:kdata=8'hff;  //invalid code
    default:kdata=8'hbc;
endcase
always@(globalcntr or curst)
```

```
case(globalcntr)
    0:
       nextst=`reset;//resets receiver
    1:
       nextst=`sync; //sends out 3 idle ordered sets
    8:
       nextst=`count; //sending counter values
    40:
       nextst=`txk; //sending control characters
    52:
       nextst=`count; //sending counter values
    60:
       nextst=`tx_err;  //sending 4 illegal codes
    64:
       nextst=`donothing; //do nothing until resync
    default:
       nextst= curst;
endcase
always @(posedge clk)
    curst<=nextst;
always@(posedge clk)
case(curst)
    `reset:  //resets receiver
    begin
       reset<=1;
       txctrl<=0;
       txdata<=datacntr;
    end
    `donothing:  //sends out /D0.0/
    begin
       reset<=0;
       txctrl<=0;
       txdata<=8'h00;
    end
    `sync:  //sends alternating /K28.5/ and /D31.7/
    begin
       reset<=0;
       if (globalcntr[0]==1)
       begin
          txdata<=8'hbc;
          txctrl<=1;
       end
       else
       begin
          txdata<=8'hff;
          txctrl<=0;
       end
    end
    `tx_err:  //sends an out of bounds control code
/K31.7/
    begin
       reset<=0;
```

```
            txctrl<=1;
            txdata<=8'hff;
        end
        `count: //sends out value of a counter
        begin
            reset<=0;
            txctrl<=0;
            txdata<=datacntr;
        end
        `txk:  //sends out all 12 K codes
        begin
            reset<=0;
            txctrl<=1;
            txdata<=kdata;
        end
        default:
        begin
            reset<=0;
            txctrl<=0;
            txdata<=datacntr;
        end
    endcase
    endmodule

    ALTGXB

    module gige8b10bgxb (
        pll_areset,
        pllenable,
        inclk,
        rx_in,
        rx_slpbk,
        rxanalogreset,
        tx_in,
        tx_ctrlenable,
        rxdigitalreset,
        tx_forcedisparity,
        txdigitalreset,
        rx_disperr,
        rx_patterndetect,
        rx_ctrldetect,
        tx_out,
        rx_errdetect,
        coreclk_out,
        rx_out,
        rx_syncstatus);
        input[0:0]  pll_areset;
        input[0:0]  pllenable;
        input[0:0]  inclk;
        input[0:0]  rx_in;
        input[0:0]  rx_slpbk;
        input[0:0]  rxanalogreset;
```

```
input[7:0]  tx_in;
input[0:0]  tx_ctrlenable;
input[0:0]  rxdigitalreset;
input[0:0]  tx_forcedisparity;
input[0:0]  txdigitalreset;
output[0:0]  rx_disperr;
output[0:0]  rx_patterndetect;
output[0:0]  rx_ctrldetect;
output[0:0]  tx_out;
output[0:0]  rx_errdetect;
output[0:0]  coreclk_out;
output[7:0]  rx_out;
output[0:0]  rx_syncstatus;
wire [0:0] sub_wire0;
wire [0:0] sub_wire1;
wire [0:0] sub_wire2;
wire [0:0] sub_wire3;
wire [7:0] sub_wire4;
wire [0:0] sub_wire5;
wire [0:0] sub_wire6;
wire [0:0] sub_wire7;
wire [0:0] rx_disperr = sub_wire0[0:0];
wire [0:0] rx_patterndetect = sub_wire1[0:0];
wire [0:0] tx_out = sub_wire2[0:0];
wire [0:0] rx_ctrldetect = sub_wire3[0:0];
wire [7:0] rx_out = sub_wire4[7:0];
wire [0:0] rx_errdetect = sub_wire5[0:0];
wire [0:0] coreclk_out = sub_wire6[0:0];
wire [0:0] rx_syncstatus = sub_wire7[0:0];
altgxbaltgxb_component (
          .pll_areset (pll_areset),
          .pllenable (pllenable),
          .inclk (inclk),
          .rx_in (rx_in),
          .rx_slpbk (rx_slpbk),
          .tx_in (tx_in),
          .rxanalogreset (rxanalogreset),
          .tx_ctrlenable (tx_ctrlenable),
          .rxdigitalreset (rxdigitalreset),
          .tx_forcedisparity (tx_forcedisparity),
          .txdigitalreset (txdigitalreset),
          .rx_disperr (sub_wire0),
          .rx_patterndetect (sub_wire1),
          .tx_out (sub_wire2),
          .rx_ctrldetect (sub_wire3),
          .rx_out (sub_wire4),
          .rx_errdetect (sub_wire5),
          .coreclk_out (sub_wire6),
          .rx_syncstatus (sub_wire7));
defparam
   altgxb_component.align_pattern = "P0101111100",
   altgxb_component.align_pattern_length = 10,
```

```
        altgxb_component.allow_gxb_merging = "OFF",
        altgxb_component.channel_width = 8,
        altgxb_component.clk_out_mode_reference = "ON",

altgxb_component.consider_enable_tx_8b_10b_i1i2_gener
ation = "ON",

altgxb_component.consider_instantiate_transmitter_pll
_param = "ON",
        altgxb_component.data_rate = 1250,
        altgxb_component.data_rate_remainder = 0,
        altgxb_component.disparity_mode = "ON",
        altgxb_component.dwidth_factor = 1,

altgxb_component.enable_tx_8b_10b_i1i2_generation =
"ON",
        altgxb_component.equalizer_ctrl_setting = 0,
        altgxb_component.flip_rx_out = "OFF",
        altgxb_component.flip_tx_in = "OFF",
        altgxb_component.force_disparity_mode = "OFF",
        altgxb_component.for_engineering_sample_device
= "OFF",
        altgxb_component.instantiate_transmitter_pll =
"ON",
        altgxb_component.intended_device_family =
"Stratix GX",
        altgxb_component.loopback_mode = "SLB",
        altgxb_component.lpm_type = "altgxb",
        altgxb_component.number_of_channels = 1,
        altgxb_component.number_of_quads = 1,
        altgxb_component.operation_mode = "DUPLEX",
        altgxb_component.pll_bandwidth_type = "HIGH",
        altgxb_component.pll_inclock_period = 8000,
        altgxb_component.preemphasis_ctrl_setting = 0,
        altgxb_component.protocol = "GIGE",
        altgxb_component.reverse_loopback_mode =
"NONE",
        altgxb_component.run_length_enable = "OFF",
        altgxb_component.rx_bandwidth_type = "NEW_LOW",
        altgxb_component.rx_data_rate = 1250,
        altgxb_component.rx_data_rate_remainder = 0,
        altgxb_component.rx_enable_dc_coupling = "OFF",
        altgxb_component.rx_force_signal_detect = "ON",
        altgxb_component.rx_ppm_setting = 1000,
        altgxb_component.signal_threshold_select = 530,
        altgxb_component.tx_termination = 2,
        altgxb_component.use_8b_10b_mode = "ON",
        altgxb_component.use_auto_bit_slip = "ON",
        altgxb_component.use_channel_align = "OFF",
        altgxb_component.use_double_data_mode = "OFF",
        altgxb_component.use_equalizer_ctrl_signal =
"OFF",
```

```
        altgxb_component.use_preemphasis_ctrl_signal =
"OFF",
        altgxb_component.use_rate_match_fifo = "ON",
        altgxb_component.use_rx_clkout = "OFF",
        altgxb_component.use_rx_coreclk = "OFF",
        altgxb_component.use_rx_cruclk = "OFF",
        altgxb_component.use_self_test_mode = "OFF",
        altgxb_component.use_symbol_align = "ON",
        altgxb_component.use_vod_ctrl_signal = "OFF",
        altgxb_component.vod_ctrl_setting = 1000;
endmodule
```

Simulation Waveform & Hardware Verification Results

Figures 6–36 and 6–37 show the complete synchronization sequence from the transmitter to the receiver for the SignalTap® II logic analyzer and the Quartus II software, respectively. The GIGE duplex channel is configured in a serial loopback mode. The synchronization pattern is sent by the transmitter. The receiver sends a `K28.4` character (`8' b9C + ctrl`) until synchronization is achieved. Although any odd number of valid `/Dx.y/` codes is supported between each `/K28.5/` code, one `/Dx.y/` is shown in this example.

The `8'h00` shown on the `rx_out` bus results from the reset pulse when the pipelined registers are reset in the receiver block.

There is good correlation between the SignalTap II logic analyzer results and the Quartus II software simulation.

Figure 6–36. GIGE Synchronization Sequence SignalTap II Results

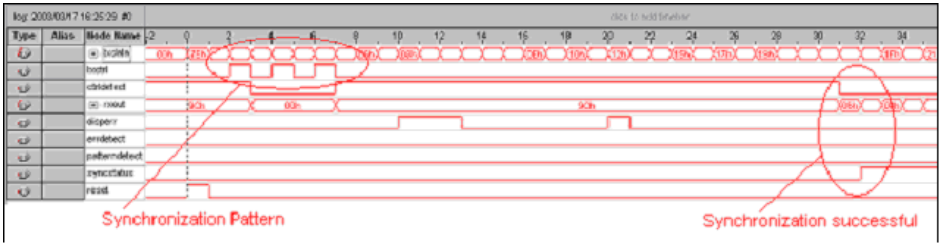
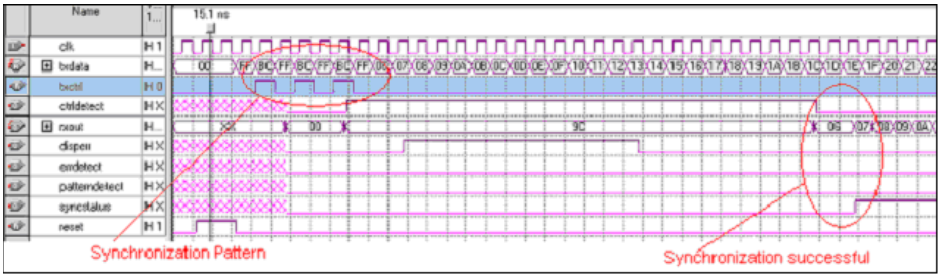


Figure 6–37. GIGE Synchronization Sequence Quartus II Software Simulation Results



Figures 6–38 and 6–39 show the loss of GIGE synchronization on receiving invalid code groups from the SignalTap II logic analyzer and the Quartus II software, respectively. On receiving four invalid codes that are separated by fewer than three valid codes, the receiver signals a loss of synchronization by deasserting the `rx_syncstatus` signal and sending a `/K28.4/` code group (`8'h9C + ctrl`). In the example, four invalid codes are transmitted with zero valid codes in between.

Figure 6–38. Loss of Synchronization SignalTap II Logic Analyzer Results

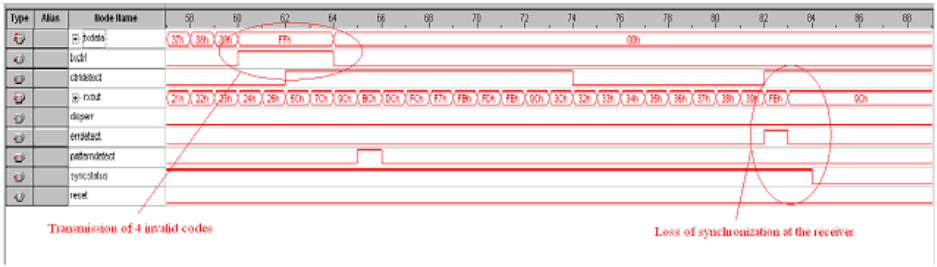
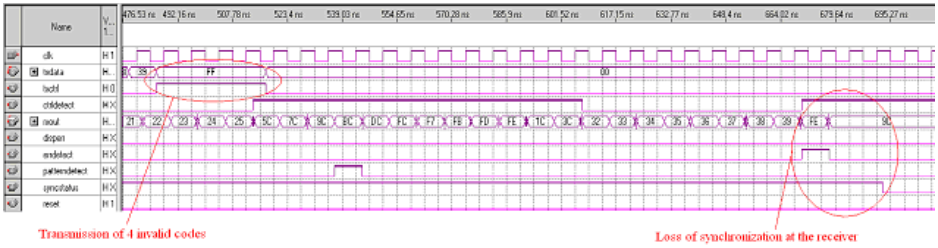


Figure 6–39. Loss of Synchronization Quartus II Simulation Results



7. Loopback Modes

SGX52007-1.2

Introduction

You can apply several loopback modes to the Stratix® GX block. The main forms of loopback are as follows:

- Serial loopback
- Parallel loopback
- Reverse serial loopback

Loopback refers to feeding the data from the transmitter directly to the receiver. Reverse loopback refers to feeding the data from the receiver directly to the transmitter. Serial loopback and parallel loopback feed data from the transmitter block to the receiver. Reverse serial loopback feeds the data from the receiver to the transmitter.

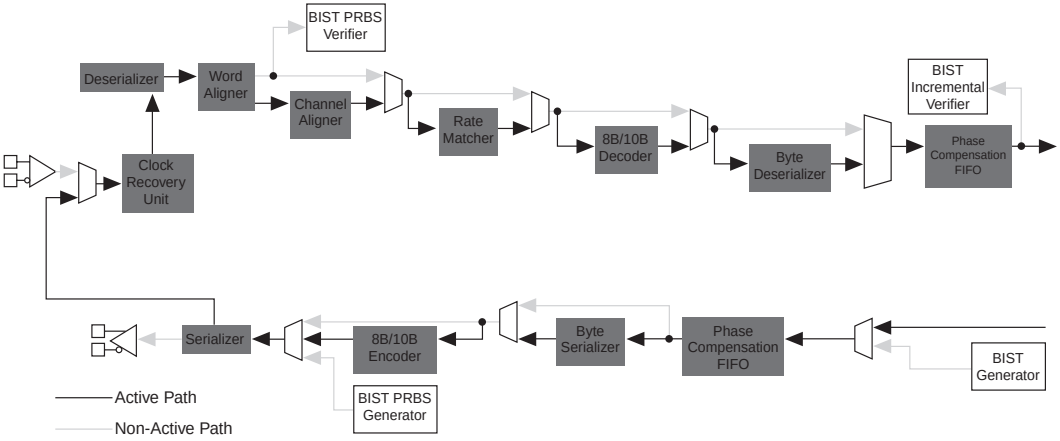
Serial Loopback

Figure 7–1 shows the data path for serial loopback. A data stream is fed to the transmitter from the FPGA logic array and has the option of using all the blocks in the transmitter. The data then traverses from the transmitter in serial form to the receiver. The serial data is the data that is transmitted from the Stratix GX device. Once the data enters the receiver in serial form, it can use any of the receiver blocks and is then fed into the FPGA logic array. The PRBS block generates data when using serial loopback.

Serial loopback is dynamically enabled on a channel-by-channel basis using the `rx_slpbk` port. When serial loopback is enabled, the output swing is reduced on the `tx_out[]` port except when the V_{OD} is set to 400 mV. In that case, the output is tri-stated.

Serial loopback is often used to check the analog portion of the transceiver. The data is retimed through different clock domains and an alignment pattern is still necessary for the word aligner.

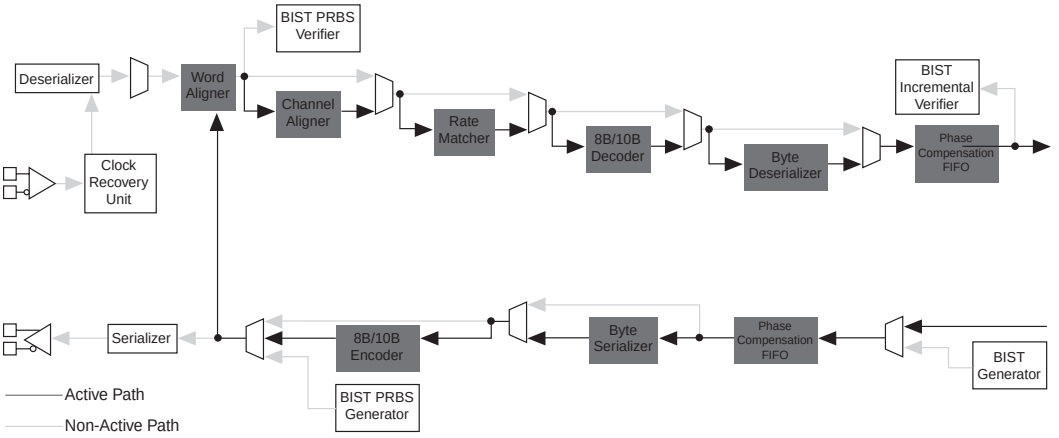
Figure 7–1. Stratix GX Block in Serial Loopback Mode



Parallel Loopback

Figure 7–2 shows the data path for parallel loopback. A data stream is fed to the transmitter from the FPGA logic array and has the option of using blocks in the transmitter block. The data then exits the transmitter into the receiver in parallel form before entering the serializer. The data enters the receiver block after the deserializer and has the option of using any of the subsequent receiver blocks before being output by the receiver into the FPGA logic array. The PRBS block generates data. When using parallel loopback, the `tx_out` ports are active, and the differential output voltage on the `tx_out` ports is based on the current setting in the Quartus® II software or on the user setting.

Figure 7–2. Stratix GX Block in Parallel Loopback Mode



Reverse Serial Loopback

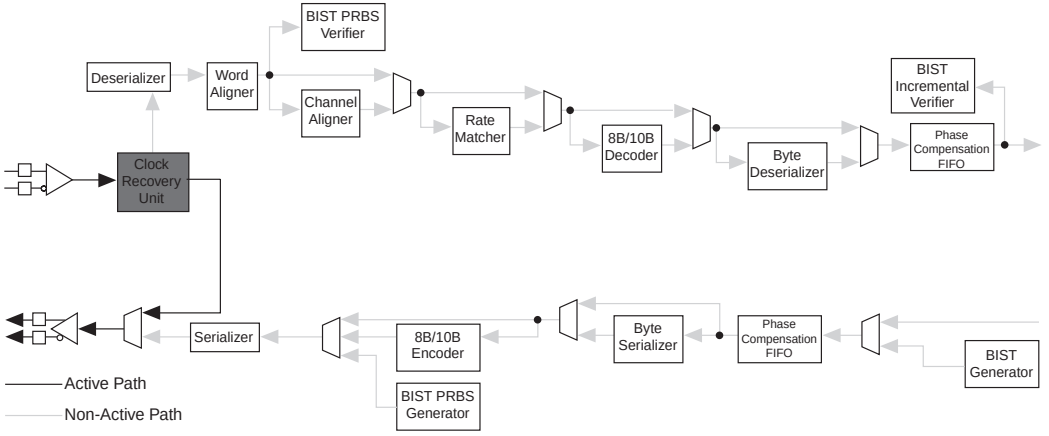
Figure 7–3 shows the data path for reverse serial loopback. Data comes in from the `rx_in` ports in the receiver. The data is then fed through the CDR block in serial form directly to the `tx_out` ports in the transmitter block.

Reverse serial loopback is enabled for all channels through the software or is dynamically enabled on a channel-by-channel basis using the `tx_srlpbk` port. When using reverse serial loopback, the V_{OD} must be 400mV.

The reverse serial loopback is enabled but the logic array is still seeing data.

Reverse serial loopback is often implemented when using a Bit Error Rate Tester (BERT).

Figure 7–3. Stratix GX Block in Reverse Serial Loopback Mode



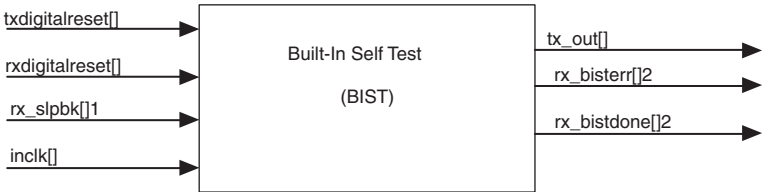
8. Stratix GX Built-In Self Test (BIST)

SGX52008-1.1

Introduction

Each Stratix® GX channel in the gigabit transceiver block contains embedded built-in self test (BIST) circuitry, which is available for quick device verification. The BIST circuitry consists of a data generator that resides in the transmitter channel and a verifier that resides in the receiver channel. Figure 8–1 shows a simplified block diagram of the BIST circuitry.

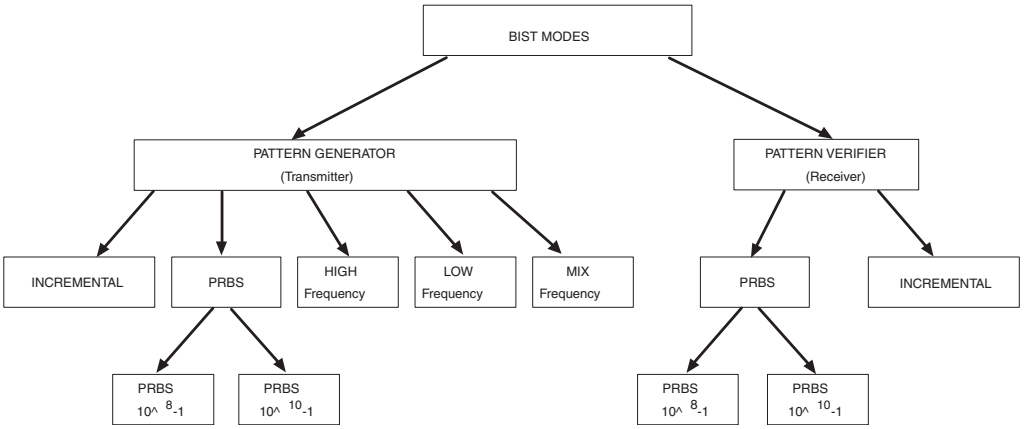
Figure 8–1. Image of Stratix GX Built-In Self Test



Notes to Figure 8–1:
(1) `rx_slpbk[]` is required in PRBS and incremental BIST modes.
(2) `rx_bisterr[]` and `rx_bistdone[]` are only available in PRBS and incremental BIST modes.

The BIST data generator is configured to generate pseudo-random binary sequence (PRBS), incremental, high-frequency, low-frequency, or mixed-frequency patterns. The BIST verifier supports only the PRBS and Incremental modes. The remaining BIST modes are intended for quick evaluations of the transmitters. The Quartus® II software simulation models do not support the PRBS patterns generated in the BIST circuit. Figure 8–2 shows the BIST modes.

Figure 8–2. Block Diagram of BIST Modes



Pattern Generator

The BIST data generator supports the following pattern generators:

- PRBS mode generator
- Incremental mode generator
- High-frequency mode generator
- Low-frequency mode generator
- Mix-frequency mode generator

PRBS Mode Generator

Pseudo-Random Bit Sequences (PRBS) are commonly used to verify the integrity and robustness of the data transmission paths. The PRBS generator is used in 8-, 16-, 10-, or 20-bit modes. In 8- or 16-bit data width modes, the PRBS generator generates 2^8-1 unique patterns. In 10- or 20-bit data modes, the PRBS generator yields 2^{10}-1 unique patterns. Table 8–1 lists the modes and their associated polynomials.

Table 8–1. BIST Generator PRBS Modes

Data Width	PRBS Mode	Polynomials
8-bit	2^8-1	$X^8 + X^7 + X^5 + X^3 + 1$
10-bit	2^{10}-1	$X^{10} + X^7 + 1$
16-bit	2^8-1	$X^8 + X^7 + X^5 + X^3 + 1$
20-bit	2^{10}-1	$X^{10} + X^7 + 1$

PRBS mode is enabled when the PRBS option is enabled in the Quartus II software. The 8b-10b encoder/decoder is bypassed automatically in this mode.

You can use PRBS generation to test the functionality of both the transmitter and receiver, to test if the BIST verifier is enabled, or to measure the quality of the transmission medium. The advantage of using a PRBS data stream is that the randomness yields an environment that stresses the transmission medium. In the data stream both random jitter and deterministic jitter are observed either by a time interval analyzer (TIA), a bit error rate tester, or an oscilloscope.

Incremental Mode Generator

In the incremental mode, the data generator sweeps through all the valid 8b/10b data and control characters. You can also enable the incremental BIST verifier to perform a quick verification of the 8B/10B encoder/decoder paths. Refer to "Pattern Verifier" on page 8–5 for more information.

Incremental mode is enabled when option 1 is selected under the **what self test mode do you want to use?** option in the Quartus II software. In this mode, the BIST generator sends out the data pattern in the following sequence: K28.5 (comma), K27.7 (Start of Frame, SOF), Data (00-FF incremental), K28.0, K28.1, K28.2, K28.3, K28.4, K28.6, K28.7, K23.7, K30.7, K29.7 (End of Frame, EOF) and repeat. The 8b/10b encoder must be enabled for proper operation.

Because the 8/b10b encoder is enabled, the data stream is DC balanced. 8b/10b encoding guarantees a run length less than 5 UI, which yields a less stressful pattern than the PRBS data. However, because the PRBS generator bypasses the 8b/10b paths, you can use the incremental BIST to test this path.

High-Frequency Mode Generator

In high-frequency mode, the BIST generator transmits a D21.5 ±(`8'b10110101`) character into the 8b/10b encoder to generate a `10'b1010101010` high-frequency character. This toggling data is the highest frequency that the data stream can transmit.

This pattern is DC balanced; the number of ones is equal to number of zeros. This fact is important when trying to perform a first-order random jitter measurement. You can measure this jitter using an oscilloscope with a histogram defined at the zero crossing point. This method is crude, but still yields a first-order estimated value, because the majority of the

deterministic data dependant components are masked out. However, for more accurate measurements, use a TIA or some type of jitter separation software to break down the random and deterministic components.

High-frequency mode is also useful when trying to characterize the high-frequency losses in the time domain. The delta amplitude difference between the high-frequency pattern and the low-frequency pattern can give you a first-order approximation of the high-frequency losses due to the skin effect and dielectric losses. This method is useful only for a first-order approximation; use extractions of RLGC values with 2D and 3D field solvers to determine more accurate loss coefficients.

High-frequency mode is enabled when option 2 is selected in the Quartus II software under **what self test mode do you want to use?** Enable the 8b/10b encoder to generate the high-frequency pattern. If it is disabled, an `8'b10110101` character is sent instead of the `10'b1010101010` character.

Low-Frequency Mode Generator

In low-frequency mode, the BIST generator transmits a K28.7 -/+ character (`8'b11111100`) into the 8b/10b encoder to generate a `10'b0011111000` or `10'b1100000111` low-frequency character. The low-frequency data transition toggles at one-tenth the data rate of the high-frequency pattern.

Like the high-frequency pattern, the low-frequency pattern is DC balanced with the number of ones equal to the number of zeros. This fact is important when trying to perform a first order random jitter measurement. You can measure this jitter using an oscilloscope with a histogram defined at the zero crossing point. This method is crude, but still yields a first-order estimated value, because the majority of the deterministic data-dependant components are masked out. However, for more accurate measurements, use a TIA or some type of jitter separation software to break down the random and deterministic components.

Because the data transitions in a slower frequency, the signal is less prone to high-frequency losses. As a result, the signal is able to rise to a higher amplitude than the high-frequency components. Therefore, the delta between the two measurements yields a first order approximation of the high-frequency losses in the time domain. Once again, this approach is useful only for a first-order approximation. Use extractions of RLGC values with 2D and 3D field solvers to determine more accurate loss coefficients.

Low-frequency mode is enabled when you select the SELF_ option 3 in the Quartus II software under **what self test mode do you want to use?** You must enable the 8b/10b encoder to generate the high-frequency pattern. If it is disabled, an `8'b11111100` character is sent instead of the `10'b0011111000` or `10'b1100000111` characters.

Mix-Frequency Mode Generator

In mix-frequency mode, the BIST generator transmits a K28.5 -/+ character (`8'b10111100`) character into the 8b/10b encoder to generate a `10'b0011111010` or `10'b1100000101` mixed-frequency character. The mixed frequency pattern contains both high-frequency and low-frequency components. This approach is useful for first-order approximation of the frequency response of the transmission medium. If captured with an oscilloscope, these frequency responses are approximated in time domain.

Mix-frequency mode is enabled when you select option 4 in the Quartus II software under **what self test mode do you want to use?** As in the high-frequency and low-frequency modes, you must enable the 8b/10b encoder in order to generate the mixed-frequency pattern.

Pattern Verifier

The BIST verifier supports the PRBS and incremental modes.

PRBS Mode Verifier

The PRBS verifier provides a quick check through the non-8b/10b path of the transceiver block. You must select the internal or external loopback mode to loop the generated data back into the verifier in the receiver. Select either a serial or parallel loopback to provide this path. A parallel loopback tests the digital portion of the transceiver while a serial loopback also tests the analog clock recovery unit (CRU) and the serializer and deserializer.

The PRBS verifier is active when the receiver channel is synchronized. The alignment pattern must be set to `16'b1000000011111111` for the 8- and 16-bit modes and to `10'b111111111` for the 10- and 20-bit modes. The data is synchronized automatically with a built in state machine, so the `rx_enacdet` signal is not required.

The verifier stops checking the patterns after receiving all the PRBS patterns (255 patterns for 8-bit mode and 1023 patterns for 10-bit mode). The `rx_bistdone` signal goes high, indicating that the verifier has completed. If the verifier detects an error before it is finished, `rx_bisterr` goes high and the value will be latched until it is reset. The `rxdigitalreset` signal must be used to re-start the PRBS verification.

Be sure you do not use the rx_apllreset signal because the re-training process of the CRU might cause false errors. A reference design is included in "Design Examples" on page 8–7.

Incremental Mode Verifier

In the incremental mode, the BIST generator transmits the data pattern in the following sequence: K28.5 (comma), K27.7 (SOF), Data (00-FF incremental), K28.0, K28.1, K28.2, K28.3, K28.4, K28.6, K28.7, K23.7, K30.7, K29.7 (EOF), and repeat.

The sync pattern on the receiver word aligner must be set to a K28.5 pattern (10'b0011111010) for proper synchronization between the generator and verifier. As in the PRBS verification mode, the synchronization is handled by a built-in state machine, so control of the rx_enacdet signal is not required.

The BIST verifier waits for the word aligner to synchronize. After synchronization, the BIST verifier checks for the following sequence: K27.7 (SOF), Data (00-FF incremental), K28.0, K28.1, K28.2, K28.3, K28.4, K28.6, K28.7, K23.7, K30.7, and K29.7 (EOF). If it does not see a K27.7 (SOF) within 31 patterns, the rx_errdetect and rx_bistdone signals go high, and the verifier stops. The verifier checks for this sequence twice before setting the rx_bistdone signal high. If any errors are detected before the verifier finishes, the rx_errdetect and rx_bistdone signals go high. Use the rxdigitalreset signal to restart the incremental verification. Do not use the rxanalogreset signal because the retraining process of the CRU might cause false errors. A reference design is included in "Design Examples" on page 8–7.

Table 8–2 shows which loopback modes are supported for each verification mode.

Table 8–2. Verification Modes

Verification Mode	Comma	Loopback Modes
2^8-1	16'b1000000011111111 (A1A2 mode)	Serial or parallel
2^{10}-1	10'b111111111 (10-bit mode)	Serial or parallel
Incremental	10'b0011111010 (10-bit mode)	Serial or parallel or post 8B/10B parallel

Design Examples

The purpose of these design examples are to show how to instantiate and operate the various BIST modes in Stratix GX devices. The following reference designs cover:

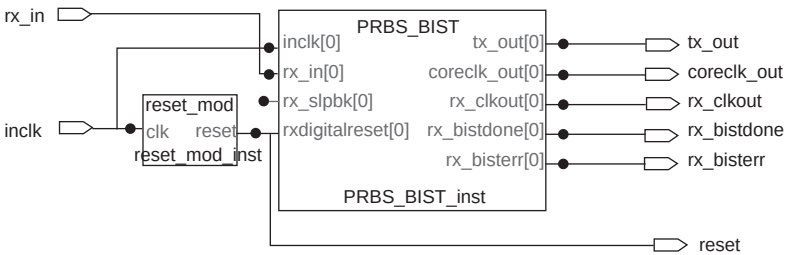
- PRBS BIST generator and verification design
- Incremental BIST generator and verification design
- High-frequency transmitter generation design
- Low-frequency transmitter generation design
- Mixed-frequency transmitter generation design

Design 1: PRBS BIST Generator & Verification Design

This design shows how to use the BIST in PRBS 2^{10}-1 mode. You can also apply this design principle to the 2^{8}-1 by changing the data-width mode, comma, and word-alignment mode as listed in Table 8–2 on page 8–6.

A useful circuit to include in the PRBS verifier is a self-timed reset controller. This controller prevents bounce conditions that might occur when an external switch is used. This design consists of a reset module (`reset.v`) that periodically toggles the `rxdigitalreset` signal of the `altgxb` instantiation (`PRBS_BIST.v`). Figure 8–3 shows a block diagram of this design.

Figure 8–3. Block Diagram of the PRBS BIST Design



Top-Level Design (PRBS.v)

```
module PRBS(
    inclk,
    rx_in,
    coreclk_out,
    tx_out,
    rx_bisterr,
    rx_bistdone,
    rx_clkout,
    reset
);
```

```
input  inclk;
input  rx_in;
output coreclk_out;
output tx_out;
output rx_bisterr;
output rx_bistdone;
output rx_clkout;
output reset;

wire   reset_wire;
wire   VCC;
assign reset = reset_wire;
assign VCC = 1;

//Altgxb Instantiation//////////////////////////

PRBS_BIST PRBS_BIST_inst(
.inclk(inclk),
.rx_in(rx_in),
.rx_slpbk(VCC),
.rxdigitalreset(reset_wire),
.coreclk_out(coreclk_out),
.rx_bistdone(rx_bistdone),
.rx_bisterr(rx_bisterr),
.rx_clkout(rx_clkout),
.tx_out(tx_out));

//Reset Module Instantiation/////////////////

reset_mod                  reset_mod_inst(
.clk(inclk),
.reset(reset_wire));
endmodule
```

Reset Module Design (reset_mod.v)

```
module reset_mod(clk, reset);
input clk;
output reset;
reg [19:0] counter;
reg reset;

always @ (posedge clk)
counter = counter +1;
always @ (counter) begin
   if ((counter >= 20'b11111111111111100000) &&
(counter <= 20'b11111111111111111111))
     reset = 1'b1;
else
   reset = 1'b0;
end
```

```
endmodule
```

altgxb Instantiation (PRBS_BIST.v)

```
module PRBS_BIST (
    inclk,
    rx_in,
    rx_slpbk,
    rxdigitalreset,
    tx_out,
    coreclk_out,
    rx_clkout,
    rx_bistdone,
    rx_bisterr);

    input[ 0:0]  inclk;
    input[ 0:0]  rx_in;
    input  [0:0]  rx_slpbk;
    input  [0:0]  rxdigitalreset;
    output [0:0]  tx_out;
    output [0:0]  coreclk_out;
    output [0:0]  rx_clkout;
    output [0:0]  rx_bistdone;
    output [0:0]  rx_bisterr;

    wire [0:0] sub_wire0;
    wire [0:0] sub_wire1;
    wire [0:0] sub_wire2;
    wire [0:0] sub_wire3;
    wire [0:0] sub_wire4;
    wire [0:0] tx_out = sub_wire0[0:0];
    wire [0:0] coreclk_out = sub_wire1[0:0];
    wire [0:0] rx_clkout = sub_wire2[0:0];
    wire [0:0] rx_bistdone = sub_wire3[0:0];
    wire [0:0] rx_bisterr = sub_wire4[0:0];

    altgxbaltgxb_component (

                                        .inclk (inclk),
              .rx_in (rx_in),
              .rx_slpbk (rx_slpbk),
              .rxdigitalreset (rxdigitalreset),
```

```
                    .tx_out (sub_wire0),
                    .coreclk_out (sub_wire1),
                    .rx_clkout (sub_wire2),
                    .rx_bistdone (sub_wire3),
                    .rx_bisterr (sub_wire4));

defparam
    altgxb_component.force_disparity_mode = "OFF",
    altgxb_component.channel_width = 20,
    altgxb_component.pll_inclock_period = 6400,
    altgxb_component.use_symbol_align = "ON",
    altgxb_component.rx_ppm_setting = 1000,
    altgxb_component.pll_bandwidth_type = "LOW",
    altgxb_component.dwidth_factor = 2,
    altgxb_component.number_of_channels = 1,
    altgxb_component.vod_ctrl_setting = 1000,
    altgxb_component.align_pattern_length = 10,
    altgxb_component.use_self_test_mode = "ON",
    altgxb_component.lpm_type = "altgxb",
    altgxb_component.use_fifo_mode = "ON",
    altgxb_component.use_vod_ctrl_signal = "OFF",
    altgxb_component.equalizer_ctrl_setting = 0,
    altgxb_component.use_auto_bit_slip = "ON",
    altgxb_component.use_rate_match_fifo = "OFF",
    altgxb_component.signal_threshold_select = 80,
    altgxb_component.self_test_mode = 0,
    altgxb_component.use_double_data_mode = "ON",
    altgxb_component.use_preemphasis_ctrl_signal =
"OFF",
    altgxb_component.protocol = "CUSTOM",
    altgxb_component.clk_out_mode_reference = "ON",
    altgxb_component.rx_bandwidth_type = "LOW",
    altgxb_component.disparity_mode = "ON",
    altgxb_component.preemphasis_ctrl_setting = 0,
    altgxb_component.loopback_mode = "SLB",
    altgxb_component.use_channel_align = "OFF",
    altgxb_component.intended_device_family =
"Stratix GX",
    altgxb_component.use_equalizer_ctrl_signal =
"OFF",
    altgxb_component.rx_enable_dc_coupling = "OFF",
    altgxb_component.run_length_enable = "OFF",
    altgxb_component.pll_use_dc_coupling = "OFF",
    altgxb_component.operation_mode = "DUPLEX",
    altgxb_component.use_8b_10b_mode = "OFF",
    altgxb_component.use_rx_clkout = "ON",
    altgxb_component.data_rate_remainder = 0,
```

```
    altgxb_component.data_rate = 3125,
    altgxb_component.align_pattern = "P1111111111",
    altgxb_component.use_rx_cruclk = "OFF",

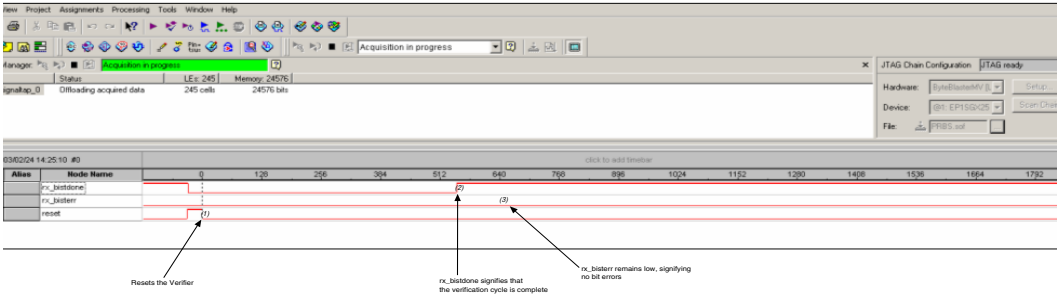
    altgxb_component.number_of_quads = 1;

endmodule
```

Results

A quick method for verifying whether the BIST verification passes or fails is to use the SignalTap® II logic analyzer in the Quartus® II software. Refer to *Application Note 280: Design Verification Using the SignalTap II Logic Analyzer* for more information on using the SignalTap II logic analyzer. The SignalTap II logic analyzer trigger is set to the falling edge of the reset output signal. Figure 8–4 is a screen shot of the SignalTap II logic analyzer results for this PRBS BIST test.

Figure 8–4. SignalTap II Logic Analyzer Results for PRBS BIST Test Design

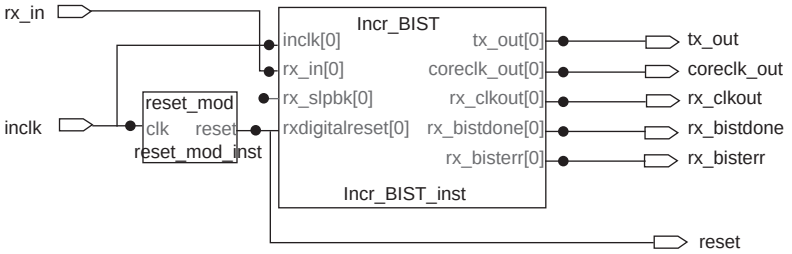


Design 2: Incremental BIST Generator & Verification Design

This design is similar to the PRBS BIST generator and verification design, except the `altgxb` megafunction is configured to the incremental BIST mode. Refer to the design for information on the ports and parameters required for `altgxb` in this mode.

As in the PRBS design, a useful circuit to include in the PRBS verifier is a self-timed reset controller. This controller prevents bounce conditions that might occur when an external switch is used. This design consists of a reset module (`reset.v`) that periodically toggles the `rxdigitalreset` signal of the `altgxb` instantiation (`Incremental_BIST.v`). Figure 8–5 shows a block diagram of this design.

Figure 8–5. Block Diagram of the Incremental BIST Design



Top-Level Design (Incremental)

```
module incremental(
    inclk,
    rx_in,

    coreclk_out,
    tx_out,
    rx_bisterr,
    rx_bistdone,
    rx_clkout,
    reset

);

input  inclk;
input  rx_in;
outputcoreclk_out;
output tx_out;
outputrx_bisterr;
output rx_bistdone;
output rx_clkout;
output reset;

wire   reset_wire;
wire   VCC;

assign reset = reset_wire;
assign VCC = 1;
Incr_BIST                       Incr_BIST_inst(
.inclk(inclk),
.rx_in(rx_in),
.rx_slpbk(VCC),
.rxdigitalreset(reset_wire),
```

```
.coreclk_out(coreclk_out),
.rx_bistdone(rx_bistdone),
.rx_bisterr(rx_bisterr),
.rx_clkout(rx_clkout),
.tx_out(tx_out));

reset_mod reset_mod_inst(
.clk(inclk),
.reset(reset_wire));

endmodule
```

Reset Module Design (reset_mod.v)

```
module reset_mod(clk, reset);
input clk;
output reset;

reg [19:0] counter;

reg reset;

always @ (posedge clk)
counter = counter +1;

always @ (counter) begin
   if ((counter >= 20'b11111111111111100000) &&
(counter <= 20'b11111111111111111111))
       reset = 1'b1;
   else
       reset = 1'b0;
end

endmodule
```

altgxb Instantiation (Incr_BIST.v)

```
module Incr_BIST (
   inclk,
   rx_in,
   rx_slpbk,
   rxdigitalreset,
   tx_out,
   coreclk_out,
   rx_clkout,
   rx_bistdone,
   rx_bisterr);
```

```
    input                                [0:0]  inclk;
    input  [0:0]  rx_in;
    input  [0:0]  rx_slpbk;
    input  [0:0]  rxdigitalreset;
    output [0:0]  tx_out;
    output [0:0]  coreclk_out;
    output [0:0]  rx_clkout;
    output [0:0]  rx_bistdone;
    output [0:0]  rx_bisterr;

    wire [0:0] sub_wire0;
    wire [0:0] sub_wire1;
    wire [0:0] sub_wire2;

    wire [0:0] sub_wire3;
    wire [0:0] sub_wire4;
    wire [0:0] tx_out = sub_wire0[0:0];
    wire [0:0] coreclk_out = sub_wire1[0:0];
    wire [0:0] rx_clkout = sub_wire2[0:0];
    wire [0:0] rx_bistdone = sub_wire3[0:0];
    wire   [0:0] rx_bisterr = sub_wire4[0:0];

    altgxb                               altgxb_component
(
                                .inclk (inclk),
              rx_in (rx_in),
              .rx_slpbk (rx_slpbk),
              .rxdigitalreset (rxdigitalreset),
              .tx_out (sub_wire0),
              .coreclk_out (sub_wire1),
              .rx_clkout (sub_wire2),
              .rx_bistdone (sub_wire3),
              .rx_bisterr (sub_wire4));

    defparam

        altgxb_component.force_disparity_mode = "OFF",
        altgxb_component.channel_width = 16,
        altgxb_component.pll_inclock_period = 6250,
        altgxb_component.use_symbol_align = "ON",
        altgxb_component.rx_ppm_setting = 1000,
        altgxb_component.pll_bandwidth_type = "LOW",
        altgxb_component.dwidth_factor = 2,
        altgxb_component.number_of_channels = 1,
        altgxb_component.vod_ctrl_setting = 1000,
        altgxb_component.align_pattern_length = 10,
        altgxb_component.use_self_test_mode = "ON",
```

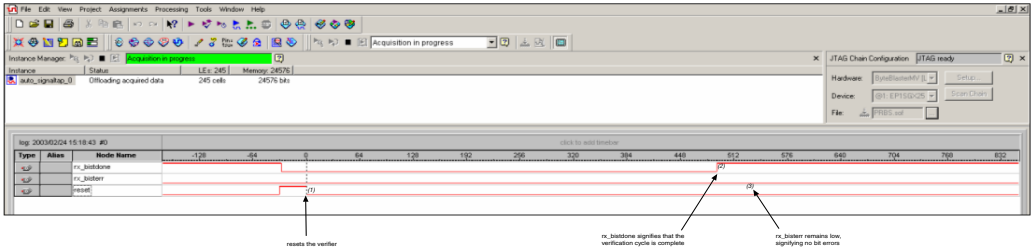
```
        altgxb_component.lpm_type = "altgxb",
        altgxb_component.use_fifo_mode = "ON",
        altgxb_component.use_vod_ctrl_signal = "OFF",
        altgxb_component.equalizer_ctrl_setting = 0,
        altgxb_component.use_auto_bit_slip = "ON",
        altgxb_component.use_rate_match_fifo = "OFF",
        altgxb_component.signal_threshold_select = 80,
        altgxb_component.self_test_mode = 1,
        altgxb_component.use_double_data_mode = "ON",
        altgxb_component.use_preemphasis_ctrl_signal =
"OFF",
        altgxb_component.protocol = "CUSTOM",
        altgxb_component.clk_out_mode_reference = "ON",
        altgxb_component.rx_bandwidth_type = "LOW",
        altgxb_component.disparity_mode = "ON",
        altgxb_component.preemphasis_ctrl_setting = 0,
        altgxb_component.loopback_mode = "SLB",
        altgxb_component.use_channel_align = "OFF",
        altgxb_component.intended_device_family =
"Stratix GX",
        altgxb_component.use_equalizer_ctrl_signal =
"OFF",
        altgxb_component.rx_enable_dc_coupling = "OFF",
        altgxb_component.run_length_enable = "OFF",
        altgxb_component.pll_use_dc_coupling = "OFF",
        altgxb_component.operation_mode = "DUPLEX",
        altgxb_component.use_8b_10b_mode = "ON",
        altgxb_component.use_rx_clkout = "ON",
        altgxb_component.data_rate_remainder = 0,
        altgxb_component.data_rate = 2560,
        altgxb_component.align_pattern = "P0011111010",
        altgxb_component.use_rx_cruclk = "OFF",
        altgxb_component.number_of_quads = 1;

endmodule
```

Results

A quick method for verifying whether the BIST verification passes or fails is to use the SignalTap II embedded logic analyzer in the Quartus II software. Refer to *Application Note 280: Design Verification Using the SignalTap II Logic Analyzer* for more information. The SignalTap II trigger is set to the falling edge of the reset output signal. Figure 8–6 is a screen shot of the SignalTap II results for the incremental BIST results.

Figure 8–6. SignalTap II Results for PRBS BIST Test Design (Resets the Verifier)



Design 3: High-Frequency Transmitter Generator Design

This design shows how to instantiate the altgxb megafunction in the high-frequency BIST mode. Because this design consists only of a single transmitter design, only the altgxb instantiation is shown. The top level simply consists of calling the megafunction instance.

altgxb Instantiation (High_Freq_BIST.v)

```
module high_freq_BIST (
       inclk,
       tx_out,
       coreclk_out);

       input  [   0:0]  inclk;
       output [0:0]  tx_out;
       output [0:0]  coreclk_out;

       wire [0:0] sub_wire0;
       wire [0:0] sub_wire1;
       wire [0:0] tx_out = sub_wire0[0:0];
       wire [0:0] coreclk_out = sub_wire1[0:0];

       altgxb                          altgxb_component
(

                                       .inclk (inclk),
                  .tx_out (sub_wire0),
                      .coreclk_out (sub_wire1));

       defparam

altgxb_component.force_disparity_mode = "OFF",
          altgxb_component.channel_width = 16,
```

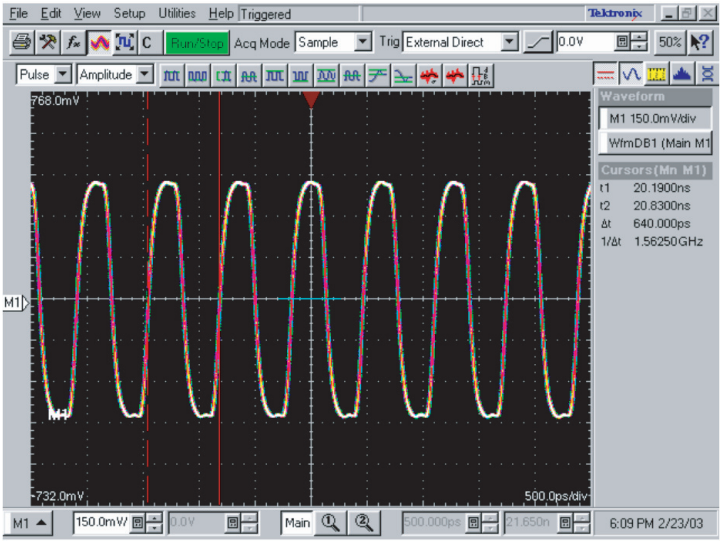
```
          altgxb_component.pll_inclock_period = 6250,
          altgxb_component.pll_bandwidth_type = "LOW"
,               altgxb_component.dwidth_factor = 2,
          altgxb_component.number_of_channels = 1
,                    altgxb_component.vod_ctrl_setting
= 1000,
          altgxb_component.use_self_test_mode = "ON",
          altgxb_component.lpm_type = "altgxb",
          altgxb_component.use_fifo_mode = "ON",
          altgxb_component.use_vod_ctrl_signal =
"OFF",
          altgxb_component.self_test_mode = 2,
          altgxb_component.use_double_data_mode =
"ON",
       altgxb_component.use_preemphasis_ctrl_signal =
"OFF",
          altgxb_component.protocol = "CUSTOM",
          altgxb_component.clk_out_mode_reference =
"ON",
          altgxb_component.preemphasis_ctrl_setting =
0,
          altgxb_component.use_channel_align = "OFF",
          altgxb_component.intended_device_family =
"Stratix GX",
          altgxb_component.pll_use_dc_coupling =
"OFF",
          altgxb_component.operation_mode = "TX",
          altgxb_component.use_8b_10b_mode = "ON",
          altgxb_component.use_rx_clkout = "OFF",
          altgxb_component.data_rate_remainder = 0,
          altgxb_component.data_rate = 2560,
          altgxb_component.use_rx_cruclk = "OFF",
          altgxb_component.number_of_quads = 1;

endmodule
```

Results

Figure 8–7 shows a screen shot of the high-frequency BIST mode. The signal was captured using a sampling oscilloscope.

Figure 8–7. High-Frequency BIST Measured on tx_out[]



Design 4: Low-Frequency Transmitter Generator Design

This design shows how to instantiate the `altgxb` megafunction in the low-frequency BIST mode. Because this design consists only of a single transmitter design, only the `altgxb` instantiation is shown. The top level simply consists of calling the megafunction instance.

altgxb Instantiation (low_freq_BIST.v)

```
module low_freq_BIST (
    inclk,
    tx_out,
    coreclk_out);

    input  [0:0]  inclk;

    output           [0:0]  tx_out;
    output [0:0]  coreclk_out;

    wire [0:0] sub_wire0;
    wire [0:0] sub_wire1;
    wire [0:0] tx_out = sub_wire0[0:0];
    wire [0:0] coreclk_out = sub_wire1[0:0];
```

```
    altgxb altgxb_component (
              inclk (inclk),
              .tx_out (sub_wire0),
              .coreclk_out (sub_wire1));

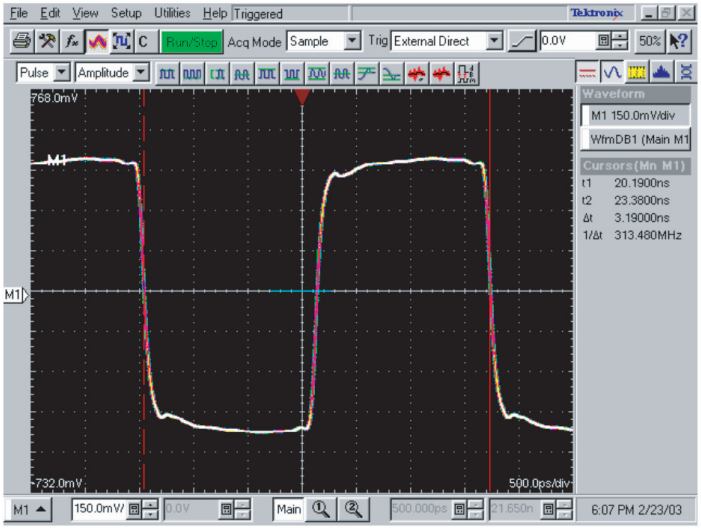
    defparam
       altgxb_component.force_disparity_mode = "OFF",
       altgxb_component.channel_width = 16,
       altgxb_component.pll_inclock_period = 6250,
       altgxb_component.pll_bandwidth_type = "LOW",
       altgxb_component.dwidth_factor = 2,
       altgxb_component.number_of_channels = 1,
       altgxb_component.vod_ctrl_setting = 1000,
       altgxb_component.use_self_test_mode = "ON",
       altgxb_component.lpm_type = "altgxb",
       altgxb_component.use_fifo_mode = "ON",
       altgxb_component.use_vod_ctrl_signal = "OFF",
       altgxb_component.self_test_mode = 3,
       altgxb_component.use_double_data_mode = "ON",
       altgxb_component.use_preemphasis_ctrl_signal =
"OFF",
       altgxb_component.protocol = "CUSTOM",
       altgxb_component.clk_out_mode_reference = "ON",
       altgxb_component.preemphasis_ctrl_setting = 0,
       altgxb_component.use_channel_align = "OFF",
       altgxb_component.intended_device_family =
"Stratix GX",
       altgxb_component.pll_use_dc_coupling = "OFF",
       altgxb_component.operation_mode = "TX",
       altgxb_component.use_8b_10b_mode = "ON",
       altgxb_component.use_rx_clkout = "OFF",
       altgxb_component.data_rate_remainder = 0,
       altgxb_component.data_rate = 2560,
       altgxb_component.use_rx_cruclk = "OFF",
       altgxb_component.number_of_quads = 1;

endmodule
```

Results

The low-frequency BIST mode is shown in Figure 8–8. The signal was captured using a sampling oscilloscope.

Figure 8–8. Low-Frequency BIST Measured on tx_out[]



Design 5: Mix-Frequency Transmitter Generator Design

The mix-frequency transmitter generator design shows how to instantiate the `altgxb` megafunction in the mix-frequency BIST mode. Because this design consists only of a single transmitter design, only the `altgxb` instantiation is shown. The top level simply consists of calling the megafunction instance.

altgxb Instantiation (mix_freq_BIST.v)

```
module mix_freq_BIST (
    inclk,
    tx_out,
    coreclk_out);

    input                           [0:0]   inclk;
    output[0:0]   tx_out;
    output[0:0]   coreclk_out;

    wire [0:0] sub_wire0;
    wire [0:0] sub_wire1;
    wire [0:0] tx_out = sub_wire0[0:0];
    wire [0:0] coreclk_out = sub_wire1[0:0];
```

```
    altgxb altgxb_component (
              inclk (inclk),
              .tx_out (sub_wire0),
              .coreclk_out (sub_wire1));

    defparam
        altgxb_component.force_disparity_mode = "OFF",
        altgxb_component.channel_width = 16,
        altgxb_component.pll_inclock_period = 6250,
        altgxb_component.pll_bandwidth_type = "LOW",
        altgxb_component.dwidth_factor = 2,
        altgxb_component.number_of_channels = 1,
        altgxb_component.vod_ctrl_setting = 1000,
        altgxb_component.use_self_test_mode = "ON",
        altgxb_component.lpm_type = "altgxb",
        altgxb_component.use_fifo_mode = "ON",

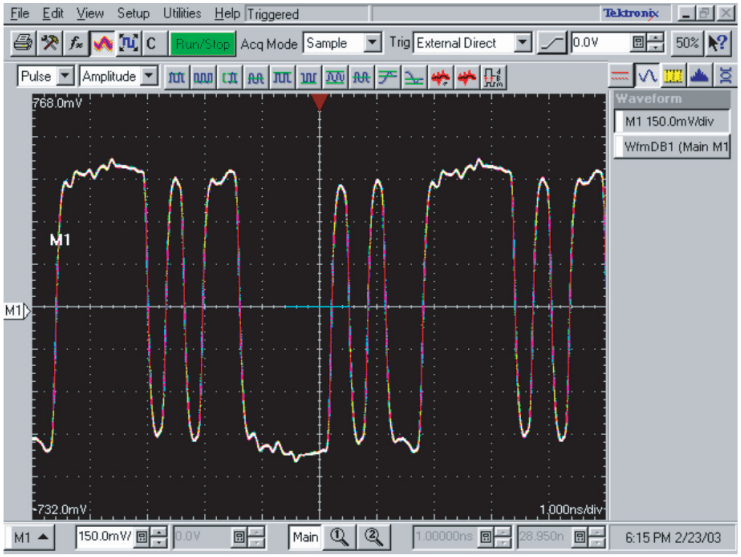
        altgxb_component.use_vod_ctrl_signal = "OFF",
        altgxb_component.self_test_mode = 4,
        altgxb_component.use_double_data_mode = "ON",
        altgxb_component.use_preemphasis_ctrl_signal =
"OFF",
        altgxb_component.protocol = "CUSTOM",
        altgxb_component.clk_out_mode_reference = "ON",
        altgxb_component.preemphasis_ctrl_setting = 0,
        altgxb_component.use_channel_align = "OFF",
        altgxb_component.intended_device_family =
"Stratix GX",
        altgxb_component.pll_use_dc_coupling = "OFF",
        altgxb_component.operation_mode = "TX",
        altgxb_component.use_8b_10b_mode = "ON",
        altgxb_component.use_rx_clkout = "OFF",
        altgxb_component.data_rate_remainder = 0,
        altgxb_component.data_rate = 2560,
        altgxb_component.use_rx_cruclk = "OFF",
        altgxb_component.number_of_quads = 1;

endmodule
```

Results

Figure 8–9 shows a screen shot of the mix-frequency BIST mode. The signal was captured using a sampling oscilloscope.

Figure 8–9. Mix-Frequency BIST Measured on tx_out[]



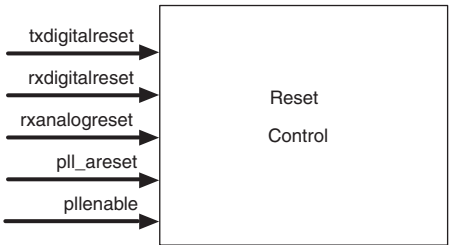
9. Reset Control & Power Down

SGX52009-1.0

Introduction

Stratix® GX transceivers offer multiple reset signals to control separate ports of the transceiver channels and transceiver blocks, as shown in Figure 9–1. The Quartus® II software sets each unused channel to a power-down mode to reduce power consumption.

Figure 9–1. Reset Control Diagram



Power On Reset (POR)

At power on, the Stratix GX transceiver uses built-in circuits that handle the reset of the digital and analog circuits. After power on reset (POR), the Stratix GX block is guaranteed to be in a known state.

USER Reset & Enable Signals

Each transceiver block and channel in the Stratix GX transceiver block has individual reset signals to reset the digital and analog portions of the channel. The `txdigitalreset`, `rxdigitalreset`, and `rxanalogreset` signals affect the channels individually. The `pll_areset` and `pllenable` signals affect the entire transceiver block.

The `pll_areset` signal is a power-down signal and powers down the entire transceiver block. The analog circuitry is powered down when the `pll_areset` signal goes high. Although there is no specific requirement on the duration of the `pll_areset` signal, Altera® lab experiments have shown that 1 ms is a safe value. If you use the `pll_areset` signal to power down the analog circuitry, Altera recommends that you use the `pll_locked` and `rx_freqlocked` signals from the transceiver block to implement your reset logic.

The rxanalogreset signal is a power-down signal and only powers down the receiver. The analog circuitry is powered down when the rxanalogreset signal goes high. Although there is no specific requirement on the duration of the rxanalogreset signal, Altera lab experiments have shown that 1 ms is a safe value. If you use the rxanalogreset signal to power down the analog circuitry, Altera recommends that you use the rx_freqlocked signal from the receiver block to implement your reset logic.

The rxdigitalreset signal resets the digital logic in the receiver section of the transceiver block. This signal is synchronized within the transceiver block. The minimum duration required on the rxdigitalreset signal is four parallel clock cycles.

The txdigitalreset signal resets the digital logic in the transmitter section of the transceiver block. This signal is synchronized within the transceiver block. The minimum duration required on the txdigitalreset signal is four parallel clock cycles.



If you use REFCLKB pins in your design, refer to the *REFCLKB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* for analog reset (pll_areset, rxanalogreset, pll_enable) refclkb usage constraints.

You do not have to use all of the reset and enable signals. If the reset and power-down signals are not used, they default to their inactive levels.

Under normal operating conditions, you do not have to power down the transmitter PLL. The PLLs should only be powered down as the last option because there is a significant delay to recover from a power down state and return to normal operation.

If the read and write pointers in the phase compensation FIFO buffers point to the same location, the buffer outputs incorrect data. This can occur during system initialization. If this occurs, use the digital reset signals (rxdigitalreset and txdigitalreset) to reset the digital logic of that channel.

Table 9–1 shows the reset and enable signals that are required for the transceiver blocks.

In 16-bit or 20-bit mode, asserting `rxdigitalreset` causes the recovered clock or the slow clock to reset. The slow clock is divided down by the deserialization factor from `rx_clkout`. Altera recommends synchronizing `rxdigitalreset` to the FPGA or the logic array clock.

Table 9–1. Reset Signal Map to Stratix GX Blocks

	Transmitter Phase Compensation FIFO Module/ Byte Serializer	Transmitter 8B/10B Encoder	Transmitter Serializer	Transmitter Analog Circuits	Transmitter PLL	Transmitter XAUI State Machine	Transmitter Analog Circuits	BIST Generators	Receiver Deserializer	Receiver Word Aligner	Receiver Deskew FIFO Module	Receiver Rate Matcher	Receiver 8B/10B Decoder	Receiver Phase Compensation FIFO Module/ Byte Deserializer	Receiver PLL / CRU	Receiver XAUI State Machine	BIST Verifiers	Receiver Analog Circuits
`rxdigitalreset`										✓	✓	✓	✓	✓		✓	✓	
`rxanalogreset`									✓						✓			✓
`txdigitalreset`	✓	✓				✓		✓										
`pll_areset`	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
`pllenable`	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Link initialization must be performed after any reset condition. You must determine when the data is valid after reset (for example, by using the `rx_syncstatus` signal in XAUI mode).

Recommended Resets

The following reset recommendations help guard against potential initialization issues during the training of the transmitter PLL and receiver PLLs. The counters that you specify in the recommendations filter out any high frequency effects to ensure that the lock signals are stable before releasing the subsequent reset signals. This action adds to the robustness of the reset sequence.

Receiver & Transmitter Reset

The configurations and design examples in this section describe how to implement a reset sequence for both the receiver and transmitter channels. The designs in this section demonstrate the reset sequence only. Each design example lists the constraints specific to the example to help you understand the design parameters and limitations. You may want to add additional escape states and other system-specific features in your design. If your design requirements and GXB configurations are different (for example, usage for multiple transceivers) from the design example, you can make necessary changes using the flow chart and waveform figures in each section as guidelines.

Train Receive CRU With Transmit PLL Output Clock

This section contains RTL examples that show some scenarios where a transceiver is programmed to duplex and the transmitter PLL output clock trains the receiver clock recovery unit (CRU).

Figure 9–2 shows the possible options on clocking the transmit and receive parallel interface for a selected data path width.

Figure 9–2. Receiver & Transmitter Clock Options

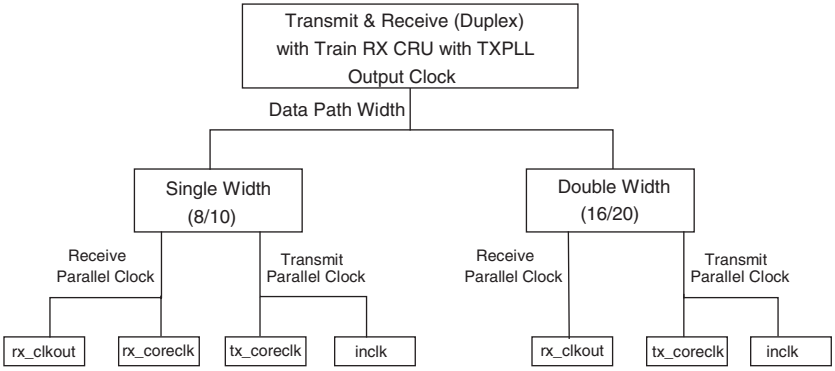
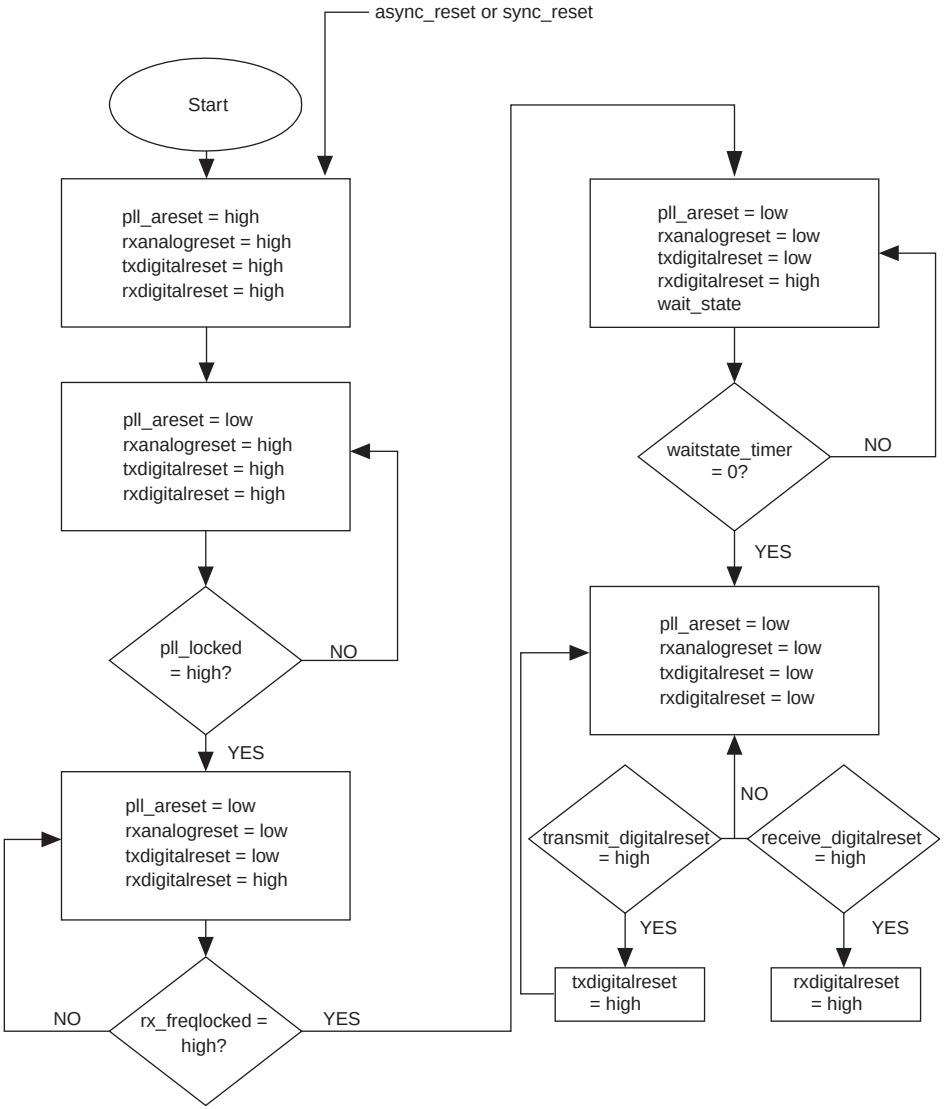


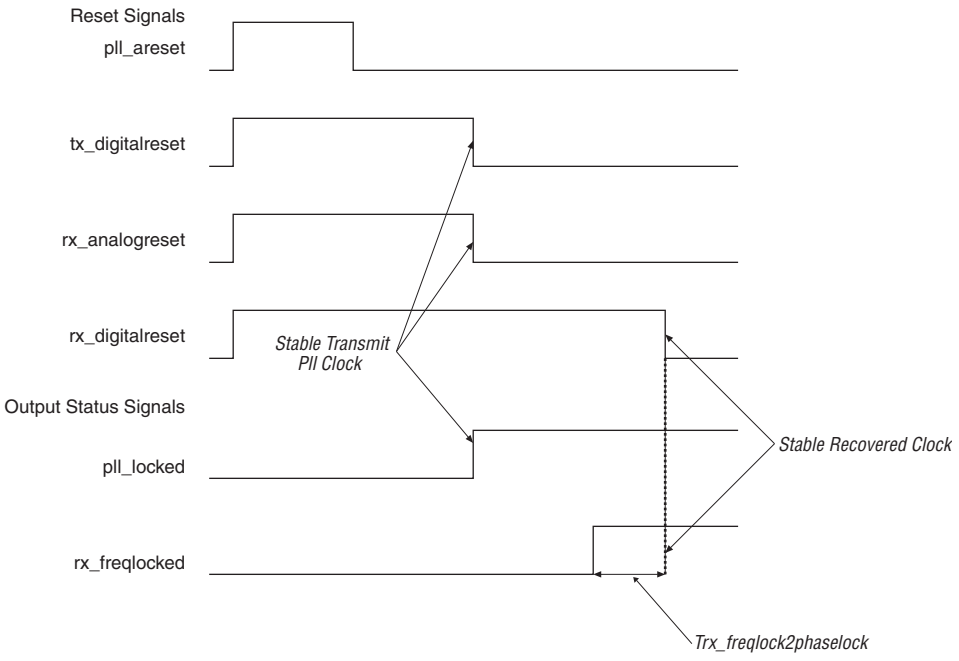
Figure 9–3 shows a situation in which you must reset both the transmitter and receiver channels.

Figure 9–3. Receiver & Transmitter Reset Sequence



The waveform in Figure 9–4 shows the functionality of the receiver and transmitter reset sequence shown in Figure 9–3. The `pll_areset` signal resets the entire transceiver block, including both the analog and digital portions of the transmitter and receiver (see Table 9–1). After the `pll_areset` signal goes low, the controller waits until the transmitter PLL is stable (pll_locked = `1'b1`) before sending the `tx_digitalreset` and `rx_analogreset` low. This ensures that the output of the transmitter PLL is stable before releasing any of the logic that it feeds. The transmitter PLL clock in this case also trains the receiver PLL. After the CRU has transitioned to locking to data from locking to the reference clock, the `rx_freqlocked` signal goes high, which allows the CRU to transition into the wait state where a timer is loaded a certain amount of time. See the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1* for the amount of time loaded into the timer. When the timer counts down, `rx_clkout` is stable. The reset controller then sends `rx_digital` low, completing the reset sequence. You will be able to monitor the BER (for example, a synchronization state machine based on the Stratix GX transceiver data) to determine whether the system is initialized and working properly.

Figure 9–4. Receiver & Transmitter Reset Sequence Waveforms



Design Example 1

This design example shows inclk as the input reference clock and the transmit parallel clock and rx_coreclk as the receive parallel clock. The design example has the following constraints:

- If your design requirements are different from the examples, use the flow charts and waveforms for each configuration as design guidelines.
- The design example requires a reset controller that generates a sync_reset (synchronous reset) for the entire system.
- The design example has an async_reset (a power down in GXB terms) and digital resets for transmit and receive. All user input digital resets must be at least four cycles long.
- This design example does not cover all the digital reset scenarios in a system that resets the digital logic of the GXB.
- In this example, whenever the rx_freqlocked signal toggles the rxdigitalreset, the receiver's digital circuit is reset. However, you can make changes to the design to avoid this if, for example, you want to debug your design without the core being reset.
- If you plan to use REFCLKB pins in your design, see the *REFCLKB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* for information about the effects of analog resets (pll_arest, rx_analogreset).

```
/*
 Copyright (c) Altera Corporation, 2004.
 This file may contain proprietary and confidential information
of  Altera Corporation
=================

We have made every effort to ensure that this design example works
correctly. If you have a question or problem that is not answered
by the information then please contact Altera Support.

*****************************************************************
Reset Sequence for the ALTGXB. The configuration of GXB for which
the following reset sequence is valid is:
    Transmit and Receive : Both used
    Datapath    : Single Width(8/10 bits)
    receive parallel clock: rx_coreclk
    Functional Mode :'Any'
    RX PLL CRU      : Train RX PLL CRU with TX PLL ouput clock
*****************************************************************/


`timescale 1ns/10ps

module reset_seq_tx_train_rx_rx_coreclk (
                            rx_coreclk,
                            inclk,

                             sync_reset,
                             async_reset,
```

```
                                    transmit_digitalreset,
                                    receive_digitalreset,
                                    pll_locked,
                                    rx_freqlocked,

                                   pll_areset,
                                   txdigitalreset,
                                    rxanalogreset,
                                    rxdigitalreset

                                   );


input inclk; //GXB input reference clock
input rx_coreclk;//Receive recovered clock

input sync_reset;      //Input: synchronous reset from the system
input async_reset;   //Input: async reset from system
input transmit_digitalreset; //Input: Reset only the transmit
digital section
input receive_digitalreset; //Input : Reset the receiver section

input rx_freqlocked;   //rx_freqlocked signal from receive;
Transition from 'lock to reference clock mode' to 'lock to data
mode'
input pll_locked;      // Transmit PLL of GXB locked


output rxdigitalreset;//GXB Receive digital reset
output rxanalogreset;//Receive power down signal
output txdigitalreset; //GXB transmit digital reset
output pll_areset;//GXB power down signal

reg rxdigitalreset;
wire rxanalogreset;
reg txdigitalreset;
reg pll_areset;
reg [2:0] state;
reg rxdigitalreset_inclk;
reg rxanalogreset_inclk;

reg rxdigitalreset_rx_coreclk_Q;
reg rxanalogreset_rx_coreclk_Q;

parameter IDLE  = 3'b000;
parameter STROBE_TXPLL_LOCKED = 3'b001;
parameter STABLE_TX_PLL = 3'b010;
parameter WAIT_STATE = 3'b011;


//Parameter value of T (2ms)based on the fastest clock (or 3.1875
Gbps)
parameter WAITSTATE_TIMER_VALUE = 1000000;

reg [19:0]waitstate_timer;  //timer - for actual value, refer
stratix data sheet

assign rxanalogreset = rxanalogreset_inclk;
```

```
always @ (posedge inclk or posedge async_reset) begin
if (async_reset)
    begin
        rxdigitalreset_inclk <= 1'b1;
        rxanalogreset_inclk <= 1'b1;
        txdigitalreset <= 1'b1;
        pll_areset <= 1'b1;
        waitstate_timer <= WAITSTATE_TIMER_VALUE;
        state       <= STROBE_TXPLL_LOCKED;
    end
else
  case (state)
    IDLE:
                if (sync_reset)      //Synchronous Reset can be
asserted in IDLE state (After reset seq has finished)
                    begin
                        rxdigitalreset_inclk <= 1'b1;
                        rxanalogreset_inclk <= 1'b1;
                        txdigitalreset <= 1'b1;
                        pll_areset <= 1'b1;
                        waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                        state       <= STROBE_TXPLL_LOCKED;

                    end
                else
                    begin
                        rxdigitalreset_inclk <= 1'b0;
                        rxanalogreset_inclk <= 1'b0;
                        pll_areset <= 1'b0;
                        state       <= IDLE;
                        if(transmit_digitalreset)
                            txdigitalreset <= 1'b1;
                        else
                            txdigitalreset <= 1'b0;
                    end

    STROBE_TXPLL_LOCKED: if (sync_reset)    //Synchronous Reset
can be asserted in IDLE state (After reset seq has finished)
                            begin
                                rxdigitalreset_inclk <= 1'b1;
                                rxanalogreset_inclk <= 1'b1;
                                txdigitalreset <= 1'b1;
                                pll_areset <= 1'b1;
                                state <= STROBE_TXPLL_LOCKED;
                            end
    //Wait untill the TXPLL is locked to inclk and TX PLL has a
stable output clock which is also fed to RX CRU
                        else if (pll_locked)
                            begin
                                 state  <= STABLE_TX_PLL;
                                 rxdigitalreset_inclk<= 1'b1;
                                 rxanalogreset_inclk <= 1'b0;
                                 txdigitalreset<= 1'b0;
                                 pll_areset <= 1'b0;
                            end
                        else
```

```
                                begin
                                    state <= STROBE_TXPLL_LOCKED;
                                    rxdigitalreset_inclk <= 1'b1;
                                    rxanalogreset_inclk <= 1'b1;
                                    txdigitalreset <= 1'b1;
                                    pll_areset <= 1'b0;
                                end
    STABLE_TX_PLL:  if (sync_reset)      //Synchronous Reset can
be asserted in IDLE state (After reset seq has finished)
                                begin
                                    rxdigitalreset_inclk <= 1'b1;
                                    rxanalogreset_inclk <= 1'b1;
                                    txdigitalreset <= 1'b1;
                                    pll_areset <= 1'b1;
                                    state <= STROBE_TXPLL_LOCKED;
                                end
                            else if (rx_freqlocked)
                                begin
                                    state <= WAIT_STATE;
                                    waitstate_timer <= waitstate_timer -
1'b1 ;
                                     rxdigitalreset_inclk<= 1'b1;
                                     rxanalogreset_inclk <= 1'b0;
                                     txdigitalreset<= 1'b0;
                                     pll_areset <= 1'b0;
                                end
                            else
                                begin
                                    state       <= STABLE_TX_PLL;
                                     rxdigitalreset_inclk<= 1'b1;
                                     rxanalogreset_inclk <= 1'b0;
                                     txdigitalreset<= 1'b0;
                                     pll_areset <= 1'b0;
                                end
    WAIT_STATE: if (sync_reset)      //Synchronous Reset can be
asserted in IDLE state (After reset seq has finished)
                                begin
                                    rxdigitalreset_inclk <= 1'b1;
                                    rxanalogreset_inclk <= 1'b1;
                                    txdigitalreset <= 1'b1;
                                    pll_areset <= 1'b1;
                                    state <= STROBE_TXPLL_LOCKED;
                                end
                        else if(rx_freqlocked)   //Condition to have
rx_freqlocked signal a stable high and should not bounce around
                                begin
                                //Decrement a Timer of 2ms (Refer
Stratix GX Datasheet for accurate value)after rx_freqlocked is
asserted
                                //This time is given to ensure the
recovered clock to be stable (No freq variations) and is locked
to incomming data
                                if(waitstate_timer == 0)
                                    begin
                                        state <= IDLE;
                                        rxdigitalreset_inclk<= 1'b0;
                                         rxanalogreset_inclk <=
1'b0;
```

```
                                        txdigitalreset<= 1'b0;
                                        pll_areset <= 1'b0;
                                    end
                                else
                                    begin
                                        waitstate_timer <=
waitstate_timer - 1'b1;
                                        rxdigitalreset_inclk<= 1'b1;
                                         rxanalogreset_inclk <=
1'b0;
                                         txdigitalreset<= 1'b0;
                                         pll_areset <= 1'b0;
                                         state <= WAIT_STATE;
                                    end
                            end
                        else
                           begin
                               rxdigitalreset_inclk<= 1'b1;
                                rxanalogreset_inclk <= 1'b0;
                                txdigitalreset<= 1'b0;
                                pll_areset <= 1'b0;
                               waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                               state  <= STABLE_TX_PLL;
                           end

    default: state = IDLE;
  endcase
end

/*synchronizing the rxdigitalreset to recovered clock domain
If rxdigitalreset is only used for Receive GXB, then
synchronization is needed because
internally the rxdigitalreset is only synchronized to recovered
clock (rx_clkout). In the above description
 of the module, a typical user likes to operate on the system clock
or PLD clock domain.
To reset the rx_coreclk domain logic in PLD fabric following reset
is useful
*/

/* rxanalogreset is optional because it is a power down signal.
The longer the duration of assertion of power down
signal, the circuit will go into a true power down state
*/
    always @(posedge rx_coreclk or posedge async_reset)
        if(async_reset)
            begin
                rxdigitalreset_rx_coreclk_Q <= 1'b1;
                rxdigitalreset  <= 1'b1;

            end
         else
            begin
                if(receive_digitalreset)
                    begin
                        rxdigitalreset_rx_coreclk_Q <= 1'b1;
                        rxdigitalreset  <= 1'b1;
```

```
                    end
                else
                    begin
                        rxdigitalreset_rx_coreclk_Q <=
rxdigitalreset_inclk;
                        rxdigitalreset   <=
rxdigitalreset_rx_coreclk_Q;
                    end
            end

endmodule
```

Design Example 2
The following design example shows `inclk` as the input reference clock and the transmit parallel clock and `rx_clkout` as the receive parallel clock.

This design example has the following constraints:

- If your design requirements are different from the examples, use the flow charts and waveforms for each configuration as design guidelines.
- The design example requires a reset controller that generates a `sync_reset` (synchronous reset) for the entire system.
- The design example contains an `async_reset` (a power down in GXB terms) and digital resets for transmit and receive. All user input digital resets must be at least four cycles long.
- This design example does not cover all the digital reset scenarios in a system that resets the digital logic of the GXB.
- In this example, whenever the `rx_freqlocked` signal toggles the `rxdigitalreset`, the receiver's digital circuit is reset. However, you can make changes to the design to avoid this if, for example, you want to debug your design without the core being reset.
- If you plan to use `REFCLKB` pins in your design, see the *REFCLKB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* for information about the effects of analog resets (`pll_arest`, `rx_analogreset`).

```
/*
 Copyright (c) Altera Corporation, 2004.
This file may contain proprietary and confidential information of
Altera Corporation

Contacting Altera
=================

We have made every effort to ensure that this design example works
correctly. If you have a question or problem that is not answered
by the information then please contact Altera Support.

*************************************************************
```

```
Reset Sequence for the ALTGXB. The configuration of GXB for which
the following
reset sequence is valid is:
    Transmit and Receive : Both used
    Datapath    : Single Width(8/10 bits) or Double Width (16/20
bits)
    receive parallel clock: rx_clkout
    Functional Mode :'Any'
    RX PLL CRU  : Train RX PLL CRU with TX PLL ouput clock

*****************************************************************/


`timescale 1ns/10ps

module reset_seq_tx_train_rx_rx_clkout (
                            rx_clkout,
                            inclk,

                             sync_reset,
                             async_reset,
                             transmit_digitalreset,
                             receive_digitalreset,
                             pll_locked,
                             rx_freqlocked,

                            pll_areset,
                            txdigitalreset,
                             rxanalogreset,
                             rxdigitalreset

                            );


input inclk; //GXB input reference clock
input rx_clkout;//Receive recovered clock

input sync_reset;      //Input: synchronous reset from the system
input async_reset;   //Input: async reset from system
input transmit_digitalreset; //Input: Reset only the transmit
digital section
input receive_digitalreset; //Input : Reset the receiver section

input rx_freqlocked;   //rx_freqlocked signal from receive;
Transition from 'lock to reference clock mode' to 'lock to data
mode'
input pll_locked;      // Transmit PLL of GXB locked


output rxdigitalreset;//GXB Receive digital reset
output rxanalogreset;//Receive power down signal
output txdigitalreset; //GXB transmit digital reset
output pll_areset;//GXB power down signal

reg rxdigitalreset;
wire rxanalogreset;
reg txdigitalreset;
reg pll_areset;
```

```
reg [2:0] state;
reg rxdigitalreset_inclk;
reg rxanalogreset_inclk;

reg rxdigitalreset_rx_clkout_Q;
reg rxanalogreset_rx_clkout_Q;

parameter IDLE = 3'b000;
parameter STROBE_TXPLL_LOCKED = 3'b001;
parameter STABLE_TX_PLL = 3'b010;
parameter WAIT_STATE = 3'b011;


//Parameter value of T (2ms)based on the fastest clock (or 3.1875
Gbps)
parameter WAITSTATE_TIMER_VALUE = 1000000;

reg [19:0]waitstate_timer;  //timer - for actual value, refer
stratix data sheet

assign rxanalogreset = rxanalogreset_inclk;

always @ (posedge inclk or posedge async_reset) begin
if (async_reset)
    begin
        rxdigitalreset_inclk <= 1'b1;
        rxanalogreset_inclk <= 1'b1;
        txdigitalreset <= 1'b1;
        pll_areset <= 1'b1;
        waitstate_timer <= WAITSTATE_TIMER_VALUE;
        state       <= STROBE_TXPLL_LOCKED;
    end
else
  case (state)
    IDLE:
                if (sync_reset)      //Synchronous Reset can be
asserted in IDLE state (After reset seq has finished)
                    begin
                        rxdigitalreset_inclk <= 1'b1;
                        rxanalogreset_inclk <= 1'b1;
                        txdigitalreset <= 1'b1;
                        pll_areset <= 1'b1;
                        waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                        state<= STROBE_TXPLL_LOCKED;

                    end
                else
                    begin
                            rxdigitalreset_inclk <= 1'b0;
                            rxanalogreset_inclk <= 1'b0;
                            pll_areset <= 1'b0;
                            state   <= IDLE;
                            if(transmit_digitalreset)
                                txdigitalreset <= 1'b1;
                            else
                                txdigitalreset <= 1'b0;
                    end
```

```
    STROBE_TXPLL_LOCKED:  if (sync_reset)     //Synchronous Reset
can be asserted in IDLE state (After reset seq has finished)
                              begin
                                  rxdigitalreset_inclk <= 1'b1;
                                  rxanalogreset_inclk <= 1'b1;
                                  txdigitalreset <= 1'b1;
                                  pll_areset <= 1'b1;
                                  state <= STROBE_TXPLL_LOCKED;
                              end
    //Wait untill the TXPLL is locked to inclk and TX PLL has a
stable output clock which is also fed to RX CRU
                          else if (pll_locked)
                              begin
                                   state <= STABLE_TX_PLL;
                                   rxdigitalreset_inclk<= 1'b1;
                                   rxanalogreset_inclk <= 1'b0;
                                   txdigitalreset<= 1'b0;
                                   pll_areset <= 1'b0;
                              end
                          else
                              begin
                                  state <= STROBE_TXPLL_LOCKED;
                                  rxdigitalreset_inclk <= 1'b1;
                                  rxanalogreset_inclk <= 1'b1;
                                  txdigitalreset <= 1'b1;
                                  pll_areset <= 1'b0;
                              end
    STABLE_TX_PLL:  if (sync_reset)      //Synchronous Reset can
be asserted in IDLE state (After reset seq has finished)
                              begin
                                  rxdigitalreset_inclk <= 1'b1;
                                  rxanalogreset_inclk <= 1'b1;
                                  txdigitalreset <= 1'b1;
                                  pll_areset <= 1'b1;
                                  state <= STROBE_TXPLL_LOCKED;
                              end
                      else if (rx_freqlocked)
                          begin
                              state <= WAIT_STATE;
                              waitstate_timer <= waitstate_timer -
1'b1 ;
                               rxdigitalreset_inclk<= 1'b1;
                               rxanalogreset_inclk <= 1'b0;
                               txdigitalreset<= 1'b0;
                               pll_areset <= 1'b0;
                          end
                      else
                          begin
                              state <= STABLE_TX_PLL;
                               rxdigitalreset_inclk<= 1'b1;
                               rxanalogreset_inclk <= 1'b0;
                               txdigitalreset<= 1'b0;
                               pll_areset <= 1'b0;
                          end
    WAIT_STATE: if (sync_reset)      //Synchronous Reset can be
asserted in IDLE state (After reset seq has finished)
                              begin
```

```
                                    rxdigitalreset_inclk <= 1'b1;
                                    rxanalogreset_inclk <= 1'b1;
                                    txdigitalreset <= 1'b1;
                                    pll_areset <= 1'b1;
                                    state <= STROBE_TXPLL_LOCKED;
                                end
                    else if(rx_freqlocked)   //Condition to have
rx_freqlocked signal a stable high and should not bounce around
                            begin
                            //Decrement a Timer of 2ms (Refer
Stratix GX Datasheet for accurate value)after rx_freqlocked is
asserted
                            //This time is given to ensure the
recovered clock to be stable (No freq variations) and is locked
to incomming data
                                if(waitstate_timer == 0)
                                    begin
                                        state <= IDLE;
                                        rxdigitalreset_inclk<= 1'b0;
                                         rxanalogreset_inclk <=
1'b0;
                                         txdigitalreset<= 1'b0;
                                         pll_areset <= 1'b0;
                                    end
                                else
                                    begin
                                        waitstate_timer <=
waitstate_timer - 1'b1;
                                        rxdigitalreset_inclk<= 1'b1;
                                         rxanalogreset_inclk <=
1'b0;
                                         txdigitalreset<= 1'b0;
                                         pll_areset <= 1'b0;
                                         state<= WAIT_STATE;
                                    end
                            end
                    else
                       begin
                            rxdigitalreset_inclk<= 1'b1;
                             rxanalogreset_inclk <= 1'b0;
                             txdigitalreset<= 1'b0;
                             pll_areset <= 1'b0;
                            waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                            state <= STABLE_TX_PLL;
                       end

    default: state = IDLE;
  endcase
end

/*synchronizing the rxdigitalreset to recovered clock domain
If rxdigitalreset is only used for Receive GXB, then this
synchronization is redundant because
internally the rxdigitalreset is synchronized to recovered clock
(rx_clkout). In the above description
 of the module, a typical designer likes to operate on the system
clock
```

```
or PLD clock domain where one would like to have a FIFO with
rx_clkout domain being write clock and
 pld clock domain(Generic name, can be any clock name) as read
clock. To reset the rx_clkout domain logic in
PLD fabric following reset is useful
*/

    always @(posedge rx_clkout or posedge async_reset)
        if(async_reset)
            begin
                rxdigitalreset_rx_clkout_Q <= 1'b1;
                rxdigitalreset <= 1'b1;

            end
         else
            begin
                if(receive_digitalreset)
                    begin
                        rxdigitalreset_rx_clkout_Q <= 1'b1;
                        rxdigitalreset  <= 1'b1;
                    end
                else
                    begin
                        rxdigitalreset_rx_clkout_Q <=
rxdigitalreset_inclk;
                        rxdigitalreset   <=
rxdigitalreset_rx_clkout_Q;
                    end
            end

endmodule
```

Train Receive CRU With Transmit PLL Output Clock Option Disabled

The configuration in this section is similar to having two independent transmit and receive PLLs with their respective input reference clocks (`inclk` and `rx_cruclk`). In this configuration, both the transmit and receive parts of the transceiver are used. Figure 9–5 shows the possible clock options for the selected transceiver configuration.

Figure 9–5. Receiver & Transmitter With No Train Receiver CRU Option

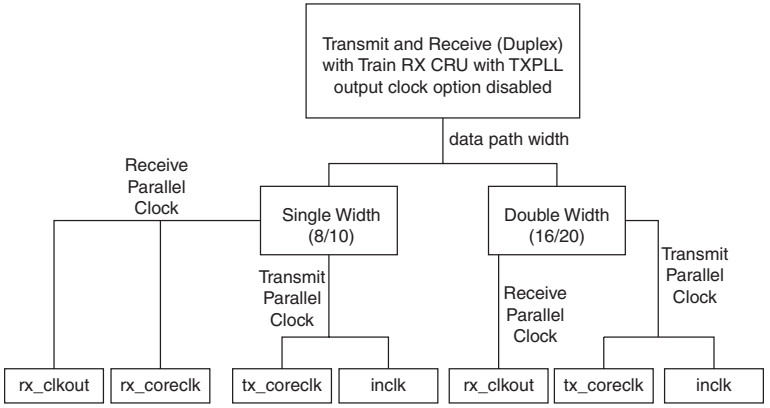
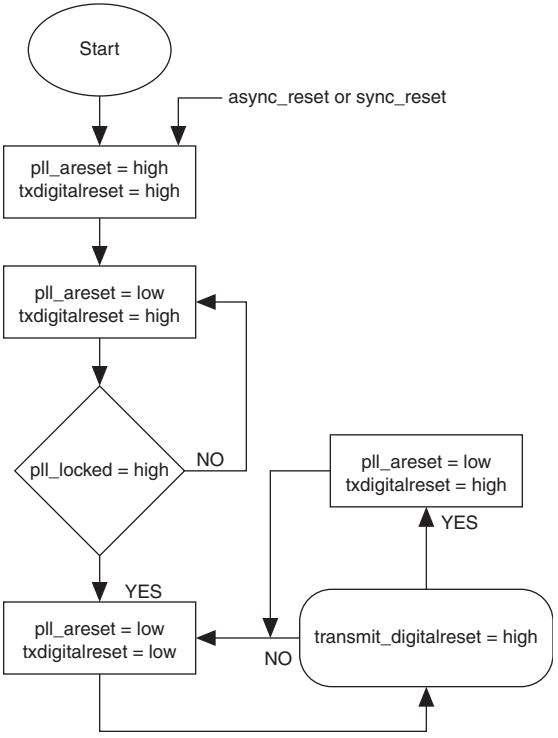


Figure 9–6 shows the transmitter reset sequence.

Figure 9–6. Transmitter Reset Sequence



The waveform in Figure 9–7 shows the functionality of the transmitter reset sequence shown in Figure 9–6. As described in Table 9–1 on page 9–3, the `pll_areset` resets the entire transceiver block, including both the analog and digital portions of the transmitter and receiver. After this signal is deasserted, the controller waits until the transmitter PLL is stable (`pll_locked = 1'b1`) before deasserting `tx_digitalreset`. This ensures that the output of the transmitter PLL is stable before releasing any of the logic that it feeds.

Figure 9–7. Transmitter Reset Sequence Waveform

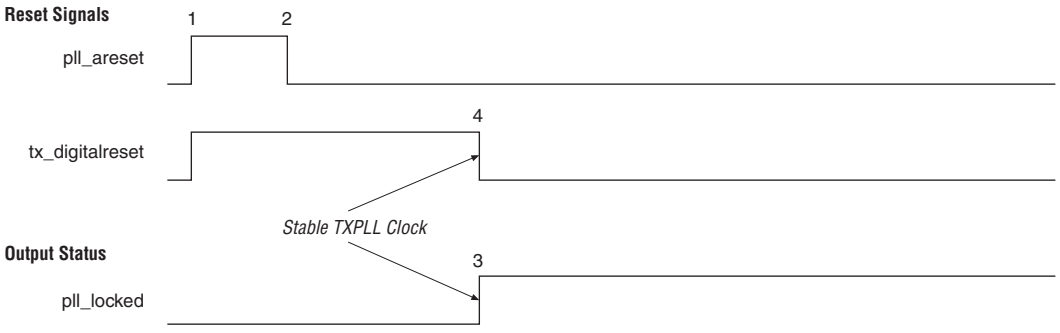
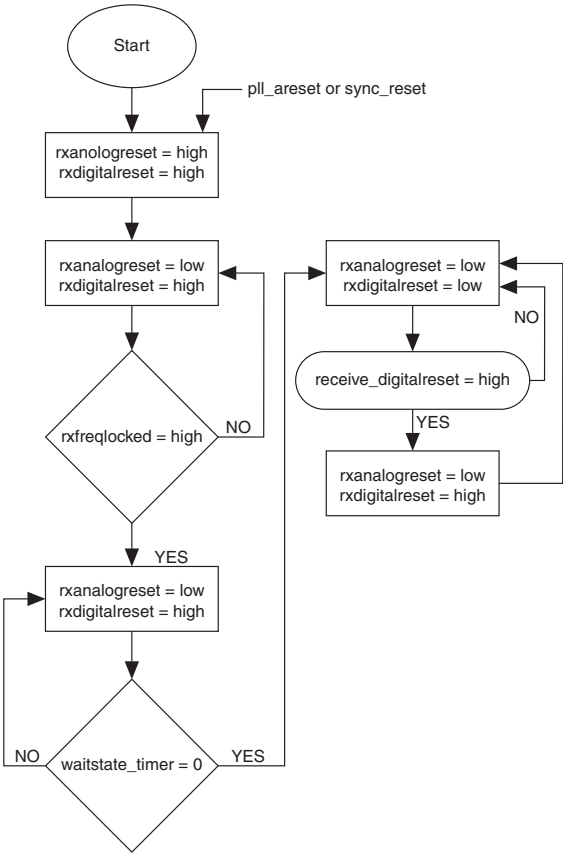


Figure 9–8 shows the receiver reset sequence.

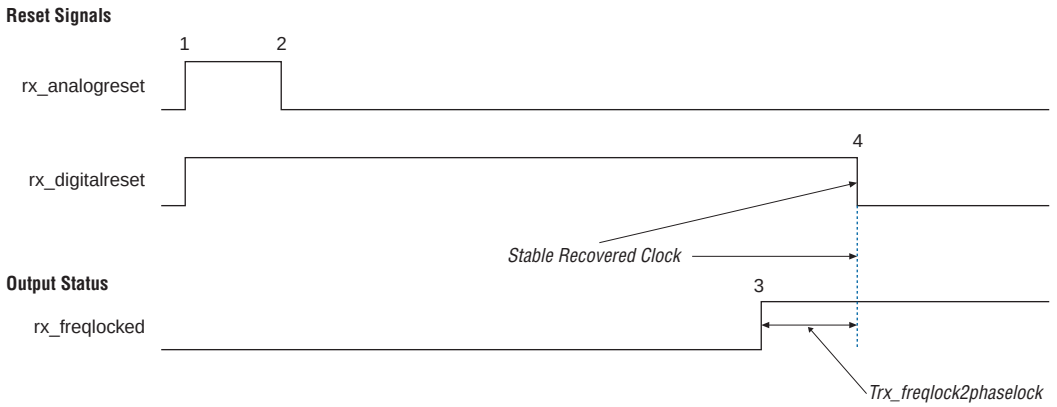
Figure 9–8. Receiver Reset Sequence



The waveform in Figure 9–9 shows the functionality of the receiver reset sequence shown in Figure 9–8. The `rx_analogreset` signal is pulsed. After the CRU has transitioned to locking to data from locking to the reference clock, the `rx_freqlocked` signal is asserted, which allows the reset sequence to transition into a wait state, where a timer is loaded with T ms. When the timer counts down the value, it signifies that `rx_clkout` is stable. The reset controller then deasserts the `rx_digital` reset, which completes the reset sequence. You should be able to monitor the BER (for example, a synchronization state machine based on the Stratix GX transceiver data) to determine whether the system is initialized and working properly.

See the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1* for the value of `Trx_freqlock2phaselock`.

Figure 9–9. Receiver Reset Sequence Waveform



Design Example 1

This design example shows `inclk` as the transmit PLL input reference clock and the transmit parallel clock, `rx_cruclk` as the receive CRU input reference clock, and `rx_coreclk` as the receive parallel clock.

This design example has following constraints:

- If your design requirements are different from the examples, use the flow charts and waveforms for each configuration as design guidelines.
- The design example requires a reset controller that generates a `sync_reset` (synchronous reset) for the entire system.
- The design example has an `async_reset` (a power down in GXB terms) and digital resets for transmit and receive. All user input digital resets must be at least four cycles long.
- This design example does not cover all the digital reset scenarios in a system that resets the digital logic of the GXB.
- In this example, whenever the `rx_freqlocked` signal toggles the `rxdigitalreset`, the receiver's digital circuit is reset. However, you can make changes to the design to avoid this if, for example, you want to debug your design without the core being reset.
- If you plan to use `REFCLKB` pins in your design, see the *REFCLKB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* for information about the effects of analog resets (`pll_arest`, `rx_analogreset`).

```
/*
 Copyright (c) Altera Corporation, 2004.
```

```
 This file may contain proprietary and confidential information of
 Altera Corporation

Contacting Altera
=================

We have made every effort to ensure that this design example works
correctly. If you have a question that is not answered by the
information, please contact Altera Support.

*****************************************************************
Reset Sequence for the ALTGXB. The configuration of GXB for which
the following
reset sequence is valid is:
    Transmit and Receive : Both used
    Datapath        : Single Width(8/10 bits)
    receive parallel clock: rx_coreclk
    Functional Mode :'Any'
    RX PLL CRU      : rx_cruclk
*****************************************************************/


`timescale 1ns/10ps

module reset_seq_tx_rx_rx_cruclk_rx_coreclk (
                            rx_coreclk,
                            inclk,
                            rx_cruclk,

                             sync_reset,
                             async_reset,
                             transmit_digitalreset,
                             receive_digitalreset,
                             pll_locked,
                             rx_freqlocked,

                            pll_areset,
                            txdigitalreset,
                             rxanalogreset,
                             rxdigitalreset

                            );


input inclk; //GXB input reference clock
input rx_cruclk;   //Receive GXB input reference clock
input rx_coreclk;//Receive recovered clock

input sync_reset;      //Input: synchronous reset from the system
input async_reset;   //Input: async reset from system
input transmit_digitalreset; //Input: Reset only the transmit
digital section
input receive_digitalreset; //Input : Reset the receiver section

input rx_freqlocked;   //rx_freqlocked signal from receive;
Transition from 'lock to reference clock mode' to 'lock to data
mode'
input pll_locked;      // Transmit PLL of GXB locked
```

```
output rxdigitalreset;//GXB Receive digital reset
output rxanalogreset;//Receive power down signal
output txdigitalreset; //GXB transmit digital reset
output pll_areset;//GXB power down signal

reg rxdigitalreset;
reg txdigitalreset;
reg pll_areset;
reg [2:0] state;
reg rxdigitalreset_rx_cruclk;

reg rxdigitalreset_rx_coreclk_Q;
reg rxanalogreset;

parameter IDLE = 3'b000;
parameter STROBE_TXPLL_LOCKED = 3'b001;
parameter STABLE_TX_PLL = 3'b010;


//Parameter value of T (2ms)based on the fastest clock (or 3.1875
Gbps)
parameter WAITSTATE_TIMER_VALUE = 1000000;

reg [19:0]waitstate_timer;  //timer - for actual value, refer
stratix data sheet


//Transmit Reset Sequence
always @ (posedge inclk or posedge async_reset) begin
if (async_reset)
    begin

        txdigitalreset <= 1'b1;
        pll_areset <= 1'b1;
        state <= STROBE_TXPLL_LOCKED;
    end
else
  case (state)
    IDLE:
                if (sync_reset)      //Synchronous Reset can be
asserted in IDLE state (After reset seq has finished)
                    begin

                        txdigitalreset <= 1'b1;
                        pll_areset <= 1'b1;
                        state<= STROBE_TXPLL_LOCKED;

                    end
                else
                    begin
                            pll_areset <= 1'b0;
                            state <= IDLE;
                            if(transmit_digitalreset)
                                txdigitalreset <= 1'b1;
                            else
                                txdigitalreset <= 1'b0;
```

```
                    end

    STROBE_TXPLL_LOCKED: if (sync_reset)    //Synchronous Reset
can be asserted in IDLE state (After reset seq has finished)
                        begin
                            txdigitalreset <= 1'b1;
                            pll_areset <= 1'b1;
                            state <= STROBE_TXPLL_LOCKED;
                        end
    //Wait untill the TXPLL is locked to inclk and TX PLL has a
stable output clock which is also fed to RX CRU
                    else if (pll_locked)
                        begin
                             state <= STABLE_TX_PLL;
                             txdigitalreset<= 1'b0;
                             pll_areset <= 1'b0;
                        end
                    else
                        begin
                            state <= STROBE_TXPLL_LOCKED;
                            txdigitalreset <= 1'b1;
                            pll_areset <= 1'b0;
                        end
    STABLE_TX_PLL:  if (sync_reset)     //Synchronous Reset can
be asserted in IDLE state (After reset seq has finished)
                        begin
                            txdigitalreset <= 1'b1;
                            pll_areset <= 1'b1;
                            state <= STROBE_TXPLL_LOCKED;
                        end
                    else
                        state <= IDLE;
    default: state = IDLE;
  endcase
end

        //Receive Reset Sequence
        always @(posedge rx_cruclk or posedge pll_areset)
                if(pll_areset)
                    begin
                        rxanalogreset <= 1'b1;
                        rxdigitalreset_rx_cruclk <= 1'b1;
                        waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                    end
                else
                    begin
                        if(sync_reset)
                            begin
                                rxanalogreset <= 1'b1;
                                rxdigitalreset_rx_cruclk<= 1'b1;
                                waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                            end
                        else
                            begin
                                rxanalogreset <= 1'b0;
                                if (rx_freqlocked)
```

```
                                    begin
                                        if(waitstate_timer == 0)
                                        begin

waitstate_timer <= waitstate_timer;
if(receive_digitalreset)

rxdigitalreset_rx_cruclk <= 1'b1;
                                                                    else

rxdigitalreset_rx_cruclk <= 1'b0;
                                    end
                                    else
                                    begin

waitstate_timer <= waitstate_timer - 1'b1;
rxdigitalreset_rx_cruclk <= 1'b1;
                                    end
                                        end
                                    else
                                        begin
                             rxdigitalreset_rx_cruclk <= 1'b1;
                                         waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                                        end
                                end
                            end




/*synchronizing the rxdigitalreset to recovered clock domain
If rxdigitalreset is used for Receive GXB, then this
synchronization is needed because
internally the rxdigitalreset is only synchronized to recovered
clock (rx_clkout).
To reset the rx_coreclk domain logic in PLD fabric following reset
is useful
*/

    always @(posedge rx_coreclk or posedge async_reset)
        if(async_reset)
            begin
                rxdigitalreset_rx_coreclk_Q <= 1'b1;
                rxdigitalreset <= 1'b1;

            end
         else
            begin
                if(receive_digitalreset)
                    begin
                        rxdigitalreset_rx_coreclk_Q <= 1'b1;
                        rxdigitalreset  <= 1'b1;
                    end
                else
                    begin
                        rxdigitalreset_rx_coreclk_Q <=
rxdigitalreset_rx_cruclk;
```

```
                        rxdigitalreset   <=
rxdigitalreset_rx_coreclk_Q;
                    end
            end

endmodule
```

Design Example 2

This design example shows `inclk` as the transmit PLL input reference clock and transmit parallel clock, `rx_cruclk` as the receive CRU input reference clock, and `rx_clkout` as the receive parallel clock.

This design example has the following constraints:

- If your design requirements are different from the examples, use the flow charts and waveforms for each configuration as design guidelines.
- The design example requires a reset controller that generates a `sync_reset` (synchronous reset) for the entire system.
- The design example has an `async_reset` (a power down in GXB terms) and digital resets for transmit and receive. All user input digital resets must be at least four cycles long.
- This design example does not cover all the digital reset scenarios in a system that resets the digital logic of the GXB.
- In this example, whenever the `rx_freqlocked` signal toggles the `rxdigitalreset`, the receiver's digital circuit is reset. However, you can make changes to the design to avoid this if, for example, you want to debug your design without the core being reset.
- If you plan to use `REFCLKB` pins in your design, see the *REFCLKB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* for information about the effects of analog resets (`pll_arest`, `rx_analogreset`).

```
/*
 Copyright (c) Altera Corporation, 2004.
 This file may contain proprietary and confidential information of
 Altera Corporation

Contacting Altera
=================

We have made every effort to ensure that this design example works
correctly. If you have a question that is not answered by the
information, please contact Altera Support.

*****************************************************************
Reset Sequence for the ALTGXB. The configuration of GXB for which
the following
reset sequence is valid is:
    Transmit and Receive : Both used
    Datapath     : Single Width(8/10 bits) or Double Width(16/20
bits)
```

```
    receive parallel clock: rx_clkout
    Functional Mode :'Any'
    RX PLL CRU : rx_cruclk

*****************************************************************/


`timescale 1ns/10ps

module reset_seq_tx_rx_rx_cruclk_rx_clkout (
                            rx_clkout,
                            inclk,
                            rx_cruclk,

                             sync_reset,
                             async_reset,
                             transmit_digitalreset,
                             receive_digitalreset,
                             pll_locked,
                             rx_freqlocked,

                            pll_areset,
                            txdigitalreset,
                             rxanalogreset,
                             rxdigitalreset

                            );


input inclk; //GXB input reference clock
input rx_cruclk;   //Receive GXB input reference clock
input rx_clkout;//Receive recovered clock

input sync_reset;      //Input: synchronous reset from the system
input async_reset;   //Input: async reset from system
input transmit_digitalreset; //Input: Reset only the transmit
digital section
input receive_digitalreset; //Input : Reset the receiver section

input rx_freqlocked;   //rx_freqlocked signal from receive;
Transition from 'lock to reference clock mode' to 'lock to data
mode'
input pll_locked;      // Transmit PLL of GXB locked


output rxdigitalreset;//GXB Receive digital reset
output rxanalogreset;//Receive power down signal
output txdigitalreset; //GXB transmit digital reset
output pll_areset;//GXB power down signal

reg rxdigitalreset;
reg txdigitalreset;
reg pll_areset;
reg [2:0] state;
reg rxdigitalreset_rx_cruclk;

reg rxdigitalreset_rx_clkout_Q;
reg rxanalogreset;
```

```
parameter IDLE      = 3'b000;
parameter STROBE_TXPLL_LOCKED = 3'b001;
parameter STABLE_TX_PLL = 3'b010;


//Parameter value of T (2ms)based on the fastest clock (or 3.1875
Gbps)
parameter WAITSTATE_TIMER_VALUE = 1000000;

reg [19:0]waitstate_timer;  //timer - for actual value, refer
stratix data sheet


//Transmit Reset Sequence
always @ (posedge inclk or posedge async_reset) begin
if (async_reset)
    begin

        txdigitalreset <= 1'b1;
        pll_areset <= 1'b1;
        state       <= STROBE_TXPLL_LOCKED;
    end
else
  case (state)
    IDLE:
                if (sync_reset)      //Synchronous Reset can be
asserted in IDLE state (After reset seq has finished)
                    begin

                        txdigitalreset <= 1'b1;
                        pll_areset <= 1'b1;
                        state   <= STROBE_TXPLL_LOCKED;

                    end
                else
                    begin
                            pll_areset <= 1'b0;
                            state       <= IDLE;
                            if(transmit_digitalreset)
                                txdigitalreset <= 1'b1;
                            else
                                txdigitalreset <= 1'b0;
                    end

    STROBE_TXPLL_LOCKED: if (sync_reset)     //Synchronous Reset
can be asserted in IDLE state (After reset seq has finished)
                            begin
                                txdigitalreset <= 1'b1;
                                pll_areset <= 1'b1;
                                state <= STROBE_TXPLL_LOCKED;
                            end
    //Wait untill the TXPLL is locked to inclk and TX PLL has a
stable output clock which is also fed to RX CRU
                        else if (pll_locked)
                            begin
                                 state <= STABLE_TX_PLL;
                                 txdigitalreset<= 1'b0;
```

```
                                    pll_areset <= 1'b0;
                                end
                            else
                                begin
                                    state <= STROBE_TXPLL_LOCKED;
                                    txdigitalreset <= 1'b1;
                                    pll_areset <= 1'b0;
                                end
    STABLE_TX_PLL:  if (sync_reset)      //Synchronous Reset can
be asserted in IDLE state (After reset seq has finished)
                                begin
                                    txdigitalreset <= 1'b1;
                                    pll_areset <= 1'b1;
                                    state       <=
STROBE_TXPLL_LOCKED;
                                end
                        else
                            state <= IDLE;
    default: state = IDLE;
  endcase
end

        //Receive Reset Sequence
        always @(posedge rx_cruclk or posedge pll_areset)
                if(pll_areset)
                    begin
                        rxanalogreset <= 1'b1;
                        rxdigitalreset_rx_cruclk <= 1'b1;
                        waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                        end
                else
                    begin
                        if(sync_reset)
                            begin
                                rxanalogreset <= 1'b1;
                                rxdigitalreset_rx_cruclk<= 1'b1;
                                waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                            end
                        else
                            begin
                                rxanalogreset <= 1'b0;
                                if (rx_freqlocked)
                                    begin
                                        if(waitstate_timer == 0)
                                            begin
                                                waitstate_timer
<= waitstate_timer;

if(receive_digitalreset)

rxdigitalreset_rx_cruclk <= 1'b1;
                                                    else

rxdigitalreset_rx_cruclk <= 1'b0;
                                                end
                                        else
```

```
                                                begin
                                                    waitstate_timer
<= waitstate_timer - 1'b1;

rxdigitalreset_rx_cruclk <= 1'b1;
                                                end
                                        end
                                    else
                                        begin

rxdigitalreset_rx_cruclk <= 1'b1;
                                             waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                                        end
                                end
                            end



/*synchronizing the rxdigitalreset to recovered clock domain
If rxdigitalreset is only used for Receive GXB, then the following
synchronization is not needed because
internally the rxdigitalreset is synchronized to recovered clock
(rx_clkout). In the above description
 of the module, a typical designer likes to operate on the system
clock
or PLD clock domain where one would like to have a FIFO with
rx_clkout domain being write clock and
may have pld clock domain(Generic name, can be any clock name) as
read clock. pld clock is optional.
To reset the rx_clkout domain logic in PLD fabric following reset
is useful
*/

    always @(posedge rx_clkout or posedge async_reset)
        if(async_reset)
            begin
                rxdigitalreset_rx_clkout_Q <= 1'b1;
                rxdigitalreset  <= 1'b1;

            end
         else
            begin
                if(receive_digitalreset)
                    begin
                        rxdigitalreset_rx_clkout_Q <= 1'b1;
                        rxdigitalreset  <= 1'b1;
                    end
                else
                    begin
                        rxdigitalreset_rx_clkout_Q <=
rxdigitalreset_rx_cruclk;
                        rxdigitalreset <=
rxdigitalreset_rx_clkout_Q;
                    end
            end
```

```
endmodule
```

Receiver Reset

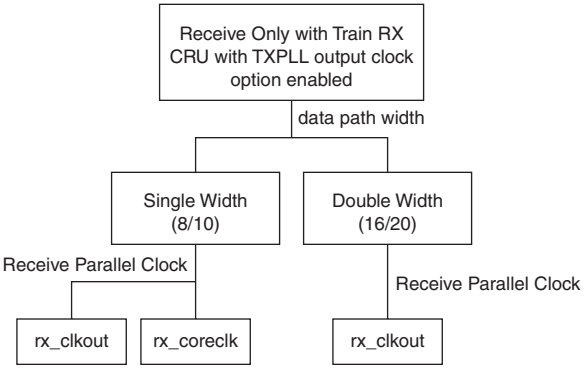
The configurations and design examples in this section describe how to implement a reset sequence for the receiver channels. This section describes the reset sequence only. Each design example lists the constraints specific to the example to help you understand the design parameters and limitations. You may want to add additional escape states and other system-specific features in your design. If your design requirements are different from the design example, you can make necessary changes using the flow chart and waveform figures in each section as guidelines.

Receive CRU With Transmit PLL Output Clock Option Enabled

This section provides some design examples that show a receive-only configuration with train receive CRU and the transmit PLL output clock enabled.

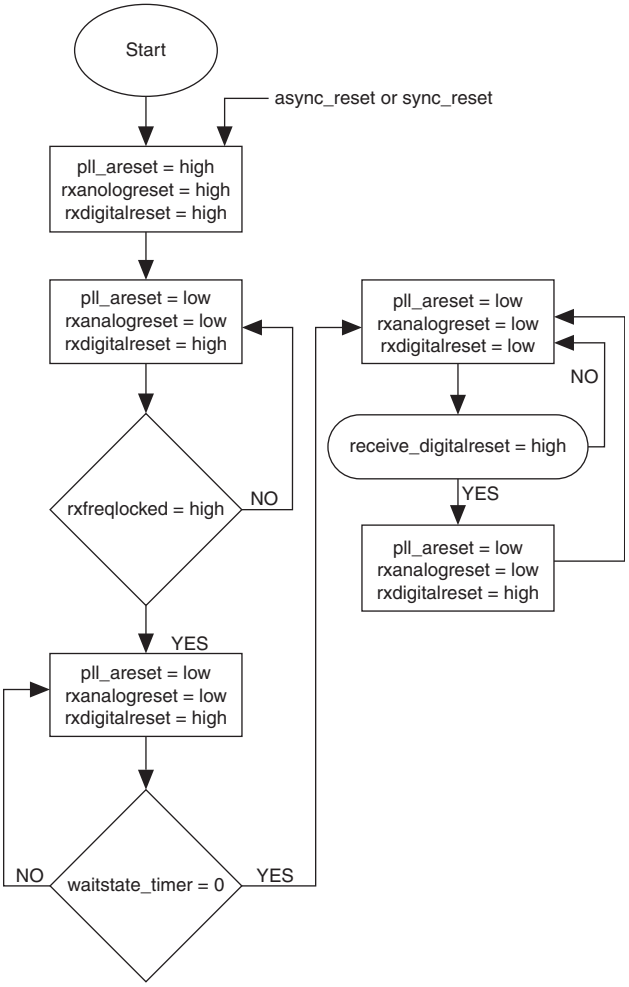
Figure 9–10 shows the receive-only clock options.

Figure 9–10. Receiver Only With Clock Options Enabled



The flow chart in Figure 9–11 shows a situation where only the receive channel requires a reset sequence.

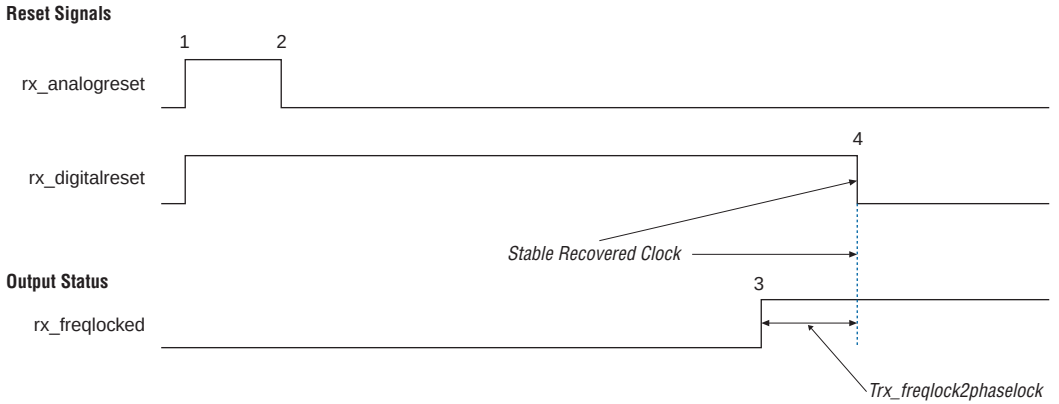
Figure 9–11. Receiver Reset Sequence



The waveform in Figure 9–12 shows the functionality of the receiver reset sequence shown in Figure 9–11. The `rx_analogreset` signal is pulsed. After the CRU has transitioned to locking-to-data from locking to the reference clock, the `rx_freqlocked` signal is asserted, which allows a reset sequence to transition into a wait state, where a timer is loaded with T ms. When the timer counts down the value, it signifies that `rx_clkout` is stable. The reset controller then deasserts the `rx_digital` reset, which completes the reset sequence.

See the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1* for the value of `Trx_freqlock2phaselock`.

Figure 9–12. Receiver Reset Sequence Waveform



Design Example 1

This design example shows a receive only configuration where `inclk` is the transmit PLL input reference clock, the output of transmit PLL trains receive CRU, and `rx_coreclk` is the receive parallel interface clock.

This design example has following constraints:

- If your design requirements are different from the examples, use the flow charts and waveforms for each configuration as design guidelines.
- The design example requires a reset controller that generates a `sync_reset` (synchronous reset) for the entire system.
- The design example has an `async_reset` (a power down in GXB terms) and digital resets for transmit and receive. All user input digital resets must be at least four cycles long.
- This design example does not cover all the digital reset scenarios in a system that resets the digital logic of the GXB.
- In this example, whenever the `rx_freqlocked` signal toggles the `rxdigitalreset`, the receiver's digital circuit is reset. However, you can make changes to the design to avoid this if, for example, you want to debug your design without the core being reset.
- If you plan to use `REFCLKB` pins in your design, see the *REFCLKB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* for information about the effects of analog resets (`pll_arest`, `rx_analogreset`).

```
/*
 Copyright (c) Altera Corporation, 2004.
 This file may contain proprietary and confidential information of
 Altera Corporation

Contacting Altera
=================

We have made every effort to ensure that this design example works
correctly. If you have a question that is not answered by the
information, please contact Altera Support.

*****************************************************************
Reset Sequence for the ALTGXB. The configuration of GXB for which
the following
reset sequence is valid is:
    Transmit and Receive : Receiver ONLY
    Datapath        : Single Width(8/10 bits) or Double Width
(16/20 bits)
    receive parallel clock: rx_coreclk
    Functional Mode :'Any'
    RX PLL CRU      : Train RX PLL CRU with TX PLL ouput clock
(refClk as shown in Mega Wizard)

*****************************************************************/


`timescale 1ns/10ps

module reset_seq_rx_ONLY_TXPLL_rx_coreclk (
                            rx_coreclk,
                            inclk,

                             sync_reset,
                             async_reset,
                             receive_digitalreset,
                             pll_locked,
                             rx_freqlocked,

                            pll_areset,
                             rxanalogreset,
                             rxdigitalreset

                            );


input inclk; //GXB input reference clock
input rx_coreclk;//Receive recovered clock

input sync_reset;      //Input: synchronous reset from the system
input async_reset;   //Input: async reset from system

input receive_digitalreset; //Input : Reset the receiver section

input rx_freqlocked;   //rx_freqlocked signal from receive;
Transition from 'lock to reference clock mode' to 'lock to data
mode'
input pll_locked;      // Transmit PLL of GXB locked
```

```
output rxdigitalreset;//GXB Receive digital reset
output rxanalogreset;//Receive power down signal

output pll_areset;//GXB power down signal

reg rxdigitalreset;
wire rxanalogreset;

reg pll_areset;
reg [2:0] state;
reg rxdigitalreset_inclk;
reg rxanalogreset_inclk;

reg rxdigitalreset_rx_coreclk_Q;
reg rxanalogreset_rx_coreclk_Q;

parameter IDLE = 3'b000;
parameter STROBE_TXPLL_LOCKED = 3'b001;
parameter STABLE_TX_PLL = 3'b010;
parameter WAIT_STATE = 3'b011;


//Parameter value of T (2ms)based on the fastest clock (or 3.1875
Gbps)
parameter WAITSTATE_TIMER_VALUE = 1000000;

reg [19:0]waitstate_timer;  //timer - for actual value, refer
stratix data sheet

assign rxanalogreset = rxanalogreset_inclk;

always @ (posedge inclk or posedge async_reset) begin
if (async_reset)
    begin
        rxdigitalreset_inclk <= 1'b1;
        rxanalogreset_inclk <= 1'b1;
        pll_areset <= 1'b1;
        waitstate_timer <= WAITSTATE_TIMER_VALUE;
        state <= STROBE_TXPLL_LOCKED;
    end
else
  case (state)
    IDLE:
                if (sync_reset)      //Synchronous Reset can be
asserted in IDLE state (After reset seq has finished)
                    begin
                        rxdigitalreset_inclk <= 1'b1;
                        rxanalogreset_inclk <= 1'b1;

                        pll_areset <= 1'b1;
                        waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                        state<= STROBE_TXPLL_LOCKED;

                    end
                else
```

```
                        begin
                                rxdigitalreset_inclk <= 1'b0;
                                rxanalogreset_inclk <= 1'b0;
                                pll_areset <= 1'b0;
                                state <= IDLE;

                        end

    STROBE_TXPLL_LOCKED:  if (sync_reset)     //Synchronous Reset
can be asserted in IDLE state (After reset seq has finished)
                                begin
                                    rxdigitalreset_inclk <= 1'b1;
                                    rxanalogreset_inclk <= 1'b1;

                                    pll_areset <= 1'b1;
                                    state <= STROBE_TXPLL_LOCKED;
                                end
    //Wait untill the TXPLL is locked to inclk and TX PLL has a
stable output clock which is also fed to RX CRU
                            else if (pll_locked)
                                begin
                                     state <= STABLE_TX_PLL;
                                     rxdigitalreset_inclk<= 1'b1;
                                     rxanalogreset_inclk <= 1'b0;

                                     pll_areset <= 1'b0;
                                end
                            else
                                begin
                                    state <= STROBE_TXPLL_LOCKED;
                                    rxdigitalreset_inclk <= 1'b1;
                                    rxanalogreset_inclk <= 1'b1;

                                    pll_areset <= 1'b0;
                                end
    STABLE_TX_PLL:  if (sync_reset)      //Synchronous Reset can
be asserted in IDLE state (After reset seq has finished)
                                begin
                                    rxdigitalreset_inclk <= 1'b1;
                                    rxanalogreset_inclk <= 1'b1;

                                    pll_areset <= 1'b1;
                                    state <= STROBE_TXPLL_LOCKED;
                                end
                        else if (rx_freqlocked)
                            begin
                                state <= WAIT_STATE;
                                waitstate_timer <= waitstate_timer -
1'b1 ;
                                 rxdigitalreset_inclk<= 1'b1;
                                 rxanalogreset_inclk <= 1'b0;

                                 pll_areset <= 1'b0;
                            end
                        else
                            begin
                                state <= STABLE_TX_PLL;
                                 rxdigitalreset_inclk<= 1'b1;
```

```
                                rxanalogreset_inclk <= 1'b0;

                                pll_areset <= 1'b0;
                            end
    WAIT_STATE: if (sync_reset)      //Synchronous Reset can be
asserted in IDLE state (After reset seq has finished)
                                begin
                                    rxdigitalreset_inclk <= 1'b1;
                                    rxanalogreset_inclk <= 1'b1;

                                    pll_areset <= 1'b1;
                                    state <= STROBE_TXPLL_LOCKED;
                                end
                    else if(rx_freqlocked)   //Condition to have
rx_freqlocked signal a stable high and should not bounce around
                            begin
                            //Decrement a Timer of 2ms (Refer
Stratix GX Datasheet for accurate value)after rx_freqlocked is
asserted
                            //This time is given to ensure the
recovered clock to be stable (Cannot have any freq variations) and
is locked to incomming data
                                if(waitstate_timer == 0)
                                    begin
                                        state <= IDLE;
                                        rxdigitalreset_inclk<= 1'b0;
                                         rxanalogreset_inclk <=
1'b0;

                                         pll_areset <= 1'b0;
                                    end
                                else
                                    begin
                                        waitstate_timer <=
waitstate_timer - 1'b1;
                                        rxdigitalreset_inclk<= 1'b1;
                                         rxanalogreset_inclk <=
1'b0;

                                         pll_areset <= 1'b0;
                                         state <= WAIT_STATE;
                                    end
                            end
                    else
                       begin
                            rxdigitalreset_inclk<= 1'b1;
                             rxanalogreset_inclk <= 1'b0;

                             pll_areset <= 1'b0;
                            waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                            state       <= STABLE_TX_PLL;
                       end

    default: state = IDLE;
  endcase
end
```

```
/*synchronizing the rxdigitalreset to recovered clock domain
If rxdigitalreset is used for Receive GXB, then this
synchronization is needed because
internally the rxdigitalreset is synchronized to recovered clock
(rx_clkout).To reset the rx_coreclk domain logic in
PLD fabric following reset is useful
*/

    always @(posedge rx_coreclk or posedge async_reset)
        if(async_reset)
            begin
                rxdigitalreset_rx_coreclk_Q <= 1'b1;
                rxdigitalreset <= 1'b1;

            end
         else
            begin
                if(receive_digitalreset)
                    begin
                        rxdigitalreset_rx_coreclk_Q <= 1'b1;
                        rxdigitalreset  <= 1'b1;
                    end
                else
                    begin
                        rxdigitalreset_rx_coreclk_Q <=
rxdigitalreset_inclk;
                        rxdigitalreset   <=
rxdigitalreset_rx_coreclk_Q;
                    end
            end

endmodule
```

Design Example 2
This design example shows a receive-only configuration where `inclk` is the transmit PLL input reference clock, the output of transmit PLL trains receive CRU, and `rx_clkout` is the receive parallel interface clock.

This design example has the following constraints:

- If your design requirements are different from the examples, use the flow charts and waveforms for each configuration as design guidelines.
- The design example requires a reset controller that generates a `sync_reset` (synchronous reset) for the entire system.
- The design example has an `async_reset` (a power down in GXB terms) and digital resets for transmit and receive. All user input digital resets must be at least four cycles long.
- This design example does not cover all the digital reset scenarios in a system that resets the digital logic of the GXB.

- In this example, whenever the rx_freqlocked signal toggles the rxdigitalreset, the receiver's digital circuit is reset. However, you can make changes to the design to avoid this if, for example, you want to debug your design without the core being reset.
- If you plan to use REFCLKB pins in your design, see the *REFCLKB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* for information about the effects of analog resets (pll_arest, rx_analogreset).

```
/*
 Copyright (c) Altera Corporation, 2004.
 This file may contain proprietary and confidential information of
 Altera Corporation

Contacting Altera
=================

We have made every effort to ensure that this design example works
correctly. If you have a question that is not answered by the
information then please contact Altera Support.

*****************************************************************
Reset Sequence for the ALTGXB. The configuration of GXB for which
the following
reset sequence is valid is:
    Transmit and Receive : Receiver ONLY
    Datapath    : Single Width(8/10 bits) or Double Width (16/20
bits)
    receive parallel clock: rx_clkout
    Functional Mode :'Any'
    RX PLL CRU : Train RX PLL CRU with TX PLL ouput clock (refClk
as shown in Mega Wizard)

*****************************************************************/

`timescale 1ns/10ps

module reset_seq_rx_ONLY_TXPLL_rx_clkout (
                             rx_clkout,
                             inclk,

                              sync_reset,
                              async_reset,
                              receive_digitalreset,
                              pll_locked,
                              rx_freqlocked,

                             pll_areset,
                              rxanalogreset,
                              rxdigitalreset

                             );


input inclk; //GXB input reference clock
input rx_clkout;//Receive recovered clock
```

```
input sync_reset;      //Input: synchronous reset from the system
input async_reset;   //Input: async reset from system

input receive_digitalreset; //Input : Reset the receiver section

input rx_freqlocked;   //rx_freqlocked signal from receive;
Transition from 'lock to reference clock mode' to 'lock to data
mode'
input pll_locked;      // Transmit PLL of GXB locked


output rxdigitalreset;//GXB Receive digital reset
output rxanalogreset;//Receive power down signal

output pll_areset;//GXB power down signal

reg rxdigitalreset;
wire rxanalogreset;

reg pll_areset;
reg [2:0] state;
reg rxdigitalreset_inclk;
reg rxanalogreset_inclk;

reg rxdigitalreset_rx_clkout_Q;
reg rxanalogreset_rx_clkout_Q;

parameter IDLE      = 3'b000;
parameter STROBE_TXPLL_LOCKED = 3'b001;
parameter STABLE_TX_PLL = 3'b010;
parameter WAIT_STATE = 3'b011;


//Parameter value of T (2ms)based on the fastest clock (or 3.1875
Gbps)
parameter WAITSTATE_TIMER_VALUE = 1000000;

reg [19:0]waitstate_timer;  //timer - for actual value, refer
stratix data sheet

assign rxanalogreset = rxanalogreset_inclk;

always @ (posedge inclk or posedge async_reset) begin
if (async_reset)
    begin
        rxdigitalreset_inclk <= 1'b1;
        rxanalogreset_inclk <= 1'b1;
        pll_areset <= 1'b1;
        waitstate_timer <= WAITSTATE_TIMER_VALUE;
        state       <= STROBE_TXPLL_LOCKED;
    end
else
  case (state)
    IDLE:
                if (sync_reset)      //Synchronous Reset can be
asserted in IDLE state (After reset seq has finished)
                    begin
```

```
                            rxdigitalreset_inclk <= 1'b1;
                            rxanalogreset_inclk <= 1'b1;

                            pll_areset <= 1'b1;
                            waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                            state<= STROBE_TXPLL_LOCKED;

                        end
                    else
                        begin
                                rxdigitalreset_inclk <= 1'b0;
                                rxanalogreset_inclk <= 1'b0;
                                pll_areset <= 1'b0;
                                state   <= IDLE;

                        end

    STROBE_TXPLL_LOCKED:  if (sync_reset)     //Synchronous Reset
can be asserted in IDLE state (After reset seq has finished)
                                begin
                                    rxdigitalreset_inclk <= 1'b1;
                                    rxanalogreset_inclk <= 1'b1;

                                    pll_areset <= 1'b1;
                                    state       <=
STROBE_TXPLL_LOCKED;
                                end
    //Wait untill the TXPLL is locked to inclk and TX PLL has a
stable output clock which is also fed to RX CRU
                            else if (pll_locked)
                                begin
                                     state <= STABLE_TX_PLL;
                                     rxdigitalreset_inclk<= 1'b1;
                                     rxanalogreset_inclk <= 1'b0;

                                     pll_areset <= 1'b0;
                                end
                            else
                                begin
                                    state <= STROBE_TXPLL_LOCKED;
                                    rxdigitalreset_inclk <= 1'b1;
                                    rxanalogreset_inclk <= 1'b1;

                                    pll_areset <= 1'b0;
                                end
    STABLE_TX_PLL:  if (sync_reset)      //Synchronous Reset can
be asserted in IDLE state (After reset seq has finished)
                                begin
                                    rxdigitalreset_inclk <= 1'b1;
                                    rxanalogreset_inclk <= 1'b1;

                                    pll_areset <= 1'b1;
                                    state <= STROBE_TXPLL_LOCKED;
                                end
                            else if (rx_freqlocked)
                                begin
                                    state <= WAIT_STATE;
```

```
                                waitstate_timer <= waitstate_timer -
1'b1 ;
                                 rxdigitalreset_inclk<= 1'b1;
                                 rxanalogreset_inclk <= 1'b0;

                                 pll_areset <= 1'b0;
                            end
                        else
                            begin
                                state <= STABLE_TX_PLL;
                                 rxdigitalreset_inclk<= 1'b1;
                                 rxanalogreset_inclk <= 1'b0;

                                 pll_areset <= 1'b0;
                            end
    WAIT_STATE: if (sync_reset)      //Synchronous Reset can be
asserted in IDLE state (After reset seq has finished)
                                begin
                                    rxdigitalreset_inclk <= 1'b1;
                                    rxanalogreset_inclk <= 1'b1;

                                    pll_areset <= 1'b1;
                                    state <= STROBE_TXPLL_LOCKED;
                                end
                    else if(rx_freqlocked)   //Condition to have
rx_freqlocked signal a stable high and should not bounce around
                            begin
                            //Decrement a Timer of 2ms (Refer
Stratix GX Datasheet for accurate value)after rx_freqlocked is
asserted
                            //This time is given to ensure the
recovered clock to be stable (Cannot have any freq variations) and
is locked to incomming data
                                if(waitstate_timer == 0)
                                    begin
                                        state <= IDLE;
                                        rxdigitalreset_inclk<= 1'b0;
                                         rxanalogreset_inclk <=
1'b0;

                                         pll_areset <= 1'b0;
                                    end
                                else
                                    begin
                                        waitstate_timer <=
waitstate_timer - 1'b1;
                                        rxdigitalreset_inclk<= 1'b1;
                                         rxanalogreset_inclk <=
1'b0;

                                         pll_areset <= 1'b0;
                                         state <= WAIT_STATE;
                                    end
                            end
                       else
                          begin
                              rxdigitalreset_inclk<= 1'b1;
                               rxanalogreset_inclk <= 1'b0;
```

```
                         pll_areset <= 1'b0;
                        waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                        state <= STABLE_TX_PLL;
                    end

    default: state = IDLE;
  endcase
end

/*synchronizing the rxdigitalreset to recovered clock domain
If rxdigitalreset is only used for receive GXB, synchronization
is redundant because internally the rxdigitalreset is
synchronized to the recovered clock (rx_clkout). In the above
description of the module, User likes to operate on the system
clock or PLD clock domain where one would like to have a FIFO with
rx_clkout domain being write clock and pld clock domain (generic
name, can be any clock name) as read clock.
To reset the rx_clkout domain logic in PLD fabric following reset
is useful
*/

    always @(posedge rx_clkout or posedge async_reset)
        if(async_reset)
            begin
                rxdigitalreset_rx_clkout_Q <= 1'b1;
                rxdigitalreset <= 1'b1;

            end
         else
            begin
                if(receive_digitalreset)
                    begin
                        rxdigitalreset_rx_clkout_Q <= 1'b1;
                        rxdigitalreset  <= 1'b1;
                    end
                else
                    begin
                        rxdigitalreset_rx_clkout_Q <=
rxdigitalreset_inclk;
                        rxdigitalreset   <=
rxdigitalreset_rx_clkout_Q;
                    end
            end

endmodule
```

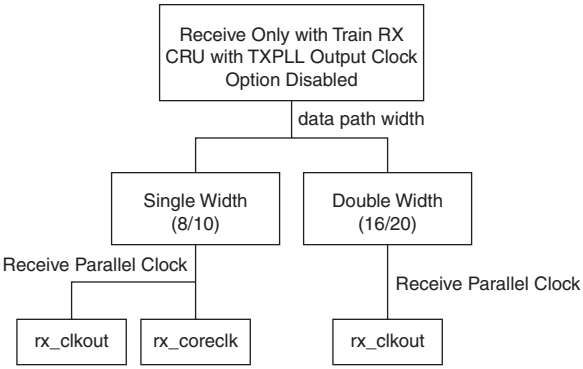
Receive CRU With Transmit PLL Output Clock Option Disabled

This section provides examples that show a receive-only configuration with a train receive CRU and the transmitter PLL output clock option disabled. The flow chart in Figure 9–11 on page 9–33 and the waveform shown in Figure 9–12 on page 9–34 are valid for this configuration also.

The difference in this configuration from the configuration in "Receive CRU With Transmit PLL Output Clock Option Enabled" on page 9–32 is that receive CRU is trained by the input pin `rx_cruclk`.

Figure 9–13 shows the receive-only configuration with clock options disabled.

Figure 9–13. Receive Clock Only With Clock Options Disabled



Design Example 1

This design example shows a receive only configuration with `rx_cruclk` as the receive CRU input reference clock and `rx_coreclk` as the receive parallel interface clock.

This design example has the following constraints:

- If your design requirements are different from the examples, use the flow charts and waveforms for each configuration as design guidelines.
- The design example requires a reset controller that generates a `sync_reset` (synchronous reset) for the entire system.
- The design example has an `async_reset` (a power down in GXB terms) and digital resets for transmit and receive. All user input digital resets must be at least four cycles long.
- This design example does not cover all the digital reset scenarios in a system that resets the digital logic of the GXB.
- In this example, whenever the `rx_freqlocked` signal toggles the `rxdigitalreset`, the receiver's digital circuit is reset. However, you can make changes to the design to avoid this if, for example, you want to debug your design without the core being reset.

- If you plan to use REFCLKB pins in your design, see the *REFCLKB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* for information about the effects of analog resets (pll_arest, rx_analogreset).

```
/*
 Copyright (c) Altera Corporation, 2004.
 This file may contain proprietary and confidential information of
 Altera Corporation

Contacting Altera
=================

We have made every effort to ensure that this design example works
correctly. If you have a question that is not answered by the
information, please contact Altera Support.

*****************************************************************
Reset Sequence for the ALTGXB. The configuration of GXB for which
the following
reset sequence is valid is:
    Transmit and Receive : Receive Only
    Datapath : Single Width(8/10 bits)
    receive parallel clock: rx_coreclk
    Functional Mode :'Any'
    RX PLL CRU : rx_cruclk

*****************************************************************/


`timescale 1ns/10ps

module reset_seq_rx_rx_cruclk_rx_coreclk (
                             rx_coreclk,
                             rx_cruclk,

                              sync_reset,
                              async_reset,
                              receive_digitalreset,
                              rx_freqlocked,

                              rxanalogreset,
                              rxdigitalreset

                             );



input rx_cruclk;   //Receive GXB input reference clock
input rx_coreclk;//Receive recovered clock

input sync_reset;      //Input: synchronous reset from the system
input async_reset;   //Input: async reset from system
input receive_digitalreset; //Input : Reset the receiver section
```

```
input rx_freqlocked;   //rx_freqlocked signal from receive;
Transition from 'lock to reference clock mode' to 'lock to data
mode'



output rxdigitalreset;//GXB Receive digital reset
output rxanalogreset;//Receive power down signal

reg rxdigitalreset;


reg rxdigitalreset_rx_cruclk;
reg rxdigitalreset_rx_coreclk_Q;
reg rxanalogreset;



//Parameter value of T (2ms)based on the fastest clock (or 3.1875
Gbps)
parameter WAITSTATE_TIMER_VALUE = 1000000;

reg [19:0]waitstate_timer;  //timer - for actual value, refer
stratix data sheet

        //Receive Reset Sequence
        always @(posedge rx_cruclk or posedge async_reset)
                if(async_reset)
                    begin
                        rxanalogreset <= 1'b1;
                        rxdigitalreset_rx_cruclk <= 1'b1;
                        waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                    end
                else
                    begin
                        if(sync_reset)
                            begin
                                rxanalogreset <= 1'b1;
                                rxdigitalreset_rx_cruclk<= 1'b1;
                                waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                            end
                        else
                            begin
                                rxanalogreset <= 1'b0;
                                if (rx_freqlocked)
                                    begin
                                        if(waitstate_timer == 0)
                                            begin
                                                waitstate_timer
<= waitstate_timer;

if(receive_digitalreset)

rxdigitalreset_rx_cruclk <= 1'b1;
                                                    else
```

```
rxdigitalreset_rx_cruclk <= 1'b0;
                                                end
                                            else
                                                begin
                                                    waitstate_timer
<= waitstate_timer - 1'b1;

rxdigitalreset_rx_cruclk <= 1'b1;
                                                end
                                        end
                                    else
                                        begin

rxdigitalreset_rx_cruclk <= 1'b1;
                                             waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                                        end
                                end
                            end



/*synchronizing the rxdigitalreset to recovered clock domain
If rxdigitalreset is used for Receive GXB, then this
synchronization is needed because
internally the rxdigitalreset is only synchronized to recovered
clock (rx_clkout).
To reset the rx_coreclk domain logic in PLD fabric following reset
is useful
*/


    always @(posedge rx_coreclk or posedge async_reset)
        if(async_reset)
            begin
                rxdigitalreset_rx_coreclk_Q <= 1'b1;
                rxdigitalreset <= 1'b1;

            end
         else
            begin
                if(receive_digitalreset)
                    begin
                        rxdigitalreset_rx_coreclk_Q <= 1'b1;
                        rxdigitalreset <= 1'b1;
                    end
                else
                    begin
                        rxdigitalreset_rx_coreclk_Q <=
rxdigitalreset_rx_cruclk;
                        rxdigitalreset   <=
rxdigitalreset_rx_coreclk_Q;
                    end
            end

endmodule
```

Design Example 2
This design example shows a receive-only configuration with `rx_cruclk` as the receive CRU input reference clock and `rx_clkout` as the receive parallel interface clock.

This design example has the following constraints:

- If your design requirements are different from the examples, use the flow charts and waveforms for each configuration as design guidelines.
- The design example requires a reset controller that generates a `sync_reset` (synchronous reset) for the entire system.
- The design example has an `async_reset` (a power down in GXB terms) and digital resets for transmit and receive. All user input digital resets must be at least four cycles long.
- This design example does not cover all the digital reset scenarios in a system that resets the digital logic of the GXB.
- In this example, whenever the `rx_freqlocked` signal toggles the `rxdigitalreset`, the receiver's digital circuit is reset. However, you can make changes to the design to avoid this if, for example, you want to debug your design without the core being reset.
- If you plan to use `REFCLKB` pins in your design, see the *REFCLKB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* for information about the effects of analog resets (`pll_arest`, `rx_analogreset`).

```
/*
 Copyright (c) Altera Corporation, 2004.
 This file may contain proprietary and confidential information of
 Altera Corporation

Contacting Altera
=================

we have made every effort to ensure that this design example works
correctly. If you have a question that is not answered by the
information, please contact Altera Support.

*****************************************************************
Reset Sequence for the ALTGXB. The configuration of GXB for which
the following
reset sequence is valid is:
    Transmit and Receive : Receive Only
    Datapath    : Single Width(8/10 bits) or Double Width(16/20
bits)
    receive parallel clock: rx_clkout
    Functional Mode :'Any'
    RX PLL CRU : rx_cruclk


*****************************************************************/
```

```
`timescale 1ns/10ps

module reset_seq_rx_rx_cruclk_rx_clkout (
                              rx_clkout,
                              rx_cruclk,

                               sync_reset,
                               async_reset,
                               receive_digitalreset,
                               rx_freqlocked,

                               rxanalogreset,
                               rxdigitalreset

                              );


input rx_cruclk;   //Receive GXB input reference clock
input rx_clkout;//Receive recovered clock

input sync_reset;      //Input: synchronous reset from the system
input async_reset;   //Input: async reset from system
input receive_digitalreset; //Input : Reset the receiver section

input rx_freqlocked;   //rx_freqlocked signal from receive;
Transition from 'lock to reference clock mode' to 'lock to data
mode'



output rxdigitalreset;//GXB Receive digital reset
output rxanalogreset;//Receive power down signal

reg rxdigitalreset;


reg rxdigitalreset_rx_cruclk;
reg rxdigitalreset_rx_clkout_Q;
reg rxanalogreset;



//Parameter value of T (2ms)based on the fastest clock (or 3.1875
Gbps)
parameter WAITSTATE_TIMER_VALUE = 1000000;

reg [19:0]waitstate_timer;  //timer - for actual value, refer
stratix data sheet

        //Receive Reset Sequence
        always @(posedge rx_cruclk or posedge async_reset)
                if(async_reset)
                    begin
                        rxanalogreset <= 1'b1;
                        rxdigitalreset_rx_cruclk <= 1'b1;
```

```
                        waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                    end
                else
                    begin
                        if(sync_reset)
                            begin
                                rxanalogreset <= 1'b1;
                                rxdigitalreset_rx_cruclk<= 1'b1;
                                waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                            end
                        else
                            begin
                                rxanalogreset <= 1'b0;
                                if (rx_freqlocked)
                                    begin
                                    if(waitstate_timer == 0)
                                            begin
                                                waitstate_timer
<= waitstate_timer;

if(receive_digitalreset)

rxdigitalreset_rx_cruclk <= 1'b1;
                                                    else

rxdigitalreset_rx_cruclk <= 1'b0;
                                                end
                                            else
                                                begin
                                                    waitstate_timer
<= waitstate_timer - 1'b1;

rxdigitalreset_rx_cruclk <= 1'b1;
                                                end
                                        end
                                    else
                                        begin

rxdigitalreset_rx_cruclk <= 1'b1;
                                             waitstate_timer <=
WAITSTATE_TIMER_VALUE;
                                        end
                                end
                        end



/*synchronizing the rxdigitalreset to recovered clock domain
If rxdigitalreset is only used for Receive GXB, then this
synchronization is not needed because internally the
rxdigitalreset is only synchronized to recovered clock
(rx_clkout). In the above description of the module, a designer
likes to operate on the system clock or PLD clock domain where one
would like to have a FIFO with rx_clkout domain being write clock
and may have pld clock domain(Generic name, can be any clock name)
```

```
as read clock. pld clock is optional. To reset the rx_clkout
domain logic in PLD fabric following reset is useful*/

    always @(posedge rx_clkout or posedge async_reset)
        if(async_reset)
            begin
                rxdigitalreset_rx_clkout_Q <= 1'b1;
                rxdigitalreset  <= 1'b1;

            end
         else
            begin
                if(receive_digitalreset)
                    begin
                        rxdigitalreset_rx_clkout_Q <= 1'b1;
                        rxdigitalreset  <= 1'b1;
                    end
                else
                    begin
                        rxdigitalreset_rx_clkout_Q <=
rxdigitalreset_rx_cruclk;
                        rxdigitalreset   <=
rxdigitalreset_rx_clkout_Q;
                    end
            end

endmodule
```

Transmitter Reset

The configurations and design examples in this section show how to implement a reset sequence for the transmitter channels. In this configuration, GXB is configured only as a transmitter. In the design examples, the `tx_coreclk` option is not shown because the reset signals (`txdigitalreset`) based on `tx_coreclk` are synchronized internally by the reset controller in the Stratix GX hard IP. This configuration only demonstrates the reset sequence. You might want to add additional escape states and other system-specific features in your design. If your design requirements are different from the design example, you can make necessary changes, using the flow chart and waveform figures in each section as guidelines.

Figure 9–14. Transmitter Only Clock Options

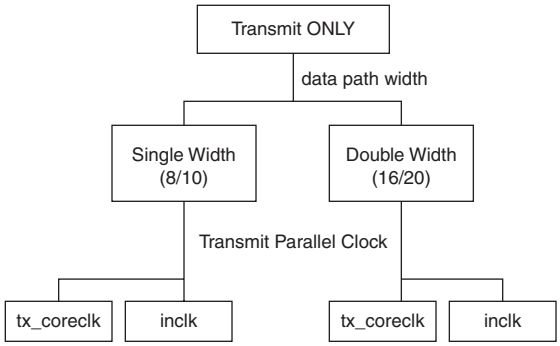
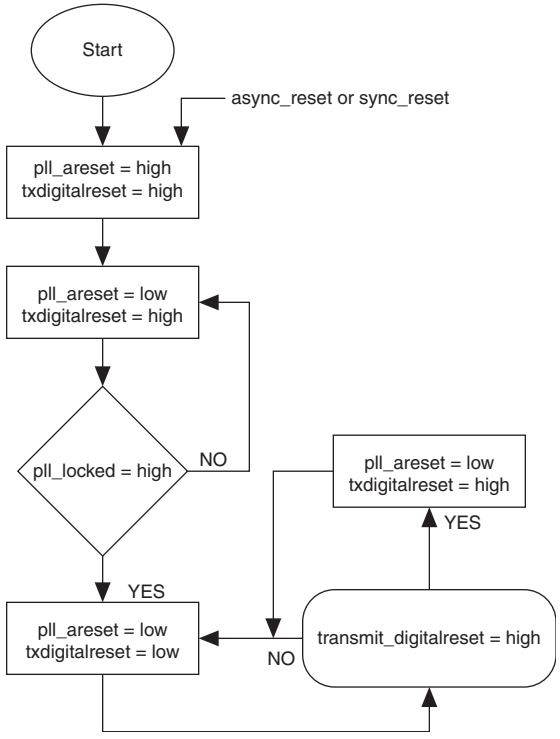


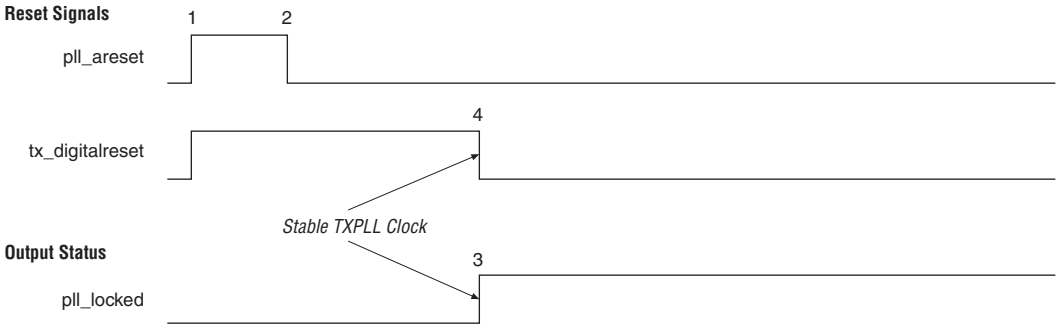
Figure 9–15 shows a situation where only the transmitter channel requires a reset sequence.

Figure 9–15. Transmitter Reset Sequence



The waveform in Figure 9–16 shows the functionality of the transmitter reset sequence shown in Figure 9–15. As described in Table 9–1 on page 9–3, the `pll_areset` resets the entire transceiver block, including both the analog and digital portions of the transmitter and receiver. After this signal is deasserted, the controller waits until the transmitter PLL is stable (`pll_locked = 1'b1`) before deasserting `tx_digitalreset`. This ensures that the output of the transmitter PLL is stable before releasing any of the logic that it feeds.

Figure 9–16. Transmitter Reset Sequence Waveform



Design Example 1

This design example shows a transmit-only configuration with `inclk` as both the transmit PLL input reference clock and the transmit parallel interface clock.

This design example has the following constraints:

- If your design requirements are different from the examples, use the flow charts and waveforms for each configuration as design guidelines.
- The design example requires a reset controller that generates a `sync_reset` (synchronous reset) for the entire system.
- The design example has an `async_reset` (a power down in GXB terms) and digital resets for transmit and receive. All user input digital resets must be at least four cycles long.
- This design example does not cover all the digital reset scenarios in a system that resets the digital logic of the GXB.
- In this example, whenever the `rx_freqlocked` signal toggles the `rxdigitalreset`, the receiver's digital circuit is reset. However, you can make changes to the design to avoid this if, for example, you want to debug your design without the core being reset.

- If you plan to use REFCLKB pins in your design, see the *REFCLKB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* for information about the effects of analog resets (pll_arest, rx_analogreset).

```
/*
 Copyright (c) Altera Corporation, 2004.
 This file may contain proprietary and confidential information of
 Altera Corporation

Contacting Altera
=================

We have made every effort to ensure that this design example works
correctly. If you have a question that is not answered by the
information, please contact Altera Support.

*****************************************************************
Reset Sequence for the ALTGXB. The configuration of GXB for which
the following
reset sequence is valid is:
    Transmit and Receive : Transmit ONLY
    Datapath    : Single Width(8/10 bits) or Double Width (16/20
bits)
    Transmit parallel clock: '-'
    Functional Mode :'Any'


*****************************************************************/


`timescale 1ns/10ps

module reset_seq_tx_ONLY (

                            inclk,

                             sync_reset,
                             async_reset,
                             transmit_digitalreset,
                             pll_locked,

                            pll_areset,
                            txdigitalreset

                            );


input inclk; //GXB input reference clock


input sync_reset;      //Input: synchronous reset from the system
input async_reset;   //Input: async reset from system
input transmit_digitalreset; //Input: Reset only the transmit
digital section
```

```
input pll_locked;      // Transmit PLL of GXB locked



output txdigitalreset; //GXB transmit digital reset
output pll_areset;//GXB power down signal


reg txdigitalreset;
reg pll_areset;
reg [1:0] state;



parameter IDLE      = 2'b00;
parameter STROBE_TXPLL_LOCKED = 2'b01;

always @ (posedge inclk or posedge async_reset) begin
if (async_reset)
    begin

        txdigitalreset <= 1'b1;
        pll_areset <= 1'b1;
        state <= STROBE_TXPLL_LOCKED;
    end
else
  case (state)
    IDLE:
                if (sync_reset)      //Synchronous Reset can be
asserted in IDLE state (After reset seq has finished)
                    begin

                        txdigitalreset <= 1'b1;
                        pll_areset <= 1'b1;
                        state   <= STROBE_TXPLL_LOCKED;

                    end
                else
                    begin

                            pll_areset <= 1'b0;
                            state   <= IDLE;
                            if(transmit_digitalreset)
                                txdigitalreset <= 1'b1;
                            else
                                txdigitalreset <= 1'b0;
                    end

    STROBE_TXPLL_LOCKED:  if (sync_reset)     //Synchronous Reset
can be asserted in IDLE state (After reset seq has finished)
                            begin

                                txdigitalreset <= 1'b1;
                                pll_areset <= 1'b1;
                                state <= STROBE_TXPLL_LOCKED;
                            end
```

```
    //Wait untill the TXPLL is locked to inclk and TX PLL has a
stable output clock which is also fed to RX CRU
                        else if (pll_locked)
                            begin
                                 state <= IDLE;
                                 txdigitalreset <= 1'b0;
                                 pll_areset <= 1'b0;
                            end
                        else
                            begin
                                state <= STROBE_TXPLL_LOCKED;
                                txdigitalreset <= 1'b1;
                                pll_areset <= 1'b0;
                            end


    default: state = IDLE;
  endcase
end

endmodule
```

Power Down

The Quartus II software automatically selects the power-down feature when you configure the Stratix GX device. All unused transceiver channels and transceiver blocks in a design are powered down to reduce the overall power consumption. The power-down feature cannot be used on the fly to turn the transceiver channels/transceiver blocks on/off without reconfiguration.

Table 9–2 details the state of the transceiver I/O pins during power-down.

Table 9–2. I/O Pin States During Power-Down (Part 1 of 2)

Operation	Transmitter Pins	Receiver Pins	REFCLKB Pins	Rref Pin
Normal operation	Transmitter	Receiver	Clk input	Ext. reference R
Power down	Tri-state *(1)*	Tri-state *(1)*	Tri-state *(2)*	Low *(3)*

Table 9–2. I/O Pin States During Power-Down (Part 2 of 2)

Operation		Transmitter Pins	Receiver Pins	REFCLKB Pins	Rref Pin
PMA loop back	Serial loop back	Tri-state *(4)* toggle	Low *(5)*	—	—
	Reverse serial loop back	Transmitter *(6)*	Receiver	—	—

Notes to Table 9–2:

(1) Either leave these pins floating or connect `n_leg` to `GXB_GND` through a 10-kΩ resistor and connect `p_leg` to `GXB_VCC` through a 10-kΩ resistor to improve the device's immunity to noise.

(2) Either leave these pins floating or connect `refclkb(+)` to `GXB_GND` through a 10-kΩ resistor and connect `refclkb(-)` to `GXB_VCC` through a 10-kΩ resistor to improve the device's immunity to noise.

(3) Altera recommends driving the reference resistor pin low for the powered down transceiver block.

(4) Transmitter output is tri-stated at the lowest V_{OD} setting and is toggling at any other setting. However, the V_{OD} is 80% of the selected V_{OD} setting.

(5) Receiver pin is pulled low internally. It must be either left floating or pulled low externally.

(6) Only the 4-mA setting is supported for reverse serial loopback.

10. Data & Control Codes

SGX52010-1.0

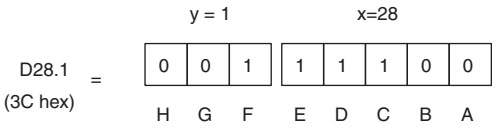
8B/10B Code

This appendix provides information about the data and control codes for the Stratix® GX device.

Code Notation

The 8B/10B data and control codes are referred to as Dx.y and Kx.y, respectively. The 8-bit byte (H G F E D C B A, where H is the MSB and A is the LSB) is broken up into 2 groups, x and y, where x is the 5 lower bits (E D C B A) and y is the upper 3 bits (H G F). Figure 10–1 shows the notation for 3C hexadecimal.

Figure 10–1. Sample Notation for 3C Hexadecimal



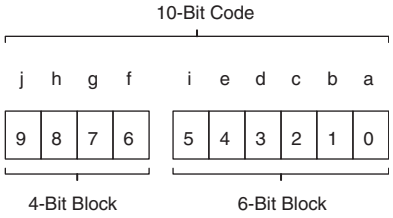
There are 256 Dx.y and 12 Kx.y valid 8-bit codes. These codes have two 10-bit equivalent codes associated with each 8-bit code. The 10-bit codes can have either a neutral disparity or a non-neutral disparity. In the case of a neutral disparity, 2 neutral disparity 10-bit codes are associated with an 8-bit code. In the case of a non-neutral disparity 10-bit code, a positive and a negative disparity code are associated with the 8-bit code.

The positive disparity 10-bit code is associated with the RD- column, and the negative disparity 10-bit code is associated with the RD+ column.

Disparity Calculation

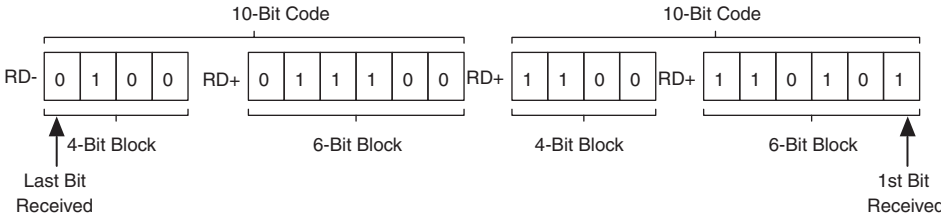
The running disparity is calculated based on the sub-blocks of the 10-bit code. The 10-bit code is divided into 2 sub blocks, a 6-bit sub-block (abcdei) and a 4-bit sub-block (fghj), as shown in Figure 10–2.

Figure 10–2. 10-Bit Grouping of 6-Bit & 4-Bit Sub-Blocks



The running disparity at the beginning of the 6-bit sub-block is the running disparity at the end of the previous 10-bit code. The running disparity of the 4-bit sub-block is the running disparity of the end of the 6-bit sub-block. The running disparity of the end of the 4-bit sub-block is the running disparity of the 10-bit code, as shown in Figure 10–3.

Figure 10–3. Running Disparity Between Sub-Blocks



The running disparity calculation rules are as follows (if the conditions are not met, then the running disparity at the end of the sub-blocks are the same as the beginning of the sub-block):

- The current running disparity at the end of a sub-block is positive if any of the following are true:
  - The sub-block contains more ones than zeros.
  - The 6-bit sub-block is `6'b000111`.
  - The 4-bit sub-block is `4'b0011`.
- The current running disparity at the end of a sub-block is negative if any of the following are true:
  - The sub-block contains more zeros than ones.
  - The 6-bit sub-block is `6'b111000`.
  - The 4-bit sub-block is `4'b1100`.

Supported Codes

The 8B/10B scheme defines the 12 control codes listed in Table 10–1 for synchronization, alignment, and general application purposes.

Table 10–1. Supported K Codes

K Code	Octal Value	8-Bit Code HGF_EDCBA	10-Bit Code RD- abcdei_fghj	10-Bit Code RD+ abcdei_fghj
K28.0	1C	8'b000_11100	10'b001111_0100	10'b110000_1011
K28.1	3C	8'b001_11100	10'b001111_1001	10'b110000_0110
K28.2	5C	8'b010_11100	10'b001111_0101	10'b110000_1010
K28.3	7C	8'b011_11100	10'b001111_0011	10'b110000_1100
K28.4	9C	8'b100_11100	10'b001111_0010	10'b110000_1101
K28.5 *(1)*	BC	8'b101_11100	10'b001111_1010	10'b110000_0101
K28.6	DC	8'b110_11100	10'b001111_0110	10'b110000_1001
K28.7	FC	8'b111_11100	10'b001111_1000	10'b110000_0111
K23.7	F7	8'b111_10111	10'b111010_1000	10'b000101_0111
K27.7	FB	8'b111_11011	10'b110110_1000	10'b001001_0111
K29.7	FD	8'b111_11101	10'b101110_1000	10'b010001_0111
K30.7	FE	8'b111_11110	10'b011110_1000	10'b100001_0111

Note to Table 10–1:

(1) K28.5 is a comma code used for word alignment and indicates an IDLE state.

Table 10–2 shows the valid data code-groups.

Table 10–2. Valid Data Code-Groups (Part 1 of 9)

Code-group Name	Octet Value	Octet Bits HGF EDCBA	Current RD-	Current RD+
			abcdei fghj	abcdei fghj
D0.0	00	000 00000	100111 0100	011000 1011
D1.0	01	000 00001	011101 0100	100010 1011
D2.0	02	000 00010	101101 0100	010010 1011
D3.0	03	000 00011	110001 1011	110001 0100
D4.0	04	000 00100	110101 0100	001010 1011
D5.0	05	000 00101	101001 1011	101001 0100
D6.0	06	000 00110	011001 1011	011001 0100
D7.0	07	000 00111	111000 1011	000111 0100
D8.0	08	000 01000	111001 0100	000110 1011

Table 10–2. Valid Data Code-Groups (Part 2 of 9)

Code-group Name	Octet Value	Octet Bits HGF EDCBA	Current RD-	Current RD+
			abcdei fghj	abcdei fghj
D9.0	09	000 01001	100101 1011	100101 0100
D10.0	0A	000 01010	010101 1011	010101 0100
D11.0	0B	000 01011	110100 1011	110100 0100
D12.0	0C	000 01100	001101 1011	001101 0100
D13.0	0D	000 01101	101100 1011	101100 0100
D14.0	0E	000 01110	011100 1011	011100 0100
D15.0	0F	000 01111	010111 0100	101000 1011
D16.0	10	000 10000	011011 0100	100100 1011
D17.0	11	000 10001	100011 1011	100011 0100
D18.0	12	000 10010	010011 1011	010011 0100
D19.0	13	000 10011	110010 1011	110010 0100
D20.0	14	000 10100	001011 1011	001011 0100
D21.0	15	000 10101	101010 1011	101010 0100
D22.0	16	000 10110	011010 1011	011010 0100
D23.0	17	000 10111	111010 0100	000101 1011
D24.0	18	000 11000	110011 0100	001100 1011
D25.0	19	000 11001	100110 1011	100110 0100
D26.0	1A	000 11010	010110 1011	010110 0100
D27.0	1B	000 11011	110110 0100	001001 1011
D28.0	1C	000 11100	001110 1011	001110 0100
D29.0	1D	000 11101	101110 0100	010001 1011
D30.0	1E	000 11110	011110 0100	100001 1011
D31.0	1F	000 11111	101011 0100	010100 1011
D0.1	20	001 00000	100111 1001	011000 1001
D1.1	21	001 00001	011101 1001	100010 1001
D2.1	22	001 00010	101101 1001	010010 1001
D3.1	23	001 00011	110001 1001	110001 1001
D4.1	24	001 00100	110101 1001	001010 1001
D5.1	25	001 00101	101001 1001	101001 1001
D6.1	26	001 00110	011001 1001	011001 1001
D7.1	27	001 00111	111000 1001	000111 1001
D8.1	28	001 01000	111001 1001	000110 1001
D9.1	29	001 01001	100101 1001	100101 1001

Table 10–2. Valid Data Code-Groups (Part 3 of 9)

Code-group Name	Octet Value	Octet Bits HGF EDCBA	Current RD-	Current RD+
			abcdei fghj	abcdei fghj
D10.1	2A	001 01010	010101 1001	010101 1001
D11.1	2B	001 01011	110100 1001	110100 1001
D12.1	2C	001 01100	001101 1001	001101 1001
D13.1	2D	001 01101	101100 1001	101100 1001
D14.1	2E	001 01110	011100 1001	011100 1001
D15.1	2F	001 01111	010111 1001	101000 1001
D16.1	30	001 10000	011011 1001	100100 1001
D17.1	31	001 10001	100011 1001	100011 1001
D18.1	32	001 10010	010011 1001	010011 1001
D19.1	33	001 10011	110010 1001	110010 1001
D20.1	34	001 10100	001011 1001	001011 1001
D21.1	35	001 10101	101010 1001	101010 1001
D22.1	36	001 10110	011010 1001	011010 1001
D23.1	37	001 10111	111010 1001	000101 1001
D24.1	38	001 11000	110011 1001	001100 1001
D25.1	39	001 11001	100110 1001	100110 1001
D26.1	3A	001 11010	010110 1001	010110 1001
D27.1	3B	001 11011	110110 1001	001001 1001
D28.1	3C	001 11100	001110 1001	001110 1001
D29.1	3D	001 11101	101110 1001	010001 1001
D30.1	3E	001 11110	011110 1001	100001 1001
D31.1	3F	001 11111	101011 1001	010100 1001
D0.2	40	010 00000	100111 0101	011000 0101
D1.2	41	010 00001	011101 0101	100010 0101
D2.2	42	010 00010	101101 0101	010010 0101
D3.2	43	010 00011	110001 0101	110001 0101
D4.2	44	010 00100	110101 0101	001010 0101
D5.2	45	010 00101	101001 0101	101001 0101
D6.2	46	010 00110	011001 0101	011001 0101
D7.2	47	010 00111	111000 0101	000111 0101
D8.2	48	010 01000	111001 0101	000110 0101
D9.2	49	010 01001	100101 0101	100101 0101
D10.2	4A	010 01010	010101 0101	010101 0101

Table 10–2. Valid Data Code-Groups (Part 4 of 9)

Code-group Name	Octet Value	Octet Bits HGF EDCBA	Current RD-	Current RD+
			abcdei fghj	abcdei fghj
D11.2	4B	010 01011	110100 0101	110100 0101
D12.2	4C	010 01100	001101 0101	001101 0101
D13.2	4D	010 01101	101100 0101	101100 0101
D14.2	4E	010 01110	011100 0101	011100 0101
D15.2	4F	010 01111	010111 0101	101000 0101
D16.2	50	010 10000	011011 0101	100100 0101
D17.2	51	010 10001	100011 0101	100011 0101
D18.2	52	010 10010	010011 0101	010011 0101
D19.2	53	010 10011	110010 0101	110010 0101
D20.2	54	010 10100	001011 0101	001011 0101
D21.2	55	010 10101	101010 0101	101010 0101
D22.2	56	010 10110	011010 0101	011010 0101
D23.2	57	010 10111	111010 0101	000101 0101
D24.2	58	010 11000	110011 0101	001100 0101
D25.2	59	010 11001	100110 0101	100110 0101
D26.2	5A	010 11010	010110 0101	010110 0101
D27.2	5B	010 11011	110110 0101	001001 0101
D28.2	5C	010 11100	001110 0101	001110 0101
D29.2	5D	010 11101	101110 0101	010001 0101
D30.2	5E	010 11110	011110 0101	100001 0101
D31.2	5F	010 11111	101011 0101	010100 0101
D0.3	60	011 00000	100111 0011	011000 1100
D1.3	61	011 00001	011101 0011	100010 1100
D2.3	62	011 00010	101101 0011	010010 1100
D3.3	63	011 00011	110001 1100	110001 0011
D4.3	64	011 00100	110101 0011	001010 1100
D5.3	65	011 00101	101001 1100	101001 0011
D6.3	66	011 00110	011001 1100	011001 0011
D7.3	67	011 00111	111000 1100	000111 0011
D8.3	68	011 01000	111001 0011	000110 1100
D9.3	69	011 01001	100101 1100	100101 0011
D10.3	6A	011 01010	010101 1100	010101 0011
D11.3	6B	011 01011	110100 1100	110100 0011

Table 10–2. Valid Data Code-Groups (Part 5 of 9)

Code-group Name	Octet Value	Octet Bits HGF EDCBA	Current RD-	Current RD+
			abcdei fghj	abcdei fghj
D12.3	6C	011 01100	001101 1100	001101 0011
D13.3	6D	011 01101	101100 1100	101100 0011
D14.3	6E	011 01110	011100 1100	011100 0011
D15.3	6F	011 01111	010111 0011	101000 1100
D16.3	70	011 10000	011011 0011	100100 1100
D17.3	71	011 10001	100011 1100	100011 0011
D18.3	72	011 10010	010011 1100	010011 0011
D19.3	73	011 10011	110010 1100	110010 0011
D20.3	74	011 10100	001011 1100	001011 0011
D21.3	75	011 10101	101010 1100	101010 0011
D22.3	76	011 10110	011010 1100	011010 0011
D23.3	77	011 10111	111010 0011	000101 1100
D24.3	78	011 11000	110011 0011	001100 1100
D25.3	79	011 11001	100110 1100	100110 0011
D26.3	7A	011 11010	010110 1100	010110 0011
D27.3	7B	011 11011	110110 0011	001001 1100
D28.3	7C	011 11100	001110 1100	001110 0011
D29.3	7D	011 11101	101110 0011	010001 1100
D30.3	7E	011 11110	011110 0011	100001 1100
D31.3	7F	011 11111	101011 0011	010100 1100
D0.4	80	100 00000	100111 0010	011000 1101
D1.4	81	100 00001	011101 0010	100010 1101
D2.4	82	100 00010	101101 0010	010010 1101
D3.4	83	100 00011	110001 1101	110001 0010
D4.4	84	100 00100	110101 0010	001010 1101
D5.4	85	100 00101	101001 1101	101001 0010
D6.4	86	100 00110	011001 1101	011001 0010
D7.4	87	100 00111	111000 1101	000111 0010
D8.4	88	100 01000	111001 0010	000110 1101
D9.4	89	100 01001	100101 1101	100101 0010
D10.4	8A	100 01010	010101 1101	010101 0010
D11.4	8B	100 01011	110100 1101	110100 0010
D12.4	8C	100 01100	001101 1101	001101 0010

Table 10–2. Valid Data Code-Groups (Part 6 of 9)

Code-group Name	Octet Value	Octet Bits HGF EDCBA	Current RD-	Current RD+
			abcdei fghj	abcdei fghj
D13.4	8D	100 01101	101100 1101	101100 0010
D14.4	8E	100 01110	011100 1101	011100 0010
D15.4	8F	100 01111	010111 0010	101000 1101
D16.4	90	100 10000	011011 0010	100100 1101
D17.4	91	100 10001	100011 1101	100011 0010
D18.4	92	100 10010	010011 1101	010011 0010
D19.4	93	100 10011	110010 1101	110010 0010
D20.4	94	100 10100	001011 1101	001011 0010
D21.4	95	100 10101	101010 1101	101010 0010
D22.4	96	100 10110	011010 1101	011010 0010
D23.4	97	100 10111	111010 0010	000101 1101
D24.4	98	100 11000	110011 0010	001100 1101
D25.4	99	100 11001	100110 1101	100110 0010
D26.4	9A	100 11010	010110 1101	010110 0010
D27.4	9B	100 11011	110110 0010	001001 1101
D28.4	9C	100 11100	001110 1101	001110 0010
D29.4	9D	100 11101	101110 0010	010001 1101
D30.4	9E	100 11110	011110 0010	100001 1101
D31.4	9F	100 11111	101011 0010	010100 1101
D0.5	A0	101 00000	100111 1010	011000 1010
D1.5	A1	101 00001	011101 1010	100010 1010
D2.5	A2	101 00010	101101 1010	010010 1010
D3.5	A3	101 00011	110001 1010	110001 1010
D4.5	A4	101 00100	110101 1010	001010 1010
D5.5	A5	101 00101	101001 1010	101001 1010
D6.5	A6	101 00110	011001 1010	011001 1010
D7.5	A7	101 00111	111000 1010	000111 1010
D8.5	A8	101 01000	111001 1010	000110 1010
D9.5	A9	101 01001	100101 1010	100101 1010
D10.5	AA	101 01010	010101 1010	010101 1010
D11.5	AB	101 01011	110100 1010	110100 1010
D12.5	AC	101 01100	001101 1010	001101 1010
D13.5	AD	101 01101	101100 1010	101100 1010

Table 10–2. Valid Data Code-Groups (Part 7 of 9)

Code-group Name	Octet Value	Octet Bits HGF EDCBA	Current RD-	Current RD+
			abcdei fghj	abcdei fghj
D14.5	AE	101 01110	011100 1010	011100 1010
D15.5	AF	101 01111	010111 1010	101000 1010
D16.5	B0	101 10000	011011 1010	100100 1010
D17.5	B1	101 10001	100011 1010	100011 1010
D18.5	B2	101 10010	010011 1010	010011 1010
D19.5	B3	101 10011	110010 1010	110010 1010
D20.5	B4	101 10100	001011 1010	001011 1010
D21.5	B5	101 10101	101010 1010	101010 1010
D22.5	B6	101 10110	011010 1010	011010 1010
D23.5	B7	101 10111	111010 1010	000101 1010
D24.5	B8	101 11000	110011 1010	001100 1010
D25.5	B9	101 11001	100110 1010	100110 1010
D26.5	BA	101 11010	010110 1010	010110 1010
D27.5	BB	101 11011	110110 1010	001001 1010
D28.5	BC	101 11100	001110 1010	001110 1010
D29.5	BD	101 11101	101110 1010	010001 1010
D30.5	BE	101 11110	011110 1010	100001 1010
D31.5	BF	101 11111	101011 1010	010100 1010
D0.6	C0	110 00000	100111 0110	011000 0110
D1.6	C1	110 00001	011101 0110	100010 0110
D2.6	C2	110 00010	101101 0110	010010 0110
D3.6	C3	110 00011	110001 0110	110001 0110
D4.6	C4	110 00100	110101 0110	001010 0110
D5.6	C5	110 00101	101001 0110	101001 0110
D6.6	C6	110 00110	011001 0110	011001 0110
D7.6	C7	110 00111	111000 0110	000111 0110
D8.6	C8	110 01000	111001 0110	000110 0110
D9.6	C9	110 01001	100101 0110	100101 0110
D10.6	CA	110 01010	010101 0110	010101 0110
D11.6	CB	110 01011	110100 0110	110100 0110
D12.6	CC	110 01100	001101 0110	001101 0110
D13.6	CD	110 01101	101100 0110	101100 0110
D14.6	CE	110 01110	011100 0110	011100 0110

Table 10–2. Valid Data Code-Groups (Part 8 of 9)

Code-group Name	Octet Value	Octet Bits HGF EDCBA	Current RD-	Current RD+
			abcdei fghj	abcdei fghj
D15.6	CF	110 01111	010111 0110	101000 0110
D16.6	D0	110 10000	011011 0110	100100 0110
D17.6	D1	110 10001	100011 0110	100011 0110
D18.6	D2	110 10010	010011 0110	010011 0110
D19.6	D3	110 10011	110010 0110	110010 0110
D20.6	D4	110 10100	001011 0110	001011 0110
D21.6	D5	110 10101	101010 0110	101010 0110
D22.6	D6	110 10110	011010 0110	011010 0110
D23.6	D7	110 10111	111010 0110	000101 0110
D24.6	D8	110 11000	110011 0110	001100 0110
D25.6	D9	110 11001	100110 0110	100110 0110
D26.6	DA	110 11010	010110 0110	010110 0110
D27.6	DB	110 11011	110110 0110	001001 0110
D28.6	DC	110 11100	001110 0110	001110 0110
D29.6	DD	110 11101	101110 0110	010001 0110
D30.6	DE	110 11110	011110 0110	100001 0110
D31.6	DF	110 11111	101011 0110	010100 0110
D0.7	E0	111 00000	100111 0001	011000 1110
D1.7	E1	111 00001	011101 0001	100010 1110
D2.7	E2	111 00010	101101 0001	010010 1110
D3.7	E3	111 00011	110001 1110	110001 0001
D4.7	E4	111 00100	110101 0001	001010 1110
D5.7	E5	111 00101	101001 1110	101001 0001
D6.7	E6	111 00110	011001 1110	011001 0001
D7.7	E7	111 00111	111000 1110	000111 0001
D8.7	E8	111 01000	111001 0001	000110 1110
D9.7	E9	111 01001	100101 1110	100101 0001
D10.7	EA	111 01010	010101 1110	010101 0001
D11.7	EB	111 01011	110100 1110	110100 1000
D12.7	EC	111 01100	001101 1110	001101 0001
D13.7	ED	111 01101	101100 1110	101100 1000
D14.7	EE	111 01110	011100 1110	011100 1000
D15.7	EF	111 01111	010111 0001	101000 1110

Table 10–2. Valid Data Code-Groups (Part 9 of 9)

Code-group Name	Octet Value	Octet Bits HGF EDCBA	Current RD-	Current RD+
			abcdei fghj	abcdei fghj
D16.7	F0	111 10000	011011 0001	100100 1110
D17.7	F1	111 10001	100011 0111	100011 0001
D18.7	F2	111 10010	010011 0111	010011 0001
D19.7	F3	111 10011	110010 1110	110010 0001
D20.7	F4	111 10100	001011 0111	001011 0001
D21.7	F5	111 10101	101010 1110	101010 0001
D22.7	F6	111 10110	011010 1110	011010 0001
D23.7	F7	111 10111	111010 0001	000101 1110
D24.7	F8	111 11000	110011 0001	001100 1110
D25.7	F9	111 11001	100110 1110	100110 0001
D26.7	FA	111 11010	010110 1110	010110 0001
D27.7	FB	111 11011	110110 0001	001001 1110
D28.7	FC	111 11100	001110 1110	001110 0001
D29.7	FD	111 11101	101110 0001	010001 1110
D30.7	FE	111 11110	011110 0001	100001 1110
D31.7	FF	111 11111	101011 0001	010100 1110

11. Ports & Parameters

SGX52011-1.1

Input Ports

Table 11–1 lists the input ports of the Stratix® GX device.

Table 11–1. Input Ports (Part 1 of 5)

Port Name	Required	Description	Comments
inclk[]	See comments	Transceiver block transmitter PLL reference input clock.	Input port [NUMBER_OF_QUADS - 1..0] wide. If you use the transmitter PLL, the inclk[] port is required. If you set the OPERATION_MODE parameter to TX or DUPLEX, the inclk[] port is required.
pll_areset[]	No	Asynchronous reset for the transceiver block transmitter PLL. This signal powers down the entire transceiver block. When placing refclkb pins, see the *REFCLKB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* for information about analog reads and refclkb pin usage constraints.	Input port [NUMBER_OF_QUADS - 1..0] wide.
rx_in[]	Yes	Transceiver block receiver channel data input port.	Input port [NUMBER_OF_CHANNELS - 1..0] wide.
rx_cruclk[]	No	Clock recovery unit (CRU) for the transceiver block receiver PLL reference input clock.	Input port [NUMBER_OF_QUADS - 1..0] wide. When you set the OPERATION_MODE parameter to TX or DUPLEX, the rx_cruclk[] port cannot be used. If you use this parameter, the transceiver block transmitter PLL cannot be instantiated.

Table 11–1. Input Ports (Part 2 of 5)

Port Name	Required	Description	Comments
rx_bitslip[]	No	Controls bit slipping circuitry in the word aligner.	Input port [NUMBER_OF_CHANNELS - 1..0] wide. If you enable the rx_bitslip port, the rx_enacdet[] port cannot be connected and the USE_AUTO_BIT_SLIP parameter must be set to OFF.
rx_enacdet[]	No	Enables alignment to the programmed pattern.	Input port [NUMBER_OF_CHANNELS - 1..0] wide. If you enable the rx_enacdet port, the rx_bitslip[] port cannot be connected, and the USE_AUTO_BIT_SLIP parameter must be set to ON.
rx_slpbk[]	No	Serial loopback input. Dynamically enables serial loopback from the transceiver block transmitter to the transceiver block receiver in the same channel.	Input port [NUMBER_OF_CHANNELS - 1..0] wide. If you enable the rx_slpbk[] input port, the OPERATION_MODE parameter must be set to DUPLEX, and the serialfdbk port of the transceiver block receiver channel must be connected.
rx_a1a2size[]	No	Detects A1A2 or A1A1A2A2 input patterns. If the signal is low (0), A1A2 patterns are detected. If the signal is high (1), A1A1A2A2 patterns are detected.	Input port [NUMBER_OF_CHANNELS - 1..0] wide. If you enable the rx_a1a2size[] port, the PROTOCOL parameter must be set to SONET.
rx_equalizerctrl[]	No	Specifies the equalizer control setting.	Input port [NUMBER_OF_CHANNELS * 3..0] wide. Use the following settings: **Incoming Signal** / **Equalizer Control Setting** 000 — 0 001 — Reserved 010 — 1 011 — Reserved 100 — 2 101 — 3 110 — Reserved 111 — 4

Table 11–1. Input Ports (Part 3 of 5)

Port Name	Required	Description	Comments
rx_locktorefclk[]	No	Control signal for transceiver block receiver PLL to lock to the reference clock.	Input port [NUMBER_OF_CHANNELS - 1..0] wide.
rx_locktodata[]	No	Control signal for transceiver block receiver PLL to lock the received data.	Input port [NUMBER_OF_CHANNELS - 1..0] wide. The rx_locktodata[] port can overwrite the rx_locktorefclk[] port.
tx_in[]	Yes	Transceiver block transmitter channel data input port.	Input port [CHANNEL_WIDTH * NUMBER_OF_CHANNELS - 1..0] wide. If you set the USE_8B_10B_MODE parameter to OFF and the USE_DOUBLE_DATA_MODE parameter is set to OFF, the deserialization factor is CHANNEL_WIDTH. If you set the USE_8B_10B_MODE parameter to OFF and the USE_DOUBLE_DATA_MODE parameter is set to ON, the deserialization factor is CHANNEL_WIDTH / 2. If you set the USE_8B_10B_MODE parameter to ON, the deserialization factor is 10.
tx_ctrlenable[]	No	Control character enable. Enables the 8B/10B encoder to identify control characters. Labels an input character as a control code.	Input port [NUMBER_OF_CHANNELS * DWIDTH_FACTOR - 1..0] wide. If tx_ctrlenable[] port is high, the sent word is a control character.
tx_forcedisparity[]	No	Force disparity. When asserted HIGH after coming out of reset, forces the 8B/10B encoder to encode only the first byte after assertion with negative disparity.	Reassertion has no affect on the disparity unless the device is reset.

Table 11–1. Input Ports (Part 4 of 5)

Port Name	Required	Description	Comments
tx_srlpbk[]	No	Reverse serial loopback input. Dynamically enables reverse serial loopback from the rx_in[] port to the tx_out[] port	Input port [NUMBER_OF_CHANNELS - 1..0] wide.
tx_vodctrl[]	No	3-bit control signal that dynamically specifies the Voltage Output Differential (VOD) control settings.	Input port [NUMBER_OF_CHANNELS * 3 - 1..0] wide. Use the following settings: **Incoming Signal / VOD Control Setting** 000 400 mV 001 800 mV 010 1,000 mV 011 1,200 mV 100 1,400 mV 101 1,600 mV 110 1,600 mV 111 1,600 mV
tx_preemphasisctrl[]	No	3-bit control signal that dynamically specifies the pre-emphasis settings.	Input port [NUMBER_OF_CHANNELS * 3 - 1..0] wide. Use the following settings: **Incoming Signal / Pre-emphasis Setting** 000 0 001 1 010 2 011 3 100 4 101 5 110 5 111 5
txdigitalreset[]	No	Sends a reset signal to the digital portion of the transmitter.	Input port [NUMBER_OF_QUADS * 4 - 1..0] wide.
rxdigitalreset[]	No	Sends a reset signal to the digital portion of the receiver.	Input port [NUMBER_OF_QUADS * 4 - 1..0] wide.
rxanalogreset[]	No	Sends a power down signal to the analog portion of the receiver.	Input port [NUMBER_OF_QUADS * 4 - 1..0] wide.

Table 11–1. Input Ports (Part 5 of 5)

Port Name	Required	Description	Comments
pllenable[]	No	Sends an enable signal to the transceiver block transmitter PLL.	Input port [NUMBER_OF_QUADS - 1..0] wide.
pll_areset[]	No	Sends a power down signal to the transceiver block transmitter PLL.	Input port [NUMBER_OF_QUADS - 1..0] wide.

Output Ports

Table 11–2 lists the output ports of the Stratix GX device.

Table 11–2. Output Ports (Part 1 of 4)

Port Name	Required	Description	Comments
pll_locked[]	No	Gives the status of the transceiver block transmitter PLL.	Output port [NUMBER_OF_QUADS - 1..0] wide. The pll_locked port is available only when the transceiver block transmitter PLL is used. The signal achieves lock status within several clock cycles in simulation. This does not necessarily reflect the real lock time in the hardware, which can take thousands of cycles for some settings.
coreclk_out[]	No	Output clock fed by the clk2 port of the transceiver block transmitter PLL.	Output port [NUMBER_OF_QUADS - 1..0] wide. If a transceiver block transmitter PLL is used, the coreclk_out port must be enabled.
rx_out[]	Yes	Transceiver block receiver PLL output data.	Output port [CHANNEL_WIDTH * NUMBER_OF_CHANNELS - 1..0] wide. If you set the USE_8B_10B_MODE parameter to OFF and the USE_DOUBLE_DATA_MODE parameter is set to OFF, the deserialization factor is CHANNEL_WIDTH. If you set the USE_8B_10B_MODE parameter to OFF and the USE_DOUBLE_DATA_MODE parameter is set to ON, the deserialization factor is CHANNEL_WIDTH / 2. If you set the USE_8B_10B_MODE parameter to ON, the deserialization factor is 10.

Table 11–2. Output Ports (Part 2 of 4)

Port Name	Required	Description	Comments
rx_clkout[]	No	Output clock from the transceiver block receiver channel.	Output port [NUMBER_OF_CHANNELS - 1..0] wide. If you set the USE_RATE_MATCH_FIFO parameter to ON and the CLK_OUT_MODE_REFERENCE parameter is set to ON, the rx_clkout[] port is the PLL reference clock with the period PLL_INCLOCK_PERIOD / DATA_RATE) * CHANNEL_WIDTH. If you set the USE_RATE_MATCH_FIFO parameter to ON and the CLK_OUT_MODE_REFERENCE parameter is set to OFF, the rx_clkout[] port is the clock output of the PLL. If you set the USE_RATE_MATCH_FIFO parameter is OFF, the value of the rx_clkout[] port is the same as the value of the rx_recovclockout[] port. If you set the USE_DOUBLE_DATA_MODE parameter OFF, the clock period must be doubled, or the clock frequency must be halved.
rx_locked[]	No	Gives the status of the transceiver block receiver channel atom.	Output port [NUMBER_OF_CHANNELS - 1..0] wide. Indicates that the transceiver block receiver PLL is locked to the reference input clock (active low). When the transceiver block receiver PLL is locked, this signal is GND. When the transceiver block receiver PLL is out of lock, this signal is VCC. The signal achieves lock status within several clock cycles in simulation. This does not necessarily reflect the real lock time in hardware, which can take thousands of cycles for some settings.
rx_channelaligned[]	Yes	Channel alignment status for the transceiver block receiver channels.	Output port [NUMBER_OF_QUADS - 1..0] wide. If the PROTOCOL parameter is set to XAUI, the rx_channelaligned[] port must be connected.

Table 11–2. Output Ports (Part 3 of 4)

Port Name	Required	Description	Comments
rx_freqlocked[]	No	Indicates whether transceiver block receiver channel is locked to the data mode in the rx_in[] port.	Output port [NUMBER_OF_CHANNELS - 1..0] wide. This port is asserted when the GXB receiver PLL moves to lock to the received data mode. At that time, the clock recovery unit (CRU) for the GXB receiver PLL is frequency and phase-locked to the reference clock and starts using the phase detector to lock onto the incoming data, but it is not yet locked to the data. It takes a finite time before the CRU for the GXB receiver PLL is locked onto the data. The signal achieves lock status within several clock cycles in simulation. This does not necessarily reflect the real lock time in hardware, which can take thousands of cycles for some settings.
rx_rlv[]	No	Indicates whether the transceiver block receiver channel violated the value specified for the RUN_LENGTH parameter.	Output port [NUMBER_OF_CHANNELS - 1..0] wide.
rx_syncstatus[]	No	Provides the status of the pattern detector and word aligner.	Output port [NUMBER_OF_CHANNELS * DWIDTH_FACTOR - 1..0] wide. If you set the PROTOCOL parameter to XAUI or GIGE, the rx_syncstatus[] port is connected to the synchronization state machine and indicates that the channel completed synchronization. If you set the PROTOCOL parameter to anything other than XAUI or GIGE, the rx_syncstatus[] port becomes a resync signal for manual synchronization of the alignment system.
rx_patterndetect[]	No	Indicates whether the pattern detector detected the programmed pattern.	Output port [NUMBER_OF_CHANNELS * DWIDTH_FACTOR - 1..0] wide.
rx_ctrldetect[]	No	Indicates whether the 8B/10B decoder detects a control code.	Output port [NUMBER_OF_CHANNELS * DWIDTH_FACTOR - 1..0] wide. If you set the USE_8B_10B_MODE parameter to OFF, the rx_ctrldetect port is not available.

Table 11–2. Output Ports (Part 4 of 4)

Port Name	Required	Description	Comments
rx_errdetect[]	No	Indicates whether the 8B/10B decoder detects an error code.	Output port [NUMBER_OF_CHANNELS * DWIDTH_FACTOR - 1..0] wide. If you set the USE_8B_10B_MODE parameter to OFF, the rx_errdetect port is not available.
rx_disperr[]	No	Indicates whether the 8B/10B decoder detects a disparity error.	Output port [NUMBER_OF_CHANNELS * DWIDTH_FACTOR - 1..0] wide.
rx_signaldetect[]	No	Indicates whether there is a legal voltage level on the input buffer.	Output port [NUMBER_OF_CHANNELS - 1..0] wide. This port is available only if the PROTOCOL parameter is set to XAUI or GIGE. This signal is always set in the modes in which it is enabled. It is still enabled even after the signal is a forced HIGH for backward compatibility with existing designs.
rx_bisterr[]	No	Error status signal for the self test.	Output port [NUMBER_OF_CHANNELS - 1..0] wide. This port is available only if the USE_SELF_TEST_MODE parameter is turned on.
rx_bistdone[]	No	Indicates whether the self test is complete.	Output port [NUMBER_OF_CHANNELS - 1..0] wide. This port is available only if the USE_SELF_TEST_MODE parameter is turned on.
rx_a1a2sizeout[]	No	Reports the a1a2size signal as seen by the word aligner.	Output port [NUMBER_OF_CHANNELS * DWIDTH_FACTOR - 1..0] wide.
tx_out[]	Yes	Serialized transceiver block transmitter channel data signal.	Output port [NUMBER_OF_CHANNELS - 1..0] wide.

Parameter Descriptions

Table 11–3 describes the Stratix GX device parameters.

Table 11–3. Parameter Descriptions (Part 1 of 6)

Parameter	Type	Required	Comments
OPERATION_MODE	String	Yes	Specifies the operation of the transceiver block transmitter PLL and transceiver block receiver PLL. Values are RX, TX, and DUPLEX. If the PROTOCOL parameter is set to XAUI, then you must set the OPERATION_MODE parameter to DUPLEX.
LOOPBACK_MODE	String	No	Specifies the operation of the loopback. Values are NONE, SLB, and PLB. If omitted, the default is NONE. Values other than NONE are available only when the OPERATION_MODE parameter is set to DUPLEX.
REVERSE_LOOPBACK _MODE	String	No	Specifies the operation of the reverse loopback. Values are NONE, and RSLB. If omitted, the default is NONE. Values other than NONE are available only when the OPERATION_MODE parameter is set to DUPLEX.
PROTOCOL	String	Yes	Specifies the protocol. Values are XAUI, SONET, GIGE, and Custom.
NUMBER_OF_CHANNELS	Integer	Yes	Specifies the number of transceiver block receiver or transmitter channels. Values are 1 through 20.
NUMBER_OF_QUADS	Integer	Yes	Specifies the number of transceiver blocks.
CHANNEL_WIDTH	Integer	Yes	Specifies the width of the dataout signal from the transceiver block receiver or transmitter channel atom. Use the following settings: **Setting of PROTOCOL Parameter** / **Channel Width Values** XAUI: 16 SONET: 8 and 16 GIGE: 8 Custom: 8, 10, 16, and 20 For more information, see the table in the comments for the DATA_RATE parameter.

Table 11–3. Parameter Descriptions (Part 2 of 6)

Parameter	Type	Required	Comments
PLL_INCLOCK_PERIOD	Integer	Yes	Specifies, in picoseconds (ps), the period or frequency of the transceiver block transmitter PLL. If omitted, the default is 0. The value of this parameter is used to compute the input clock frequency in MHz; (1 / PLL_INCLOCK_PERIOD) * 1,000,000. When you specify PLL_INCLOCK_PERIOD, the CRU_INCLOCK_PERIOD parameter cannot be used. For more information, see the table in the comments for the DATA_RATE parameter.
DATA_RATE	Integer	Yes	Specifies, in Mbps, the rate of data from the transceiver block transmitter channel and the transceiver block receiver channel. If omitted, the default is 0.
DATA_RATE_REMAINDER	Integer	No	Specifies, in bits per second (bps), the remainder of the DATA_RATE parameter; DATA_RATE * 1,000,000. This parameter helps to specify non-integral data rates. If omitted, the default is 0.
USE_DOUBLE_DATA_MODE	String	No	Specifies whether to use double data width mode. If you enable this parameter, the CHANNEL_WIDTH parameter value is 16 or 20. When the CHANNEL_WIDTH parameter value is 8 or 10, the transceiver block receiver channel is not in double data width mode. Values are ON and OFF. If omitted, the default is OFF.
USE_8B_10B_MODE	String	No	Specifies whether to use the 8B/10B decoder. If this parameter is turned on, the deserialization factor is 10, and the CHANNEL_WIDTH parameter value is 8 or 16. Values are ON and OFF. If omitted, the default is OFF.
DWIDTH_FACTOR	Integer	No	Specifies the width of the double data factor. Values are 1 and 2. If omitted, the default is 1. A value of 1 means that value of the USE_DOUBLE_DATA_MODE parameter is OFF, and a value of 2 means that value of the USE_DOUBLE_DATA_MODE parameter is ON.
CRU_INCLOCK_PERIOD	Integer	No	Specifies, in picoseconds (ps), the period or frequency of the transceiver block receiver PLL. If omitted, the default is 0. The value of this parameter computes the input clock frequency in MHz; (1 / CRU_INCLOCK_PERIOD) * 1,000,000. When you specify the CRU_INCLOCK_PERIOD, the PLL_INCLOCK_PERIOD parameter cannot be used. For more information, see the table in the comments for the DATA_RATE parameter.

Table 11–3. Parameter Descriptions (Part 3 of 6)

Parameter	Type	Required	Comments
RUN_LENGTH	Integer	No	Specifies the maximum run length supported for the incoming data signal. This parameter is ignored if the RUN_LENGTH_ENABLE parameter is turned off. If the deserialization factor is 8, legal values are 4–128, in multiples of four. If the deserialization factor is 10, legal values are 5–160, in multiples of five. If omitted, the default is 0.
RUN_LENGTH_ENABLE	String	No	Specifies whether to use run length detection. Values are ON and OFF. If omitted, the default is OFF.
USE_CHANNEL_ALIGN	String	No	Specifies whether to use the channel aligner and enable the deskew system. The USE_CHANNEL_ALIGN parameter can only be used when the PROTOCOL parameter is set to XAUI. Values are ON and OFF. If omitted, the default is OFF.
USE_AUTO_BIT_SLIP	String	No	Specifies whether to use internal bit slipping circuitry. If you enable this parameter, the bitslip transceiver block receiver channel is ignored, and an internal register controls the byte alignment functions. If this parameter is turned off, the bit slipping operation is controlled by the rx_bitslip[] signal. Values are ON and OFF. If omitted, the default is OFF.
USE_SYMBOL_ALIGN	String	No	Specifies whether to use the word aligner. Values are ON and OFF. If omitted, the default is ON.
ALIGN_PATTERN	String	No	Specifies the pattern of 7, 10, or 16 bits used by the word aligner for the USE_SYMBOL_ALIGN parameter. If the USE_SYMBOL_ALIGN parameter is turned off, this parameter is ignored. If the PROTOCOL parameter is set to XAUI or GIGE, the only legal pattern is 0101111100.
ALIGN_PATTERN_LENGTH	Integer	No	Specifies the length of the ALIGN_PATTERN parameter. Values are 7, 10, or 16. If omitted, the default is 0.
CLK_OUT_MODE_ REFERENCE	String	No	Specifies whether to use the clock that operates the post rate matching FIFO module of the transceiver block receiver channel. This parameter is ignored if the USE_RATE_MATCH_FIFO parameter is turned off. Values are ON and OFF. If omitted, the default is OFF.
USE_SELF_TEST_MODE	String	No	Indicates whether to use the built-in self test mode. Values are ON and OFF. If omitted, the default is OFF.

Table 11–3. Parameter Descriptions (Part 4 of 6)

Parameter	Type	Required	Comments
SELF_TEST_MODE	Integer	No	Indicates which self test mode to use. Values are 0, 1, 2, 3, or 4. If omitted, the default is 0. This parameter is ignored if the USE_SELF_TEST_MODE parameter is turned off. Use the following settings: **Value** **Test** 0 PRBS 1 00-FF, followed by 5 K28.5+ and repeat 2 High frequency pattern, repeat of D21.5- D21.5+ 3 Low frequency pattern, repeat of K28.7- K28.7+ 4 Mixed frequency pattern, repeat of K28.5- K28.5+
USE_EQUALIZER_CTRL_SIGNAL	String	No	Specifies whether to use the equalizer control signal. Values are ON and OFF. If omitted, the default is OFF.
EQUALIZER_CTRL_SETTING	Integer	No	Specifies the magnitude of the equalizer control settings. Refer to the Quartus® II help menu for more information regarding the variable values. This parameter should be specified if the USE_EQUALIZER_CTRL_SIGNAL parameter is turned off.
SIGNAL_LOSS_THRESHOLD_SELECT	Integer	No	Specifies the signal loss threshold. Refer to the Quartus II software Help menu for more information about the variable values.
PLL_BANDWIDTH_TYPE	String	No	Specifies the transceiver block receiver PLL or the transceiver block transmitter bandwidth type. Values are LOW and HIGH. If omitted, the default is HIGH.
RX_BANDWIDTH_TYPE	String	No	Specifies the transceiver block receiver PLL bandwidth type. Values are LOW and HIGH. If omitted, the default is HIGH.
PLL_ENABLE_DC_COUPLING	String	No	Specifies whether to enable DC coupling on the transceiver block transmitter PLL clock input. Values are ON and OFF. If omitted, the default is OFF.
RX_ENABLE_DC_COUPLING	String	No	Specifies whether to enable DC coupling on the transceiver block receiver PLL data input. Values are ON and OFF. If omitted, the default is OFF.
USE_VOD_CTRL_SIGNAL	String	No	Specifies whether the VOD control signal is used. If this parameter is turned on, the tx_vodctrl port is used and the VOD_CTRL_SETTING parameter is ignored. Values are ON and OFF. If omitted, the default is OFF.

Table 11–3. Parameter Descriptions (Part 5 of 6)

Parameter	Type	Required	Comments
VOD_CTRL_SETTING	Integer	No	Specifies, in mV, the value of the V_{OD} control signal. Values are 400, 800, 1000, 1200, 1400, 1600. If omitted, the default is 1000.
USE_PREEMPHASIS_ CTRL_SIGNAL	String	No	Specifies whether the pre-emphasis control signal is used. If you enable this parameter, the tx_preemphasisctrl port is used and the PREEMPHASIS_CTRL_SETTING parameter is ignored. Values are ON and OFF. If omitted, the default is OFF.
PREEMPHASIS_CTRL_ SETTING	Integer	No	Specifies, as a percentage of the V_{OD} control setting, the value of the pre-emphasis control signal. Values are 0, 1, 2, 3, 4, or 5. If omitted, the default is 0.
USE_RX_CLKOUT	String	No	Specifies whether the output clock from the transceiver block receiver channel is used. Values are ON and OFF. If omitted, the default is OFF. If you enable this parameter, the rx_clkout port must be used.
USE_RX_CRUCLK	String	No	Specifies whether the clock recovery unit (CRU) is used for the transceiver block receiver PLL reference input clock. Values are ON and OFF. If omitted, the default is OFF. If you enable this parameter, the rx_cruclk port must be used.
RX_PPM_SETTING	Integer	No	Specifies the value of the PPM threshold between the transceiver block receiver PLL VCO and the clock recovery unit (CRU). Values are 125, 250, 500, or 1000. If omitted, the default is 1000.
RX_FORCE_SIGNAL_ DETECT	String	No	Specifies whether the rx_signaldetect[] port is used. Values are ON and OFF. If omitted, the default is ON.
USE_RX_CORECLK	String	No	Specifies whether the rx_coreclk[] port is used. Values are ON and OFF. If omitted, the default is OFF.
USE_TX_CORECLK	String	No	Specifies whether the tx_coreclk[] port is used. Values are ON and OFF. If omitted, the default is OFF.
FLIP_RX_OUT	String	No	Specifies whether the transceiver block receiver channel output data bits are flipped. Values are ON and OFF. If omitted, the default is OFF.
FLIP_TX_IN	String	No	Specifies whether the transceiver block transmitter channel input data bits are flipped. Values are ON and OFF. If omitted, the default is OFF.
INSTANTIATE_ TRANSMITTER_PLL	String	No	Specifies whether the transceiver block transmitter PLL is instantiated. Values are ON and OFF. If omitted, the default is OFF.

Table 11–3. Parameter Descriptions (Part 6 of 6)

Parameter	Type	Required	Comments
CONSIDER_INSTANTIATE_TRANSMITTER_PLL	String	No	Specifies whether the INSTANTIATE_TRANSMITTER_PLL parameter is turned on. Values are ON and OFF. If omitted, the default is OFF.
TX_TERMINATION	Integer	No	Specifies the termination setting on the pin fed by the transceiver block transmitter channel tx_out[] port. Values are 0, 1, 2, or 3. If omitted, the default is 2. Use the following settings: **Value** — **Termination Setting** 0 — 150 Ω 1 — 120 Ω 2 — 100 Ω 3 — External termination
RX_DATA_RATE	Integer	No	Specifies, in Mbps, the rate of data from the GXB receiver channel. If omitted, the default is 0.
RX_DATA_RATE_REMAINDER	Integer	No	Specifies, in bits per second (bps), the remainder of the RX_DATA_RATE parameter; RX_DATA_RATE * 1000000. This parameter helps to specify non-integral data rates. If omitted, the default is 0.
INTENDED_DEVICE_FAMILY	String	No	Use this parameter for modeling and behavioral simulation purposes. Create the altgxb megafunction with the MegaWizard® Plug-in Manager to calculate the value for this parameter.

12. REFCLKB Pin Constraints

SGX52012-1.0

Known Issues

This document discusses issues you might encounter in certain configurations of the Stratix® GX device. These issues are a result of a combination of resource utilization (REFCLKB pins that use Inter Quad (IQ) lines, signals (pllenable, pll_areset, and rxanalogreset), and software modeling.

The potential issues for certain configurations are described below:

1. The pll_areset and pllenable signals in a transceiver block reset its dedicated clock (REFCLKB) pad.

   If the design uses the REFCLKB pin of the active transceiver block to feed core logic, there is no clock to the core logic if the pll_areset or pllenable signal is asserted. This problem could be severe if the clock from the REFCLKB pin is also feeding the reset controller/logic in the core logic. The system may never come out of reset in this configuration.

   Modeling in the Quartus® II software simulation does not reveal this issue. This issue also exists for multiple transceiver block applications that use the REFCLKB pin for clocking.

2. Asserting the rxanalogreset signal of all four channels in a transceiver block resets its dedicated clock pad.

   This might happen if, in the design, only the receive portions of the channels are used and the REFCLKB pin of the transceiver block is used to feed the clock to the core logic. This can affect any logic using the clock from this pad. This behavior is not modeled in the Quartus II software simulation.

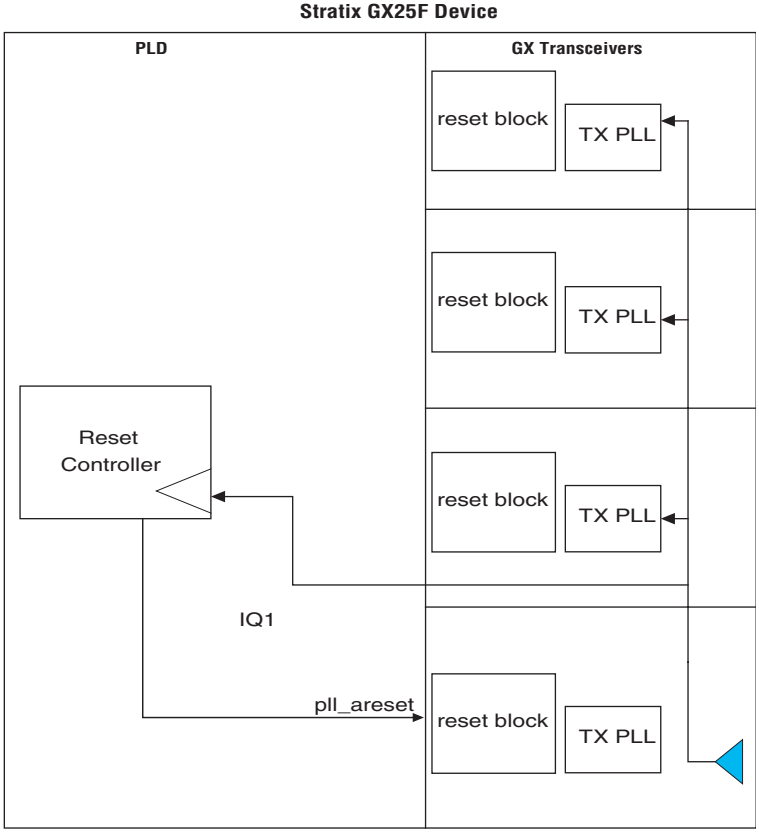
   If the reference clock to the receiver (rx_cruclk) is routed globally, the clock pad is not in use and will not flat-line the reference clock.

3. When the rxanalogreset signal of all four channels in a transceiver block is asserted, it also resets the transmitter PLL.

If the design is using the transmitter PLL output to drive any clock into the core (any part of the device), asserting the `rxanalogreset` signal of all four channels also resets the transmitter PLL and affects the clock that feeds the core logic. This behavior is not modeled in the Quartus II software simulation.

Figure 12–1 shows an example of a problem configuration using the Stratix GX 25F device. This configuration is also applicable to all devices in the Stratix GX device family.

Figure 12–1. Example Configuration



Quartus II Software Messages

The following sections provide details on the feedback provided by the various versions of the Quartus II software when the design contains configurations 1, 2, or 3, described previously.

Quartus II Software to Version 3.0 SP2

This version of the Quartus II software allows configurations 1, 2, and 3, and provides no warnings. Because the behavior of the configuration is not modeled, differences in functionality exist between simulations and actual silicon.

Altera® recommends that you do not run the reset controller off the clock from the REFCLKB pin. This might result in the signals to the GXB being deasserted asynchronously (with respect to the clocks from the GXB). However, a serial link usually goes through an initialization phase. Therefore, metastability issues might not be critical because the logic has enough time to recover during initialization.

Quartus II Software Version 4.0

For configurations 1 and 2, the Quartus II software version 4.0 prevents the **.sof** file from being generated.

- The assembler (ASM) will have an internal error (IE). Here are some of the messages from the Quartus II software log:
  - Internal Error: Sub-system: ASM, File: asm_cdr.cpp, Line: 839, Illegal Clock Placement. Problem between Quad PLL and the Quad Clock Pad. PLL has Enable connected
  - Internal Error: Sub-system: ASM, File: asm_cdr.cpp, Line: 840, Illegal Clock Placement. Problem between Quad PLL and the Quad Clock Pad. PLL has Reset connected
  - Internal Error: Sub-system: ASM, File: asm_cdr.cpp, Line: 863, Illegal Clock Placement. Problem between RX Channel Resets and the Quad Clock Pad. All RXs in Quad have Resets connected



Please refer to recommendation n in "Recommendations" on page 12–5 if you have an existing design (including the board) and cannot make changes.

The Quartus II software version 4.0 supports configuration 3. There are no warnings and results vary from simulation to actual silicon.

Quartus II Software Versions 4.0 SP1 & 4.1

For configurations 1, 2, and 3, the Quartus II software versions 4.0 SP1 and 4.1 provide an error message if the resets are used as described in those configurations. Unconstrained flows create a fit that does not have the clock pin reset problem or a no-fit if such a fit cannot be found. Quartus II software versions 4.0 SP1 and 4.1 provide an error message.

- User assignments that force the previously discussed issues with clock configuration yield a user error during fitter operation.

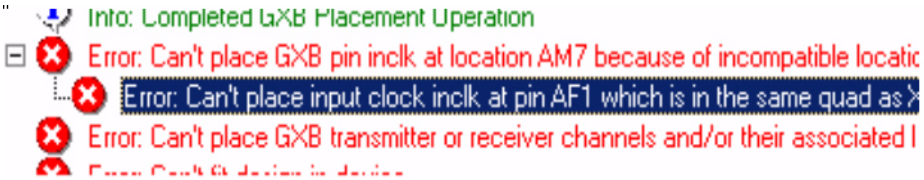
Examples of the error messages from the Quartus II software log are shown below. These are generated for configurations 1 and 2.

- Error: Can't place GXB pin `HSDI_CLK1_IN_I` at location AM7 because of incompatible location or I/O standard assignments
  Error: Can't place input clock `HSDI_CLK1_IN_I` at pin AM7 which is in the same quad as XGMII

- Error: Can't place GXB transmitter or receiver channels and/or their associated I/O pins due to illegal location or I/O standard assignments or inappropriate device
  Error: Can't fit design in device

  Figure 12–2 shows an example of an error message.

Figure 12–2. Error Messages



The following error messages are generated in the Quartus II software fitter for configuration 3.

- Error: XGMII GXB_RX:GXB_RX_b | CUSTOM_RX:CUSTOM_RX_inst | altgxb:altgxb_component | xgm_machine[0] exists in a Quad that has no GXB Transmitters and has GXB Transmitter PLL GXB_RX:GXB_RX_b | CUSTOM_RX:CUSTOM_RX_inst | altgxb:altgxb_component | pll[0], but all the `RXANALOGRESET` signals are connected. This is not allowed.

- Error: XGMII GXB_RX:GXB_RX_a|CUSTOM_RX:CUSTOM_RX_inst|altgxb:altgxb_component|xgm_machine[0] exists in a Quad that has no GXB Transmitters and has GXB Transmitter PLL GXB_RX:GXB_RX_a|CUSTOM_RX:CUSTOM_RX_inst|altgxb:altgxb_component|pll[0], but all the RXANALOGRESET signals are connected. This is not allowed.



Refer to recommendation n in the section "Recommendations" on page 12–5 if you have an existing design (including the board) and wish to keep the design.

Recommendations

The reset sequences described here are only recommendations. These recommendations are to guard against potential initialization issues. However, the transmitter PLLs within the transceivers are robust and should track the reference clock during a loss of lock conditions without requiring a reset. This reset signal must be used only if the PLLs fall into an unrecoverable state. During lab testing at the factory, the Stratix GX device did not require that the PLLs be reset. The PLLs have a wide pull in range and were able to relock to their respective reference clocks.

There might be situations where the read and write pointers in the phase compensation FIFO overlap. This will manifest as incorrect transmit data and in some cases, receive data. A digital reset should correct this error.

Here are the reset and clocking recommendations:

- Do not run the reset controller off the clock from the REFCLKB pin. This might result in the signals to the GXB being deasserted asynchronously (with respect to the clocks from the GXB). However, a serial link usually goes through an initialization phase. Hence, any concerns of meta-stability issues may not be critical as the logic has enough time to recover during initialization.

  To issue a `pll_areset` or `pllenable` or `rxanalogreset` (receive only) with configurations 1 or 2 using the Quartus II software version 4.0, use the following INI variable:

  ```
  asm_skip_gxb_clock_fanout_restriction=on
  ```

  For the Quartus II software version 4.0 SP1 (for configurations 1, 2, and 3), the setting in the Quartus II Settings File (**.qsf**) must be

  ```
  set_global_assignment -name
  STRATIXGX_ALLOW_CLOCK_FANOUT_WITH_ANALOG_RESET ON
  ```

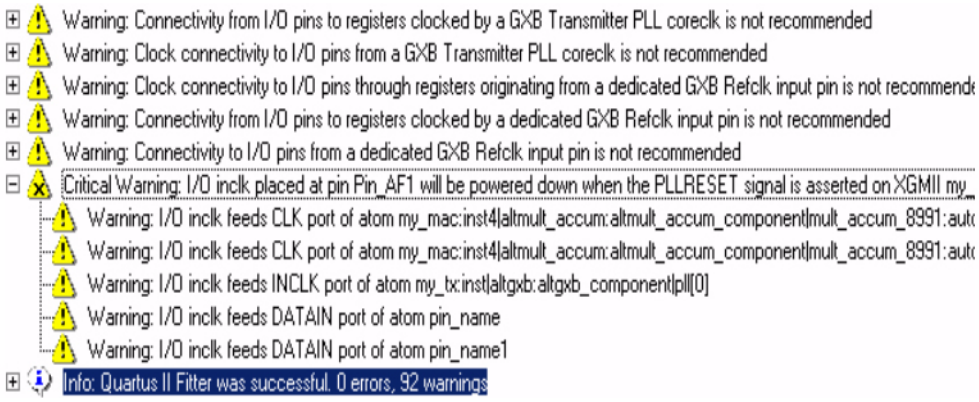
The equivalent INI setting is

```
asm_skip_gxb_clock_fanout_restriction = on
```

If you use the INI variable, you are not required to make any changes to the QSF. When you use the QSF setting, the Quartus II software issues a critical warning if you implement any of the configurations (1, 2, or 3). This critical warning message warns that these clock and reset configurations are not modeled in simulation and there will be differences in behavior between functional simulations and actual silicon.

Figure 12–3 shows an example of a critical warning.

Figure 12–3. Critical Warning



The critical warning in this case warns you that the `pll_areset` signal will power-down the clock pad.

- Do not use the `pll_areset`, `pllenable`, or `rxanalogreset` signals in receive-only configurations. The Stratix GX device is used in various systems and configurations. Based on the feedback and tests performed in the lab, assertion of any or all of these reset signals is not required for the PLLs to recover if they lose lock.

  The `pll_areset` and `pllenable` signals are power-down signals Assertion of these signals powers down the entire transceiver block. `rxanalogreset` is also power-down signal. Assertion of this signal powers down the receiver. For more information about these signals, see the *Reset Control & Power Down* chapter of the *Stratix GX Device Handbook, Volume 2*.

- You can use the REFCLKB pin on an unused transceiver block to bring in the clock. Altera recommends that you study the IQ routing to ensure that the transceiver block to be used can be reached from the selected REFCLKB pin. It might be possible to use the global clock pin for lower input clock frequencies. If the design requires using more than one transceiver block, there might be some restrictions because of the limited global clock resources. Please refer to the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1* for more information on clocking resources available for each device in the family.

- For designs that have receive-only configurations, try these solutions:

  - While asserting rxanalogreset, ensure (if possible) that all four channels are not being reset at the same time.
  - Do not use the transmitter PLL, train receive PLL from the receiver input reference clock (rx_cruclk).

You must carefully evaluate your design based on the recommendations in this appendix. Because you can configure the Stratix GX device in many different ways, there might be some configurations that are not covered by this document. Please contact ALTERA Applications for resolution on issues that are not addressed in this document.

Section II. Clock Management

This section provides information on clock management in Stratix® GX devices. It describes the enhanced and fast phase-locked loops (PLLs) that support clock management and synthesis for on-chip clock management, external system clock management, and high-speed I/O interfaces.

This section includes the following chapter:

- Chapter 13, General-Purpose PLLs in Stratix & Stratix GX Devices

Revision History

The table below shows the revision history for Chapter 13.

Chapter(s)	Date / Version	Changes Made
13	July 2005, v3.2	Chapter updated as part of the *Stratix Device Handbook* update.
	September 2004 v3.1	Added document to *Stratix GX Device Handbook*.

13. General-Purpose PLLs in Stratix & Stratix GX Devices

S52001-3.2

Introduction

Stratix® and Stratix GX devices have highly versatile phase-locked loops (PLLs) that provide robust clock management and synthesis for on-chip clock management, external system clock management, and high-speed I/O interfaces. There are two types of PLLs in each Stratix and Stratix GX device: enhanced PLLs and fast PLLs. Each device has up to four enhanced PLLs, which are feature-rich, general-purpose PLLs supporting advanced capabilities such as external feedback, clock switchover, phase and delay control, PLL reconfiguration, spread spectrum clocking, and programmable bandwidth. There are also up to eight fast PLLs per device, which offer general-purpose clock management with multiplication and phase shifting as well as high-speed outputs to manage the high-speed differential I/O interfaces.

The Altera® Quartus® II software enables the PLLs and their features without requiring any external devices.

Tables 13–1 and 13–2 show PLL availability for Stratix and Stratix GX devices, respectively.

Table 13–1. Stratix Device PLL Availability

Device	Fast PLLs								Enhanced PLLs			
	1	2	3	4	7	8	9	10	5 *(1)*	6 *(1)*	11 *(2)*	12 *(2)*
EP1S10	✓	✓	✓	✓					✓	✓		
EP1S20	✓	✓	✓	✓					✓	✓		
EP1S25	✓	✓	✓	✓					✓	✓		
EP1S30	✓	✓	✓	✓	✓ *(3)*	✓ *(3)*	✓ *(3)*	✓ *(3)*	✓	✓		
EP1S40	✓	✓	✓	✓	✓ *(3)*	✓ *(3)*	✓ *(3)*	✓ *(3)*	✓	✓	✓ *(3)*	✓ *(3)*
EP1S60	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
EP1S80	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Notes to Table 13–1:

(1) PLLs 5 and 6 each have eight single-ended outputs or four differential outputs.

(2) PLLs 11 and 12 each have one single-ended output.

(3) EP1S30 and EP1S40 devices do not support these PLLs in the 780-pin FineLine BGA® package.

Table 13–2. Stratix GX Device PLL Availability

Device	Fast PLLs				Enhanced PLLs			
	1	2	7	8	5	6	11	12
EP1S10C	✓	✓			✓	✓		
EP1S10D	✓	✓			✓	✓		
EP1S25C	✓	✓			✓	✓		
EP1S25D	✓	✓			✓	✓		
EP1S25F	✓	✓			✓	✓		
EP1S40D	✓	✓	✓	✓	✓	✓	✓	✓
EP1S40G	✓	✓	✓	✓	✓	✓	✓	✓

Table 13–3 shows the enhanced and fast PLL features in Stratix and Stratix GX devices.

Table 13–3. Stratix & Stratix GX PLL Features

Feature	Enhanced PLL	Fast PLL
Clock multiplication and division	*m*/(*n* × post-scale counter) *(1)*	*m*/(post-scale counter) *(2)*
Phase shift	Down to 156.25-ps increments *(3)*, *(4)*	Down to 125-ps increments *(3)*, *(4)*
Clock switchover	✓	
PLL reconfiguration	✓	
Programmable bandwidth	✓	
Spread spectrum clocking	✓	
Programmable duty cycle	✓	✓
Number of internal clock outputs	6	3 *(5)*
Number of external clock outputs	Four differential/eight singled-ended or one single-ended *(6)*	*(7)*
Number of feedback clock inputs	2 *(8)*	

Notes to Table 13–3:

(1) For enhanced PLLs, *m*, *n*, range from 1 to 512 and post-scale counters g, l, e range from 1 to 1024 with 50% duty cycle. With a non-50% duty cycle the post-scale counters g, l, e range from 1 to 512.
(2) For fast PLLs, *m*, *n*, and post-scale counters range from 1 to 32.
(3) The smallest phase shift is determined by the voltage controlled oscillator (VCO) period divided by 8.
(4) For degree increments, Stratix and Stratix GX devices can shift all output frequencies in increments of at least 45° . Smaller degree increments are possible depending on the frequency and divide parameters.
(5) PLLs 7, 8, 9, and 10 have two output ports per PLL. PLLs 1, 2, 3, and 4 have three output ports per PLL. On Stratix GX devices, PLLs 3, 4, 9, and 10 are not available for general-purpose use.
(6) Every Stratix and Stratix GX device has two enhanced PLLs (PLLs 5 and 6) with either eight single-ended outputs or four differential outputs each. Two additional enhanced PLLs (PLLs 11 and 12) in EP1S80, EP1S60, EP1S40 (PLL 11 and 12 not supported for F780 package), and EP1SGX40 devices each have one single-ended output.
(7) Fast PLLs can drive to any I/O pin as an external clock. For high-speed differential I/O pins, the device uses a data channel to generate `txclkout`.
(8) Every Stratix and Stratix GX device has two enhanced PLLs with one single-ended or differential external feedback input per PLL.

Figure 13–1 shows a top-level diagram of the Stratix device and PLL floorplan. Figure 13–2 shows a top-level diagram of the Stratix GX device and PLL floorplan. See "Clocking" on page 13–39 for more detail on PLL connections to global and regional clocks.

Figure 13–1. Stratix PLL Locations

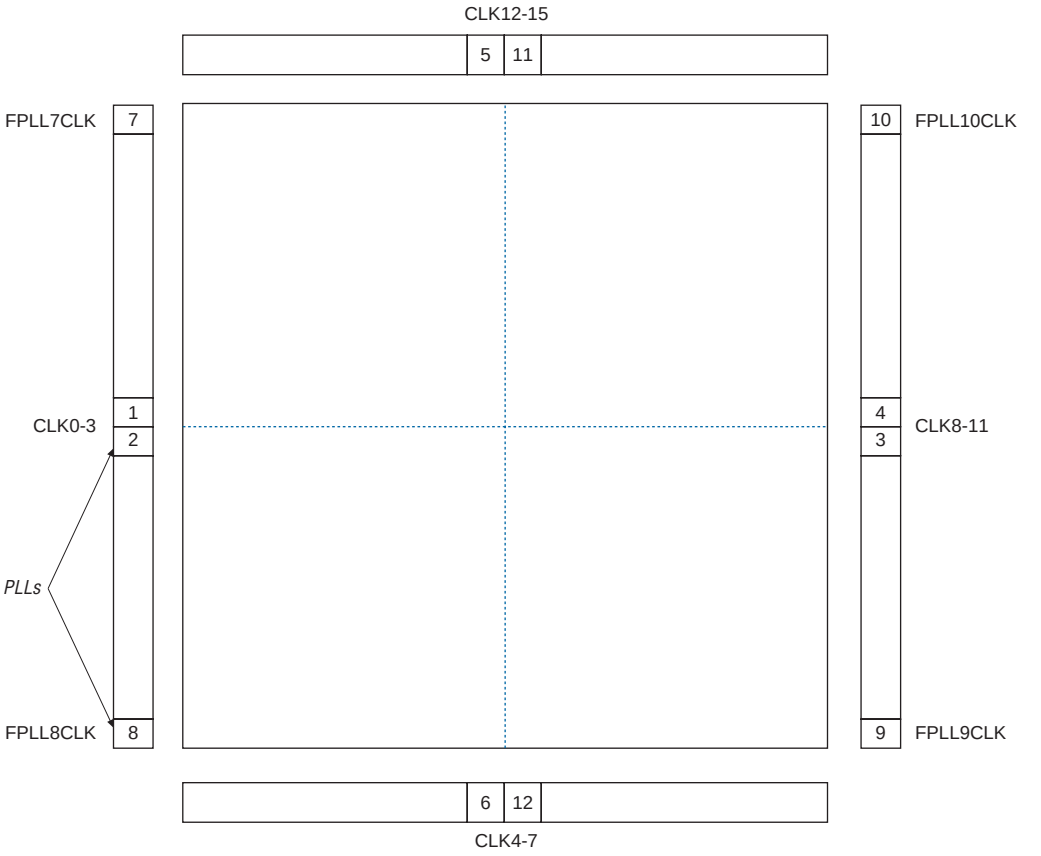
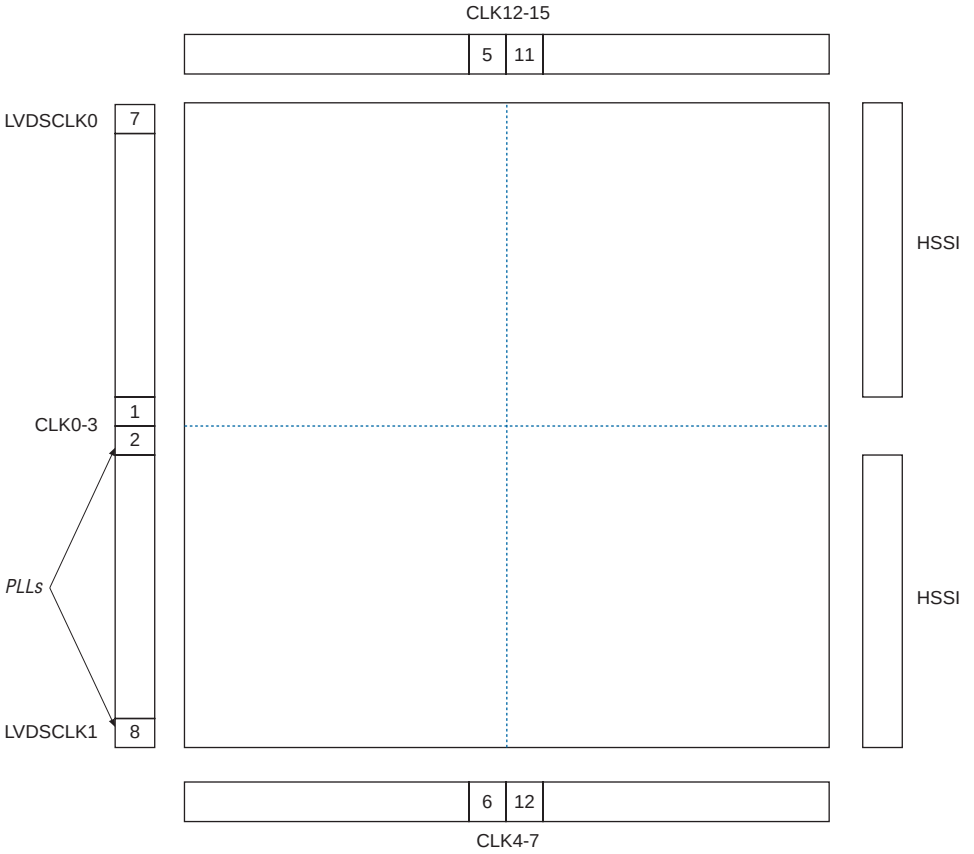


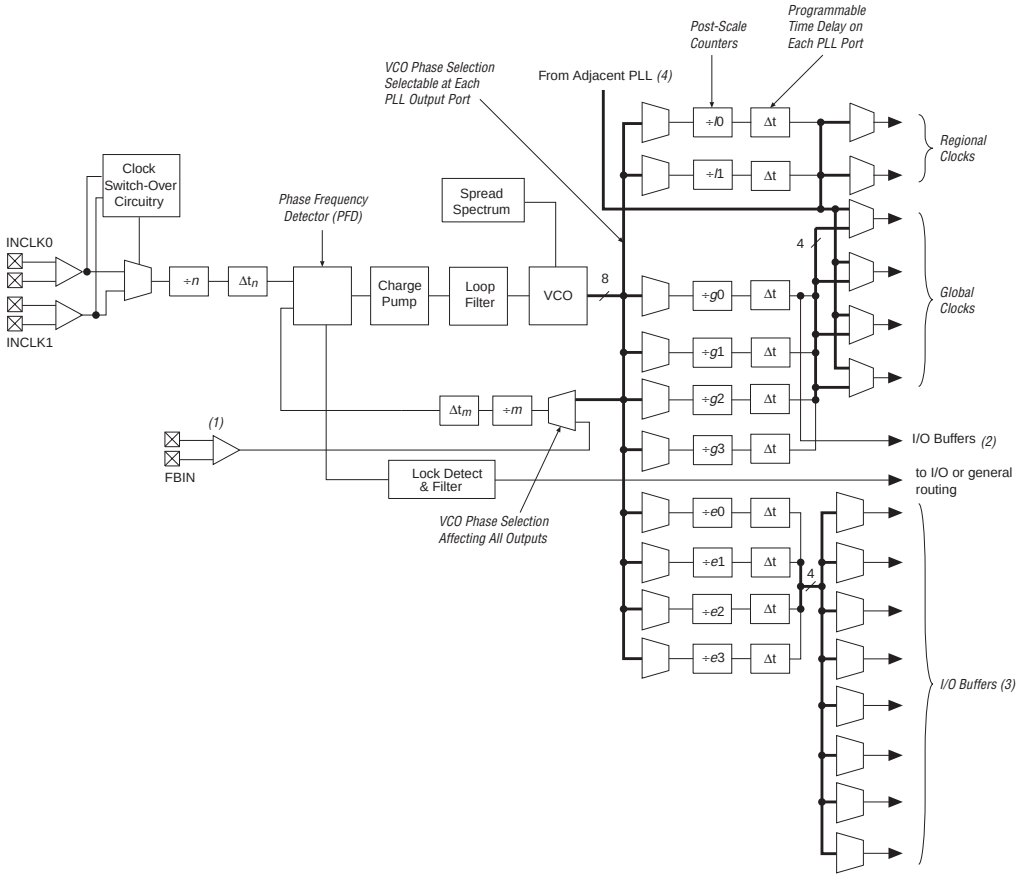
Figure 13–2. Stratix GX PLL Locations



Enhanced PLLs

Stratix and Stratix GX devices contain up to four enhanced PLLs with advanced clock management features. Figure 13–3 shows a diagram of the enhanced PLL.

Figure 13–3. Stratix & Stratix GX Enhanced PLL

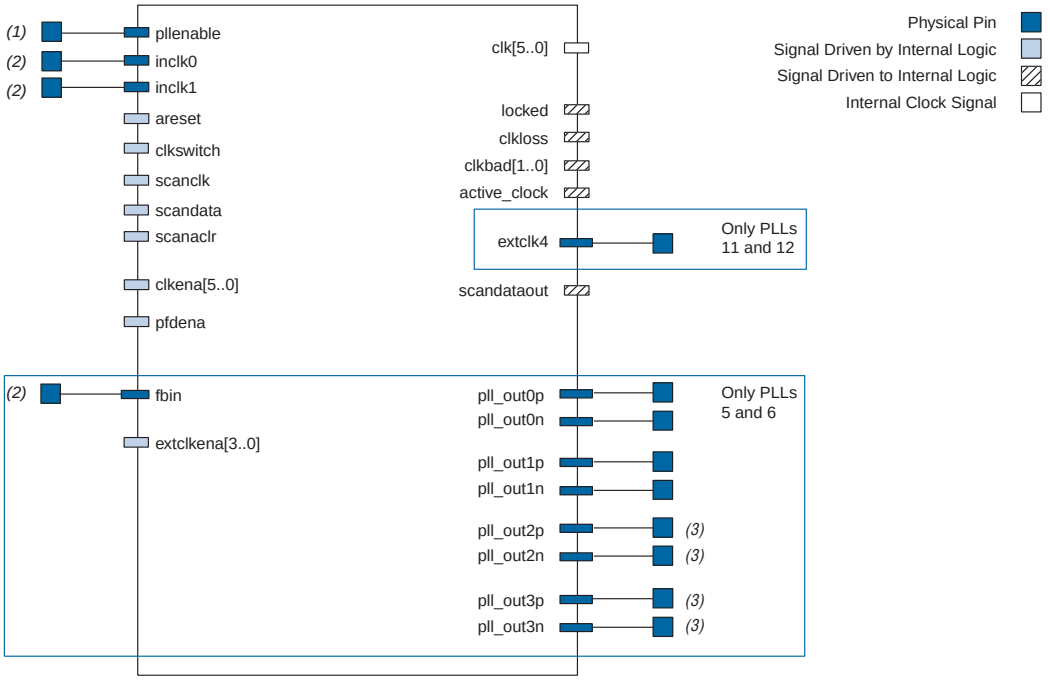


Notes to Figure 13–3:

(1) External feedback is available in PLLs 5 and 6.
(2) This single-ended external output is available from the *g*0 counter for PLLs 11 and 12.
(3) These four counters and external outputs are available in PLLs 5 and 6.
(4) This connection is only available on EP1SGX40 Stratix GX devices and EP1S40 and larger Stratix devices. For example, PLLs 5 and 11 are adjacent and PLLs 6 and 12 are adjacent. The EP1S40 device in the F780 package does not support PLLs 11 and 12.

Figure 13–4 shows all the possible ports of the enhanced PLLs.

Figure 13–4. Enhanced PLL Signals



Notes to Figure 13–4:

(1) This input pin is shared by all enhanced and fast PLLs.

(2) These are either single-ended or differential pins.

(3) EP1S10, EP1S20, and EP1S25 devices in 672-pin ball grid array (BGA) and 484- and 672-pin FineLine BGA packages only have two pairs of external clocks (i.e., `pll_out0p`, `pll_out0n`, `pll_out1p`, and `pll_out1n`).

Tables 13–4 and 13–5 describe all the enhanced PLL ports.

Table 13–4. Enhanced PLL Input Signals

Port	Description	Source	Destination
inclk[1..0]	Primary and secondary reference clock inputs to PLL	Pin	×*n* counter
fbin	External feedback input to the PLL (PLLs 5 and 6 only)	Pin	Phase frequency detector (PFD)
pllena	Enable pin for enabling or disabling all or a set of PLLs—active high	Pin	General PLL control signal
clkswitch	Switchover signal used to initiate external clock switchover control—this signal switches the clock on the rising edge of clkswitch	Logic array	PLL switchover circuit
areset	Signal used to reset the PLL which re-synchronizes all the counter outputs—active high	Logic array	General PLL control signal
clkena[5..0]	Enable clock driving regional or global clock—active high	Logic array	Clock output
extclkena[3..0]	Enable clock driving external clock (PLLs 5 and 6 only)—active high	Logic array	Clock output
pfdena	Enables the outputs from the phase frequency detector—active high	Logic array	PFD
scanclk	Serial clock signal for the real-time PLL control feature	Logic array	Reconfiguration circuit
scandata	Serial input data stream for the real-time PLL control feature	Logic array	Reconfiguration circuit
scanaclr	Serial shift register reset clearing all registers in the serial shift chain—active high	Logic array	Reconfiguration circuit

Table 13–5. Enhanced PLL Output Signals

Port	Description	Source	Destination
clk[5..0]	PLL outputs driving regional or global clock	PLL counter	Internal Clock
pll_out[3..0]p/n	pll_out[3..0] are PLL outputs driving the four differential or eight single-ended external clock output pins for PLLs 5 or 6. p or n are the positive (p) and negative (n) pins for differential pins.	PLL counter	Pin(s)
extclk4	PLL output driving external clock output pin from PLLs 11 and 12	PLL *g*0 counter	Pin
clkloss	Signal indicating the switchover circuit detected a switchover condition	PLL switchover circuit	Logic array
clkbad[1..0]	Signals indicating which reference clock is no longer toggling. clkbad1 indicates inclk1 status, clkbad0 indicates inclk0 status	PLL switchover circuit	Logic array
locked	Lock output from lock detect circuit—active high	PLL lock detect	Logic array
activeclock	Signal to indicate which clock (1 = inclk0 or 0 = inclk1) is driving the PLL.	PLL clock multiplexer	Logic array
scandataout	Output of the last shift register in the scan chain	PLL scan chain	Logic array

Clock Multiplication & Division

Each Stratix and Stratix GX device enhanced PLL provides clock synthesis for PLL output ports using $m/(n \times$ post-scale counter) scaling factors. The input clock is divided by a pre-scale counter, n, and is then multiplied by the m feedback factor. The control loop drives the VCO to match $f_{IN} \times (m/n)$. Each output port has a unique post-scale counter that divides down the high-frequency VCO.

For multiple PLL outputs with different frequencies, the VCO is set to the least common multiple of the output frequencies that meets its frequency specifications. Then, the post-scale counters scale down the output frequency for each output port. For example, if output frequencies required from one PLL are 33 and 66 MHz, then the Quartus II software sets the VCO to 330 MHz (the least common multiple of 33 and 66 MHz within the VCO range).

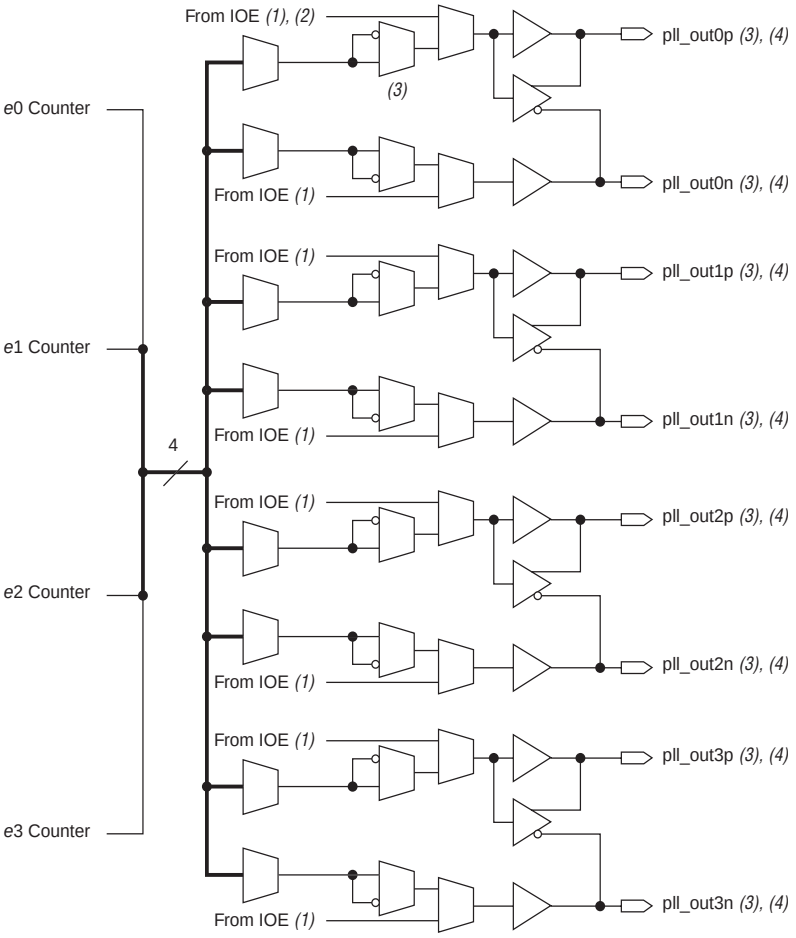
There is one pre-scale counter, n, and one multiply counter, m, per PLL, with a range of 1 to 512 on each. There are two post-scale counters (l) for regional clock output ports, four counters (g) for global clock output ports, and up to four counters (e) for external clock outputs, all ranging from 1 to 1024 with a 50% duty cycle setting. The post-scale counters

range from 1 to 512 with any non-50% duty cycle setting. The Quartus II software automatically chooses the appropriate scaling factors according to the input frequency, multiplication, and division values entered into the `altpll` MegaWizard Plug-In Manager.

External Clock Outputs

Enhanced PLLs 5 and 6 each support up to eight single-ended clock outputs (or four differential pairs). See Figure 13–5.

Figure 13–5. External Clock Outputs for PLLs 5 & 6



Notes to Figure 13–5:

(1) LE: logic element.

(2) The design can use each external clock output pin as a general-purpose output pin from the logic array. These pins are multiplexed with IOE outputs.

(3) Two single-ended outputs are possible per output counter—either two outputs of the same frequency and phase or one shifted 180° .

(4) EP1S10, EP1S20, and EP1S25 devices in 672-pin ball grid array (BGA) and 484- and 672-pin FineLine BGA packages only have two pairs of external clocks (i.e., `pll_out0p`, `pll_out0n`, `pll_out1p`, and `pll_out1n`).

Any of the four external output counters can drive the single-ended or differential clock outputs for PLLs 5 and 6. This means one counter or frequency can drive all output pins available from PLL 5 or PLL 6. Each

pair of output pins (four pins total) has dedicated VCC and GND pins to reduce the output clock's overall jitter by providing improved isolation from switching I/O pins.

For PLLs 5 and 6, each pin of a single-ended output pair can either be in phase or 180° out of phase. The Quartus II software transfers the NOT gate in the design into the IOE to implement 180° phase with respect to the other pin in the pair. The clock output pin pairs support the same I/O standards as standard output pins (in the top and bottom banks) as well as LVDS, LVPECL, PCML, HyperTransport™ technology, differential HSTL, and differential SSTL. Table 13–6 shows which I/O standards the enhanced PLL clock pins support. When in single-ended or differential mode, one power pin supports two differential or four single-ended pins. Both outputs use the same standards in single-ended mode to maintain performance. You can also use the external clock output pins as user output pins if external enhanced PLL clocking is not needed.

The enhanced PLL can also drive out to any regular I/O pin through the global or regional clock network. The jitter on the output clock is not guaranteed for this case.

Table 13–6. I/O Standards Supported for Enhanced PLL Pins (Part 1 of 2)

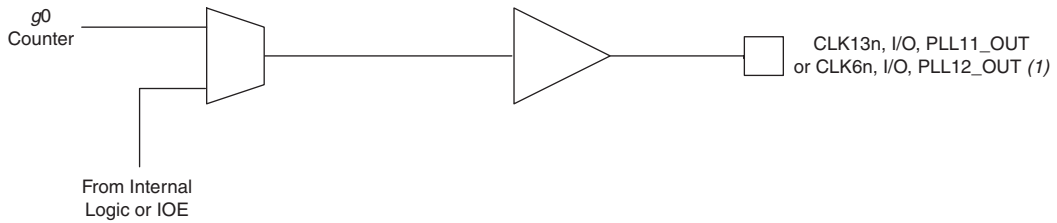
I/O Standard	Input			Output
	INCLK	FBIN	PLLENABLE	EXTCLK
LVTTL	✓	✓	✓	✓
LVCMOS	✓	✓	✓	✓
2.5 V	✓	✓		✓
1.8 V	✓	✓		✓
1.5 V	✓	✓		✓
3.3-V PCI	✓	✓		✓
3.3-V PCI-X 1.0	✓	✓		✓
LVPECL	✓	✓		✓
PCML	✓	✓		✓
LVDS	✓	✓		✓
HyperTransport technology	✓	✓		✓
Differential HSTL	✓			✓
Differential SSTL				✓
3.3-V GTL	✓	✓		✓

Table 13–6. I/O Standards Supported for Enhanced PLL Pins (Part 2 of 2)

I/O Standard	Input			Output
	INCLK	FBIN	PLLENABLE	EXTCLK
3.3-V GTL+	✓	✓		✓
1.5-V HSTL Class I	✓	✓		✓
1.5-V HSTL Class II	✓	✓		✓
1.8-V HSTL Class I	✓	✓		✓
1.8-V HSTL Class II	✓	✓		✓
SSTL-18 Class I	✓	✓		✓
SSTL-18 Class II	✓	✓		✓
SSTL-2 Class I	✓	✓		✓
SSTL-2 Class II	✓	✓		✓
SSTL-3 Class I	✓	✓		✓
SSTL-3 Class II	✓	✓		✓
AGP (1× and 2×)	✓	✓		✓
CTT	✓	✓		✓

Enhanced PLLs 11 and 12 support one single-ended output each (see Figure 13–6). These outputs do not have their own VCC and GND signals. Therefore, to minimize jitter, do not place switching I/O pins next to this output pin.

Figure 13–6. External Clock Outputs for Enhanced PLLs 11 & 12



Note to Figure 13–6:
(1) For PLL11, this pin is CLK13n; for PLL 12 this pin is CLK6n.

Stratix and Stratix GX devices can drive any enhanced PLL driven through the global clock or regional clock network to any general I/O pin as an external output clock. The jitter on the output clock is not guaranteed for these cases.

Clock Feedback

The following three feedback modes in Stratix and Stratix GX device enhanced PLLs allow multiplication and/or phase shifting:

- Zero delay buffer: The external clock output pin is phase-aligned with the clock input pin for zero delay. Altera recommends using the same I/O standard on the input clock and the output clocks for optimum performance.
- External feedback: The external feedback input pin, FBIN, is phase-aligned with the clock input, CLK, pin. Aligning these clocks allows you to remove clock delay and skew between devices. This mode is only possible for PLLs 5 and 6. PLLs 5 and 6 each support feedback for one of the dedicated external outputs, either one single-ended or one differential pair. In this mode, one encounter feeds back to the PLL FBIN input, becoming part of the feedback loop.
- Normal mode: If an internal clock is used in this mode, it is phase-aligned to the input clock pin. The external clock output pin has a phase delay relative to the clock input pin if connected in this mode.
- No compensation: In this mode, the PLL does not compensate for any clock networks or external clock outputs.

Table 13–7 shows which modes are supported by which PLL type.

Table 13–7. Clock Feedback Mode Availability

Clock Feedback Mode	Mode Available in	
	Enhanced PLLs	Fast PLLs
No compensation mode	Yes	Yes
Normal Mode	Yes	Yes
Zero delay buffer mode	Yes	No
External feedback mode	Yes	No

Phase Shifting

Stratix and Stratix GX device enhanced PLLs provide advanced programmable phase shifting. You set these parameters in the Quartus II software.

Phase Delay

The Quartus II software automatically sets the phase taps and counter settings according to the phase shift entry. You enter a desired phase shift and the Quartus II software automatically sets the closest setting achievable. This type of phase shift is not reconfigurable during system operation. For phase shifting, enter a phase shift (in degrees or time units) for each PLL clock output port or for all outputs together in one shift.

You can select phase-shifting values in time units with a resolution of 156.25 to 416.66 ps. This resolution is a function of frequency input and the multiplication and division factors (that is, it is a function of the VCO period), with the finest step being equal to an eighth (×0.125) of the VCO period. Each clock output counter can choose a different phase of the VCO period from up to eight taps for individual fine-step selection. Also, each clock output counter can use a unique initial count setting to achieve individual coarse-shift selection in steps of one VCO period. The combination of coarse and fine shifts allows phase shifting for the entire input clock period.

The equation to determine the precision of the phase shifting in degrees is: 45° ÷ post-scale counter value. Therefore, the maximum step size is 45°, and smaller steps are possible depending on the multiplication and division ratio necessary on the output counter port.

This type of phase shift provides the highest precision since it is the least sensitive to process, supply, and temperature variation.

Lock Detect

The lock output indicates that there is a stable clock output signal in phase with the reference clock. Without any additional circuitry, the lock signal may toggle as the PLL begins tracking the reference clock. You may need to gate the lock signal for use as a system control. The lock signal from the locked port can drive the logic array or an output pin.

Whenever the PLL loses lock for any reason (be it excessive `inclk` jitter, clock switchover, PLL reconfiguration, power supply noise, etc.), the PLL must be reset with the `areset` signal to guarantee correct phase relationship between the PLL output clocks. If the phase relationship between the input clock versus output clock, and between different output clocks from the PLL is not important in your design, the PLL need not be reset.



See the *Stratix FPGA Errata Sheet* for more information on implementing the gated lock signal in your design.

Programmable Duty Cycle

The programmable duty cycle allows enhanced PLLs to generate clock outputs with a variable duty cycle. This feature is supported on each enhanced PLL post-scale counter (*g*0..*g*3, *l*0..*l*3, *e*0..*e*3). The duty cycle setting is achieved by a low and high time count setting for the post-scale counters. The Quartus II software uses the frequency input and the required multiply or divide rate to determine the duty cycle choices. The precision of the duty cycle is determined by the post-scale counter value chosen on an output. The precision is defined by 50% divided by the post-scale counter value. The closest value to 100% is not achievable for a given counter value. For example, if the *g*0 counter is 10, then steps of 5% are possible for duty cycle choices between 5 to 90%.

If the device uses external feedback, you must set the duty cycle for the counter driving off the device to 50%.

General Advanced Clear & Enable Control

There are several control signals for clearing and enabling PLLs and PLL outputs. You can use these signals to control PLL resynchronization and gate PLL output clocks for low-power applications.

The `pllenable` pin is a dedicated pin that enables/disables PLLs. When the `pllenable` pin is low, the clock output ports are driven by `GND` and all the PLLs go out of lock. When the `pllenable` pin goes high again, the PLLs relock and resynchronize to the input clocks. You can choose which PLLs are controlled by the `pllenable` signal by connecting the `pllenable` input port of the `altpll` megafunction to the common `pllenable` input pin.

The `areset` signals are reset/resynchronization inputs for each PLL. The `areset` signal should be asserted every time the PLL loses lock to guarantee correct phase relationship between the PLL output clocks. Users should include the `areset` signal in designs if any of the following conditions are true:

- PLL reconfiguration or clock switchover enables in the design
- Phase relationships between output clocks need to be maintained after a loss of lock condition

The device input pins or logic elements (LEs) can drive these input signals. When driven high, the PLL counters reset, clearing the PLL output and placing the PLL out of lock. The VCO sets back to its nominal setting (~700 MHz). When driven low again, the PLL resynchronizes to its input as it relocks. If the target VCO frequency is below this nominal

frequency, then the output frequency starts at a higher value than desired as the PLL locks. If the system cannot tolerate this, the clkena signal can disable the output clocks until the PLL locks.

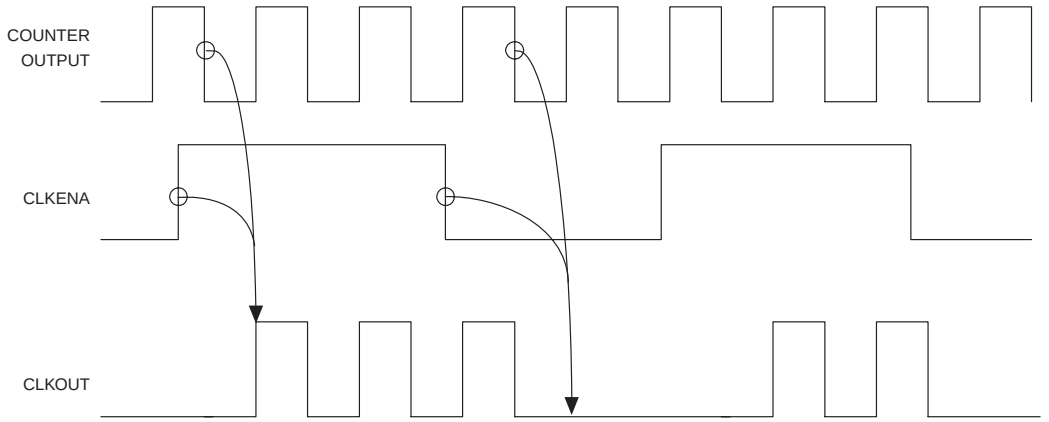
The pfdena signals control the phase frequency detector (PFD) output with a programmable gate. If you disable the PFD, the VCO operates at its last set value of control voltage and frequency with some long-term drift to a lower frequency. The system continues running when the PLL goes out of lock or the input clock is disabled. By maintaining the last locked frequency, the system has time to store its current settings before shutting down. You can either use your own control signal or a clkloss status signal to trigger pdfena.

The clkena signals control the enhanced PLL regional and global outputs. Each regional and global output port has its own clkena signal. The clkena signals synchronously disable or enable the clock at the PLL output port by gating the outputs of the *g* and *l* counters. The clkena signals are registered on the falling edge of the counter output clock to enable or disable the clock without glitches.

Figure 13–7 shows the waveform example for a PLL clock port enable. The PLL can remain locked independent of the clkena signals since the loop-related counters are not affected. This feature is useful for applications that require a low power or sleep mode. Upon re-enabling, the PLL does not need a resynchronization or relock period. The clkena signal can also disable clock outputs if the system is not tolerant to frequency overshoot during resynchronization.

The extclkena signals work in the same way as the clkena signals, but they control the external clock output counters (*e*0, *e*1, *e*2, and *e*3). Upon re-enabling, the PLL does not need a resynchronization or relock period unless the PLL is using external feedback mode. In order to lock in external feedback mode, the external output must drive the board trace back to the FBIN pin.

Figure 13–7. extclkena Signals



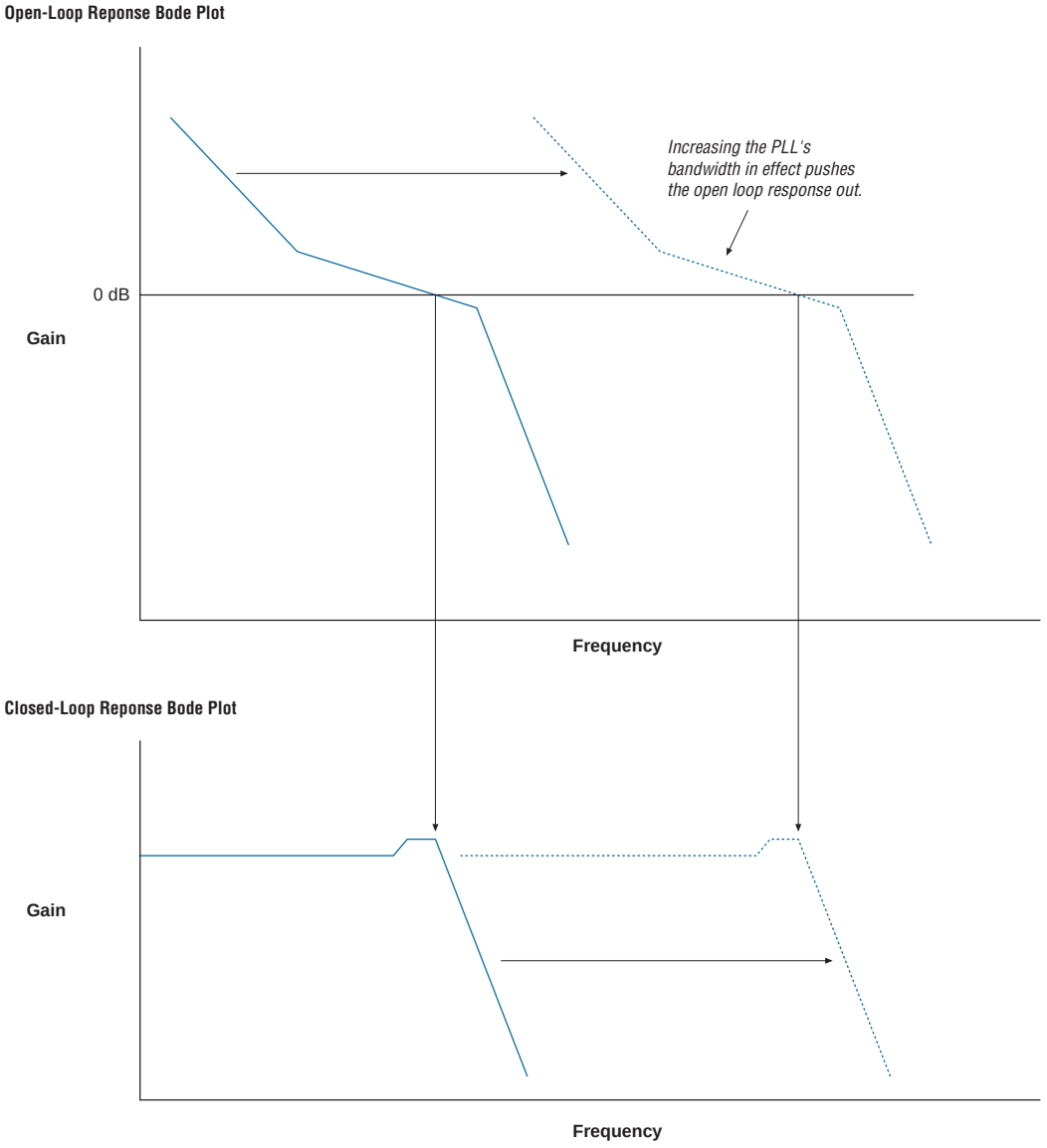
Programmable Bandwidth

Enhanced PLLs provide advanced control of the PLL bandwidth using the programmable characteristics of the PLL loop, including loop filter and charge pump.

Background

The PLL bandwidth is the measure of the PLLs ability to track the input clock and jitter. It is determined by the -3-dB frequency of the closed-loop gain in the PLL or approximately the unity gain point for open loop PLL response. As Figure 13–8 shows, these points correspond to approximately the same frequency.

Figure 13–8. Open- & Closed-Loop Response Bode Plots



A high-bandwidth PLL provides a fast lock time and tracks jitter on the reference clock source, passing it through to the PLL output. A low-bandwidth PLL filters out reference clock jitter, but increases lock time. Stratix device enhanced PLLs allow you to control the bandwidth over a

finite range to customize the PLL characteristics for a particular application. Applications that require clock switchover (such as TDMA, frequency hopping wireless, and redundant clocking) can benefit from the programmable bandwidth feature of the Stratix and Stratix GX PLLs.

The bandwidth and stability of such a system is determined by a number of factors including the charge pump current, the loop filter resistor value, the high-frequency capacitor value (in the loop filter), and the *m*-counter value. You can use the Quartus II software to control these factors and to set the bandwidth to the desired value within a given range.

You can set the bandwidth to the appropriate value to balance the need for jitter filtering and lock time. Figures 13–9 and 13–10 show the output of a low- and high-bandwidth PLL, respectively, as it locks onto the input clock.

Figure 13–9. Low-Bandwidth PLL Lock Time

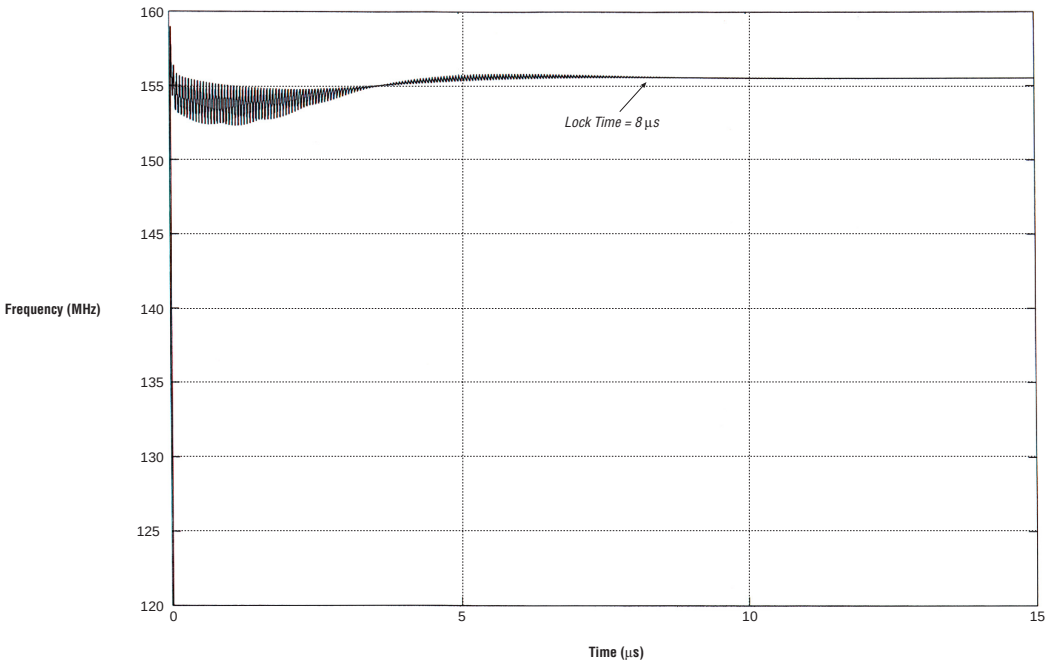
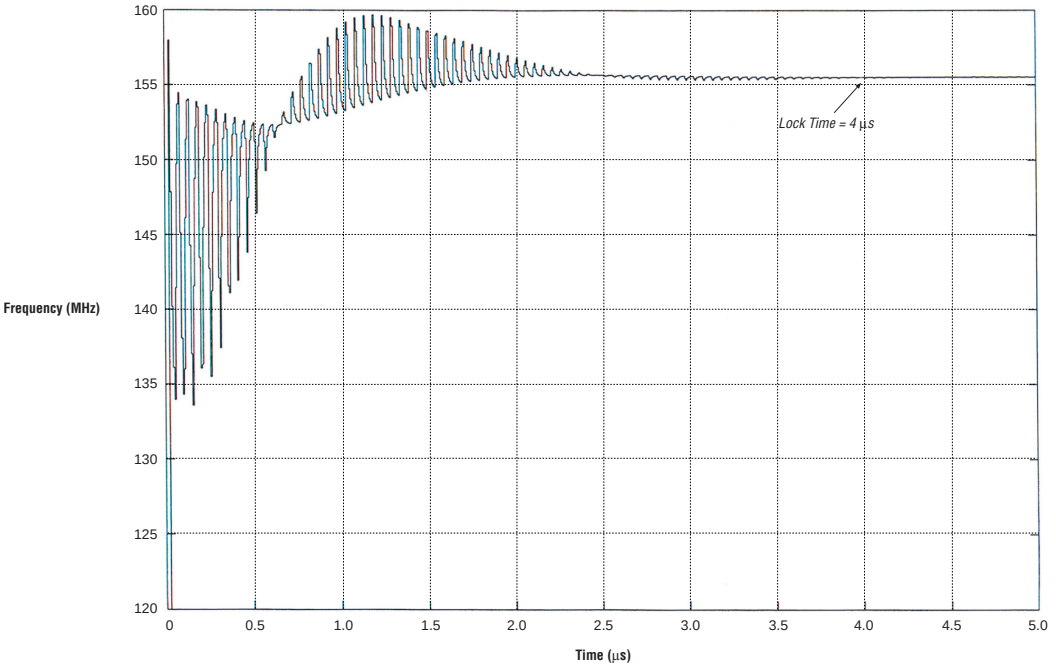


Figure 13–10. High-Bandwidth PLL Lock Time



A high-bandwidth PLL may benefit a system with two cascaded PLLs. If the first PLL uses spread spectrum (as user-induced jitter), the second PLL needs a high bandwidth so it can track the jitter that is feeding it. A low-bandwidth PLL may, in this case, lose lock due to the spread spectrum-induced jitter on the input clock.

A low-bandwidth PLL may benefit a system using clock switchover. When the clock switchover happens, the PLL input temporarily stops. A low-bandwidth PLL would react more slowly to changes to its input clock and take longer to drift to a lower frequency (caused by the input stopping) than a high-bandwidth PLL. Figures 13–11 and 13–12 demonstrate this property.

The two plots show the effects of clock switchover with a low- or high-bandwidth PLL. When the clock switchover happens, the output of the low-bandwidth PLL (see Figure 13–11) drifts to lower frequency much slower than the high-bandwidth PLL output (see Figures 13–12).

Figure 13–11. Effect of Low Bandwidth on Clock Switchover

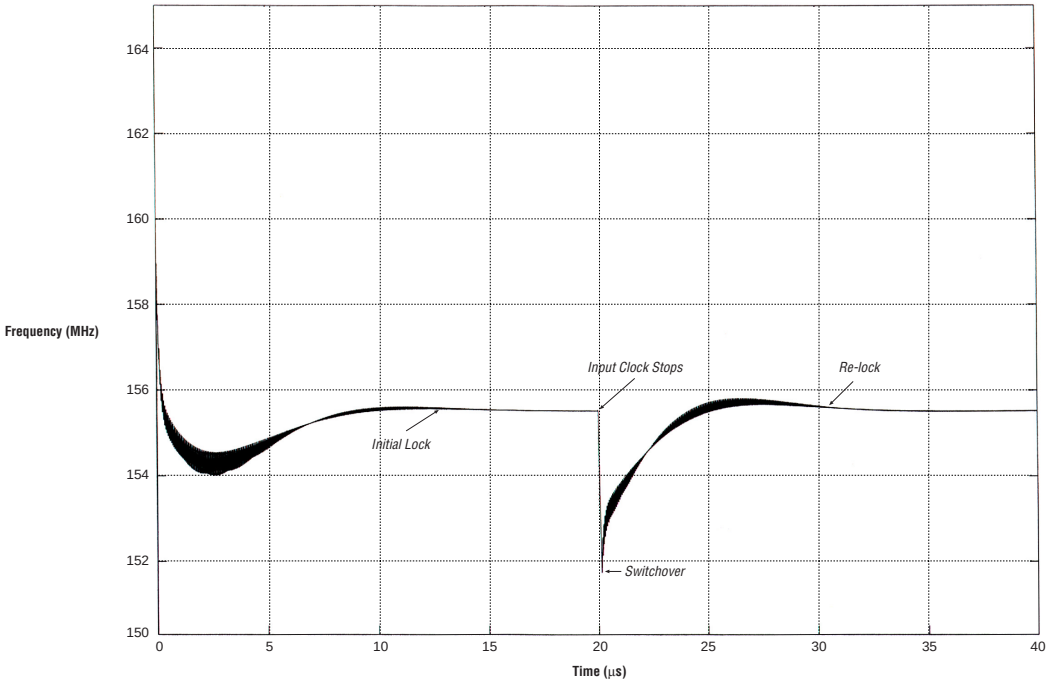
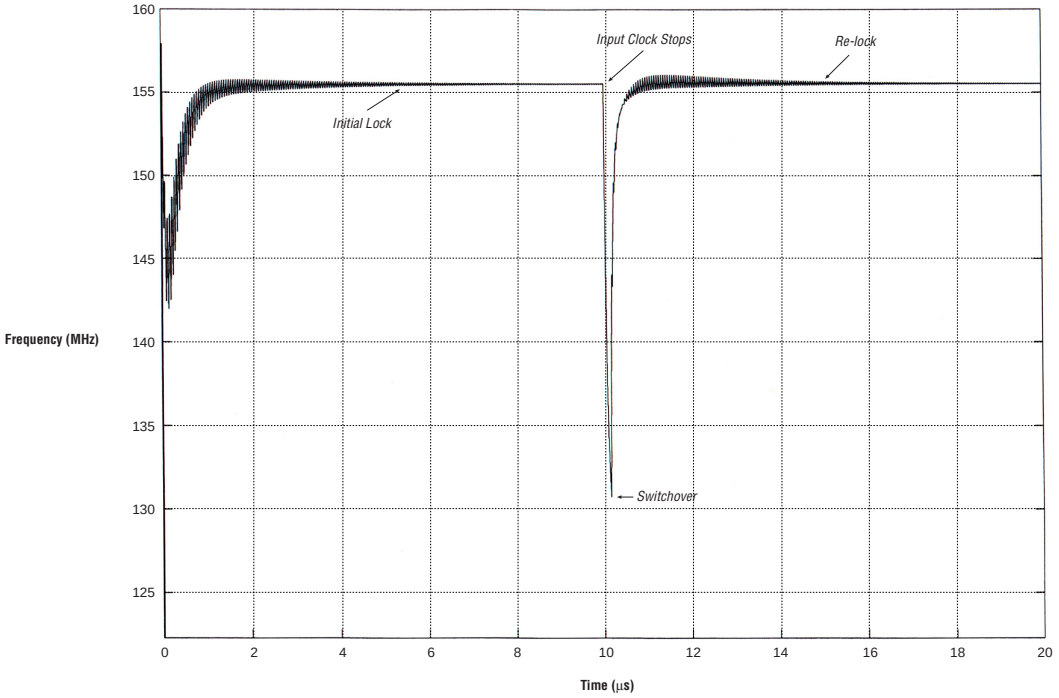


Figure 13–12. Effect of High Bandwidth on Clock Switchover



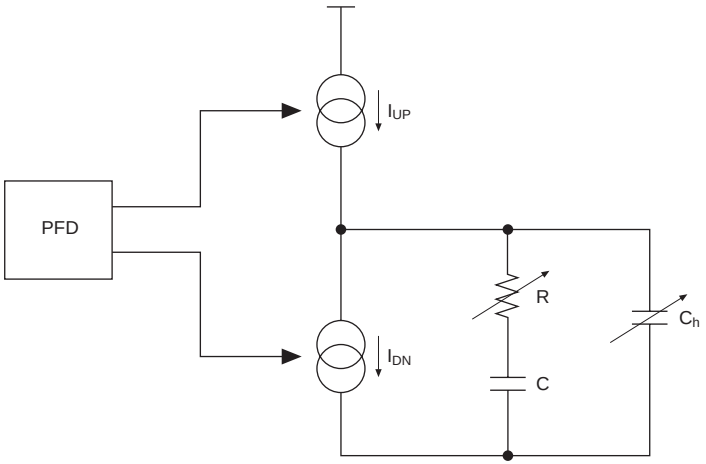
Implementation

Traditionally, external components such as the VCO or loop filter control a PLL's bandwidth. Most loop filters are made up of passive components, such as resistors and capacitors, which take up unnecessary board space and increase cost. With Stratix and Stratix GX device enhanced PLLs, all the components are contained within the device to increase performance and decrease cost.

Stratix and Stratix GX device enhanced PLLs implement programmable bandwidth by giving you control of the charge pump current and loop filter resistor (R) and high-frequency capacitor (C_h) values (see Table 13–8). The Stratix and Stratix GX device enhanced PLL bandwidth ranges from approximately 150 kHz to 2 MHz.

The charge pump current directly affects the PLL bandwidth. The higher the charge pump current, the higher the PLL bandwidth. You can choose from a fixed set of values for the charge pump current. Figure 13–13 shows the loop filter and the components that you can set via the Quartus II software.

Figure 13–13. Loop Filter Programmable Components



Software Support

The Quartus II software provides two levels of programmable bandwidth control. The first level allows you to enter a value for the desired bandwidth directly into the Quartus II software using the MegaWizard® Plug-In Manager. Alternatively, you can set the `bandwidth` parameter in the `altpll` function to the desired bandwidth. The Quartus II software then chooses each individual bandwidth parameter to achieve the desired setting. If designs cannot achieve the desired bandwidth setting, the Quartus II software selects the closest achievable value. For preset low, medium, and high bandwidth settings, the Quartus II software sets the bandwidth as follows:

- Low bandwidth is set at 150 KHz
- Medium bandwidth is set at 800 KHz
- High bandwidth is set at 2 Mhz

If you choose Auto bandwidth, the Quartus II software chooses the PLL settings and you can get a bandwidth setting outside the 150-Khz to 2-Mhz range.

An advanced level of control is also possible for precise control of the loop filter parameters. This level allows you to specifically select the charge pump current, loop filter resistor value, and loop filter (high frequency) capacitor value. These parameters are: charge_pump_current, loop_filter_r, and loop_filter_c. Each parameter supports the specific range of values listed in Table 13–8.

Table 13–8. Advanced Loop Filter Parameters

Parameter	Values
Resistor values (kΩ)	1, 2, 3, 4, 7, 8, 9, 10
High-frequency capacitance values (pF)	5, 10, 15, 20
Charge pump current settings (µA)	10, 15, 20, 24, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 100, 112, 135, 148, 164, 212



For more information on PLL software support in the Quartus II software, see the *altpll Megafunction User Guide*.

Clock Switchover



For more information on implementing clock switchover, see *AN 313: Implementing Clock Switchover in Stratix & Stratix GX Devices.*

Spread-Spectrum Clocking

Digital clocks are generally square waves with short rise times and a 50% duty cycle. These high-speed digital clocks concentrate a significant amount of energy in a narrow bandwidth at the target frequency and at the higher frequency harmonics. This results in high energy peaks and increased electromagnetic interference (EMI). The radiated noise from the energy peaks travels in free air and, if not minimized, can lead to corrupted data and intermittent system errors, which can jeopardize system reliability.

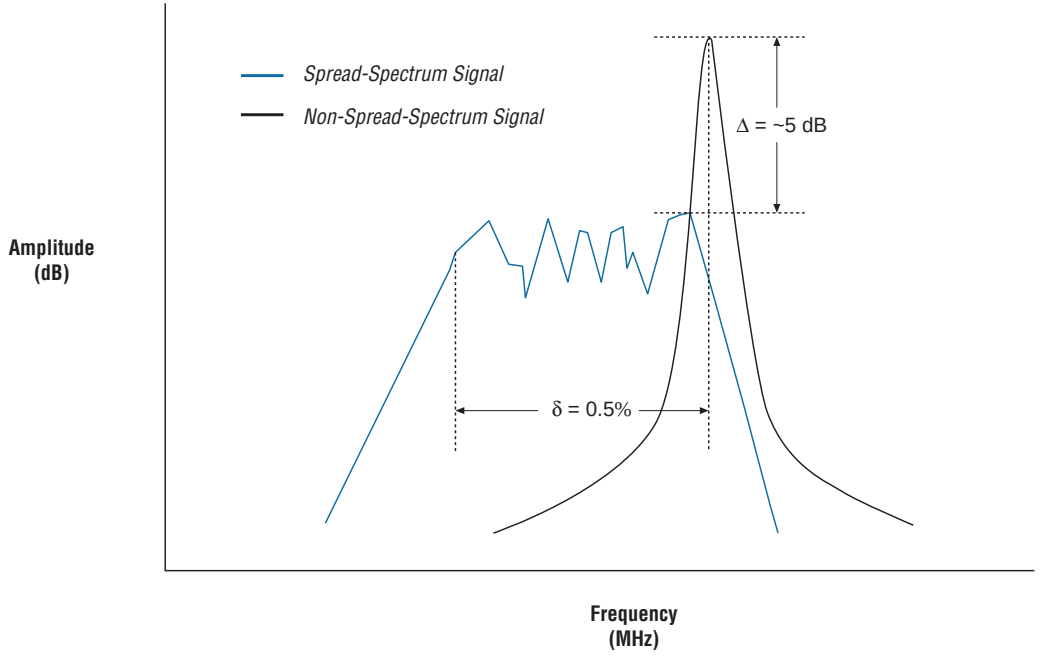
Background

Traditional methods for limiting EMI include shielding, filtering, and multi-layer printed circuit boards (PCBs). However, these methods significantly increase the overall system cost and sometimes are not enough to meet EMI compliance. Spread-spectrum technology provides a simple and effective technique for reducing EMI emissions without additional cost and the trouble of re-designing a board.

Spread-spectrum technology modulates the target frequency over a small range. For example, if a 100-MHz signal has a 0.5% down-spread modulation, then the frequency is swept from 99.5 to 100 MHz. Figure 13–14 gives a graphical representation of the energy present in a spread-spectrum signal as opposed to a non-spread-spectrum signal. It is apparent that instead of concentrating the energy at the target frequency, the energy is re-distributed across a wider band of frequencies, which reduces peak energy.

Not only is there a reduction in the fundamental peak EMI components, but there is also a reduction in EMI of the higher order harmonics. Since some regulations focus on peak EMI emissions, rather than average EMI emissions, spread-spectrum technology is a valuable method of EMI reduction.

Figure 13–14. Spread-Spectrum Signal Energy versus Non-Spread-Spectrum Signal Energy



Spread-spectrum technology would benefit a design with high EMI emissions and/or strict EMI requirements. Device-generated EMI is dependent on frequency, output voltage swing amplitude, and slew rate. For example, a design using LVDS already has low EMI emissions

because of the low-voltage swing. The differential LVDS signal also allows for EMI rejection within the signal. Therefore, this situation may not require spread-spectrum technology.

Description

Stratix and Stratix GX device enhanced PLLs feature spread-spectrum technology to reduce the EMI emitted from the device. The enhanced PLL provides up to a 0.5% down spread (–0.5%) using a triangular, also known as linear, modulation profile. The modulation frequency is programmable and ranges from approximately 30 to 150 kHz. The spread percentage is based on the clock input to the PLL and the *m* and *n* settings. Spread-spectrum technology reduces the peak energy by 2 to 5 dB at the target frequency. However, this number is dependent on bandwidth and the *m* and *n* counter values and can vary from design to design.

Spread percentage, also known as modulation width, is defined as the percentage that the design modulates the target frequency. A negative (–) percentage indicates a down spread, a positive (+) percentage indicates an up spread, and a (±) indicates a center spread. Modulation frequency is the frequency of the spreading signal or how fast the signal sweeps from the minimum to the maximum frequency. Down-spread modulation shifts the target frequency down by half the spread percentage, centering the modulated waveforms on a new target frequency.

The *m* and *n* counter values are toggled at the same time between two fixed values. The loop filter then slowly changes the VCO frequency to provide the spreading effect, which results in a triangular modulation. An additional spread-spectrum counter (shown in Figure 13–15) sets the modulation frequency. Figure 13–15 shows how spread-spectrum technology is implemented in the Stratix device enhanced PLL.

Figure 13–15. Spread-Spectrum Circuit Block Diagram

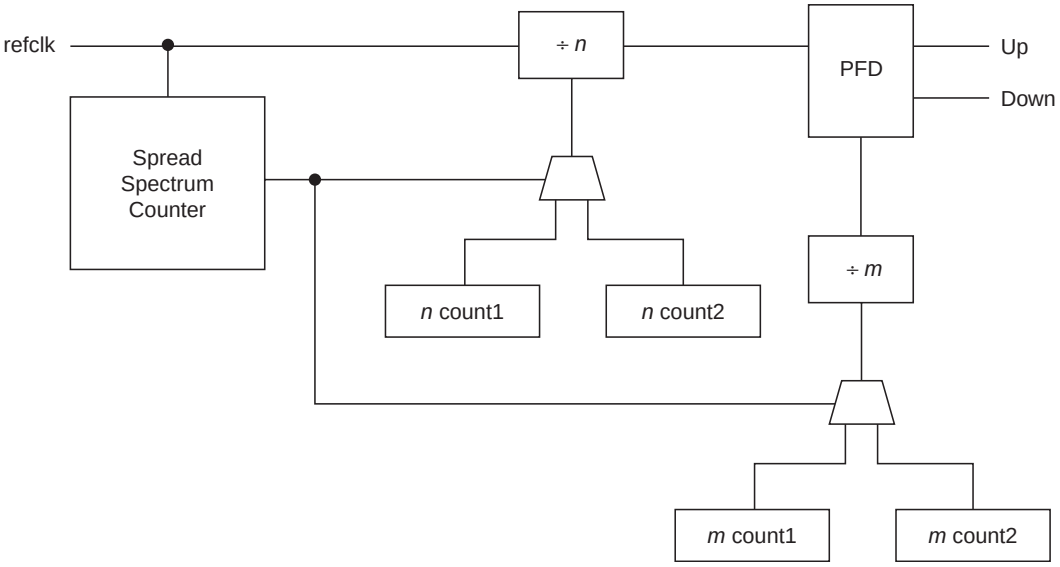


Figure 13–16 shows a VCO frequency waveform when toggling between different counter values. Since the enhanced PLL switches between two different *m* and *n* values, the result is a straight line between two frequencies, which gives a linear modulation. The magnitude of modulation is determined by the ratio of two *m*/*n* sets. The percent spread is determined by:

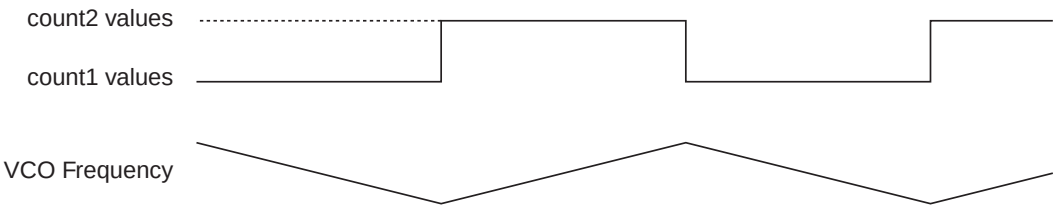
$$\text{percent spread} = (f_{VCOmax} - f_{VCOmin})/f_{VCOmax} = 1 - [(m_2 \times n_1)/(m_1 \times n_2)]$$

The maximum and minimum VCO frequency is defined as:

$$f_{VCOmax} = (m_1/n_1) \times f_{ref}$$

$$f_{VCOmin} = (m_2/n_2) \times f_{ref}$$

Figure 13–16. VCO Frequency Modulation Waveforms



Software Support

You can enter the desired down-spread percentage and modulation frequency in the MegaWizard Plug-In Manager through the Quartus II software. Alternatively, the MegaWizard Plug-In Manager can set the `downspread` parameter in the `altpll` megafunction to the desired down-spread percentage. Timing analysis ensures the design operates at the maximum spread frequency and meets all timing requirements.

For more information on PLL software support in the Quartus II software, see the *altpll Megafunction User Guide*.

Guidelines

If the design cascades PLLs, the source, or upstream PLL should have a low bandwidth setting, while the destination, or downstream PLL should have a high bandwidth setting. The upstream PLL must have a low bandwidth setting because a PLL does not generate jitter higher than its bandwidth. The downstream PLL must have a high bandwidth setting to track the jitter. The design must use the spread-spectrum feature in a low-bandwidth PLL and, therefore, the Quartus II software automatically sets the spread-spectrum PLL's bandwidth to low.

Designs cannot use spread-spectrum PLLs with the programmable bandwidth feature.

Stratix and Stratix GX devices can accept a spread-spectrum input with typical modulation frequencies. However, the device cannot automatically detect that the input is a spread-spectrum signal. Instead, the input signal looks like deterministic jitter at the input of the downstream PLL.

Spread spectrum should only have a minor effect on period jitter, but period jitter increases. Period jitter is the deviation of a clock's cycle time from its previous cycle position. Period jitter measures the variation of a clock's output transition from its ideal position over consecutive edges.

With down-spread modulation, the peak of the modulated waveform is the actual target frequency. Therefore, the system never exceeds the maximum clock speed. To maintain reliable communication, the entire system/subsystem should use the Stratix or Stratix GX device as the clock source. Communication could fail if the Stratix or Stratix GX logic array is clocked by the spread-spectrum clock, but the data it receives from another device is not.

Since spread spectrum affects the *m* counter values, all spread-spectrum PLL outputs are affected. Therefore, if only one spread-spectrum signal is needed, the clock signal should use a separate PLL without other outputs from that PLL.

No special considerations are needed when using spread spectrum with the clock switchover feature. This is because the clock switchover feature does not affect the *m* and *n* counter values, which are the counter values that are switching when using spread spectrum.

PLL Reconfiguration



See *AN 282: Implementing PLL Reconfiguration in Stratix & Stratix GX Devices* for information on PLL reconfiguration.

Enhanced PLL Pins

Table 13–9 shows the physical pins and their purpose for the Enhanced PLLs. For `inclk` port connections to pins see "Clocking" on page 13–39.

Table 13–9. Enhanced PLL Pins (Part 1 of 2)

Pin	Description
CLK4p/n	Single-ended or differential pins that can drive the inclk port for PLL 6.
CLK5p/n	Single-ended or differential pins that can drive the inclk port for PLL 6.
CLK6p/n	Single-ended or differential pins that can drive the inclk port for PLL 12.
CLK7p/n	Single-ended or differential pins that can drive the inclk port for PLL 12.
CLK12p/n	Single-ended or differential pins that can drive the inclk port for PLL 11.
CLK13p/n	Single-ended or differential pins that can drive the inclk port for PLL 11.
CLK14p/n	Single-ended or differential pins that can drive the inclk port for PLL 5.
CLK15p/n	Single-ended or differential pins that can drive the inclk port for PLL 5.
PLL5_FBp/n	Single-ended or differential pins that can drive the fbin port for PLL 5.
PLL6_FBp/n	Single-ended or differential pins that can drive the fbin port for PLL 6.
PLLENABLE	Dedicated input pin that drives the pllena port of all or a set of PLLs. If you do not use this pin, connect it to ground.

Table 13–9. Enhanced PLL Pins (Part 2 of 2)

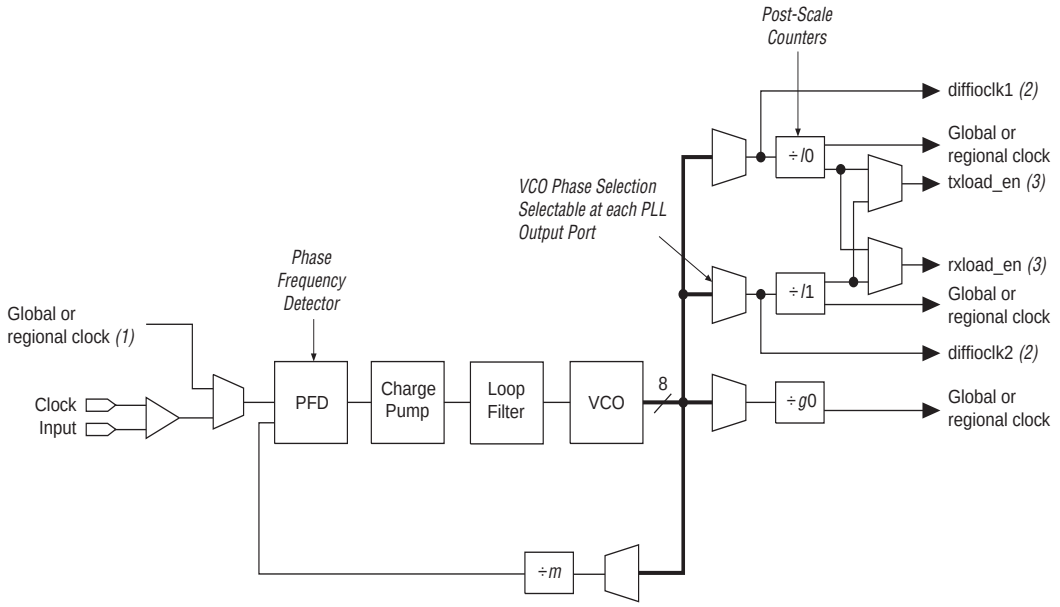
Pin	Description
PLL5_OUT[3..0]p/n	Single-ended or differential pins driven by extclk[3..0] ports from PLL 5.
PLL6_OUT[3..0]p/n	Single-ended or differential pins driven by extclk[3..0] ports from PLL 6.
PLL11_OUT, CLK13n	Single-ended output pin driven by clk0 port from PLL 11.
PLL12_OUT, CLK6n	Single-ended output pin driven by clk0 port from PLL 12.
VCCA_PLL5	Analog power for PLL 5. Connect this pin to 1.5 V, even if the PLL is not used.
VCCG_PLL5	Guard ring power for PLL 5. Connect this pin to 1.5 V, even if the PLL is not used.
GNDA_PLL5	Analog ground for PLL 5. You can connect this pin to the GND plane on the board.
GNDG_PLL5	Guard ring ground for PLL 5. You can connect this pin to the GND plane on the board.
VCCA_PLL6	Analog power for PLL 6. Connect this pin to 1.5 V, even if the PLL is not used.
VCCG_PLL6	Guard ring power for PLL 6. Connect this pin to 1.5 V, even if the PLL is not used.
GNDA_PLL6	Analog ground for PLL 6. You can connect this pin to the GND plane on the board.
GNDG_PLL6	Guard ring ground for PLL 6. You can connect this pin to the GND plane on the board.
VCCA_PLL11	Analog power for PLL 11. Connect this pin to 1.5 V, even if the PLL is not used.
VCCG_PLL11	Guard ring power for PLL 11. Connect this pin to 1.5 V, even if the PLL is not used.
GNDA_PLL11	Analog ground for PLL 11. You can connect this pin to the GND plane on the board.
GNDG_PLL11	Guard ring ground for PLL 11.You can connect this pin to the GND plane on the board.
VCCA_PLL12	Analog power for PLL 12. Connect this pin to 1.5 V, even if the PLL is not used.
VCCG_PLL12	Guard ring power for PLL 12. Connect this pin to 1.5 V, even if the PLL is not used.
GNDA_PLL12	Analog ground for PLL 12. You can connect this pin to the GND plane on the board.
GNDG_PLL12	Guard ring ground for PLL 12. You can connect this pin to the GND plane on the board.
VCC_PLL5_OUTA	External clock output V_{CCIO} power for PLL5_OUT0p, PLL5_OUT0n, PLL5_OUT1p, and PLL5_OUT1n outputs from PLL 5.
VCC_PLL5_OUTB	External clock output V_{CCIO} power for PLL5_OUT2p, PLL5_OUT2n, PLL5_OUT3p, and PLL5_OUT3n outputs from PLL 5.
VCC_PLL6_OUTA	External clock output V_{CCIO} power for PLL5_OUT0p, PLL5_OUT0n, PLL5_OUT1p, and PLL5_OUT1n outputs from PLL 6.
VCC_PLL6_OUTB	External clock output V_{CCIO} power for PLL5_OUT2p, PLL5_OUT2n, PLL5_OUT3p, and PLL5_OUT3n outputs from PLL 6.

Fast PLLs

Stratix devices contain up to eight fast PLLs and Stratix GX devices contain up to four fast PLLs. Both device PLLs have high-speed differential I/O interface ability along with general-purpose features. Figure 13–17 shows a diagram of the fast PLL. This section discusses the

general purpose abilities of the Fast PLL. For information on the high-speed differential I/O interface capabilities, see the *High-Speed Differential I/O Interfaces in Stratix Devices* chapter.

Figure 13–17. Stratix & Stratix GX Fast PLL Block Diagram



Notes to Figure 13–17:

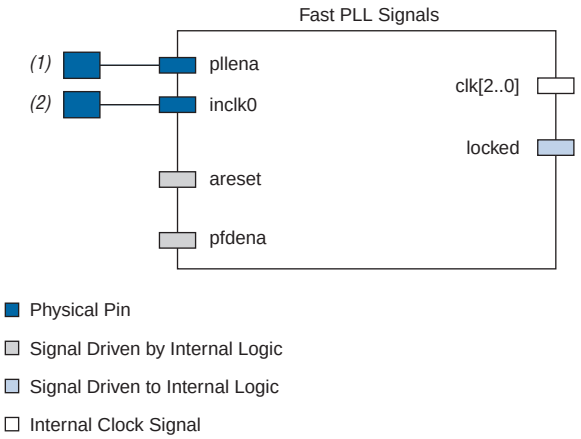
(1) The global or regional clock input can be driven by an output from another PLL or any dedicated CLK or FCLK pin. It cannot be driven by internally-generated global signals.

(2) In high-speed differential I/O support mode, this high-speed PLL clock feeds the SERDES. Stratix and Stratix GX devices only support one rate of data transfer per fast PLL in high-speed differential I/O support mode.

(3) This signal is a high-speed differential I/O support SERDES control signal.

Figure 13–18 shows all possible ports related to fast PLLs.

Figure 13–18. Fast PLL Ports & Physical Destinations



Notes to Figure 13–18:
(1) This input pin is shared by all enhanced and fast PLLs.
(2) This input pin is either single-ended or differential.

Tables 13–10 and 13–11 show the description of all fast PLL ports.

Table 13–10. Fast PLL Input Signals

Name	Description	Source	Destination
inclk1	Reference clock input to PLL	Pin	PFD
pllena	Enable pin for enabling or disabling all or a set of PLLs – active high	Pin	PLL control signal
areset	Signal used to reset the PLL which re-synchronizes all the counter outputs—active high	Logic array	PLL control signal
pfdena	Enables the up/down outputs from the phase-frequency detector—active high	Logic array	PFD

Table 13–11. Fast PLL Output Signals

Name	Description	Source	Destination
clk[2..0]	PLL outputs driving regional or global clock	PLL counter	Internal clock
locked	Lock output from lock detect circuit—active high	PLL lock detect	Logic array

Clock Multiplication & Division

Stratix and Stratix GX device fast PLLs provide clock synthesis for PLL output ports using *m*/(post scaler) scaling factors. The input clock is multiplied by the *m* feedback factor. Each output port has a unique post scale counter to divide down the high-frequency VCO. There is one multiply counter, *m*, per fast PLL with a range of 1 to 32. There are three post-scale counters (*g*0, *l*0, and *l*1) for the regional and global clock output ports. All post-scale counters range from 1 to 32. If the design uses a high-speed serial interface, you can set the output counter to 1 to allow the high-speed VCO frequency to drive the SERDES.

External Clock Outputs

Each fast PLL supports differential or single-ended outputs for source-synchronous transmitters or for general-purpose external clocks. There are no dedicated external clock output pins. The fast PLL global or regional outputs can drive any I/O pin as an external clock output pin. The I/O standards supported by any particular bank determines what standards are possible for an external clock output driven by the fast PLL in that bank. See the *Selectable I/O Standards in Stratix & Stratix GX Devices* chapter in the *Stratix Device Handbook, Volume 2* or the *Stratix GX Device Handbook, Volume 2* for output standard support.

Table 13–12 shows the I/O standards supported by fast PLL input pins.

Table 13–12. Fast PLL Port I/O Standards (Part 1 of 2)

I/O Standard	Input	
	INCLK	PLLENABLE
LVTTL	✓	✓
LVCMOS	✓	✓
2.5 V	✓	
1.8 V	✓	
1.5 V	✓	
3.3-V PCI		
3.3-V PCI-X 1.0		
LVPECL	✓	
PCML	✓	
LVDS	✓	
HyperTransport technology	✓	
Differential HSTL	✓	

Table 13–12. Fast PLL Port I/O Standards (Part 2 of 2)

I/O Standard	Input	
	INCLK	PLLENABLE
Differential SSTL		
3.3-V GTL		
3.3-V GTL+	✓	
1.5-V HSTL Class I	✓	
1.5-V HSTL Class II		
1.8-V HSTL Class I	✓	
1.8-V HSTL Class II		
SSTL-18 Class I	✓	
SSTL-18 Class II		
SSTL-2 Class I	✓	
SSTL-2 Class II	✓	
SSTL-3 Class I	✓	
SSTL-3 Class II	✓	
AGP (1× and 2×)		
CTT	✓	

Phase Shifting

Stratix and Stratix GX device fast PLLs have advanced clock shift ability to provide programmable phase shift. These parameters are set in the Quartus II software.

The Quartus II software automatically sets the phase taps and counter settings according to the phase shift entry. Enter a desired phase shift and the Quartus II software automatically sets the closest setting achievable. This type of phase shift is not reconfigurable during system operation. You can enter a phase shift (in degrees or time units) for each PLL clock output port or for all outputs together in one shift. You can perform phase shifting in time units with a resolution range of 125 to 416.66 ps to create a function of frequency input and the multiplication and division factors (that is, it is a function of the VCO period), with the finest step being equal to an eighth (×0.125) of the VCO period. Each clock output counter can choose a different phase of the VCO period from up to eight taps for individual fine-step selection. Also, each clock output counter can use a unique initial count setting to achieve individual coarse shift selection in steps of one VCO period. The combination of coarse and grain shifts allows phase shifting for the entire input clock period.

The equation to determine the precision of phase in degrees is: 45° ÷ post-scale counter value. Therefore, the maximum step size is 45°, and smaller steps are possible depending on the multiplication and division ratio necessary on the output counter port.

This type of phase shift provides the highest precision since it is the least sensitive to process, supply, and temperature variation.

Programmable Duty Cycle

The programmable duty cycle allows the fast PLL to generate clock outputs with a variable duty cycle. This feature is supported on each fast PLL post-scale counter. *g*0, *l*0, and *l*1 all support programmable duty. You use a low- and high-time count setting for the post-scale counters to set the duty cycle.

The Quartus II software uses the frequency input and multiply/divide rate desired to select the post-scale counter, which determines the possible choices for each duty cycle. The precision of the duty cycle is determined by the post-scale counter value chosen on an output. The precision is defined by 50% divided by the post-scale counter value. The closest value to 100% is not achievable for a given counter value. For example, if the *g*0 counter is 10, then steps of 5% are possible for duty cycle choices between 5 to 90%.

If the device uses external feedback, you must set the duty cycle for the counter driving off the device to 50%.

Control Signals

The lock output indicates a stable clock output signal in phase with the reference clock. Unlike enhanced PLLs, fast PLLs do not have a lock filter counter.

The `pllenable` pin is a dedicated pin that enables/disables both PLLs. When the `pllenable` pin is low, the clock output ports are driven by `GND` and all the PLLs go out of lock. When the `pllenable` pin goes high again, the PLLs relock and resynchronize to the input clocks. You can choose which PLLs are controlled by the `pllenable` by connecting the `pllenable` input port of the `altpll` megafunction to the common `pllenable` input pin.

The `areset` signals are reset/resynchronization inputs for each fast PLL. The Stratix and Stratix GX devices can drive these input signals from an input pin or from LEs. When driven high, the PLL counters reset, clearing the PLL output and placing the PLL out of lock. The VCO sets back to its nominal setting (~700 MHz). When driven low again, the PLL

resynchronizes to its input clock as it relocks. If the target VCO frequency is below this nominal frequency, then the output frequency starts at a higher value then desired as it locks.

The pfdena signals control the PFD output with a programmable gate. If you disable the PFD, the VCO operates at its last set value of control voltage and frequency with some long-term drift to a lower frequency. The system continues running when the PLL goes out of lock or the input clock disables. By maintaining the last locked frequency, the system has time to store its current settings before shutting down.

If the PLL loses lock for any reason (for example, because of excessive inclk jitter, clock switchover, PLL reconfiguration, or power supply noise), the PLL must be reset with the areset signal to guarantee correct phase relationship between the PLL output clocks. If the phase relationship between the input clock and the output clock and between different output clocks from the PLL is not important in your design, it is not necessary to reset the PLL.

Pins

Table 13–13 shows the physical pins and their purpose for the Fast PLLs. For inclk port connections to pins see "Clocking" on page 13–39.

Table 13–13. Fast PLL Pins (Part 1 of 3)

Pin	Description
CLK0p/n	Single-ended or differential pins that can drive the inclk port for PLL 1 or 7.
CLK1p/n	Single-ended or differential pins that can drive the inclk port for PLL 1.
CLK2p/n	Single-ended or differential pins that can drive the inclk port for PLL 2 or 8.
CLK3p/n	Single-ended or differential pins that can drive the inclk port for PLL 2.
CLK8p/n	Single-ended or differential pins that can drive the inclk port for PLL 3 or 9. *(1)*
CLK9p/n	Single-ended or differential pins that can drive the inclk port for PLL 3. *(1)*
CLK10p/n	Single-ended or differential pins that can drive the inclk port for PLL 4 or 10. *(1)*
CLK11p/n	Single-ended or differential pins that can drive the inclk port for PLL 4. *(1)*
FPLL7CLKp/n	Single-ended or differential pins that can drive the inclk port for PLL 7.
FPLL8CLKp/n	Single-ended or differential pins that can drive the inclk port for PLL 8.
FPLL9CLKp/n	Single-ended or differential pins that can drive the inclk port for PLL 9. *(1)*
FPLL10CLKp/n	Single-ended or differential pins that can drive the inclk port for PLL 10. *(1)*
PLLENABLE	Dedicated input pin that drives the pllena port of all or a set of PLLs. If you do not use this pin, connect it to ground.
VCCA_PLL1	Analog power for PLL 1. Connect this pin to 1.5 V, even if the PLL is not used.

Table 13–13. Fast PLL Pins (Part 2 of 3)

Pin	Description
VCCG_PLL1	Guard ring power for PLL 1. Connect this pin to 1.5 V, even if the PLL is not used.
GNDA_PLL1	Analog ground for PLL 1. You can connect this pin to the GND plane on the board.
GNDG_PLL1	Guard ring ground for PLL 1. You can connect this pin to the GND plane on the board.
VCCA_PLL2	Analog power for PLL 2. Connect this pin to 1.5 V, even if the PLL is not used.
VCCG_PLL2	Guard ring power for PLL 2. Connect this pin to1.5 V, even if the PLL is not used.
GNDA_PLL2	Analog ground for PLL 2. You can connect this pin to the GND plane on the board.
GNDG_PLL2	Guard ring ground for PLL 2. You can connect this pin to the GND plane on the board.
VCCA_PLL3	Analog power for PLL 3. Connect this pin to 1.5 V, even if the PLL is not used. *(1)*
VCCG_PLL3	Guard ring power for PLL 3. Connect this pin to 1.5 V, even if the PLL is not used. *(1)*
GNDA_PLL3	Analog ground for PLL 3. You can connect this pin to the GND plane on the board. *(1)*
GNDG_PLL3	Guard ring ground for PLL 3. You can connect this pin to the GND plane on the board. *(1)*
VCCA_PLL4	Analog power for PLL 4. Connect this pin to 1.5 V, even if the PLL is not used. *(1)*
VCCG_PLL4	Guard ring power for PLL 4. Connect this pin to 1.5 V, even if the PLL is not used. *(1)*
GNDA_PLL4	Analog ground for PLL 4. You can connect this pin to the GND plane on the board. *(1)*
GNDG_PLL4	Guard ring ground for PLL 4. You can connect this pin to the GND plane on the board. *(1)*
VCCA_PLL7	Analog power for PLL 7. Connect this pin to 1.5 V, even if the PLL is not used.
VCCG_PLL7	Guard ring power for PLL 7. Connect this pin to 1.5 V, even if the PLL is not used.
GNDA_PLL7	Analog ground for PLL 7. You can connect this pin to the GND plane on the board.
GNDG_PLL7	Guard ring ground for PLL 7. You can connect this pin to the GND plane on the board.
VCCA_PLL8	Analog power for PLL 8. Connect this pin to 1.5 V, even if the PLL is not used.
VCCG_PLL8	Guard ring power for PLL 8. Connect this pin to 1.5 V, even if the PLL is not used.
GNDA_PLL8	Analog ground for PLL 8. You can connect this pin to the GND plane on the board.
GNDG_PLL8	Guard ring ground for PLL 8. You can connect this pin to the GND plane on the board.
VCCA_PLL9	Analog power for PLL 9. Connect this pin to 1.5 V, even if the PLL is not used. *(1)*
VCCG_PLL9	Guard ring power for PLL 9. Connect this pin to 1.5 V, even if the PLL is not used. *(1)*

Table 13–13. Fast PLL Pins (Part 3 of 3)

Pin	Description
GNDA_PLL9	Analog ground for PLL 9. You can connect this pin to the GND plane on the board. *(1)*
GNDG_PLL9	Guard ring ground for PLL 9. You can connect this pin to the GND plane on the board. *(1)*
VCCA_PLL10	Analog power for PLL 10. Connect this pin to 1.5 V, even if the PLL is not used. *(1)*
VCCG_PLL10	Guard ring power for PLL 10. Connect this pin to 1.5 V, even if the PLL is not used. *(1)*
GNDA_PLL10	Analog ground for PLL 10. Connect this pin to the GND plane on the board. *(1)*
GNDG_PLL10	Guard ring ground for PLL 10. You can connect this pin to the GND plane on the board. *(1)*

Note to Table 13–13:

(1) PLLs 3, 4, 9, and 10 are not available on Stratix GX devices for general-purpose configuration. These PLLs are part of the HSSI block. See *AN 236: Using Source-Synchronous Signaling with DPA in Stratix GX Devices* for more information.

Clocking

Stratix and Stratix GX devices provide a hierarchical clock structure and multiple PLLs with advanced features. The large number of clocking resources in combination with the clock synthesis precision provided by enhanced and fast PLLs provides a complete clock management solution.

Global & Hierarchical Clocking

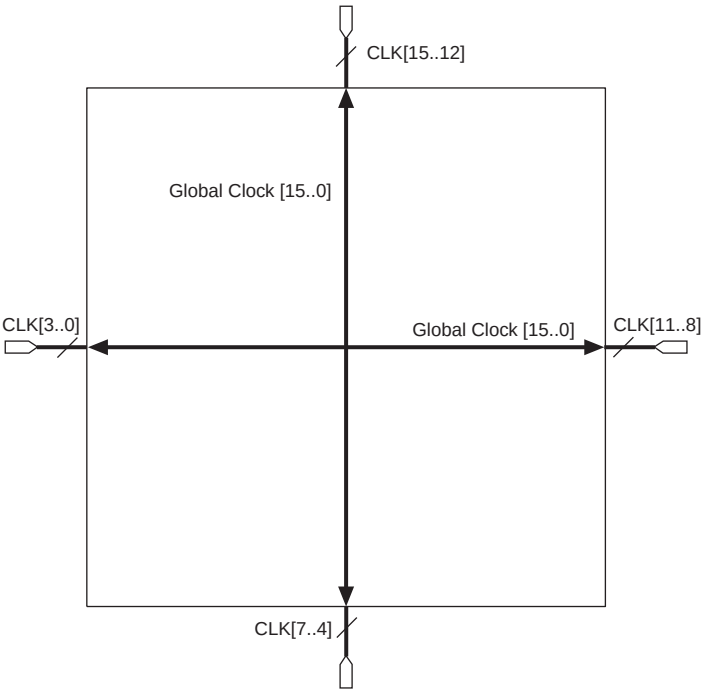
Stratix and Stratix GX devices provide 16 dedicated global clock networks, 16 regional clock networks (4 per device quadrant), and 8 dedicated fast regional clock networks. These clocks are organized into a hierarchical clock structure that allows for up to 22 clocks per device region with low skew and delay. This hierarchical clocking scheme provides up to 48 unique clock domains within Stratix and Stratix GX devices.

There are 16 dedicated clock pins (CLK[15..0]) on Stratix devices and 12 dedicated clock pins (CLK[11..0]) on Stratix GX devices to drive either the global or regional clock networks. Four clock pins drive each side of the Stratix device, as shown in Figures 13–19 and 13–20. On Stratix GX devices, four clock pins drive the top, left, and bottom sides of the device. The clocks on the right side of the device are not available for general-purpose PLLs. Enhanced and fast PLL outputs can also drive the global and regional clock networks.

Global Clock Network

These clocks drive throughout the entire device, feeding all device quadrants. All resources within the device—IOEs, LEs, DSP blocks, and all memory blocks—can use the global clock networks as clock sources. These resources can also be used for control signals, such as clock enables and synchronous or asynchronous clears fed from the external pin. Internal logic can also drive the global clock networks for internally generated global clocks and asynchronous clears, clock enables, or other control signals with large fanout. Figure 13–19 shows the 16 dedicated CLK pins driving global clock networks.

Figure 13–19. Global Clocking

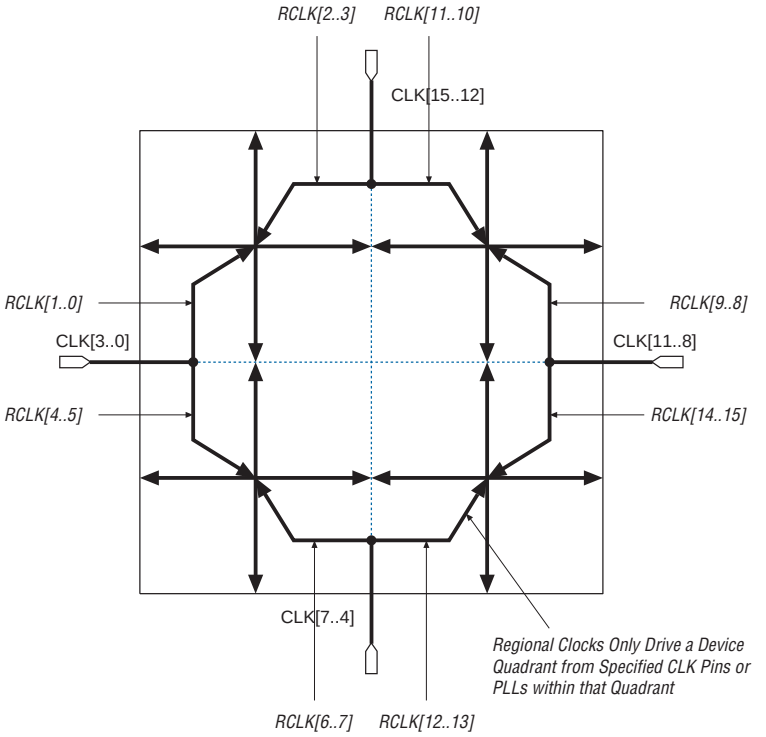


Regional Clock Network

There are four regional clock networks within each quadrant of the Stratix or Stratix GX device that are driven by the same dedicated CLK[15..0] input pins or from PLL outputs. From a top view of the silicon, RCLK[0..3] are in the top-left quadrant, RCLK[8..11] are in the top-right quadrant, RCLK[4..7] are in the bottom-left quadrant, and

RCLK[12..15] are in the bottom-right quadrant. The regional clock networks only pertain to the quadrant they drive into. The regional clock networks provide the lowest clock delay and skew for logic contained within a single quadrant. RCLK clock networks cannot be driven by internal logic. The CLK clock pins symmetrically drive the RCLK networks within a particular quadrant, as shown in Figure 13–20. See Figures 13–21 and 13–22 for RCLK connections from PLLs and CLK pins.

Figure 13–20. Regional Clocks



Clock Input Connections

Two CLK pins drive each enhanced PLL. You can use either one or both pins for clock switchover inputs into the PLL. Either pin can be the primary clock source for clock switchover, which is controlled in the Quartus II software. Enhanced PLLs 5 and 6 also have feedback input pins as shown in Table 13–14.

Input clocks for fast PLLs 1, 2, 3, and 4 come from CLK pins. Stratix GX devices use PLLs 3 and 4 in the HSSI block only. A multiplexer chooses one of two possible CLK pins to drive each PLL. This multiplexer is not a clock switchover multiplexer and is only used for clock input connectivity.

Either a FPLLCLK input pin or a CLK pin can drive the fast PLLs in the corners (7, 8, 9, and 10) when used for general purpose. CLK pins cannot drive these fast PLLs in high-speed differential I/O mode. PLLs 9 and 10 are used for the HSSI block in Stratix GX devices and are not available.

Table 13–14 shows which PLLs are available for each Stratix device and which input clock pin drives which PLLs.

Table 13–14. Stratix Clock Input Sources For Enhanced & Fast PLLs (Part 1 of 2)

Clock Input Pins	All Stratix Devices						EP1S30, EP1S40, EP1S60 & EP1S80 Devices Only				EP1S40 (3), EP1S60 & EP1S80 Devices Only	
	PLL 1 (1)	PLL 2 (1)	PLL 3 (1)	PLL 4 (1)	PLL 5 (2)	PLL 6 (2)	PLL 7 (1)	PLL 8 (1)	PLL 9 (1)	PLL 10 (1)	PLL 11 (2)	PLL 12 (2)
CLK0p/n	✓						✓					
CLK1p/n	✓											
CLK2p/n		✓						✓				
CLK3p/n		✓										
CLK4p/n						✓						
CLK5p/n						✓						
CLK6p/n												✓
CLK7p/n												✓
CLK8p/n			✓						✓			
CLK9p/n			✓									
CLK10p/n				✓						✓		
CLK11p/n				✓								
CLK12p/n											✓	
CLK13p/n											✓	
CLK14p/n					✓							
CLK15p/n					✓							

Table 13–14. Stratix Clock Input Sources For Enhanced & Fast PLLs (Part 2 of 2)

Clock Input Pins	All Stratix Devices						EP1S30, EP1S40, EP1S60 & EP1S80 Devices Only				EP1S40 (3), EP1S60 & EP1S80 Devices Only	
	PLL 1 (1)	PLL 2 (1)	PLL 3 (1)	PLL 4 (1)	PLL 5 (2)	PLL 6 (2)	PLL 7 (1)	PLL 8 (1)	PLL 9 (1)	PLL 10 (1)	PLL 11 (2)	PLL 12 (2)
FPll7clk							✓					
FPll8clk								✓				
FPll9clk									✓			
FPll10clk										✓		
Clock Feedback Input Pins												
Pll5_fbp/n					✓							
Pll6_fbp/n						✓						

Notes to Table 13–14:

(1) This is a fast PLL. The global or regional clocks in a fast PLL's quadrant can drive the fast PLL input. A pin or other PLL must drive the global or regional source. The source cannot be driven by internally generated logic before driving the fast PLL.

(2) This is an enhanced PLL.

(3) The EP1S40 device in the F780 package does not support PLLs 11 and 12.

Clock Output Connections

Enhanced PLLs have outputs for two regional clock outputs and four global outputs. There is line sharing between clock pins, global and regional clock networks and all PLL outputs. Check Tables 13–15 and 13–16 and Figures 13–21 and 13–22 to make sure that the clocking scheme is valid. The Quartus II software automatically maps to regional and global clocks to avoid any restrictions. Enhanced PLLs 5 and 6 drive out to single-ended pins as shown in Table 13–15. PLLs 11 and 12 drive out to single-ended pins.

You can connect each fast PLL 1, 2, 3, or 4 outputs (*g*0, *l*0, and *l*1) to either a global or a regional clock. (PLLs 3 and 4 are not available on Stratix GX devices.) There is line sharing between clock pins, FPLLCLK pins, global and regional clock networks and all PLL outputs. Check Figures 13–21 and 13–22 to make sure that the clocking is valid. The Quartus II software automatically maps to regional and global clocks to avoid any restrictions.

Table 13–15 shows the global and regional clocks that each PLL drives outputs to for Stratix devices. Table 13–16 shows the global and regional clock network each of the CLK and FPLLCLK pins drive when bypassing the PLL.

Table 13–15. Stratix Global & Regional Clock Output Line Sharing for Enhanced & Fast PLLs (Part 1 of 2)

Clock Network	All Devices						EP1S30, EP1S40, EP1S60 & EP1S80 Devices Only				EP1S40 (5), EP1S60 & EP1S80 Devices Only	
	PLL 1 (1)	PLL 2 (1)	PLL 3 (1)	PLL 4 (1)	PLL 5 (2)	PLL 6 (2)	PLL 7 (1)	PLL 8 (1)	PLL 9 (1)	PLL 10 (1)	PLL 11 (2)	PLL 12 (2)
GCLK0	✓	✓					✓	✓				
GCLK1	✓	✓					✓	✓				
GCLK2	✓	✓					✓	✓				
GCLK3	✓	✓					✓	✓				
GCLK4						✓						✓
GCLK5						✓						✓
GCLK6						✓						✓
GCLK7						✓						✓
GCLK8			✓	✓					✓	✓		
GCLK9			✓	✓					✓	✓		
GCLK10			✓	✓					✓	✓		
GCLK11			✓	✓					✓	✓		
GCLK12					✓						✓	
GCLK13					✓						✓	
GCLK14					✓						✓	
GCLK15					✓						✓	
RCLK0	✓	✓					✓					
RCLK1	✓	✓					✓					
RCLK2					✓						✓	
RCLK3					✓						✓	
RCLK4	✓	✓						✓				
RCLK5	✓	✓						✓				

Table 13–15. Stratix Global & Regional Clock Output Line Sharing for Enhanced & Fast PLLs (Part 2 of 2)

Clock Network	All Devices						EP1S30, EP1S40, EP1S60 & EP1S80 Devices Only				EP1S40 (5), EP1S60 & EP1S80 Devices Only	
	PLL 1 (1)	PLL 2 (1)	PLL 3 (1)	PLL 4 (1)	PLL 5 (2)	PLL 6 (2)	PLL 7 (1)	PLL 8 (1)	PLL 9 (1)	PLL 10 (1)	PLL 11 (2)	PLL 12 (2)
RCLK6						✓						✓
RCLK7						✓						✓
RCLK8			✓	✓						✓		
RCLK9			✓	✓						✓		
RCLK10					✓						✓	
RCLK11					✓						✓	
RCLK12						✓						✓
RCLK13						✓						✓
RCLK14			✓	✓					✓			
RCLK15			✓	✓					✓			
External Clock Output												
PLL5_OUT [3..0]p/n					✓							
PLL6_OUT [3..0]p/n						✓						
PLL11_OUT (3)											✓	
PLL12_OUT (4)												✓

Notes to Table 13–15:

(1) This is a fast PLL.
(2) This is an enhanced PLL.
(3) This pin is a tri-purpose pin; it can be an I/O pin, CLK13n, or used for PLL 11 output.
(4) This pin is a tri-purpose pin; it can be an I/O pin, CLK7n, or used for PLL 12 output.
(5) The EP1S40 device in the F780 package does not support PLLs 11 and 12.

Table 13–16. Stratix CLK & FPLLCLK Input Pin Connections to Global & Regional Clock Networks Note (1)

Clock Network	CLK Pins															FPLLCLK (2)			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	7	8	9	10
GCLK0	✓																✓	✓		
GCLK1		✓															✓	✓		
GCLK2			✓														✓	✓		
GCLK3				✓													✓	✓		
GCLK4					✓															
GCLK5						✓														
GCLK6							✓													
GCLK7								✓												
GCLK8									✓										✓	✓
GCLK9										✓									✓	✓
GCLK10											✓								✓	✓
GCLK11												✓							✓	✓
GCLK12													✓							
GCLK13														✓						
GCLK14															✓					
GCLK15																✓				
RCLK0	✓																✓			
RCLK1		✓															✓			
RCLK2															✓			✓		
RCLK3																✓		✓		
RCLK4			✓																	
RCLK5				✓																
RCLK6					✓															
RCLK7						✓														
RCLK8											✓								✓	
RCLK9												✓							✓	
RCLK10													✓							✓
RCLK11														✓						✓
RCLK12							✓													

Table 13–16. Stratix CLK & FPLLCLK Input Pin Connections to Global & Regional Clock Networks Note (1)

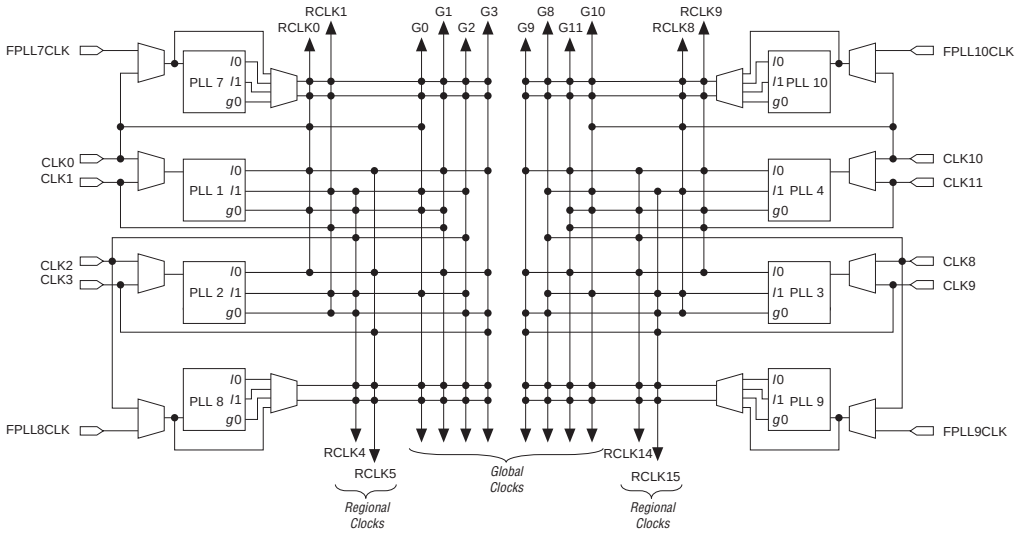
Clock Network	CLK Pins															FPLLCLK (2)			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	7	8	9	10
RCLK13								✓												
RCLK14									✓											
RCLK15										✓										

Notes to Table 13–16:
(1) The CLK and FPLLCLK pins cannot drive.
(2) The FPLLCLK pin is only available in EP1S80, EP1S60, EP1S40, and EP1S30 devices.

The fast PLLs also drive high-speed SERDES clocks for differential I/O interfacing. For information on these FPLLCLK pins, see the *High-Speed Differential I/O Interfaces in Stratix Devices* chapter.

Figure 13–21 shows the global and regional clock input and output connections from the enhanced. Figure 13–21 shows graphically the same information as Tables 13–15 and 13–16 but with the added detail of where each specific PLL output port drives to.

Figure 13–21. Global & Regional Clock Connections from Side Clock Pins & Fast PLL Outputs



Notes to Figures 13–21:

(1) The global or regional clocks in a fast PLL's quadrant can drive the fast PLL input. A dedicated pin or other PLL must drive the global or regional source. The source cannot be driven by internally generated logic before driving the fast PLL.

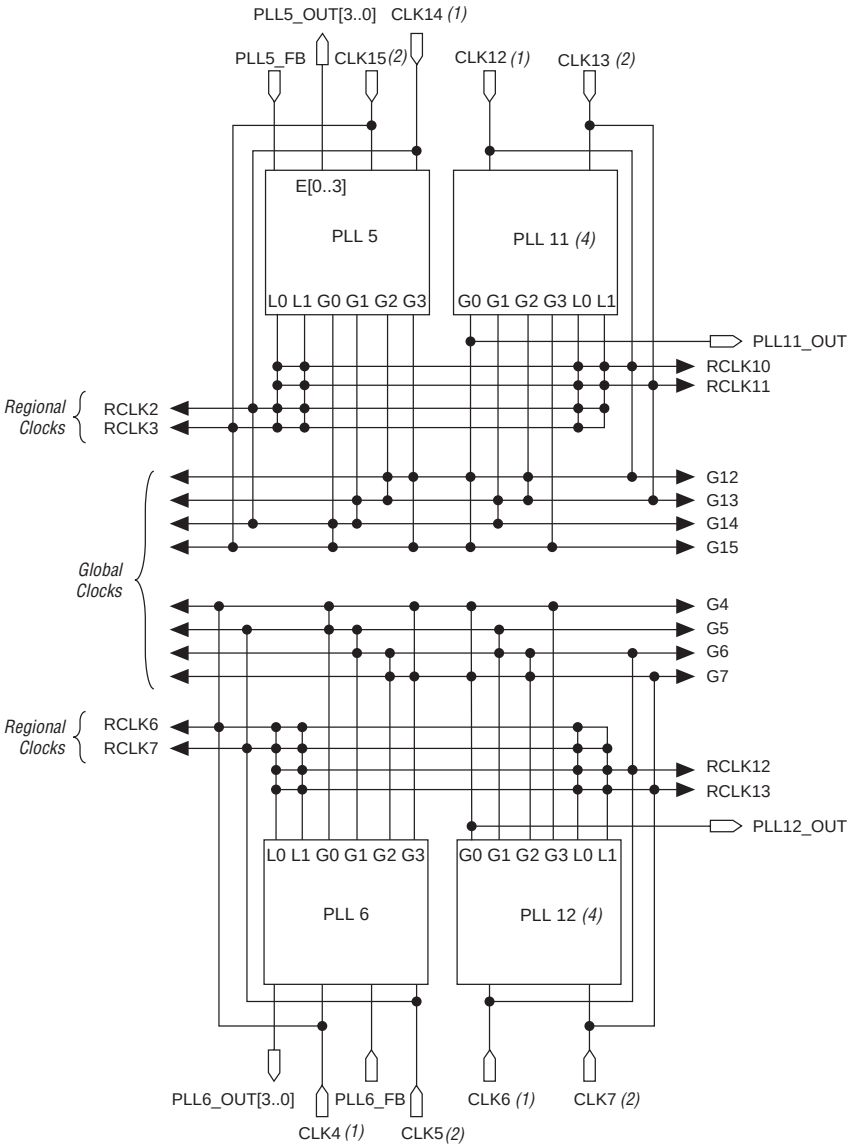
(2) PLLs 3, 4, 9, and 10 are used for the HSSI block in Stratix GX devices and are not available for this use.

When using a fast PLL to compensate for clock delays to drive logic on the chip, the clock delay from the input pin to the clock input port of the PLL is compensated only if the clock is fed by the dedicated input pin closest to the PLL. If the fast PLL gets its input clock from a global or regional clock or from another dedicated clock pin, which does not directly feed the fast PLL, the clock signal is first routed onto a global clock network. The signal then drives into the PLL. In this case, the clock delay is not fully compensated and the delay compensation is equal to the clock delay from the dedicated clock pin closest to the PLL to the clock input port of the PLL.

For example, if you use `CLK0` to feed PLL 7, the input clock path delay is not fully compensated, but if `FPLL7CLK` feeds PLL 7, the input clock path delay is fully compensated.

Figure 13–22 shows the global and regional clock input and output connections from the fast PLLs. Figure 13–22 shows graphically the same information as Tables 13–15 and 13–16 but with the added detail of where each specific PLL output port drives to.

Figure 13–22. Global & Regional Clock Connections from Top Clock Pins & Enhanced PLL Outputs



Notes to Figures 13–22:

(1) CLK4, CLK6, CLK12, and CLK14 feed the corresponding PLL's inclk0 port.

(2) CLK5, CLK7, CLK13, and CLK15 feed the corresponding PLL's inclk1 port.

Board Layout

The enhanced and fast PLL circuits in Stratix and Stratix GX devices contain analog components embedded in a digital device. These analog components have separate power and ground pins to minimize noise generated by the digital components. Both Stratix and Stratix GX enhanced and fast PLLs use separate VCC and ground pins to isolate circuitry and improve noise resistance.

VCCA & GNDA

Each enhanced and fast PLL uses separate VCC and ground pin pairs for their analog circuitry. The analog circuit power and ground pin for each PLL is called PLL<*PLL number*>_VCCA and PLL<*PLL number*>_GNDA. Connect the VCCA power pin to a 1.5-V power supply, even if you do not use the PLL. Isolate the power connected to VCCA from the power to the rest of the Stratix and Stratix GX device or any other digital device on the board. You can use one of three different methods of isolating the VCCA pin: separate V_{CCA} power planes, a partitioned V_{CCA} island within the V_{CCINT} plane, and thick V_{CCA} traces.

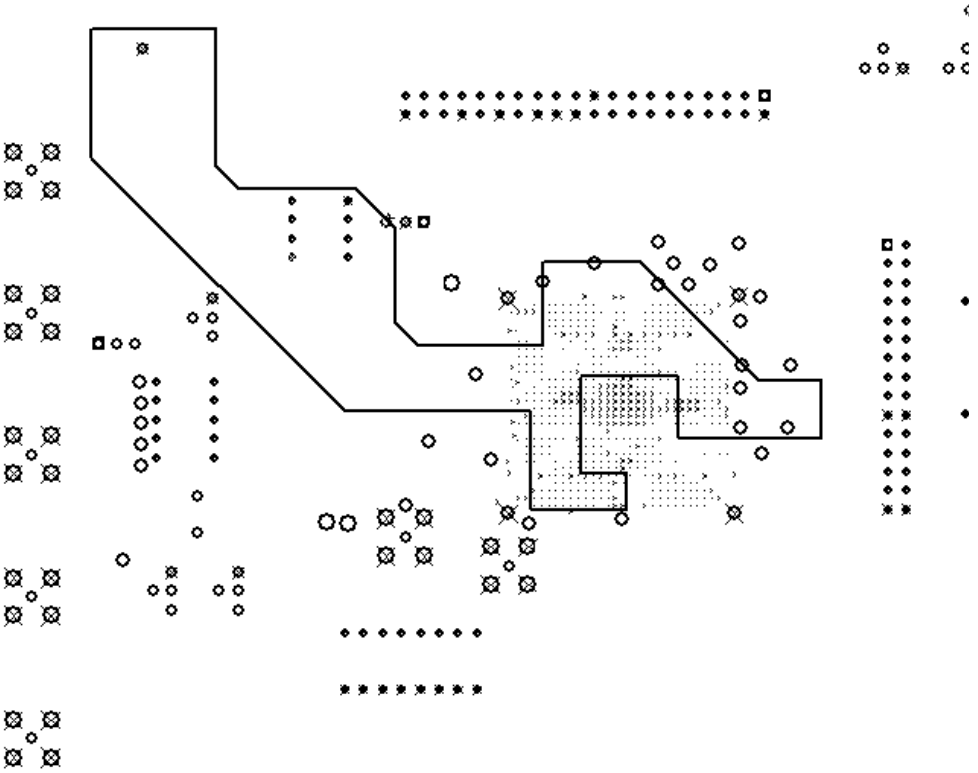
Separate V_{CCA} Power Plane

A mixed signal system is already partitioned into analog and digital sections, each with its own power planes on the board. To isolate the VCCA pin using a separate V_{CCA} power plane, connect the VCCA pin to the analog 1.5-V power plane.

Partitioned V_{CCA} Island within V_{CCINT} Plane

Fully digital systems do not have a separate analog power plane on the board. Because it is expensive to add new planes to the board, you can create islands for VCCA_PLL. Figure 13–23 shows an example board layout with an analog power island. The dielectric boundary that creates the island should be 25 mils thick. Figure 13–23 shows a partitioned plane within V_{CCINT} for V_{CCA}.

Figure 13–23. V_{CCINT} Plane Partitioned for V_{CCA} Island

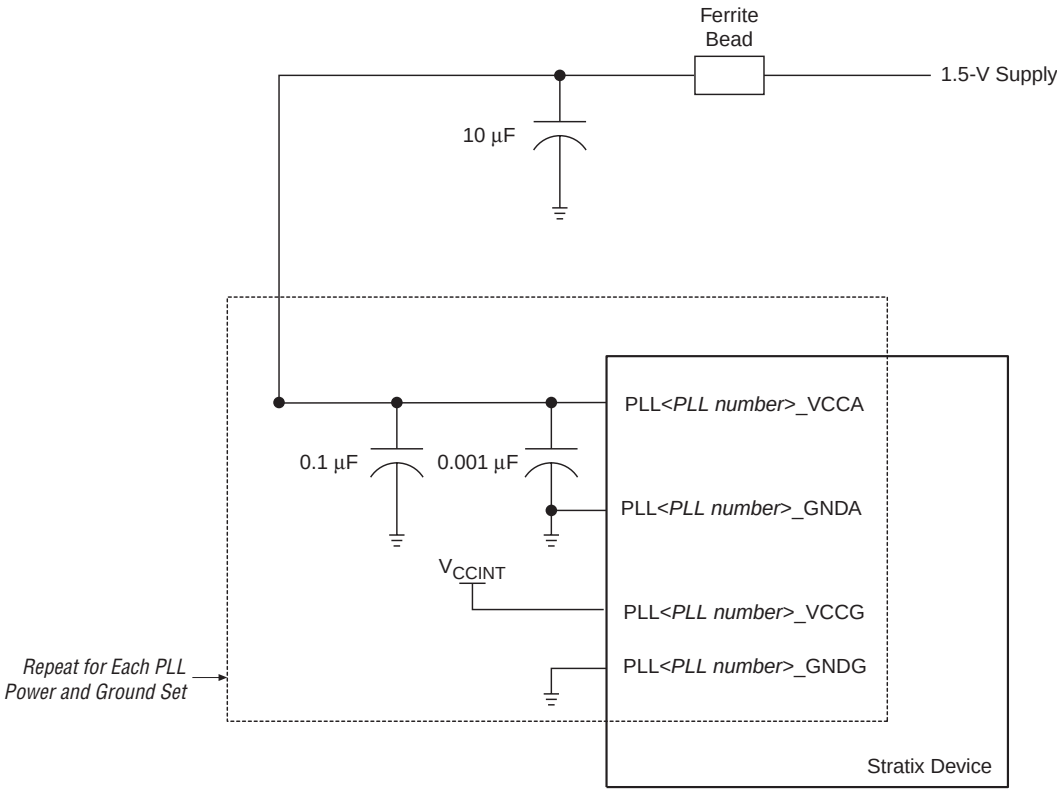


Thick V_{CCA} Trace

Because of board constraints, you might not be able to partition a V_{CCA} island. Instead, run a thick trace from the power supply to each V_{CCA} pin. The traces should be at least 20 mils thick.

In each of these three cases, you should filter each V_{CCA} pin with a decoupling circuit shown in Figure 13–24. Place a ferrite bead that exhibits high impedance at frequencies of 50 MHz or higher and a 10-µF tantalum parallel capacitor where the power enters the board. Decouple each V_{CCA} pin with a 0.1-µF and 0.001-µF parallel combination of ceramic capacitors located as close as possible to the Stratix or Stratix GX device. You can connect the GNDA pins directly to the same ground plane as the device's digital ground.

Figure 13–24. PLL Power Schematic for Stratix or Stratix GX PLLs



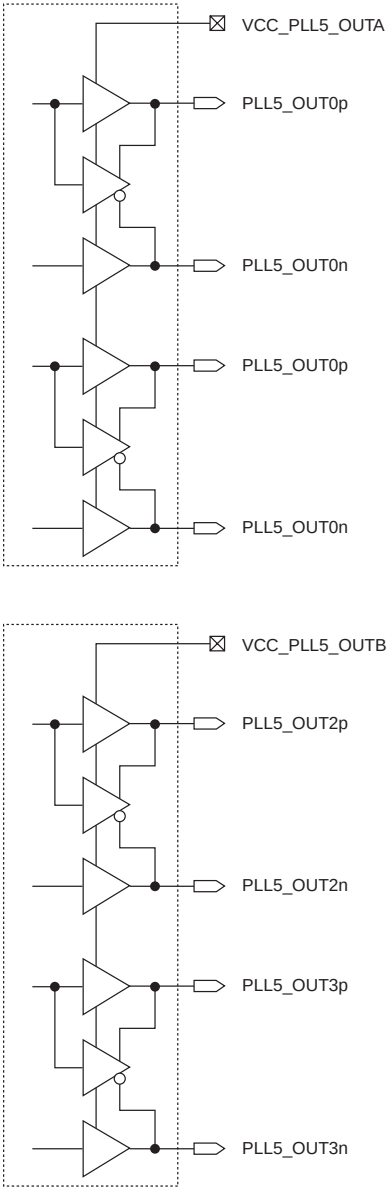
VCCG & GNDG

The guard ring power and ground pins are called PLL<*PLL number*>_VCCG and PLL<*PLL number*>_GNDG. The guard ring isolates the PLL circuit from the rest of the device. Connect these guard ring V_{CCG} pins to the quietest digital supply on the board. In most systems, this is the digital 1.5-V supply supplied to the device's V_{CCINT} pins. Connect the V_{CCG} pins to a power supply even if you do not use the PLL. You can connect the GNDG pins directly to the same ground plane as the device's digital ground. See Figure 13–24.

External Clock Output Power

Enhanced PLLs 5 and 6 also have isolated power pins for their dedicated external clock outputs (VCC_PLL5_OUTA and VCC_PLL5_OUTB, or VCC_PLL6_OUTA and VCC_PLL6_OUTB, respectively). PLLs 5 and 6 both have two banks of outputs. Each bank is powered by a unique output power, OUTA or OUTB, as illustrated in Figure 13–25. These outputs can by powered by 3.3, 2.5, 1.8, or 1.5 V depending on the I/O standard for the clock output in the A or B groups.

Figure 13–25. External Clock Output Pin Association to Output Power Note (1)

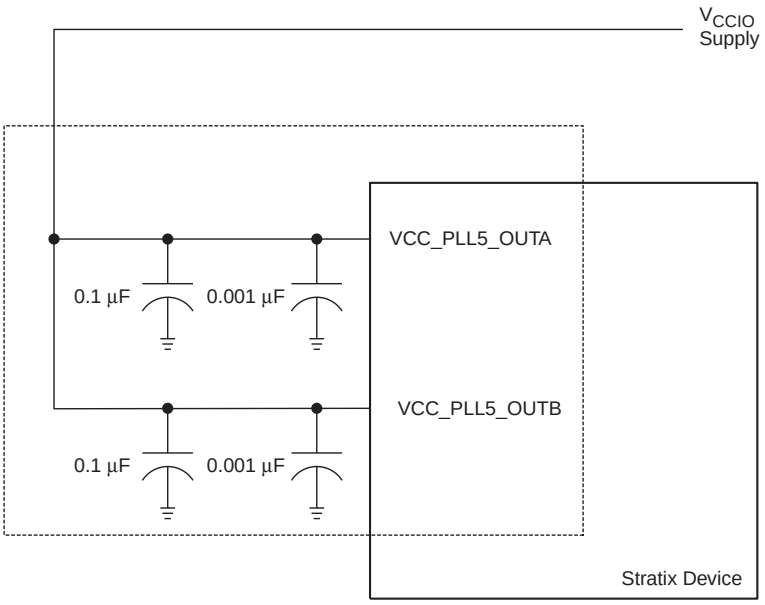


Note to Figure 13–25:

(1) These pins apply to PLL 5. The figure for PLL 6 is similar, except that the pin names begin with the prefix `PLL6` instead of `PLL5`.

Filter each isolated power pin with a decoupling circuit shown in Figure 13–26. Decouple the isolated power pins with a 0.1-µF and a 0.001-µF parallel combination of ceramic capacitors located as close as possible to the Stratix device.

Figure 13–26. Stratix PLL External Clock Output Power Ball Connections
Note (1)



Note to Figure 13–26:
(1) Figure 13–26 also applies to VCC_PLL6_OUTA/B.

Guidelines

Use the following guidelines for optimal jitter performance on the external clock outputs from enhanced PLLs 5 and 6. If all outputs are running at the same frequency, these guidelines are not necessary to improve performance.

- When driving two or more clock outputs from PLL 5 or 6, separate the outputs into the two groups shown in Figure 13–24. For example, if you are driving 100- and 200-MHz clock outputs off-chip from PLL 5, place one output on PLL5_OUT0p (powered by VCC_PLL5_OUTA) and the other output on PLL5_OUT2p (powered by VCC_PLL5_OUTB). Since the output buffers are powered by different pins, they are less susceptible to bimodal jitter. Bimodal jitter is a deterministic jitter not caused by the PLL but rather by coincident edges of clock outputs that are multiples of each other.
- Use phase shift to ensure edges are not coincident on all the clock outputs.
- Use phase shift to skew clock edges with respect to each other for best jitter performance.

  Delay shift (time delay elements) are no longer supported in Stratix PLLs. Use the phase shift feature to implement the desired time shift.

- If you cannot drive multiple clocks of different frequencies and phase shifts or isolate banks, you should control the drive capability on the lower frequency clock. Reducing how much current the output buffer has to supply can reduce the noise. Minimize capacitive load on the slower frequency output and configure the output buffer to drive slow slew rate and lower current strength. The higher frequency output should have an improved performance, but this may degrade the performance of your lower frequency clock output.

Conclusion

Stratix and Stratix GX device enhanced PLLs provide you with complete control of your clocks and system timing. These PLLs are capable of offering flexible system level clock management that was previously only available in discrete PLL devices. The embedded PLLs meet and exceed the features offered by these high-end discrete devices, reducing the need for other timing devices in the system.

Section III. Memory

This section provides information about the supported external memory interfaces and the TriMatrix™ memory structure in Stratix® GX and Stratix devices.

This section includes the following chapters:

- Chapter 14, TriMatrix Embedded Memory Blocks in Stratix & Stratix GX Devices
- Chapter 15, External Memory Interfaces in Stratix & Stratix GX Devices

Revision History

The table below shows the revision history for Chapters 14 and 15.

Chapter(s)	Date / Version	Changes Made
14	July 2005, v3.3	Chapter updated as part of the *Stratix Device Handbook* update.
	January 2005 v3.2	Added document to the *Stratix GX Device Handbook*.
15	June 2006, v3.3	Chapter updated as part of the *Stratix Device Handbook* update.
	July 2005, v3.2	Chapter updated as part of the *Stratix Device Handbook* update.
	September 2004 v3.1	Added document to the *Stratix GX Device Handbook*.

14. TriMatrix Embedded Memory Blocks in Stratix & Stratix GX Devices

S52003-3.3

Introduction

Stratix® and Stratix GX devices feature the TriMatrix™ memory structure, composed of three sizes of embedded RAM blocks. TriMatrix memory includes 512-bit M512 blocks, 4-Kbit M4K blocks, and 512-Kbit M-RAM blocks, each of which is configurable to support a wide range of features. Offering up to 10 Mbits of RAM and up to 12 terabits per second of device memory bandwidth, the TriMatrix memory structure makes the Stratix and Stratix GX families ideal for memory-intensive applications.

TriMatrix Memory

TriMatrix memory structures can implement a wide variety of complex memory functions. For example, use the small M512 blocks for first-in first-out (FIFO) functions and clock domain buffering where memory bandwidth is critical. The M4K blocks are an ideal size for applications requiring medium-sized memory, such as asynchronous transfer mode (ATM) cell processing. M-RAM blocks enhance programmable logic device (PLD) memory capabilities for large buffering applications, such as internet protocol (IP) packet buffering and system cache.

TriMatrix memory blocks support various memory configurations, including single-port, simple dual-port, true dual-port (also known as bidirectional dual-port), shift-register, ROM, and FIFO mode. The TriMatrix memory architecture also includes advanced features and capabilities, such as byte enable support, parity-bit support, and mixed-port width support. This chapter describes the various TriMatrix memory modes and features.

Table 14–1 summarizes the features supported by the three sizes of TriMatrix memory.



For more information on selecting which memory block to use, see *AN 207: TriMatrix Memory Selection Using the Quartus II Software.*

Table 14–1. Summary of TriMatrix Memory Features

Feature	M512 Block	M4K Block	M-RAM Block
Performance	319 MHz	290 MHz	287 MHz
Total RAM bits (including parity bits)	576	4,608	589,824
Configurations	512 × 1 256 × 2 128 × 4 64 × 8 64 × 9 32 × 16 32 × 18	4K × 1 2K × 2 1K × 4 512 × 8 512 × 9 256 × 16 256 × 18 128 × 32 128 × 36	64K × 8 64K × 9 32K × 16 32K × 18 16K × 32 16K × 36 8K × 64 8K × 72 4K × 128 4K × 144
Parity bits	✓	✓	✓
Byte enable		✓	✓
Single-port memory	✓	✓	✓
Simple dual-port memory	✓	✓	✓
True dual-port memory		✓	✓
Embedded shift register	✓	✓	
ROM	✓	✓	
FIFO buffer	✓	✓	✓
Simple dual-port mixed width support	✓	✓	✓
True dual-port mixed width support		✓	✓
Memory initialization file (**.mif**)	✓	✓	
Mixed-clock mode	✓	✓	✓
Power-up condition	Outputs cleared	Outputs cleared	Outputs unknown
Register clears	Input and output registers *(1)*	Input and output registers *(2)*	Output registers
Same-port read-during-write	New data available at positive clock edge	New data available at positive clock edge	New data available at positive clock edge
Mixed-port read-during-write	Outputs set to unknown or old data	Outputs set to unknown or old data	Unknown output

Notes to Table 14–1:

(1) The `rden` register on the M512 memory block does not have a clear port.

(2) On the M4K block, asserting the clear port of the `rden` and byte enable registers drives the output of these registers high.

The extremely high memory bandwidth of the Stratix and Stratix GX device families is a result of increased memory capacity and speed. Table 14–2 shows the memory capacity for TriMatrix memory blocks in each Stratix device. Table 14–3 shows the memory capacity for TriMatrix memory blocks in each Stratix GX device.

Table 14–2. TriMatrix Memory Distribution in Stratix Devices

Device	M512 Columns/Blocks	M4K Columns/Blocks	M-RAM Blocks	Total RAM Bits
EP1S10	4 / 94	2 / 60	1	920,448
EP1S20	6 / 194	2 / 82	2	1,669,248
EP1S25	6 / 224	3 / 138	2	1,944,576
EP1S30	7 / 295	3 / 171	4	3,317,184
EP1S40	8 / 384	3 / 183	4	3,423,744
EP1S60	10 / 574	4 / 292	6	5,215,104
EP1S80	11 / 767	4 / 364	9	7,427,520

Table 14–3. TriMatrix Memory Distribution in Stratix GX Devices

Device	M512 Columns/Blocks	M4K Columns/Blocks	M-RAM Blocks	Total RAM Bits
EP1SGX10	4 / 94	2 / 60	1	920,448
EP1SGX25	6 / 224	3 / 138	2	1,944,576
EP1SGX40	8 / 384	3 / 183	4	3,423,744

Clear Signals

When applied to input registers, the asynchronous clear signal for the TriMatrix embedded memory immediately clears the input registers. However, the output of the memory block does not show the effects until the next clock edge. When applied to output registers, the asynchronous clear signal clears the output registers and the effects are seen immediately.

Parity Bit Support

The memory blocks support a parity bit for each byte. Parity bits are in addition to the amount of memory in each RAM block. For example, the M512 block has 576 bits, 64 of which are optionally used for parity bit

storage. The parity bit, along with logic implemented in logic elements (LEs), can implement parity checking for error detection to ensure data integrity. Parity-size data words can also store user-specified control bits.

Byte Enable Support

In the M4K and M-RAM blocks, byte enables can mask the input data so that only specific bytes of data are written. The unwritten bytes retain the previous written value. The write enable signals (`wren`), in conjunction with the byte enable signals (`byteena`), controls the RAM block's write operations. The default value for the `byteena` signals is high (enabled), in which case writing is controlled only by the `wren` signals.

Asserting the clear port of the byte enable registers drives the byte enable signals to their default high level.

M4K Blocks

M4K blocks support byte writes when the write port has a data width of 16, 18, 32, or 36 bits. Table 14–4 summarizes the byte selection.

Table 14–4. Byte Enable for M4K Blocks *Notes (1), (2)*

byteena	datain ×18	datain ×36
[0] = 1	[8..0]	[8..0]
[1] = 1	[17..9]	[17..9]
[2] = 1	–	[26..18]
[3] = 1	–	[35..27]

Notes to Table 14–4:

(1) Any combination of byte enables is possible.

(2) Byte enables can be used in the same manner with 8-bit words, i.e., in ×16 and ×32 modes.

M-RAM Blocks

M-RAM blocks support byte enables for the ×16, ×18, ×32, ×36, ×64, and ×72 modes. In the ×128 or ×144 simple dual-port mode, the two sets of `byteena` signals (`byteena_a` and `byteena_b`) combine to form the necessary 16 byte enables. Tables 14–5 and 14–6 summarize the byte selection.

Table 14–5. Byte Enable for M-RAM Blocks *Notes (1), (2)*

byteena	datain ×18	datain ×36	datain ×72
[0] = 1	[8..0]	[8..0]	[8..0]
[1] = 1	[17..9]	[17..9]	[17..9]
[2] = 1	–	[26..18]	[26..18]
[3] = 1	–	[35..27]	[35..27]
[4] = 1	–	–	[44..36]
[5] = 1	–	–	[53..45]
[6] = 1	–	–	[62..54]
[7] = 1	–	–	[71..63]

Notes to Table 14–5:
(1) Any combination of byte enables is possible.
(2) Byte enables can be used in the same manner with 8-bit words, that is, in ×16, ×32, and ×64 modes.

Table 14–6. M-RAM Combined Byte Selection for × 144 Mode (Part 1 of 2), *Notes (1), (2)*

byteena_a	datain ×144
[0] = 1	[8..0]
[1] = 1	[17..9]
[2] = 1	[26..18]
[3] = 1	[35..27]
[4] = 1	[44..36]
[5] = 1	[53..45]
[6] = 1	[62..54]
[7] = 1	[71..63]
[8] = 1	[80..72]
[9] = 1	[89..81]
[10] = 1	[98..90]
[11] = 1	[107..99]

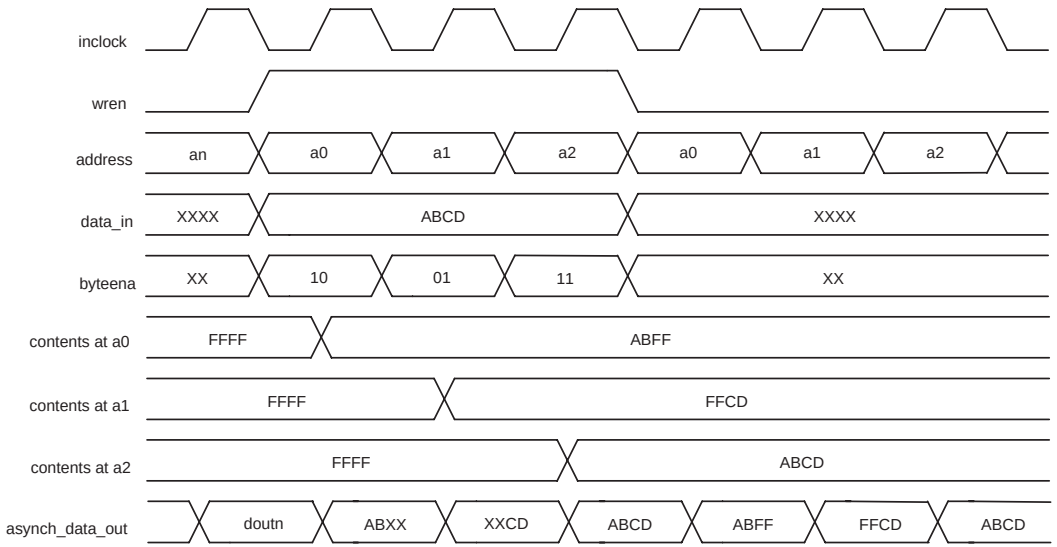
Table 14–6. M-RAM Combined Byte Selection for ×144 Mode (Part 2 of 2), Notes (1), (2)	
byteena_a	**datain ×144**
[12] = 1	[116..108]
[13] = 1	[125..117]
[14] = 1	[134..126]
[15] = 1	[143..135]

Notes to Table 14–6:
(1) Any combination of byte enables is possible.
(2) Byte enables can be used in the same manner with 8-bit words, i.e., in ×16, ×32, ×64, and ×128 modes.

Byte Enable Functional Waveform

Figure 14–1 shows how both the `wren` and the `byteena` signals control the write operations of the RAM.

Figure 14–1. Byte Enable Functional Waveform Note (1)



Note to Figure 14–1:
(1) For more information on simulation output when a read-during-write occurs at the same address location, see "Read-During-Write Operation at the Same Address" on page 14–25.

Using TriMatrix Memory

The TriMatrix memory blocks include input registers that synchronize writes and output registers to pipeline designs and improve system performance. All TriMatrix memory blocks are pipelined, meaning that all inputs are registered, but outputs are either registered or combinatorial. TriMatrix memory can emulate a flow-through memory by using combinatorial outputs.

For more information, see *AN 210: Converting Memory from Asynchronous to Synchronous for Stratix & Stratix GX Designs*.

Depending on the TriMatrix memory block type, the memory can have various modes, including:

- Single-port
- Simple dual-port
- True dual-port (bidirectional dual-port)
- Shift-register
- ROM
- FIFO

Implementing Single-Port Mode

Single-port mode supports non-simultaneous reads and writes. Figure 14–2 shows the single-port memory configuration for TriMatrix memory. All memory block types support the single-port mode.

Figure 14–2. Single-Port Memory *Note (1)*



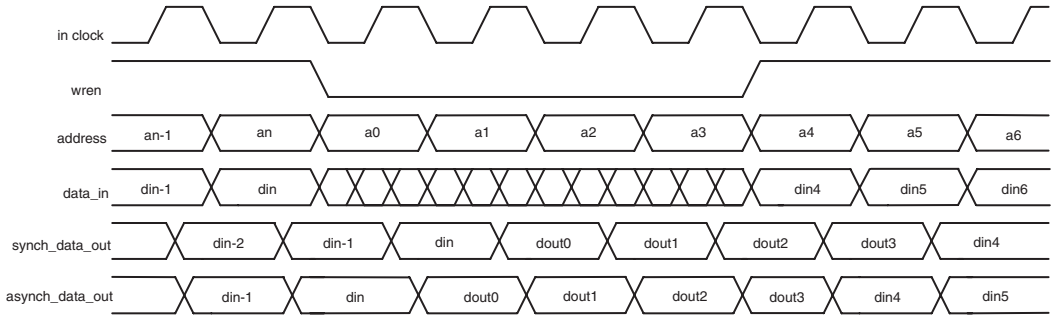
Note to Figure 14–2:
(1) Two single-port memory blocks can be implemented in a single M4K block.

M4K memory blocks can also be divided in half and used for two independent single-port RAM blocks. The Altera Quartus II software automatically uses this single-port memory packing when running low on memory resources. To force two single-port memories into one M4K block, first ensure that each of the two independent RAM blocks is equal to or less than half the size of the M4K block. Second, assign both single-port RAMs to the same M4K block.

In the single-port RAM configuration, the outputs can only be in read-during-write mode, which means that during the write operation, data written to the RAM flows through to the RAM outputs. When the output registers are bypassed, the new data is available on the rising edge of the same clock cycle it was written on. For more information about read-during-write mode, see "Read-During-Write Operation at the Same Address" on page 14–25.

Figure 14–3 shows timing waveforms for read and write operations in single-port mode.

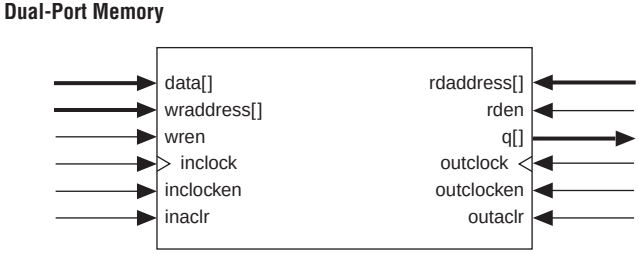
Figure 14–3. Single-Port Timing Waveforms



Implementing Simple Dual-Port Mode

Simple dual-port memory supports a simultaneous read and write. Figure 14–4 shows the simple dual-port memory configuration for TriMatrix memory. All memory block types support this configuration.

Figure 14–4. Simple Dual-Port Memory Note (1)



Note to Figure 14–4:
(1) Simple dual-port RAM supports read/write clock mode in addition to the input/output clock mode shown.

TriMatrix memory supports mixed-width configurations, allowing different read and write port widths. When using mixed-width mode, the LSB is written to or read from first. For example, take a RAM that is set up in mixed-width mode with write data width ×8 and read data width ×2. If a binary 00000001 is written to write dress 0, the following is read out of the ×2 output side:

Read Address	×2 data
00	01(LSB of ×8 data)
01	00
10	00
11	00(MSB of ×8 data)

Tables 14–7 to 14–9 show the mixed width configurations for the M512, M4K, and M-RAM blocks, respectively.

Table 14–7. M512 Block Mixed-Width Configurations (Simple Dual-Port Mode)

Read Port	Write Port						
	512 × 1	256 × 2	128 × 4	64 × 8	32 × 16	64 × 9	32 × 18
512 × 1	✓	✓	✓	✓	✓		
256 × 2	✓	✓	✓	✓	✓		
128 × 4	✓	✓	✓		✓		
64 × 8	✓	✓		✓			
32 × 16	✓	✓	✓		✓		
64 × 9						✓	
32 × 18							✓

Table 14–8. M4K Block Mixed-Width Configurations (Simple Dual-Port Mode) (Part 1 of 2)

Read Port	Write Port								
	4K × 1	2K × 2	1K × 4	512 × 8	256 × 16	128 × 32	512 × 9	256 × 18	128 × 36
4K × 1	✓	✓	✓	✓	✓	✓			
2K × 2	✓	✓	✓	✓	✓	✓			
1K × 4	✓	✓	✓	✓	✓	✓			
512 × 8	✓	✓	✓	✓	✓	✓			
256 × 16	✓	✓	✓	✓	✓	✓			

Table 14–8. M4K Block Mixed-Width Configurations (Simple Dual-Port Mode) (Part 2 of 2)

Read Port	Write Port								
	4K × 1	2K × 2	1K × 4	512 × 8	256 × 16	128 × 32	512 × 9	256 × 18	128 × 36
128 × 32	✓	✓	✓	✓	✓	✓			
512 × 9							✓	✓	✓
256 × 18							✓	✓	✓
128 × 36							✓	✓	✓

Table 14–9. M-RAM Block Mixed-Width Configurations (Simple Dual-Port Mode)

Read Port	Write Port				
	64K × 9	32K × 18	16K × 36	8K × 72	4K × 144
64K × 9	✓	✓	✓	✓	
32K × 18	✓	✓	✓	✓	
16K × 36	✓	✓	✓	✓	
8K × 72	✓	✓	✓	✓	
4K × 144					✓

M512 blocks support serializer and deserializer (SERDES) applications. By using the mixed-width support in combination with double data rate (DDR) I/O standards, the block can function as a SERDES to support low-speed serial I/O standards using global or regional clocks.



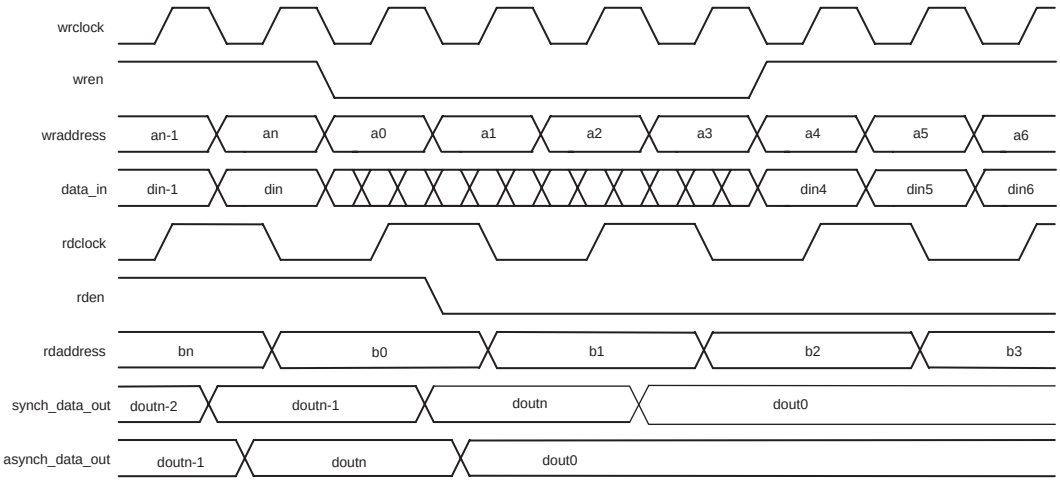
For more information on Stratix device I/O structure see the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1*. For more information on Stratix GX device I/O structure see the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1*.

In simple dual-port mode, the M512 and M4K blocks have one write enable and one read enable signal. The M512 does not support a clear port on the `rden` register. On the M4K block, asserting the clear port of the `rden` register drives `rden` high, which allows the read operation to occur. When the read enable is deactivated, the current data is retained at the output ports. If the read enable is activated during a write operation with the same address location selected, the simple dual-port RAM output is either unknown or can be set to output the old data stored at the memory address. For more information, see "Read-During-Write Operation at the Same Address" on page 14–25.

M-RAM blocks have one write enable signal in simple dual-port mode. To perform a write operation, the write enable is held high. The M-RAM block is always enabled for read operation. If the read address and the write address select the same address location during a write operation, the M-RAM block output is unknown.

Figure 14–5 shows timing waveforms for read and write operations in simple dual-port mode.

Figure 14–5. Simple Dual-Port Timing Waveforms *Note (1)*



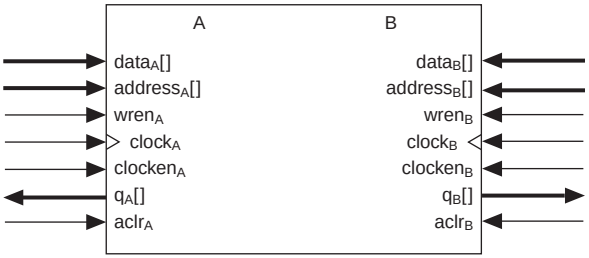
Note to Figure 14–5:

(1) The `rden` signal is not available in the M-RAM block. A M-RAM block in simple dual-port mode is always reading out the data stored at the current read address location.

Implementing True Dual-Port Mode

M4K and M-RAM blocks offer a true dual-port mode to support any combination of two-port operations: two reads, two writes, or one read and one write at two different clock frequencies. Figure 14–6 shows the true dual-port memory configuration for TriMatrix memory.

Figure 14–6. True Dual-Port Memory Note (1)



Note to Figure 14–6:

(1) True dual-port memory supports input/output clock mode in addition to the independent clock mode shown.

The widest bit configuration of the M4K and M-RAM blocks in true dual-port mode is 256 × 16-bit (× 18-bit with parity) and 8K × 64-bit (× 72-bit with parity), respectively. The 128 × 32-bit (× 36-bit with parity) configuration of the M4K block and the 4K × 128-bit (× 144-bit with parity) configuration of the M-RAM block are unavailable because the number of output drivers is equivalent to the maximum bit width of the respective memory block. Because true dual-port RAM has outputs on two ports, the maximum width of the true dual-port RAM equals half of the total number of output drivers. Tables 14–10 and 14–11 list the possible M4K RAM block and M-RAM block configurations, respectively.

Table 14–10. M4K Block Mixed-Port Width Configurations (True Dual-Port)

Port A	Port B						
	4K × 1	2K × 2	1K × 4	512 × 8	256 × 16	512 × 9	256 × 18
4K × 1	✓	✓	✓	✓	✓		
2K × 2	✓	✓	✓	✓	✓		
1K × 4	✓	✓	✓	✓	✓		
512 × 8	✓	✓	✓	✓	✓		
256 × 16	✓	✓	✓	✓	✓		
512 × 9						✓	✓
256 × 18						✓	✓

Table 14–11. M-RAM Block Mixed-Port Width Configurations (True Dual-Port)

Port A	Port B			
	64K × 9	32K × 18	16K × 36	8K × 72
64K × 9	✓	✓	✓	✓
32K × 18	✓	✓	✓	✓
16K × 36	✓	✓	✓	✓
8K × 72	✓	✓	✓	✓

In true dual-port configuration, the RAM outputs can only be configured for read-during-write mode. This means that during write operation, data being written to the A or B port of the RAM flows through to the A or B outputs, respectively. When the output registers are bypassed, the new data is available on the rising edge of the same clock cycle it was written on. For waveforms and information on mixed-port read-during-write mode, see "Read-During-Write Operation at the Same Address" on page 14–25.

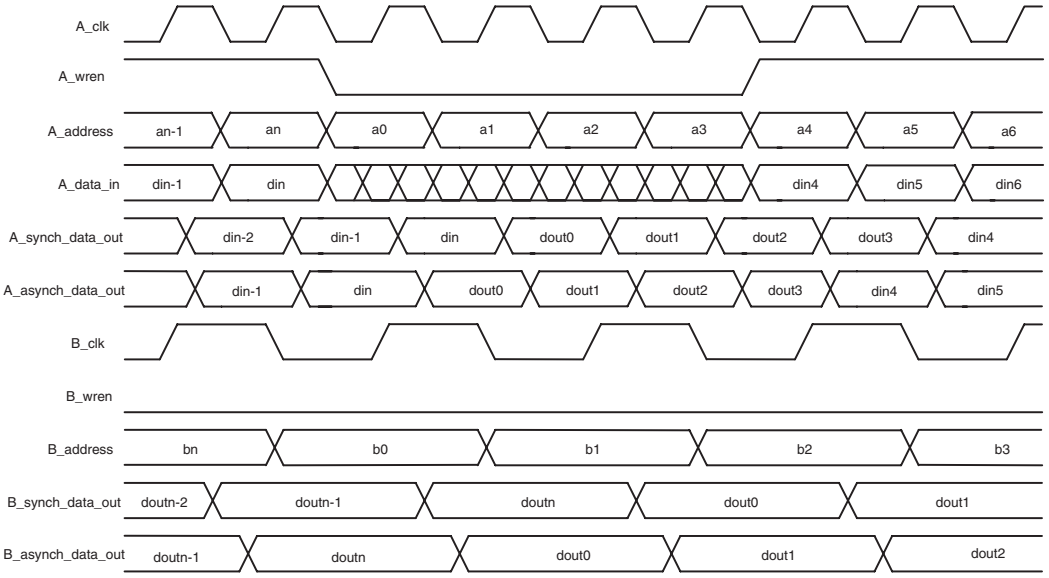
Potential write contentions must be resolved external to the RAM because writing to the same address location at both ports results in unknown data storage at that location. Data is written on the rising edge of the write clock for the M-RAM block. For a valid write operation to the same address of the M-RAM block, the rising edge of the write clock for port A must occur following the maximum write cycle time interval after the rising edge of the write clock for port B. Since data is written into the M512 and M4K blocks at the falling edge of the write clock, the rising edge of the write clock for port A should occur following half of the maximum write cycle time interval after the falling edge of the write clock for port B. If this timing is not met, the data stored in that particular address is invalid.



See the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* or the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1* for the maximum synchronous write cycle time.

Figure 14–7 shows true dual-port timing waveforms for write operation at port A and read operation at port B.

Figure 14–7. True Dual-Port Timing Waveforms



Implementing Shift-Register Mode

Embedded memory block configurations can implement shift registers for digital signal processing (DSP) applications, such as finite impulse response (FIR) filters, pseudo-random number generators, multi-channel filtering, and auto-correlation and cross-correlation functions. These and other DSP applications require local data storage, traditionally implemented with standard flip-flops that can quickly consume many logic cells for large shift registers. A more efficient alternative is to use embedded memory as a shift register block, which saves logic cell and routing resources and provides a more efficient implementation.

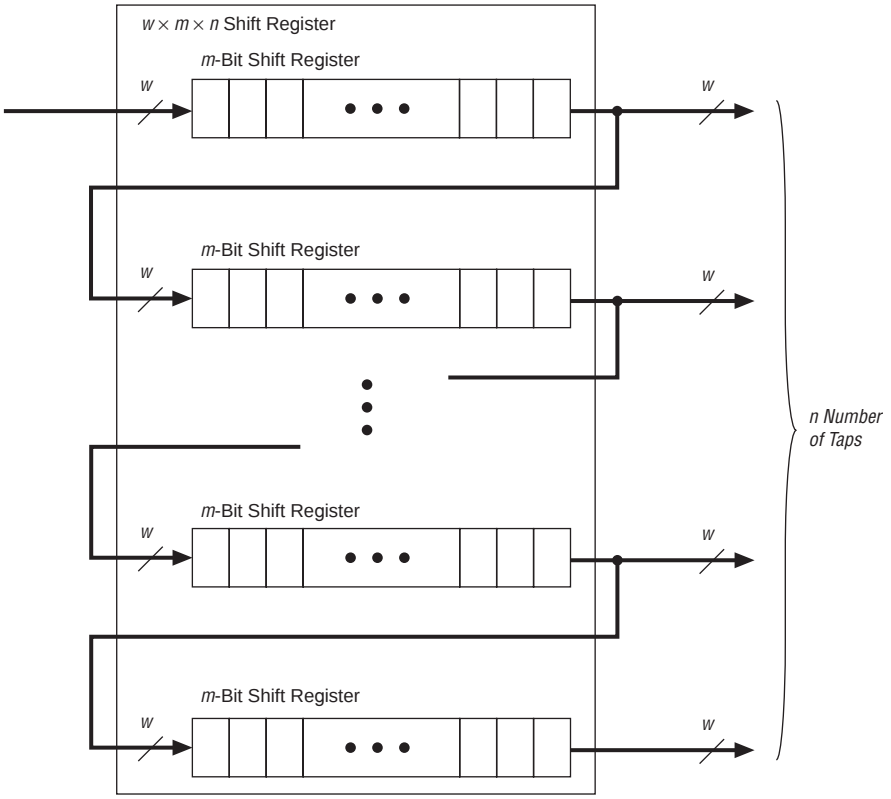
The size of a ($w \times m \times n$) shift register is determined by the input data width (w), the length of the taps (m), and the number of taps (n). The size of a ($w \times m \times n$) shift register must be less than or equal to the maximum number of memory bits in the respective block: 576 bits for the M512 block and 4,608 bits for the M4K block. In addition, the size of $w \times n$ must be less than or equal to the maximum width of the respective block: 18 bits for the M512 block and 36 bits for the M4K block. If a larger shift register is required, the memory blocks can be cascaded together.



M-RAM blocks do not support the shift-register mode.

Data is written into each address location at the falling edge of the clock and read from the address at the rising edge of the clock. The shift-register mode logic automatically controls the positive and negative edge clocking to shift the data in one clock cycle. Figure 14–8 shows the TriMatrix memory block in the shift-register mode.

Figure 14–8. Shift-Register Memory Configuration



Implementing ROM Mode

The M512 and the M4K blocks support ROM mode. Use a memory initialization file (**.mif**) to initialize the ROM contents of M512 and M4K blocks. The M-RAM block does not support ROM mode.

All Stratix memory configurations must have synchronous inputs; therefore, the address lines of the ROM are registered. The outputs can be registered or combinatorial. The ROM read operation is identical to the read operation in the single-port RAM configuration.

Implementing FIFO Buffers

While the small M512 memory blocks are ideal for designs with many shallow FIFO buffers, all three memory sizes support FIFO mode.

All memory configurations have synchronous inputs; however, the FIFO buffer outputs are always combinatorial. Simultaneous read and write from an empty FIFO is not supported.

Clock Modes

Depending on the TriMatrix memory mode, independent, input/output, read/write, and/or single-port clock modes are available. Table 14–12 shows the clock modes supported by the TriMatrix memory modes.

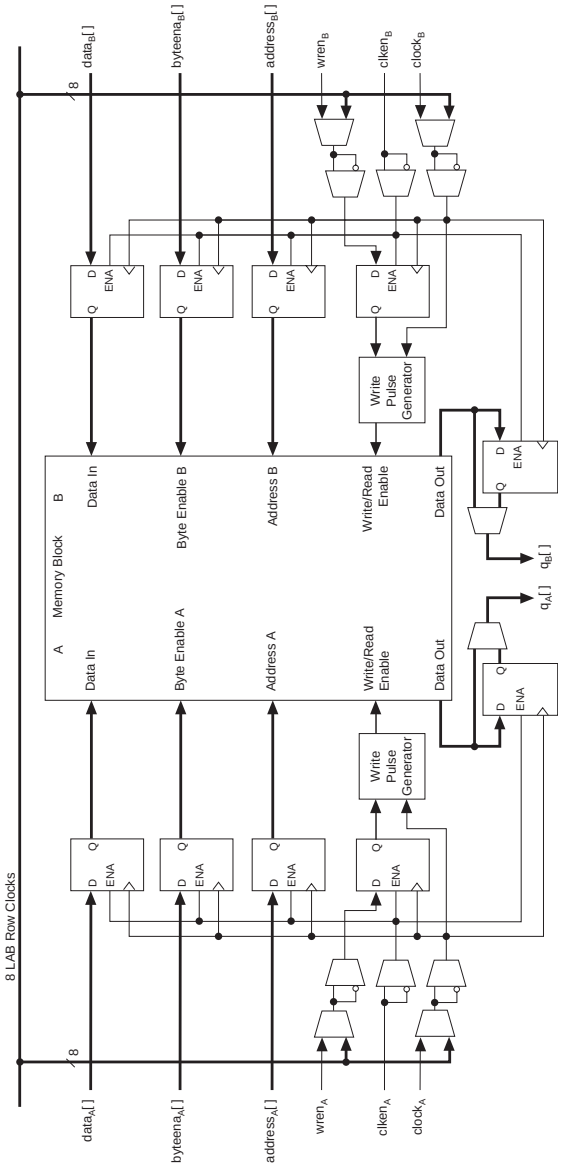
Table 14–12. TriMatrix Memory Clock Modes

Clocking Mode	True-Dual Port Mode	Simple Dual-Port Mode	Single-Port Mode
Independent	✓		
Input/output	✓	✓	
Read/write		✓	
Single-port			✓

Independent Clock Mode

The TriMatrix memory blocks can implement independent clock mode for true dual-port memory. In this mode, a separate clock is available for each port (A and B). Clock A controls all registers on the port A side, while clock B controls all registers on the port B side. Each port also supports independent clock enables and asynchronous clear signals for port A and B registers. Figure 14–9 shows a TriMatrix memory block in independent clock mode.

Figure 14–9. Independent Clock Mode *Note (1), (2)*



Note to Figure 14–9:

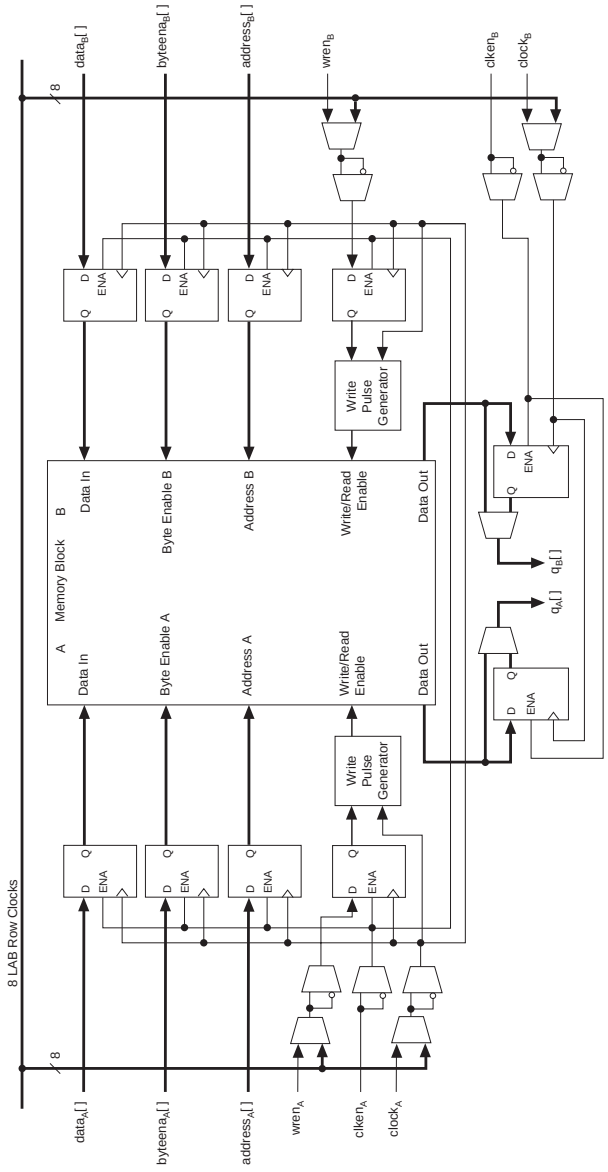
(1) Violating the setup or hold time on the address registers could corrupt the memory contents. This applies to both read and write operations.

(2) All registers shown have asynchronous clear ports, except when using the M-RAM. M-RAM blocks have asynchronous clear ports on their output registers only.

Input/Output Clock Mode

The TriMatrix memory blocks can implement input/output clock mode for true and simple dual-port memory. On each of the two ports, A and B, one clock controls all registers for inputs into the memory block: data input, `wren`, and address. The other clock controls the block's data output registers. Each memory block port also supports independent clock enables and asynchronous clear signals for input and output registers. Figures 14–10 and 14–11 show the memory block in input/output clock mode for true and simple dual-port modes, respectively.

Figure 14–10. Input/Output Clock Mode in True Dual-Port Mode Note (1)

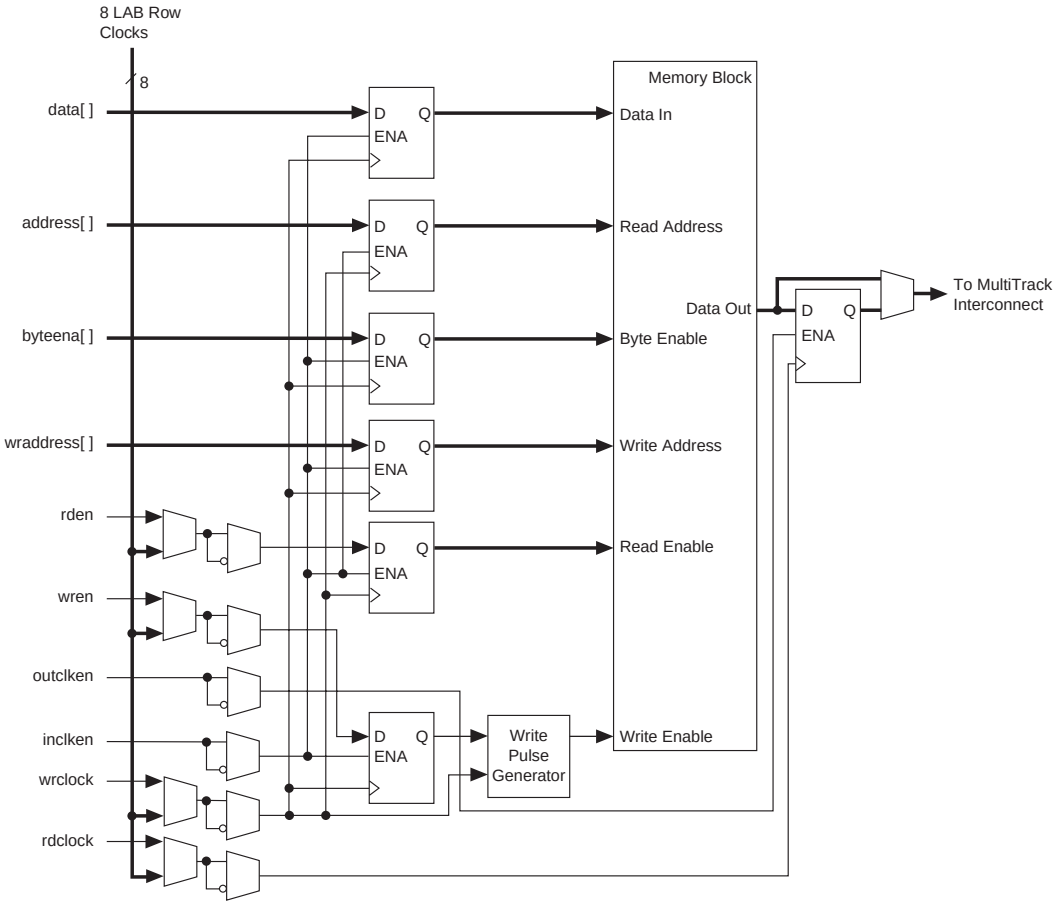


Note to Figure 14–10:

(1) Violating the setup or hold time on the address registers could corrupt the memory contents. This applies to both read and write operations.

All registers shown have asynchronous clear ports, except when using the M-RAM. M-RAM blocks have asynchronous clear ports on their output registers only.

Figure 14–11. Input/Output Clock Mode in Simple Dual-Port Mode *Notes (1), (2), (3), (4)*



Notes to Figure 14–11:

(1) The `rden` signal is not available in the M-RAM block. A M-RAM block in simple dual-port mode is always reading out the data stored at the current read address location.

(2) For more information on the MultiTrack™ interconnect, see the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* or the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1*.

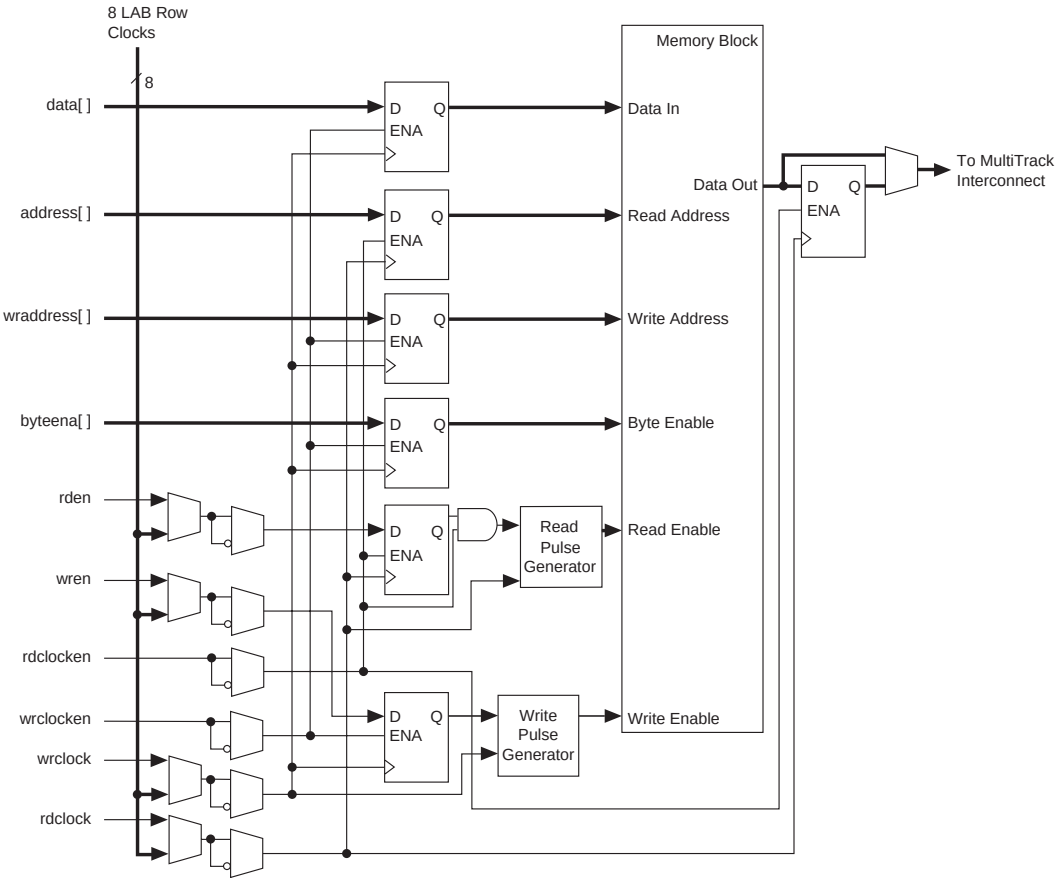
(3) All registers shown have asynchronous clear ports, except when using the M-RAM. M-RAM blocks have asynchronous clear ports on their output registers only.

(4) Violating the setup or hold time on the address registers could corrupt the memory contents. This applies to both read and write operations.

Read/Write Clock Mode

The TriMatrix memory blocks can implement read/write clock mode for simple dual-port memory. This mode can use up to two clocks. The write clock controls the block's data inputs, `wraddress`, and `wren`. The read clock controls the data output, `rdaddress`, and `rden`. The memory blocks support independent clock enables for each clock and asynchronous clear signals for the read- and write-side registers. Figure 14–12 shows a memory block in read/write clock mode.

Figure 14–12. Read/Write Clock Mode in Simple Dual-Port Mode Notes (1), (2), (3)



Notes to Figure 14–12:

(1) For more information on the MultiTrack interconnect, see the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* or the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1*.

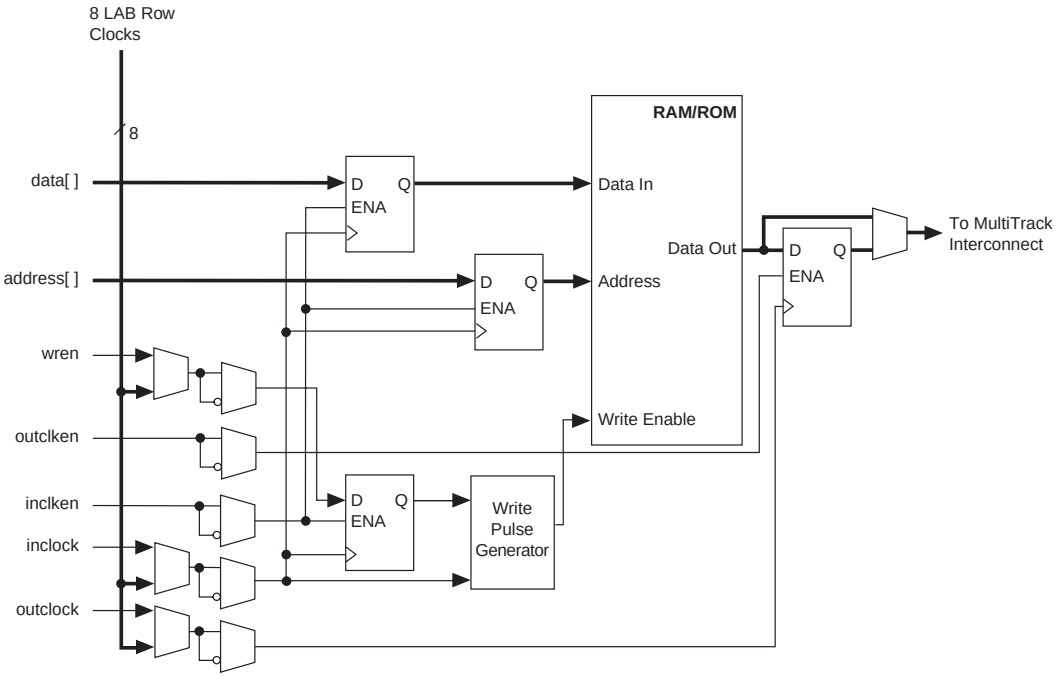
(2) All registers shown have asynchronous clear ports, except when using the M-RAM. M-RAM blocks have asynchronous clear ports on their output registers only.

(3) Violating the setup or hold time on the address registers could corrupt the memory contents. This applies to both read and write operations.

Single-Port Mode

The TriMatrix memory blocks can implement single-port clock mode for single-port memory mode. Single-port mode is used when simultaneous reads and writes are not required. See Figure 14–13. A single block in a memory block can support up to two single-port mode RAM blocks in M4K blocks.

Figure 14–13. Single-Port Mode *Notes (1), (2), (3)*



Notes to Figure 14–13:

(1) For more information on the MultiTrack interconnect, see the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* or the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1*.

(2) All registers shown have asynchronous clear ports, except when using the M-RAM. M-RAM blocks have asynchronous clear ports on their output registers only.

(3) Violating the setup or hold time on the address registers could corrupt the memory contents. This applies to both read and write operations.

Designing With TriMatrix Memory

When instantiating TriMatrix memory you must understand the various features that set it apart from other memory architectures. The following sections describe some of the important attributes and functionality of TriMatrix memory.

For information on the difference between APEX-style memory and TriMatrix memory, see the *Transitioning APEX Designs to Stratix Devices* chapter.

Selecting TriMatrix Memory Blocks

The Quartus II software automatically partitions user-defined memory into embedded memory blocks using the most efficient size combinations. The memory can also be manually assigned to a specific block size or a mixture of block sizes. Table 14–1 on page 14–2 is a guide for selecting a TriMatrix memory block size based on supported features.

Violating the setup or hold time on the address registers could corrupt the memory contents. This applies to both read and write operations.

For more information on selecting which memory block to use, see *AN 207: TriMatrix Memory Selection Using the Quartus II Software*.

Violating the setup or hold time on the address registers could corrupt the memory contents. This applies to both read and write operations.

Pipeline & Flow-Through Modes

TriMatrix memory architecture implements synchronous (pipelined) RAM by registering both the input and output signals to the RAM block. All TriMatrix memory inputs are registered providing synchronous write cycles. In synchronous operation, RAM generates its own self-timed strobe write enable (`wren`) signal derived from the global or regional clock. In contrast, a circuit using asynchronous RAM must generate the RAM `wren` signal while ensuring its data and address signals meet setup and hold time specifications relative to the `wren` signal. The output registers can be bypassed.

In an asynchronous memory neither the input nor the output is registered. While Stratix and Stratix GX devices do not support asynchronous memory, they do support a flow-through read where the output data is available during the clock cycle when the read address is driven into it. Flow-through reading is possible in the simple and true dual-port modes of the M512 and M4K blocks by clocking the read enable and read address registers on the negative clock edge and bypassing the output registers.

For more information, see *AN 210: Converting Memory from Asynchronous to Synchronous for Stratix & Stratix GX Devices*.

Power-up Conditions & Memory Initialization

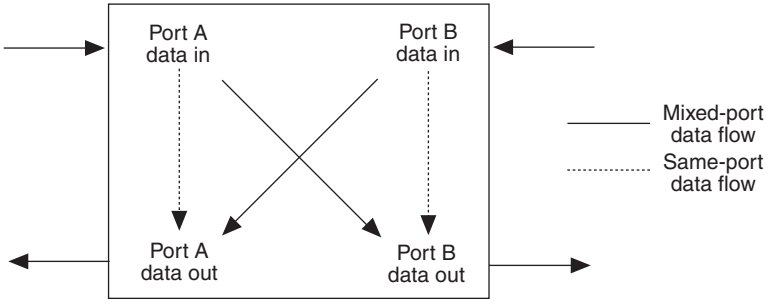
Upon power-up, TriMatrix memory is in an idle state. The M512 and M4K block outputs always power-up to zero, regardless of whether the output registers are used or bypassed. Even if a memory initialization file is used to pre-load the contents of the RAM block, the outputs still power-up cleared. For example, if address `0` is pre-initialized to `FF`, the M512 and M4K blocks power-up with the output at `00`.

M-RAM blocks do not support memory initialization files; therefore, they cannot be pre-loaded with data upon power-up. M-RAM blocks combinatorial outputs and memory controls always power-up to an unknown state. If M-RAM block outputs are registered, the registers power-up cleared. The undefined output appears one clock cycle later. The output remains undefined until a read operation is performed on an address that has been written to.

Read-During-Write Operation at the Same Address

The following two sections describe the functionality of the various RAM configurations when reading from an address during a write operation at that same address. There are two types of read-during-write operations: same-port and mixed-port. Figure 14–14 illustrates the difference in data flow between same-port and mixed-port read-during-write.

Figure 14–14. Read-During-Write Data Flow

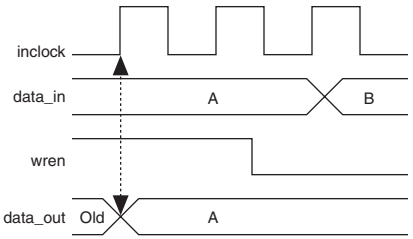


Same-Port Read-During-Write Mode

For read-during-write operation of a single-port RAM or the same port of a true dual-port RAM, the new data is available on the rising edge of the same clock cycle it was written on. This behavior is valid on all memory-block sizes. See Figure 14–15 for a sample functional waveform.

When using byte enables in true dual-port RAM mode, the outputs for the masked bytes on the same port are unknown. (See Figure 14–1 on page 14–6.) The non-masked bytes are read out as shown in Figure 14–15.

Figure 14–15. Same-Port Read-During-Write Functionality Note (1)



Note to Figure 14–15:
(1) Outputs are not registered.

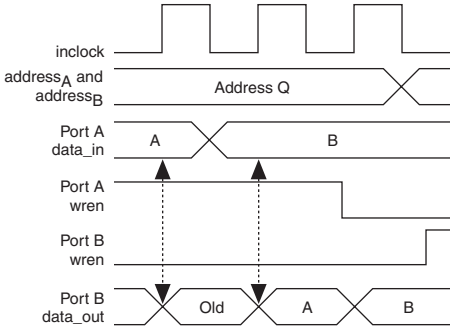
Mixed-Port Read-During-Write Mode

This mode is used when a RAM in simple or true dual-port mode has one port reading and the other port writing to the same address location with the same clock.

The READ_DURING_WRITE_MODE_MIXED_PORTS parameter for M512 and M4K memory blocks determines whether to output the old data at the address or a "don't care" value. Setting this parameter to OLD_DATA outputs the old data at that address. Setting this parameter to DONT_CARE outputs a "don't care" or unknown value. See Figures 14–16 and 14–17 for sample functional waveforms showing this operation. These figures assume that the outputs are not registered.

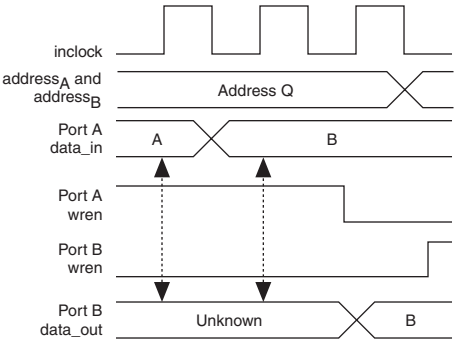
The DONT_CARE setting allows memory implementation in any TriMatrix memory block. The OLD_DATA setting restricts memory implementation to only M512 or M4K memory blocks. Selecting DONT_CARE gives the compiler more flexibility when placing memory functions into TriMatrix memory.

Figure 14–16. Mixed-Port Read-During-Write: OLD_DATA



For mixed-port read-during-write operation of the same address location of a M-RAM block, the RAM outputs are unknown, as shown in Figure 14–17.

Figure 14–17. Mixed-Port Read-During-Write: DONT_CARE



Mixed-port read-during-write is not supported when two different clocks are used in a dual-port RAM. The output value will be unknown during a mixed-port read-during-write operation.

Conclusion

TriMatrix memory, an enhanced RAM architecture with extremely high memory bandwidth in Stratix and Stratix GX devices, gives advanced control of memory applications with features such as byte enables, parity bit storage, and shift-register mode, as well as mixed-port width support and true dual-port mode.

15. External Memory Interfaces in Stratix & Stratix GX Devices

S52008-3.3

Introduction

Stratix® and Stratix GX devices support a broad range of external memory interfaces such as double data rate (DDR) SDRAM, RLDRAM II, quad data rate (QDR) SRAM, QDRII SRAM, zero bus turnaround (ZBT) SRAM, and single data rate (SDR) SDRAM. The dedicated phase-shift circuitry allows the Stratix and Stratix GX devices to interface at twice the system clock speed with an external memory (up to 200 MHz/400 Mbps).

Typical I/O architectures transmit a single data word on each positive clock edge and are limited to the associated clock speed using this protocol. To achieve a 400-megabits per second (Mbps) transfer rate, a SDR system requires a 400-MHz clock. Many new applications have introduced a DDR I/O architecture as an alternative to SDR architectures. While SDR architectures capture data on one edge of a clock, the DDR architectures captures data on both the rising and falling edges of the clock, doubling the throughput for a given clock frequency and accelerating performance. For example, a 200-MHz clock can capture a 400-Mbps data stream, enhancing system performance and simplifying board design.

Most current memory architectures use a DDR I/O interface. These DDR memory standards cover a broad range of applications for embedded processor systems, image processing, storage, communications, and networking. This chapter describes the hardware features in Stratix and Stratix GX devices that facilitate the high-speed memory interfacing for each memory standard. It then briefly explains how each memory standard uses the features of the Stratix and Stratix GX devices.



You can use this document with *AN 329: ZBT SRAM Controller Reference Design for Stratix & Stratix GX Devices*, *AN 342: Interfacing DDR SDRAM with Stratix & Stratix GX Devices*, and *AN 349: QDR SRAM Controller Reference Design for Stratix & Stratix GX Devices*.

External Memory Standards

The following sections provide an overview on using the Stratix and Stratix GX device external memory interfacing features.

DDR SDRAM

DDR SDRAM is a memory architecture that transmits and receives data at twice the clock speed of traditional SDR architectures. These devices transfer data on both the rising and falling edge of the clock signal.

Interface Pins

DDR devices use interface pins including data, data strobe, clock, command, and address pins. Data is sent and captured at twice the clock rate by transferring data on both the positive and negative edge of a clock. The commands and addresses only use one active edge of a clock.

Connect the memory device's DQ and DQS pins to the DQ and DQS pins, respectively, as listed in the Stratix and Stratix GX devices pin table. DDR SDRAM also uses active-high data mask pins for writes. You can connect DM pins to any of the I/O pins in the same bank as the DQ pins of the FPGA. There is one DM pin per DQS/DQ group.

DDR SDRAM ×16 devices use two DQS pins, and each DQS pin is associated with eight DQ pins. However, this is not the same as the ×16 mode in Stratix and Stratix GX devices. To support a ×16 DDR SDRAM, you need to configure the Stratix and Stratix GX FPGAs to use two sets of DQ pins in ×8 mode. Similarly if your ×32 memory device uses four DQS pins where each DQS pin is associated with eight DQ pins, you need to configure the Stratix and Stratix GX FPGA to use four sets of pins in ×8 mode.

You can also use any I/O pins in banks 1, 2, 5, or 6 to interface with DDR SDRAM devices. These banks do not have dedicated circuitry, though.

You can also use any of the user I/O pins for commands and addresses to the DDR SDRAM.



For more information, see *AN 342: Interfacing DDR SDRAM with Stratix & Stratix GX Devices*.

If the DDR SDRAM device supports ECC, the design uses a DQS/DQ group for ECC pins. You can use any of the user I/O pins for commands and addresses.

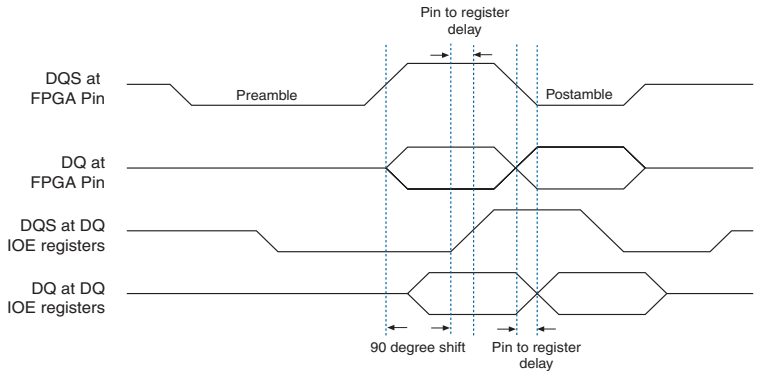
Because of the symmetrical setup and hold time for the command and address pins at the memory, you might need to generate these signals from the system clock's negative edge.

The clocks to the SDRAM device are called CK and CK#. Use any of the user I/O pins via the DDR registers to generate the CK and CK# signals to meet the DDR SDRAM t_{DQSS} requirement. The memory device's t_{DQSS} requires that the DQS signal's positive edge write operations must be within 25% of the positive edge of the DDR SDRAM clock input. Using user I/O pins for CK and CK# ensures that any PVT variations seen by the DQS signal are tracked by these pins, too.

Read & Write Operations

When reading from the DDR SDRAM, the DQS signal coming into the Stratix and Stratix GX device is edge-aligned with the DQ pins. The dedicated circuitry center-aligns the DQS signal with respect to the DQ signals and the shifted DQS bus drives the clock input of the DDR input registers. The DDR input registers bring the data from the DQ signals to the device. The system clock clocks the DQS output enable and output paths. The -90° shifted clock clocks the DQ output enable and output paths. Figure 15–1 shows an example of the DQ and DQS relationship during a burst-of-two read. It shows where the DQS signal is center-aligned in the IOE.

Figure 15–1. Example of Where a DQS Signal is Center-Aligned in the IOE



When writing to the DDR SDRAM, the DQS signal must be center-aligned with the DQ pins. Two PLL outputs are needed to generate the DQS signal and to clock the DQ pins. The DQS are clocked by the 0° phase-shift PLL output, while the DQ pins are clocked by the -90° phase-shifted PLL output. Figure 15–2 shows the DQS and DQ relationship during a DDR SDRAM burst-of-two write.

Figure 15–2. DQ & DQS Relationship During a Burst-of-Two Write

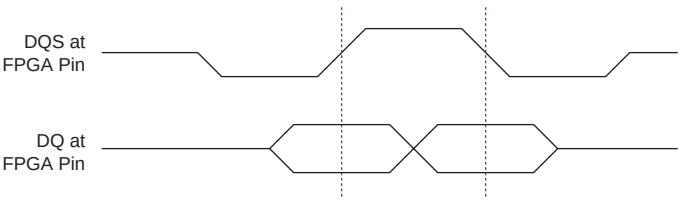
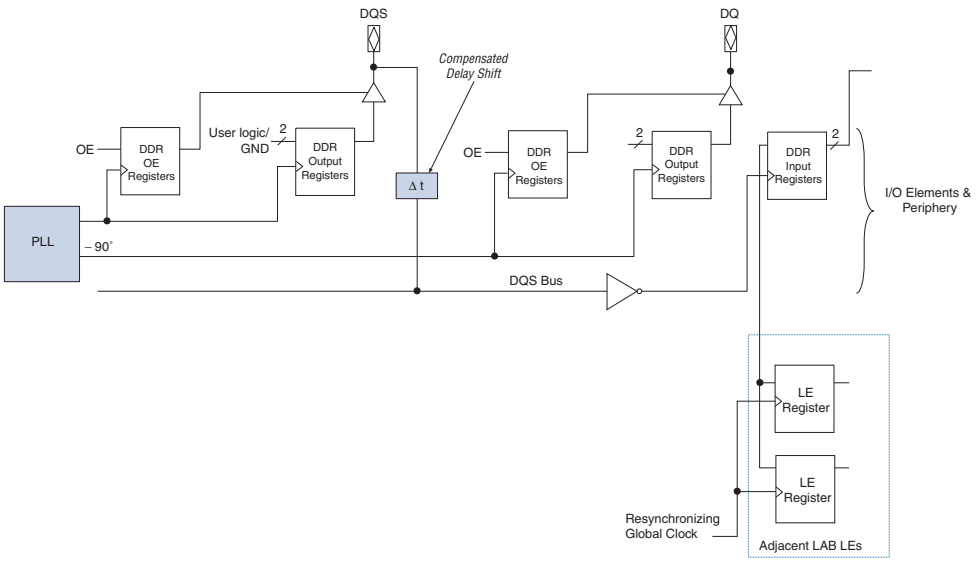


Figure 15–3 shows DDR SDRAM interfacing from the I/O through the dedicated circuitry to the logic array. When the DQS pin acts as an input strobe, the dedicated circuitry shifts the incoming DQS pin by either 72° or 90° and clocks the DDR input registers. Because of the DDR input registers architecture in Stratix and Stratix GX devices, the shifted DQS signal must be inverted. The DDR registers outputs are sent to two LE registers to be synchronized with the system clock.

Refer to the *DC & Switching Characteristics* chapter in volume 1 of the *Stratix Device Handbook* for frequency limits regarding the 72 and 90° phase shift for DQS.

Figure 15–3. DDR SDRAM Interfacing



For more information on DDR SDRAM specifications, see JEDEC standard publications JESD79C from **www.jedec.org**, or see *AN 342: Interfacing DDR SDRAM with Stratix & Stratix GX Devices*.

RLDRAM II

RLDRAM II provides fast random access as well as high bandwidth and high density, making this memory technology ideal for high-speed network and communication data storage applications. The fast random access speeds in RLDRAM II devices make them a viable alternative to SRAM devices at a lower cost. Additionally, RLDRAM II devices have minimal latency to support designs that require fast response times.

Interface Pins

RLDRAM II devices use interface pins such as data, clock, command, and address pins. There are two types of RLDRAM II memory: common I/O (CIO) and separate I/O (SIO). The data pins in RLDRAM II CIO device are bidirectional while the data pins in a RLDRAM II SIO device are uni-directional. Instead of bidirectional data strobes, RLDRAM II uses differential free-running read and write clocks to accompany the data. As in DDR SDRAM, data is sent and captured at twice the clock rate by transferring data on both the positive and negative edge of a clock. The commands and addresses still only use one active edge of a clock.

If the data pins are bidirectional, connect them to the Stratix and Stratix GX device DQ pins. If the data pins are uni-directional, connect the RLDRAM II device Q ports to the Stratix and Stratix GX device DQ pins and connect the D ports to any user I/O pins in I/O banks 3, 4, 7, and 8. RLDRAM II also uses active-high data mask pins for writes. You can connect DM pins to any of the I/O pins in the same bank as the DQ pins of the FPGA. When interfacing with SIO devices, connect the DM pins to any of the I/O pins in the same bank as the D pins. There is one DM pin per DQS/DQ group.

Connect the read clock pins (QK) to Stratix and Stratix GX device DQS pins. You must configure the DQS signals as bidirectional pins. However, since QK pins are output-only pins from the memory, RLDRAM memory interfacing in Stratix and Stratix GX devices requires that you ground the DQS and DQSn pin output enables. The Stratix and Stratix GX devices use the shifted QK signal from the DQS logic block to capture data. You can leave the QK# signal of the RLDRAM II device unconnected.

RLDRAM II devices have both input clocks (CK and CK#) and write clocks (DK and DK#). Use the external clock buffer to generate CK, CK#, DK, and DK# to meet the CK, CK#, DK, and DK# skew requirements from the RLDRAM II device. If you are interfacing with multiple RLDRAM II devices, perform IBIS simulations to analyze the loading effects on the clock pair.

You can use any of the user I/O pins for commands and addresses. RLDRAM II also offers QVLD pins to indicate the read data availability. Connect the QVLD pins to the Stratix and Stratix GX device DQVLD pins, listed in the pin table.

Read & Write Operations

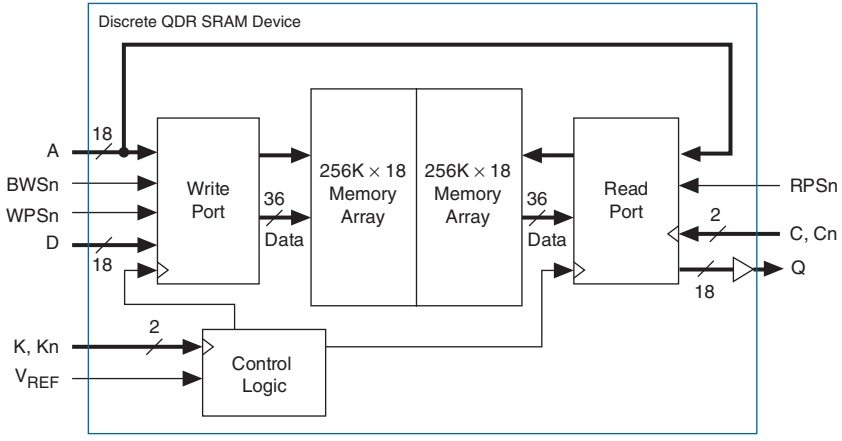
When reading from the RLDRAM II device, data is sent edge-aligned with the read clock QK or QK# signal. When writing to the RLDRAM II device, data must be center-aligned with the write clock (DK or DK# signal). The Stratix and Stratix GX device RLDRAM II interface uses the

same scheme as in DDR SDRAM interfaces whereby the dedicated circuitry is used during reads to center-align the data and the read clock inside the FPGA and the PLL center-aligns the data and write clock outputs. The data and clock relationship for reads and writes in RLDRAM II is similar to those in DDR SDRAM as already depicted in Figure 15–1 on page 15–3 and Figure 15–3 on page 15–4.

QDR & QDRII SRAM

QDR SRAM provides independent read and write ports that eliminate the need for bus turnaround. The memory uses two sets of clocks: K and Kn for write access, and optional C and Cn for read accesses, where Kn and Cn are the inverse of the K and C clocks, respectively. You can use differential HSTL I/O pins to drive the QDR SRAM clock into the Stratix and Stratix GX devices. The separate write data and read data ports permit a transfer rate up to four words on every cycle through the DDR circuitry. Stratix and Stratix GX devices support both burst-of-two and burst-of-four QDR SRAM architectures, with clock cycles up to 167 MHz using the 1.5-V HSTL Class I or Class II I/O standard. Figure 15–4 shows the block diagram for QDR SRAM burst-of-two architecture.

Figure 15–4. QDR SRAM Block Diagram for Burst-of-Two Architecture



QDRII SRAM is a second generation of QDR SRAM devices. It can transfer four words per clock cycle, fulfilling the requirements facing next-generation communications system designers. QDRII SRAM devices provide concurrent reads and writes, zero latency, and increased data throughput. Stratix and Stratix GX devices support QDRII SRAM at speeds up to 200 MHz since the timing requirements for QDRII SRAM are not as strict as QDR SRAM.

Interface Pins

QDR and QDRII SRAM uses two separate, uni-directional data ports for read and write operations, enabling quad data-rate data transfer. Both QDR and QDRII SRAM use shared address lines for reads and writes. Stratix and Stratix GX devices utilize dedicated DDR I/O circuitry for the input and output data bus and the K and Kn output clock signals.

Both QDR and QDRII SRAM burst-of-two devices sample the read address on the rising edge of the K clock and sample the write address on the rising edge of the Kn clock while QDR and QDRII SRAM burst-of-four devices sample both read and write addresses on the K clock's rising edge. You can use any of the Stratix and Stratix GX device user I/O pins in I/O banks 3, 4, 7, and 8 for the D write data ports, commands, and addresses.

QDR SRAM uses the following clock signals: input clocks K and Kn and output clocks C and Cn. In addition to the aforementioned two pairs of clocks, QDRII SRAM also uses echo clocks CQ and CQn. Clocks Cn, Kn, and CQn are logical complements of clocks C, K, and CQ respectively. Clocks C, Cn, K, and Kn are inputs to the QDRII SRAM while clocks CQ and CQn are outputs from the QDRII SRAM. Stratix and Stratix GX devices use single-clock mode for single-device QDR and QDRII SRAM interfacing where the K and Kn are used for both read and write operations, and the C and Cn clocks are unused. Use both C or Cn and K or Kn clocks when interfacing with a bank of multiple QDRII SRAM devices with a single controller.

You can generate C, Cn, K, and Kn clocks using any of the I/O registers in I/O banks 3, 4, 7, or 8 via the DDR registers. Due to strict skew requirements between K and Kn signals, use adjacent pins to generate the clock pair. Surround the pair with buffer pins tied to V_{CC} and ground for better noise immunity from other signals.

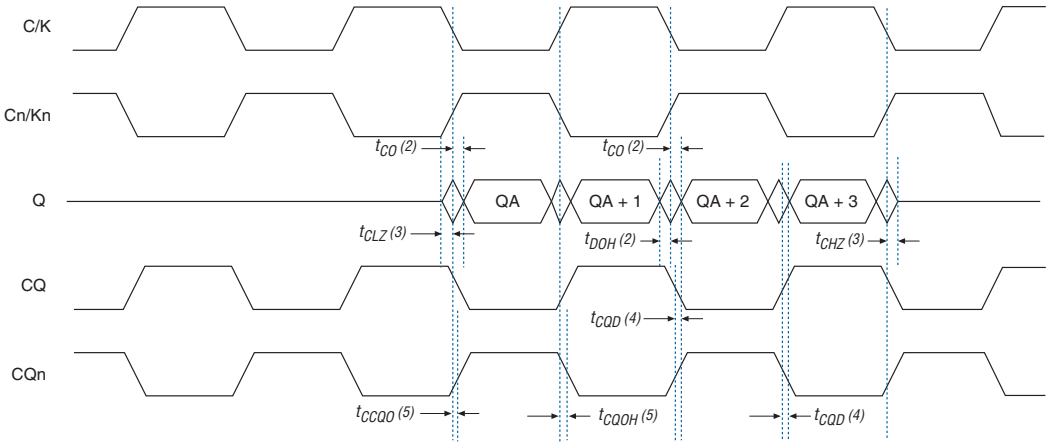
In general, all output signals to the QDR and QDRII SRAM should use the top and bottom banks (I/O banks 3, 4, 7, or 8). You can place the input signals from the QDR and QDRII SRAM in any I/O banks.

Read & Write Operations

Figure 15–5 shows the data and clock relationships in QDRII SRAM devices at the memory pins during reads. QDR and QDRII SRAM devices send data within a t_{CO} time after each rising edge of the input clock C or Cn in multi-clock mode, or the input clock K or Kn in single clock mode. Data is valid until t_{DOH} time, after each rising edge of the C or Cn in multi-

clock mode, or K or Kn in single clock mode. The edge-aligned CQ and CQn clocks accompany the read data for data capture in Stratix and Stratix GX devices.

Figure 15–5. Data & Clock Relationship During a QDRII SRAM Read *Note (1)*



Notes to Figure 15–5:

(1) The timing parameter nomenclature is based on the Cypress QDRII SRAM data sheet for CY7C1313V18.
(2) CO is the data clock-to-out time and t_{DOH} is the data output hold time between burst.
(3) t_{CLZ} and t_{CHZ} are bus turn-on and turn-off times respectively.
(4) t_{CQD} is the skew between CQn and data edges.
(5) t_{CQQO} and t_{CQOH} are skew between the C or Cn (or K or Kn in single-clock mode) and the CQ or CQn clocks.

When writing to QDRII SRAM devices, data is generated by the write clock, while the K clock is 90° shifted from the write clock, creating a center-aligned arrangement.

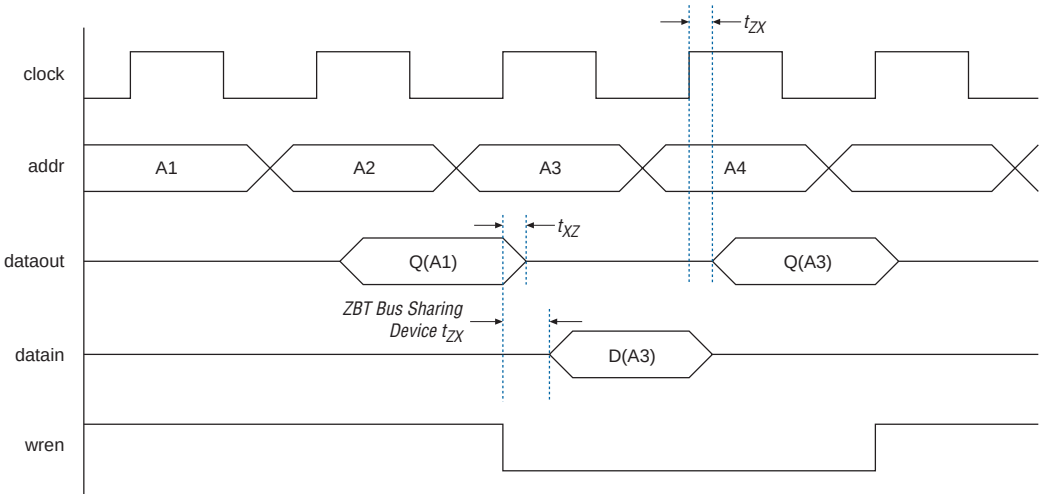
Go to **www.qdrsram.com** for the QDR SRAM and QDRII SRAM specifications. For more information on QDR and QDRII SRAM interfaces in Stratix and Stratix GX devices, see *AN 349: QDR SRAM Controller Reference Design for Stratix & Stratix GX Devices*.

ZBT SRAM

ZBT SRAM eliminate dead bus cycles when turning a bidirectional bus around between reads and writes or between writes and reads. ZBT allows for 100% bus utilization because ZBT SRAM can be read or written on every clock cycle. Bus contention can occur when shifting from a write cycle to a read cycle or vice versa with no idle cycles in between. ZBT SRAM allows small amounts of bus contention. To avoid bus contention, the output clock-to-low-impedance time (t_{ZX}) must be greater

than the clock-to-high-impedance time (t_{XZ}). Stratix and Stratix GX device I/O pins can interface with ZBT SRAM devices at up to 200 MHz and can meet ZBT t_{CO} and t_{SU} timing requirements by controlling phase delay in clocks to the OE or output and input registers using an enhanced PLL. Figure 15–6 shows a flow-through ZBT SRAM operation where A1 and A3 are read addresses and A2 and A4 are write addresses. For pipelined ZBT SRAM operation, data is delayed by another clock cycle. Stratix and Stratix GX devices support up to 200-MHz ZBT SRAM operation using the 2.5-V or 3.3-V LVTTL I/O standard.

Figure 15–6. t_{ZX} & t_{XZ} Timing Diagram



Interface Pins

ZBT SRAM uses one system clock input for all clocking purposes. Only the rising edge of this clock is used, since ZBT SRAM uses a single data rate scheme. The data bus, DQ, is bidirectional. There are three control signals to the ZBT SRAM: `RW_N`, `BW_N`, and `ADV_LD_N`. You can use any of the Stratix and Stratix GX device user I/O pins to interface to the ZBT SRAM device.

For more information on ZBT SRAM Interfaces in Stratix devices, see *AN 329: ZBT SRAM Controller Reference Design for Stratix & Stratix GX Devices*.

DDR Memory Support Overview

Table 15–1 shows the external RAM support in Stratix EP1S10 through EP1S40 devices and all Stratix GX devices. Table 15–2 shows the external RAM support in Stratix EP1S60 and EP1S80 devices.

Table 15–1. External RAM Support in Stratix EP1S10 through EP1S40 & All Stratix GX Devices

DDR Memory Type	I/O Standard	Maximum Clock Rate (MHz)						
		-5 Speed Grade	-6 Speed Grade		-7 Speed Grade		-8 Speed Grade	
		Flip-Chip	Flip-Chip	Wire-Bond	Flip-Chip	Wire-Bond	Flip-Chip	Wire-Bond
DDR SDRAM *(1)*, *(2)*	SSTL-2	200	167	133	133	100	100	100
DDR SDRAM - side banks *(2)*, *(3)*, *(4)*	SSTL-2	150	133	110	133	100	100	100
RLDRAM II *(4)*	1.8-V HSTL	200	*(5)*	*(5)*	*(5)*	*(5)*	*(5)*	*(5)*
QDR SRAM *(6)*	1.5-V HSTL	167	167	133	133	100	100	100
QDRII SRAM *(6)*	1.5-V HSTL	200	167	133	133	100	100	100
ZBT SRAM *(7)*	LVTTL	200	200	200	167	167	133	133

Notes to Table 15–1:

(1) These maximum clock rates apply if the Stratix device uses DQS phase-shift circuitry to interface with DDR SDRAM. DQS phase-shift circuitry is only available on the top and bottom I/O banks (I/O banks 3, 4, 7, and 8).
(2) For more information on DDR SDRAM, see *AN 342: Interfacing DDR SDRAM with Stratix & Stratix GX Devices*.
(3) DDR SDRAM is supported on the Stratix device side I/O banks (I/O banks 1, 2, 5, and 6) without dedicated DQS phase-shift circuitry. The read DQS signal is ignored in this mode.
(4) These performance specifications are preliminary.
(5) This device does not support RLDRAM II.
(6) For more information on QDR or QDRII SRAM, see *AN 349: QDR SRAM Controller Reference Design for Stratix & Stratix GX Devices*.
(7) For more information on ZBT SRAM, see *AN 329: ZBT SRAM Controller Reference Design for Stratix and Stratix GX Devices*.

Table 15–2. External RAM Support in Stratix EP1S60 & EP1S80 (Part 1 of 2)

DDR Memory Type	I/O Standard	Maximum Clock Rate (MHz)		
		-5 Speed Grade	-6 Speed Grade	-7 Speed Grade
DDR SDRAM *(1)*, *(2)*	SSTL-2	167	167	133
DDR SDRAM - side banks *(2)*, *(3)*	SSTL-2	150	133	133
QDR SRAM *(4)*	1.5-V HSTL	133	133	133
QDRII SRAM *(4)*	1.5-V HSTL	167	167	133

Table 15–2. External RAM Support in Stratix EP1S60 & EP1S80 (Part 2 of 2)

DDR Memory Type	I/O Standard	Maximum Clock Rate (MHz)		
		-5 Speed Grade	-6 Speed Grade	-7 Speed Grade
ZBT SRAM *(5)*	LVTTL	200	200	167

Notes to Table 15–2:

(1) These maximum clock rates apply if the Stratix device uses DQS phase-shift circuitry to interface with DDR SDRAM. DQS phase-shift circuitry is only available on the top and bottom I/O banks (I/O banks 3, 4, 7, and 8).
(2) For more information on DDR SDRAM, see *AN 342: Interfacing DDR SDRAM with Stratix & Stratix GX Devices*.
(3) DDR SDRAM is supported on the side banks (I/O banks 1, 2, 5, and 6) with no dedicated DQS phase-shift circuitry. The read DQS signal is ignored in this mode.
(4) For more information on QDR or QDRII SRAM, see *AN 349: QDR SRAM Controller Reference Design for Stratix & Stratix GX Devices*.
(5) For more information on ZBT SRAM, see *AN 329: ZBT SRAM Controller Reference Design for Stratix and Stratix GX Devices*.

Stratix and Stratix GX devices support the data strobe or read clock signal (DQS) used in DDR SDRAM, and RLDRAM II devices. DQS signals are associated with a group of data (DQ) pins.

Stratix and Stratix GX devices contain dedicated circuitry to shift the incoming DQS signals by 0°, 72°, and 90°. The DQS phase-shift circuitry uses a frequency reference to dynamically generate control signals for the delay chains in each of the DQS pins, allowing it to compensate for process, voltage, and temperature (PVT) variations. The dedicated circuitry also creates consistent margins that meet your data sampling window requirements.



Refer to the *DC & Switching Characteristics* chapter in volume 1 of the *Stratix Device Handbook* for frequency limits regarding the 72 and 90° phase shift for DQS.

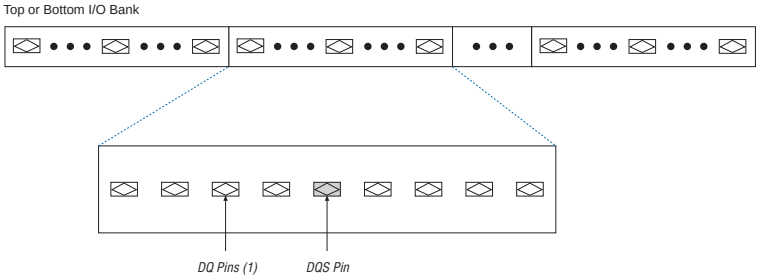
In addition to the DQS dedicated phase-shift circuitry, every I/O element (IOE) in Stratix and Stratix GX devices contains six registers and one latch to achieve DDR operation. There is also a programmable delay chain in the IOE that can help reduce contention when interfacing with ZBT SRAM devices.

DDR Memory Interface Pins

Stratix and Stratix GX devices use data (DQ), data strobe (DQS), and clock pins to interface with DDR SDRAM and RLDRAM II devices. This section explains the pins used in the DDR SDRAM and RLDRAM II interfaces. For QDR, QDRII, and ZBT SRAM interfaces, see the "External Memory Standards" section.

Figure 15–7 shows the DQ and DQS pins in ×8 mode.

Figure 15–7. Stratix & Stratix GX Device DQ & DQS Groups in × 8 Mode



Note to Figure 15–7:
(1) There are at least eight DQ pins per group.

Data & Data Strobe Pins

Stratix and Stratix GX data pins for the DDR memory interfaces are called DQ pins. The Stratix and Stratix GX device I/O banks at the top (I/O banks 3 and 4) and the bottom (I/O banks 7 and 8) of the device support DDR SDRAM and RLDRAM II up to 200 MHz. These pins support DQS signals with DQ bus modes of ×8, ×16, or ×32. Stratix and Stratix GX devices can support either bidirectional data strobes or uni-directional read clocks. Depending on the external memory interface, either the memory device's read data strobes or read clocks feed the DQS pins.

For ×8 mode, there are up to 20 groups of programmable DQS and DQ pins—10 groups in I/O banks 3 and 4 and 10 groups in I/O banks 7 and 8 (see Table 15–3). Each group consists of one DQS pin and a set of eight DQ pins.

For ×16 mode, there are up to eight groups of programmable DQS and DQ pins—four groups in I/O banks 3 and 4, and four groups in I/O banks 7 and 8. The EP1S20 device supports seven ×16 mode groups. The EP1S10 device does not support ×16 mode. All other devices support the full eight groups. See Table 15–3. Each group consists of one DQS and 16 DQ pins. In ×16 mode, `DQS1T`, `DQS3T`, `DQS6T`, and `DQS8T` pins on the top side of the device, and `DQS1B`, `DQS3B`, `DQS6B`, and `DQS8B` pins on the

bottom side of the device are dedicated DQS pins. The DQS2T, DQS7T, DQS2B, and DQS7B pins are dedicated DQS pins for ×32 mode, and each group consists of one DQS and 32 DQ pins.

Table 15–3. DQS & DQ Bus Mode Support *Note (1)*

Device	Package	Number of ×8 Groups	Number of ×16 Groups	Number of ×32 Groups
EP1S10	672-pin BGA 672-pin FineLine BGA®	12 *(2)*	0	0
	484-pin FineLine BGA 780-pin FineLine BGA	16 *(3)*	0	4
EP1S20	484-pin FineLine BGA	18 *(4)*	7 *(5)*	4
	672-pin BGA 672-pin FineLine BGA	16 *(3)*	7 *(5)*	4
	780-pin FineLine BGA	20	7 *(5)*	4
EP1S25	672-pin BGA 672-pin FineLine BGA	16 *(3)*	8	4
	780-pin FineLine BGA 1,020-pin FineLine BGA	20	8	4
EP1S30	956-pin BGA 780-pin FineLine BGA 1,020-pin FineLine BGA	20	8	4
EP1S40	956-pin BGA 1,020-pin FineLine BGA 1,508-pin FineLine BGA	20	8	4
EP1S60	956-pin BGA 1,020-pin FineLine BGA 1,508-pin FineLine BGA	20	8	4
EP1S80	956-pin BGA 1,508-pin FineLine BGA 1,923-pin FineLine BGA	20	8	4

Notes to Table 15–3:

(1) For V_{REF} guidelines, see the *Selectable I/O Standards in Stratix & Stratix GX Devices* chapter of the *Stratix Device Handbook, Volume 2* or the *Stratix GX Handbook, Volume 2*.
(2) These packages have six groups in I/O banks 3 and 4 and six groups in I/O banks 7 and 8.
(3) These packages have eight groups in I/O banks 3 and 4 and eight groups in I/O banks 7 and 8.
(4) This package has nine groups in I/O banks 3 and 4 and nine groups in I/O banks 7 and 8.
(5) These packages have three groups in I/O banks 3 and 4 and four groups in I/O banks 7 and 8.

The DQS pins are marked in the Stratix and Stratix GX device pin table as DQS[9..0]T or DQS[9..0]B, where T stands for top and B for bottom. The corresponding DQ pins are marked as DQ[9..0]T[7..0], where [9..0] indicates which DQS group the pins belong to. The numbering scheme starts from right to left on the package bottom view. When not used as DQ or DQS pins, these pins are available as user I/O pins.

You can also create a design in a mode other than the ×8, ×16, or ×32 mode. The Quartus® II software uses the next larger mode with the unused DQ pins available as regular use I/O pins. For example, if you create a design for ×9 mode for an RLDRAM II interface (nine DQ pins driven by one DQS pin), the Quartus II software implements a ×16 mode with seven DQ pins unconnected to the DQS bus. These seven unused DQ pins can be used as regular I/O pins.



On the top and bottom side of the device, the DQ and DQS pins must be configured as bidirectional DDR pins to enable the DQS phase-shift circuitry. If you only want to use the DQ and/or DQS pins as inputs, you need to set the output enable of the DQ and/or DQS pins to ground. Use the altdqs and altdq megafunctions to configure the DQS and DQ pins, respectively. However, you should use the Altera® IP Toolbench to create the data path for your memory interfaces.

Stratix and Stratix GX device side I/O banks (I/O banks 1, 2, 5, and 6) support SDR SDRAM, ZBT SRAM, QDR SRAM, QDRII SRAM, and DDR SDRAM interfaces and can use any of the user I/O pins in these banks for the interface. Since these I/O banks do not have any dedicated circuitry for memory interfacing, they can support DDR SDRAM up to 150 MHz in -5 speed grade devices. However, these I/O banks do not support the HSTL-18 Class II I/O standard, which is required to interface with RLDRAM II.

Clock Pins

You can use any of the DDR I/O registers in the top or bottom bank of the device (I/O banks 3, 4, 7, or 8) to generate clocks to the memory device. You can also use any of the DDR I/O registers in the side I/O banks 1, 2, 5, or 6 to generate clocks for DDR SDRAM interfaces on the side I/O banks (not using the DQS circuitry).

Command & Address Pins

You can use any of the user I/O pins in the top or bottom bank of the device (I/O banks 3, 4, 7, or 8) for commands and addresses. For DDR SDRAM, you can also use any of the user I/O pins in the side I/O banks 1, 2, 5, or 6, regardless of whether you use the DQS phase-shift circuitry or not.

Other Pins (Parity, DM, ECC & QVLD Pins)

You can use any of the DQ pins for the parity pins in Stratix and Stratix GX devices. However, this may mean that you are using the next larger DQS/DQ mode. For example, if you need a parity bit for each byte of data, you are actually going to have nine DQ pins per DQS pin. The Quartus II software then implements a ×16 mode, with the seven unused DQ pins available as user I/O pins.

The data mask (DM) pins are only required when writing to DDR SDRAM and RLDRAM II devices. A low signal on the DM pins indicates that the write is valid. If the DM signal is high, the memory masks the DQ signals. You can use any of the I/O pins in the same bank as the DQ pins for the DM signals. Each group of DQS and DQ signals requires a DM pin. The DDR register, clocked by the –90° shifted clock, creates the DM signals, similar to DQ output signals.

Some DDR SDRAM devices support error correction coding (ECC), which is a method of detecting and automatically correcting errors in data transmission. Connect the DDR ECC pins to a Stratix and Stratix GX device DQS/DQ group. In 72-bit DDR SDRAM, there are eight ECC pins in addition to the 64 data pins. The memory controller needs extra logic to encode and decode the ECC data.

QVLD pins are used in RLDRAM II interfacing to indicate the read data availability. There is one QVLD pin per RLDRAM II device. A high on QVLD indicates that the memory is outputting the data requested. Similar to DQ inputs, this signal is edge-aligned with the RLDRAM II read clocks, QK and QK#, and is sent half a clock cycle before data starts coming out of the memory. You can connect QVLD pins to any of the I/O pins in the same bank as the DQ pins for the QVLD signals.

DQS Phase-Shift Circuitry

Two single phase-shifting reference circuits are located on the top and bottom of the Stratix and Stratix GX devices. Each circuit is driven by a system reference clock that is of the same frequency as the DQS signal. Clock pins `CLK[15..12]p` feed the phase-shift circuitry on the top of the device and clock pins `CLK[7..4]p` feed the phase-shift circuitry on the

bottom of the device. The phase-shift circuitry cannot be fed from other sources such as the LE or the PLL internal output clocks. This phase-shift circuitry is used for DDR SDRAM and RLDRAM II interfaces. For best performance, turn off the input reference clock to the DQS phase-shift circuitry when reading from the DDR SDRAM or RLDRAM II. This is to avoid any DLL jitter incorrectly shifting the DQS signal while the FPGA is capturing data.



The I/O pins in I/O banks 1, 2, 5, and 6 can interface with the DDR SDRAM at up to 150 MHz. See *AN 342: Interfacing DDR SDRAM with Stratix & Stratix GX Devices*.

A compensated delay element on each DQS pin allows for either a 90° or a 72° phase shift, which automatically centers input DQS signals with the data valid window of their corresponding DQ data signals. The DQS signals drive a local DQS bus within the top and bottom I/O banks. This DQS bus is an additional resource to the I/O clocks and clocks DQ input registers with the DQS signal.



Refer to the *DC & Switching Characteristics* chapter in volume 1 of the *Stratix Device Handbook* for frequency limits regarding the 72 and 90° phase shift for DQS.

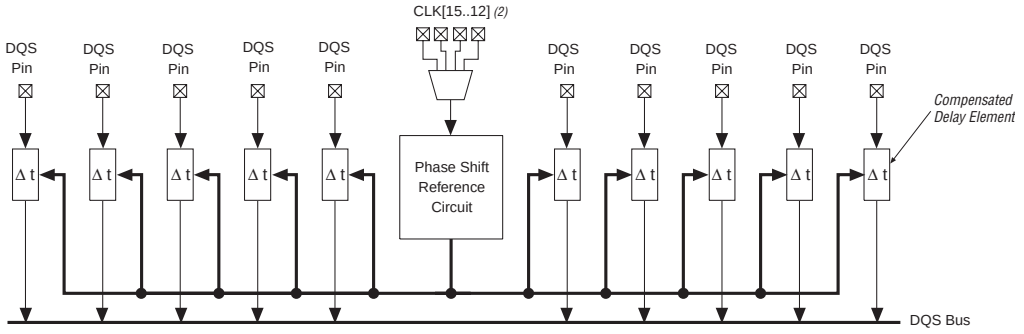
The phase-shifting reference circuit on the top of the device controls the compensated delay elements for all 10 DQS pins located at the top of the device. The phase-shifting reference circuit on the bottom of the device controls the compensated delay elements for all 10 DQS pins located on the bottom of the device. All 10 delay elements (DQS signals) on either the top or bottom of the device shift by the same degree amount. For example, all 10 DQS pins on the top of the device can be shifted by 90° and all 10 DQS pins on the bottom of the device can be shifted by 72°. The reference circuit requires a maximum of 256 system reference clock cycles to set the correct phase on the DQS delay elements.



This applies only to the initial phase calculation. Altera recommends that you enable the DLL during the refresh cycle of the DDR SDRAM. Enabling the DLL for the duration of the minimum refresh time is sufficient for recalculating the phase shift.

Figure 15–8 shows the phase-shift reference circuit control of each DQS delay shift on the top of the device. This same circuit is duplicated on the bottom of the device.

Figure 15–8. DQS & DQSn Pins & the DQS Phase-Shift Circuitry *Note (1)*



Notes to Figure 15–8:

(1) There are up to 10 DQS and DQSn pins available on the top or the bottom of the Stratix and Stratix GX devices.

(2) Clock pins `CLK[15..12]p` feed the phase-shift circuitry on the top of the device and clock pins `CLK[7..4]p` feed the phase circuitry on the bottom of the device. The reference clock can also be used in the logic array.

The phase-shift circuitry is only used during read transactions where the DQS pins are acting as input clocks or strobes. The phase-shift circuitry can shift the incoming DQS signal by 0°, 72°, and 90°. The shifted DQS signal is then inverted and used as a clock or a strobe at the DQ IOE input registers.

Refer to the *DC & Switching Characteristics* chapter in volume 1 of the *Stratix Device Handbook* for frequency limits regarding the 72 and 90° phase shift for DQS.

The DQS phase-shift circuitry is bypassed when 0° shift is chosen. The routing delay between the pins and the IOE registers is matched with high precision for both the DQ and DQS signal when the 72° or 90° phase shift is used. With the 0° phase shift, the skew between DQ and the DQS signals at the IOE register has been minimized. See Table 15–4 for the Quartus II software reported number on the DQ and DQS path to the IOE when the DQS is set to 0° phase shift.

Table 15–4. Quartus II Reported Number on the DQS Path to the IOE *Note (1)*

Speed Grade	DQ2IOE	DQS2IOE	Unit
-5	0.908	1.008	ns
-6	0.956	1.061	ns
-7	1.098	1.281	ns

Table 15–4. Quartus II Reported Number on the DQS Path to the IOE *Note (1)*

Speed Grade	DQ2IOE	DQS2IOE	Unit
-8	1.293	1.635	ns

Note to Table 15–4:
(1) These are reported by Quartus II version 4.0. Check the latest version of the Quartus II software for the most current information.

To generate the correct phase shift, you must provide a clock signal of the same frequency as the DQS signal to the DQS phase-shift circuitry. Any of the `CLK[15..12]p` clock pins can feed the phase circuitry on the top of the device (I/O banks 3 and 4) and any of the `CLK[7..4]p` clock pins can feed the phase circuitry on the bottom of the device (I/O banks 7 and 8). Both the top and bottom phase-shift circuits need unique clock pins for the reference clock. You cannot use any internal clock sources to feed the phase-shift circuitry, but you can route internal clock sources off-chip and then back into one of the allowable clock input pins.

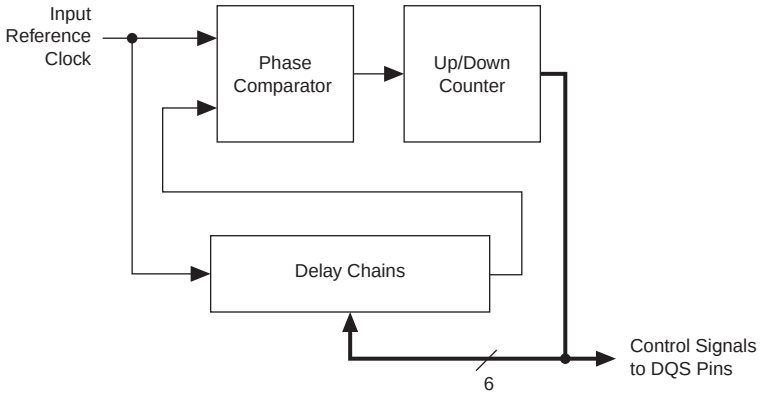
DLL

The DQS phase-shift circuitry uses a DLL to dynamically measure the clock period needed by the DQS pin (see Figure 15–9). The DQS phase-shift circuitry then uses the clock period to generate the correct phase shift. The DLL in the Stratix and Stratix GX devices DQS phase-shift circuitry can operate between 100 and 200 MHz. The phase-shift circuitry needs a maximum of 256 clock cycles to calculate the correct phase shift. Data sent during these clock cycles may not be properly captured.



You can still use the DQS phase-shift circuitry for DDR SDRAM interfaces that are less than 100 MHz. The DQS signal is shifted by about 2.5 ns. This shifted DQS signal is not in the center of the DQ signals, but it is shifted enough to capture the correct data in this low-frequency application.

Figure 15–9. Simplified Diagram of the DQS Phase-Shift Circuitry



The input reference clock goes into the DLL to a chain of delay elements. The phase comparator compares the signal coming out of the end of the delay element chain to the input reference clock. The phase comparator then issues the `upndn` signal to the up/down counter. This signal increments or decrements a six-bit delay setting (control signals to DQS pins) that increases or decreases the delay through the delay element chain to bring the input reference clock and the signals coming out of the delay element chain in phase.

The shifted DQS signal then goes to the DQS bus to clock the IOE input registers of the DQ pins. It cannot go into the logic array for other purposes.

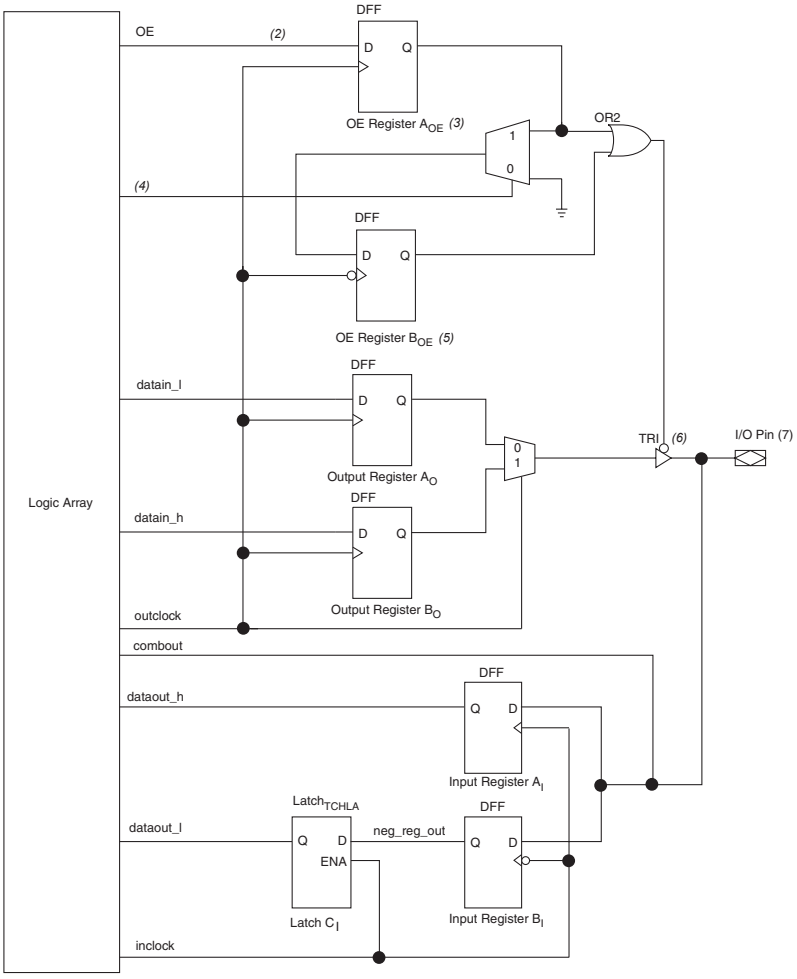
For external memory interfaces that use a bidirectional read strobe like DDR SDRAM, the DQS signal is low before going to or coming from a high-impedance state (see Figure 15–1 on page 15–3). The state where DQS is low just after a high-impedance state is called the preamble and the state where DQS is low just before it returns to high-impedance state is called the postamble. There are preamble and postamble specifications for both read and write operations in DDR SDRAM. To ensure data is not lost when there is noise on the DQS line at the end of a read postamble time, you need to add soft postamble circuitry to disable the clocks at the DQ IOE registers.

For more information, the DQS Postamble soft logic is described in *AN 342: Interfacing DDR SDRAM with Stratix & Stratix GX Devices*. The Altera DDR SDRAM controller MegaCore® generates this logic as open-source code.

DDR Registers

Each Stratix and Stratix GX IOE contains six registers and one latch. Two registers and a latch are used for input, two registers are used for output, and two registers are used for output enable control. The second output enable register provides the write preamble for the DQS strobe in the DDR external memory interfaces. This negative-edge output enable register extends the high-impedance state of the pin by a half clock cycle to provide the external memory's DQS preamble time specification. Figure 15–10 shows the six registers and the latch in the Stratix and Stratix GX IOE and Figure 15–11 shows how the second OE register extends the DQS high impedance state by half a clock cycle during a write operation.

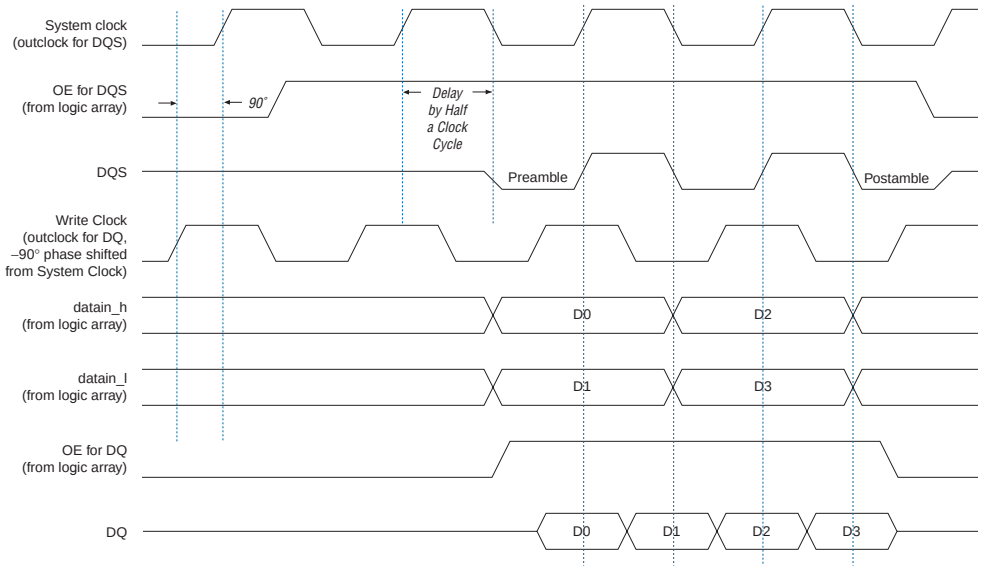
Figure 15–10. Bidirectional DDR I/O Path in Stratix & Stratix GX Devices *Note (1)*



Notes to Figure 15–10:

(1) All control signals can be inverted at the IOE. No programmable delay chains are shown in this diagram.
(2) The OE signal is active low, but the Quartus II software implements this as active high and automatically adds an inverter before input to the A_{OE} register during compilation.
(3) The A_{OE} register generates the enable signal for general-purpose DDR I/O applications.
(4) This select line is to choose whether the OE signal should be delayed by half-a-clock cycle.
(5) The B_{OE} register generates the delayed enable signal for the write strobes and write clock for memory interfaces.
(6) The tristate enable is active low by default. You can design it to be active high. The combinational control path for the tristate is not shown in this diagram.
(7) You can also have combinational output to the I/O pin; this path is not shown in the diagram.

Figure 15–11. Extending the OE Disable by Half-a-Clock Cycle for a Write Transaction *Note (1)*

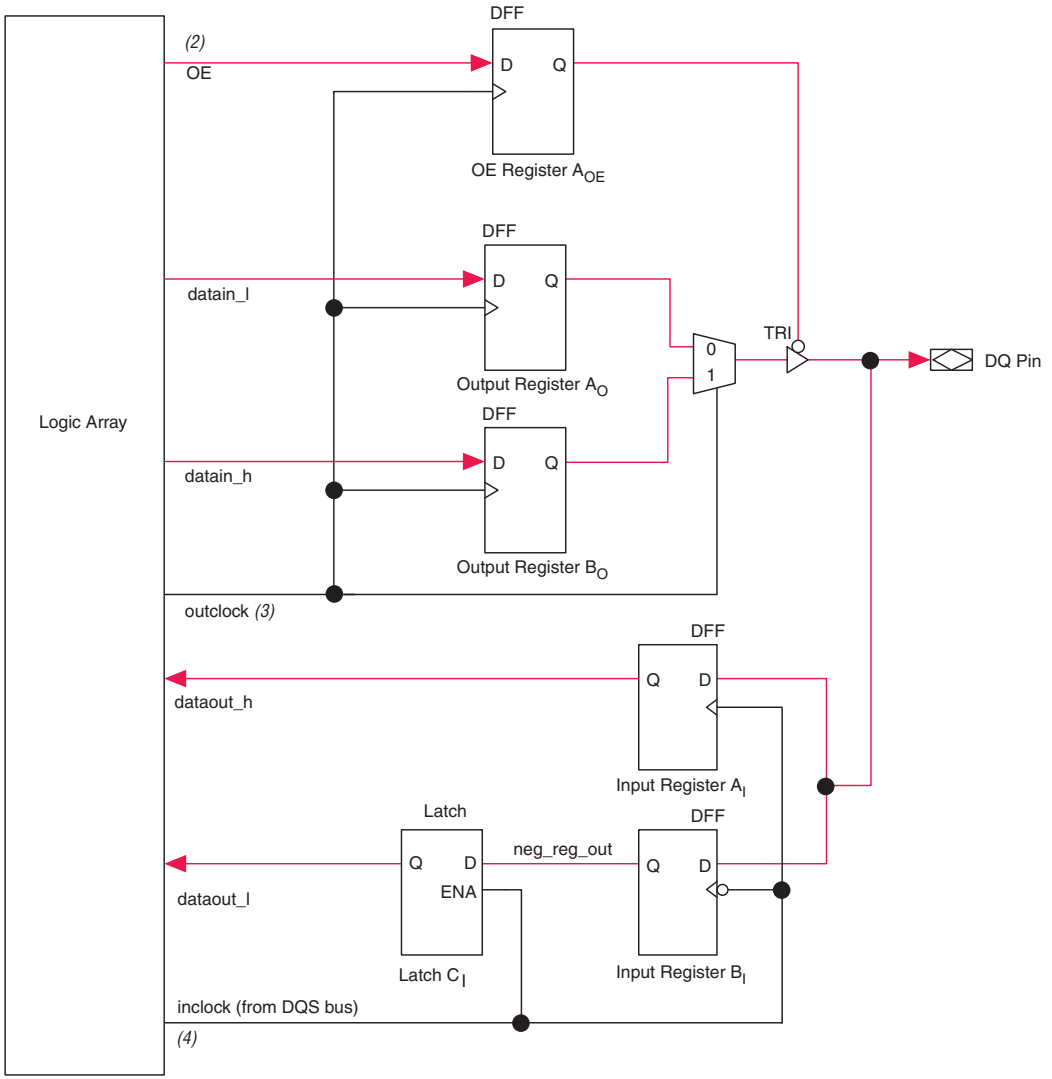


Note to Figure 15–11:

(1) The waveform reflects the software simulation result. The OE signal is an active low on the device. However, the Quartus II software implements this signal as an active high and automatically adds an inverter before the A_{OE} register D input.

Figures 15–12 and 15–13 summarize the IOE registers used for the DQ and DQS signals.

Figure 15–12. DQ Configuration in Stratix & Stratix GX IOE *Note (1)*



Notes to Figure 15–12:

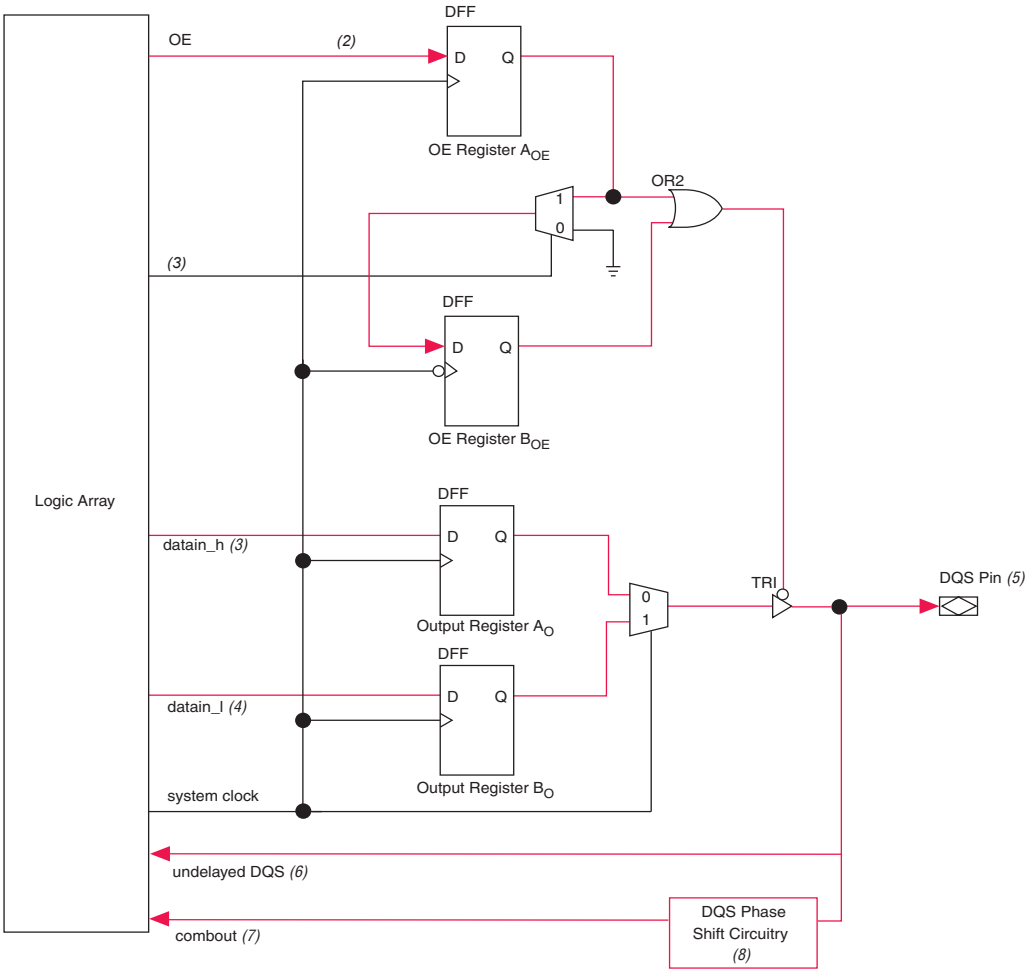
(1) You can use the `altdq` megafunction to generate the DQ signals.

(2) The `OE` signal is active low, but the Quartus II software implements this as active high and automatically adds an inverter before the `OE` register A_{OE} during compilation.

(3) The `outclock` signal is phase shifted –90° from the system clock.

(4) The shifted DQS signal must be inverted before going to the IOE. The inversion is automatic if you use the `altdq` megafunction to generate the DQ signals.

Figure 15–13. DQS Configuration in Stratix & Stratix GX IOE *Note (1)*



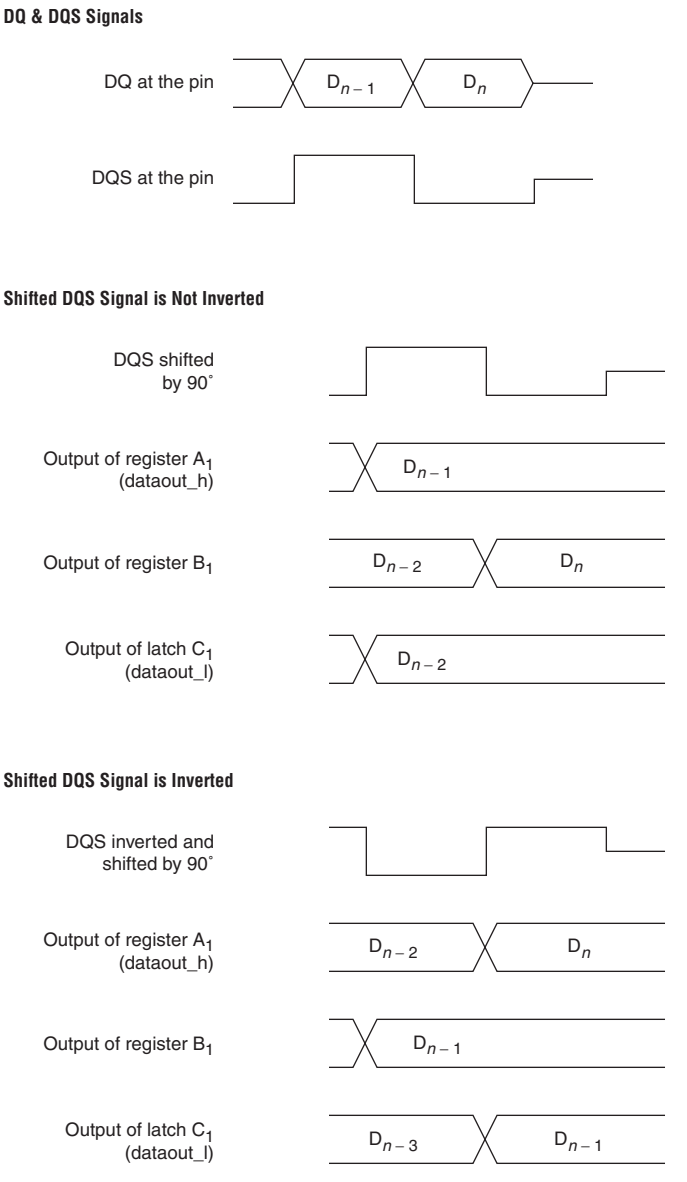
Notes to Figure 15–13:

(1) You can use the `altdqs` megafunction to generate the DQS signals.
(2) The `OE` signal is active low, but the Quartus II software implements this as active high and automatically adds an inverter before `OE` register A_{OE} during compilation.
(3) The select line can be chosen in the `altdqs` MegaWizard Plug-In Manager.
(4) The `datain_l` and `datain_h` pins are usually connected to V_{CC} and ground, respectively.
(5) DQS postamble handling is not shown in this diagram. For more information, see *AN 342: Interfacing DDR SDRAM with Stratix & Stratix GX Devices*.
(6) This undelayed DQS signal goes to the LE for the soft postamble circuitry.
(7) You must invert this signal before it reaches the DQ IOE. This signal is automatically inverted if you use the `altdq` megafunction to generate the DQ signals. Connect this port to the `inclock` port in the `altdq` megafunction.
(8) DQS phase-shift circuitry is only available on DQS pins.

The Stratix and Stratix GX DDR IOE structure requires you to invert the incoming DQS signal by using a NOT gate to ensure proper data transfer. The `altdq` megafunction automatically adds the inverter when it generates the DQ signals. As shown in Figure 15–10, the `inclock` signal's rising edge clocks the A_I register, `inclock` signal's falling edge clocks the B_I register, and latch C_I is opened when `inclock` is one. In a DDR memory read operation, the last data coincides with DQS being low. If you do not invert the DQS pin, you do not get this last data because the latch does not open until the next rising edge of the DQS signal. The NOT gate is inserted automatically if the `altdq` megafunction is used; otherwise you need to add the NOT gate manually.

Figure 15–14 shows waveforms of the circuit shown in Figure 15–12. The second set of waveforms in Figure 15–14 shows what happens if the shifted DQS signal is not inverted; the last data, D_n, does not get latched into the logic array as DQS goes to tristate after the read postamble time. The third set of waveforms in Figure 15–14 shows a proper read operation with the DQS signal inverted after the 90° shift; the last data D_n does get latched. In this case the outputs of register A_I and latch C_I, which correspond to `dataout_h` and `dataout_l` ports, are now switched because of the DQS inversion.

Figure 15–14. DQ Captures with Non-Inverted & Inverted Shifted DQS



PLL

When using the Stratix and Stratix GX top and bottom I/O banks (I/O banks 3, 4, 7, or 8) to interface with a DDR memory, at least one PLL with two outputs is needed to generate the system clock and the write clock. The system clock generates the DQS write signals, commands, and addresses. The write clock is –90° shifted from the system clock and generates the DQ signals during writes.

When using the Stratix and Stratix GX side I/O banks 1, 2, 5, or 6 to interface with DDR SDRAM devices, two PLLs may be needed per I/O bank for best performance. The side I/O banks do not have dedicated circuitry, so one PLL captures data from the DDR SDRAM and another PLL generates the write signals, commands, and addresses to the DDR SDRAM device. Stratix and Stratix GX devices side I/O banks can support DDR SDRAM up to 150 MHz.



For more information, see *AN 342: Interfacing DDR SDRAM with Stratix & Stratix GX Devices*.

Conclusion

Stratix and Stratix GX devices support SDR SDRAM, DDR SDRAM, RLDRAM II, QDR SDRAM, QDRII SRAM, and ZBT SRAM external memories. Stratix and Stratix GX devices feature high-speed interfaces that transfer data between external memory devices at up to 200 MHz/400 Mbps. Phase-shift circuitry in the Stratix and Stratix GX devices allows you to ensure that clock edges are properly aligned.

Section IV. I/O Standards

This section provides information about the I/O standards and interfaces for Stratix and Stratix® GX devices.

This section includes the following chapters:

- Chapter 16, Selectable I/O Standards in Stratix & Stratix GX Devices
- Chapter 17, High-Speed Source-Synchronous Differential I/O Interfaces in Stratix GX Devices

Revision History

The table below shows the revision history for Chapters 16 and 17.

Chapter(s)	Date / Version	Changes Made
16	June 2006, v3.4	Chapter updated as part of the *Stratix Device Handbook* update.
	July 2005, v3.3	Updated as part of the *Stratix Device Handbook* update.
	January 2005, v3.2	Added document to *Stratix GX Device Handbook*.
17	June 2006, v1.2	• Updated "DPA Input Support" section. • Updated "Receiver Data Realignment In DPA Mode" section. • Updated the "Software Support" section, including updating the MegaWizard Plug-In Manager figures to match the Quartus II software GUI.
	August 2005, v1.1	Updated Table 17-3.

16. Selectable I/O Standards in Stratix & Stratix GX Devices

S52004-3.4

Introduction

The proliferation of I/O standards and the need for higher I/O performance have made it critical that devices have flexible I/O capabilities. Stratix® and Stratix GX programmable logic devices (PLDs) feature programmable I/O pins that support a wide range of industry I/O standards, permitting increased design flexibility. These I/O capabilities enable fast time-to-market and high-performance solutions to meet the demands of complex system designs. Additionally, Stratix and Stratix GX devices simplify system board design and make it easy to connect to microprocessors, peripherals, memories, gate arrays, programmable logic circuits, and standard logic functions.

This chapter provides guidelines for using one or more industry I/O standards in Stratix and Stratix GX devices, including:

- Stratix and Stratix GX I/O standards
- High-speed interfaces
- Stratix and Stratix GX I/O banks
- Programmable current drive strength
- Hot socketing
- Differential on-chip termination
- I/O pad placement guidelines
- Quartus® II software support

Stratix & Stratix GX I/O Standards

Stratix and Stratix GX devices support a wide range of industry I/O standards as shown in the *Stratix Device Family Data Sheet* section in the *Stratix Device Handbook, Volume 1* and the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1*. Several applications that use these I/O standards are listed in Table 16–1.

Table 16–1. I/O Standard Applications & Performance (Part 1 of 2) Note (1)

I/O Standard	Application	Performance
3.3-V LVTTL/LVCMOS	General purpose	350 MHz
2.5-V LVTTL/LVCMOS	General purpose	350 MHz
1.8-V LVTTL/LVCMOS	General purpose	250 MHz
1.5-V LVCMOS	General purpose	225 MHz
PCI/CompactPCI	PC/embedded systems	66 MHz

Table 16–1. I/O Standard Applications & Performance (Part 2 of 2) Note (1)

I/O Standard	Application	Performance
PCI-X 1.0	PC/embedded systems	133 MHz
AGP 1× and 2×	Graphics processors	66 to 133 MHz
SSTL-3 Class I and II	SDRAM	167 MHz
SSTL-2 Class I and II	DDR I SDRAM	160 to 400 Mbps
HSTL Class I	QDR SRAM/SRAM/CSIX	150 to 225 MHz
HSTL Class II	QDR SRAM/SRAM/CSIX	150 to 250 MHz
Differential HSTL	Clock interfaces	150 to 225 MHz
GTL	Backplane driver	200 MHz
GTL+	Pentium processor interface	133 to 200 MHz
LVDS	Communications	840 Mbps
HyperTransport technology	Motherboard interfaces	800 Mbps
LVPECL	PHY interface	840 Mbps
PCML	Communications	840 Mbps
Differential SSTL-2	DDR I SDRAM	160 to 400 Mbps
CTT	Back planes and bus interfaces	200 MHz

Note to Table 16–1:

(1) These performance values are dependent on device speed grade, package type (flip-chip or wirebond) and location of I/Os (top/bottom or left/right). See the *DC & Switching Characteristics* chapter of the *Stratix Device Handbook, Volume 1*.

3.3-V Low Voltage Transistor-Transistor Logic (LVTTL) - EIA/JEDEC Standard JESD8-B

The 3.3-V LVTTL I/O standard is a general-purpose, single-ended standard used for 3.3-V applications. The LVTTL standard defines the DC interface parameters for digital circuits operating from a 3.0-V or 3.3-V power supply and driving or being driven by LVTTL-compatible devices.

The LVTTL input standard specifies a wider input voltage range of $-0.5\ V \leq V_I \leq 3.8\ V$. Altera allows an input voltage range of $-0.5\ V \leq V_I \leq 4.1\ V$. The LVTTL standard does not require input reference voltages or board terminations.

Stratix and Stratix GX devices support both input and output levels for 3.3-V LVTTL operation.

3.3-V LVCMOS - EIA/JEDEC Standard JESD8-B

The 3.3-V low voltage complementary metal oxide semiconductor (LVCMOS) I/O standard is a general-purpose, single-ended standard used for 3.3-V applications. The LVCMOS standard defines the DC interface parameters for digital circuits operating from a 3.0-V or 3.3-V power supply and driving or being driven by LVCMOS-compatible devices.

The LVCMOS standard specifies the same input voltage requirements as LVTTL (–0.5 V $\leq V_I \leq$ 3.8 V). The output buffer drives to the rail to meet the minimum high-level output voltage requirements. The 3.3-V I/O standard does not require input reference voltages or board terminations.

Stratix and Stratix GX devices support both input and output levels for 3.3-V LVCMOS operation.

2.5-V LVTTL Normal Voltage Range - EIA/JEDEC Standard EIA/JESD8-5

The 2.5-V I/O standard is used for 2.5-V LVTTL applications. This standard defines the DC interface parameters for high-speed, low-voltage, non-terminated digital circuits driving or being driven by other 2.5-V devices. The input and output voltage ranges are:

- The 2.5-V normal range input standards specify an input voltage range of – 0.3 V $\leq V_I \leq$ 3.0 V.
- The normal range minimum high-level output voltage requirement (V_{OH}) is 2.1 V.

Stratix and Stratix GX devices support both input and output levels for 2.5-V LVTTL operation.

2.5-V LVCMOS Normal Voltage Range - EIA/JEDEC Standard EIA/JESD8-5

The 2.5-V I/O standard is used for 2.5-V LVCMOS applications. This standard defines the DC interface parameters for high-speed, low-voltage, non-terminated digital circuits driving or being driven by other 2.5-V parts. The input and output voltage ranges are:

- The 2.5-V normal range input standards specify an input voltage range of – 0.5 V $\leq V_I \leq$ 3.0 V.
- The normal range minimum V_{OH} requirement is 2.1 V.

Stratix and Stratix GX devices support both input and output levels for 2.5-V LVCMOS operation.

1.8-V LVTTL Normal Voltage Range - EIA/JEDEC Standard EIA/JESD8-7

The 1.8-V I/O standard is used for 1.8-V LVTTL applications. This standard defines the DC interface parameters for high-speed, low-voltage, non-terminated digital circuits driving or being driven by other 1.8-V parts. The input and output voltage ranges are:

- The 1.8-V normal range input standards specify an input voltage range of – 0.5 V $\leq V_I \leq$ 2.3 V.
- The normal range minimum V_{OH} requirement is V_{CCIO} – 0.45 V.

Stratix and Stratix GX devices support both input and output levels for 1.8-V LVTTL operation.

1.8-V LVCMOS Normal Voltage Range - EIA/JEDEC Standard EIA/JESD8-7

The 1.8-V I/O standard is used for 1.8-V LVCMOS applications. This standard defines the DC interface parameters for high-speed, low-voltage, non-terminated digital circuits driving or being driven by other 1.8-V devices. The input and output voltage ranges are:

- The 1.8-V normal range input standards specify an input voltage range of – 0.5 V $\leq V_I \leq$ 2.5 V.
- The normal range minimum V_{OH} requirement is V_{CCIO} – 0.45 V.

Stratix and Stratix GX devices support both input and output levels for 1.8-V LVCMOS operation.

1.5-V LVCMOS Normal Voltage Range - EIA/JEDEC Standard JESD8-11

The 1.5-V I/O standard is used for 1.5-V applications. This standard defines the DC interface parameters for high-speed, low-voltage, non-terminated digital circuits driving or being driven by other 1.5-V devices. The input and output voltage ranges are:

- The 1.5-V normal range input standards specify an input voltage range of – 0.5 V $\leq V_I \leq$ 2.0 V.
- The normal range minimum V_{OH} requirement is 1.05 V.

Stratix and Stratix GX devices support both input and output levels for 1.5-V LVCMOS operation.

1.5-V HSTL Class I & II - EIA/JEDEC Standard EIA/JESD8-6

The high-speed transceiver logic (HSTL) I/O standard is used for applications designed to operate in the 0.0- to 1.5-V HSTL logic switching range. This standard defines single ended input and output specifications for all HSTL-compliant digital integrated circuits. The single ended input standard specifies an input voltage range of $-0.3\ V \le V_I \le V_{CCIO} + 0.3\ V$. Stratix and Stratix GX devices support both input and output levels specified by the 1.5-V HSTL I/O standard. The input clock is implemented using dedicated differential input buffers. Two single-ended output buffers are automatically programmed to have opposite polarity so as to implement a differential output clock. Additionally, the 1.5-V HSTL I/O standard in Stratix and Stratix GX devices is compatible with the 1.8-V HSTL I/O standard in APEX™ 20KE and APEX 20KC devices because the input and output voltage thresholds are compatible. See Figures 16–1 and 16–2. Stratix and Stratix GX devices support both input and output levels with V_{REF} and V_{TT}.

Figure 16–1. HSTL Class I Termination

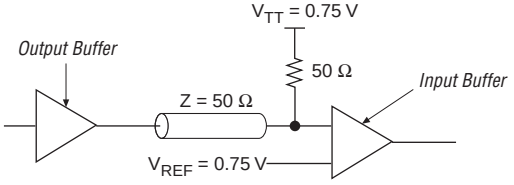
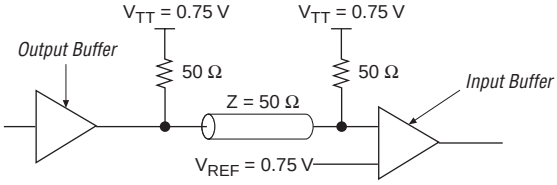


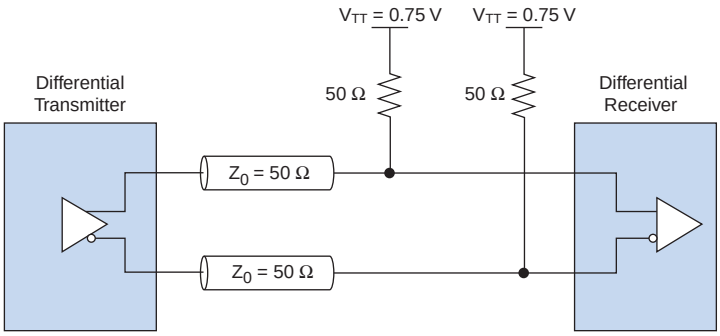
Figure 16–2. HSTL Class II Termination



1.5-V Differential HSTL - EIA/JEDEC Standard EIA/JESD8-6

The differential HSTL I/O standard is used for applications designed to operate in the 0.0- to 1.5-V HSTL logic switching range such as quad data rate (QDR) memory clock interfaces. The differential HSTL specification is the same as the single ended HSTL specification. The standard specifies an input voltage range of $-0.3\ V \le V_I \le V_{CCIO} + 0.3\ V$. Differential HSTL does not require an input reference voltage, however, it does require a 50 Ω resistor termination resistor to V_{TT} at the input buffer (see Figure 16–3). Stratix and Stratix GX devices support both input and output clock levels for 1.5-V differential HSTL. The input clock is implemented using dedicated differential input buffer. Two single-ended output buffers are automatically programmed to have opposite polarity so as to implement a differential output clock.

Figure 16–3. 1.5-V Differential HSTL Class I Termination



3.3-V PCI Local Bus - PCI Special Interest Group PCI Local Bus Specification Rev. 2.3

The PCI local bus specification is used for applications that interface to the PCI local bus, which provides a processor-independent data path between highly integrated peripheral controller components, peripheral add-in boards, and processor/memory systems. The conventional PCI specification revision 2.3 defines the PCI hardware environment including the protocol, electrical, mechanical, and configuration specifications for the PCI devices and expansion boards. This standard requires 3.3-V V_{CCIO}. Stratix and Stratix GX devices are fully compliant with the 3.3-V *PCI Local Bus Specification Revision 2.3* and meet 64-bit/66-MHz operating frequency and timing requirements. The 3.3-V PCI standard does not require input reference voltages or board terminations. Stratix and Stratix GX devices support both input and output levels.

3.3-V PCI-X 1.0 Local Bus - PCI-SIG PCI-X Local Bus Specification Revision 1.0a

The PCI-X 1.0 standard is used for applications that interface to the PCI local bus. The standard enables the design of systems and devices that operate at clock speeds up to 133 MHz, or 1 gigabit per second (Gbps) for a 64-bit bus. The PCI-X 1.0 protocol enhancements enable devices to operate much more efficiently, providing more usable bandwidth at any clock frequency. By using the PCI-X 1.0 standard, devices can be designed to meet PCI-X 1.0 requirements and operate as conventional 33- and 66-MHz PCI devices when installed in those systems. This standard requires 3.3-V $V_{CCIO.}$ Stratix and Stratix GX devices are fully compliant with the 3.3-V *PCI-X Specification Revision 1.0a* and meet the 133-MHz operating frequency and timing requirements. The 3.3-V PCI standard does not require input reference voltages or board terminations. Stratix and Stratix GX devices support both input and output levels.

3.3-V Compact PCI Bus - PCI SIG PCI Local Bus Specification Revision 2.3

The Compact PCI local bus specification is used for applications that interface to the PCI local bus. It follows the *PCI Local Bus Specification Revision 2.3* plus additional requirements in PCI Industrial Computers Manufacturing Group (PICMG) specifications PICMG 2.0 R3.0, CompactPCI specification, and the hot swap requirements in PICMG 2.1 R2.0, CompactPCI Hot Swap Specification. This standard has similar electrical requirements as LVTTL and requires 3.3-V $V_{CCIO.}$ Stratix and Stratix GX devices are compliant with the Compact PCI electrical requirements. The 3.3-V PCI standard does not require input reference voltages or board terminations. Stratix and Stratix GX devices support both input and output levels.

3.3-V 1× AGP - Intel Corporation Accelerated Graphics Port Interface Specification 2.0

The AGP interface is a platform bus specification that enables high-performance graphics by providing a dedicated high-speed port for the movement of large blocks of 3-dimensional texture data between a PC's graphics controller and system memory. The 1× AGP I/O standard is a single-ended standard used for 3.3-V graphics applications. The 1× AGP input standard specifies an input voltage range of $-0.5\ V \leq V_I \leq V_{CCIO} + 0.5\ V$. The 1× AGP standard does not require input reference voltages or board terminations. Stratix and Stratix GX devices support both input and output levels.

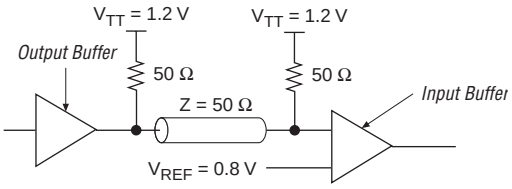
3.3-V 2× AGP - Intel Corporation Accelerated Graphics Port Interface Specification 2.0

The 2× AGP I/O standard is a voltage-referenced, single-ended standard used for 3.3-V graphics applications. The 2× AGP input standard specifies an input voltage range of $-0.5V \leq V_I \leq V_{CCIO} + 0.5V$. The 2× AGP standard does not require board terminations. Stratix and Stratix GX devices support both input and output levels.

GTL - EIA/JEDEC Standard EIA/JESD8-3

The GTL I/O standard is a low-level, high-speed back plane standard used for a wide range of applications from ASICs and processors to interface logic devices. The GTL standard defines the DC interface parameters for digital circuits operating from power supplies of 2.5, 3.3, and 5.0 V. The GTL standard is an open-drain standard, and Stratix and Stratix GX devices support a 2.5- or 3.3-V V_{CCIO} to meet this standard. GTL requires a 0.8-V V_{REF} and open-drain outputs with a 1.2-V V_{TT} (see Figure 16–4). Stratix and Stratix GX devices support both input and output levels.

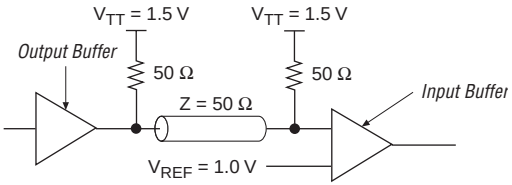
Figure 16–4. GTL Termination



GTL+

The GTL+ I/O standard is used for high-speed back plane drivers and Pentium processor interfaces. The GTL+ standard defines the DC interface parameters for digital circuits operating from power supplies of 2.5, 3.3, and 5.0 V. The GTL+ standard is an open-drain standard, and Stratix and Stratix GX devices support a 2.5- or 3.3-V V_{CCIO} to meet this standard. GTL+ requires a 1.0-V V_{REF} and open-drain outputs with a 1.5-V V_{TT} (see Figure 16–5). Stratix and Stratix GX devices support both input and output levels.

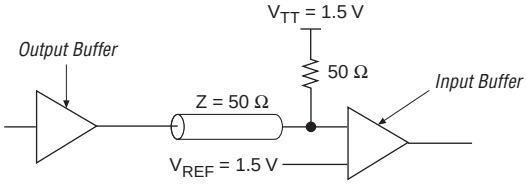
Figure 16–5. GTL+ Termination



CTT - EIA/JEDEC Standard JESD8-4

The CTT I/O standard is used for backplanes and memory bus interfaces. The CTT standard defines the DC interface parameters for digital circuits operating from 2.5- and 3.3-V power supplies. The CTT standard does not require special circuitry to interface with LVTTL or LVCMOS devices when the CTT driver is not terminated. The CTT standard requires a 1.5-V V_{REF} and a 1.5-V V_{TT} (see Figure 16–6). Stratix and Stratix GX devices support both input and output levels.

Figure 16–6. CTT Termination



SSTL-3 Class I & II - EIA/JEDEC Standard JESD8-8

The SSTL-3 I/O standard is a 3.3-V memory bus standard used for applications such as high-speed SDRAM interfaces. This standard defines the input and output specifications for devices that operate in the SSTL-3 logic switching range of 0.0 to 3.3 V. The SSTL-3 standard specifies an input voltage range of $-0.3\ V \le V_I \le V_{CCIO} + 0.3\ V$. SSTL-3 requires a 1.5-V V_{REF} and a 1.5-V V_{TT} to which the series and termination resistors are connected (see Figures 16–7 and 16–8). Stratix and Stratix GX devices support both input and output levels.

Figure 16–7. SSTL-3 Class I Termination

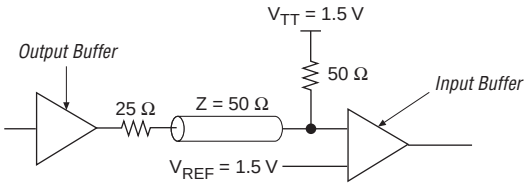
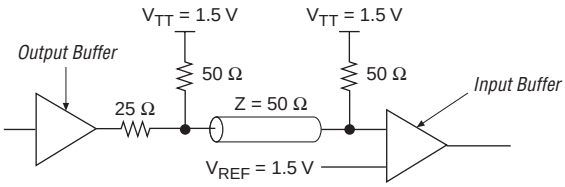


Figure 16–8. SSTL-3 Class II Termination



SSTL-2 Class I & II - EIA/JEDEC Standard JESD8-9A

The SSTL-2 I/O standard is a 2.5-V memory bus standard used for applications such as high-speed DDR SDRAM interfaces. This standard defines the input and output specifications for devices that operate in the SSTL-2 logic switching range of 0.0 to 2.5 V. This standard improves operation in conditions where a bus must be isolated from large stubs. The SSTL-2 standard specifies an input voltage range of $-0.3\ V \le V_I \le V_{CCIO} + 0.3\ V$. SSTL-2 requires a 1.25-V V_{REF} and a 1.25-V V_{TT} to which the series and termination resistors are connected (see Figures 16–9 and 16–10). Stratix and Stratix GX devices support both input and output levels.

Figure 16–9. SSTL-2 Class I Termination

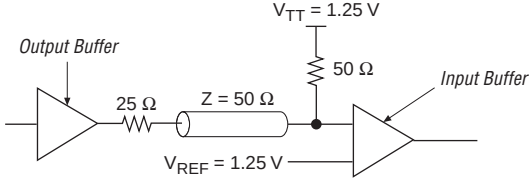
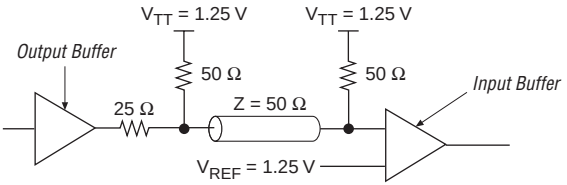


Figure 16–10. SSTL-2 Class II Termination



SSTL-18 Class I & II - EIA/JEDEC Preliminary Standard JC42.3

The SSTL-18 I/O standard is a 1.8-V memory bus standard. This standard is similar to SSTL-2 and defines input and output specifications for devices that are designed to operate in the SSTL-18 logic switching range 0.0 to 1.8 V. SSTL-18 requires a 0.9-V V_{REF} and a 0.9-V V_{TT} to which the series and termination resistors are connected. See Figures 16–11 and 16–12 for details on SSTL-18 Class I and II termination. Stratix and Stratix GX devices support both input and output levels.

Figure 16–11. SSTL-18 Class I Termination

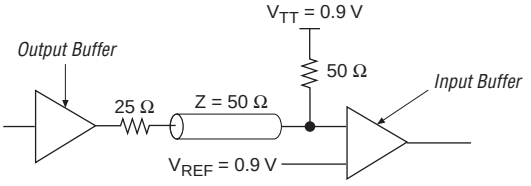
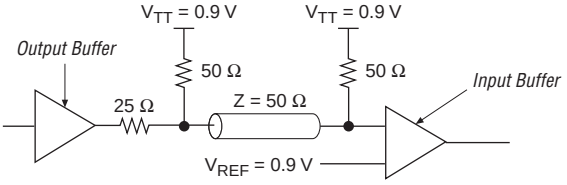


Figure 16–12. SSTL-18 Class II Termination

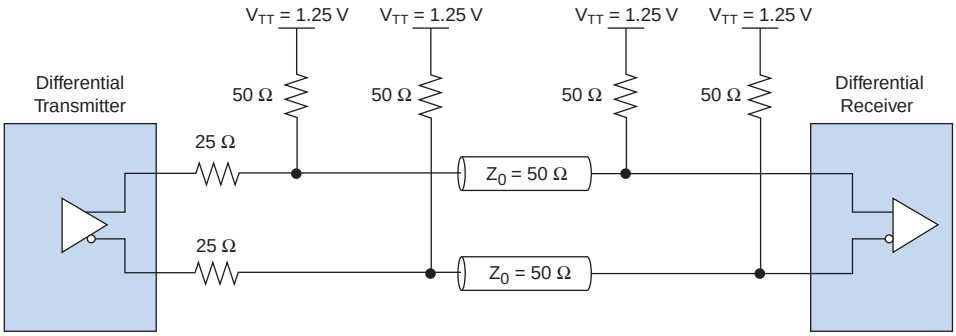


Differential SSTL-2 - EIA/JEDEC Standard JESD8-9A

The differential SSTL-2 I/O standard is a 2.5-V standard used for applications such as high-speed DDR SDRAM clock interfaces. This standard supports differential signals in systems using the SSTL-2

standard and supplements the SSTL-2 standard for differential clocks. The differential SSTL-2 standard specifies an input voltage range of $-0.3\ V \le V_I \le V_{CCIO} + 0.3\ V$. The differential SSTL-2 standard does not require an input reference voltage differential. See Figure 16–13 for details on differential SSTL-2 termination. Stratix and Stratix GX devices support output clock levels for differential SSTL-2 Class II operation. The output clock is implemented using two single-ended output buffers which are programmed to have opposite polarity.

Figure 16–13. Differential SSTL-2 Class II Termination



LVDS - ANSI/TIA/EIA Standard ANSI/TIA/EIA-644

The LVDS I/O standard is a differential high-speed, low-voltage swing, low-power, general-purpose I/O interface standard requiring a 3.3-V V_{CCIO}. This standard is used in applications requiring high-bandwidth data transfer, backplane drivers, and clock distribution. The ANSI/TIA/EIA-644 standard specifies LVDS transmitters and receivers capable of operating at recommended maximum data signaling rates of 655 Mbps. However, devices can operate at slower speeds if needed, and there is a theoretical maximum of 1.923 Gbps. Stratix and Stratix GX devices meet the ANSI/TIA/EIA-644 standard.

Due to the low voltage swing of the LVDS I/O standard, the electromagnetic interference (EMI) effects are much smaller than CMOS, TTL, and PECL. This low EMI makes LVDS ideal for applications with low EMI requirements or noise immunity requirements. The LVDS standard does not require an input reference voltage, however, it does require a 100 Ω termination resistor between the two signals at the input buffer. Stratix and Stratix GX devices include an optional differential LVDS termination resistor within the device using differential on-chip termination. Stratix and Stratix GX devices support both input and output levels.

For more information on the LVDS I/O standard in Stratix devices, see the *High-Speed Differential I/O Interfaces in Stratix Devices* chapter.

LVPECL

The LVPECL I/O standard is a differential interface standard requiring a 3.3-V V_{CCIO}. The standard is used in applications involving video graphics, telecommunications, data communications, and clock distribution. The high-speed, low-voltage swing LVPECL I/O standard uses a positive power supply and is similar to LVDS, however, LVPECL has a larger differential output voltage swing than LVDS. The LVPECL standard does not require an input reference voltage, but it does require a 100-Ω termination resistor between the two signals at the input buffer. See Figures 16–14 and 16–15 for two alternate termination schemes for LVPECL. Stratix and Stratix GX devices support both input and output levels.

Figure 16–14. LVPECL DC Coupled Termination

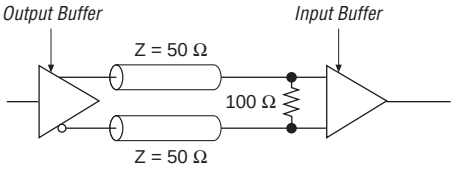
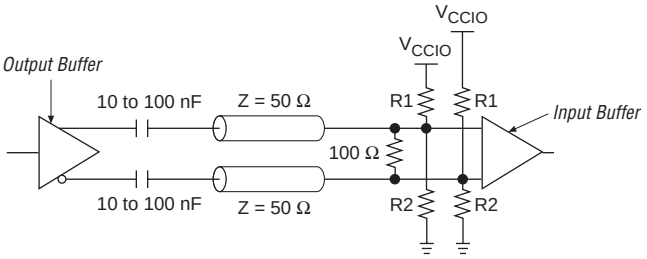


Figure 16–15. LVPECL AC Coupled Termination



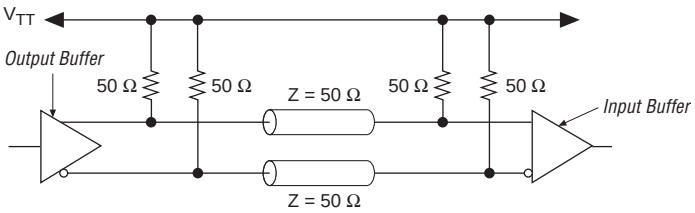
Pseudo Current Mode Logic (PCML)

The PCML I/O standard is a differential high-speed, low-power I/O interface standard used in applications such as networking and telecommunications. The standard requires a 3.3-V V_{CCIO}. The PCML I/O standard consumes less power than the LVPECL I/O standard. The

PCML standard is similar to LVPECL, but PCML has a reduced voltage swing, which allows for a faster switching time and lower power consumption. The PCML standard uses open drain outputs and requires a differential output signal. See Figure 16–16 for details on PCML termination. Stratix and Stratix GX devices support both input and output levels.

Additionally, Stratix GX devices support 1.5-V PCML as described in the *Stratix GX Device Handbook, Volume 1*.

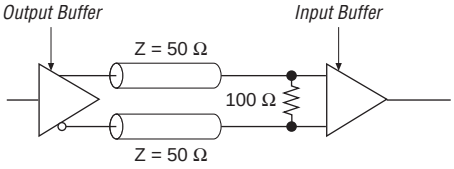
Figure 16–16. PCML Termination



HyperTransport Technology - HyperTransport Consortium

The HyperTransport technology I/O standard is a differential high-speed, high-performance I/O interface standard requiring a 2.5-V V_{CCIO}. This standard is used in applications such as high-performance networking, telecommunications, embedded systems, consumer electronics, and Internet connectivity devices. The HyperTransport technology I/O standard is a point-to-point standard in which each HyperTransport technology bus consists of two point-to-point unidirectional links. Each link is 2 to 32 bits. The HyperTransport technology standard does not require an input reference voltage. However, it does require a 100-Ω termination resistor between the two signals at the input buffer. See Figure 16–17 for details on HyperTransport technology termination. Stratix and Stratix GX devices support both input and output levels.

Figure 16–17. HyperTransport Technology Termination





See the *Stratix Device Family Data Sheet* section in the *Stratix Device Handbook, Volume 1*; the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1*; and the *High-Speed Differential I/O Interfaces in Stratix Devices* chapter for more information on differential I/O standards.

High-Speed Interfaces

In addition to current industry physical I/O standards, Stratix and Stratix GX devices also support a variety of emerging high-speed interfaces. This section provides an overview of these interfaces.

OIF-SPI4.2

This implementation agreement is widely used in the industry for OC-192 and 10-Gbps multi-service system interfaces. SONET and SDH are synchronous transmission systems over which data packets are transferred. POS-PHY Level 4 is a standard interface for switches and routers, and defines the operation between a physical layer (PHY) device and link layer devices (ATM, Internet protocol, and Gigabit Ethernet) for bandwidths of OC-192 ATM, POS, and 10-Gigabit Ethernet applications. Some key POS-PHY Level 4 system features include:

- Large selection of POS-PHY Level 4-based PHYs
- Independent of data protocol
- Wide industry support
- LVDS I/O standard to improve signal integrity
- Inband addressing/control
- Out of band flow control
- Scalable architecture
- Over 622-Mbps operation
- Dynamic interface timing mode

POS-PHY Level 4 operates at a wide range of frequencies.

OIF-SFI4.1

This implementation agreement is widely used in the industry for interfacing physical layer (PHY) to the serializer-deserializer (SERDES) devices in OC-192 and 10 Gbps multi-service systems. The POS-PHY Level 4 interface standard defines the SFI-4 standard. POS-PHY Level 4: SFI-4 is a standardized 16-bit × 622-Mbps line-side interface for 10-Gbps applications. Internet LAN and WAN architectures use telecommunication SONET protocols for data transferring data over the PHY layer. SFI-4 interfaces between OC-192 SERDES and SONET framers.

10 Gigabit Ethernet Sixteen Bit Interface (XSBI) - IEEE Draft Standard P802.3ae/D2.0

10 Gigabit Ethernet XSBI is an interface standard for LANs, metropolitan area networks (MANs), storage area networks (SANs), and WANs. 10 Gigabit Ethernet XSBI provides many features for efficient, effective high-speed networking, including easy migration to higher performance levels without disruption, lower cost of ownership including acquisition and support versus other alternatives, familiar management tools and common skills, ability to support new applications and data protocols, flexibility in network design, and multiple vendor sourcing and interoperability.

Under the ISO Open Systems Interconnection (OSI) model, Ethernet is a Layer 2 protocol. 10 Gigabit Ethernet XSBI uses the IEEE 802.3 Ethernet media access control (MAC) protocol, Ethernet frame format, and the minimum/maximum frame size. An Ethernet PHY corresponding to OSI layer 1 connects the media to the MAC layer that corresponds to OSI layer 2. The PHY is divided into a physical media dependent (PMD) element, such as optical transceivers, and a physical coding sub-layer (PCS), which has coding and a serializer/multiplexor. This standard defines two PHY types, including the LAN PHY and the WAN PHY, which are distinguished by the PCS. The 10 Gigabit Ethernet XSBI standard is a full-duplex technology standard that can increase the speed and distance of Ethernet.

RapidIO Interconnect Specification Revision 1.1

The RapidIO interface is a communications standard used to connect devices on a circuit board and circuit boards on a backplane. RapidIO is a packet-switched interconnect standard designed for embedded systems such as those used in networking and communications. The RapidIO interface standard is a high-performance interconnect interface used for transferring data and control information between microprocessors, DSPs, system memory, communications and network processors, and peripheral devices in a system.

RapidIO replaces existing peripheral bus and processor technologies such as PCI. Some features of RapidIO include multiprocessing support, an open standard, flexible topologies, higher bandwidth, low latency, error management support in hardware, small silicon footprint, widely available process and I/O technologies, and transparency to existing applications and operating system software. The RapidIO standard provides 10-Gbps device bandwidth using 8-bit-wide input and output data ports. RapidIO uses LVDS technology, has the capability to be scaled to multi-GHz frequencies, and features a 10-bit interface.

HyperTransport Technology - HyperTransport Consortium

The HyperTransport technology I/O standard is a differential high-speed, high performance I/O interface standard developed for communications and networking chip-to-chip communications. HyperTransport technology is used in applications such as high-performance networking, telecommunications, embedded systems, consumer electronics, and Internet connectivity devices. The HyperTransport technology I/O standard is a point-to-point (one source connected to exactly one destination) standard that provides a high-performance interconnect between integrated circuits in a system, such as on a motherboard.

Stratix devices support HyperTransport technology at data rates up to 800 Mbps and 32 bits in each direction. HyperTransport technology uses an enhanced differential signaling technology to improve performance. HyperTransport technology supports data widths of 2, 4, 8, 16, or 32 bits in each direction. HyperTransport technology in Stratix and Stratix GX devices operates at multiple clock speeds up to 400 MHz.

UTOPIA Level 4 – ATM Forum Technical Committee Standard AF-PHY-0144.001

The UTOPIA Level 4 frame-based interface standard allows device manufacturers and network developers to develop components that can operate at data rates up to 10 Gbps. This standard increases interface speeds using LVDS I/O and advanced silicon technologies for fast data transfers.

UTOPIA Level 4 provides new control techniques and a 32-, 16-, or 8-bit LVDS bus, a symmetric transmit/receive bus structure for easier application design and testability, nominal data rates of 10 Gbps, in-band control of cell delimiters and flow control to minimize pin count, source-synchronous clocking, and supports variable length packet systems. UTOPIA Level 4 handles sustained data rates for OC-192 and supports ATM cells. UTOPIA Level 4 also supports interconnections across motherboards, daughtercards, and backplane interfaces.

Stratix & Stratix GX I/O Banks

Stratix devices have eight I/O banks in addition to the four enhanced PLL external clock output banks, as shown in Table 16–2 and Figure 16–18. I/O banks 3, 4, 7, and 8 support all single-ended I/O standards. I/O banks 1, 2, 5, and 6 support differential HSTL (on input clocks), LVDS, LVPECL, PCML, and HyperTransport technology, as well as all single-ended I/O standards except HSTL Class II, GTL, SSTL-18 Class II, PCI/PCI-X 1.0, and 1× /2× AGP. The four enhanced PLL external clock output banks (I/O banks 9, 10, 11, and 12) support clock outputs all

single-ended I/O standards in addition to differential SSTL-2 and HSTL (both on the output clock only). Since Stratix devices support both non-voltage-referenced and voltage-referenced I/O standards, there are different guidelines when working with either separately or when working with both.

Table 16–2. I/O Standards Supported in Stratix I/O Banks (Part 1 of 2)

I/O Standard	I/O Bank								Enhanced PLL External Clock Output Banks			
	1	2	3	4	5	6	7	8	9	10	11	12
3.3-V LVTTL/LVCMOS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.5-V LVTTL/LVCMOS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.8-V LVTTL/LVCMOS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.5-V LVCMOS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PCI/PCIX//Compact PCI			✓	✓			✓	✓	✓	✓	✓	✓
AGP 1×			✓	✓			✓	✓	✓	✓	✓	✓
AGP 2×			✓	✓			✓	✓	✓	✓	✓	✓
SSTL-3 Class I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SSTL-3 Class II	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SSTL-2 Class I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SSTL-2 Class II	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SSTL-18 Class I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SSTL-18 Class II			✓	✓			✓	✓	✓	✓	✓	✓
Differential SSTL-2 (output clocks)									✓	✓	✓	✓
HSTL Class I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.5-V HSTL Class I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.8-V HSTL Class I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HSTL Class II			✓	✓			✓	✓	✓	✓	✓	✓
1.5-V HSTL Class II			✓	✓			✓	✓	✓	✓	✓	✓
1.8-V HSTL Class II			✓	✓			✓	✓	✓	✓	✓	✓
Differential HSTL (input clocks)	✓	✓	✓	✓	✓	✓	✓	✓				
Differential HSTL (output clocks)									✓	✓	✓	✓

Table 16–2. I/O Standards Supported in Stratix I/O Banks (Part 2 of 2)

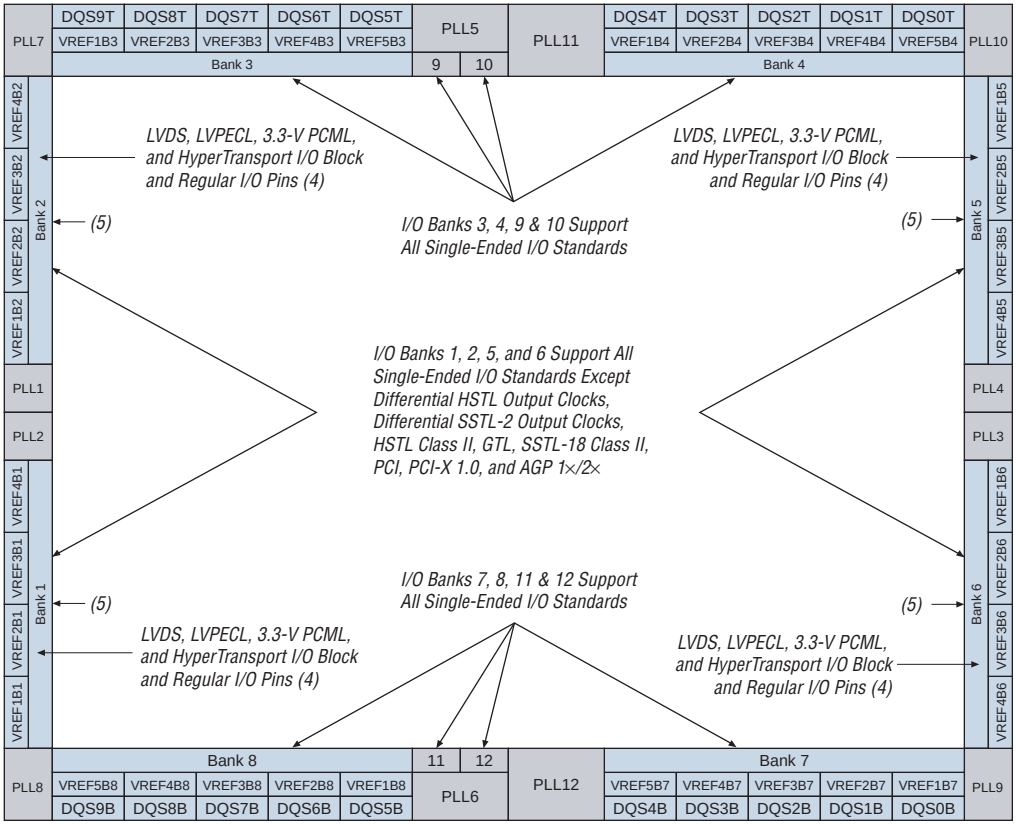
I/O Standard	I/O Bank								Enhanced PLL External Clock Output Banks			
	1	2	3	4	5	6	7	8	9	10	11	12
GTL			✓	✓			✓	✓	✓	✓	✓	✓
GTL+	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CTT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LVDS	✓	✓	(1)	(1)	✓	✓	(1)	(1)	(2)	(2)	(2)	(2)
HyperTransport technology	✓	✓	(1)	(1)	✓	✓	(1)	(1)	(2)	(2)	(2)	(2)
LVPECL	✓	✓	(1)	(1)	✓	✓	(1)	(1)	(2)	(2)	(2)	(2)
PCML	✓	✓	(1)	(1)	✓	✓	(1)	(1)	(2)	(2)	(2)	(2)

Notes to Table 16–2:

(1) This I/O standard is only supported on input clocks in this I/O bank.

(2) This I/O standard is only supported on output clocks in this I/O bank.

Figure 16–18. Stratix I/O Banks *Notes (1), (2), (3)*



Notes to Figure 16–18:

(1) Figure 16–18 is a top view of the silicon die. This corresponds to a top-down view for non-flip-chip packages, but is a reverse view for flip-chip packages.

(2) Figure 16–18 is a graphic representation only. See the pin list and the Quartus II software for exact locations.

(3) Banks 9 through 12 are enhanced PLL external clock output banks.

(4) If the high-speed differential I/O pins are not used for high-speed differential signaling, they can support all of the I/O standards except HSTL Class II, GTL, SSTL-18 Class II, PCI, PCI-X 1.0, and AGP 1×/2×.

(5) For guidelines on placing single-ended I/O pads next to differential I/O pads, see "I/O Pad Placement Guidelines" on page 16–30.

Tables 16–3 and 16–4 list the I/O standards that Stratix GX enhanced and fast PLL pins support. Figure 16–19 shows the I/O standards that each Stratix GX I/O bank supports.

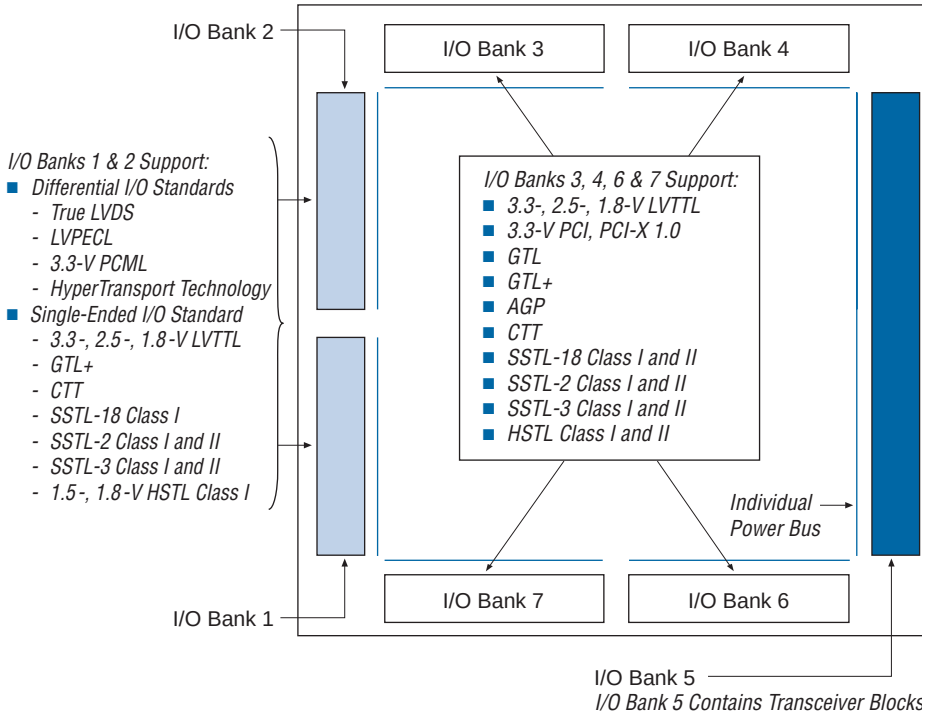
Table 16–3. I/O Standards Supported in Stratix & Stratix GX Enhanced PLL Pins

I/O Standard	Input			Output
	INCLK	FBIN	PLLENABLE	EXTCLK
LVTTL	✓	✓	✓	✓
LVCMOS	✓	✓	✓	✓
2.5 V	✓	✓		✓
1.8 V	✓	✓		✓
1.5 V	✓	✓		✓
3.3-V PCI	✓	✓		✓
3.3-V PCI-X 1.0	✓	✓		✓
LVPECL	✓	✓		✓
3.3-V PCML	✓	✓		✓
LVDS	✓	✓		✓
HyperTransport technology	✓	✓		✓
Differential HSTL	✓			✓
Differential SSTL				✓
3.3-V GTL	✓	✓		✓
3.3-V GTL+	✓	✓		✓
1.5-V HSTL Class I	✓	✓		✓
1.5-V HSTL Class II	✓	✓		✓
SSTL-18 Class I	✓	✓		✓
SSTL-18 Class II	✓	✓		✓
SSTL-2 Class I	✓	✓		✓
SSTL-2 Class II	✓	✓		✓
SSTL-3 Class I	✓	✓		✓
SSTL-3 Class II	✓	✓		✓
AGP (1× and 2×)	✓	✓		✓
CTT	✓	✓		✓

Table 16–4. I/O Standards Supported in Stratix & Stratix GX Fast PLL Pins

I/O Standard	Input	
	INCLK	PLLENABLE
LVTTL	✓	✓
LVCMOS	✓	✓
2.5 V	✓	
1.8 V	✓	
1.5 V	✓	
3.3-V PCI		
3.3-V PCI-X 1.0		
LVPECL	✓	
3.3-V PCML	✓	
LVDS	✓	
HyperTransport technology	✓	
Differential HSTL	✓	
Differential SSTL		
3.3-V GTL		
3.3-V GTL+		
1.5V HSTL Class I	✓	
1.5V HSTL Class II		
SSTL-18 Class I	✓	
SSTL-18 Class II		
SSTL-2 Class I	✓	
SSTL-2 Class II	✓	
SSTL-3 Class I	✓	
SSTL-3 Class II	✓	
AGP (1× and 2×)		
CTT	✓	

Figure 16–19. Stratix GX I/O Banks



There is some flexibility with the number of I/O standards each Stratix I/O bank can simultaneously support. The following sections provide guidelines for mixing non-voltage-referenced and voltage-referenced I/O standards in Stratix devices.

Non-Voltage-Referenced Standards

Each Stratix I/O bank has its own VCCIO pins and supports only one V_{CCIO}, either 1.5, 1.8, 2.5 or 3.3 V. A Stratix I/O bank can simultaneously support any number of input signals with different I/O standard assignments, as shown in Table 16–5.

Table 16–5. Acceptable Input Levels for LVTTL/LVCMOS

Bank V_{CCIO}	Acceptable Input Levels			
	3.3 V	2.5 V	1.8 V	1.5 V
3.3 V	✓	✓		
2.5 V	✓	✓		
1.8 V	✓ (2)	✓ (2)	✓	✓ (1)
1.5 V	✓ (2)	✓ (2)	✓	✓

Notes to Table 16–5:

(1) Because the input signal will not drive to the rail, the input buffer does not completely shut off, and the I/O current will be slightly higher than the default value.

(2) These input values overdrive the input buffer, so the pin leakage current will be slightly higher than the default value.

For output signals, a single I/O bank can only support non-voltage-referenced output signals driving at the same voltage as V_{CCIO}. A Stratix I/O bank can only have one V_{CCIO} value, so it can only drive out that one value for non-voltage referenced signals. For example, an I/O bank with a 2.5-V V_{CCIO} setting can support 2.5-V LVTTL inputs and outputs, HyperTransport technology inputs and outputs, and 3.3-V LVCMOS inputs (not output or bidirectional pins).



If the output buffer overdrives the input buffer, you must turn on the **Allow voltage overdrive for LVTTL/LVCMOS** option in the Quartus II software. To see this option, click the **Device & Pin Options** button in the **Device** page of the **Settings** dialog box (Assignments menu). Then click the **Pin Placement** tab in the **Device & Pin Options** dialog box.

Voltage-Referenced Standards

To accommodate voltage-referenced I/O standards, each Stratix I/O bank supports multiple VREF pins feeding a common VREF bus. The number of available VREF pins increases as device density increases. If these pins are not used as VREF pins, they can not be used as generic I/O pins.

An I/O bank featuring single-ended or differential standards can support voltage-referenced standards as long as all voltage-referenced standards use the same V_{REF} setting. For example, although one I/O bank can implement both SSTL-3 and SSTL-2 I/O standards, I/O pins using these standards must be in different banks since they require different V_{REF} values

For voltage-referenced inputs, the receiver compares the input voltage to the voltage reference and does not take into account the V_{CCIO} setting. Therefore, the V_{CCIO} setting is irrelevant for voltage referenced inputs.

Voltage-referenced bidirectional and output signals must be the same as the I/O bank's V_{CCIO} voltage. For example, although you can place an SSTL-2 input pin in any I/O bank with a 1.25-V V_{REF} level, you can only place SSTL-2 output pins in an I/O bank with a 2.5-V V_{CCIO}.

Mixing Voltage Referenced & Non-Voltage Referenced Standards

Non-voltage referenced and voltage referenced pins can safely be mixed in a bank by applying each of the rule-sets individually. For example, on I/O bank can support SSTL-3 inputs and 1.8-V LVCMOS inputs and outputs with a 1.8-V V_{CCIO} and a 1.5-V V_{REF}. Similarly, an I/O bank can support 1.5-V LVCMOS, 3.3-V LVTTL (inputs, but not outputs), and HSTL I/O standards with a 1.5-V V_{CCIO} and 0.75-V V_{REF}.

For the voltage-referenced examples, see the "I/O Pad Placement Guidelines" section. For details on how the Quartus II software supports I/O standards, see the "Quartus II Software Support"section.

Drive Strength

Each I/O standard supported by Stratix and Stratix GX devices drives out a minimum drive strength. When an I/O is configured as LVTTL or LVCMOS I/O standards, you can specify the current drive strength, as summarized in Table 16–7.

Standard Current Drive Strength

Each I/O standard supported by Stratix and Stratix GX devices drives out a minimum drive strength. Table 16–6 summarizes the minimum drive strength of each I/O standard.

Table 16–6. Minimum Current Drive Strength of Each I/O Standard

I/O Standard	Current Strength, I_{OL}/I_{OH} (mA)
GTL	40 *(1)*
GTL+	34 *(1)*
SSTL-3 Class I	8
SSTL-3 Class II	16
SSTL-2 Class I	8.1
SSTL-2 Class II	16.4
SSTL-18 Class I	6.7
SSTL-18 Class II	13.4
1.5-V HSTL Class I	8
1.5-V HSTL Class II	16
CTT	8
AGP 1×	I_{OL} = 1.5, I_{OH} = –0.5

Note to Table 16–6:
(1) Because this I/O standard uses an open drain buffer, this value refers to I_{OL}.

When the SSTL-2 Class I and II I/O standards are implemented on top or bottom I/O pins, the drive strength is designed to be higher than the drive strength of the buffer when implemented on side I/O pins. This allows the top or bottom I/O pins to support 200-MHz operation with the standard 35-pF load. At the same time, the current consumption when using top or bottom I/O pins is higher than the side I/O pins. The high current strength may not be necessary for certain applications where the value of the load is less than the standard test load (e.g., DDR interface). The Quartus II software allows you to reduce the drive strength when the I/O pins are used for the SSTL-2 Class I or Class II I/O standard and being implemented on the top or bottom I/O through the Current Strength setting. Select the minimum strength for lower drive strength.

Programmable Current Drive Strength

The Stratix and Stratix GX device I/O pins support various output current drive settings as shown in Table 16–7. These programmable drive strength settings help decrease the effects of simultaneously switching outputs (SSO) in conjunction with reducing system noise. The supported settings ensure that the device driver meets the I_{OH} and I_{OL} specifications for the corresponding I/O standard.

Table 16–7. Programmable Drive Strength

I/O Standard	I_{OH} / I_{OL} Current Strength Setting (mA)
3.3-V LVTTL	24 *(1)*, 16, 12, 8, 4
3.3-V LVCMOS	24 *(2)*, 12 *(1)*, 8, 4, 2
2.5-V LVTTL/LVCMOS	16 *(1)*, 12, 8, 2
1.8-V LVTTL/LVCMOS	12 *(1)*, 8, 2
1.5-V LVCMOS	8 *(1)*, 4, 2

Notes to Table 16–7:
(1) This is the Quartus II software default current setting.
(2) I/O banks 1, 2, 5, and 6 do not support this setting.

These drive-strength settings are programmable on a per-pin basis (for output and bidirectional pins only) using the Quartus II software. To modify the current strength of a particular pin, see "Programmable Drive Strength Settings" on page 16–40.

Hot Socketing

Stratix devices support hot socketing without any external components. In a hot socketing situation, a device's output buffers are turned off during system power-up or power-down. Stratix and Stratix GX devices support any power-up or power-down sequence (V_{CCIO} and V_{CCINT}) to simplify designs. For mixed-voltage environments, you can drive signals into the device before or during power-up or power-down without damaging the device. Stratix and Stratix GX devices do not drive out until the device is configured and has attained proper operating conditions.

Even though you can power up or down the V_{CCIO} and V_{CCINT} power supplies in any sequence you should not power down any I/O bank(s) that contains the configuration pins while leaving other I/O banks powered on. For power up and power down, all supplies (V_{CCINT} and all V_{CCIO} power planes) must be powered up and down within 100 ms of one another. This prevents I/O pins from driving out.

You can power up or power down the V_{CCIO} and V_{CCINT} pins in any sequence. The power supply ramp rates can range from 100 ns to 100 ms. During hot socketing, the I/O pin capacitance is less than 15 pF and the clock pin capacitance is less than 20 pF.

DC Hot Socketing Specification

The hot socketing DC specification is $| I_{IOPIN} | < 300 \mu A$.

AC Hot Socketing Specification

The hot socketing AC specification is $| I_{IOPIN} | < 8$ mA for 10 ns or less.

This specification takes into account the pin capacitance, but not board trace and external loading capacitance. Additional capacitance for trace, connector, and loading must be considered separately.

I_{IOPIN} is the current at any user I/O pin on the device. The DC specification applies when all VCC supplies to the device are stable in the powered-up or powered-down conditions. For the AC specification, the peak current duration because of power-up transients is 10 ns or less. For more information, refer to the *Hot-Socketing & Power-Sequencing Feature & Testing for Altera Devices* white paper.

I/O Termination

Although single-ended, non-voltage-referenced I/O standards do not require termination, Altera recommends using external termination to improve signal integrity where required.

The following I/O standards do not require termination:

- LVTTL
- LVCMOS
- 2.5 V
- 1.8 V
- 1.5 V
- 3.3-V PCI/Compact PCI
- 3.3-V PCI-X 1.0
- 3.3-V AGP 1×

Voltage-Referenced I/O Standards

Voltage-referenced I/O standards require both an input reference voltage, $V_{REF,}$ and a termination voltage, V_{TT}. Off-chip termination on the board should be used for series and parallel termination.

For more information on termination for voltage-referenced I/O standards, see the *Selectable I/O Standards in Stratix & Stratix GX Devices* chapter in the *Stratix Device Handbook, Volume 2*; or the *Stratix GX Device Handbook, Volume 2*.

Differential I/O Standards

Differential I/O standards typically require a termination resistor between the two signals at the receiver. The termination resistor must match the differential load impedance of the bus. Stratix and Stratix GX devices provide an optional differential termination on-chip resistor when using LVDS.

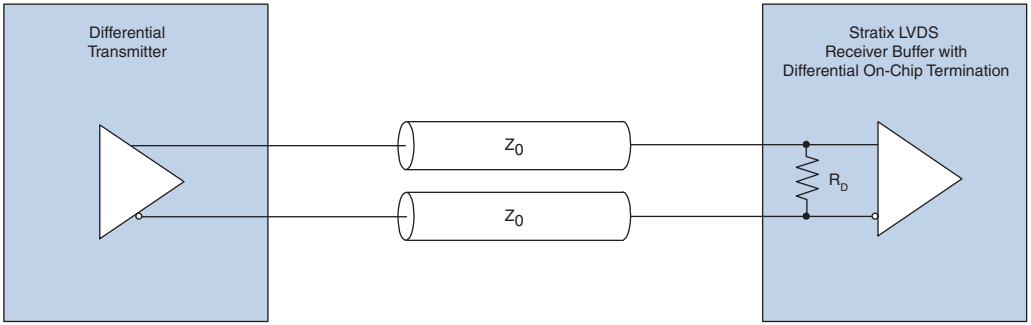
See the *High-Speed Differential I/O Interfaces in Stratix Devices* chapter for more information on differential I/O standards and their interfaces.

For differential I/O standards, I/O banks support differential termination when V_{CCIO} equals 3.3 V.

Differential Termination (R_D)

Stratix devices support differential on-chip termination for source-synchronous LVDS signaling. The differential termination resistors are adjacent to the differential input buffers on the device. This placement eliminates stub effects, improving the signal integrity of the serial link. Using differential on-chip termination resistors also saves board space. Figure 16–20 shows the differential termination connections for Stratix and Stratix GX devices.

Figure 16–20. Differential Termination



Differential termination for Stratix devices is supported for the left and right I/O banks. Differential termination for Stratix GX devices is supported for the left, source-synchronous I/O bank. Some of the clock input pins are in the top and bottom I/O banks, which do not support differential termination. Clock pins CLK[1,3,8,10] support differential on-chip termination. Clock pins CLK[0,2,9,11], CLK[4-7], and CLK[12-15] do not support differential on-chip termination.

Transceiver Termination

Stratix GX devices feature built-in on-chip termination within the transceiver at both the transmit and receive buffers. This termination improves signal integrity and provides support for the 1.5-V PCML I/O standard.

I/O Pad Placement Guidelines

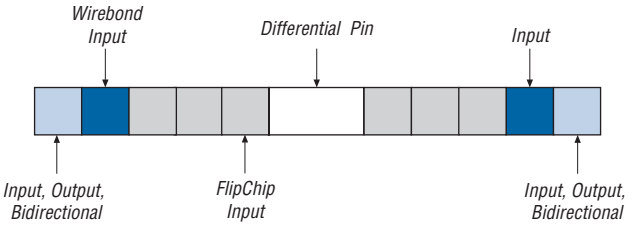
This section provides pad placement guidelines for the programmable I/O standards supported by Stratix and Stratix GX devices and includes essential information for designing systems using the devices' selectable I/O capabilities. These guidelines will reduce noise problems so that FPGA devices can maintain an acceptable noise level on the line from the V_{CCIO} supply. Since Altera FPGAs require that a separate V_{CCIO} power each bank, these noise issues do not have any effect when crossing bank boundaries and these guidelines do not apply. Although pad placement rules need not be considered between I/O banks, some rules must be considered if you are using a `VREF` signal in a `PLLOUT` bank. Note that the signals in the `PLLOUT` banks share the V_{REF} supply with neighboring I/O banks and, therefore, must adhere to the V_{REF} rules discussed in "VREF Pad Placement Guidelines".

Differential Pad Placement Guidelines

To avoid cross coupling and maintain an acceptable noise level on the V_{CCIO} supply, there are restrictions on the placement of single-ended I/O pads in relation to differential pads. Use the following guidelines for placing single-ended pads with respect to differential pads in Stratix devices. These guidelines apply for LVDS, HyperTransport technology, LVPECL, and PCML I/O standards. The differential pad placement guidelines do not apply for differential HSTL and differential SSTL output clocks since each differential output clock is essentially implemented using two single-ended output buffers. These rules do not apply to differential HSTL input clocks either even though the dedicated input buffers are used. However, both differential HSTL and differential SSTL output standards must adhere to the single-ended (`VREF`) pad placement restrictions discussed in "VREF Pad Placement Guidelines".

- For flip-chip packages, there are no restrictions for placement of single-ended input signals with respect to differential signals (see Figure 16–21). For wire-bond packages, single ended input pads may only be placed four or more pads away from a differential pad.
- Single-ended outputs and bidirectional pads may only be placed five or more pads away from a differential pad (see Figure 16–21), regardless of package type.

Figure 16–21. Legal Pin Placement *Note (1)*



Note to Figure 16–21:
(1) Input pads on a flip-chip packages have no restrictions.

VREF Pad Placement Guidelines

Restrictions on the placement of single-ended voltage-referenced I/O pads with respect to VREF pads help maintain an acceptable noise level on the V_{CCIO} supply and to prevent output switching noise from shifting the VREF rail. The following guidelines are for placing single-ended pads in Stratix devices.

Input Pins

Each VREF pad supports a maximum of 40 input pads with up to 20 on each side of the VREF pad.

Output Pins

When a voltage referenced input or bidirectional pad does not exist in a bank, there is no limit to the number of output pads that can be implemented in that bank. When a voltage referenced input exists, each VREF pad supports 20 outputs for thermally enhanced FineLine BGA® and thermally enhanced BGA cavity up packages or 15 outputs for Non-thermally enhanced cavity up and non-thermally enhanced FineLine BGA packages.

Bidirectional Pins

Bidirectional pads must satisfy input and output guidelines simultaneously. If the bidirectional pads are all controlled by the same OE and there are no other outputs or voltage referenced inputs in the bank, then there is no case where there is a voltage referenced input active at the same time as an output. Therefore, the output limitation does not apply. However, since the bidirectional pads are linked to the same OE, the bidirectional pads act as inputs at the same time. Therefore, the input limitation of 40 input pads (20 on each side of the VREF pad) applies.

If any of the bidirectional pads are controlled by different output enables (OE) and there are no other outputs or voltage referenced inputs in the bank, then there may be a case where one group of bidirectional pads is acting as inputs while another group is acting as outputs. In such cases, apply the formulas shown in Table 16–8.

Table 16–8. Input-Only Bidirectional Pin Limitation Formulas

Package Type	Formula
Thermally enhanced FineLine BGA and thermally enhanced BGA cavity up	<*Total number of bidirectional pads*> – <*Total number of pads from the smallest group of pads controlled by an OE*> ≤20 (per VREF pad)
Non-thermally enhanced cavity up and non-thermally enhanced FineLine BGA	<*Total number of bidirectional pads*> – <*Total number of pads from the smallest group of pads controlled by an OE*> ≤15 (per VREF pad).

Consider a thermally enhanced FineLine BGA package with eight bidirectional pads controlled by OE1, eight bidirectional pads controlled by OE2, and six bidirectional pads controlled by OE3. While this totals 22 bidirectional pads, it is safely allowable because there would be a maximum of 16 outputs per VREF pad possible assuming the worst case where OE1 and OE2 are active and OE3 is inactive. This is particularly relevant in DDR SDRAM applications.

When at least one additional voltage referenced input and no other outputs exist in the same VREF bank, then the bidirectional pad limitation must simultaneously adhere to the input and output limitations. See the following equation.

<*Total number of bidirectional pads*> + <*Total number of input pads*> ≤40 (20 on each side of the VREF pad)

The previous equation accounts for the input limitations, but you must apply the appropriate equation from Table 16–9 to determine the output limitations.

Table 16–9. Bidirectional pad Limitation Formulas (Where VREF Inputs Exist)

Package Type	Formula
Thermally enhanced FineLine BGA and thermally enhanced BGA cavity up	<*Total number of bidirectional pads*> ≤20 (per VREF pad)
Non-thermally enhanced cavity up and non-thermally enhanced FineLine BGA	<*Total number of bidirectional pads*> ≤15 (per VREF pad)

When at least one additional output exists but no voltage referenced inputs exist, apply the appropriate formula from Table 16–10.

Table 16–10. Bidirectional Pad Limitation Formulas (Where VREF Outputs Exist)

Package Type	Formula
Thermally enhanced FineLine BGA and thermally enhanced BGA cavity up	<*Total number of bidirectional pads*> + <*Total number of additional output pads*> – <*Total number of pads from the smallest group of pads controlled by an OE*> ≤20 (per VREF pad)
Non-thermally enhanced cavity up and non-thermally enhanced FineLine BGA	<*Total number of bidirectional pads*> + <*Total number of additional output pads*> – <*Total number of pads from the smallest group of pads controlled by an OE*> ≤15 (per VREF pad)

When additional voltage referenced inputs and other outputs exist in the same VREF bank, then the bidirectional pad limitation must again simultaneously adhere to the input and output limitations. See the following equation.

<*Total number of bidirectional pads*> + <*Total number of input pads*> ≤40 (20 on each side of the VREF pad)

The previous equation accounts for the input limitations, but you must apply the appropriate equation from Table 16–9 to determine the output limitations.

Table 16–11. Bidirectional Pad Limitation Formulas (Multiple VREF Inputs & Outputs)

Package Type	Formula
Thermally enhanced FineLine BGA and thermally enhanced BGA cavity up	<*Total number of bidirectional pads*> + <*Total number of additional output pads*> ≤20 (per VREF pad)
non-thermally enhanced cavity up and non-thermally enhanced FineLine BGA	<*Total number of bidirectional pads*> + <*Total number of additional output pads*> ≤15 (per VREF pad)

In addition to the pad placement guidelines, use the following guidelines when working with V_{REF} standards:

- Each bank can only have a single V_{CCIO} voltage level and a single V_{REF} voltage level at a given time. Pins of different I/O standards can share the bank if they have compatible V_{CCIO} values (see Table 16–12 for more details).
- In all cases listed above, the Quartus II software generates an error message for illegally placed pads.

Output Enable Group Logic Option in Quartus II

The Quartus II software can check a design to make sure that the pad placement does not violate the rules mentioned above. When the software checks the design, if the design contains more bidirectional pins than what is allowed, the Quartus II software returns a fitting error. When all the bidirectional pins are either input or output but not both (for example, in a DDR memory interface), you can use the **Output Enable Group Logic** option. Turning on this option directs the Quartus II Fitter to view the specified nodes as an output enable group. This way, the Fitter does not violate the requirements for the maximum number of pins driving out of a V_{REF} bank when a voltaged-referenced input pin or bidirectional pin is present.

In a design that implements DDR memory interface with dq, dqs and dm pins utilized, there are two ways to enable the above logic options. You can enable the logic options through the Assignment Editor or by adding the following assignments to your project's ESF file:

```
OPTIONS_FOR_INDIVIDUAL_NODES_ONLY
{
    dq  : OUTPUT_ENABLE_GROUP 1;
    dqs : OUTPUT_ENABLE_GROUP 1;
```

```
    dm  : OUTPUT_ENABLE_GROUP 1;
}
```

As a result, the Quartus II Fitter does not count the bidirectional pin potential outputs, and the number of V_{REF} bank outputs remains in the legal range.

Toggle Rate Logic Option in Quartus II

You should specify the pin's output toggling rate in order to perform a stricter pad placement check in the Quartus II software. Specify the frequency at which a pin toggles in the Quartus II Assignment Editor. This option is useful for adjusting the pin toggle rate in order to place them closer to differential pins. The option directs the Quartus II Fitter toggle-rate checking while allowing you to place a single-ended pin closer to a differential pin.

DC Guidelines

Variables affecting the DC current draw include package type and desired termination methods. This section provides information on each of these variables and also shows how to calculate the DC current for pin placement.



The Quartus II software automatically takes these variables into account during compilation.

For any 10 consecutive output pads in an I/O bank, Altera recommends a maximum current of 200 mA for thermally enhanced FineLine BGA and thermally enhanced BGA cavity up packages and 164 mA for non-thermally enhanced cavity up and non-thermally enhanced FineLine BGA packages. The following equation shows the current density limitation equation for thermally enhanced FineLine BGA and thermally enhanced BGA cavity up packages:

$$\sum_{pin}^{pin+9} I_{pin} < 200\ mA$$

The following equation shows the current density limitation equation for non-thermally enhanced cavity up and non-thermally enhanced FineLine BGA packages:

$$\sum_{pin}^{pin+9} I_{pin} < 164\ mA$$

Table 16–12 shows the DC current specification per pin for each I/O standard. I/O standards not shown in the table do not exceed these current limitations.

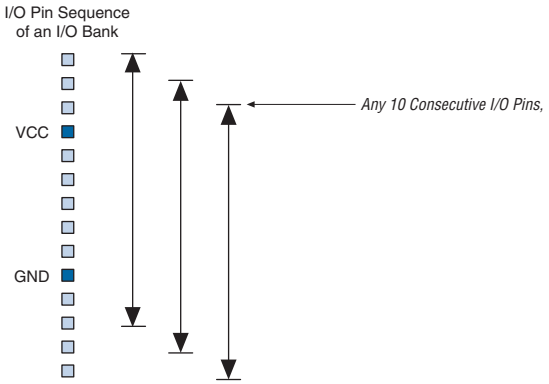
Table 16–12. I/O Standard DC Specification *Note (1)*

Pin I/O Standard	I_{PIN} (mA)		
	3.3-V V_{CCIO}	2.5-V V_{CCIO}	1.5-V V_{CCIO}
GTL	40	40	-
GTL+	34	34	-
SSTL-3 Class I	8	-	-
SSTL-3 Class II	16	-	-
CTT	8	-	-
SSTL-2 Class I	-	8.1	-
SSTL-2 Class II	-	16.4	-
HSTL Class I	-	-	8
HSTL Class II	-	-	16

Note to Table 16–12:
(1) The current rating on a V_{REF} pin is less than 10µA.

For more information on Altera device packaging, see the *Package Information for Stratix Devices* chapter in the *Stratix Device Handbook, Volume 2*.

Figure 16–22. Current Draw Limitation Guidelines



Any 10 consecutive I/O pads cannot exceed 200 mA in thermally enhanced FineLine BGA and thermally enhanced BGA cavity up packages or 164 mA in non-thermally enhanced cavity up and non-thermally enhanced FineLine BGA packages.

For example, consider a case where a group of 10 consecutive pads are configured as follows for a thermally enhanced FineLine BGA and thermally enhanced BGA cavity up package:

- Number of SSTL-3 Class I output pads = 3
- Number of GTL+ output pads = 4
- The rest of the surrounding I/O pads in the consecutive group of 10 are unused

In this case, the total current draw for these 10 consecutive I/O pads would be:

(# of SSTL-3 Class I pads × 8 mA) +
(# of GTL+ output pads × 34 mA) = (3 × 8 mA) + (4 × 34 mA) = 160 mA

In the above example, the total current draw for all 10 consecutive I/O pads is less than 200 mA.

Power Source of Various I/O Standards

For Stratix and Stratix GX devices, the I/O standards are powered by different power sources. To determine which source powers the input buffers, see Table 16–13. All output buffers are powered by V_{CCIO}.

Table 16–13. The Relationships Between Various I/O Standards and the Power Sources

I/O Standard	Power Source
2.5V/3.3V LVTTL	V_{CCIO}
PCI/PCI-X 1.0	V_{CCIO}
AGP	V_{CCIO}
1.5V/1.8V	V_{CCIO}
GTL	V_{CCINT}
GTL+	V_{CCINT}
SSTL	V_{CCINT}
HSTL	V_{CCINT}
CTT	V_{CCINT}
LVDS	V_{CCINT}
LVPECL	V_{CCINT}
PCML	V_{CCINT}
HyperTransport	V_{CCINT}

Quartus II Software Support

You specify which programmable I/O standards to use for Stratix and Stratix GX devices with the Quartus II software. This section describes Quartus II implementation, placement, and assignment guidelines, including

- Compiler Settings
- Device & Pin Options
- Assign Pins
- Programmable Drive Strength Settings
- I/O Banks in the Floorplan View
- Auto Placement & Verification

Compiler Settings

You make Compiler settings in the **Compiler Settings** dialog box (Processing menu). Click the **Chips & Devices** tab to specify the device family, specific device, package, pin count, and speed grade to use for your design.

Device & Pin Options

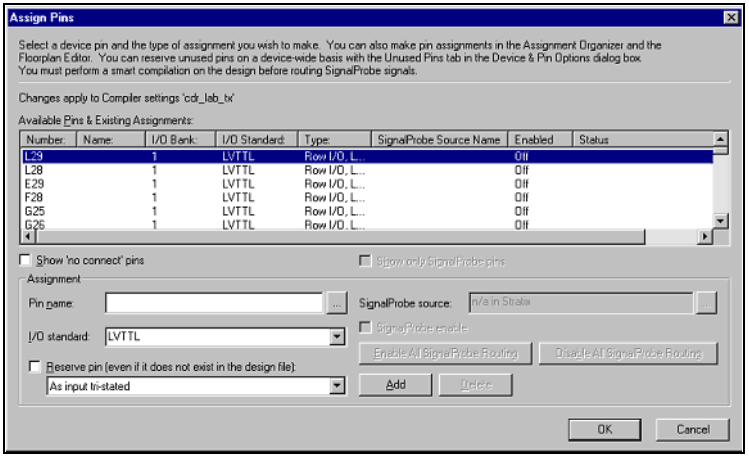
Click **Device & Pin Options** in the **Compiler Settings** dialog box to access the I/O pin settings. For example, in the **Voltage** tab you can select a default I/O standard for all pins for the targeted device. I/O pins that do not have a specific I/O standard assignment default this standard. Click **OK** when you are done setting I/O pin options to return to the **Compiler Settings** dialog box.

Assign Pins

Click **Assign Pins** in the **Compiler Settings** dialog box to view the device's pin settings and pin assignments (see Figure 16–23). You can view the pin settings under **Available Pins & Existing Assignments**. The listing does not include V_{REF} pins because they are dedicated pins. The information for each pin includes:

- Number
- Name
- I/O Bank
- I/O Standard
- Type (e.g., row or column I/O and differential or control)
- SignalProbe Source Name
- Enabled (that is, whether SignalProbe routing is enabled or disabled
- Status

Figure 16–23. Assign Pins



When you assign an I/O standard that requires a reference voltage to an I/O pin, the Quartus II software automatically assigns V_{REF} pins. See the Quartus II Help for instructions on how to use an I/O standard for a pin.

Programmable Drive Strength Settings

To make programmable drive strength settings, perform the following steps:

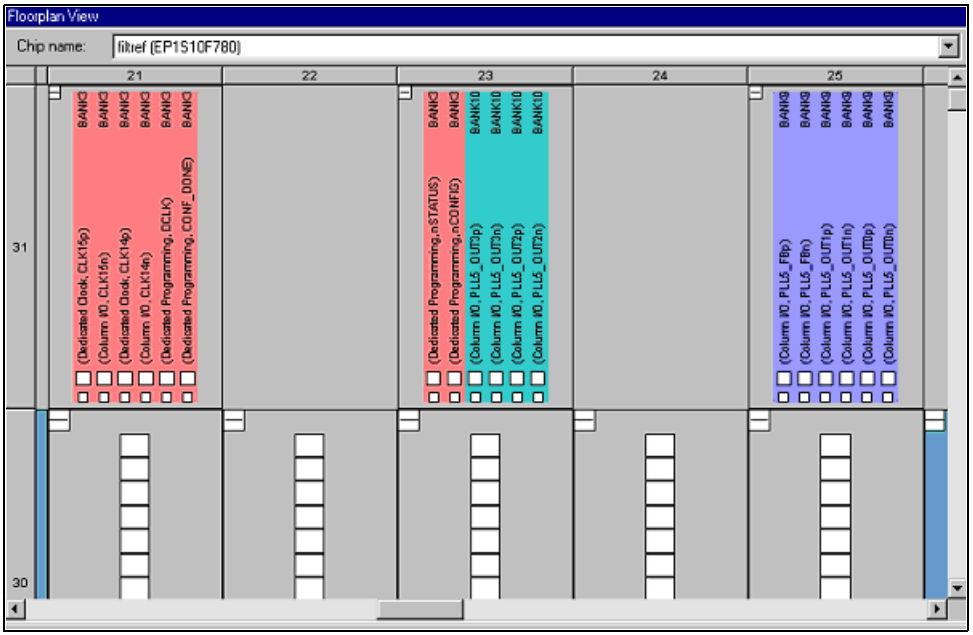
1. In the Tools menu, choose **Assignment Organizer**.
2. Choose the **Edit specific entity & node settings for:** setting, then select the output or bidirectional pin to specify the current strength for.
3. In the **Assignment Categories** dialog box, select **Options for Individual Nodes Only**.
4. Select **Click here to add a new assignment**.
5. In the **Assignment** dialog box, set the **Name** field to **Current Strength** and set the **Setting** field to the desired, allowable value.
6. Click **Add**.
7. Click **Apply**, then **OK**.

I/O Banks in the Floorplan View

You can view the arrangement of the device I/O banks in the **Floorplan View** (View menu) as shown in Figure 16–24. You can assign multiple I/O standards to the I/O pins in any given I/O bank as long as the V_{CCIO} of the standards is the same. Pins that belong to the same I/O bank must use the same V_{CCIO} signal.

Each device I/O pin belongs to a specific, numbered I/O bank. The Quartus II software color codes the I/O bank to which each I/O pin and V_{CCIO} pin belong. Turn on the **Show I/O Banks** option to display the I/O bank color and the bank numbers for each pin.

Figure 16–24. Floorplan View Window



Auto Placement & Verification of Selectable I/O Standards

The Quartus II software automatically verifies the placement for all I/O and V_{REF} pins and performs the following actions.

- Automatically places I/O pins of different V_{REF} standards without pin assignments in separate I/O banks and enables the V_{REF} pins of these I/O banks.
- Verifies that voltage-referenced I/O pins requiring different V_{REF} levels are not placed in the same bank.
- Reports an error message if the current limit is exceeded for a Stratix or Stratix GX power bank, as determined by the equation documented in "DC Guidelines" on page 16–35.
- Reserves the unused high-speed differential I/O channels and regular user I/O pins in the high-speed differential I/O banks when any of the high-speed differential I/O channels are being used.
- Automatically assigns V_{REF} pins and I/O pins such that the current requirements are met and I/O standards are placed properly.

Conclusion

Stratix and Stratix GX devices provide the I/O capabilities to allow you to work with current and emerging I/O standards and requirements. Today's complex designs demand increased flexibility to work with the wide variety of available I/O standards and to simplify board design. With Stratix and Stratix GX device features, such as hot socketing and differential on-chip termination, you can reduce board design interface costs and increase your development flexibility.

More Information

For more information, see the following sources:

- The *Stratix Device Family Data Sheet* section in the *Stratix Device Handbook, Volume 1*
- The *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1*
- The *High-Speed Differential I/O Interfaces in Stratix Devices* chapter
- *AN 224: High-Speed Board Layout Guidelines*

References

For more information, see the following references:

- Stub Series Terminated Logic for 2.5-V (SSTL-2), JESD8-9B, Electronic Industries Association, December 2000.
- High-Speed Transceiver Logic (HSTL) – A 1.5-V Output Buffer Supply Voltage Based Interface Standard for Digital Integrated Circuits, EIA/JESD8-6, Electronic Industries Association, August 1995.
- 1.5-V +/- 0.1 V (Normal Range) and 0.9 V – 1.6 V (Wide Range) Power Supply Voltage and Interface Standard for Non-terminated Digital Integrated Circuits, JESD8-11, Electronic Industries Association, October 2000.
- 1.8-V +/- 0.15 V (Normal Range) and 1.2 V – 1.95 V (Wide Range) Power Supply Voltage and Interface Standard for Non-terminated Digital Integrated Circuits, JESD8-7, Electronic Industries Association, February 1997.
- Center-Tap-Terminated (CTT) Low-Level, High-Speed Interface Standard for Digital Integrated Circuits, JESD8-9A, Electronic Industries Association, November 1993.
- 2.5-V +/- 0.2V (Normal Range) and 1.8-V to 2.7V (Wide Range) Power Supply Voltage and Interface Standard for Non-terminated Digital Integrated Circuits, JESD8-5, Electronic Industries Association, October 1995.
- Interface Standard for Nominal 3V/ 3.3-V Supply Digital Integrated Circuits, JESD8-B, Electronic Industries Association, September 1999.
- Gunning Transceiver Logic (GTL) Low-Level, High-Speed Interface Standard for Digital Integrated Circuits, JESD8-3, Electronic Industries Association, November 1993.

- Accelerated Graphics Port Interface Specification 2.0, Intel Corporation.
- Stub Series Terminated Logic for 1.8-V (SSTL-18), Preliminary JC42.3, Electronic Industries Association.
- PCI Local Bus Specification, Revision 2.2, PCI Special Interest Group, December 1998.
- PCI-X Local Bus Specification, Revision 1.0a, PCI Special Interest Group.
- UTOPIA Level 4, AF-PHY-0144.001, ATM Technical Committee.
- POS-PHY Level 4: SPI-4, OIF-SPI4-02.0, Optical Internetworking Forum.
- POS-PHY Level 4: SFI-4, OIF-SFI4-01.0, Optical Internetworking Forum.
- Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits, ANSI/TIA/EIA-644, American National Standards Institute/Telecommunications Industry/Electronic Industries Association, October 1995.

17. High-Speed Source-Synchronous Differential I/O Interfaces in Stratix GX Devices

SGX52013-1.2

Introduction

Expansion in the telecommunications market and growth in Internet use requires systems to move more data faster than ever. To meet this demand, system designers rely on solutions such as differential signaling and emerging high-speed interface standards including RapidIO, POS-PHY 4, SFI-4, or XSBI.

These new protocols support differential data rates up to 1 gigabit per second (Gbps) and higher. At these high data rates, it becomes more challenging to manage the skew between the clock and data signals. One solution to this challenge is to use clock data recovery (CDR) to eliminate skew between data channels and clock signals. Another potential solution, dynamic phase alignment (DPA), is beginning to be incorporated by some of these protocols.

The Stratix® GX family of devices are the first FPGA devices to have an embedded dynamic phase aligner. This application note explains how to take advantage of the DPA feature in device high-speed I/O circuitry to increase system efficiencies and bandwidth. It will describe the skew issue in high-speed systems and provide a brief description of the source-synchronous circuitry in Stratix GX devices. The document will then describe an overview of the DPA block, I/O support with DPA, fast PLL support with DPA, a full description of DPA operation, and finally a comparison between CDR and source-synchronous interfaces.

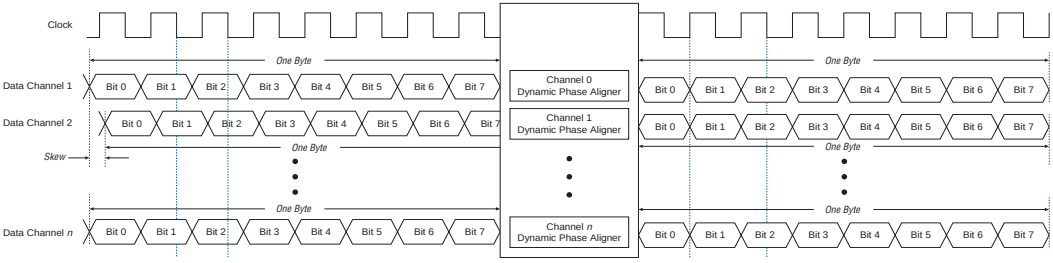
The source-synchronous high-speed interface in Stratix GX devices is a dedicated circuit embedded into the programmable logic device (PLD) allowing for high-speed communications. *AN 202: Using High-Speed Differential I/O Standards in Stratix Devices* provides information on Stratix GX device high-speed I/O standard features and functions.

Skew & Dynamic Phase Alignment

A typical problem designers face with high-speed source-synchronous systems is when clock or data signal transitions occur at different times with respect to each other (see Figure 17–1). When this happens, the receiver does not sample the data at the correct time, causing system errors. This problem is due to the inherent skew of the transmitter device, varying trace lengths and capacitive loading, variations in threshold voltages, transmission-line mis-terminations, or system reconfigurations. This results in inaccurate data transmission from one point to another and interrupted communication between components within the system.

A dynamic clock-data synchronization or phase alignment solution is optimal for high-speed systems because it provides a better tolerance to signal noise without the higher power consumption of devices which correct for skew using an individual analog PLL for each receiver channel. The dynamic phase aligner in Stratix GX devices shares the same components across many receiver channels, therefore reducing power consumption.

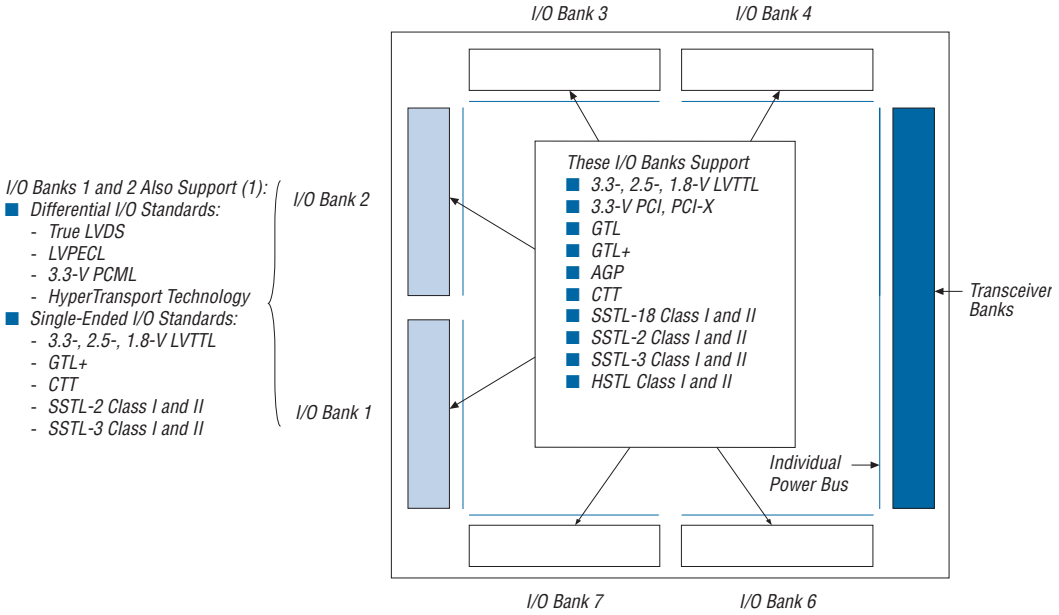
Figure 17–1. Clock to Data Skew



Stratix GX I/O Banks

Stratix GX devices contain seven I/O banks, as shown in Figure 17–2. I/O banks one and two support high-speed LVDS, LVPECL, 3.3-V PCML, HSTL class I and II, and SSTL-2 class I and II inputs and outputs. These two banks also incorporate an embedded dynamic phase aligner within the source-synchronous interface (see Figure 17–2). The dynamic phase aligner corrects for the phase difference between the clock and data lines caused by skew. The dynamic phase aligner operates automatically and continuously without requiring a fixed training pattern, and allows the source-synchronous circuitry to capture data correctly regardless of the channel-to-clock skew.

Figure 17–2. DPA Support in Stratix GX Devices



Note to Figure 17–2:
(1) You can only use the differential receiver and clock input pins as inputs for single-ended standards.

Dedicated Source-Synchronous Circuitry

The differential I/O channels in Stratix GX I/O banks 1 and 2 can interface with LVDS, LVPECL, or 3.3-V PCML I/O standards in source-synchronous mode. Stratix GX devices transmit or receive serial channels along with clocks. The receiving Stratix GX device can multiply the low-speed clock by a factor of 1, 2, 4, 8, or 10 for serializer/deserializer (SERDES) operation. The SERDES factor (*J*) can be 4, 8, or 10 (only 8 or 10 with DPA) and determines the width of the bus driving into the logic array. The SERDES factor (*J*) does not have to equal the clock-multiplication value (*W*). The Stratix GX device can bypass the dedicated SERDES for a serialization or deserialization factor of 1 or 2. If the serialization/deserialization factor is 2, the I/O element (IOE) uses the

double data rate (DDR) input and output. Table 17–1 shows the clock multiplication factors and the SERDES factors supported by Stratix GX devices.

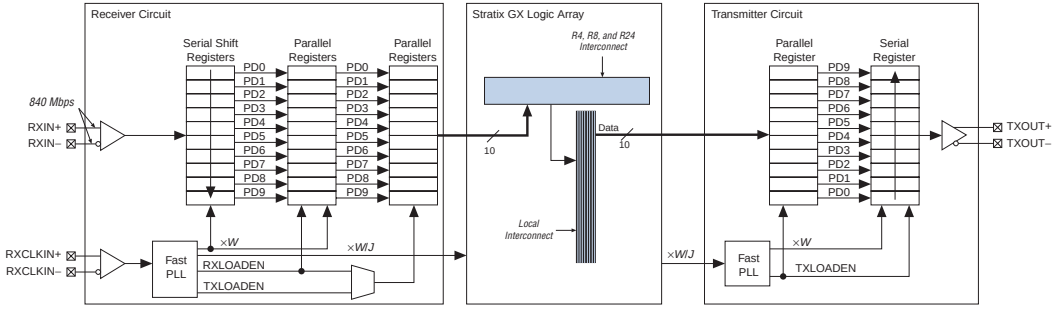
Table 17–1. Clock Multiplication Factors

Factor	Integer
Clock Multiplication *W*	1, 2, 4, 8, or 10
SERDES *J*	4, 8, or 10 *(1)*

Note to Table 17–1:
(1) The SERDES factor *J* can only be 8 or 10 when using DPA.

In the receiver circuitry, the fast PLL generates the high-frequency clock to deserialize the serial data through a shift register. The parallel data is synchronized with the low-frequency clock, and the receiver sends both to the logic array. On the transmitter side, the parallel data from the logic array is first fed into a parallel-in, serial-out shift register synchronized with the low-frequency clock and then transmitted out by the output buffers. Figure 17–3 shows the dedicated receiver and transmitter interface. For more information on the Stratix GX source-synchronous operation, refer to *AN 202: Using High-Speed Differential I/O Interfaces in Stratix Devices*.

Figure 17–3. Source-Synchronous Differential I/O Receiver/Transmitter Interface



The enable signal RXLOADEN loads the parallel data into the next parallel register on the second rising edge of the low-frequency clock in both modes (with or without DPA). Figure 17–4 shows the clock and data relationship in the receiver.

Figure 17–4. Receiver Timing Diagram

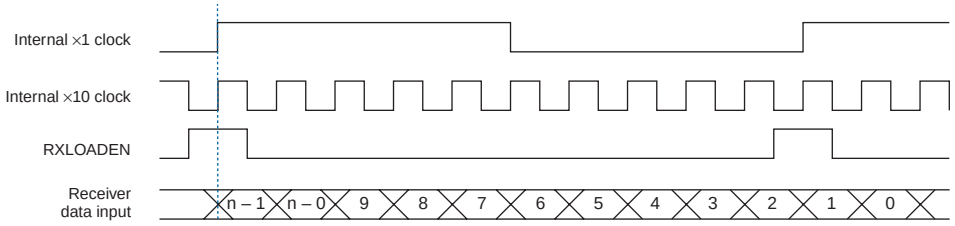
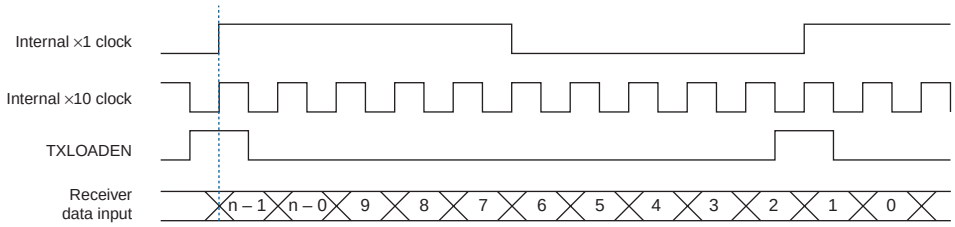


Figure 17–5 shows the timing relationship between the data and clock in the Stratix GX transmitter in ×10 mode.

Figure 17–5. Transmitter Timing Diagram

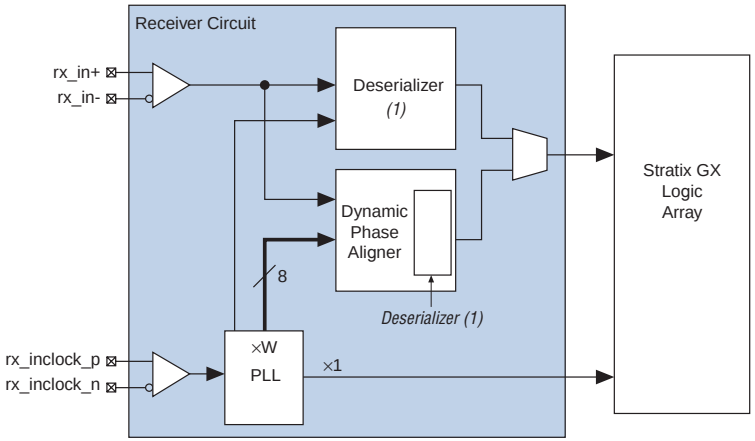


DPA Block Overview

Each Stratix GX receiver channel features a DPA block. The block contains a dynamic phase selector for phase detection and selection, a SERDES, a synchronizer, and a data realigner circuit. You can bypass the dynamic phase aligner without affecting the basic source-synchronous operation of the channel by using a separate deserializer shown in Figure 17–6.

The dynamic phase aligner uses both the source clock and the serial data. The dynamic phase aligner automatically and continuously tracks fluctuations caused by system variations and self-adjusts to eliminate the phase skew between the multiplied clock and the serial data. Figure 17–6 shows the relationship between Stratix GX source-synchronous circuitry and the Stratix GX source-synchronous circuitry with DPA.

Figure 17–6. Source-Synchronous DPA Circuitry



Unlike the de-skew function in APEX™ 20KE and APEX 20KC devices or the clock-data synchronization (CDS) circuit in APEX II devices, you do not have to use a fixed training pattern with DPA in Stratix GX devices or assert a pin to activate the circuit.

DPA Input Support

Stratix GX device I/O banks 1 and 2 contain dedicated circuitry to support differential I/O standards at speeds up to 1 Gbps with DPA (or up to 840 Mbps without DPA). Stratix GX device source-synchronous circuitry supports LVDS, LVPECL, and 3.3-V PCML I/O standards. Additionally, the clock input pins in I/O banks 1 and 2 support differential HSTL. Table 17–2 shows the I/O standards that the dynamic phase aligner supports and their corresponding supply voltage. All Stratix GX device differential receiver input pins and clock pins in I/O banks 1 and 2 are dedicated input pins. Transmitter pins can be either input or output pins for both differential and single-ended I/O standards. Refer to Table 17–3.

Table 17–2. DPA Differential I/O Standards

I/O Standard	V_{CC} I/O (V)
LVDS, LVPECL, 3.3-V PCML	3.3

Table 17–3. Bank 1 & 2 Input Pins

Input Pin Type	I/O Standard	Receiver Pin	Transmitter Pin
Differential	Differential	Input only	Output only
Single ended	Single ended	Input only	Input or output

Interface & Fast PLL

This section describes the number of channels that support DPA and their relationship with the PLL in Stratix GX devices. EP1SGX10 and EP1SGX25 devices have two dedicated fast PLLs and EP1SGX40 devices have four dedicated fast PLLs for clock multiplication. Table 17–4 shows the maximum number of channels in each Stratix GX device that support DPA.

Table 17–4. Stratix GX Source-Synchronous Differential I/O Resources

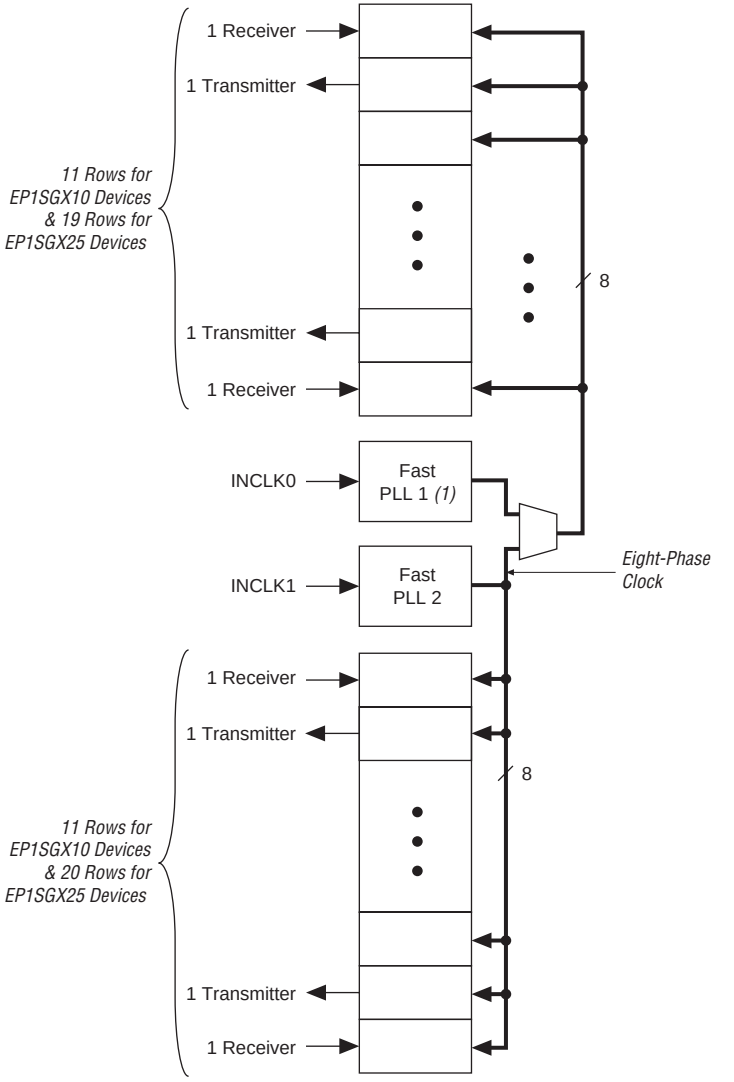
Device	Fast PLLs	Pin Count	Receiver Channels *(1)*	Transmitter Channels *(1)*	Receiver & Transmitter Channel Speed (Gbps) *(2)*	LEs
EP1SGX10C	2 *(3)*	672	22	22	1	10,570
EP1SGX10D	2 *(3)*	672	22	22	1	10,570
EP1SGX25C	2	672	39	39	1	25,660
EP1SGX25D	2	672	39	39	1	25,660
		1,020	39	39	1	25,660
EP1SGX25F	2	1,020	39	39	1	25,660
EP1SGX40D	4 *(4)*	1,020	45	45	1	41,250
EP1SGX40G	4 *(4)*	1,020	45	45	1	41,250

Notes to Table 17–4:

(1) This is the number of receiver or transmitter channels in the source-synchronous (I/O bank 1 and 2) interface of the device.
(2) Receiver channels operate at 1,000 Mbps with DPA. Without DPA, the receiver channels operate at 840 Mbps.
(3) One of the two fast PLLs in EP1SGX10C and EP1SGX10D devices supports DPA.
(4) Two of the four fast PLLs in EP1SGX40D and EP1SGX40G devices support DPA

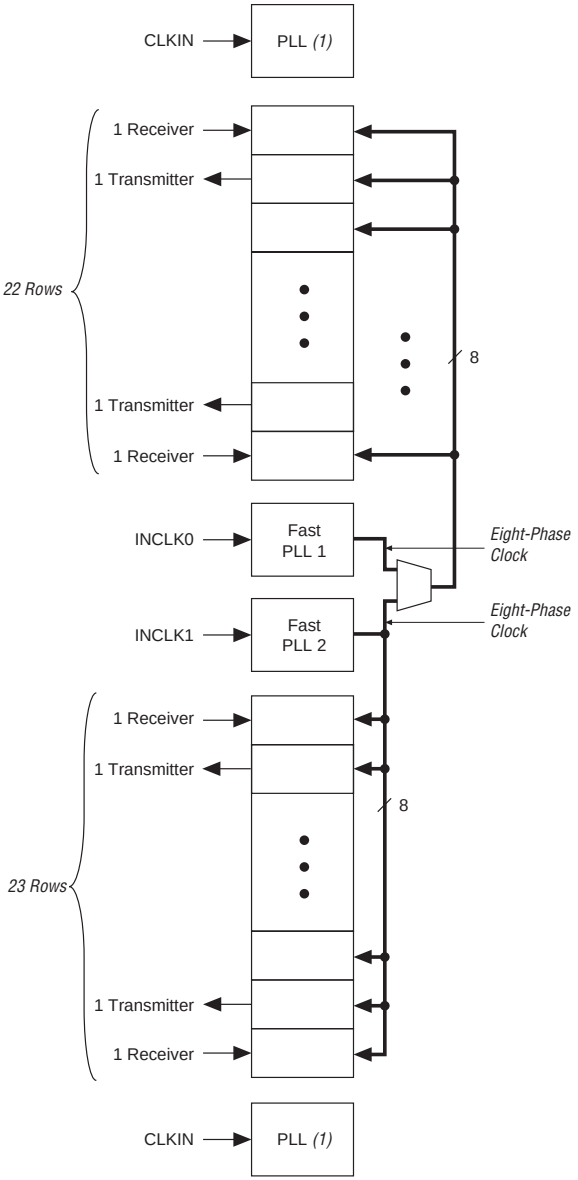
The receiver and transmitter channels are interleaved so that each I/O row in I/O banks 1 and 2 of the device has one receiver channel and one transmitter channel per row. Figures 17–7 and 17–8 show the fast PLL and channels with DPA layout in EP1SGX10, EP1SGX25, and EP1SGX40 devices. In EP1SGX10 devices, only fast PLL 2 supports DPA operations.

Figure 17–7. PLL & Channel Layout in EP1SGX10 & EP1SGX25 Devices



Note to Figure 17–7:
(1) Fast PLL 1 in EP1SGX10 devices does not support DPA.

Figure 17–8. PLL & Channel Layout in EP1SGX40 Devices *Notes (1), (2), (3)*



Notes to Figure 17–8:

(1) Corner PLLs do not support DPA.

(2) Not all eight phases are used by the receiver channel or transmitter channel in non-DPA mode.

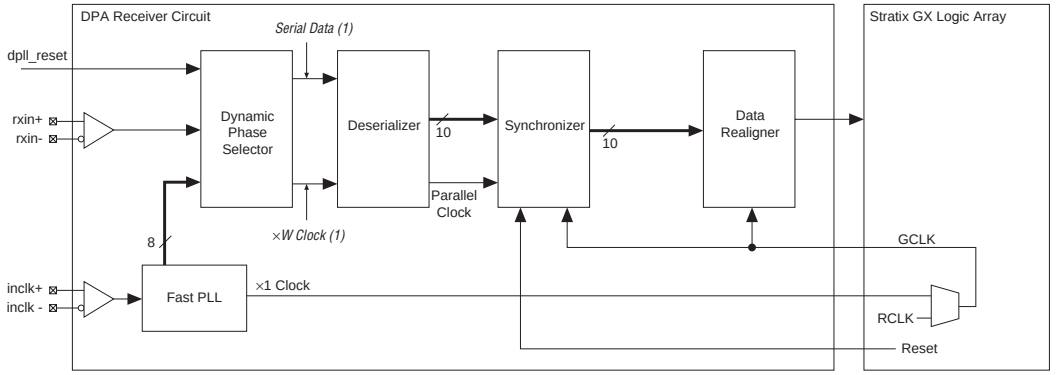
(3) The center PLLs can only clock 20 transceivers in either direction. Using Fast PLL2, you can clock a total of 40 transceivers, 20 in each direction.

DPA Operation

The DPA receiver circuitry contains the dynamic phase selector, the deserializer, the synchronizer, and the data realigner (see Figure 17–9). This section describes the DPA operation, synchronization and data realignment. You can enable or disable DPA operation on a channel-to-channel basis. In the SERDES with DPA mode, the source clock is fed to the fast PLL through the dedicated clock input pins. This clock is multiplied by the multiplication value *W* to match the serial data rate.

For information on the deserializer, see "Dedicated Source-Synchronous Circuitry" on page 17–3.

Figure 17–9. DPA Receiver Circuit

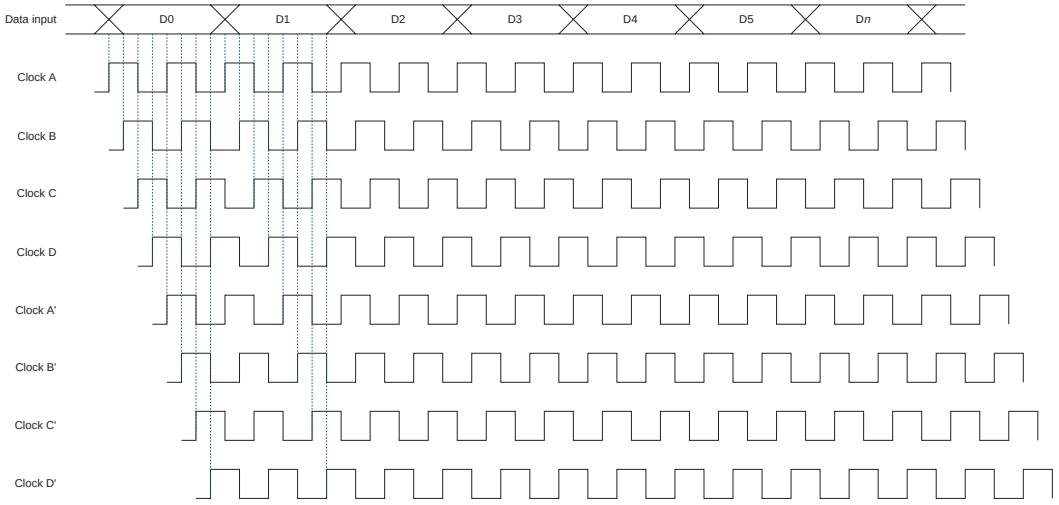


Note to Figure 17–9:
(1) These are phase-matched and retimed high-speed clocks and data.

The dynamic phase selector matches the phase of the high-speed clock and data before sending them to the deserializer.

The fast PLL supplies eight phases of the same clock (each a separate tap from a four-stage differential voltage-controlled oscillator (VCO)) to all the differential channels associated with the selected fast PLL. The DPA circuitry inside each channel locks to a phase closest to the serial data's phase and sends the retimed data and the selected clock to the deserializer. Each channel's DPA circuit can independently choose a different clock phase. The data phase detection and the clock phase selection process is automatic and continuous. The eight phases of clock gives the DPA circuit a granularity of one eighth of the unit interval (UI) or 125 ps at 1 Gbps. Figure 17–10 illustrates the clocks generated by the fast PLL circuitry and their relationship to a data stream.

Figure 17–10. Fast PLL Clocks & Data Input



Protocols, Training Pattern & DPA Lock Time

The dynamic phase aligner uses a fast PLL for clock multiplication, and the dynamic phase selector for the phase detection and alignment. The dynamic phase aligner uses the high-speed clock out of the dynamic phase selector to deserialize high-speed data and the receiver's source synchronous operations.

At each rising edge of the clock, the dynamic phase selector determines the phase difference between the clock and the data and automatically compensates for the phase difference between the data and clock.

The actual lock time for different data patterns varies depending on the data's transition density (how often the data switches between 1 and 0) and jitter characteristic. The DPA circuitry is designed to lock onto any data pattern with sufficient transition density, so the circuitry will work with current and future protocols. Experiments and simulations show that the DPA circuitry locks when the data patterns listed in Table 17–5 are repeated for the specified number of times. There are other suitable patterns not shown in Table 17–5 and/or pattern lengths, but the lock time may vary. The circuit can adjust for any phase variation that may occur during operation.

If the dynamic phase selector loses lock, the DPA circuitry sends a loss-of-lock signal for each channel to the logic array. You can then pull the dynamic phase selector RESET signal low to reset the dynamic phase selector. You can also reset the DPA operation by asserting the DPA RESET node.

Table 17–5. Training Patterns for Different Protocols

Protocols	Training Pattern	Number of Repetitions
SPI-4, NPSI	Ten 0's, ten 1's (0000000000111111111)	256
RapidIO	Four 0's, four 1's (00001111) or one 1, two 0's, one 1, four 0's (10010000)	
Other designs	Eight alternating 1's and 0's (10101010 or 01010101)	
SFI-4, XSBI	Not specified	

Phase Synchronizer

Each receiver has its own dynamic phase synchronizer. The receiver dynamic phase synchronizer aligns the phase of the parallel data from all the receivers to one global clock. The synchronizers in each channel consist of a first-in first-out (FIFO) buffer clocked by the global clock (GCLK) and parallel clock. The global clock (GCLK) and parallel clock input into the synchronizers must have identical frequency and differ only in phase. Therefore, the operation does not require an empty/full flag or read/write enable signals. The dynamic phase selector aligns each data signal with one of the eight phases of the global clock, so each signal has the same frequencies. Each synchronizer is written with a different clock phase, depending on the phase of the received data. The global clock reads all synchronizers, so all data is the same phase for use in the logic array.

Receiver Data Realignment In DPA Mode

While DPA operation aligns the incoming clock phase to the incoming data phase, it does not guarantee the parallelization boundary or byte boundary. When the dynamic phase aligner realigns the data bits, the bits may be shifted out of byte alignment, as shown in Figure 17–11.

Figure 17–11. Misaligned Captured Bit



The dynamic phase selector and synchronizer align the clock and data based on the power-up of both communicating devices, and the channel to channel skew. However, the dynamic phase selector and synchronizer cannot determine the byte boundary, and the data may need to be byte-aligned. The dynamic phase aligner's data realignment circuitry shifts data bits to correct bit misalignments.

The Stratix GX circuitry contains a data-realignment feature controlled by the logic array. Stratix GX devices perform data realignment on the parallel data after the deserialization block. The data realignment can be performed per channel for more flexibility. The data alignment operation requires a state machine to recognize a specific pattern. The procedure requires the bits to be slipped on the data stream to correctly align the incoming data to the start of the byte boundary.

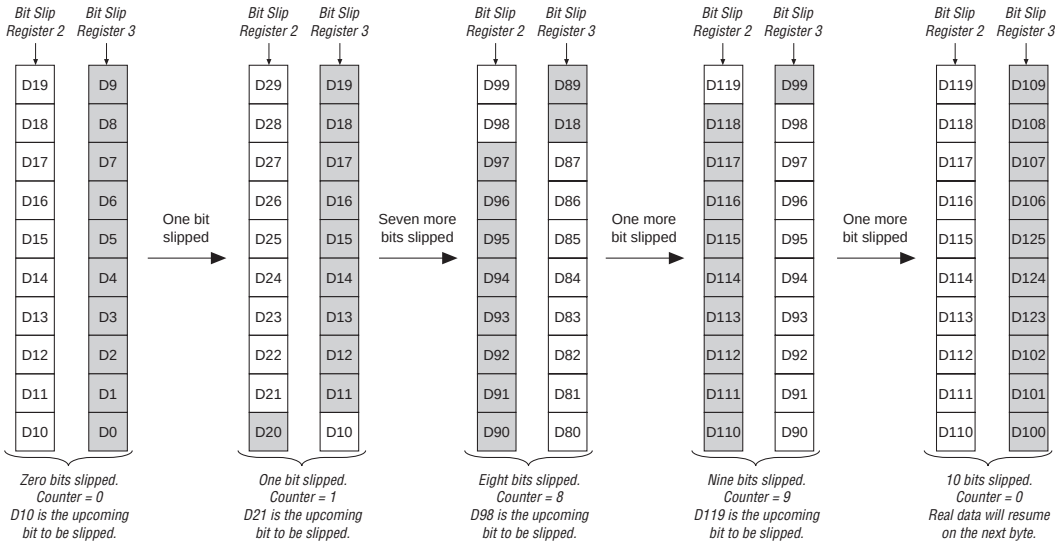
The DPA uses its realignment circuitry and the global clock for data realignment. Either a device pin or the logic array asserts the internal `rx_channel_data_align` node to activate the DPA data-realignment circuitry. Switching this node from low to high activates the realignment circuitry and the data being transferred to the logic array is shifted by one bit.

A state machine and additional logic can monitor the incoming parallel data and compare it against a known pattern. If the incoming data pattern does not match the known pattern, you can activate the `rx_channel_data_align` node again. Repeat this process until the realigner detects the desired match between the known data pattern and incoming parallel data pattern.

The DPA data-realignment circuitry allows further realignment beyond what the *J* multiplication factor allows. You can set the *J* multiplication factor to be 8 or 10. However, since data must be continuously clocked in on each low-speed clock cycle, the upcoming bit to be realigned and previous *n* –1 bits of data will be selected each time the data realignment logic's counter passes *n* –1. At this point the data is selected entirely from bit-slip register 3 (see Figure 17–12) as the counter is reset to 0. The logic

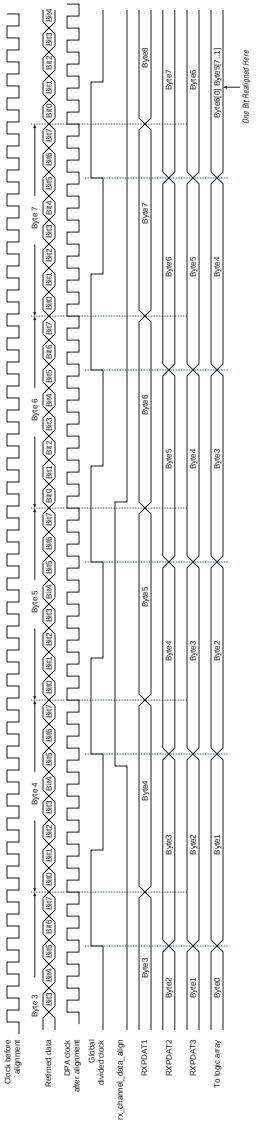
array receives a new valid byte of data on the next divided low-speed clock cycle. Figure 17–12 shows the data realignment logic output selection from data in the data realignment register 2 and data realignment register 3 based on its current counter value upon continuous request of data slipping from the logic array.

Figure 17–12. DPA Data Realigner



Use the rx_channel_data_align signal within the device to activate the data realigner. You can use internal logic or an external pin to control the rx_channel_data_align signal. To ensure the rising edge of the rx_channel_data_align signal is latched into the control logic, the rx_channel_data_align signal should stay high for at least two low-frequency clock cycles. Figure 17–13 shows the timing diagram of the DPA circuit. The byte boundary of the data is shifted by one bit on each rising-edge of the rx_channel_data_align signal. Thus one bit will be lost every time the data is slipped.

Figure 17–13. Data Realignment to Clock Timing Relationship



In order to manage the alignment procedure, a state machine should be built in the FPGA logic array to generate the realignment signal. The following guidelines outline the requirements for this state machine.

- The design must include an input synchronizing register to ensure that data is synchronized to the ×*W*/*J* clock.
- After the state machine, use another synchronizing register to capture the generated `rx_channel_data_align` signal and synchronize it to the ×*W*/*J* clock.
- The skew in the path from the output of this synchronizing register to the PLL is undefined, so the state machine must generate a pulse that is high for two ×*W*/*J* clock periods.
- The `rx_channel_data_align` generator circuitry only generates a single fast clock period pulse for each `rx_channel_data_align` pulse, so you cannot generate additional `rx_channel_data_align` pulses until the signal comparing the incoming data to the alignment pattern is reset low.
- To guarantee the state machine does not incorrectly generate multiple `rx_channel_data_align` pulses to shift a single bit, the state machine must hold the `rx_channel_data_align` signal low for at least three ×1 clock periods between pulses.

Source-Synchronous Circuitry with DPA vs. CDR

The DPA feature and source-synchronous channels are complementary features within Stratix GX devices to be used with high-speed transceiver blocks. The channels on the transceiver side of the device use an embedded circuit dedicated for receiving and transmitting serial data streams to and from the system board at frequencies up to 3.125 Gbps. These channels are clustered in serial transceiver blocks that contain four channels each and handle complex encoding and decoding schemes. If your system requires more than twenty channels, but your data rate is between 0.622 and 1.0 Gbps and you don't require the complex coding or decoding schemes, you can use the channels in banks 1 and 2 to implement the source-synchronous channels with DPA.

Software Support

You can configure the Stratix GX LVDS transmitter and receiver blocks using the MegaWizard® Plug-In Manager in the Quartus® II software. The wizard is a GUI-based ports and parameter selector for the `altlvds` megafunction. This section describes the available options for the Stratix GX LVDS transmitter and receiver.

Figure 17–14 shows the first page of the MegaWizard Plug-In Manager. With this page you can create a new megafunction, edit an existing megafunction, or copy an existing megafunction to create a variant. This section describes how to create a new megafunction.

Figure 17–14. MegaWizard Plug-In Manager (Page 1)

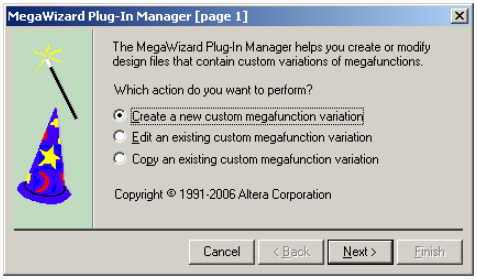
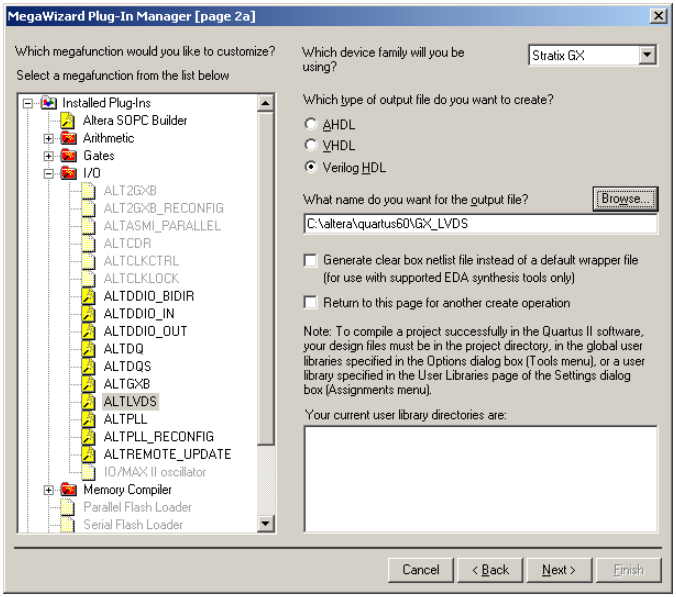


Figure 17–15 allows you to choose the megafunction to configure and the device family. You can also choose the HDL language you want the output file to be compatible with. For schematic entry, any HDL can be selected. You must provide a base name for the output files. Figure 17–15 shows the second page of the wizard.

Figure 17–15. MegaWizard Plug-In Manager (Page 2a)



Stratix GX Transmitter

Stratix GX transmitter setup starts on the page shown in Figure 17–16.

On this page, you select which device the megafunction is for in **Use which device family?** This selection activates available options for each device family. For example, for Stratix GX devices, the DPA option is available, but the Use External PLL option is not.

You can also set the number of channels and the deserialization factor. The deserialization factor determines the parallel clock frequency and the word width in the PLD logic array.

Figure 17–16. MegaWizard Plug-In Manager - altlvds Transmitter

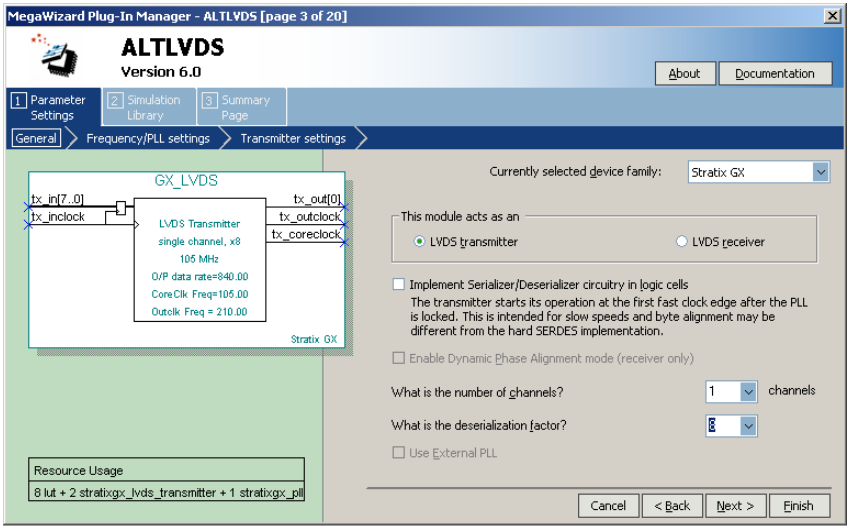


Figure 17–17 shows the `altlvds` MegaWizard Plug-In Manager page where you select the data rate and the transmitter clocking. The maximum input clock frequency is 717 MHz.

You can adjust the phase relation of the incoming data and reference clock in the **What is the phase alignment of data with respect to the rising edge of tx_inclock? (in degrees)** option.

You can enable the `tx_pll_enable` and `pll_areaset` ports. The `tx_pll_enable` port disables or enables the fast PLL used for the current instance. The `pll_areset` port resets all the counters to the fast PLL.

The **Use shared PLL(s) for receiver and transmitter** option allows you to merge the PLL for the receiver and transmitter under the correct conditions (the same data rate, SERDES factor, and input clock frequency).

Use the **Register `tx_in` input port using** option as necessary to transfer data from the PLD to the transmitter. Turn off this option if the design already contains a layer of registers before the PLD to transmitter interface. The clock feeding the register is fed by `tx_inclock` or `tx_coreclock`, depending on what feeds the data path in the PLD logic array.

Figure 17–17. MegaWizard Plug-In Manager - altlvds Transmitter

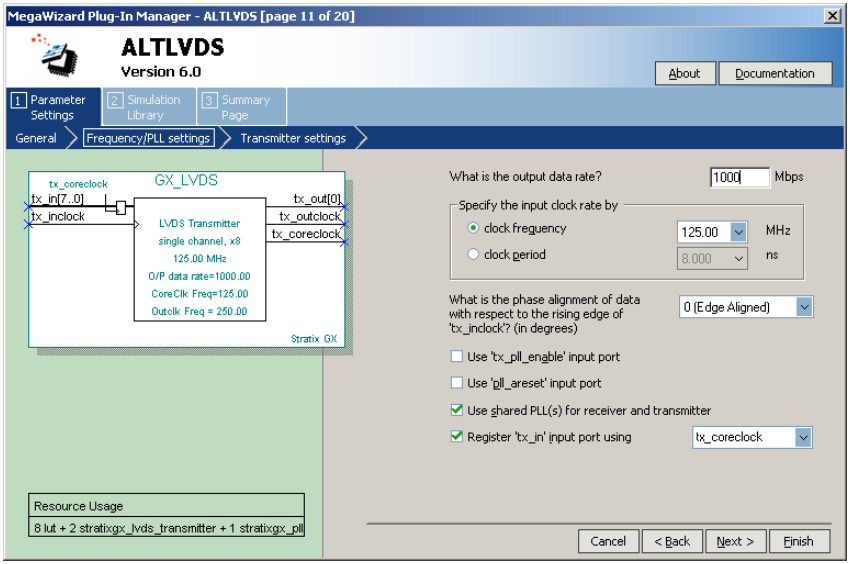
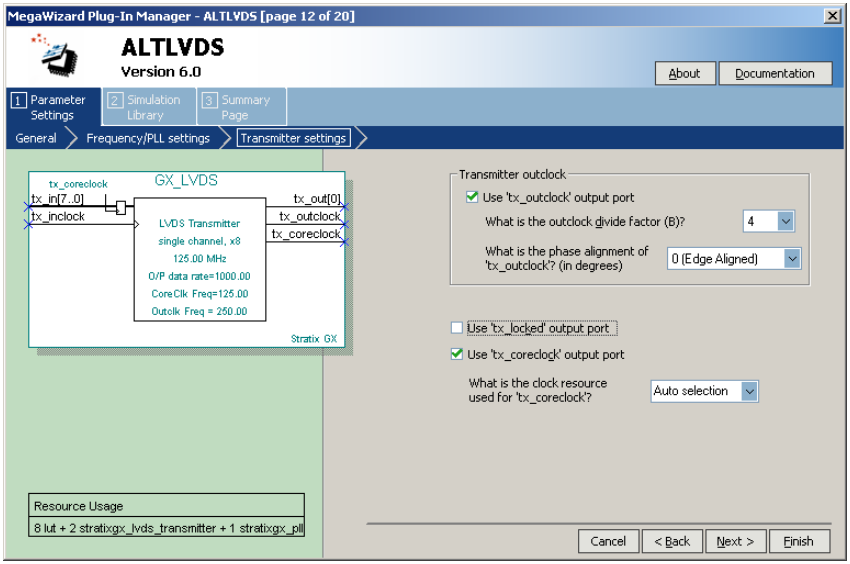


Figure 17–18 shows the last transmitter options page of the wizard. On this page you can enable the `tx_locked` and `tx_coreclock` ports. The `tx_locked` port indicates if the fast PLL is locked to the reference clock. The `tx_coreclock` port supplies the PLD with a clock, and is useful when the frequency of `tx_inclock` does not match the data rate divided by the SERDES factor.

You can set `tx_coreclock` to use a specific clock resource or, if you choose Auto selection, the Quartus II software automatically allocates an available clock resource. You set up the division factor and phase of the output clock independently of the input clock.

Figure 17–18. MegaWizard Plug-In Manager - altlvds Transmitter



Stratix GX Receiver Without DPA

Stratix GX receiver setup without DPA starts on the page shown in Figure 17–19. The number of channels and deserialization factor are similar to those of the transmitter. The DPA option is available for the Stratix GX family. In Figure 17–19, DPA mode is not selected.

Figure 17–19. MegaWizard Plug-In Manager - altlvds Receiver Without DPA

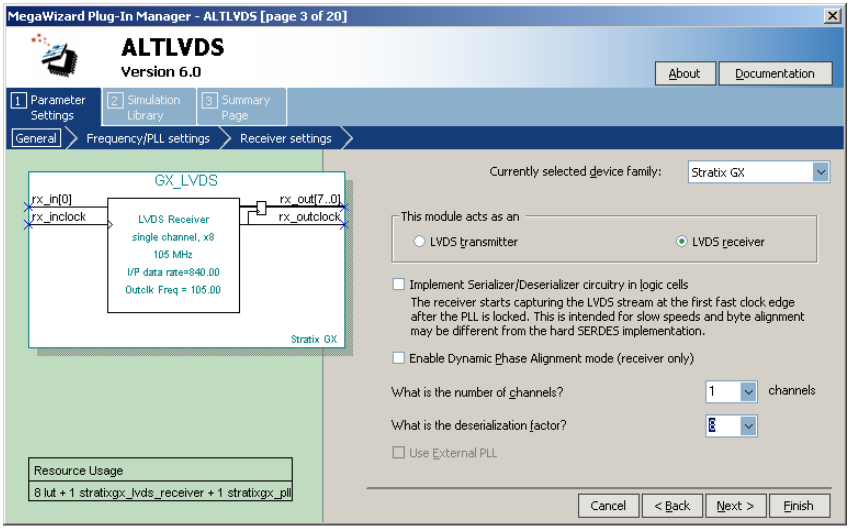


Figure 17–20 shows the page of the wizard where you can select the data rate and the receiver clocking. The maximum input clock frequency is 717 MHz. The maximum data rate for non-DPA operations is 840 Mbps. With DPA, the maximum data rate is 1000 Mbps.

The **Use shared PLL(s) for receiver and transmitter** option allows you to merge the PLL for the receiver and transmitter under the correct conditions (the same data rate, SERDES factor, and input clock frequency).

You can select the phase relationship of `rx_inclock` and `rx_in` using the **What is the phase alignment of data with respect to the rising edge of rx_inclock (in degrees)?** option.

Figure 17–20. MegaWizard Plug-In Manager - altlvds Receiver Without DPA

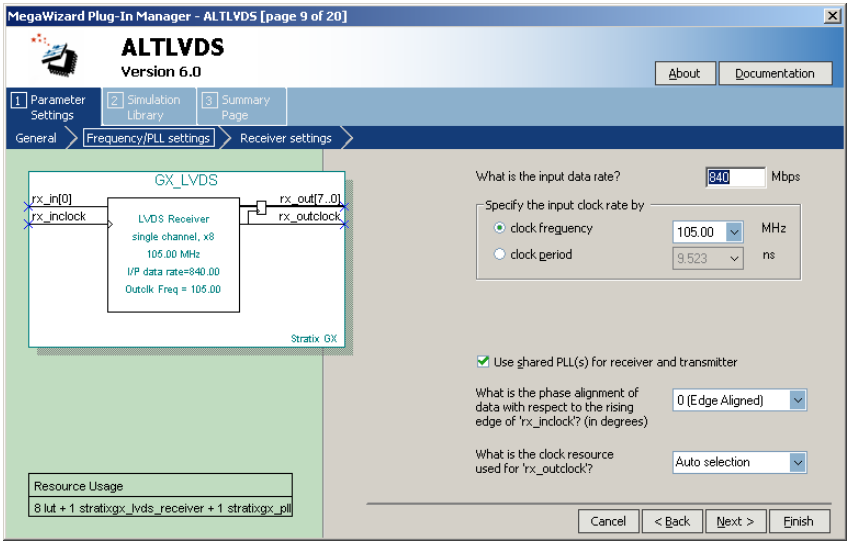
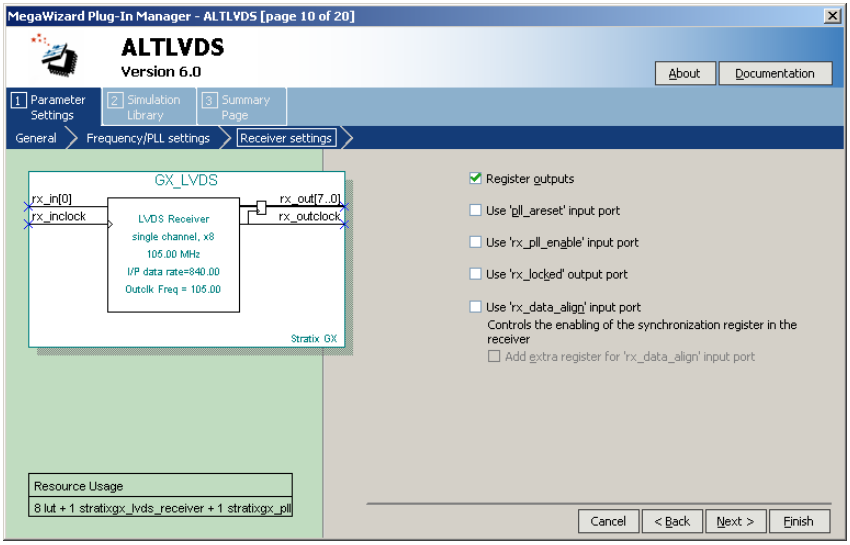


Figure 17–21 shows the last configuration page for the `altlvds` receiver without DPA.

On this page, turn on the **Register outputs** option to facilitate proper SERDES to PLD data transfer. You can turn off this option if there is already a layer of registers before the SERDES to reduce latency.

You can enable the `pll_areset`, `rx_pll_enable`, `rx_locked`, and `rx_data_align` ports on this page. The `pll_areset` port resets the counters in the fast PLL. The `rx_pll_enable` port disables or enables the fast PLL in this receiver instance. The `rx_locked` port indicates when the PLL is locked to the `rx_inclock` frequency and phase. The `rx_data_align` port pauses the fast PLL clock, thereby skipping the reception of the next bit. The `rx_data_align` port affects all channels of the receiver instance at the same time.

Figure 17–21. MegaWizard Plug-In Manager - altlvds Receiver Without DPA



Stratix GX Receiver with DPA

Stratix GX receiver setup with DPA starts on page three. The number of channels and the deserialization factor are similar to those of the transmitter. The DPA option is available for the Stratix GX family. Figure 17–22 shows the page of the `altlvds` MegaWizard Plug-In Manager with DPA selected.

Figure 17–22. MegaWizard Plug-In Manager - altlvds Receiver with DPA

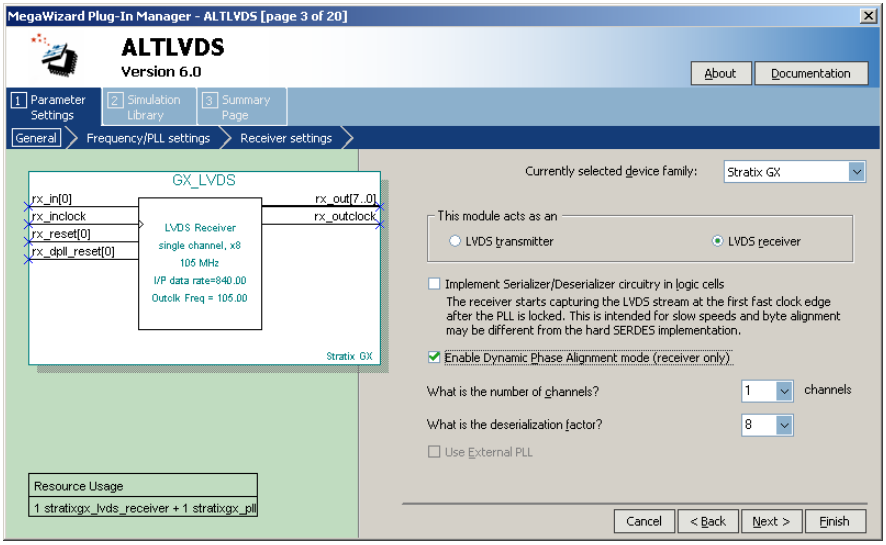


Figure 17–23 shows the page where you select the data rate and input clocking for the receiver. The maximum data rate is 1000 Mbps and the maximum input clock frequency is 717 MHz.

The **Use shared PLL(s) for receiver and transmitter** option allows you to merge the PLL for the receiver and transmitter under the right conditions (the same data rate, SERDES factor, and input clock frequency).

You can set `rx_outclock` to use a specific clock resource or, if you select Auto selection, the Quartus II software automatically allocates an available clock resource.

You must turn on **Enable the FIFO for DPA channels** option when in DPA mode. This FIFO buffer compensates for any phase differences between the selected phase in the DPA block and `rx_outclock`. There might be data errors if this option is turned off.

Figure 17–23. MegaWizard Plug-In Manager - altlvds Receiver with DPA

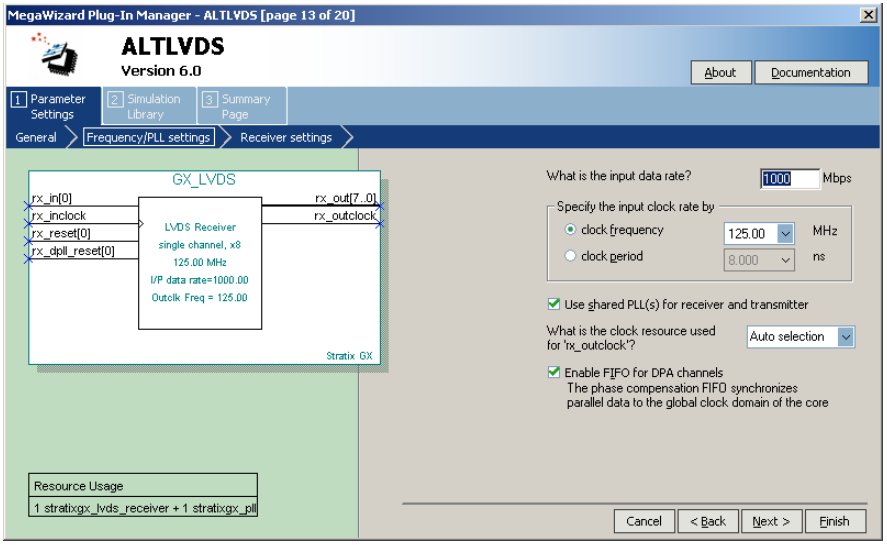


Figure 17–24 shows the last configuration page for the receiver in DPA mode.

Turn on the **Register outputs** option unless there is a layer of register in the user logic right before the SERDES block.

The `pll_areset` port resets the counters in the fast PLL. The `rx_pll_enable` port disables or enables the fast PLL in this receiver instance.

The `rx_channel_data_align` port is a channel-driven port that slips a bit for every rising edge on this port. Each DPA receiver channel has its own `rx_channel_data_align` port that can be used independently of each other.

The `rx_coreclk` and `rx_locked` ports operate the same as in the non-DPA receiver channel. The `rx_locked` port indicates when the PLL is locked to the `rx_inclock` frequency and phase. Turn on the `rx_coreclk` input port option to synchronize the `rx_outdata` to a local PLD clock instead of the parallel clock generated by the fast PLL.

Figure 17–24. MegaWizard Plug-In Manager - altlvds Receiver with DPA

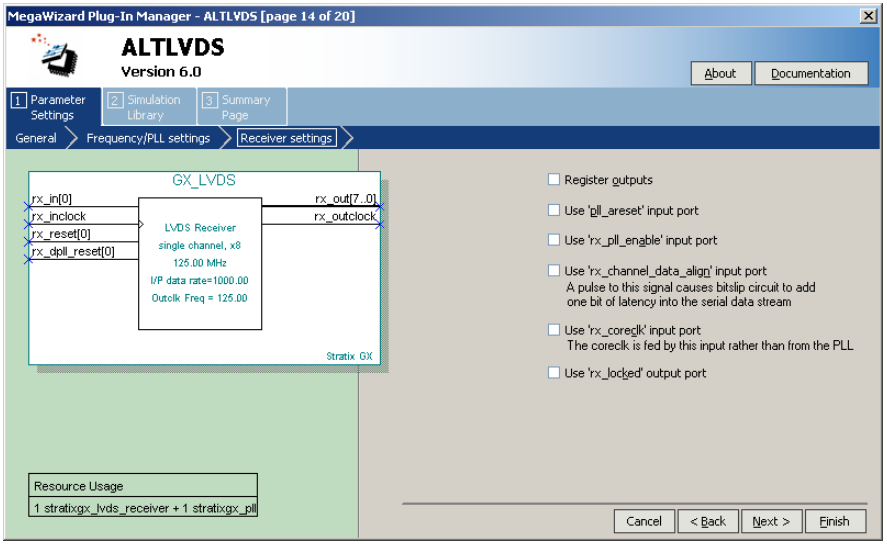


Figure 17–25 shows the Simulation Libraries page of the wizard.

Figure 17–25. MegaWizard Plug-In Manager - altlvds

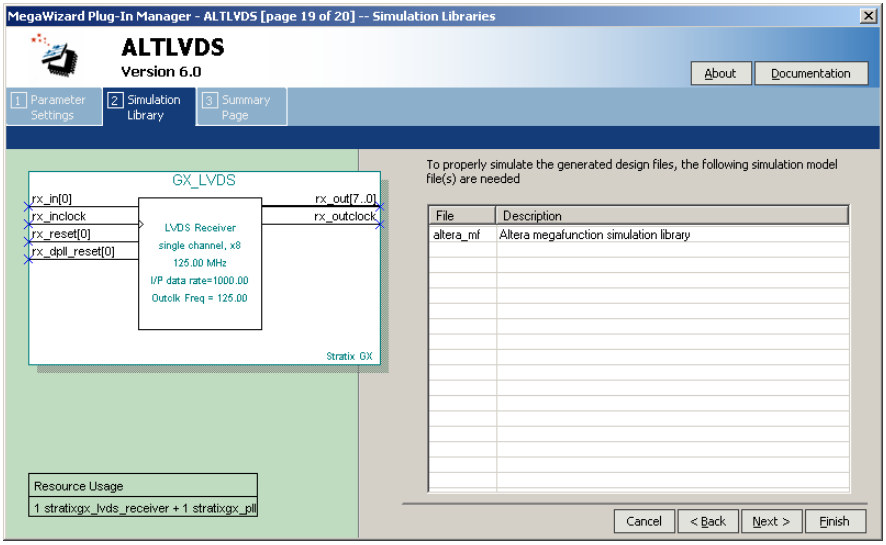
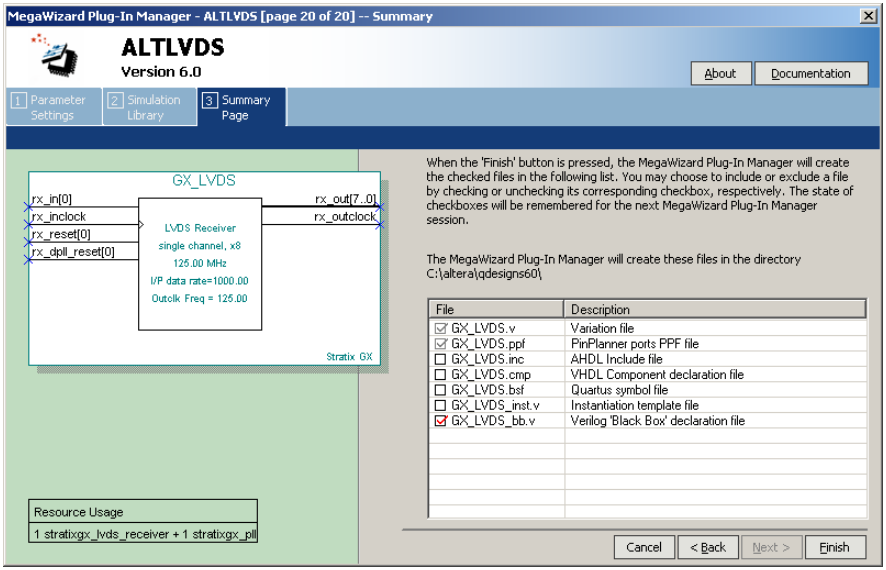


Figure 17–26 shows the last page of the MegaWizard Plug-In Manager. This page shows the files the Quartus II software generates with the options you selected on previous pages.

Figure 17–26. MegaWizard Plug-In Manager - altlvds Receiver



Summary

DPA technology eliminates the restriction of phase-matching the serial data and the source clock at the receiver channels. As a result, DPA eliminates tight board routing and topology restrictions, simplifies channel-to-channel skew calculation, and improves system performance. The combination of DPA technology with 3.125-Gbps transceivers allows Stratix GX devices to address a variety of applications and to effectively implement silicon bridges between protocols.

Section V. Digital Signal Processing

This section provides information for design and optimization of digital signal processing (DSP) functions and arithmetic operations in the on-chip DSP blocks.

This section includes the following chapters:

- Chapter 18, DSP Blocks in Stratix & Stratix GX Devices
- Chapter 19, Implementing High Performance DSP Functions in Stratix & Stratix GX Devices

Revision History

The table below shows the revision history for Chapters 18 and 19.

Chapter(s)	Date / Version	Changes Made
18	July 2005, v2.2	Chapter updated as part of the *Stratix Device Handbook* update.
	September 2004 v2.1	Added document to the *Stratix GX Device Handbook*.
19	September 2004 v1.1	Added document to the *Stratix GX Device Handbook*.

18. DSP Blocks in Stratix & Stratix GX Devices

S52006-2.2

Introduction

Traditionally, designers had to make a trade-off between the flexibility of off-the-shelf digital signal processors and the performance of custom-built devices. Altera® Stratix® and Stratix GX devices eliminate the need for this trade-off by providing exceptional performance combined with the flexibility of programmable logic devices (PLDs). Stratix and Stratix GX devices have dedicated digital signal processing (DSP) blocks, which have high-speed parallel processing capabilities, that are optimized for DSP applications. DSP blocks are ideal for implementing DSP applications that need high data throughput.

The most commonly used DSP functions are finite impulse response (FIR) filters, complex FIR filters, infinite impulse response (IIR) filters, fast Fourier transform (FFT) functions, discrete cosine transform (DCT) functions, and correlators. These functions are the building blocks for more complex systems such as wideband code division multiple access (W-CDMA) basestations, voice over Internet protocol (VoIP), and high-definition television (HDTV).

Although these functions are complex, they all use similar building blocks such as multiply-adders and multiply-accumulators. Stratix and Stratix GX DSP blocks combine five arithmetic operations—multiplication, addition, subtraction, accumulation, and summation—to meet the requirements of complex functions and to provide improved performance.

This chapter describes the Stratix and Stratix GX DSP blocks, and explains how you can use them to implement high-performance DSP functions. It addresses the following topics:

- Architecture
- Operational Modes
- Software Support



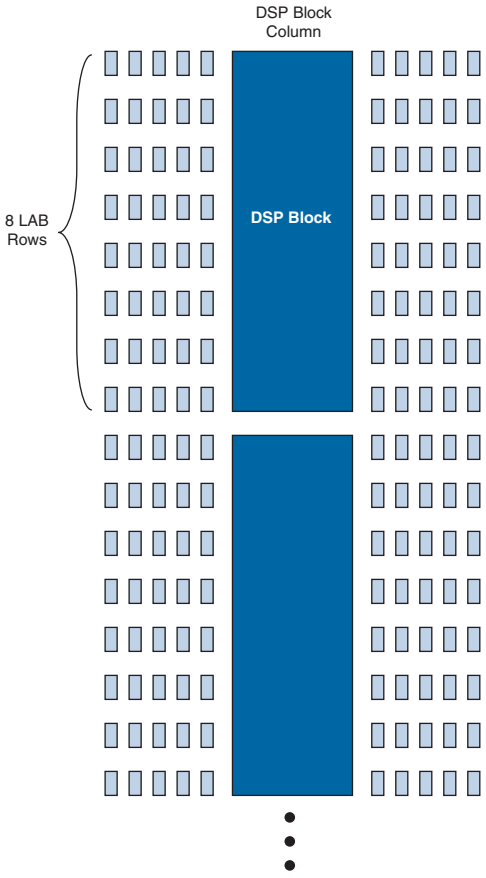
See the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* and the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1* for more information on Stratix and Stratix GX devices, respectively.

DSP Block Overview

Each Stratix and Stratix GX device has two columns of DSP blocks that efficiently implement multiplication, multiply accumulate (MAC), and filtering functions. Figure 18–1 shows one of the columns with surrounding LAB rows. You can configure each DSP block to support:

- Eight 9 × 9 bit multipliers
- Four 18 × 18 bit multipliers
- One 36 × 36 bit multiplier

Figure 18–1. DSP Blocks Arranged in Columns



The multipliers can then feed an adder or an accumulator block, depending on the DSP block operational mode. Additionally, you can use the DSP block input registers as shift registers to implement applications such as FIR filters efficiently. The number of DSP blocks per column

increases with device density. Tables 18–1 and 18–2 describe the number of DSP blocks in each Stratix and Stratix GX device, respectively, and the multipliers that you can implement.

Table 18–1. Number of DSP Blocks in Stratix Devices Note (1)

Device	DSP Blocks	9 × 9 Multipliers	18 × 18 Multipliers	36 × 36 Multipliers
EP1S10	6	48	24	6
EP1S20	10	80	40	10
EP1S25	10	80	40	10
EP1S30	12	96	48	12
EP1S40	14	112	56	14
EP1S60	18	144	72	18
EP1S80	22	176	88	22

Table 18–2. Number of DSP Blocks in Stratix GX Devices Note (1)

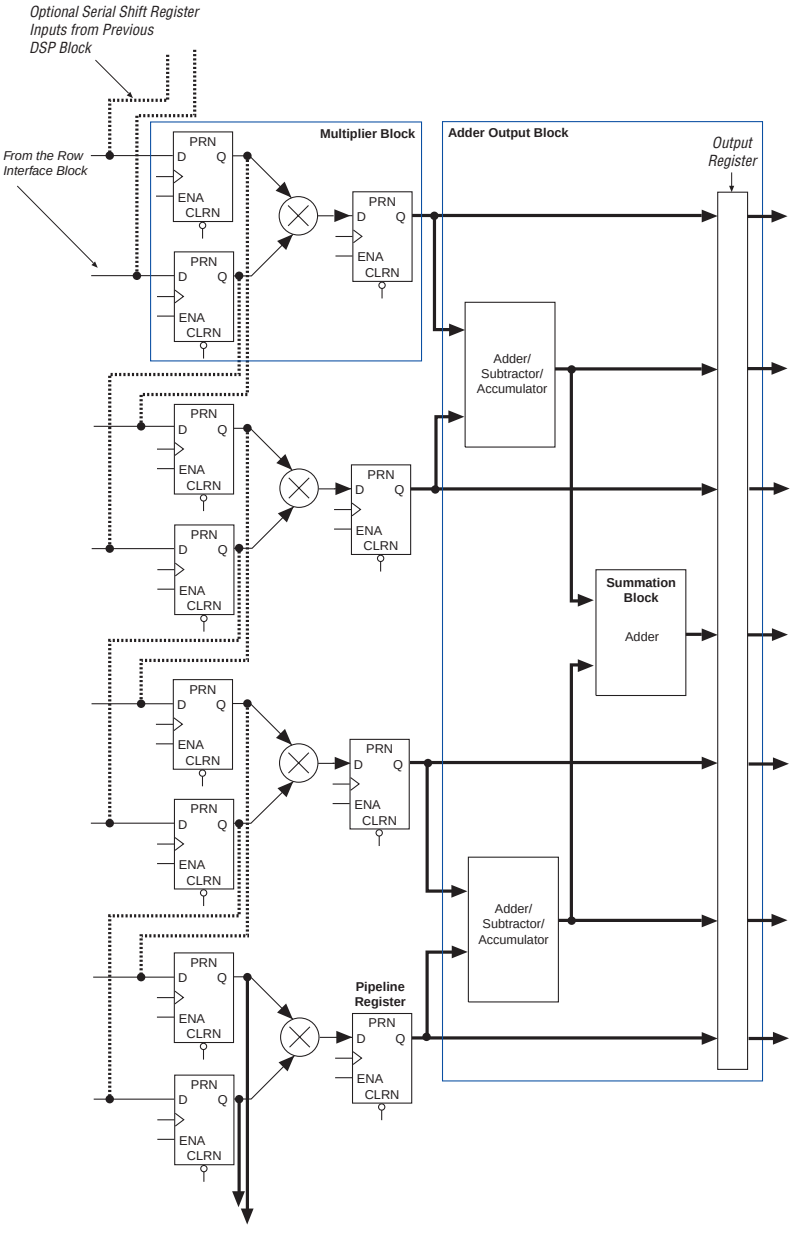
Device	DSP Blocks	9 × 9 Multipliers	18 × 18 Multipliers	36 × 36 Multipliers
EP1SGX10C	6	48	24	6
EP1SGX10D	6	48	24	6
EP1SGX25C	10	80	40	10
EP1SGX25D	10	80	40	10
EP1SGX25F	10	80	40	10
EP1SGX40D	14	112	56	14
EP1SGX40G	14	112	56	14

Note to Tables 18–1 and 18–2:

(1) Each device has either the number of 9 × 9-, 18 × 18-, or 36 × 36-bit multipliers shown.The total number of multipliers for each device is not the sum of all the multipliers.

Figure 18–2 shows the DSP block operating as an 18 × 18 multiplier.

Figure 18–2. DSP Block in 18 × 18 Mode



Architecture

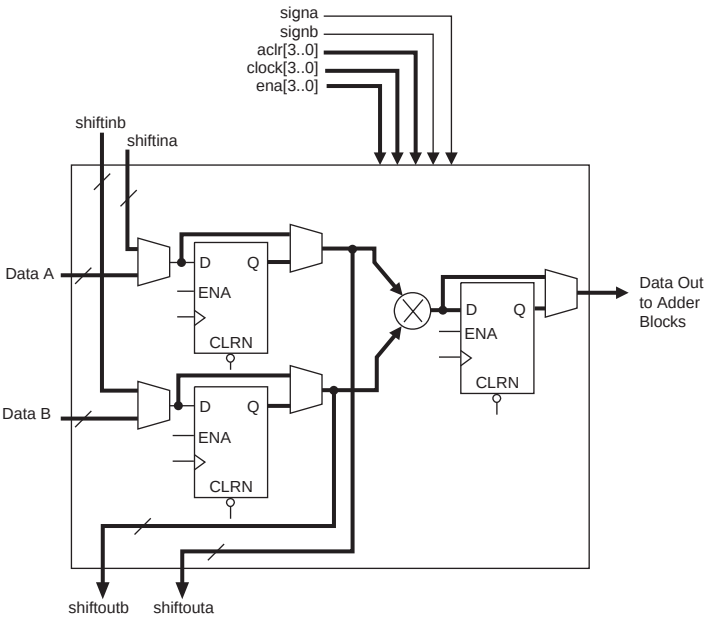
The DSP block consists of the following elements:

- A multiplier block
- An adder/subtractor/accumulator block
- A summation block
- An output interface
- Output registers
- Routing and control signals

Multiplier Block

Each multiplier block has input registers, a multiplier stage, and a pipeline register. See Figure 18–3.

Figure 18–3. Multiplier Block Architecture



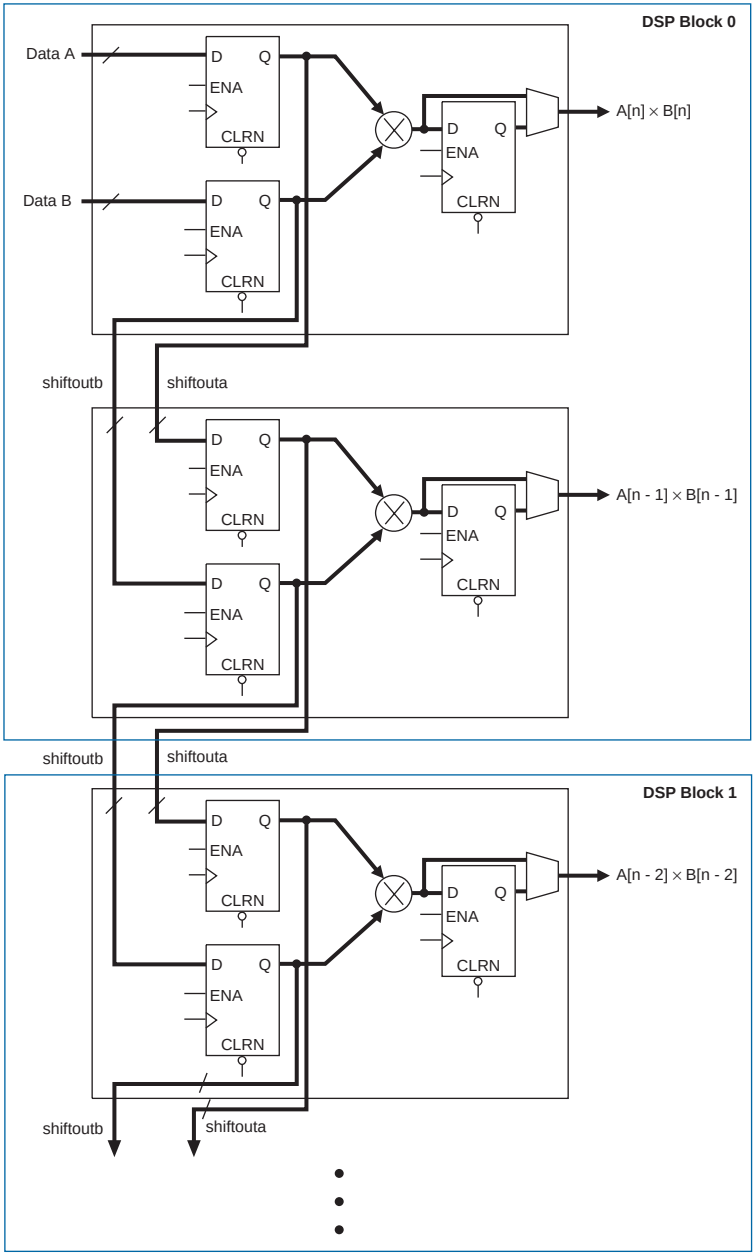
Input Registers

Each operand feeds an input register or the multiplier directly. The DSP block has the following signals (one of each controls every input and output register):

- `clock[3..0]`
- `ena[3..0]`
- `aclr[3..0]`

The input registers feed the multiplier and drive two dedicated shift output lines, `shiftouta` and `shiftoutb`. The shift outputs from one multiplier block directly feed the adjacent multiplier block in the same DSP block (or the next DSP block), as shown in Figure 18–4 on page 18–7, to form a shift register chain. This chain can terminate in any block, i.e., you can create any length of shift register chain up to 224 registers. A shift register is useful in DSP applications such as FIR filters. When implementing 9 × 9 and 18 × 18 multipliers, you do not need external logic to create the shift register chain because the input shift registers are internal to the DSP block. This implementation greatly reduces the required LE count and routing resources, and produces repeatable timing.

Figure 18–4. Shift Register Chain



Multiplier Stage

The multiplier stage supports 9 × 9, 18 × 18, or 36 × 36 multiplication. (The multiplier stage also support smaller multipliers. See "Operational Modes" on page 18–18 for details.) Based on the data width, a single DSP block can perform many multiplications in parallel.

The multiplier operands can be signed or unsigned numbers. Two signals, `signa` and `signb`, indicate the representation of the two operands. For example, a logic 1 on the `signa` signal indicates that data A is a signed number; a logic 0 indicates an unsigned number. The result of the multiplication is signed if any one of the operands is a signed number, as shown in Table 18–3.

Table 18–3. Multiplier Signed Representation

Data A	Data B	Result
Unsigned	Unsigned	Unsigned
Unsigned	Signed	Signed
Signed	Unsigned	Signed
Signed	Signed	Signed

The `signa` and `signb` signals affect the entire DSP block. Therefore, all of the data A inputs feeding the same DSP block must have the same sign representation. Similarly, all of the data B inputs feeding the same DSP block must have the same sign representation. The multiplier offers full precision regardless of the sign representation.



By default, the Altera Quartus® II software sets the multiplier to perform unsigned multiplication when the `signa` and `signb` signals are not used.

Pipeline Registers

The output from the multiplier can feed a pipeline register or be bypassed. You can use pipeline registers for any multiplier size; pipelining is useful for increasing the DSP block performance, particularly when using subsequent adder stages.



In the DSP block, pipelining improves the performance of 36 × 36 multipliers. For 18 × 18 multipliers and smaller, pipelining adds latency but does not improve performance.

Adder/Output Block

The adder/output block has the following elements (See Figure 18–5 on page 18–10):

- An adder/subtractor/accumulator block
- A summation block
- An output select multiplexer
- Output registers

You can configure the adder/output block as:

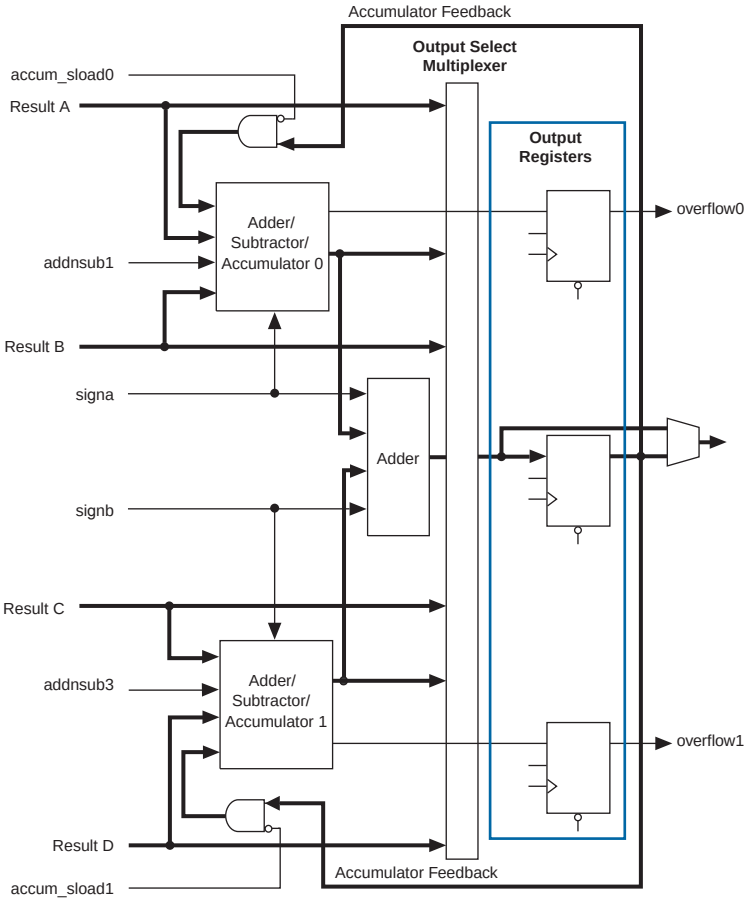
- A pure output interface
- An accumulator
- A simple one-level adder
- A two-level adder with dynamic addition/subtraction control on the first-level adder
- The final stage of a 36-bit multiplier

The output select multiplexer sets the output of the DSP block. You can register the adder/output block's output using the output registers.



You cannot use the adder/output block independently of the multiplier.

Figure 18–5. Adder/Output Block



Adder/Subtractor/Accumulator Block

The adder/subtractor/accumulator is the first level of the adder/output block. You can configure the block as an accumulator or as an adder/subtractor.

Accumulator

When the adder/subtractor/accumulator is configured as an accumulator, the output of the adder/output block feeds back to the accumulator as shown in Figure 18–5. You can use the

accum_sload[1..0] signals to clear the accumulator asynchronously. This action is not the same as resetting the output registers. You can clear the accumulation and begin a new one without losing any clock cycles.

The overflow signal goes high on the positive edge of the clock when the accumulator overflows or underflows. In the next clock cycle, however, the overflow signal resets to zero even though an overflow (or underflow) occurred in the previous clock cycle. Use a latch to preserve the overflow condition indefinitely (until the latch is cleared).

Adder/Subtractor

The addnsub[1..0] signals select addition or subtraction: high for addition and low for subtraction. You can control the addnsub[1..0] signals using external logic; therefore, the first-level block can switch from an adder to a subtractor dynamically, simply by changing the addnsub[1..0] signals. If the first stage is configured as a subtractor, the output is A - B and C - D.

The adder/subtractor also uses two signals, signa and signb, like the multiplier block. These signals indicate the sign representation of both operands together. You can register the signals with a latency of 1 or 2 clock cycles.

Summation Block

The output from the adder/subtractor feeds to an optional summation block, which is an adder block that sums the outputs of the adder/subtractor. The summation block is important in applications such as FIR filters.

Output Select Multiplexer

The outputs from the various elements of the adder/output block are routed through an output select multiplexer. Based on the DSP block operational mode, the outputs of the multiplier block, adder/subtractor/accumulator, or summation block feed straight to the output, bypassing the remaining blocks in the DSP block.



The output select multiplier configuration is configured automatically by software.

Output Registers

You can use the output registers to register the DSP block output. Like the input registers, the output registers are controlled by the four clock[3..0], aclr[3..0], and ena[3..0] signals. You can use the output registers in any DSP block operational mode.



The output registers form part of the accumulator in the multiply-accumulate mode.

Routing Structure & Control Signals

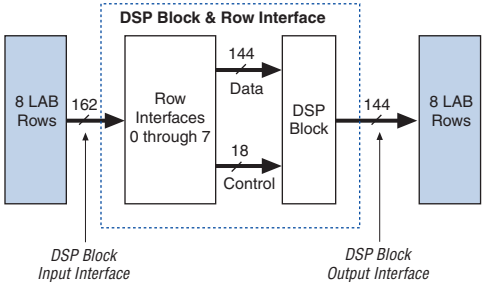
This section describes the interface between the DSP blocks and the row interface blocks. It also describes how the DSP block generates control signals and how the signals route from the row interface to the DSP block.

DSP Block Interface

The DSP blocks are organized in columns, which provides efficient horizontal communication between the blocks and the column-based memory blocks. The DSP block communicates with other parts of the device through an input and output interface. Each DSP block, including the input and output interface, is 8 logic array blocks (LABs) long.

The DSP block and row interface blocks consist of eight blocks that connect to eight adjacent LAB rows on the left and right. Each of the eight blocks has two regions: right and left, one per row. The DSP block receives 144 data input signals and 18 control signals for a total of 162 input signals. This block drives out 144 data output signals; 2 of the data signals can be used as overflow signals (`overflow`). Figure 18–6 provides an overview of the DSP block and its interface to adjacent LABs.

Figure 18–6. DSP Block Interface to Adjacent LABs



Input Interface
The DSP block input interface has 162 input signals from adjacent LABs; 18 data signals per row and 18 control signals per block.

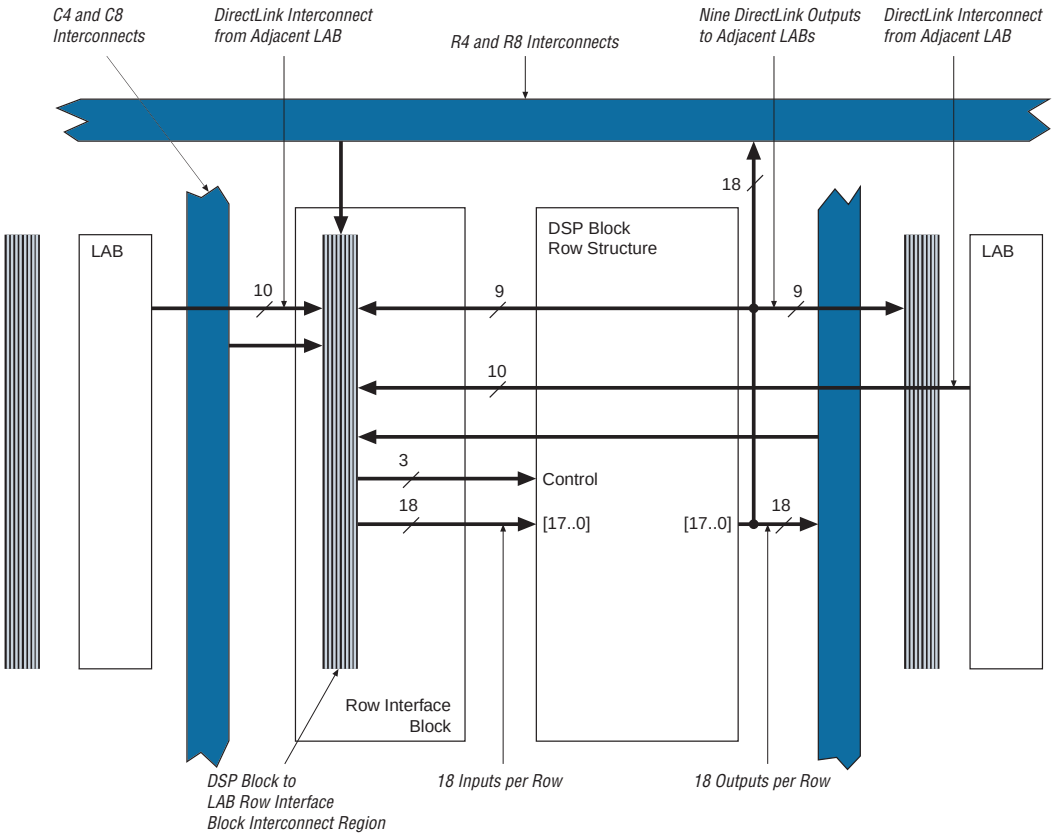
Output Interface
The DSP block output interface drives 144 outputs to adjacent LABs, 18 signals per row from 8 rows.

Because the DSP block outputs communicate horizontally, and because each DSP block row has more outputs than an LAB (18 from the DSP block compared to 10 from an LAB), the DSP block has double the number of row channel drivers compared to an LAB. The DSP block has the same number of row channels, but the row channels are staggered as if there were two LABs within each block. The DSP blocks have the same number of column channels as LABs because DSP blocks communicate primarily through row channels.

Row Interface Block

Each row interface block connects to the DSP block row structure with 21 signals. Because each DSP block has eight row interface blocks, this block receives 162 signals from the eight row interfaces. Of the 162 signals, 144 are data inputs and 18 are control signals. Figure 18–7 on page 18–14 shows one row block within the DSP block.

Figure 18–7. DSP Row Interface Block



Control Signals in the Row Interface Block

The DSP block has a set of input registers, a pipeline register, and an output register. Each register is grouped in banks that share the same clock and clear resources:

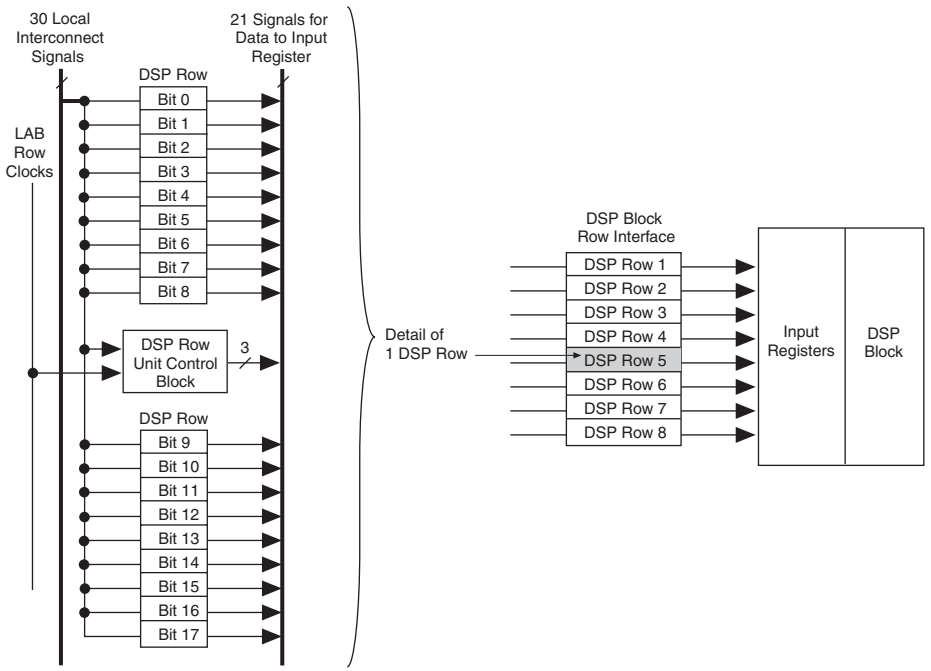
- 1- to 9-bit banks for the input register
- 1- to 18-bit banks for the pipeline register
- 18 bits for the output register

The row interface block generates the control signals and routes them to the DSP block. Each DSP block has 18 control signals:

- Four clock signals (`clock[3..0]`), which are available to each bank of DSP blocks
- Four clear signals (`aclr[3..0]`), which are available to each bank of DSP blocks
- Four clock enable signals (`ena[3..0]`), which the whole DSP block can use
- `signa` and `signb`, which are specific to each DSP block
- `addnsub[1..0]` signals
- `accum_sload[1..0]` signals

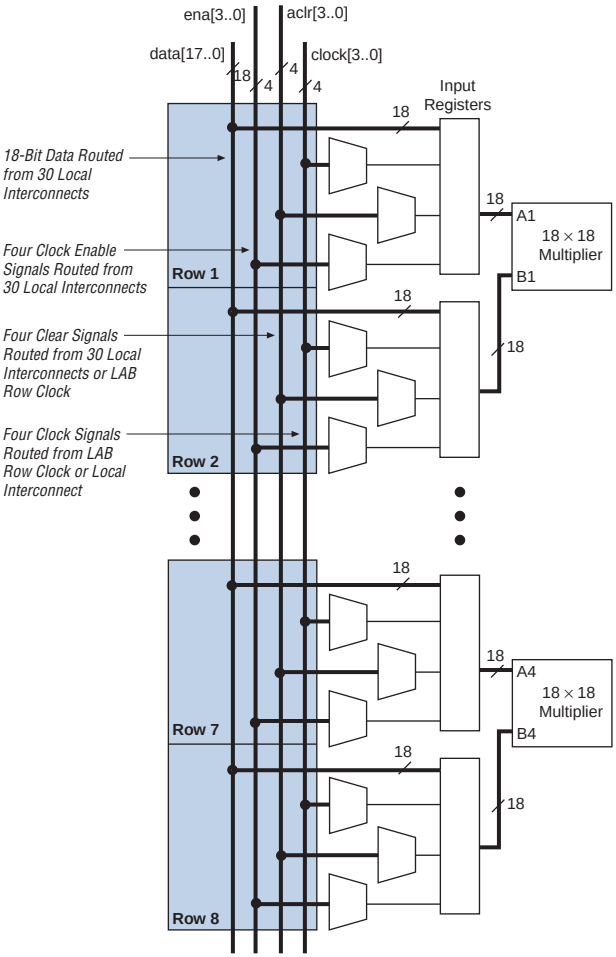
The `signa`, `signb`, and `addnsub[1..0]`, `accum_sload[1..0]` signals have independent clocks and clears and can be registered individually. When each 18 × 18 multiplier in the DSP block splits in half to two 9 × 9 multipliers, each 9 × 9 multiplier has independent control signals. Figure 18–8 shows the DSP block row interface and shows how it generates the data and control signals.

Figure 18–8. DSP Block Row Interface



The DSP block interface generates the clock signals from LAB row clocks or the local interconnect. The clear signals are generated from the local interconnects within each DSP block row interface or from LAB row clocks. The four clock enable signals are generated from the 30 local interconnects from the same LAB rows that generate the clock signals. The clock enable is paired with the clock because the enable logic is implemented at the interface. Figure 18–9 shows the signal distribution within the row interface block.

Figure 18–9. DSP Block Row Interface Signal Distribution



Each row block provides 18 bits of data to the multiplier (i.e., one of the operands to the multiplier), which are routed through the 30 local interconnects within each DSP row interface block. Any signal in the device can be the source of the 18-bit multiplier data, by connecting to the local row interconnect through any row or column.

Each control signal routes through one of the eight rows of the DSP block. Table 18–4 shows the 18 control signals and the row to which each one routes.

Table 18–4. Control Signals in DSP Block

Signal Name	Row	Description
signa	1	DSP block-wide signed and unsigned control signals for all multipliers. The multiplier outputs are unsigned only if both signa and signb are low.
signb	6	
addnsub1	3	Controls addition or subtraction of the two one-level adders. The addnsub0 signal controls the top two one-level adders; the addnsub1 signal controls the bottom two one-level adders. A high indicates addition; a low indicates subtraction.
addnsub3	7	
accum_sload0	2	Resets the feedback input to the accumulator. The signal asynchronously clears the accumulator and allows new accumulation to begin without losing any clock cycles. The accum_sload0 controls the top two one-level adders, and the accum_sload1 controls the bottom two one-level adders. A low is for normal accumulation operations and a high is for zeroing the accumulator.
accum_sload1	7	
clock0	3	DSP block-wide clock signals.
clock1	4	
clock2	5	
clock3	6	
aclr0	1	DSP block-wide clear signals.
aclr1	4	
aclr2	5	
aclr3	7	
ena[3..0]	Same rows as the Clock Signals	DSP block-wide clock enable signals.

Input/Output Data Interface Routing

The 30 local interconnects generate the 18 inputs to the row interface blocks. The 21 outputs of the row interface block are the inputs to the DSP row block (see Figure 18–7 on page 18–14).

The row interface block has DirectLink™ connections that connect the DSP block input or output signals to the left and right adjacent LABs at each row. (The DirectLink connections provide interconnects between LABs and adjacent blocks.) The DirectLink connection reduces the use of row and column interconnects, providing higher performance and flexibility.

Each row interface block receives 10 DirectLink connections from the right adjacent LABs and 10 from the left adjacent LABs. Additionally, the row interface block receives signals from the DSP block, making a total of 30 local interconnects for each row interface block. All of the row and column resources within the DSP block can access this interconnect region (see Figure 18–7 on page 18–14).

A DSP block has nine outputs that drive the right adjacent LAB and nine that drive the left adjacent LAB through DirectLink interconnects. All 18 outputs drive any row or column.

Operational Modes

You can use the DSP block in one of four operational modes, depending on your application needs (see Table 18–4). The Quartus II software has built-in megafunctions that you can use to control the mode. After you have made your parameter settings using the megafunction's MegaWizard® Plug-In, the Quartus II software automatically configures the DSP block.

Table 18–5. DSP Block Operational Modes

Mode	9 × 9	18 × 18	36 × 36
Simple multiplier	Eight multipliers with eight product outputs	Four multipliers with four product outputs	One multiplier
Multiply accumulator	Two 34-bit multiply-accumulate blocks	Two 52-bit multiply-accumulate blocks	–
Two-multiplier adder	Four two-multiplier adders	Two two-multiplier adders	–
Four-multiplier adder	Two four-multiplier adders	One four-multiplier adder	–

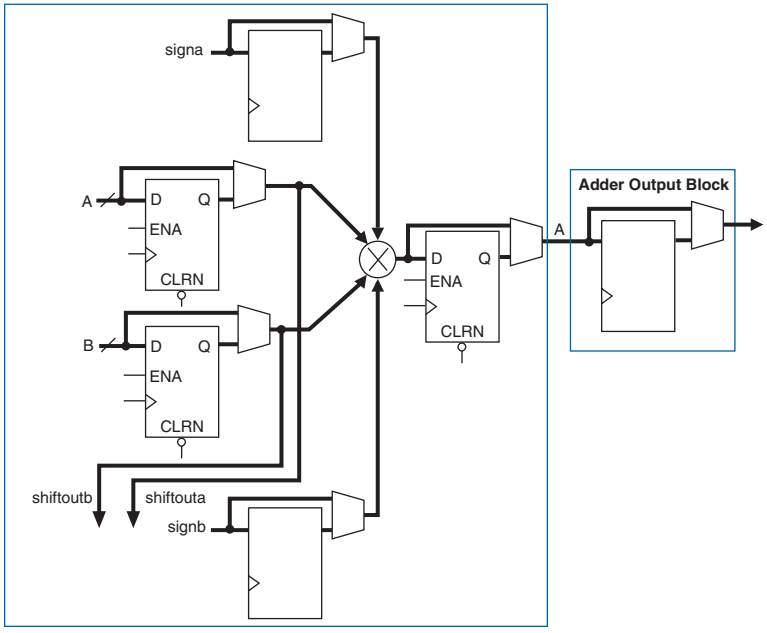
Simple Multiplier Mode

In simple multiplier mode, the DSP block performs individual multiplication operations for general-purpose multipliers and for applications such as equalizer coefficient updates that require many individual multiplication operations.

9- & 18-Bit Multipliers

You can configure each DSP block multiplier for 9 or 18 bits. A single DSP block can support up to 8 individual 9-bit or smaller multipliers, or up to 4 individual multipliers with operand widths between 10- and 18-bits. Figure 18–10 shows the simple multiplier mode.

Figure 18–10. Simple Multiplier Mode



The multiplier operands can accept signed integers, unsigned integers, or a combination. The `signa` and `signb` signals are dynamic and can be registered in the DSP block. Additionally, you can register the multiplier inputs and results independently. Pipelining the result, using the pipeline registers in the block, increases the performance of the DSP block.

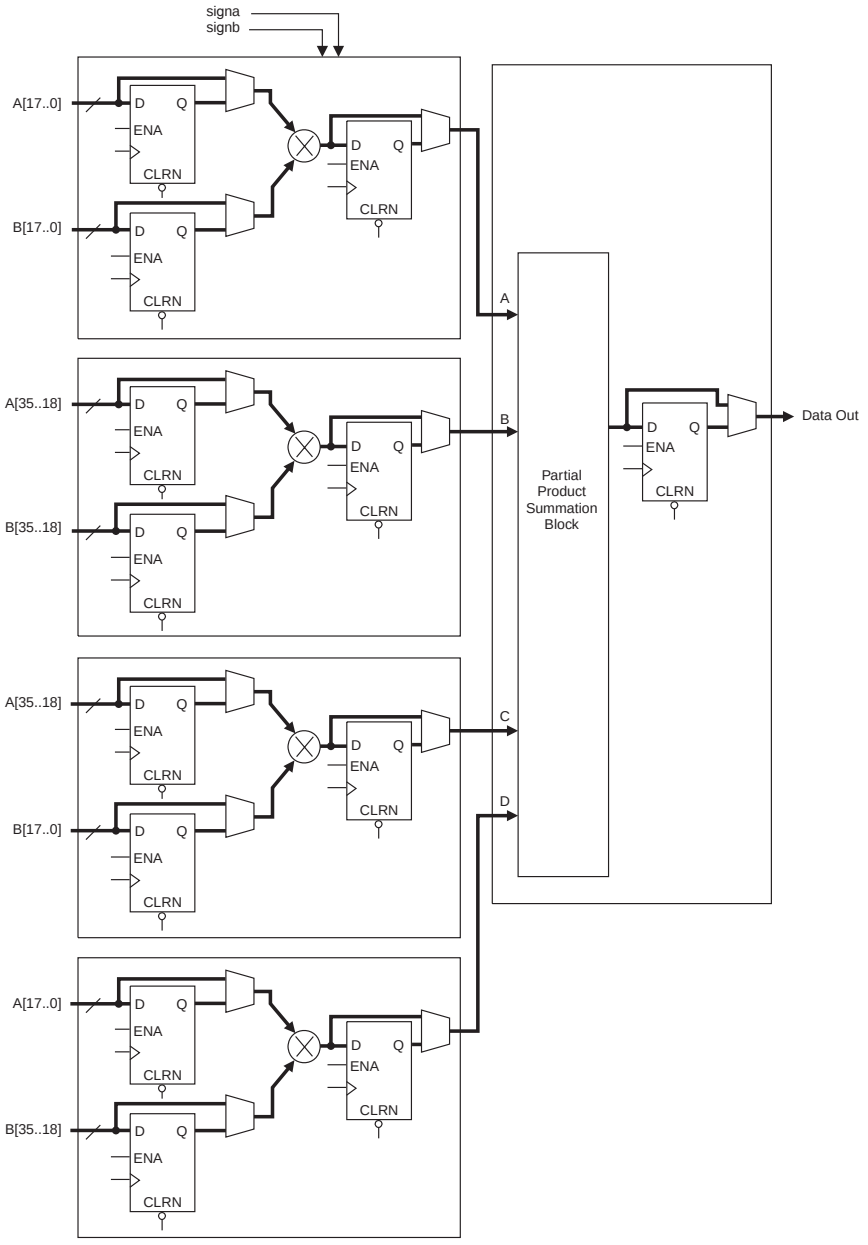
36-Bit Multiplier

The 36-bit multiplier is a subset of the simple multiplier mode. It uses the entire DSP block to implement one 36 × 36-bit multiplier. The four 18-bit multipliers are fed part of each input, as shown in Figure 18–11 on page 18–21. The adder/output block adds the partial products using the

summation block. You can use pipeline registers between the multiplier stage and the summation block. The 36 × 36-bit multiplier supports signed and unsigned operation.

The 36-bit multiplier is useful when your application needs more than 18-bit precision, for example, for mantissa multiplication of precision floating-point arithmetic applications.

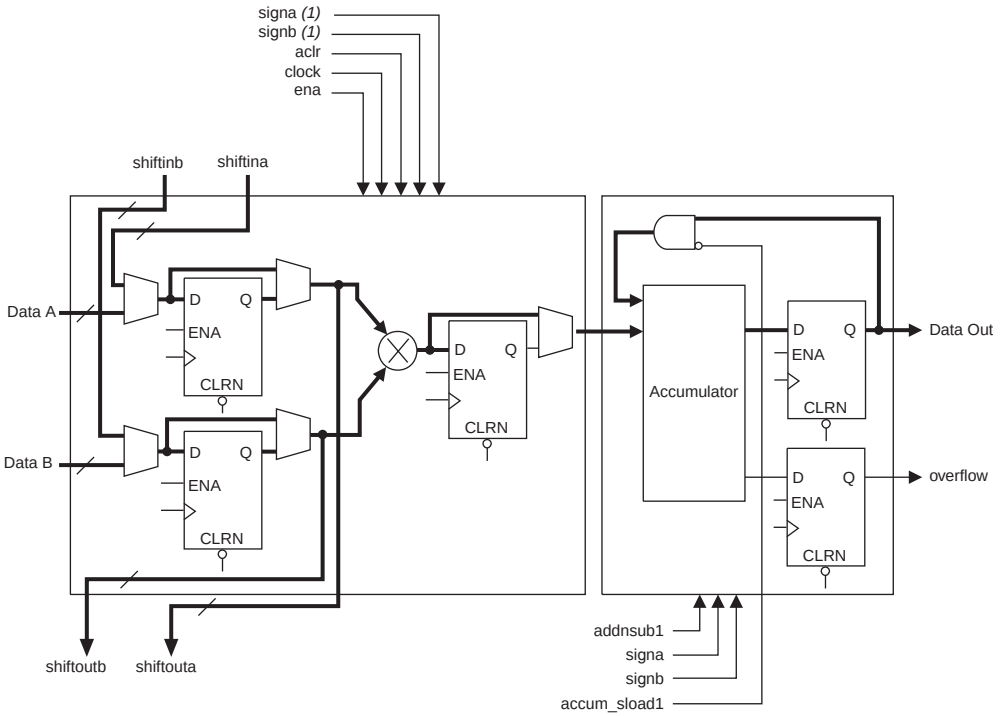
Figure 18–11. 36-Bit Multiplier



Multiply Accumulator Mode

In multiply accumulator mode, the output of the multiplier stage feeds the adder/output block, which is configured as an accumulator or subtractor (see Figure 18–12). You can implement up to two independent 18-bit multiply accumulators in one DSP block. The Quartus II software implements smaller multiplier-accumulators by tying the unused low-order bits of an 18-bit multiplier to ground.

Figure 18–12. Multiply Accumulator Mode



Note to Figure 18–12:
(1) The `signa` and `signb` signals are the same in the multiplier stage and the adder/output block.

The multiply accumulator output can be up to 52 bits wide for a maximum 36-bit result with 16-bits of accumulation. In this mode, the DSP block uses output registers and the `accum_sload` and `overflow` signals. The `accum_sload[1..0]` signal synchronously loads the multiplier result to the accumulator output. This signal can be unregistered or registered once or twice. The DSP block can then begin a new accumulation without losing any clock cycles. The `overflow` signal indicates an overflow or underflow in the accumulator. This signal is

cleared for the next accumulation cycle, and you can use an external latch to preserve the signal. You can use the `addnsub[1..0]` signals to perform accumulation or subtraction dynamically.



If you want to use DSP blocks and your design only has an accumulator, you can use a multiply by one followed by an accumulator to force the software to implement the logic in the DSP block.

Two-Multiplier Adder Mode

The two-multiplier adder mode uses the adder/output block to add or subtract the outputs of the multiplier block, which is useful for applications such as FFT functions and complex FIR filters. Additionally, in this mode, the DSP block outputs two sums or differences for multipliers up to 18 bits, or 4 sums or differences for 9-bit or smaller multipliers. A single DSP block can implement one 18 × 18-bit complex multiplier or two 9 × 9-bit complex multipliers.

A complex multiplication can be written as:

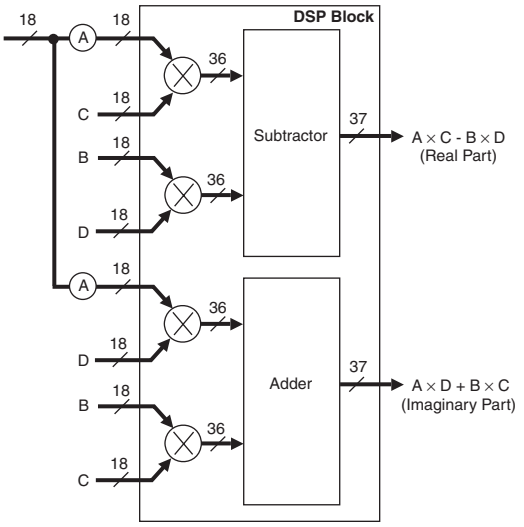
$$(a + jb) \times (c + jd) = (a \times c - b \times d) + j \times (a \times d + b \times c)$$

In this mode, a single DSP block calculates the real part ($a \times c - b \times d$) using one adder/subtractor/accumulator and the imaginary part ($a \times d + b \times c$) using another adder/subtractor/accumulator for data up to 18 bits. Figure 18–13 shows an 18-bit complex multiplication. For data widths up to 9 bits, the DSP block can perform two complex multiplications using four one-level adders. Resources outside of the DSP block route each input to the two multiplier inputs.



You can only use the adder block if it follows multiplication operations.

Figure 18–13. Complex Multiplier Implemented Using Two-Multiplier Adder Mode



Four-Multiplier Adder Mode

In the four-multiplier adder mode, which you can use for 1-dimensional and 2-dimensional filtering applications, the DSP block adds the results of two adder/subtractor/accumulators in a final stage (the summation block).

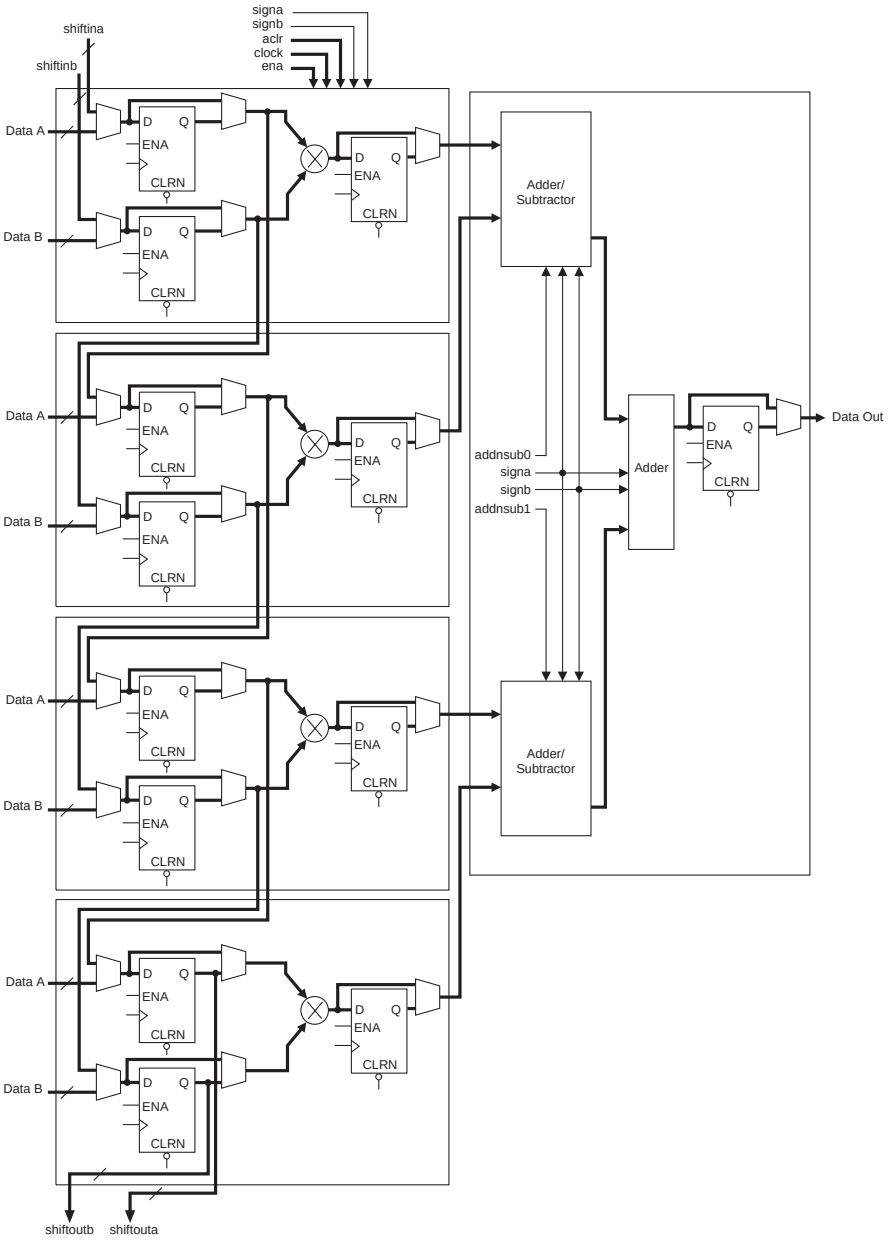


You can only use the adder block if it follows multiplication operations.

9- & 18-Bit Summation Blocks

A single DSP block can implement one 18 × 18 or two 9 × 9 summation blocks (see Figure 18–14 on page 18–25). The multiplier product widths must be the same size.

Figure 18–14. Four-Multiplier Adder Mode



FIR Filters

The four-multiplier adder mode can be used for FIR filter and complex FIR filter applications. The DSP block combines a four-multiplier adder with the input registers configured as shift registers. One set of shift inputs contains the filter data, while the other holds the coefficients, which can be loaded serially or in parallel (see Figure 18–15).

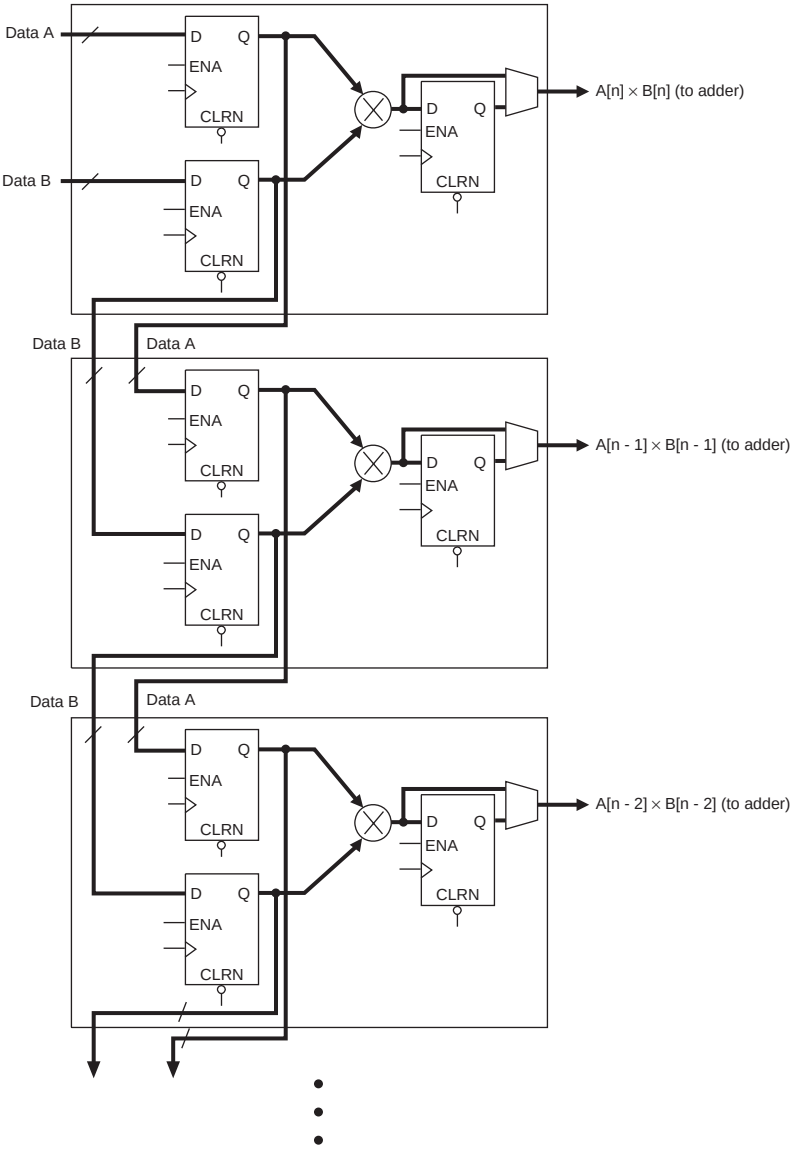
The input shift register eliminates the need for shift registers external to the DSP block (e.g., implemented in device logic elements). This architecture simplifies filter design and improves performance because the DSP block implements all of the filter circuitry.



Serial shift inputs in 36-bit simple multiplier mode require external registers.

One DSP block can implement an entire 18-bit FIR filter with up to four taps. For FIR filters larger than four taps, you can cascade DSP blocks with additional adder stages implemented in logic elements.

Figure 18–15. Input Shift Registers Configured for a FIR Filter



Software Support

Altera provides two distinct methods for implementing various modes of the DSP block in your design: instantiation and inference. Both methods use the following three Quartus II megafunctions:

- lpm_mult
- altmult_add
- altmult_accum

You can instantiate the megafunctions in the Quartus II software to use the DSP block. Alternatively, with inference, you can create a HDL design an synthesize it using a third-party synthesis tool like LeonardoSpectrum or Synplify or Quartus II Native Synthesis that infers the appropriate megafunction by recognizing multipliers, multiplier adders, and multiplier accumulators. Using either method, the Quartus II software maps the functionality to the DSP blocks during compilation.



See the *Implementing High-Performance DSP Functions in Stratix & Stratix GX Devices* chapter in the *Stratix Device Handbook, Volume 2* or the *Stratix GX Device Handbook, Volume 2* for more information on using DSP blocks to implement high-performance DSP functions such as FIR filters, IIR filters, and discreet cosine transforms (DCTs).



See Quartus II On-Line Help for instructions on using the megafunctions and the MegaWizard Plug-In Manager.



For more information on DSP block inference support, see the *Recommended HDL Coding Styles* chapter of the *Quartus II Development Software Handbook v4.1, Volume 1*.

Conclusion

The Stratix and Stratix GX device DSP blocks are optimized to support DSP applications that need high data throughput, such as FIR filters, FFT functions, and encoders. These DSP blocks are flexible and can be configured in one of four operational modes to suit any application need. The DSP block's adder/subtractor/accumulator and the summation blocks minimize the amount of logic resources used and provide efficient routing. This efficiency results in improved performance and data throughput for DSP applications. The Quartus II software, together with the LeonardoSpectrum and Synplify software, provides a complete and easy-to-use flow for implementing functionality in the DSP block.

19. Implementing High Performance DSP Functions in Stratix & Stratix GX Devices

S52007-1.1

Introduction

Digital signal processing (DSP) is a rapidly advancing field. With products increasing in complexity, designers face the challenge of selecting a solution with both flexibility and high performance that can meet fast time-to-market requirements. DSP processors offer flexibility, but they lack real-time performance, while application-specific standard products (ASSPs) and application-specific integrated circuits (ASICs) offer performance, but they are inflexible. Only programmable logic devices (PLDs) offer both flexibility and high performance to meet advanced design challenges.

The mathematical theory underlying basic DSP building blocks—such as the finite impulse response (FIR) filter, infinite impulse response (IIR) filter, fast fourier transform (FFT), and direct cosine transform (DCT)—is computationally intensive. Altera® Stratix® and Stratix GX devices feature dedicated DSP blocks optimized for implementing arithmetic operations, such as multiply, multiply-add, and multiply-accumulate.

In addition to DSP blocks, Stratix and Stratix GX devices have TriMatrix™ embedded memory blocks that feature various sizes that can be used for data buffering, which is important for most DSP applications. These dedicated hardware features make Stratix and Stratix GX devices an ideal DSP solution.

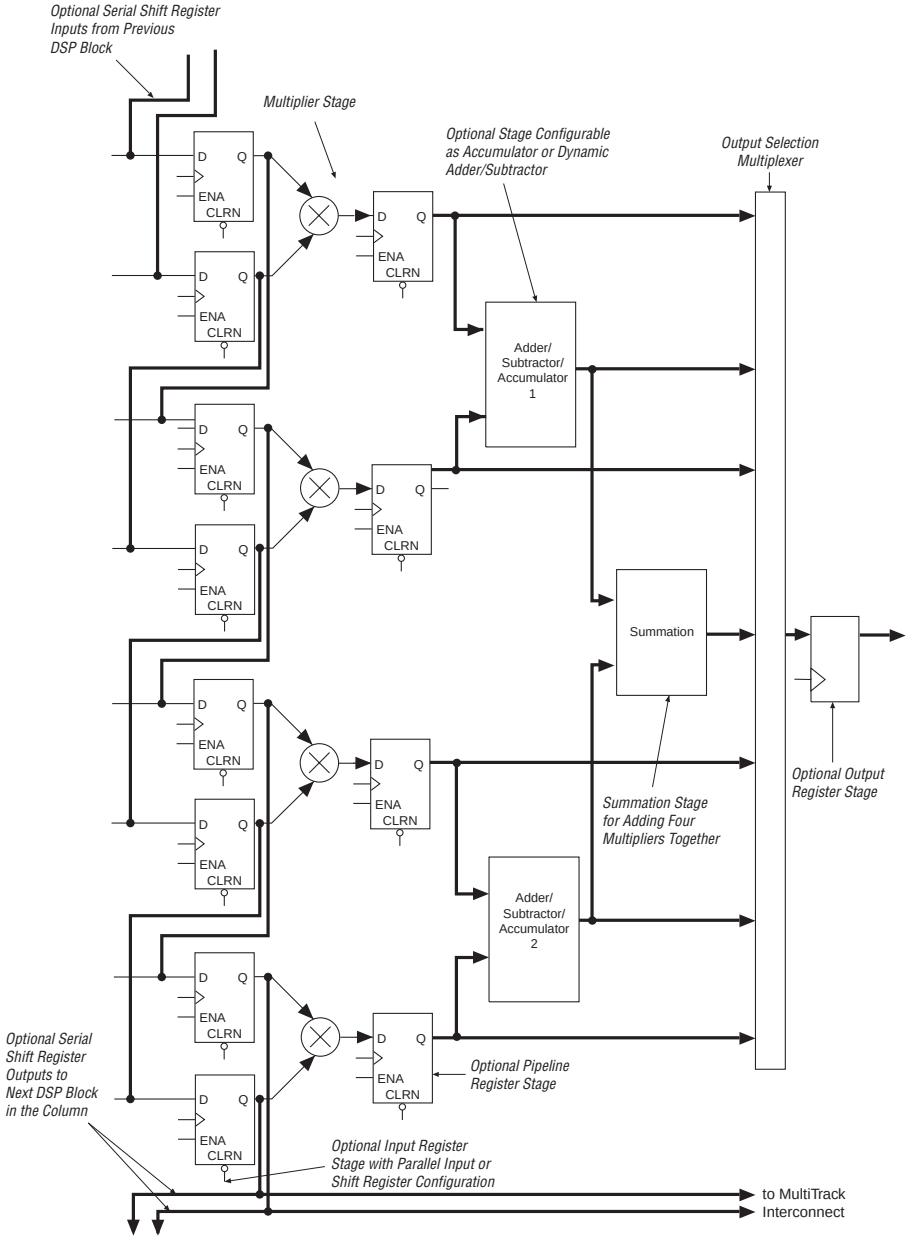
This chapter describes the implementation of high performance DSP functions, including filters, transforms, and arithmetic functions, using Stratix and Stratix GX DSP blocks. The following topics are discussed:

- FIR filters
- IIR filters
- Matrix manipulation
- Discrete Cosine Transform
- Arithmetic functions

Stratix & Stratix GX DSP Block Overview

Stratix and Stratix GX devices feature DSP blocks that can efficiently implement DSP functions, including multiply, multiply-add, and multiply-accumulate. The DSP blocks also have three built-in registers sets: the input registers, the pipeline registers at the multiplier output, and the output registers. Figure 19–1 shows the DSP block operating in the 18 × 18-bit mode.

Figure 19–1. DSP Block Diagram for 18 x 18-bit Mode



The DSP blocks are organized into columns enabling efficient horizontal communication with adjacent TriMatrix memory blocks. Tables 19–1 and 19–2 show the DSP block resources in Stratix and Stratix GX devices, respectively.

Table 19–1. DSP Block Resources in Stratix Devices

Device	DSP Blocks	Maximum 9 × 9 Multipliers	Maximum 18 × 18 Multipliers	Maximum 36 × 36 Multipliers
EP1S10	6	48	24	6
EP1S20	10	80	40	10
EP1S25	10	80	40	10
EP1S30	12	96	48	12
EP1S40	14	112	56	14
EP1S60	18	144	72	18
EP1S80	22	176	88	22

Table 19–2. DSP Block Resources in Stratix GX Devices

Device	DSP Blocks	Maximum 9 × 9 Multipliers	Maximum 18 × 18 Multipliers	Maximum 36 × 36 Multipliers
EP1SGX10C	6	48	24	6
EP1SGX10D	6	48	24	6
EP1SGX25C	10	80	40	10
EP1SGX25D	10	80	40	10
EP1SGX25F	10	80	40	10
EP1SGX40D	14	112	56	14
EP1SGX40G	14	112	56	14

Each DSP block supports either eight 9 × 9-bit multipliers, four 18-bit multipliers, or one 36 × 36-bit multiplier. These multipliers can feed an adder or an accumulator unit based on the operation mode. Table 19–3 shows the different operation modes for the DSP blocks.

Table 19–3. Operation Modes for DSP Blocks

DSP Block Mode	Number & Size of Multipliers per DSP Block		
	9 x 9-bit	18 x 18-bit	36 x 36-bit
Simple multiplier	Eight multipliers with eight product outputs	Four multipliers with four product outputs	One multiplier with one product output
Multiply-accumulate	Two multiply and accumulate (34 bit)	Two multiply and accumulate (52 bit)	
Two-multipliers adder	4 two-multipliers adders	2 two-multipliers adders	
Four-multipliers adder	2 four-multipliers adder	1 four-multipliers adder	

Implementing multipliers, multiply-adders, and multiply-accumulators in the DSP blocks has a performance advantage over logic cell implementation. Using DSP blocks also reduces logic cell and routing resource consumption. To achieve higher performance, register each stage of the DSP block to allow pipelining. For implementing applications, such as FIR filters, efficiently use the input registers of the DSP block as shift registers.



For more information on DSP blocks, see the *DSP Blocks in Stratix & Stratix GX Devices* chapter.

TriMatrix Memory Overview

Stratix and Stratix GX devices feature the TriMatrix memory structure, composed of three sizes of embedded RAM blocks. These include the 512-bit size M512 block, the 4-Kbit size M4K block, and the 512-Kbit size M-RAM block. Each block is configurable to support a wide range of features.

Tables 19–4 and 19–5 show the number of memory blocks in each Stratix and Stratix GX device, respectively.

Table 19–4. TriMatrix Memory Resources in Stratix Devices (Part 1 of 2)

Device	M512	M4K	M-RAM
EP1S10	94	60	1
EP1S20	194	82	2
EP1S25	224	138	2

Table 19–4. TriMatrix Memory Resources in Stratix Devices (Part 2 of 2)

Device	M512	M4K	M-RAM
EP1S30	295	171	4
EP1S40	384	183	4
EP1S60	574	292	6
EP1S80	767	364	9

Table 19–5. TriMatrix Memory Resources in Stratix GX Devices

Device	M512	M4K	M-RAM
EP1SGX10C	94	60	1
EP1SGX10D	94	60	1
EP1SGX25C	224	138	2
EP1SGX25D	224	138	2
EP1SGX25F	224	138	2
EP1SGX40D	384	183	4
EP1SGX40G	384	183	4

Most DSP applications require local data storage for intermediate buffering or for filter storage. The TriMatrix memory blocks enable efficient use of available resources for each application.

The M512 and M4K memory blocks can implement shift registers for applications, such as multi-channel filtering, auto-correlation, and cross-correlation functions. Implementing shift registers in embedded memory blocks reduces logic cell and routing resource consumption.



For more information on TriMatrix memory blocks, see the *TriMatrix Embedded Memory Blocks in Stratix & Stratix GX Devices* chapter.

DSP Function Overview

The following sections describe commonly used DSP functions. Each section illustrates the implementation of a basic DSP building block, including FIR and IIR filters, in Stratix and Stratix GX devices using DSP blocks and TriMatrix memory blocks.

Finite Impulse Response (FIR) Filters

This section describes the basic theory and implementation of basic FIR filters, time-domain multiplexed (TDM) FIR filters, and interpolation and decimation polyphase FIR filters. An introduction to the complex FIR filter is also presented in this section.

FIR Filter Background

Digital communications systems use FIR filters for a variety of functions, including waveform shaping, anti-aliasing, band selection, decimation/interpolation, and low pass filtering. The basic structure of a FIR filter consists of a series of multiplications followed by an addition.

The following equation represents an FIR filter operation:

$$y(n) = x(n) \otimes n(n)$$

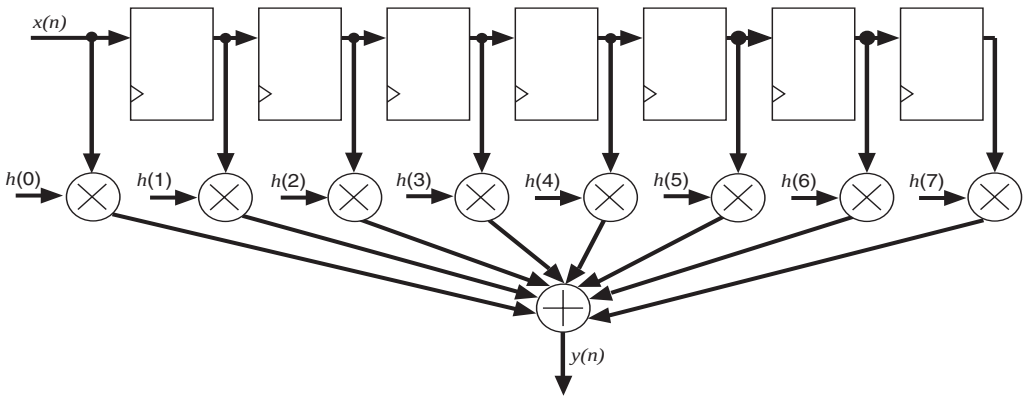
$$y(n) = \sum_{i=0}^{L-1} x(n-i)h(i)$$

where:

$x(n)$ represents the sequence of input samples
$h(n)$ represents the filter coefficients
L is the number of filter taps

A sample FIR filter with L=8 is shown in Figure 19–2.

Figure 19–2. Basic FIR Filter



This example filter in Figure 19–2 uses the input values at eight different time instants to produce an output. Hence, it is an 8-tap filter. Each register provides a unit sample delay. The delayed inputs are multiplied with their respective filter coefficients and added together to produce the output. The width of the output bus depends on the number of taps and the bit width of the input and coefficients.

Basic FIR Filter

A basic FIR filter is the simplest FIR filter type. As shown in Figure 19–2, a basic FIR filter has a single input channel and a single output channel.

Basic FIR Filter Implementation

Stratix and Stratix GX devices' dedicated DSP blocks can implement basic FIR filters. Because these DSP blocks have closely integrated multipliers and adders, filters can be implemented with minimal routing resources and delays. For implementing FIR filters, the DSP blocks are configured in the four-multipliers adder mode.

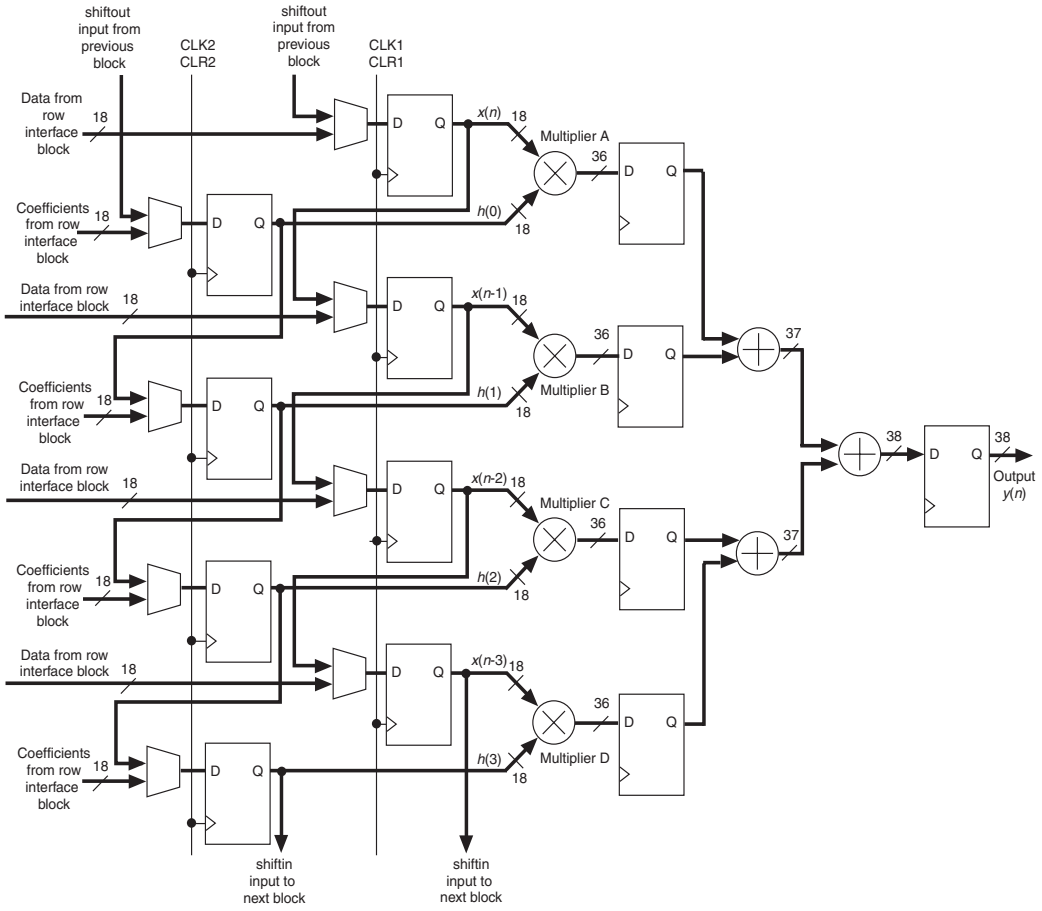


See the *DSP Blocks in Stratix & Stratix GX Devices* chapter for more information on the different modes of the DSP blocks.

This section describes the implementation of an 18-bit 8-tap FIR filter. Because Stratix and Stratix GX devices support modularity, cascading two 4-tap filters can implement an 8-tap filter. Larger FIR filters can be designed by extending this concept. Users can also increase the number of taps available per DSP block if 18 bits of resolution are not required. For example, by using only 9 bits of resolution for input samples and coefficient values, 8 multipliers are available per DSP block. Therefore, a 9-bit 8-tap filter can be implemented in a single DSP block provided an external adder is implemented in logic cells.

The four-multipliers adder mode, shown in Figure 19–3, provides four 18 × 18-bit multipliers and three adders in a single DSP block. Hence, it can implement a 4-tap filter. The data width of the input and the coefficients is 18 bits, which results in a 38-bit output for a 4-tap filter.

Figure 19–3. Hardware View of a DSP Block in Four-Multipliers Adder Mode Notes (1), (2), (3)



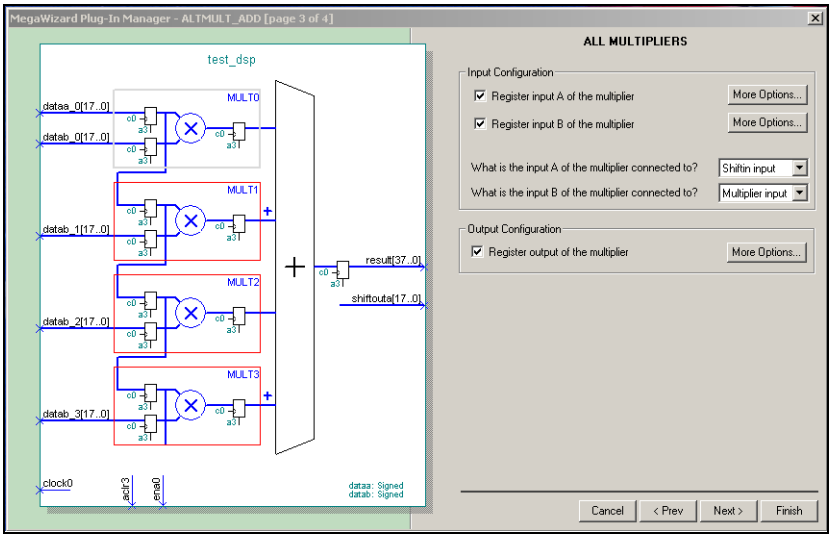
Notes to Figure 19–3:

(1) The input registers feed the multiplier blocks. These registers can increase the DSP block performance, but are optional. These registers can also function as shift registers if the dedicated shiftin/shiftout signals are used.

(2) The pipeline registers are fed by the multiplier blocks. These registers can increase the DSP block performance, but are optional.

(3) The output registers register the DSP block output. These registers can increase the DSP block performance, but are optional.

Figure 19–4. Quartus II Software View of MegaWizard Implementation of a DSP Block in Four-Multipliers Adder Mode



Each input register of the DSP block provides a shiftout output that connects to the shiftin input of the adjacent input register of the same DSP block. The registers on the boundaries of a DSP block also connect to the registers of adjacent DSP blocks through the use of shiftin/shiftout connections. These connections create register chains spanning multiple DSP blocks, which makes it easy to increase the length of FIR filters. Figure 19–5 shows two DSP blocks connected to create an 8-tap FIR filter. Filters with more taps can be implemented by connecting DSP blocks in a similar manner, provided sufficient DSP blocks are available in the device.



Adding the outputs of the two DSP blocks requires an external adder which can be implemented using logic cells.

The input data can be fed directly or by using the shiftout/shiftin chains, which allow a single input to shift down the register chain inside the DSP block. The input to each of the registers has a multiplexer, hence, the data can be fed either from outside the DSP block or the preceding register. This can be selected from the MegaWizard® in the Quartus® II software, as shown in Figure 19–4. The example in Figure 19–5 uses the shiftout/shiftin flip-flop chains where the multiplexers are configured to use these chains. In this example, the flip-flops inside the DSP blocks serve as the taps of the FIR filter.

When the coefficients are loaded in parallel, they can be fed directly from memory elements or any other muxing scheme. This facilitates the implementation of an adaptive (variable) filter.

Further, if the user wants to implement the shift register chains external to the DSP block, this can be done by using the `altshift_taps` megafunction. In this case, the coefficient and data shifting is done external to the DSP block. The DSP block is only used to implement the multiplications and the additions.

Parallel vs. Serial Implementation

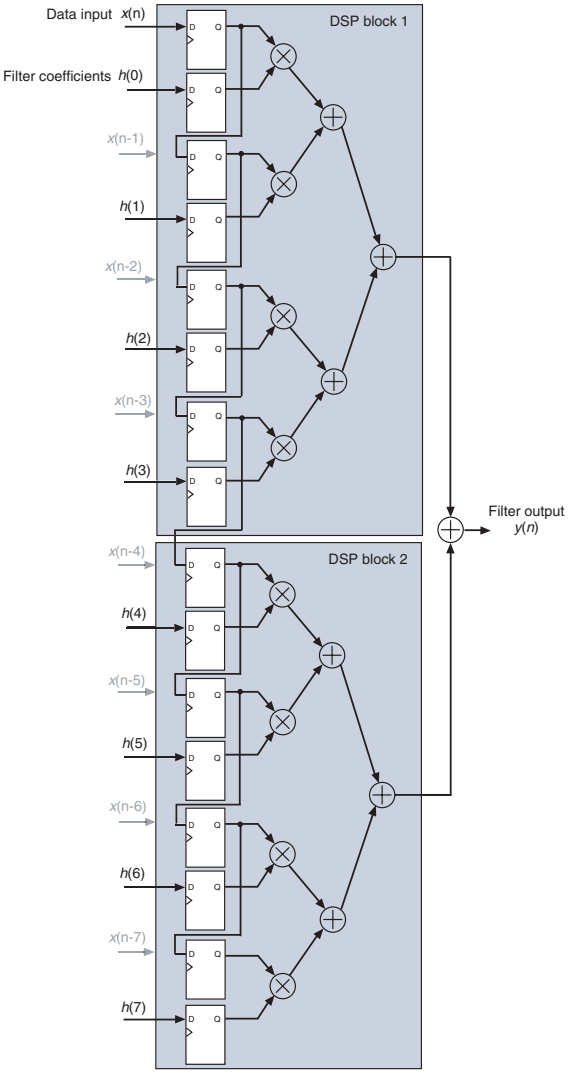
The fastest implementations are fully parallel, but consume more logic resources than serial implementations. To trade-off performance for logic resources, implement a serial scheme with a specified number of taps. To facilitate this, Altera provides the FIR Compiler core through its MegaCore program. The FIR Compiler is an easy-to-use, fully-integrated graphical user interface (GUI) based FIR filter design software.



For more information on the FIR Compiler MegaCore, visit the Altera web site at **www.altera.com** and search for "FIR compiler" in the "Intellectual Property" page.

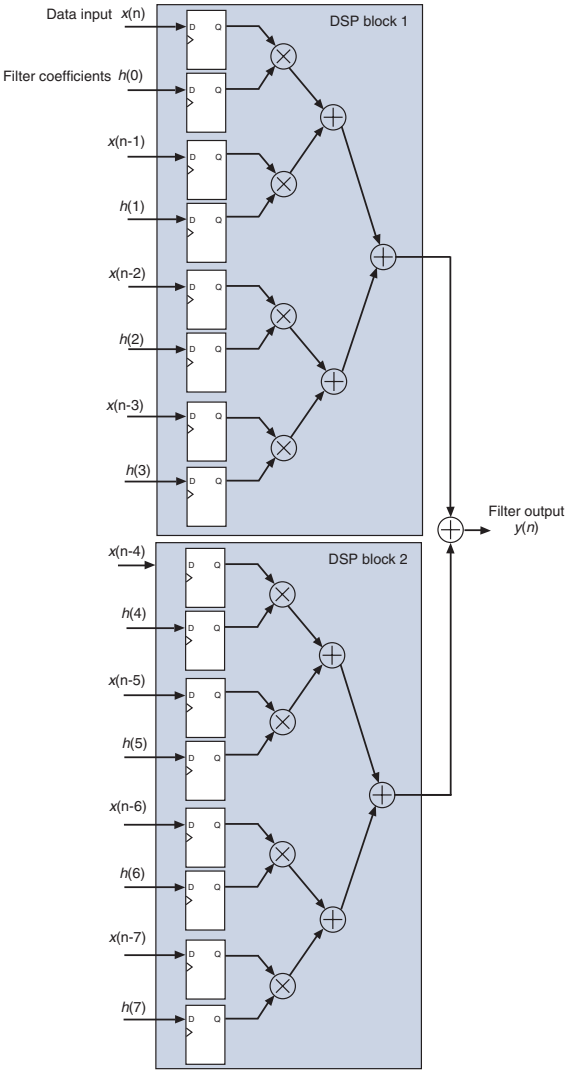
It is important to note that the four-multipliers adder mode allows a DSP block to be configured for parallel or serial input. When it is configured for parallel input, as shown in Figure 19–6, the data input and the coefficients can be loaded directly without the need for shiftin/shiftout chains between adjacent registers in the DSP block. When the DSP block is configured for serial input, as shown in Figure 19–5, the shiftin/shiftout chains create a register cascade both within the DSP block and also between adjacent DSP blocks.

Figure 19–5. Serial Loading 18-Bit 8-Tap FIR Filter Using Two DSP Blocks
Notes (1), (2), (3)



Notes to Figure 19–5:
(1) Unused ports grayed out.
(2) The indexing $x(n\text{-}1)$, ..., $x(n\text{-}7)$ refers to the case of parallel loading and should be ignored here. This indexing is retained in this figure for consistency with other figures in this chapter.
(3) To increase the DSP block performance, include the pipeline and output registers. See Figure 19–3 on page 19–8 for the details.

Figure 19–6. Parallel Loading 18-Bit 8-Tap FIR Filter Using Two DSP Blocks
Notes (1), (2)



Notes to Figure 19–6:
(1) The indexing $x(n\text{-}1)$, ..., $x(n\text{-}7)$ refers to the case of parallel loading.
(2) To increase the DSP block performance, include the input, pipeline, and output registers. See Figure 19–3 on page 19–8 for the details.

Basic FIR Filter Implementation Results

Table 19–6 shows the results of the serial implementation of an 18-bit 8 tap FIR filter as shown in Figure 19–5 on page 19–11

Table 19–6. Basic FIR Filter Implementation Results

Part	EP1S10F780
Utilization	LCELL: 130/10570 (1%) DSP Block 9-bit elements: 16/48 (33%) Memory bits: 288/920448 (<1%)
Performance	247 MHz

Basic FIR Filter Design Example

Download the Basic FIR Filter (**base_fir.zip**) design example from the Design Examples section of the Altera web site at **www.altera.com.**

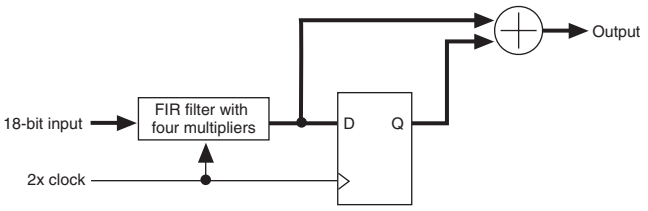
Time-Domain Multiplexed FIR Filters

A TDM FIR filter is clocked n-times faster than the sample rate in order to reuse the same hardware. Consider the 8-tap filter shown in Figure 19–2. The TDM technique can be used with a TDM factor of 2, i.e., n = 2, to implement this filter using only four multipliers, provided the filter is clocked two times faster internally.

To understand this concept, consider Figure 19–7 that shows a TDM filter with a TDM factor of 2. A 2× -multiplied clock is required to run the filter. On cycle 0 of the 2× clock, the user loads four coefficients into the four multiplier inputs. The resulting output is stored in a register. On cycle 1 of the 2× clock, the user loads the remaining four coefficients into the multiplier inputs. The output of cycle 1 is added with the output of cycle 0 to create the overall output. See the "TDM Filter Implementation" on page 19–14 section for details on the coefficient loading schedule.

The TDM implementation shown in Figure 19–7 requires only four multipliers to achieve the functionality of an 8-tap filter. Thus, TDM is a good way to save logic resources, provided the multipliers can run at n-times the clock speed. The coefficients can be stored in ROM/RAM, or any other muxing scheme.

Figure 19–7. Block Diagram of 8-Tap FIR Filter with TDM Factor of n=2



TDM Filter Implementation

TDM FIR filters are implemented in Stratix and Stratix GX devices by configuring the DSP blocks in the multiplier-adder mode. Figure 19–9 shows the implementation of an 8-tap TDM FIR filter (n=2) with 18 bits of data and coefficient inputs. Because the input data needs to be loaded into the DSP block in parallel, a shift register chain is implemented using a combination of logic cells and the altshift_taps function. This shift register is clocked with the same data sample rate (clock 1×). The filter coefficients are stored in ROM and loaded into the DSP block in parallel as well. Because the TDM factor is 2, both the ROM and DSP block are clocked with clock 2×.

Table 19–7. Operation of TDM Filter (Shown in Figure 19–9 on page 19–16)

Cycle of 2× Clock	Cycle Output	Operation	Overall Output, y(n)
0	$y_0 = x(n\text{-}1)h(1) + x(n\text{-}3)h(3) + x(n\text{-}5)h(5) + x(n\text{-}7)h(7)$	Store result	N/A
1	$y_1 = x(n)h(0) + x(n\text{-}2)h(2) + x(n\text{-}4)h(4) + x(n\text{-}6)h(6)$	Generate output	$y(n) = y_0 + y_1$
2	$y_2 = x(n)h(1) + x(n\text{-}2)h(3) + x(n\text{-}4)h(5) + x(n\text{-}6)h(7)$	Store result	N/A
3	$y_3 = x(n\text{+}1)h(0) + x(n\text{-}1)h(2) + x(n\text{-}3)h(4) + x(n\text{-}5)h(6)$	Generate output	$y(n) = y_2 + y_3$
4	$y_4 = x(n\text{+}1)h(1) + x(n\text{-}1)h(3) + x(n\text{-}3)h(5) + x(n\text{-}5)h(7)$	Store result	N/A
5	$y_5 = x(n\text{+}2)h(0) + x(n)h(2) + x(n\text{-}2)h(4) + x(n\text{-}4)h(6)$	Generate output	$y(n) = y_4 + y_5$
6	$y_6 = x(n\text{+}2)h(1) + x(n)h(3) + x(n\text{-}2)h(5) + x(n\text{-}4)h(7)$	Store result	N/A
7	$y_7 = x(n\text{+}3)h(0) + x(n\text{+}1)h(2) + x(n\text{-}1)h(4) + x(n\text{-}3)h(6)$	Generate output	$y(n) = y_6 + y_7$

Figure 19–8 and Table 19–7 show the coefficient loading schedule. For example, during cycle 0, only the flip-flops corresponding to $h(1)$, $h(3)$, $h(5)$, and $h(7)$ are enabled. This produces the temporary output, y_0, which is stored in a flip-flop outside the DSP block. During cycle 1, only the flip-

flops corresponding to $h(0)$, $h(2)$, $h(4)$ and $h(6)$ are enabled. This produces the temporary output, y_1, which is added to y_0 to produce the overall output, $y(n)$. The following shows what the overall output, $y(n)$, equals:

$$y(n) = y_0 + y_1$$

$$y(n) = x(0)h(0) + x(n-1)h(1) + x(n-2)h(2) + x(n-3)h(3) + x(n-4)h(4) + x(n-5)h(5) + x(n-6)h(6) + x(n-7)h(7)$$

This is identical to the output of the 8-tap filter shown in Figure 19–2. After cycle 1, this process is repeated at every cycle.

Figure 19–8. Coefficient Loading Schedule in a TDM Filter

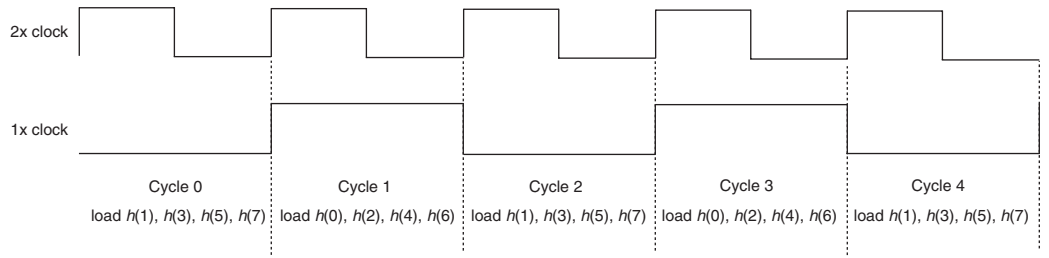
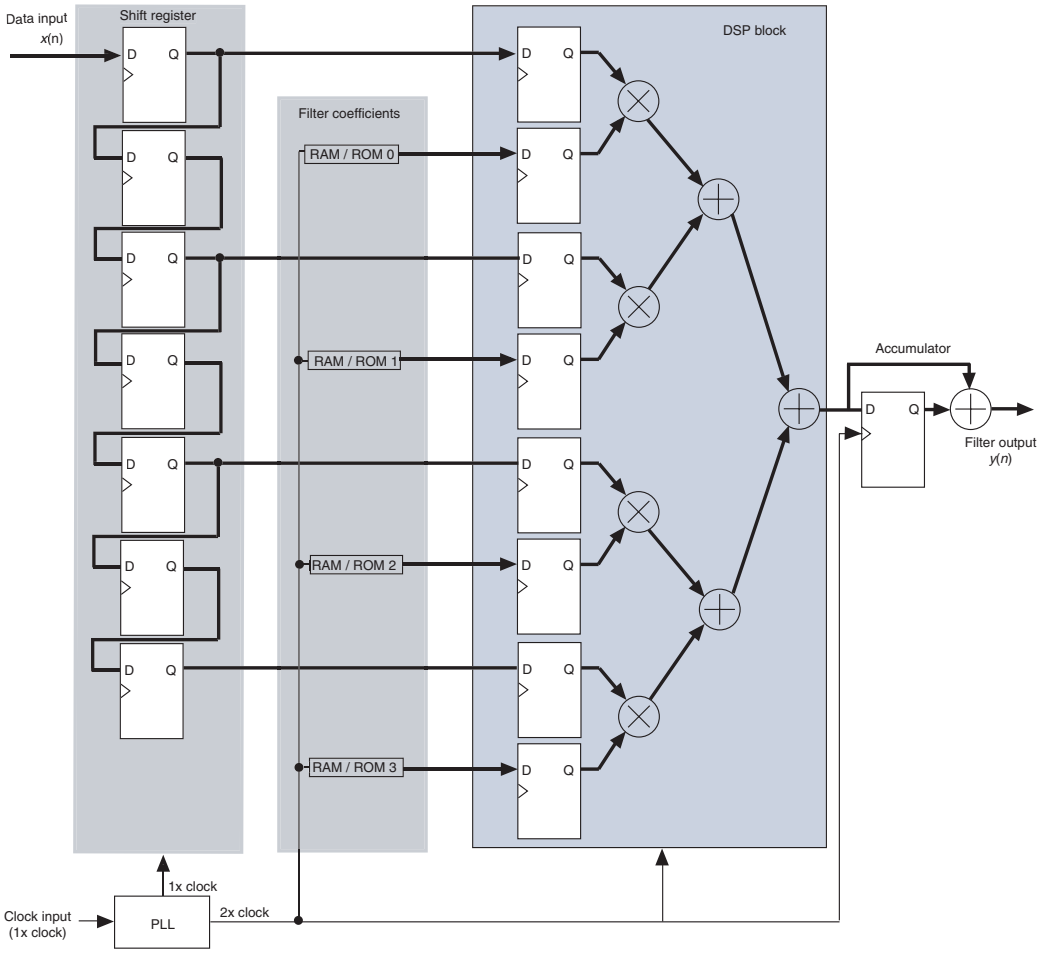


Figure 19–9. TDM FIR Filter Implementation Note (1)



Note to Figure 19–9:

(1) To increase the DSP block performance, include the pipeline and output registers. See Figure 19–3 on page 19–8 for details.

If the TDM factor is more than 2, then a multiply-accumulator needs to be implemented. This multiply-accumulator can be implemented using the soft logic outside the DSP block if all the multipliers of the DSP block are needed. Alternatively, the multiply-accumulator may be implemented inside the DSP block if all the multipliers of the DSP block are not needed. The accumulator needs to be zeroed at the start of each new sample input. The user also needs a way to store additional sample inputs in memory. For example, consider a sample rate of r and TDM factor of 4. Then, the

user needs a way to accept this sample data and send it at a 4r rate to the input of the DSP block. One way to do this is using a first-in-first-out (FIFO) memory with input clocked at rate r and output clocked at rate 4r. The FIFO may be implemented in the TriMatrix memory.

TDM Filter Implementation Results

Table 19–8 shows the results of the implementation of an 18-bit 8-tap TDM FIR filter as shown in Figure 19–9 on page 19–16.

Table 19–8. TDM Filter Implementation Results

Part	EP1S10F780
Utilization	Lcell: 196/10570 (1%) DSP Block 9-bit elements: 8/48 (17%) Memory bits: 360/920448 (<1%)
Performance	240 MHz *(1)*

Note to Table 19–8:

(1) This refers to the performance of the DSP blocks. The input and output rate is 120 million samples per second (MSPS), clocked in and out at 120 MHz.

TDM Filter Design Example

Download the TDM FIR Filter (**tdm_fir.zip**) design example from the Design Examples section of the Altera web site at **www.altera.com.**

Polyphase FIR Interpolation Filters

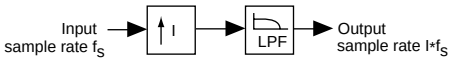
An interpolation filter can be used to increase sample rate. An interpolation filter is efficiently implemented with a polyphase FIR filter. DSP systems frequently use polyphase filters because they simplify overall system design and also reduce the number of computations per cycle required of the hardware. This section first describes interpolation filters and then how to implement them as polyphase filters in Stratix and Stratix GX devices. See the "Polyphase FIR Decimation Filters" on page 19–24 section for a discussion of decimation filters.

Interpolation Filter Basics

An interpolation filter increases the output sample rate by a factor of `I` through the insertion if `I`-1 zeros between input samples, a process known as zero padding. After the zero padding, the output samples in time domain are separated by $T_s/I = 1/(I \times f_s)$, where T_s and f_s are the sample period and sample frequency of the original signal, respectively. Figure 19–10 shows the concept of signal interpolation.

Inserting zeros between the samples creates reflections of the original spectrum, thus, a low pass filter is needed to filter out the reflections.

Figure 19–10. Block Diagram Representation of Interpolation



To see how interpolation filters work, consider the Nyquist Sampling Theorem. This theorem states that the maximum frequency of the input to be sampled must be smaller than $f_s/2$, where f_s is the sampling frequency, to avoid aliasing. This frequency, $f_s/2$, is also known as the Nyquist frequency (F_n). Typically, before a signal is sampled using an analog to digital converter (ADC), it needs to be low pass filtered using an analog anti-aliasing filter to prevent aliasing. If the input frequency spectrum extends close to the Nyquist frequency, then the first alias is also close to the Nyquist frequency. Therefore, the low pass filter needs to be very sharp to reject this alias. A very sharp analog filter is hard to design and manufacture and could increase passband ripple, thereby compromising system performance.

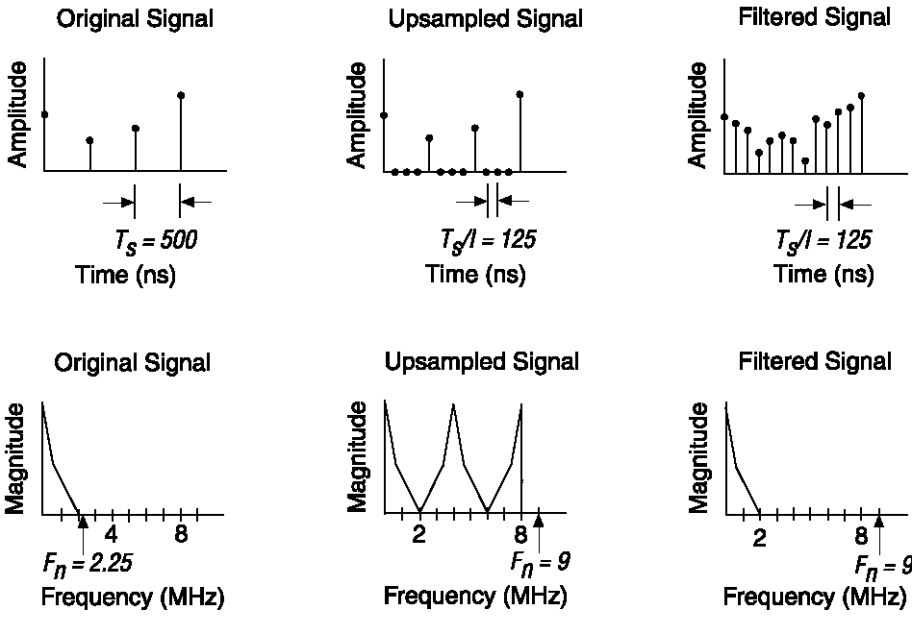
The solution is to increase the sampling rate of the ADC, so that the new Nyquist frequency is higher and the spacing between the desired signal and the alias is also higher. Zero padding as described above increase the sample rate. This process also known as upsampling (oversampling) relaxes the roll off requirements of the anti-aliasing filter. Consequently, a simpler filter achieves alias suppression. A simpler analog filter is easier to implement, does not compromise system performance, and is also easier to manufacture.

Similarly, the digital to analog converter (DAC) typically interpolates the data before the digital to analog conversion. This relaxes the requirement on the analog low pass filter at the output of the DAC.

The interpolation filter does not need to run at the oversampled (upsampled) rate of $f_s \times I$. This is because the extra sample points added are zeros, so they do not contribute to the output.

Figure 19–11 shows the time and frequency domain representation of interpolation for a specific case where the original signal spectrum is limited to 2 MHz and the interpolation factor (I) is 4. The Nyquist frequency of the upsampled signal must be greater than 8 MHz, and is chosen to be 9 MHz for this example.

Figure 19–11. Time & Frequency Domain Representations of Interpolation for I = 4



As an example, CD players use interpolation, where the nominal sample rate of audio input is 44.1 kilosamples per second. A typical implementation might have an interpolation (oversampling) factor of 4 generating 176.4 kilosamples per second of oversampled data stream.

Polyphase Interpolation Filters

A direct implementation of an interpolation filter, as shown in Figure 19–10, imposes a high computational burden. For example, if the filter is 16 taps long and a multiplication takes one cycle, then the number of computations required per cycle is 16× I. Depending on the interpolation factor (I), this number can be quite big and may not be achievable in hardware. A polyphase implementation of the low pass filter can reduce the number of computations required per cycle, often by a large factor, as will be evident later in this section.

The polyphase implementation "splits" the original filter into I polyphase filters whose impulse responses are defined by the following equation:

$$h_k(n) = h(k + nI)$$

where:

k = 0,1, …, I-1
n = 0,1, …, P-1
P = L/I = length of polyphase filters
L = length of the filter (selected to be a multiple of I)
I = interpolation factor
$h(n)$ = original filter impulse response

This equation implies that the first polyphase filter, $h_0(n)$, has coefficients $h(0)$, $h(I)$, $h(2I)$,..., $h((P-1)I)$. The second polyphase filter, $h_1(n)$, has coefficients $h(1)$, $h(1+I)$, $h(1+2I)$, ..., $h(1+(P-1)I)$. Continuing in this way, the last polyphase filter, $hI_{-1}(n)$, has coefficients $h(I-1)$, $h((I-1)+N)$, $h((I-1)+2I)$, ..., $h((I-1)+(P-1)I)$.

An example helps in understanding the polyphase implementation of interpolation. Consider the polyphase representation of a 16-tap low pass filter with an interpolation factor of 4. Thus, the output is given below:

$$y(n) = \sum_{i=0}^{15} h(n - iI)x(i)$$

Referring back to Figure 19–11 on page 19–19, the only nonzero samples of the input are $x(0)$, $x(4)$, $x(8,)$ and $x(12)$. The first output, $y(0)$, only depends on $h(0)$, $h(4)$, $h(8)$ and $h(12)$ because $x(i)$ is zero for $i \neq 0, 4, 8, 12$. Table 19–9 shows the coefficients required to generate output samples.

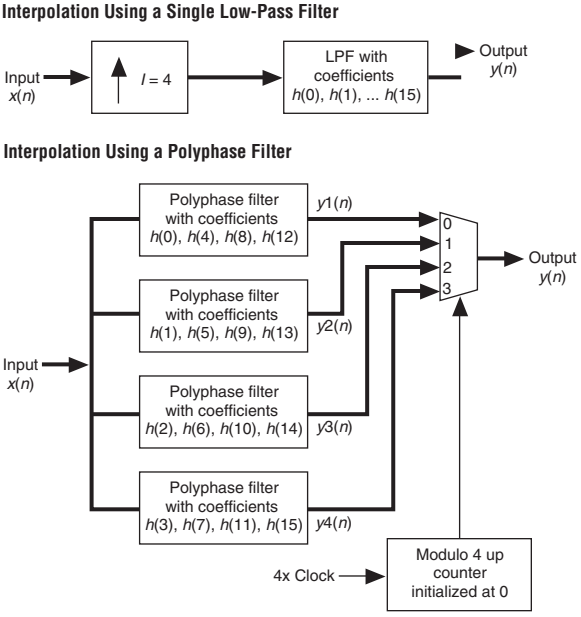
Table 19–9. Decomposition of a 16-Tap Interpolating Filter into Four Polyphase Filters

Output Sample	Coefficients Required	Polyphase Filter Impulse Response
y(0), y(4)...	h(0), h(4), h(8), h(12)	$h_0(n)$
y(1), y(5)...	h(1), h(5), h(9), h(13)	$h_1(n)$
y(2), y(6)...	h(2), h(6), h(10), h(14)	$h_2(n)$
y(3), y(7)...	h(3), h(7), h(11), h(15)	$h_3(n)$

Table 19–9 shows that this filter operation can be represented by four parallel polyphase filters. This is shown in Figure 19–12. The outputs from the filters are multiplexed to generate the overall output. The multiplexer is controlled by a counter, which counts up modulo-I starting at 0.

It is illuminating to compare the computational requirements of the direct implementation versus polyphase implementation of the low pass filter. In the direct implementation, the number of computations per cycle required is 16 × I = 16 × 4 = 64. In the polyphase implementation, the number of computations per cycle required is 4 × 4 = 16. This is because there are four polyphase filters, each with four taps.

Figure 19–12. Polyphase Representation of I=4 Interpolation Filter



Polyphase Interpolation Filter Implementation

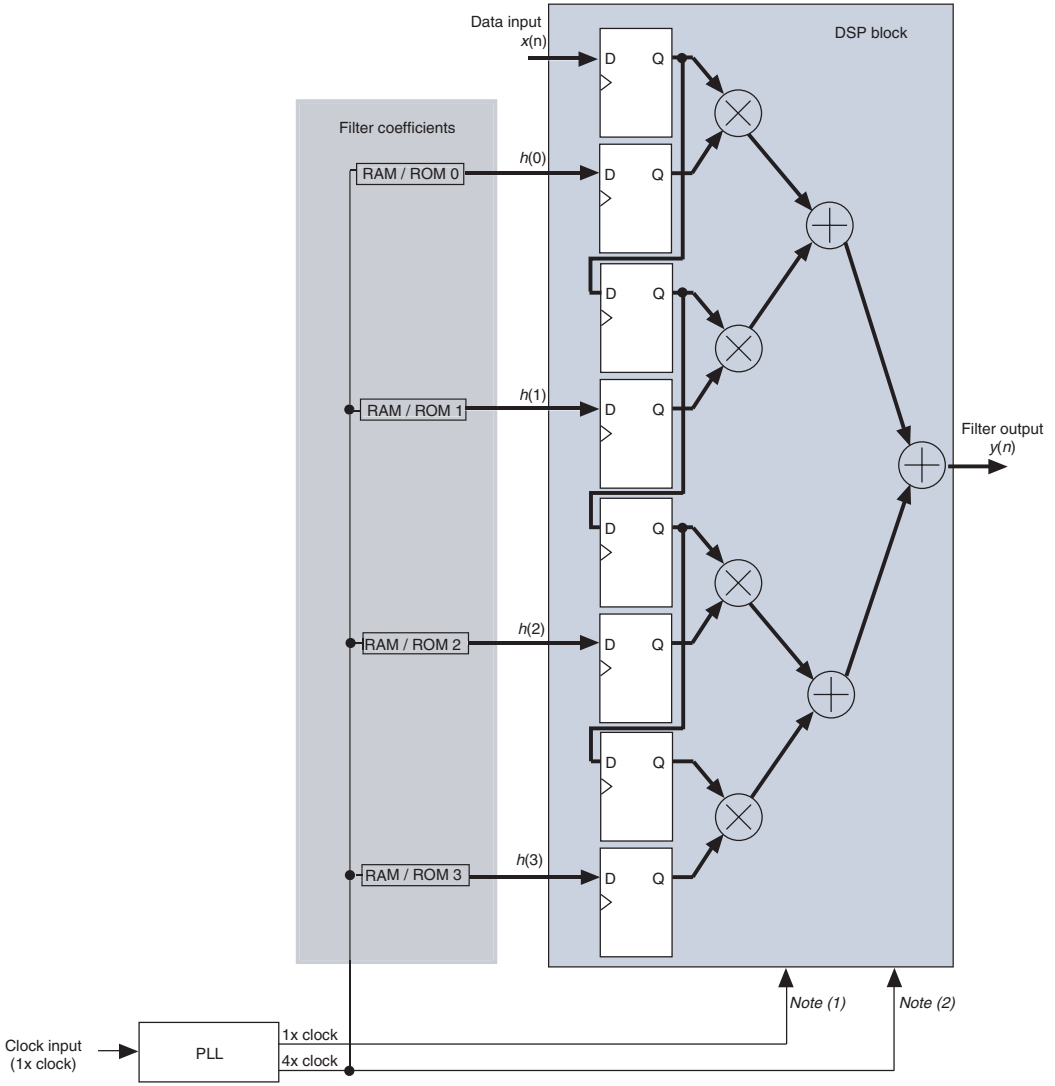
Figure 19–13 shows the Stratix or Stratix GX implementation of the polyphase interpolation filter in Figure 19–12. The four polyphase filters share the same hardware, which is a 4-tap filter. One Stratix or Stratix GX DSP block can implement one 4-tap filter with 18-bit wide data and coefficients. A multiplexer can be used to load new coefficient values on every cycle of the 4× clock. Stratix and Stratix GX phase lock loops (PLLs) can generate the 4× clock. In the first cycle of the 4× clock, the user needs to load coefficients for polyphase filter $h_0(n)$; in the second cycle of the 4×

clock, the users needs to load coefficients of the polyphase filter $h_1(n)$ and so on. Table 19–10 summarizes the coefficient loading schedule. The output, $y(n)$, is clocked using the 4× clock.

Table 19–10. Polyphase Interpolation (I=4) Filter Coefficient Loading Schedule

Cycle of 4× Clock	Coefficients to Load	Corresponding RAM/ROM
1, 5,...	h(0), h(4), h(8), h(12)	0, 1, 2, 3
2, 6,...	h(1), h(5), h(9), h(13)	0, 1, 2, 3
3, 7,...	h(2), h(6), h(10), h(14)	0, 1, 2, 3
4, 8,...	h(3), h(7), h(11), h(15)	0, 1, 2, 3

Figure 19–13. Implementation of the Polyphase Interpolation Filter (I=4) Notes (1), (2), (3)



Notes to Figure 19–13:

(1) The 1× clock feeds the input data shiftin register chain.

(2) The 4× clock feeds the input registers for the filter coefficients and other optional registers in the DSP block. See *Note (3)*.

(3) To increase the DSP block performance, include the pipeline, and output registers. See Figure 19–3 for the details.

Polyphase Interpolation Filter Implementation Results

Table 19–11 shows the results of the polyphase interpolation filter implementation in a Stratix device shown in Figure 19–13.

Table 19–11. Polyphase Interpolation Filter Implementation Results

Part	EP1S10F780
Utilization	Lcell: 3/10570 (<1%) DSP Block 9-bit elements: 8/48 (17%) Memory bits: 288/920448 (<1%)
Performance	240 MHz *(1)*

Note to Table 19–11:

(1) This refers to the performance of the DSP blocks, as well as the output clock rate. The input rate is 60 MSPS, clocked in at 60MHz.

Polyphase Interpolation Filter Design Example

Download the Interpolation FIR Filter (**interpolation_fir.zip**) design example from the Design Examples section of the Altera web site at **www.altera.com.**

Polyphase FIR Decimation Filters

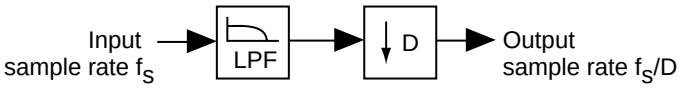
A decimation filter can be used to decrease the sample rate. A decimation filter is efficiently implemented with a polyphase FIR filter. DSP systems frequently use polyphase filters because they simplify overall system design and also reduce the number of computations per cycle required of the hardware. This section first describes decimation filters and then how to implement them as polyphase filters in Stratix devices. See the "Polyphase FIR Interpolation Filters" section for a discussion of interpolation filters.

Decimation Filter Basics

A decimation filter decreases the output sample rate by a factor of D through keeping only every D-th input sample. Consequently, the samples at the output of the decimation filter are separated by $D \times T_s = D/f_s$, where T_s and f_s are the sample period and sample frequency of the original signal, respectively. Figure 19–14 shows the concept of signal decimation.

The signal needs to be low pass filtered before downsampling can begin in order to avoid the reflections of the original spectrum from being aliased back into the output signal.

Figure 19–14. Block Diagram Representation of Decimation

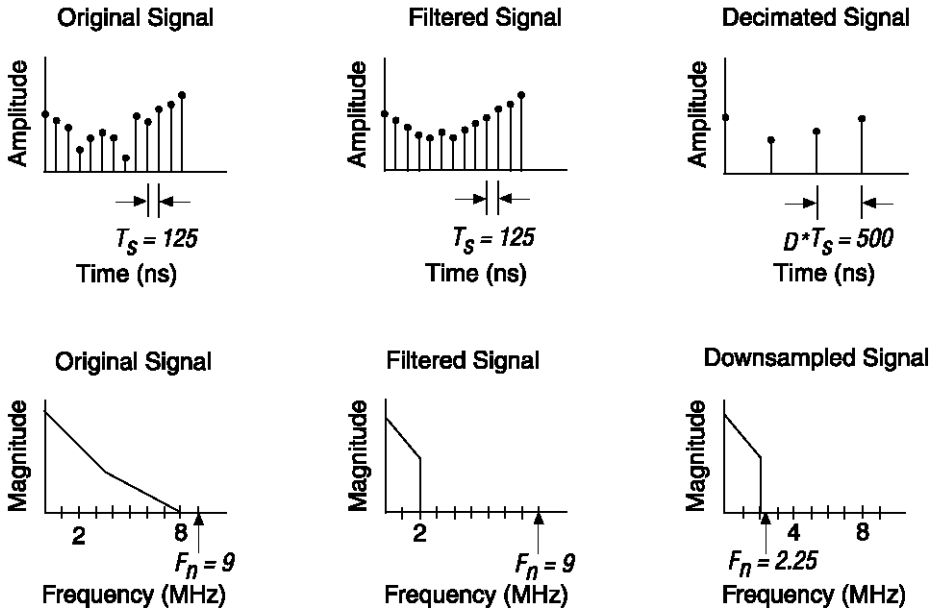


Decimation filters reverse the effect of the interpolation filters. Before the decimation process, a low pass filter is applied to the signal to attenuate noise and aliases present beyond the Nyquist frequency. The filtered signal is then applied to the decimation filter, which processes every D-th input. Therefore the values between samples D, D-1, D-2 etc. are ignored. This allows the filter to run M times slower than the input data rate.

In a typical system, after the analog to digital conversion is complete, the data needs to be filtered to remove aliases inherent in the sampled data. Further, at this point there is no need to continue to process this data at the higher sample (oversampled) rate. Therefore, a decimation FIR filter at the output of the ADC lowers the data rate to a value that can be processed digitally.

Figure 19–15 shows a specific example where a signal spread over 8 MHz is decimated by a factor of 4 to 2 MHz. The Nyquist frequency of the downsampled signal must be greater than 2 MHz, and is chosen to be 2.25 MHz in this example.

Figure 19–15. Time & Frequency Domain Representations of Decimation for D=4



Polyphase Decimation Filters

Figure 19–14 shows a direct implementation of a decimation filter, which imposes a high computational burden. For example, if the filter is 16 taps long and a multiplication takes one cycle, the number of computations required per cycle is 16× D. Depending on the decimation factor (D), this number can be quite big and may not be achievable in hardware. A polyphase implementation of the low pass filter can reduce the number of computations required, often by a large ratio, as will be evident later in this section.

The polyphase implementation "splits" the original filter into D polyphase filters with impulse responses defined by the following equation.

$$h_k(n) = h(k + nD)$$

where:

k = 0,1, …, D-1
n = 0,1, …, P-1
P = L/D = length of polyphase filters
L is the length of the filter (selected to be a multiple of D)
D is the decimation factor
$h(n)$ is the original filter impulse response

This equation implies that the first polyphase filter, $h_0(n)$, has coefficients $h(0)$, $h(D)$, $h(2D)$…$h((P-1)D)$. The second polyphase filter, $h_1(n)$, has coefficients $h(1)$, $h(1+D)$, $h(1+2D)$, ..., $h(1+(P-1)D)$. Continuing in this way, the last polyphase filter, $h_{D-1}(n)$ has coefficients $h(D-1)$, $h((D-1)+D)$, $h((D-1)+2D)$, ..., $h((D-1)+(P-1)D)$.

An example helps in the understanding of the polyphase implementation of decimation. Consider the polyphase representation of a 16-tap low pass filter with a decimation factor of 4. The output is given by:

$$y(n) = \sum_{i=0}^{15} h(i)x(nD-i)$$

Referring to Figure 19–15 on page 19–26, it is clear that the output, $y(n)$ is discarded for $n \neq 0, 4, 8, 12$, hence the only values of $y(n)$ that need to be computed are $y(0)$, $y(4)$, $y(8)$, $y(12)$. Table 19–12 shows which coefficients are required to generate the output samples.

Table 19–12. Decomposition of a 16-Tap Decimation Filter into Four Polyphase Filters

Output Sample *(1)*	Coefficients Required	Polyphase Filter Impulse Response
$y(0)_0, y(4)_0, \ldots$	h(0), h(4), h(8), h(12)	$h_0(n)$
$y(0)_1, y(4)_1, \ldots$	h(1), h(5), h(9), h(13)	$h_1(n)$
$y(0)_2, y(4)_2, \ldots$	h(2), h(6), h(10), h(14)	$h_2(n)$
$y(0)_3, y(4)_3, \ldots$	h(3), h(7), h(11), h(15)	$h_3(n)$

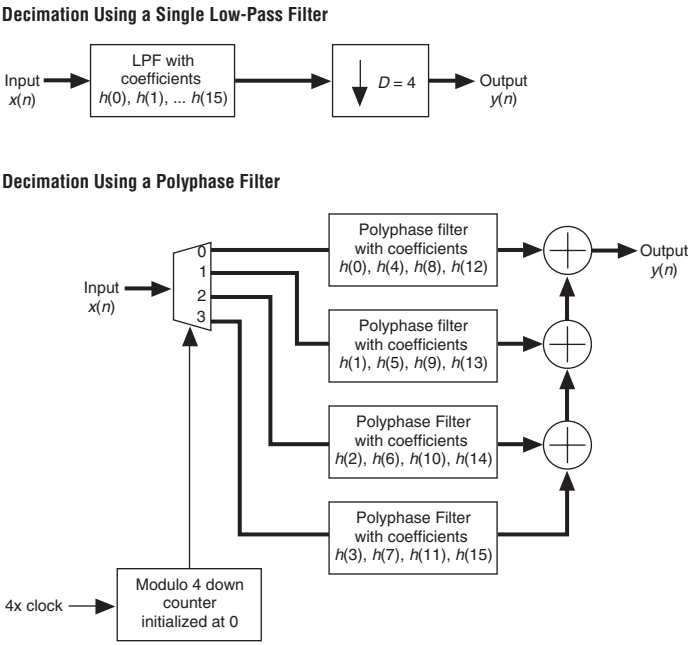
Note to Table 19–12:
(1) The output sample is the sum of the results from four polyphase filters: $y(n) = y(n)_0 + y(n)_1 + y(n)_2 + y(n)_3$.

Table 19–12 shows that the overall decimation filter operation can be represented by 4 parallel polyphase filters. Figure 19–16 shows the polyphase representation of the decimation filter. A demultiplexer at the input ensures that the input is applied to only one polyphase filter at a

time. The demultiplexer is controlled by a counter, which counts down modulo-D starting at 0. The overall output is taken by adding the outputs of all the filters.

The polyphase representation of the decimation filter also reduces the computational requirement. For the example in Figure 19–16, the direct implementation requires $16 \times D = 16 \times 4 = 64$ computations per cycle, whereas the polyphase implementation requires only $4 \times 4 \times 1 = 16$ computations per cycle. This saving in computational complexity is quite significant and is often a very convincing reason to use polyphase filters.

Figure 19–16. Polyphase Filter Representation of a D=4 Decimation Filter



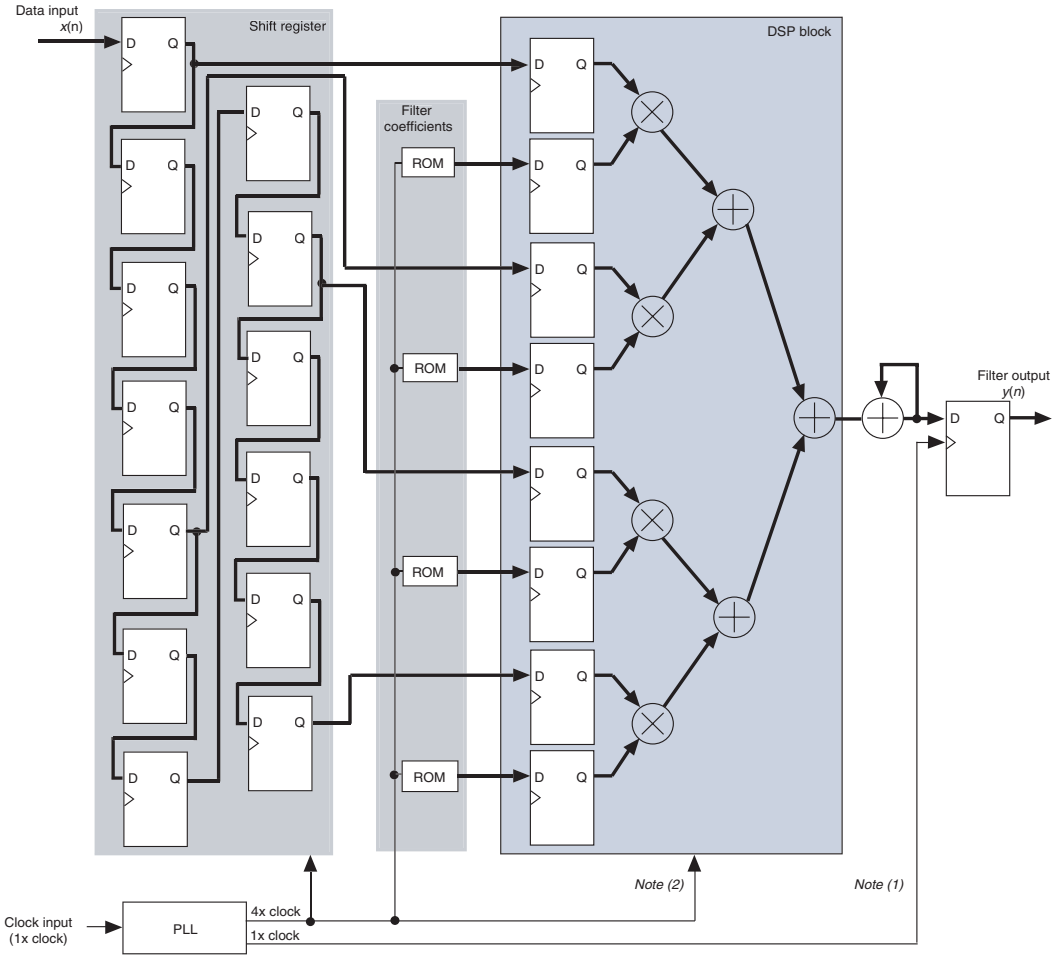
Polyphase Decimation Filter Implementation

Figure 19–17 shows the decimation polyphase filter example of Figure 19–16 as it would fit into Stratix or Stratix GX DSP blocks. The coefficients of the polyphase filters need to be cycled using the schedule shown in Table 19–13. The output *y(n)*, is clocked using the 1× clock.

Table 19–13. Coefficient Loading Schedule for Polyphase Decimation Filter (D=4)

Cycle of 4× Clock	Coefficients to Load	Corresponding RAM/ROM
1, 5,...	h(0), h(4), h(8), h(12)	0, 1, 2, 3
2, 6,...	h(3), h(7), h(11), h(15)	0, 1, 2, 3
3, 7,...	h(2), h(6), h(10), h(14)	0, 1, 2, 3
4, 8,...	h(1), h(5), h(9), h(13)	0, 1, 2, 3

Figure 19–17. Implementation of the Polyphase Decimation Filter (D=4) Notes (1), (2), (3)



Notes to Figure 19–17:

(1) The 1× clock feeds the register after the accumulator block.

(2) The 4× clock feeds the shift register for the data, the input registers for both the data and filter coefficients, the other optional registers in the DSP block (see *Note (3)*), and the accumulator block.

(3) To increase the DSP block performance, include the pipeline, and output registers. See Figure 19–3 on page 19–8 for the details.

Polyphase Decimation Filter Implementation Results

Table 19–14 shows the results of the polyphase decimation filter implementation in a Stratix device shown in Figure 19–17.

Table 19–14. Polyphase Decimation Filter Implementation Results

Part	EP1S10F780
Utilization	Lcell: 168/10570 (1%) DSP Block 9-bit elements: 8/48 (17%) Memory bits: 300/920448(<1%)
Performance	240 MHz *(1)*

Note to Table 19–14:

(1) This refers to the performance of the DSP blocks, as well as the input clock rate. The output rate is 60 MSPS (clocked out at 60MHz).

Polyphase Decimation Filter Design Example

Download the Decimation FIR Filter (**decimation_fir.zip**) design example from the Design Examples section of the Altera web site at **www.altera.com.**

Complex FIR Filter

A complex FIR filter takes real and imaginary input signals and performs the filtering operation with real and imaginary filter coefficients. The output also consists of real and imaginary signals. Therefore, a complex FIR filter is similar to a regular FIR filter except for the fact that the input, output, and coefficients are all complex numbers.

One example application of complex FIR filters is equalization. Consider a Phase Shift Keying (PSK) system; a single complex channel can represent the I and Q data channels. A FIR filter with complex coefficients could then process both data channels simultaneously. The filter coefficients are chosen in order to reverse the effects of intersymbol interference (ISI) inherent in practical communication channels. This operation is called equalization. Often, the filter is adaptive, i.e. the filter coefficients can be varied as desired, to optimize performance with varying channel characteristics.

A complex variable FIR filter is a cascade of complex multiplications followed by a complex addition. Figure 19–18 shows a block diagram representation of a complex FIR filter. To compute the overall output of the FIR filter, it is first necessary to determine the output of each complex multiplier. This can be expressed as:

$$y_{real} = x_{real} \times h_{real} - x_{imag} \times h_{imag}$$
$$y_{imag} = x_{real} \times h_{imag} + h_{real} \times x_{imag}$$

where:

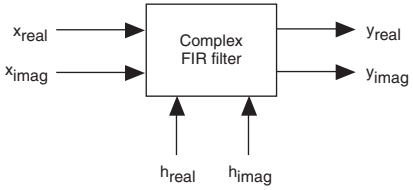
x_{real} is the real input signal
x_{imag} is the imaginary input signal
h_{real} is the real filter coefficients
h_{imag} is the imaginary filter coefficients
y_{real} is the real output signal
y_{imag} is the imaginary output signal

In complex representation, this equals:

$$y_{real} + jy_{imag} = (x_{real} + jx_{imag}) \times (h_{real} + jh_{imag})$$

The overall real channel output is obtained by adding the real channel outputs of all the multipliers. Similarly, the overall imaginary channel output is obtained by adding the imaginary channel outputs of all the multipliers.

Figure 19–18. Complex FIR Filter Block Diagram



Complex FIR Filter Implementation

Complex filters can be easily implemented in Stratix devices with the DSP blocks configured in the two-multipliers adder mode. One DSP block can implement a 2-tap complex FIR filter with 9-bit inputs, or a single tap complex FIR filter with 18-bit inputs. DSP blocks can be cascaded to implement complex filters with more taps.

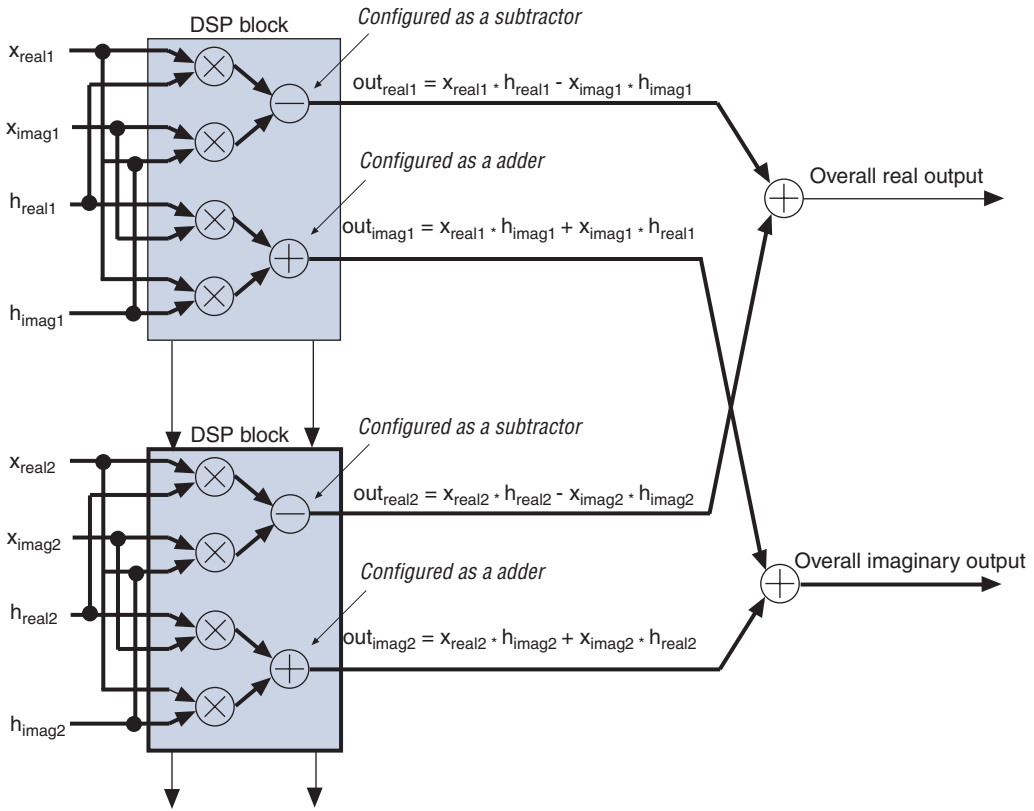


The two-multipliers adder mode has two adders, each adding the outputs of two multipliers. One of the adders is configured as a subtractor.

For more information on the different modes of the DSP blocks, see the *DSP Blocks in Stratix & Stratix GX Devices* chapter.

Figure 19–19 shows an example of a 2-tap complex FIR filter design with 18-bit inputs. The real and the complex outputs of the DSP blocks are added externally to generate the overall real and imaginary output. As in the case of basic, TDM, or polyphase FIR filters, the coefficients may be loaded in series or parallel.

Figure 19–19. 2-Tap 18-Bit Complex FIR Filter Implementation



Infinite Impulse Response (IIR) Filters

Another class of digital filters are IIR filters. These are recursive filters where the current output is dependent on previous outputs. In order to maintain stability in an IIR filter, careful design consideration must be given, especially to the effects of word-length to avoid unbounded conditions. The following section discusses the general theory and applications behind IIR Filters.

IIR Filter Background

The impulse response of an IIR filter extends for an infinite amount of time because their output is based on feedback from previous outputs. The general expression for IIR filters is:

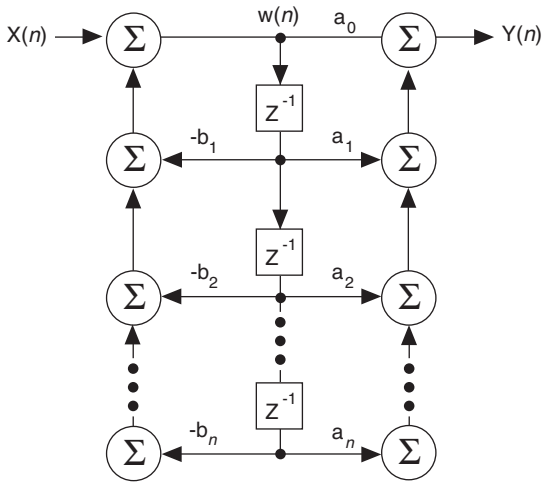
$$y(n) = \sum_{i=0}^{n} a(i)x(n-i) - \sum_{i=1}^{n} b(i)y(n-i)$$

where a_i and b_i represent the coefficients in the feed-forward path and feedback path, respectively, and n represents the filter order. These coefficients determine where the poles and zeros of the IIR filter lie. Consequently, they also determine how the filter functions (i.e., cut-off frequencies, band pass, low pass, etc.).

The feedback feature makes IIR filters useful in high data throughput applications that require low hardware usage. However, feedback also introduces complications and caution must be taken to make sure these filters are not exposed to situations in which they may become unstable. The complications include phase distortion and finite word length effects, but these can be overcome by ensuring that the filter always operates within its intended range.

Figure 19–20 shows a direct form II structure of an IIR filter.

Figure 19–20. Direct Form II Structure of an IIR Filter



The transfer function for an IIR filter is:

$$H(z) = \frac{\sum_{i=0}^{n} a_i z^{-i}}{1 + \sum_{i=1}^{n} b_i z^{-i}}$$

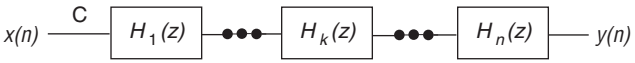
The numerator contains the zeros of the filter and the denominator contains the poles. The IIR filter structure requires a multiplication followed by an accumulation. Constructing the filter directly from the transfer function shown above may result in finite word length limitations and make the filter potentiality unstable. This becomes more critical as the filter order increases, because it only has a finite number of bits to represent the output. To prevent overflow or instability, the transfer function can be split into two or more terms representing several second order filters called biquads. These biquads can be individually scaled and cascaded, splitting the poles into multiples of two. For example, an IIR filter having ten poles should be split into five biquad sections. Doing this minimizes quantization and recursive accumulation errors.

This cascaded structure rearranges the transfer function. This is shown in the equation below, where each product term is a second order IIR filter. If n is odd, the last product term is a first order IIR filter:

$$H(z) = C \times \prod_{k=1}^{(n+1)/2} \frac{a_{0k} + a_{1k}z^{-1} + a_{2k}z^{-2}}{1 + b_{1k}z^{-1} + b_{2k}z^{-2}} = C \times \prod_{k=1}^{(n+1)/2} H_k(z)$$

Figure 19–21 shows the cascaded structure.

Figure 19–21. Cascaded IIR Filter



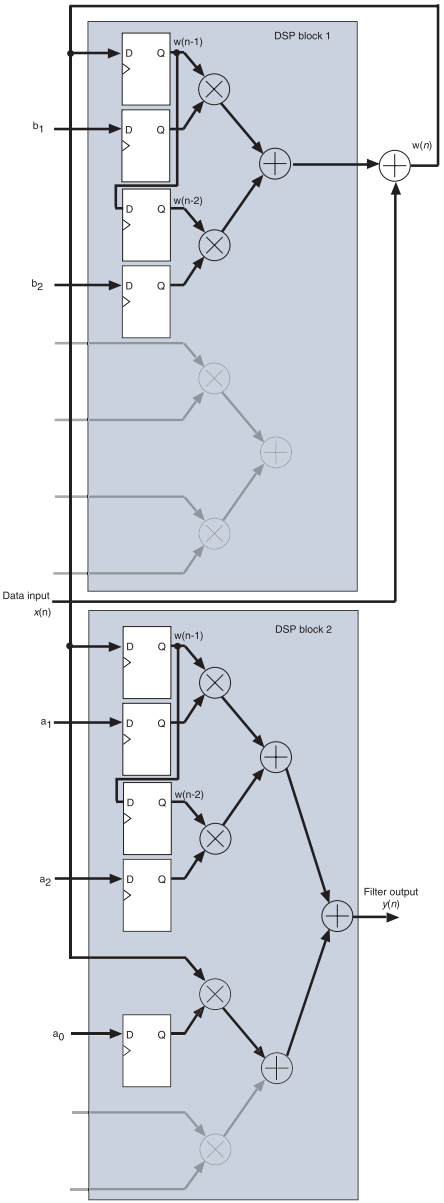
Basic IIR Filters

In this section, the basic IIR filter is implemented using cascaded second order blocks or biquads in the direct form II structure.

Basic IIR FIlter Implementation

Multiplier blocks, adders and delay elements can implement a basic IIR filter. The Stratix architecture lends itself to IIR filters because of its embedded DSP blocks, which can easily be configured to perform these operations. The `altmult_add` megafunction can be used to implement the multiplier-adder mode in the DSP blocks. Figure 19–22 shows the implementation of an individual biquad using Stratix and Stratix GX DSP blocks.

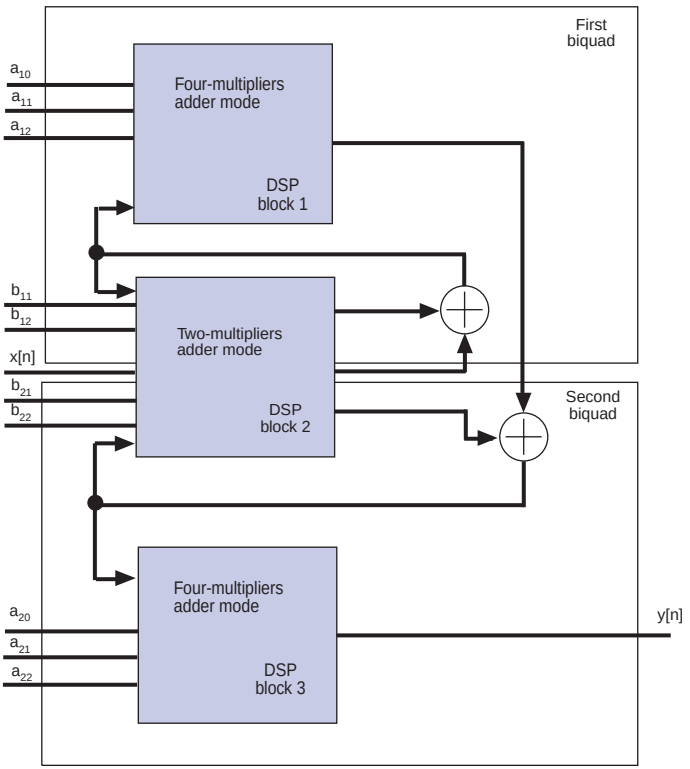
Figure 19–22. IIR Filter Biquad Note (1)



Note to Figure 19–22:
(1) Unused ports are grayed out.

The first DSP block in Figure 19–22 is configured in the two-multipliers adder mode, and the second DSP block is in the four-multipliers adder mode. For an 18-bit input to the IIR filter, each biquad requires five multipliers and five adders (two DSP blocks). One of the adders is implemented using logic elements. Cascading several biquads together can implement more complex, higher order IIR filters. It is possible to insert registers in between the biquad stages to improve the performance. Figure 19–23 shows a 4thorder IIR filter realized using two cascaded biquads in three DSP blocks.

Figure 19–23. Two Cascaded Biquads



Basic IIR Filter Implementation Results

Table 19–15 shows the results of implementing a 4th order IIR filter in a Stratix device.

Table 19–15. 4th Order IIR Filter Implementation Results

Part	EP1S10F780C5
Utilization	Lcell: 102/10570(<1%) DSP Block 9-bit elements: 24/48 (50%) Memory bits: 0/920448(0%)
Performance	73 MHz
Latency	4 clock cycles

Basic IIR Filter Design Example

Download the 4th Order IIR Filter (**iir.zip**) design example from the Design Examples section of the Altera web site at **www.altera.com.**

Butterworth IIR Filters

Butterworth filters are the most popular version of IIR analog filters. These filters are also known as "maximally flat" because they have no passband ripple. Additionally, they have a monotonic response in both the stopband and the passband. Butterworth filters trade-off roll off steepness for their no ripple characteristic. The distinguishing Butterworth filter feature is its poles are arranged in a uniquely symmetrical manner along a circle in the *s*-plane. The expression for the Butterworth filter's magnitude-squared function is as follows:

$$\left|H_c(j\omega)\right|^2 = \frac{1}{1+\left(\frac{j\omega}{j\omega_c}\right)^{2N}}$$

where:

ω_c is the cut-off frequency
N is the filter order

The filter's cutoff characteristics become sharper as *N* increases. If a substitution is made such that $j\omega = s$, then the following equation is derived:

$$H_c(s)H_c(-s) = \frac{1}{1+\left(\frac{s}{j\omega_c}\right)^{2N}}$$

with poles at:

$$s_k = (-1)^{\frac{1}{2N}}(j\omega_c) = \omega_c e^{\left(\frac{j\pi}{2N}\right)(2k+N-1)} \quad \text{for } k=0,1,\ldots,2N-1$$

There are 2N poles on the circle with a radius of ω_c in the *s*-plane. These poles are evenly spaced at π/N intervals along the circle. The poles chosen for the implementation of the filter lie in the left half of the *s*-plane, because these generate a stable, causal filter.

Each of the impulse invariance, the bilinear, and matched *z* transforms can transform the Laplace transform of the Butterworth filter into the *z*-transform.

- Impulse invariance transforms take the inverse of the Laplace transform to obtain the impulse response, then perform a *z*-transform on the sampled impulse response. The impulse invariance method can cause some aliasing.
- The bilinear transform maps the entire $j\omega$-axis in the *s*-plane to one revolution of the unit circle in the *z*-plane. This is the most popular method because it inherently eliminates aliasing.
- The matched *z*-transform maps the poles and the zeros of the filter directly from the *s*-plane to the *z*-plane. Usually, these transforms are transparent to the user because several tools, such as MATLAB, exist for determining the coefficients of the filter. The *z*-transform generates the coefficients much like in the basic IIR filter discussed earlier.

Butterworth Filter Implementation

For digital designs, consideration must be made to optimize the IIR biquad structure so that it maps optimally into logic. Because speed is often a critical requirement, the goal is to reduce the number of operations per biquad. It is possible to reduce the number of multipliers needed in each biquad to just two.

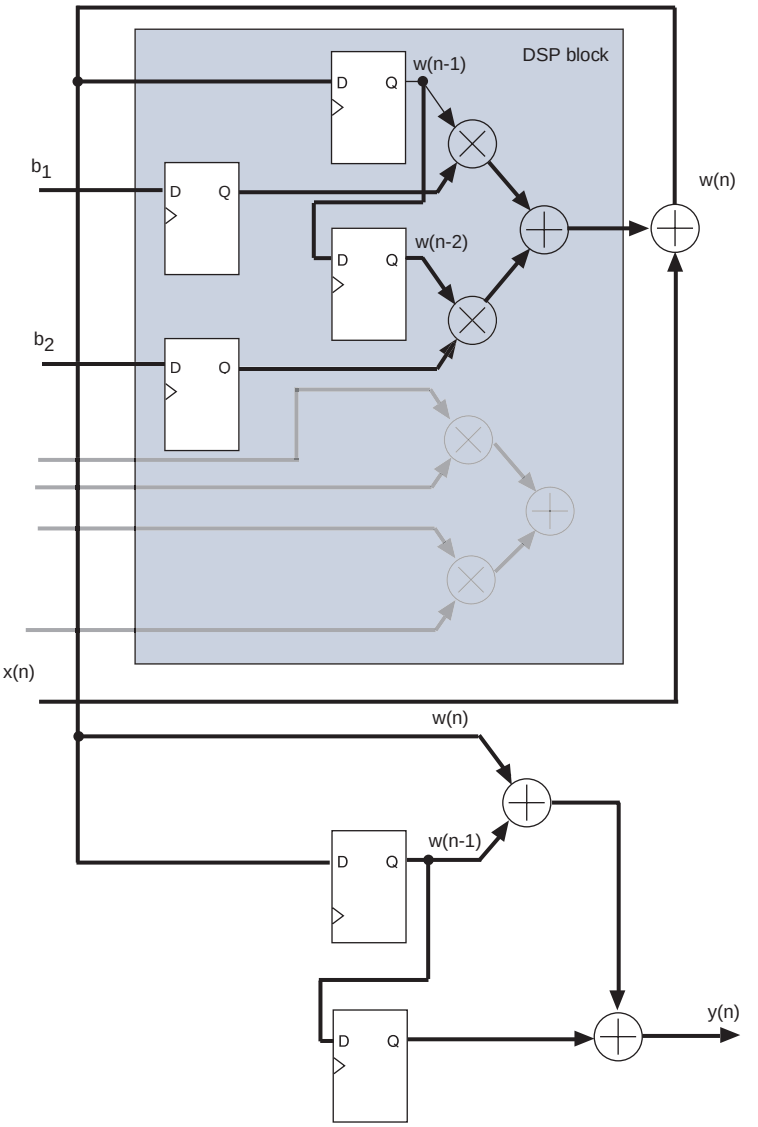
Through the use of integer feedforward multiplies, which can be implemented by combining addition, shifting, and complimenting operations, a Butterworth filter's transfer function biquad can be optimized for logic synthesis. The most efficient transformation is that of an all pole filter. This is because there is a unique relationship between the feedforward integer coefficients of the filter represented as:

$$H(z) = \frac{1 + 2z^{-1} + z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}$$

As can be seen by this equation, the z^{-1} coefficient in the numerator (representing the feedforward path) is twice the other two operands (z^{-2} and 1). This is always the case in the transformation from the frequency to the digital domain. This represents the normalized response, which is faster and smaller to implement in hardware than real multipliers. It introduces a scaling factor as well, but this can be corrected at the end of the cascade chain through a single multiply.

Figure 19–24 shows how a Butterworth filter biquad is implemented in a Stratix or Stratix GX device.

Figure 19–24. Butterworth Filter Biquad Notes (1), (2)



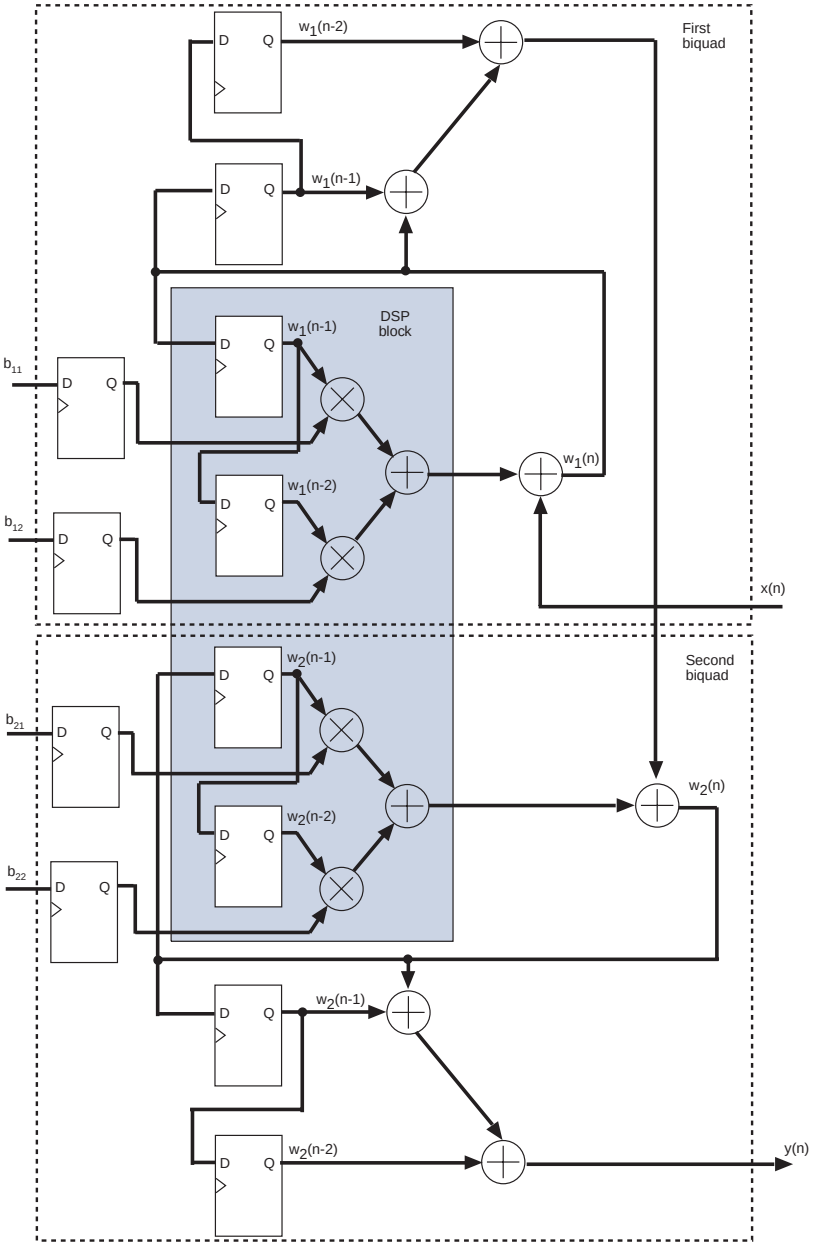
Notes to Figure 19–24:

(1) Unused ports are grayed out.

(2) The z^{-1} coefficient is a multiple of the other coefficients (z-2 and 1) in the feedforward path. This is implemented using a shift operation.

The DSP block in Figure 19–24 is configured in multiply and add mode. The three external adders are implemented in logic elements and therefore are not part of the DSP block. Therefore, for an 18-bit input, each biquad requires half a DSP block and three logic element adders. The gain factor can be compensated for at the end of the filtering stage and is not shown in this simple example. More complex, higher order Butterworth filters can be realized by cascading several biquads together, as in the IIR example. Figure 19–25 below shows a 4th order Butterworth filter using two cascaded biquads in a single DSP block.

Figure 19–25. Cascaded Butterworth Biquads Note (1)



Note to Figure 19–25:

(1) The gain factor is compensated for at the end of the filtering stage and is not shown in this figure.

Butterworth Filter Implementation Results

Table 19–16 shows the results of implementing a 4th order Butterworth filter as shown in Figure 19–25.

Table 19–16. 4th Order Butterworth Filter Implementation Results

Part	EP1S10F780C6
Utilization	Lcell: 251/10570(2%) DSP Block 9-bit elements: 16/48 (33%) Memory bits: 0/920448 (0%)
Performance	80 MHz
Latency	4 clock cycles

Butterworth Filter Design Example

Download the 4th Order Butterworth Filter (**butterworth.zip**) design example from the Design Examples section of the Altera web site at **www.altera.com.**

Matrix Manipulation

DSP relies heavily on matrix manipulation. The key idea is to transform the digital signals into a format that can then be manipulated mathematically.

This section describes an example of matrix manipulation used in 2-D convolution filter, and its implementation in a Stratix device.

Background on Matrix Manipulation

A matrix can represent all digital signals. Apart from the convenience of compact notation, matrix representation also exploits the benefits of linear algebra. As with one-dimensional, discrete sequences, this advantage becomes more apparent when processing multi-dimensional signals.

In image processing, matrix manipulation is important because it requires analysis in the spatial domain. Smoothing, trend reduction, and sharpening are examples of common image processing operations, which are performed by convolution. This can also be viewed as a digital filter operation with the matrix of filter coefficients forming a convolutional kernel, or mask.

Two-Dimensional Filtering & Video Imaging

FIR filtering for video applications and image processing in general is used in many applications, including noise removal, image sharpening to feature extraction.

For noise removal, the goal is to reduce the effects of undesirable, contaminative signals that have been linearly added to the image. Applying a low pass filter or smoothing function flattens the image by reducing the rapid pixel-to-pixel variation in gray levels and, ultimately, removing noise. It also has a blurring effect usually used as a precursor for removing unwanted details before extracting certain features from the image.

Image sharpening focuses on the fine details of the image and enhances sharp transitions between the pixels. This acts as a high-pass filter that reduces broad features like the uniform background in an image and enhances compact features or details that have been blurred.

Feature extraction is a form of image analysis slightly different from image processing. The goal of image analysis in general is to extract information based on certain characteristics from the image. This is a multiple step process that includes edge detection. The easiest form of edge detection is the derivative filter, using gradient operators.

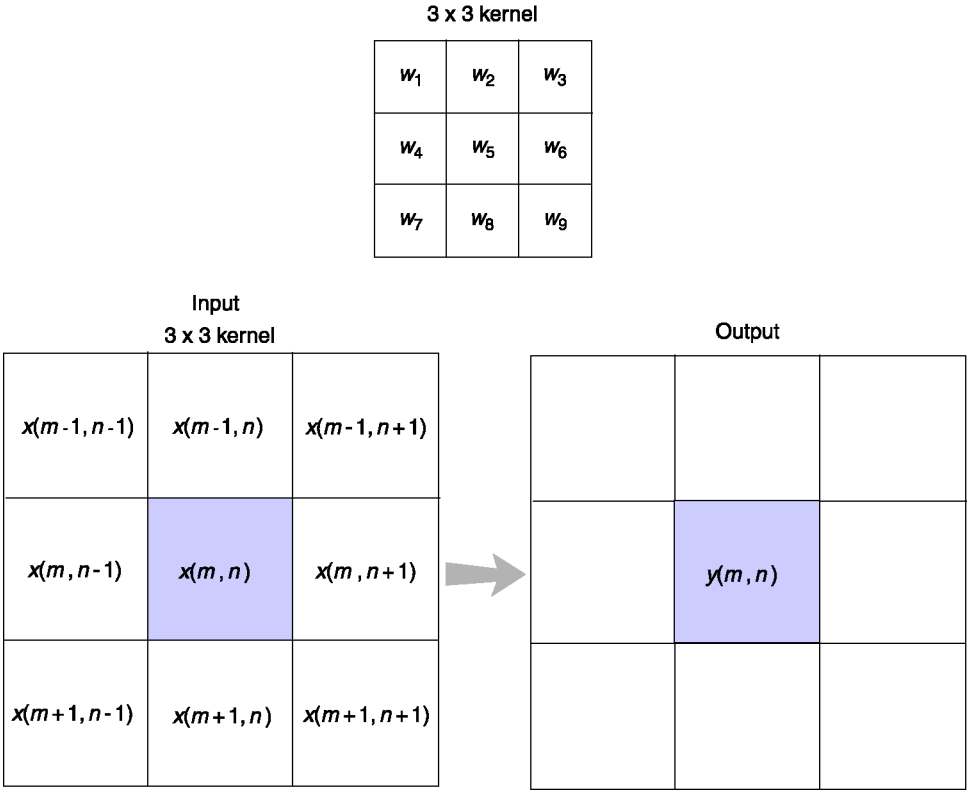
All of the operations above involve transformation of the input image. This can be presented as the convolution of the two-dimensional input image, *x*(m,n) with the impulse response of the transform, *f*(k,l), resulting in *y*(m,n) which is the output image.

$$y(m,\ n) = f(k,\ l) \otimes x(m,\ n)$$

$$y(m,\ n) = \sum_{k=-N}^{N} \sum_{l=-N}^{N} f(k,\ l)x(m-k,\ n-l)$$

The *f*(k,l) function refers to the matrix of filter coefficients. Because the matrix operation is analogous to a filter operation, the matrix itself is considered the impulse response of the filter. Depending on the type of operation, the choice of the convolutional kernel or mask, *f*(k,l) is different. Figure 19–26 shows an example of convolving a 3 × 3 mask with a larger image.

Figure 19–26. Convolution Using a 3 × 3 Kernel



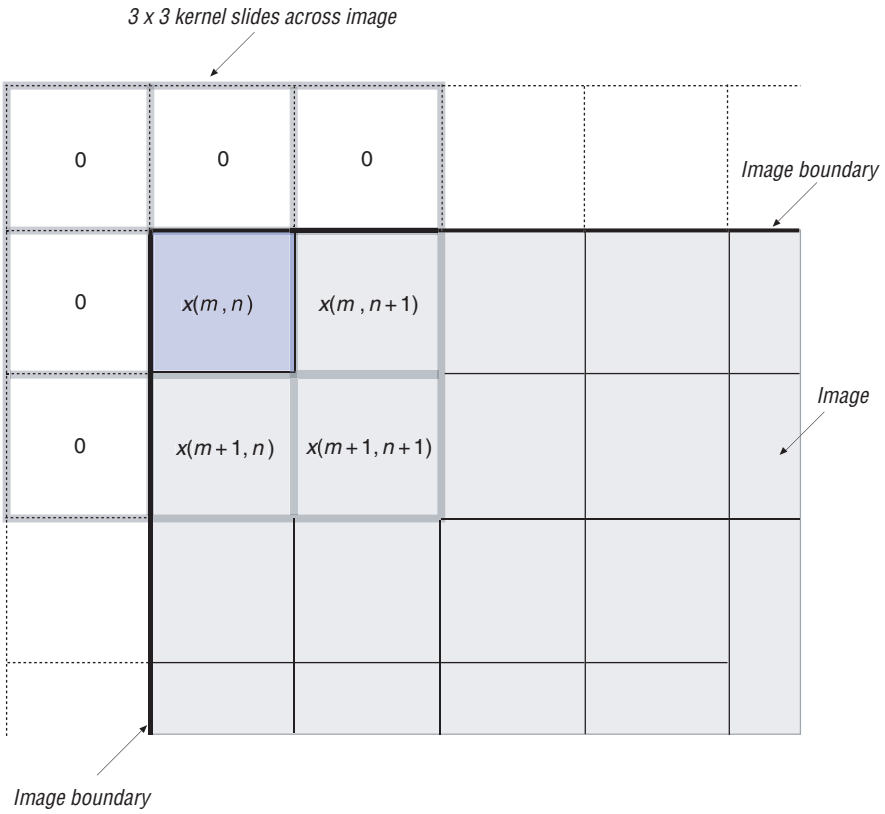
The output pixel value, $y(m,n)$ depends on the surrounding pixel values in the input image, as well as the filter weights:

$$y(m, n) = w_1x(m-1, n-1) + w_2x(m-1, n) + w_3x(m-1, n+1) + w_4x(m, n-1) + w_5x(m, n) + w_6x(m, n+1) + w_7x(m+1, n-1) + w_8x(m+1, n) + w_9x(m+1, n+1)$$

To complete the transformation, the kernel slides across the entire image. For pixels on the edge of the image, the convolution operation does not have a complete set of input data. To work around this problem, the pixels on the edge can be left unchanged. In some cases, it is acceptable to have an output image of reduced size. Alternatively, the matrix effect can be applied to edge pixels as if they are surrounded on the "empty" side by

black pixels, that is pixels with value zero. This is similar to padding the edges of the input image matrix with zeros and is referred to as the free boundary condition. This is shown in Figure 19–27.

Figure 19–27. Using Free Boundary Condition for Edge Pixels

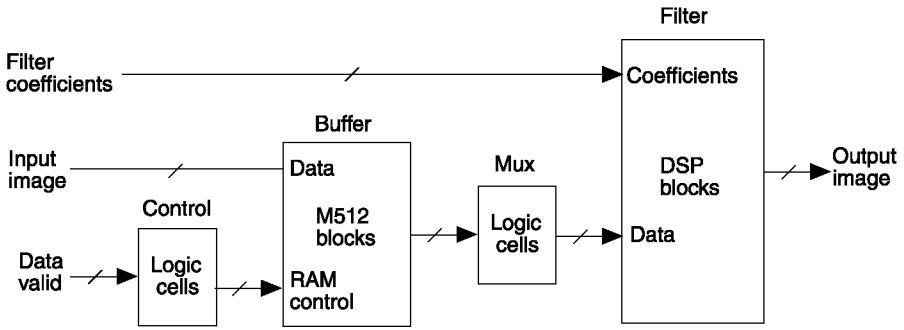


Convolution Implementation

This design example shows a 3 × 3 2-D FIR filter that takes in an 8 × 8 input image with gray pixel values ranging from 0-255 (8-bit). Data is fed in serially starting from the top left pixel, moving horizontally on a row-by-row basis. Next the data is stored in three separate RAM blocks in the buffering stage. Each M512 memory block represents a line of the image, and this is cycled through. For a 32 × 32 input image, the design needs M4K memory blocks. For larger images (640 × 480), this can be extended to M-RAM blocks or other buffering schemes. The control logic block provides the RAM control signals to interleave the data across all three

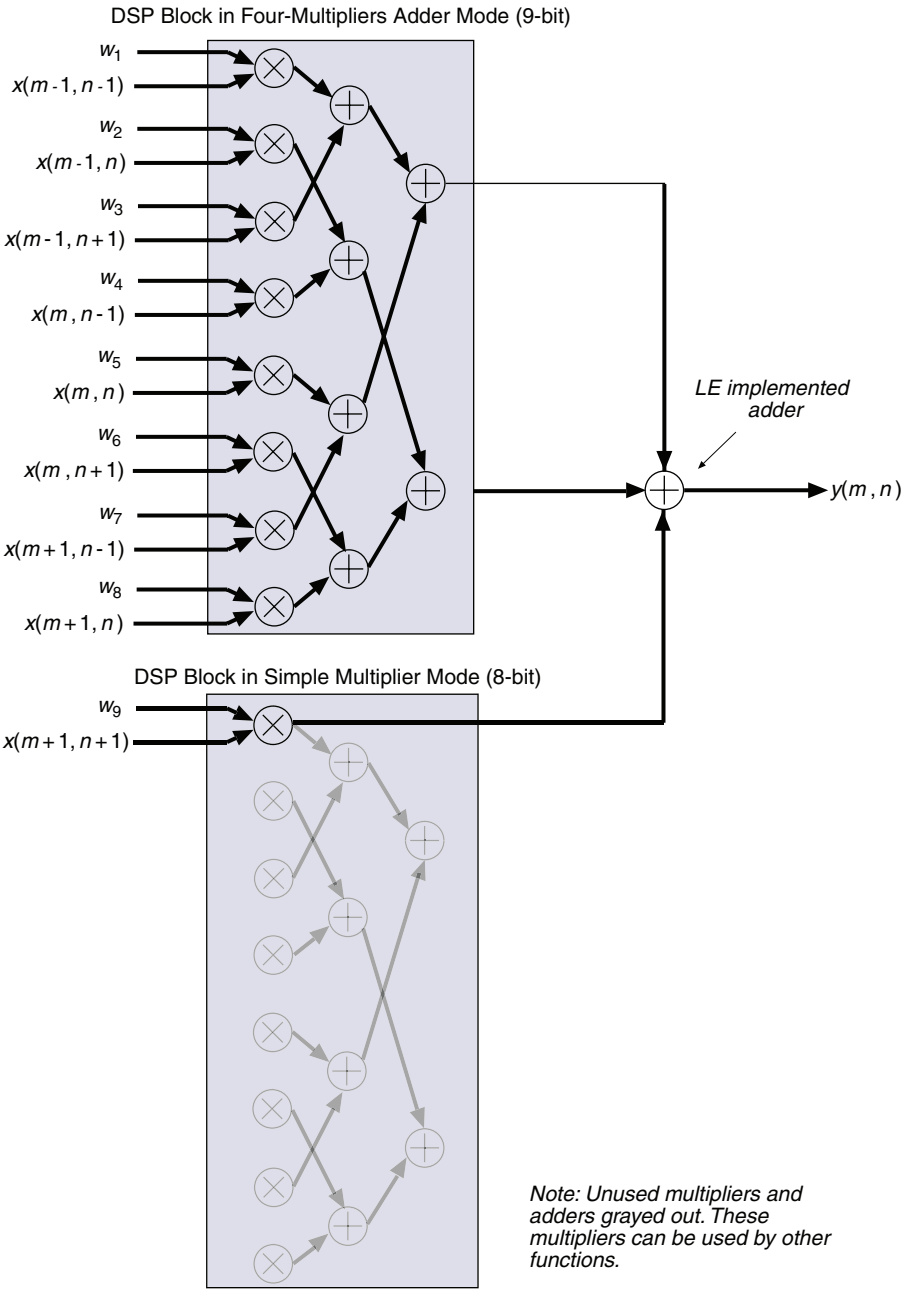
RAM blocks. The 9-bit signed filter coefficients feed directly into the filter block. As the data is shifted out from the RAM blocks, the multiplexer block checks for edge pixels and uses the free boundary condition. Figure 19–28 shows a top-level diagram of the design.

Figure 19–28. Block Diagram on Implementation of 3 × 3 Convolutional Filter for an 8 × 8 Pixel Input Image



The 3 × 3 filter block implements the nine multiply-add operations in parallel using two DSP blocks. One DSP block can implement eight of these multipliers. The second DSP block implements the ninth multiplier. The first DSP block is in the four-multipliers adder mode, and the second is in simple multiplier mode. In addition to the two DSP blocks, an external adder is required to sum the output of all nine multipliers. Figure 19–29 shows this implementation.

Figure 19–29. Implementation of 3 × 3 Convolutional Filter Block

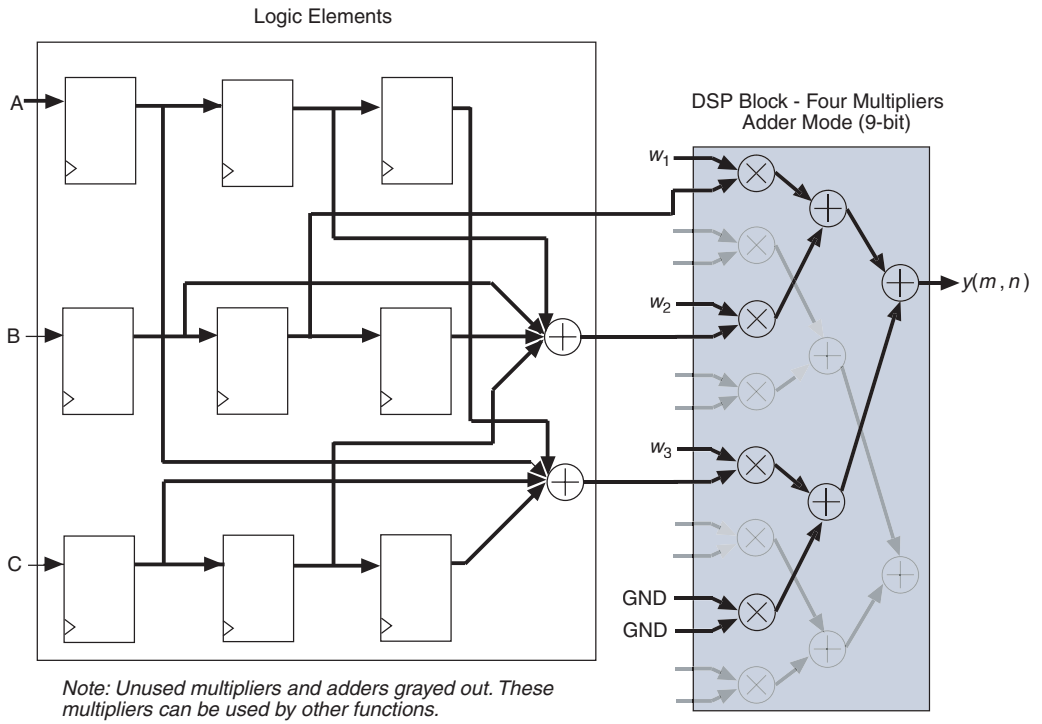


In cases where a symmetric 2-D filter is used, pixels sharing the same filter coefficients from three separate line-stores A, B, and C can be added together prior to the multiplication operation. This reduces the number of multipliers used. Referring to Figure 19–30, w_1, w_2, and w_3 are the filter coefficients. Figure 19–31 shows the implementation of this circular symmetric filter.

Figure 19–30. Symmetric 3 × 3 Kernel

w_3	w_2	w_3
w_2	w_1	w_2
w_3	w_2	w_3

Figure 19–31. Details on Implementation of Symmetric 3 × 3 Convolution Filter Block



Note: Unused multipliers and adders grayed out. These multipliers can be used by other functions.

Convolution Implementation Results

Table 19–17 shows the results of the 3 × 3 2-D FIR filter implementation in Figure 19–28.

Table 19–17. 3 × 3 2-D Convolution Filter Implementation Results

Part	EP1S10F780
Utilization	Lcell: 372/10570 (3%) DSP block 9-bit elements: 9/48 (18%) Memory bits: 768/920448 (<1%)
Performance	226 MHz
Latency	15 clock cycles

The design requires the input to be an 8 × 8 image, with 8-bit input data and 9-bit filter coefficient width. The output is an image of the same size.

Convolution Design Example

Download the 3 × 3 2-D Convolutional Filter (**two_d_fir.zip**) design example from the Design Examples section of the Altera web site at **www.altera.com.**

Discrete Cosine Transform (DCT)

The discrete cosine transform (DCT) is widely used in video and audio compression, for example in JPEG, MPEG video, and MPEG audio. It is a form of transform coding, which is the preferred method for compression techniques. Images tend to compact their energy in the frequency domain making compression in the frequency domain much more effective. This is an important element in compressing data, where the goal is to have a high data compression rate without significant degradation in the image quality.

DCT Background

Similar to the discrete fourier transform (DFT), the DCT is a function that maps the input signal or image from the spatial to the frequency domain. It transforms the input into a linear combination of weighted basis functions. These basis functions are the frequency components of the input data.

For 1-D with input data $x(n)$ of size N, the DCT coefficients $Y(k)$ are:

$$Y(k) = \frac{\alpha(k)}{2} \sum_{n=0}^{N-1} x(n) \cos\left(\frac{(2n+1)\pi k}{2N}\right) \quad \text{for } 0 \leq k \leq N-1$$

where:

$$\alpha(k) = \sqrt{\frac{1}{N}} \quad \text{for } k = 0$$

$$\alpha(k) = \sqrt{\frac{2}{N}} \quad \text{for } 1 \leq k \leq N-1$$

For 2-D with input data $x(m,n)$ of size $N \times N$, the DCT coefficients for the output image, $Y(p,q)$ are:

$$Y(p, q) = \frac{\alpha(p)\alpha(q)}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x(m, n) \cos\left(\frac{(2m+1)\pi p}{2N}\right) \cos\left(\frac{(2n+1)\pi q}{2N}\right)$$

where:

$$\alpha(p) = \sqrt{\frac{1}{N}} \quad \text{for } p = 0$$

$$\alpha(q) = \sqrt{\frac{1}{N}} \quad \text{for } q = 0$$

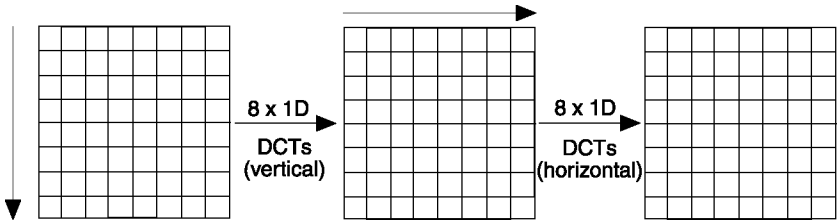
$$\alpha(p) = \sqrt{\frac{2}{N}} \quad \text{for } 1 \leq p \leq N-1$$

$$\alpha(q) = \sqrt{\frac{2}{N}} \quad \text{for } 1 \leq q \leq N-1$$

2-D DCT Algorithm

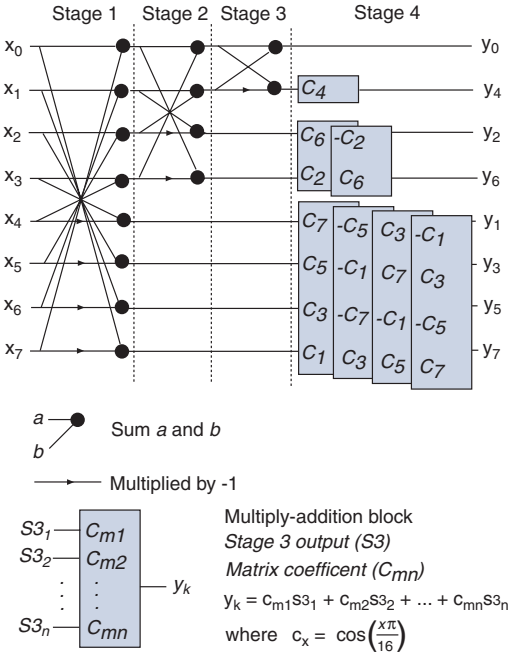
The 2-D DCT can be thought of as an extended 1-D DCT applied twice; once in the x direction and again in the y direction. Because the 2-D DCT is a separable transform, it is possible to calculate it using efficient 1-D algorithms. Figure 19–32 illustrates the concept of a separable transform.

Figure 19–32. A 2-D DCT is a Separable Transform



This section uses a standard algorithm proposed in [1]. Figure 19–33 shows the flow graph for the algorithm. This is similar to the butterfly computation of the fast fourier transform (FFT). Similar to the FFT algorithms, the DCT algorithm reduces the complexity of the calculation by decomposing the computation into successively smaller DCT components. The even coefficients (y_0, y_2, y_4, y_6) are calculated in the upper half and the odd coefficients (y_1, y_3, y_5, y_7) in the lower half. As a result of the decomposition, the output is reordered as well.

Figure 19–33. Implementing an N=8 Fast DCT



The following defines in matrix format, the 8-point 1-D DCT of Figure 19–33:

$$\left[Y_{1D}\right] = \left[x\right] \times \left[Add_1\right] \times \left[Add_2\right] \times \left[Add_3\right] \times \left[C\right]$$

where:

[*x*] is the 1 × 8 input matrix

$$\left[Add_1\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

$$\left[Add_2\right] = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\left[Add_3\right] = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[C] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & C_4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_6 & -C_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_2 & C_6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & C_7 & -C_5 & C_3 & -C_1 \\ 0 & 0 & 0 & 0 & C_5 & -C_1 & C_7 & C_3 \\ 0 & 0 & 0 & 0 & C_3 & -C_7 & -C_1 & -C_5 \\ 0 & 0 & 0 & 0 & C_1 & C_3 & C_5 & C_7 \end{bmatrix}$$

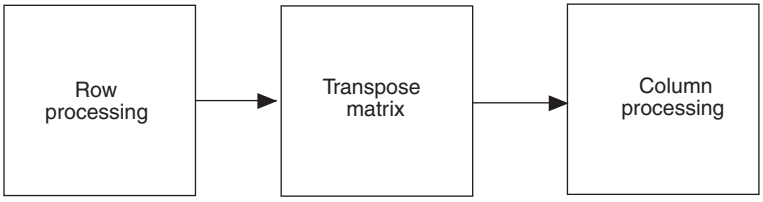
$$C_x = \cos\frac{\pi x}{16}$$

All of the additions in stages 1, 2 and 3 of Figure 19–32 appear in symmetric add and subtract pairs. The entire first stage is simply four such pairs in a very typical cross-over pattern. This pattern is repeated in stages 2 and 3. Multiplication operations are confined to stage 4 in the algorithm. This implementation is shown in more detail in the next section.

DCT Implementation

In taking advantage of the separable transform property of the DCT, the implementation can be divided into separate stages; row processing and column processing. However, some data restructuring is necessary before applying the column processing stage to the results from the row processing stage. The data buffering stage must transpose the data first. Figure 19–34 shows the different stages.

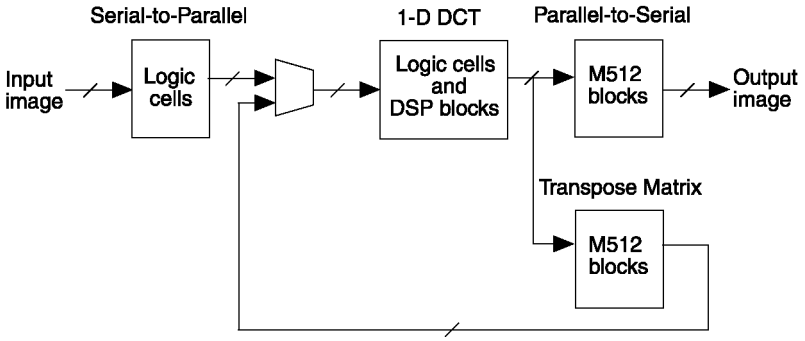
Figure 19–34. Three Separate Stages in Implementing the 2-D DCT



Because the row processing and column processing blocks share the same 1-D 8-point DCT algorithm, the hardware implementation shows this block as being shared. The DCT algorithm requires a serial-to-parallel conversion block at the input because it works on blocks of eight data

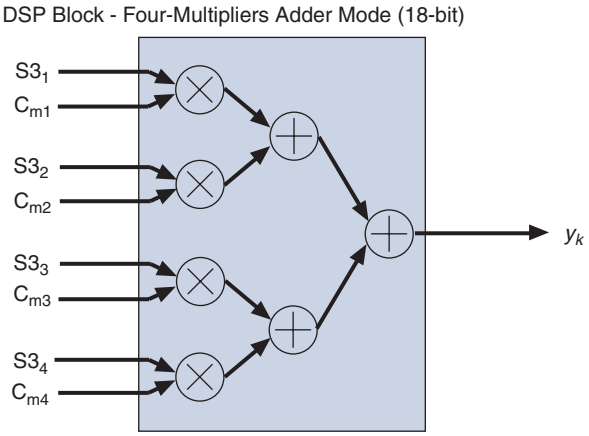
points in parallel. There is also a parallel-to-serial conversion block at the output because the column processing stage generates the output image column-by-column. In order to have the output in the same order as the input (i.e., row-by-row), this conversion is necessary. Appropriate scaling needs to be applied to the completed transform but this can be combined with the quantization stage which often follows a DCT [1]. Figure 19–35 shows a top-level block diagram of this design.

Figure 19–35. Block Diagram on Serial Implementation of 2-D DCT



The implementation of the 1-D DCT block is based on the algorithm shown in Figure 19–33. The simple addition and subtraction operations in stages 1, 2 and 3 are implemented using logic cells. The multiply and multiply-addition operations in stage 4 are implemented using DSP blocks in the Stratix device in the simple multiplier mode, two-multiplier adder mode, and the four-multiplier adder mode. An example of the multiply-addition block is shown in Figure 19–36.

Figure 19–36. Details on the Implementation of the Multiply-Addition Operation in Stage 4 of the 1-D DCT Algorithm



Note to Figure 19–36:
(1) Referring to Figure 19–33. $S3_n$ is an output from stage 3 of the DCT and C_{mn} is a matrix coefficient. $C_x = \cos(x\pi/16)$.

DCT Implementation Results

Table 19–18 shows the results of implementing a 2-D DCT with 18-bit precision, as shown in Figure 19–35.

Table 19–18. 2-D DCT Implementation Results

Part	EP1S20F780
Utilization	Lcell: 1717/18460 (9%) DSP Block 9-bit element: 18/80 (22%) Memory bits: 2816/1669248 (<1%)
Performance	165 MHz
Latency	80 clock cycles

DCT Design Example

Download the 2-D convolutional filter (**d_dct.zip**) design example from the Design Examples section of the Altera web site at **www.altera.com.**

Arithmetic Functions

Arithmetic functions, such as trigonometric functions, including sine, cosine, magnitude and phase calculation, are important DSP elements. This section discusses the implementation of a simple vector magnitude function in a Stratix device.

Background

Complex numbers can be expressed in two parts: real and imaginary.

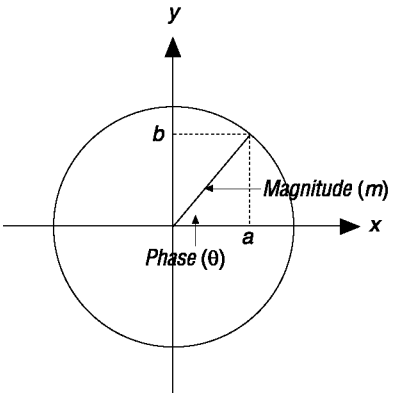
$$z = a + jb$$

where:

a is the real part
b is the imaginary part
$j^2 = -1$

In a two-dimensional plane, a vector (a,b) with reference to the origin (0,0) can also be represented as a complex number. In essence, the x-axis represents the real part, and the y-axis represents the imaginary part (see Figure 19–37).

Figure 19–37. Magnitude of Vector (a,b)



Complex numbers can be converted to phase and amplitude or magnitude representation, using a Cartesian-to-polar coordinate conversion. For a vector (a,b), the phase and magnitude representation is the following:

$$\text{Magnitude } m = \sqrt{a^2 + b^2}$$

$$\text{Phase angle } \theta = \tan^{-1}(b/a)$$

This conversion is useful in different applications, such as position control and position monitoring in robotics. It is also important to have these transformations at very high speeds to accommodate real-time processing.

Arithmetic Function Implementation

A common approach to implementing these arithmetic functions is using the coordinate rotation digital computer (CORDIC) algorithm. The CORDIC algorithm calculates the trigonometric functions of sine, cosine, magnitude, and phase using an iterative process. It is made up of a series of micro-rotations of the vector by a set of predetermined constants, which are powers of 2.

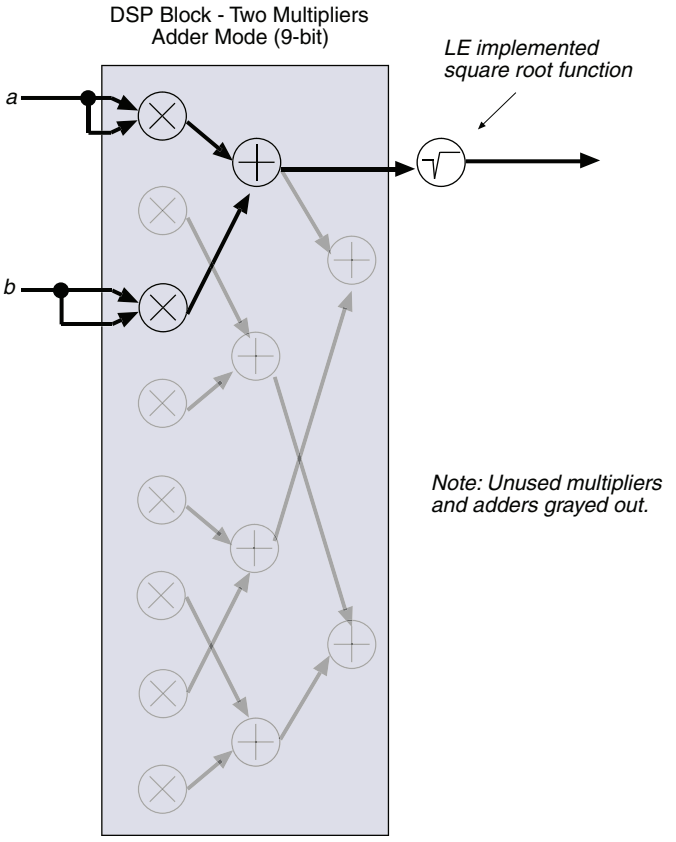
Using binary arithmetic, this algorithm essentially replaces multipliers with shift and add operations. In Stratix devices, it is possible to calculate some of these arithmetic functions directly, without having to implement the CORDIC algorithm.

This section describes a design example that calculates the magnitude of a 9-bit signed vector (a,b) using a pipelined version of the square root function available at the Altera IP Megastore. To calculate the sum of the squares of the input ($a^2 + b^2$), configure the DSP block in the two-multipliers adder mode. The square root function is implemented using an iterative algorithm similar to the long division operation. The binary numbers are paired off, and subtracted by a trial number. Depending on if the remainder is positive or negative, each bit of the square root is determined and the process is repeated. This square root function does not require memory and is implemented in logic cells only.

In this example, the input bit precision (IN_PREC) feeding into the square root macro is set to twenty, and the output precision (OUT_PREC) is set to ten. The number of precision bits is parameterizable. Also, there is a third parameter, PIPELINE, which controls the architecture of the square root macro. If this parameter is set to YES, it includes pipeline stages in the square root macro. If set to NO, the square root macro becomes a single-cycled combinatorial function.

Figure 19–38 shows the implementation the magnitude design.

Figure 19–38. Implementing the Vector Magnitude Function



Arithmetic Function Implementation Results

Table 19–19 shows the results of the implementation shown in Figure 19–38 with the PIPELINE parameter set to YES. Table 19–20 shows the results of the implementation shown in Figure 19–38 with the PIPELINE parameter set to NO.

Table 19–19. Vector Magnitude Function Implementation Results (PIPELINE=YES)

Part	EP1S10F780
Utilization	Lcell: 497/10570 (4%) DSP block 9-bit elements: 2/48 (4%) Memory bits: 0/920448 (0%)
Performance	194 MHz
Latency	15 clock cycles

Table 19–20. Vector Magnitude Function Implementation Results (PIPELINE=NO)

Part	EP1S10F780
Utilization	Lcell: 244/10570 (2%) DSP block 9-bit elements: 2/48 (4%) Memory bits: 0/920448 (0%)
Performance	30 MHz
Latency	3 clock cycles

Arithmetic Function Design Example

Download the Vector Magnitude Function (**magnitude.zip**) design example from the Design Examples section of the Altera web site at **www.altera.com.**

Conclusion

The DSP blocks in Stratix and Stratix GX devices are optimized to support DSP functions requiring high data throughput, such as FIR filters, IIR filters and the DCT. The DSP blocks are flexible and configurable in different operation modes based on the application's needs. The TriMatrix memory provides the data storage capability often needed in DSP applications.

The DSP blocks and TriMatrix memory in Stratix and Stratix GX devices offer performance and flexibility that translates to higher performance DSP functions.

References

See the following for more information:

- *Optimal DCT for Hardware Implementation*
  M. Langhammer. Proceedings of International Conference on Signal Processing Applications & Technology (ICSPAT) '95, October 1995
- *Digital Signal Processing: Principles, Algorithms, and Applications*
  John G. Proakis, Dimitris G. Manolakis. Prentice Hall
- *Hardware Implementation of Multirate Digital Filters*
  Tony San. Communication Systems Design, April 2000
- *AN 73: Implementing FIR Filters in FLEX Devices*
- *Efficient Logic Synthesis Techniques for IIR Filters*
  M.Langhammer. Proceedings of International Conference on Signal Processing Applications & Technology (ICSPAT) '95, October 1995



Stratix GX Device Handbook, Volume 3



101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com

SGX5V3-1.2

Copyright © 2006 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Altera Lead-Free

I.S. EN ISO 9001

Contents

Chapter Revision Dates vii

About This Handbook ix
How to Contact Altera ix
Typographic Conventions ix

Section I. Configuration & Remote System Upgrades
Revision History ..... Section I–1

Chapter 1. Configuring Stratix & Stratix GX Devices
Introduction 1–1
Device Configuration Overview 1–2
MSEL[2..0] Pins 1–3
V_{CCSEL} Pins 1–3
PORSEL Pins 1–5
nIO_PULLUP Pins 1–5
TDO & nCEO Pins 1–6
Configuration File Size 1–6
Altera Configuration Devices 1–7
Configuration Schemes 1–7
PS Configuration 1–7
FPP Configuration 1–21
PPA Configuration 1–30
JTAG Programming & Configuration 1–36
JTAG Programming & Configuration of Multiple Devices 1–39
Configuration with JRunner Software Driver 1–41
Jam STAPL Programming & Test Language 1–42
Configuring Using the MicroBlaster Driver 1–51
Device Configuration Pins 1–51

Chapter 2. Remote System Configuration with Stratix & Stratix GX Devices
Introduction 2–1
Remote Configuration Operation 2–1
Remote System Configuration Modes 2–3
Remote System Configuration Components 2–5
Quartus II Software Support 2–12
altremote_update Megafunction 2–14
Remote Update WYSIWYG ATOM 2–17

Using Enhanced Configuration Devices 2–19
Local Update Programming File Generation 2–21
Remote Update Programming File Generation 2–32
Combining MAX Devices & Flash Memory 2–42
Using an External Processor 2–43
Conclusion 2–44

Section II. Design Guidelines

Revision History ..... Section II–1

Chapter 3. Transitioning APEX Designs to Stratix & Stratix GX Devices

Introduction 3–1
General Architecture 3–1
Logic Elements 3–2
MultiTrack Interconnect 3–3
DirectDrive Technology 3–4
Architectural Element Names 3–5
TriMatrix Memory 3–8
Same-Port Read-During-Write Mode 3–10
Mixed-Port Read-During-Write Mode 3–11
Memory Megafunctions 3–12
FIFO Conditions 3–13
Design Migration Mode in Quartus II Software 3–13
DSP Block 3–16
DSP Block Megafunctions 3–16
PLLs & Clock Networks 3–18
Clock Networks 3–18
PLLs 3–19
I/O Structure 3–25
External RAM Interfacing 3–25
I/O Standard Support 3–26
High-Speed Differential I/O Standards 3–26
altlvds Megafunction 3–29
Configuration 3–30
Configuration Speed & Schemes 3–30
Remote Update Configuration 3–31
JTAG Instruction Support 3–31
Conclusion 3–32

Chapter 4. Stratix GX Board Design Guidelines

Introduction 4–1
Board Design Overview 4–2
Support Circuitry Design 4–3
Clock Circuitry 4–4
Isolated Power & Ground Plane Design 4–6

Power Circuitry 4–6
Decoupling Circuitry Design 4–13
Plane Capacitance 4–21
Plane & Island Design 4–24
Transmission Lines 4–33
Transmission Line Topologies 4–33
Transmission Line Termination 4–52
Transmission Line Routing 4–58
Other Transmission Line Issues 4–64
Miscellaneous 4–76
Component Selection for High-Speed Design 4–76
S-Parameters 4–79
Smith Chart 4–81
AC Versus DC Coupling 4–82
Unused Pin Connections 4–84
Power Trace Thickness 4–84

Chapter 5. Quartus II Software Fitter Warnings

Suppressing Fitter Warnings 5–5
Design Suggestions 5–5

Chapter Revision Dates

The chapters in this book, *Stratix GX Device Handbook, Volume 3*, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1. Configuring Stratix & Stratix GX Devices
Revised: *July 2005*
Part number: *S52013-3.2*

Chapter 2. Remote System Configuration with Stratix & Stratix GX Devices
Revised: *September 2004*
Part number: *S52015-3.1*

Chapter 3. Transitioning APEX Designs to Stratix & Stratix GX Devices
Revised: *July 2005*
Part number: *S52012-3.0*

Chapter 4. Stratix GX Board Design Guidelines
Revised: *February 2005*
Part number: *SGX53001-1.0*

Chapter 5. Quartus II Software
Fitter Warnings
Revised: *March 2005*
Part number: *SGX53002-1.0*

About This Handbook

This handbook provides comprehensive information about the Altera® Stratix® GX family of devices.

How to Contact Altera

For the most up-to-date information about Altera products, go to the Altera world-wide web site at www.altera.com. For technical support on this product, go to www.altera.com/mysupport. For additional information about Altera products, consult the sources shown below.

Information Type	USA & Canada	All Other Locations
Technical support	www.altera.com/mysupport/	www.altera.com/mysupport/
	(800) 800-EPLD (3753) (7:00 a.m. to 5:00 p.m. Pacific Time)	+1 408-544-8767 7:00 a.m. to 5:00 p.m. (GMT -8:00) Pacific Time
Product literature	www.altera.com	www.altera.com
Altera literature services	literature@altera.com	literature@altera.com
Non-technical customer service	(800) 767-3753	+ 1 408-544-7000 7:00 a.m. to 5:00 p.m. (GMT -8:00) Pacific Time
FTP site	ftp.altera.com	ftp.altera.com

Typographic Conventions

This document uses the typographic conventions shown below.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: **Save As** dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: $\mathbf{f_{MAX}}$, **\qdesigns** directory, **d:** drive, **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Document titles are shown in italic type with initial capital letters. Example: *AN 75: High-Speed Board Design.*

Visual Cue	Meaning
Italic type	Internal timing parameters and variables are shown in italic type. Examples: t_{PIA}, $n + 1$. Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: *<file name>*, *<project name>*.**pof** file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
"Subheading Title"	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: "Typographic Conventions."
`Courier type`	Signal and port names are shown in lowercase Courier type. Examples: `data1`, `tdi`, `input.` Active-low signals are denoted by suffix `n`, e.g., `resetn`. Anything that must be typed exactly as it appears is shown in Courier type. For example: `c:\qdesigns\tutorial\chiptrip.gdf`. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword `SUBDESIGN`), as well as logic function names (e.g., `TRI`) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ● •	Bullets are used in a list of items when the sequence of the items is not important.
✓	The checkmark indicates a procedure that consists of one step only.
☞	The hand points to information that requires special attention.
CAUTION	The caution indicates required information that needs special consideration and understanding and should be read prior to starting or continuing with the procedure or process.
	The warning indicates information that should be read prior to starting or continuing the procedure or processes
↵	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information on a particular topic.

Section I. Configuration & Remote System Upgrades

This section provides information on Stratix® and Stratix GX device configuration and remote system upgrades. It also provides configuration information for the supported configuration schemes in Stratix and Stratix GX devices.

This section includes the following chapters:

- Chapter 1, Configuring Stratix & Stratix GX Devices
- Chapter 2, Remote System Configuration with Stratix & Stratix GX Devices

Revision History

The table below shows the revision history for Chapters 1 and 2.

Chapter(s)	Date / Version	Changes Made
1	July 2005, v3.2	Updated as part of the *Stratix Device Handbook* update.
	September 2004 v3.1	Added chapter to *Stratix GX Device Handbook*.
2	September 2004 v3.1	Added chapter to *Stratix GX Device Handbook*.

1. Configuring Stratix & Stratix GX Devices

S52013-3.2

Introduction

You can configure Stratix® and Stratix GX devices using one of several configuration schemes. All configuration schemes use either a microprocessor, configuration device, or a download cable. See Table 1–1.

Table 1–1. Stratix & Stratix GX Device Configuration Schemes

Configuration Scheme	Typical Use
Fast passive parallel (FPP)	Configuration with a parallel synchronous configuration device or microprocessor interface where eight bits of configuration data are loaded on every clock cycle.
Passive serial (PS)	Configuration with a serial synchronous microprocessor interface or the MasterBlaster™ communications cable, USB Blaster, ByteBlaster™ II, or ByteBlasterMV parallel port download cable.
Passive parallel asynchronous (PPA)	Configuration with a parallel asynchronous microprocessor interface. In this scheme, the microprocessor treats the target device as memory.
Remote/local update FPP	Configuration using a Nios™ (16-bit ISA) and Nios® II (32-bit ISA) or other embedded processor. Allows you to update the Stratix or Stratix GX device configuration remotely using the FPP scheme to load data.
Remote/local update PS	Passive serial synchronous configuration using a Nios or other embedded processor. Allows you to update the Stratix or Stratix GX device configuration remotely using the PS scheme to load data.
Remote/local update PPA	Passive parallel asynchronous configuration using a Nios or other embedded processor. In this scheme, the Nios microprocessor treats the target device as memory. Allows you to update the Stratix or Stratix GX device configuration remotely using the PPA scheme to load data.
Joint Test Action Group (JTAG)	Configuration through the IEEE Std. 1149.1 JTAG pins. You can perform JTAG configuration with either a download cable or an embedded device. Ability to use SignalTap® II Embedded Logic Analyzer.

This chapter discusses how to configure one or more Stratix or Stratix GX devices. It should be used together with the following documents:

- *MasterBlaster Serial/USB Communications Cable Data Sheet*
- *USB Blaster USB Port Download Cable Development Tools Data Sheet*
- *ByteBlaster II Parallel Port Download Cable Data Sheet*
- *ByteBlasterMV Parallel Port Download Cable Data Sheets*
- *Configuration Devices for SRAM-Based LUT Devices Data Sheet*
- *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet*

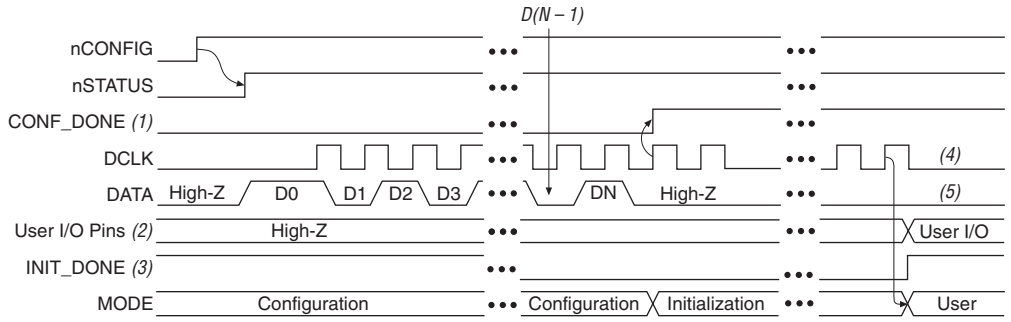
- The *Remote System Configuration with Stratix & Stratix GX Devices* chapter

For more information on setting device configuration options or generating configuration files, see the *Software Setting* chapter in *Volume 2* of the *Configuration Handbook*.

Device Configuration Overview

During device operation, the FPGA stores configuration data in SRAM cells. Because SRAM memory is volatile, you must load the SRAM cells with the configuration data each time the device powers up. After configuration, the device must initialize its registers and I/O pins. After initialization, the device enters user mode. Figure 1–1 shows the state of the device during the configuration, initialization, and user mode.

Figure 1–1. Stratix & Stratix GX Configuration Cycle



Notes to Figure 1–1:

(1) During initial power up and configuration, CONF_DONE is low. After configuration, CONF_DONE goes high. If the device is reconfigured, CONF_DONE goes low after nCONFIG is driven low.

(2) User I/O pins are tri-stated during configuration. Stratix and Stratix GX devices also have a weak pull-up resistor on I/O pins during configuration that are enabled by nIO_PULLUP. After initialization, the user I/O pins perform the function assigned in the user's design.

(3) If the INIT_DONE pin is used, it will be high because of an external 10 kΩ resistor pull-up when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low.

(4) DCLK should not be left floating. It should be driven high or low.

(5) DATA0 should not be left floating. It should be driven high or low.

You can load the configuration data for the Stratix or Stratix GX device using a passive configuration scheme. When using any passive configuration scheme, the Stratix or Stratix GX device is incorporated into a system with an intelligent host, such as a microprocessor, that controls the configuration process. The host supplies configuration data from a storage device (e.g., a hard disk, RAM, or other system memory). When using passive configuration, you can change the target device's

functionality while the system is in operation by reconfiguring the device. You can also perform in-field upgrades by distributing a new programming file to system users.

The following sections describe the MSEL[2..0], VCCSEL, PORSEL, and nIO_PULLUP pins used in Stratix and Stratix GX device configuration.

MSEL[2..0] Pins

You can select a Stratix or Stratix GX device configuration scheme by driving its MSEL2, MSEL1, and MSEL0 pins either high or low, as shown in Table 1–2.

Table 1–2. Stratix & Stratix GX Device Configuration Schemes

Description	MSEL2	MSEL1	MSEL0
FPP configuration	0	0	0
PPA configuration	0	0	1
PS configuration	0	1	0
Remote/local update FPP *(1)*	1	0	0
Remote/local update PPA *(1)*	1	0	1
Remote/local update PS *(1)*	1	1	0
JTAG-based configuration *(3)*	*(2)*	*(2)*	*(2)*

Notes to Table 1–2:

(1) These schemes require that you drive a secondary pin RUnLU to specify whether to perform a remote update or local update.

(2) Do not leave MSEL pins floating. Connect them to V_{CCIO} or GND. These pins support the non-JTAG configuration scheme used in production. If only JTAG configuration is used you should connect the MSEL pins to ground.

(3) JTAG-based configuration takes precedence over other configuration schemes, which means the MSEL pins are ignored.

The MSEL[] pins can be tied to V_{CCIO} of the I/O bank they reside in or ground.

V_{CCSEL} Pins

You can configure Stratix and Stratix GX devices using the 3.3-, 2.5-, 1.8-, or 1.5-V LVTTL I/O standard on configuration and JTAG input pins. VCCSEL is a dedicated input on Stratix and Stratix GX devices that selects between 3.3-V/2.5-V input buffers and 1.8-V/1.5-V input buffers for dedicated configuration input pins. A logic low supports 3.3-V/2.5-V signaling, and a logic high supports 1.8-V/1.5-V signaling. A logic high can also support 3.3-V/2.5-V signaling. VCCSEL affects the configuration

related I/O banks (3, 4, 7, and 8) where the following pins reside: TDI, TMS, TCK, TRST, MSEL0, MSEL1, MSEL2, nCONFIG, nCE, DCLK, PLL_ENA, CONF_DONE, nSTATUS. The VCCSEL pin can be pulled to 1.5, 1.8, 2.5, or 3.3-V for a logic high level. There is an internal 2.5-kΩ pull-down resistor on VCCSEL. Therefore, if you are using a pull-up resister to pull up this signal, you need to use a 1-kΩ resistor.

VCCSEL also sets the power-on-reset (POR) trip point for all the configuration related I/O banks (3, 4, 7, and 8), ensuring that these I/O banks have powered up to the appropriate voltage levels before configuration begins. Upon power-up, the FPGA does not release nSTATUS until V_{CCINT} and all of the V_{CCIO}s of the configuration I/O banks are above their POR trip points. If you set VCCSEL to ground (logic low), this sets the POR trip point for all configuration I/O banks to a voltage consistent with 3.3-V/2.5-V signaling. When VCCSEL = 0, the POR trip point for these I/O banks may be as high as 1.8 V. If V_{CCIO} of any of the configuration banks is set to 1.8 or 1.5 V, the voltage supplied to this I/O bank(s) may never reach the POR trip point, which will not allow the FPGA to begin configuration.



If the V_{CCIO} of I/O banks 3, 4, 7, or 8 is set to 1.5 or 1.8 V and the configuration signals used require 3.3-V or 2.5-V signaling you should set VCCSEL to V_{CC} (logic high) in order to lower the POR trip point to enable successful configuration.

Table 1–3 shows how you should set the VCCSEL depending on the V_{CCIO} setting of the configuration I/O banks and your configuration input signaling voltages.

Table 1–3. VCCSEL Setting

V_{CCIO} (banks 3,4,7,8)	Configuration Input Signaling Voltage	V_{CCSEL}
3.3-V/2.5-V	3.3-V/2.5-V	GND
1.8-V/1.5-V	3.3-V/2.5-V/1.8-V/1.5-V	VCC
3.3-V/2.5-V	1.8-V/1.5-V	Not Supported

The VCCSEL signal does not control any of the dual-purpose pins, including the dual-purpose configuration pins, such as the DATA[7..0] and PPA pins (nWS, nRS, CS, nCS, and RDYnBSY). During configuration, these dual-purpose pins drive out voltage levels corresponding to the V_{CCIO} supply voltage that powers the I/O bank containing the pin. After configuration, the dual-purpose pins inherit the I/O standards specified in the design.

PORSEL Pins

PORSEL is a dedicated input pin used to select POR delay times of 2 ms or 100 ms during power-up. When the PORSEL pin is connected to ground, the POR time is 100 ms; when the PORSEL pin is connected to VCC, the POR time is 2 ms. There is an internal 2.5-kΩ pull-down resistor on PORSEL. Therefore if you are using a pull-up resistor to pull up this signal, you need to use a 1-kΩ resistor.

When using enhanced configuration devices to configure Stratix devices, make sure that the PORSEL setting of the Stratix device is the same or faster than the PORSEL setting of the enhanced configuration device. If the FPGA is not powered up after the enhanced configuration device exits POR, the CONF_DONE signal will be high since the pull-up resistor is pulling this signal high. When the enhanced configuration device exits POR, OE of the enhanced configuration device is released and pulled high by a pull-up resistor. Since the enhanced configuration device sees its nCS/CONF_DONE signal also high, it enters a test mode. Therefore, you must ensure the FPGA powers up before the enhanced configuration device exits POR.

For more margin, the 100-ms setting can be selected when using an enhanced configuration device to allow the Stratix FPGA to power-up before configuration is attempted (see Table 1–4).

Table 1–4. PORSEL Settings

PORSEL Settings	POR Time (ms)
GND	100
V_{CC}	2

nIO_PULLUP Pins

The nIO_PULLUP pin enables a built-in weak pull-up resistor to pull all user I/O pins to VCCIO before and during device configuration. If nIO_PULLUP is connected to V_{CC} during configuration, the weak pull-ups on all user I/O pins and all dual-purpose pins are disabled. If connected to ground, the pull-ups are enabled during configuration. The nIO_PULLUP pin can be pulled to 1.5, 1.8, 2.5, or 3.3-V for a logic level high. There is an internal 2.5-kΩ pull-down resistor on nIO_PULLUP. Therefore, if you are using a pull-up resistor to pull up this signal, you need to use a 1-kΩ resistor.

TDO & nCEO Pins

TDO and nCEO pins drive out the same voltage levels as the V_{CCIO} that powers the I/O bank where the pin resides. You must select the V_{CCIO} supply for the bank containing TDO accordingly. For example, when using the ByteBlasterMV cable, the V_{CCIO} for the bank containing TDO must be powered up at 3.3-V. The current strength for TDO is 12 mA.

Configuration File Size

Tables 1–5 and 1–6 summarize the approximate configuration file size required for each Stratix and Stratix GX device. To calculate the amount of storage space required for multi-device configurations, add the file size of each device together.

Table 1–5. Stratix Configuration File Sizes

Device	Raw Binary File (.rbf) Size (Bits)
EP1S10	3,534,640
EP1S20	5,904,832
EP1S25	7,894,144
EP1S30	10,379,368
EP1S40	12,389,632
EP1S60	17,543,968
EP1S80	23,834,032

Table 1–6. Stratix GX Configuration File Sizes

Device	Raw Binary File Size (Bits)
EP1SGX10C	3,579,928
EP1SGX10D	3,579,928
EP1SGX25C	7,951,248
EP1SGX25D	7,951,248
EP1SGX25F	7,951,248
EP1SGX40D	12,531,440
EP1SGX40G	12,531,440

You should only use the numbers in Tables 1–5 and 1–6 to estimate the file size before design compilation. The exact file size may vary because different Altera® Quartus® II software versions may add a slightly

different number of padding bits during programming. However, for any specific version of the Quartus II software, any design targeted for the same device has the same configuration file size.

Altera Configuration Devices

The Altera enhanced configuration devices (EPC16, EPC8, and EPC4 devices) support a single-device configuration solution for high-density FPGAs and can be used in the FPP and PS configuration schemes. They are ISP-capable through its JTAG interface. The enhanced configuration devices are divided into two major blocks, the controller and the flash memory.



For information on enhanced configuration devices, see the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* and the *Using Altera Enhanced Configuration Devices* chapter in the *Configuration Handbook*.

The EPC2 and EPC1 configuration devices provide configuration support for the PS configuration scheme. The EPC2 device is ISP-capable through its JTAG interface. The EPC2 and EPC1 can be cascaded to hold large configuration files.



For more information on EPC2, EPC1, and EPC1441 configuration devices, see the *Configuration Devices for SRAM-Based LUT Devices Data Sheet*.

Configuration Schemes

This section describes how to configure Stratix and Stratix GX devices with the following configuration schemes:

- PS Configuration with Configuration Devices
- PS Configuration with a Download Cable
- PS Configuration with a Microprocessor
- FPP Configuration
- PPA Configuration
- JTAG Programming & Configuration
- JTAG Programming & Configuration of Multiple Devices

PS Configuration

PS configuration of Stratix and Stratix GX devices can be performed using an intelligent host, such as a MAX® device, microprocessor with flash memory, an Altera configuration device, or a download cable. In the PS scheme, an external host (MAX device, embedded processor, configuration device, or host PC) controls configuration. Configuration data is clocked into the target Stratix devices via the `DATA0` pin at each rising edge of `DCLK`.

PS Configuration with Configuration Devices

The configuration device scheme uses an Altera configuration device to supply data to the Stratix or Stratix GX device in a serial bitstream (see Figure 1–3).

In the configuration device scheme, nCONFIG is usually tied to V_{CC} (when using EPC16, EPC8, EPC4, or EPC2 devices, nCONFIG may be connected to nINIT_CONF). Upon device power-up, the target Stratix or Stratix GX device senses the low-to-high transition on nCONFIG and initiates configuration. The target device then drives the open-drain CONF_DONE pin low, which in-turn drives the configuration device's nCS pin low. When exiting power-on reset (POR), both the target and configuration device release the open-drain nSTATUS pin.

Before configuration begins, the configuration device goes through a POR delay of up to 200 ms to allow the power supply to stabilize (power the Stratix or Stratix GX device before or during the POR time of the configuration device). This POR delay has a maximum of 200 ms for EPC2 devices. For enhanced configuration devices, you can select between 2 ms and 100 ms by connecting PORSEL pin to VCC or GND, accordingly. During this time, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target device's nSTATUS pin. When the target and configuration devices complete POR, they release nSTATUS, which is then pulled high by a pull-up resistor.

When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. When all devices are ready, the configuration device clocks data out serially to the target devices using an internal oscillator.

After successful configuration, the Stratix FPGA starts initialization using the 10-MHz internal oscillator as the reference clock. After initialization, this internal oscillator is turned off. The CONF_DONE pin is released by the target device and then pulled high by a pull-up resistor. When initialization is complete, the FPGA enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

If an error occurs during configuration, the target device drives its nSTATUS pin low, resetting itself internally and resetting the configuration device. If the **Auto-Restart Configuration on Frame Error** option—available in the Quartus II **Global Device Options** dialog box (Assign menu)—is turned on, the device reconfigures automatically if an error occurs. To find this option, choose **Compiler Settings** (Processing menu), then click on the **Chips & Devices** tab.

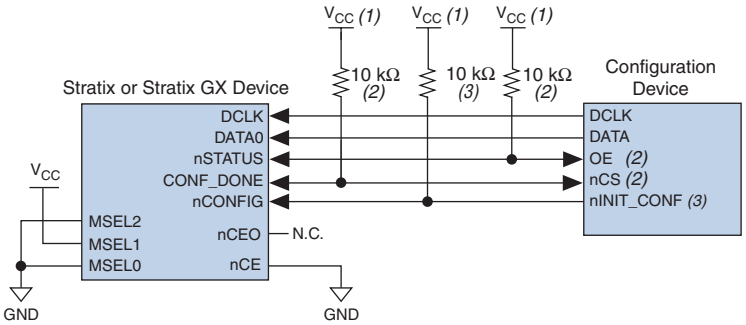
If this option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low to restart configuration. The external system can pulse nCONFIG if it is under system control rather than tied to V_{CC}. When configuration is complete, the target device releases CONF_DONE, which disables the configuration device by driving nCS high. The configuration device drives DCLK low before and after configuration.

In addition, if the configuration device sends all of its data and then detects that CONF_DONE has not gone high, it recognizes that the target device has not configured successfully. In this case, the configuration device pulses its OE pin low for a few microseconds, driving the target device's nSTATUS pin low. If the **Auto-Restart Configuration on Frame Error** option is set in the software, the target device resets and then pulses its nSTATUS pin low. When nSTATUS returns high, the configuration device reconfigures the target device. When configuration is complete, the configuration device drives DCLK low.

Do not pull CONF_DONE low to delay initialization. Instead, use the Quartus II software's **Enable User-Supplied Start-Up Clock (CLKUSR)** option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain initialize together. When CONF_DONE is driven low after device configuration, the configuration device recognizes that the target device has not configured successfully.

Figure 1–2 shows how to configure one Stratix or Stratix GX device with one configuration device.

Figure 1–2. Single Device Configuration Circuit



Notes to Figure 1–2:

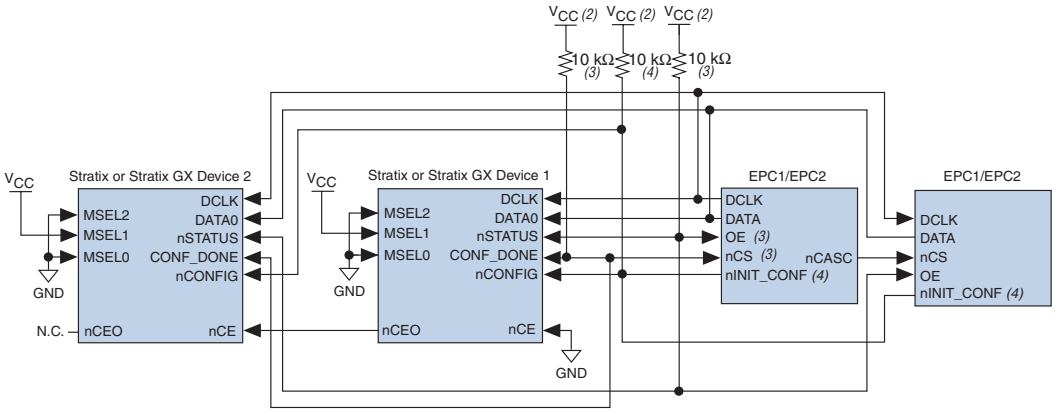
(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The enhanced configuration devices and EPC2 devices have internal programmable pull-ups on OE and nCS. You should only use the internal pull-ups of the configuration device if the nSTATUS and CONF_DONE signals are pulled up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V). If external pull-ups are used, they should be 10 kΩ

(3) The nINIT_CONF pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} through a resistor. he nINIT_CONF pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the nINIT_CONF pin.

Figure 1–3 shows how to configure multiple Stratix and Stratix GX devices with multiple EPC2 or EPC1 configuration devices.

Figure 1–3. Multi-Device Configuration Circuit Note (1)



Notes to Figure 1–3:

(1) When performing multi-device active serial configuration, you must generate the configuration device programmer object file (**.pof**) from each project's SOF. You can combine multiple SOFs using the Quartus II software through the **Device & Pin Option** dialog box. For more information on how to create configuration and programming files, see the *Software Settings* section in the *Configuration Handbook, Volume 2*.

(2) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(3) The enhanced configuration devices and EPC2 devices have internal programmable pull-ups on OE and nCS. You should only use the internal pull-ups of the configuration device if the nSTATUS and CONF_DONE signals are pulled up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V). If external pull-ups are used, they should be 10 kΩ

(4) The nINIT_CONF pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} through a resistor. The nINIT_CONF pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the nINIT_CONF pin.

After the first Stratix or Stratix GX device completes configuration during multi-device configuration, its nCEO pin activates the second device's nCE pin, prompting the second device to begin configuration. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

In addition, all nSTATUS pins are tied together; thus, if any device (including the configuration devices) detects an error, configuration stops for the entire chain. Also, if the first configuration device does not detect CONF_DONE going high at the end of configuration, it resets the chain by pulsing its OE pin low for a few microseconds. This low pulse drives the OE pin low on the second configuration device and drives nSTATUS low on all Stratix and Stratix GX devices, causing them to enter an error state.

If the **Auto-Restart Configuration on Frame Error** option is turned on in the software, the Stratix or Stratix GX device releases its nSTATUS pins after a reset time-out period. When the nSTATUS pins are released and pulled high, the configuration devices reconfigure the chain. If the **Auto-**

Restart Configuration on Frame Error option is not turned on, the Stratix or Stratix GX devices drive nSTATUS low until they are reset with a low pulse on nCONFIG.

You can also cascade several EPC2/EPC1 configuration devices to configure multiple Stratix and Stratix GX devices. When all data from the first configuration device is sent, it drives nCASC low, which in turn drives nCS on the subsequent configuration device. Because a configuration device requires less than one clock cycle to activate a subsequent configuration device, the data stream is uninterrupted.

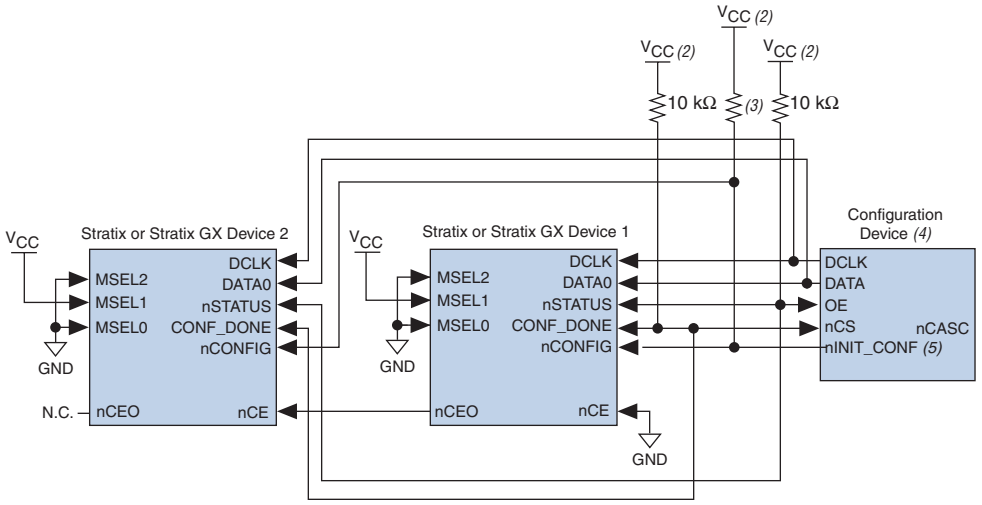


You cannot cascade enhanced (EPC16, EPC8, and EPC4) configuration devices.

You can use a single configuration chain to configure multiple Stratix and Stratix GX devices. In this scheme, the nCEO pin of the first device is connected to the nCE pin of the second device in the chain. If there are additional devices, connect the nCE pin of the next device to the nCEO pin of the previous device. To configure properly, all of the device CONF_DONE and nSTATUS pins must be tied together.

Figure 1–4 shows an example of configuring multiple Stratix and Stratix GX devices using a configuration device.

Figure 1–4. Configuring Multiple Stratix & Stratix GX Devices with A Single Configuration Device Note (1)



Notes to Figure 1–4:

(1) When performing multi-device active serial configuration, you must generate the configuration device programmer object file (**.pof)** from each project's SOF. You can combine multiple SOFs using the Quartus II software through the **Device & Pin Option** dialog box. For more information on how to create configuration and programming files, see the *Software Settings* section in the *Configuration Handbook, Volume 2*.

(2) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(3) The enhanced configuration devices and EPC2 devices have internal programmable pull-ups on `OE` and `nCS`. You should only use the internal pull-ups of the configuration device if the `nSTATUS` and `CONF_DONE` signals are pulled up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V). If external pull-ups are used, they should be 10 kΩ

(4) EPC16, EPC8, and EPC4 configuration devices cannot be cascaded.

(5) The `nINIT_CONF` pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If `nINIT_CONF` is not used, `nCONFIG` must be pulled to V_{CC} through a resistor. The `nINIT_CONF` pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the `nINIT_CONF` pin.

Table 1–7 shows the status of the device DATA pins during and after configuration.

Table 1–7. DATA Pin Status Before & After Configuration

Pins	Stratix or Stratix GX Device	
	During	After
DATA0 *(1)*	Used for configuration	User defined
DATA[7..1] *(2)*	Used in some configuration modes	User defined
I/O Pins	Tri-state	User defined

Notes to Table 1–7:

(1) The status shown is for configuration with a configuration device.

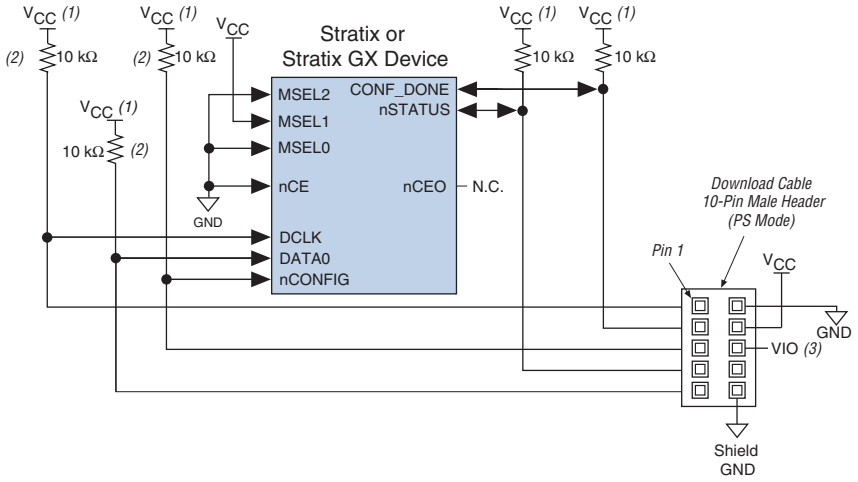
(2) The function of these pins depends upon the settings specified in the Quartus II software using the **Device & Pin Option** dialog box (see the *Software Settings* section in the *Configuration Handbook, Volume 2*, and the Quartus II Help software for more information).

PS Configuration with a Download Cable

In PS configuration with a download cable, an intelligent host transfers data from a storage device to the Stratix or Stratix GX device through the MasterBlaster, USB-Blaster, ByteBlaster II or ByteBlasterMV cable. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin. The programming hardware then places the configuration data one bit at a time on the device's DATA0 pin. The data is clocked into the target device until CONF_DONE goes high. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

When using programming hardware for the Stratix or Stratix GX device, turning on the **Auto-Restart Configuration on Frame Error** option does not affect the configuration cycle because the Quartus II software must restart configuration when an error occurs. Additionally, the **Enable User-Supplied Start-Up Clock (CLKUSR)** option has no affect on the device initialization since this option is disabled in the SOF when programming the FPGA using the Quartus II software programmer and a download cable. Therefore, if you turn on the **CLKUSR** option, you do not need to provide a clock on CLKUSR when you are configuring the FPGA with the Quartus II programmer and a download cable. Figure 1–5 shows PS configuration for the Stratix or Stratix GX device using a MasterBlaster, USB-Blaster, ByteBLaster II or ByteBlasterMV cable.

Figure 1–5. PS Configuration Circuit with a Download Cable



Notes to Figure 1–5:

(1) You should connect the pull-up resistor to the same supply voltage as the MasterBlaster (VIO pin) or ByteBlasterMV cable.

(2) The pull-up resistors on the DATA0 and DCLK pins are only needed if the download cable is the only configuration scheme used on the board. This is to ensure that the DATA0 and DCLK pins are not left floating after configuration. For example, if the design also uses a configuration device, the pull-up resistors on the DATA0 and DCLK pins are not necessary.

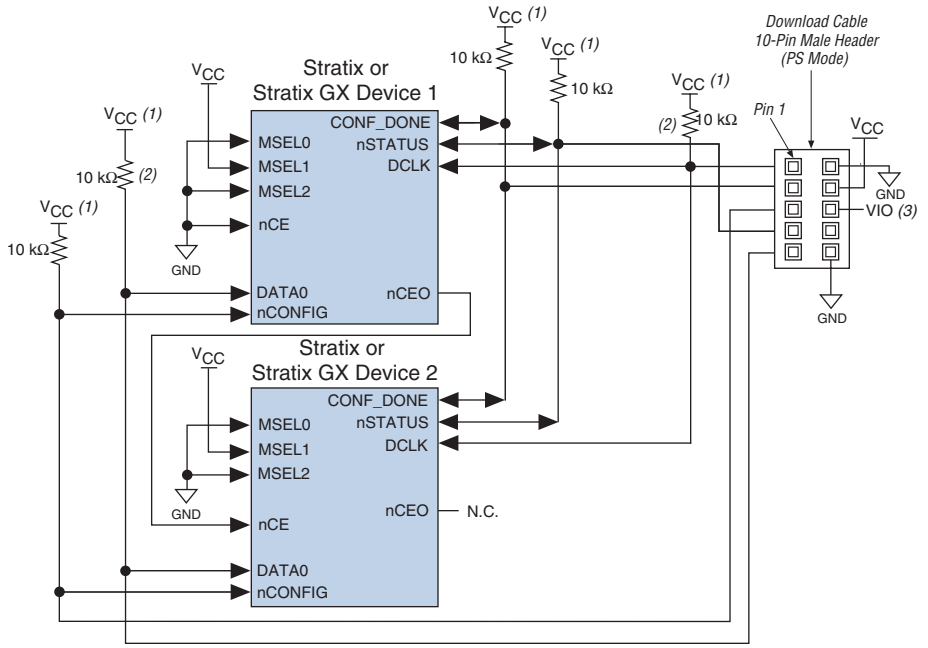
(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. This pin is a no-connect pin for the ByteBlasterMV header.

You can use programming hardware to configure multiple Stratix and Stratix GX devices by connecting each device's nCEO pin to the subsequent device's nCE pin. All other configuration pins are connected to each device in the chain.

Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time. In addition, because the nSTATUS pins are tied together, the entire chain halts configuration if any device detects an error. In this situation, the Quartus II software must restart configuration; the **Auto-Restart Configuration on Frame Error** option does not affect the configuration cycle.

Figure 1–6 shows how to configure multiple Stratix and Stratix GX devices with a MasterBlaster or ByteBlasterMV cable.

Figure 1–6. Multi-Device PS Configuration with a Download Cable



Notes to Figure 1–6:

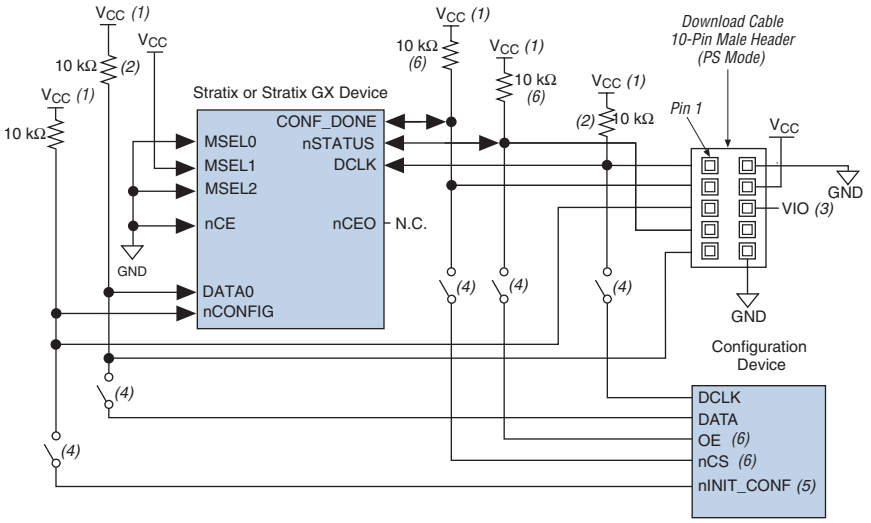
(1) You should connect the pull-up resistor to the same supply voltage as the MasterBlaster (VIO pin) or ByteBlasterMV cable.

(2) The pull-up resistors on the DATA0 and DCLK pins are only needed if the download cable is the only configuration scheme used on the board. This is to ensure that the DATA0 and DCLK pins are not left floating after configuration. For example, if the design also uses a configuration device, the pull-up resistors on the DATA0 and DCLK pins are not necessary.

(3) V_{IO} is a reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. See the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value.

If you are using a download cable to configure device(s) on a board that also has configuration devices, you should electrically isolate the configuration devices from the target device(s) and cable. One way to isolate the configuration devices is to add logic, such as a multiplexer, that can select between the configuration devices and the cable. The multiplexer device should allow bidirectional transfers on the nSTATUS and CONF_DONE signals. Another option is to add switches to the five common signals (CONF_DONE, nSTATUS, DCLK, nCONFIG, and DATA0) between the cable and the configuration devices. The last option is to remove the configuration devices from the board when configuring with the cable. Figure 1–7 shows a combination of a configuration device and a download cable to configure a Stratix or Stratix GX device.

Figure 1–7. Configuring with a Combined PS & Configuration Device Scheme



Notes to Figure 1–7:

(1) You should connect the pull-up resistor to the same supply voltage as the configuration device.

(2) The pull-up resistors on the DATA0 and DCLK pins are only needed if the download cable is the only configuration scheme used on the board. This is to ensure that the DATA0 and DCLK pins are not left floating after configuration. For example, if the design also uses a configuration device, the pull-up resistors on the DATA0 and DCLK pins are not necessary.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the target device's V_{CCIO}. This is a no-connect pin for the ByteBlasterMV header.

(4) You should not attempt configuration with a download cable while a configuration device is connected to a Stratix or Stratix GX device. Instead, you should either remove the configuration device from its socket when using the download cable or place a switch on the five common signals between the download cable and the configuration device. Remove the download cable when configuring with a configuration device.

(5) If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(6) If external pull-ups are used on CONF_DONE and nSTATUS pins, they should always be 10 kΩ resistors. You can use the internal pull-ups of the configuration device only if the CONF_DONE and nSTATUS signals are pulled-up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V).

For more information on how to use the MasterBlaster or ByteBlasterMV cables, see the following documents:

- *USB-Blaster USB Port Download Cable Data Sheet*
- *MasterBlaster Serial/USB Communications Cable Data Sheet*
- *ByteBlasterMV Parallel Port Download Cable Data Sheet*
- *ByteBlaster II Parallel Port Download Cable Data Sheet*

PS Configuration with a Microprocessor

In PS configuration with a microprocessor, a microprocessor transfers data from a storage device to the target Stratix or Stratix GX device. To initiate configuration in this scheme, the microprocessor must generate a low-to-high transition on the nCONFIG pin and the target device must release nSTATUS. The microprocessor or programming hardware then places the configuration data one bit at a time on the DATA0 pin of the Stratix or Stratix GX device. The least significant bit (LSB) of each data byte must be presented first. Data is clocked continuously into the target device until CONF_DONE goes high.

After all configuration data is sent to the Stratix or Stratix GX device, the CONF_DONE pin goes high to show successful configuration and the start of initialization. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. If you are using the **clkusr** option, after all data is transferred clkusr must be clocked an additional 136 times for the Stratix or Stratix GX device to initialize properly. Driving DCLK to the device after configuration is complete does not affect device operation.

Handshaking signals are not used in PS configuration modes. Therefore, the configuration clock speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists. You can pause configuration by halting DCLK for an indefinite amount of time.

If the target device detects an error during configuration, it drives its nSTATUS pin low to alert the microprocessor. The microprocessor can then pulse nCONFIG low to restart the configuration process. Alternatively, if the **Auto-Restart Configuration on Frame Error** option is turned on in the Quartus II software, the target device releases nSTATUS after a reset time-out period. After nSTATUS is released, the microprocessor can reconfigure the target device without needing to pulse nCONFIG low.

The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. If the microprocessor sends all data and the initialization clock starts but CONF_DONE and INIT_DONE have not gone high, it must reconfigure the target device. By default the INIT_DONE output is disabled. You can enable the INIT_DONE output by turning on **Enable INIT_DONE output** option in the Quartus II software.

If you do not turn on the **Enable INIT_DONE output** option in the Quartus II software, you are advised to wait for the maximum value of t_{CD2UM} (see Table 1–8) after the CONF_DONE signal goes high to ensure the device has been initialized properly and that it has entered user mode.

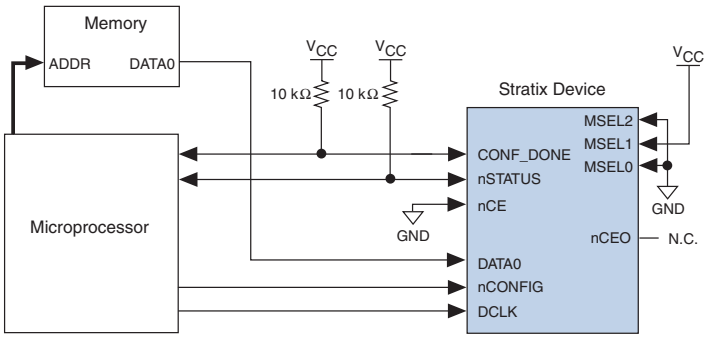
During configuration and initialization, and before the device enters user mode, the microprocessor must not drive the CONF_DONE signal low.



If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

Figure 1–8 shows the circuit for PS configuration with a microprocessor.

Figure 1–8. PS Configuration Circuit with Microprocessor



PS Configuration Timing

Figure 1–9 shows the PS configuration timing waveform for Stratix and Stratix GX devices. Table 1–8 shows the PS timing parameters for Stratix and Stratix GX devices.

Table 1–8. PS Timing Parameters for Stratix & Stratix GX Devices

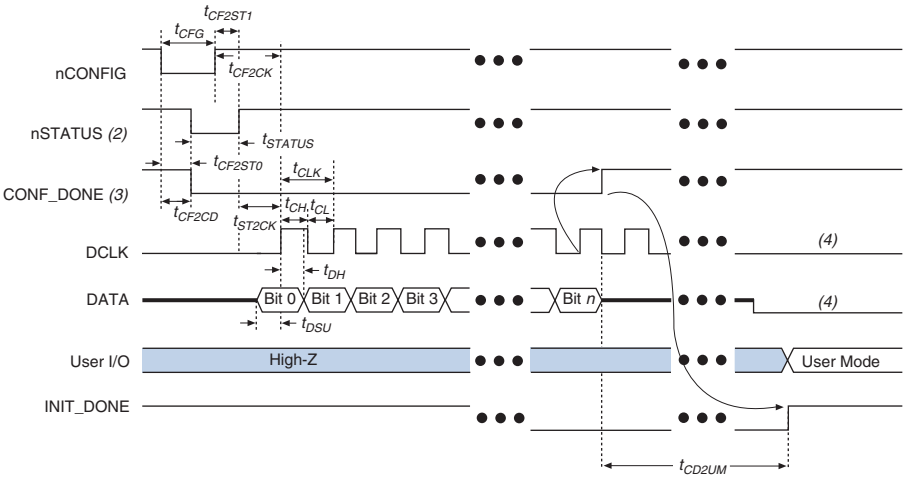
Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CF2ST1}	nCONFIG high to nSTATUS high		40 *(2)*	µs
t_{CFG}	nCONFIG low pulse width	40		µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(2)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	7		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH}	DCLK high time	4		ns
t_{CL}	DCLK low time	4		ns
t_{CLK}	DCLK period	10		ns
f_{MAX}	DCLK maximum frequency		100	MHz
t_{CD2UM}	CONF_DONE high to user mode *(1)*	6	20	µs

Notes to Table 1–8:

(1) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 136 to obtain this value.

(2) This value is obtainable if users do not delay configuration by extending the nSTATUS low pulse width.

Figure 1–9. PS Timing Waveform for Stratix & Stratix GX Devices Note (1)



Notes to Figure 1–9:

(1) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(2) Upon power-up, the Stratix II device holds nSTATUS low for the time of the POR delay.
(3) Upon power-up, before and during configuration, CONF_DONE is low.
(4) DCLK should not be left floating after configuration. It should be driven high or low, whichever is convenient. DATA[] is available as user I/Os after configuration and the state of these pins depends on the dual-purpose pin settings.

FPP Configuration

Parallel configuration of Stratix and Stratix GX devices meets the continuously increasing demand for faster configuration times. Stratix and Stratix GX devices can receive byte-wide configuration data per clock cycle, and guarantee a configuration time of less than 100 ms with a 100-MHz configuration clock. Stratix and Stratix GX devices support programming data bandwidth up to 800 megabits per second (Mbps) in this mode. You can use parallel configuration with an EPC16, EPC8, or EPC4 device, or a microprocessor.

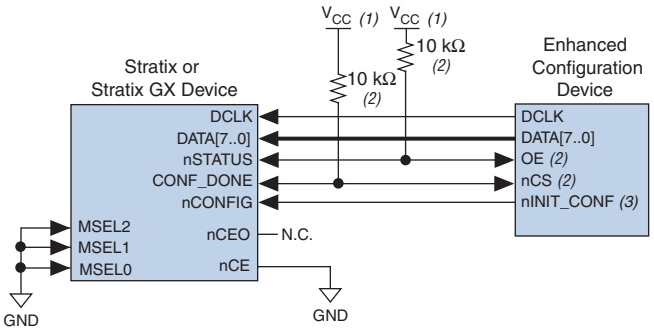
This section discusses the following schemes for FPP configuration in Stratix and Stratix GX devices:

- FPP Configuration Using an Enhanced Configuration Device
- FPP Configuration Using a Microprocessor

FPP Configuration Using an Enhanced Configuration Device

When using FPP with an enhanced configuration device, it supplies data in a byte-wide fashion to the Stratix or Stratix GX device every DCLK cycle. See Figure 1–10.

Figure 1–10. FPP Configuration Using Enhanced Configuration Devices



Notes to Figure 1–10:

(1) The pull-up resistors should be connected to the same supply voltage as the configuration device.

(2) The enhanced configuration devices and EPC2 devices have internal programmable pull-ups on OE and nCS. You should only use the internal pull-ups of the configuration device if the nSTATUS and CONF_DONE signals are pulled up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V). If external pull-ups are used, they should be 10 kΩ

(3) The nINIT_CONF pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} through a resistor. The nINIT_CONF pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the nINIT_CONF pin.

In the enhanced configuration device scheme, nCONFIG is tied to nINIT_CONF. On power up, the target Stratix or Stratix GX device senses the low-to-high transition on nCONFIG and initiates configuration. The target Stratix or Stratix GX device then drives the open-drain CONF_DONE pin low, which in-turn drives the enhanced configuration device's nCS pin low.

Before configuration starts, there is a 2-ms POR delay if the PORSEL pin is connected to V_{CC} in the enhanced configuration device. If the PORSEL pin is connected to ground, the POR delay is 100 ms. When each device determines that its power is stable, it releases its nSTATUS or OE pin. Because the enhanced configuration device's OE pin is connected to the target Stratix or Stratix GX device's nSTATUS pin, configuration is delayed until both the nSTATUS and OE pins are released by each device. The nSTATUS and OE pins are pulled up by a resistor on their respective

devices once they are released. When configuring multiple devices, connect the nSTATUS pins together to ensure configuration only happens when all devices release their OE or nSTATUS pins. The enhanced configuration device then clocks data out in parallel to the Stratix or Stratix GX device using a 66-MHz internal oscillator, or drives it to the Stratix or Stratix GX device through the EXTCLK pin.

If there is an error during configuration, the Stratix or Stratix GX device drives the nSTATUS pin low, resetting itself internally and resetting the enhanced configuration device. The Quartus II software provides an **Auto-restart configuration after error** option that automatically initiates the reconfiguration whenever an error occurs. See the *Software Settings* chapter in Volume 2 of the *Configuration Handbook* for information on how to turn this option on or off.

If this option is turned off, you must set monitor nSTATUS to check for errors. To initiate reconfiguration, pulse nCONFIG low. The external system can pulse nCONFIG if it is under system control rather than tied to V_{CC}. Therefore, nCONFIG must be connected to nINIT_CONF if you want to reprogram the Stratix or Stratix GX device on the fly.

When configuration is complete, the Stratix or Stratix GX device releases the CONF_DONE pin, which is then pulled up by a resistor. This action disables the EPC16, EPC8, or EPC4 enhanced configuration device as nCS is driven high. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. When initialization is complete, the Stratix or Stratix GX device enters user mode. The enhanced configuration device drives DCLK low before and after configuration.



CONF_DONE goes high one byte early in parallel synchronous (FPP) and asynchronous (PPA) modes using a microprocessor with **.rbf**, **.hex**, and **.ttf** file formats. This does not apply to FPP mode for enhanced configuration devices using **.pof** file format. This also does not apply to serial modes.

If, after sending out all of its data, the enhanced configuration device does not detect CONF_DONE going high, it recognizes that the Stratix or Stratix GX device has not configured successfully. The enhanced configuration device pulses its OE pin low for a few microseconds, driving the nSTATUS pin on the Stratix or Stratix GX device low. If the **Auto-restart configuration after error** option is on, the Stratix or Stratix GX device resets and then pulses its nSTATUS low. When nSTATUS returns high, reconfiguration is restarted (see Figure 1–11 on page 1–25).

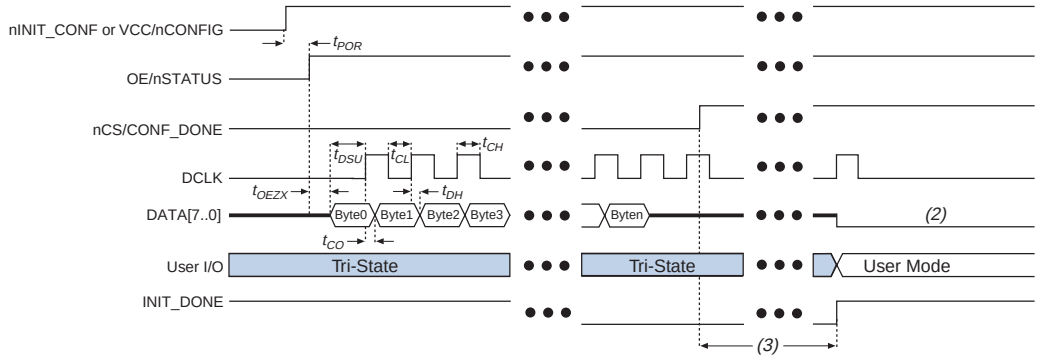
Do not drive CONF_DONE low after device configuration to delay initialization. Instead, use the **Enable User-Supplied Start-Up Clock (CLKUSR)** option in the **Device & Pin Options** dialog box. You can use this option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain initialize together.

After the first Stratix or Stratix GX device completes configuration during multi-device configuration, its nCEO pin activates the second Stratix or Stratix GX device's nCE pin, prompting the second device to begin configuration. Because CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time. Because nSTATUS pins are tied together, configuration stops for the whole chain if any device (including enhanced configuration devices) detects an error. Also, if the enhanced configuration device does not detect a high on CONF_DONE at the end of configuration, it pulses its OE low for a few microseconds to reset the chain. The low OE pulse drives nSTATUS low on all Stratix and Stratix GX devices, causing them to enter an error state. This state is similar to a Stratix or Stratix GX device detecting an error.

If the **Auto-restart configuration after error** option is on, the Stratix and Stratix GX devices release their nSTATUS pins after a reset time-out period. When the nSTATUS pins are released and pulled high, the configuration device reconfigures the chain. If the **Auto-restart configuration after error** option is off, nSTATUS stays low until the Stratix and Stratix GX devices are reset with a low pulse on nCONFIG.

Figure 1–11 shows the FPP configuration with a configuration device timing waveform for Stratix and Stratix GX devices.

Figure 1–11. FPP Configuration with a Configuration Device Timing Waveform Note (1)



Notes to Figure 1–11:

(1) For timing information, see the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet*.
(2) The configuration device drives DATA high after configuration.
(3) Stratix and Stratix GX devices enter user mode 136 clock cycles after CONF_DONE goes high.

FPP Configuration Using a Microprocessor

When using a microprocessor for parallel configuration, the microprocessor transfers data from a storage device to the Stratix or Stratix GX device through configuration hardware. To initiate configuration, the microprocessor needs to generate a low-to-high transition on the nCONFIG pin and the Stratix or Stratix GX device must release nSTATUS. The microprocessor then places the configuration data to the DATA[7..0] pins of the Stratix or Stratix GX device. Data is clocked continuously into the Stratix or Stratix GX device until CONF_DONE goes high.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists. You can pause configuration by halting DCLK for an indefinite amount of time.

After all configuration data is sent to the Stratix or Stratix GX device, the CONF_DONE pin goes high to show successful configuration and the start of initialization. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. If you are using the **clkusr** option, after all data is transferred clkusr must be clocked an additional 136 times for the Stratix or Stratix GX device to initialize properly. Driving DCLK to the device after configuration is complete does not affect device operation. By

default, the INIT_DONE output is disabled. You can enable the INIT_DONE output by turning on the **Enable INIT_DONE output** option in the Quartus II software.

If you do not turn on the **Enable INIT_DONE output** option in the Quartus II software, you are advised to wait for maximum value of t_{CD2UM} (see Table 1–9) after the CONF_DONE signal goes high to ensure the device has been initialized properly and that it has entered user mode.

During configuration and initialization and before the device enters user mode, the microprocessor must not drive the CONF_DONE signal low.



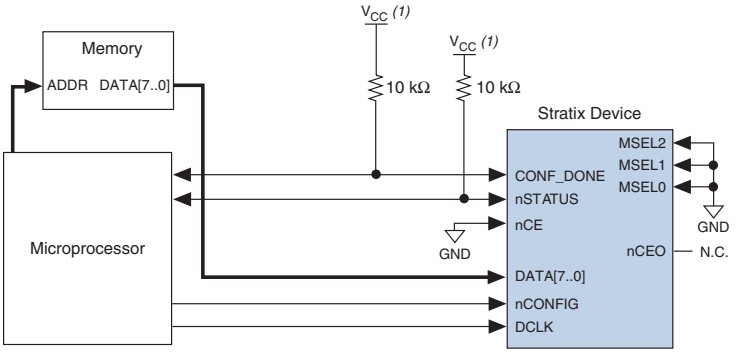
If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

If the Stratix or Stratix GX device detects an error during configuration, it drives nSTATUS low to alert the microprocessor. The pin on the microprocessor connected to nSTATUS must be an input. The microprocessor can then pulse nCONFIG low to restart the configuration error. With the **Auto-restart configuration after error** option on, the Stratix or Stratix GX device releases nSTATUS after a reset time-out period. After nSTATUS is released, the microprocessor can reconfigure the Stratix or Stratix GX device without pulsing nCONFIG low.

The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. If the microprocessor sends all the data and the initialization clock starts but CONF_DONE and INIT_DONE have not gone high, it must reconfigure the Stratix or Stratix GX device. After waiting the specified 136 DCLK cycles, the microprocessor should restart configuration by pulsing nCONFIG low.

Figure 1–12 shows the circuit for Stratix and Stratix GX parallel configuration using a microprocessor.

Figure 1–12. Parallel Configuration Using a Microprocessor

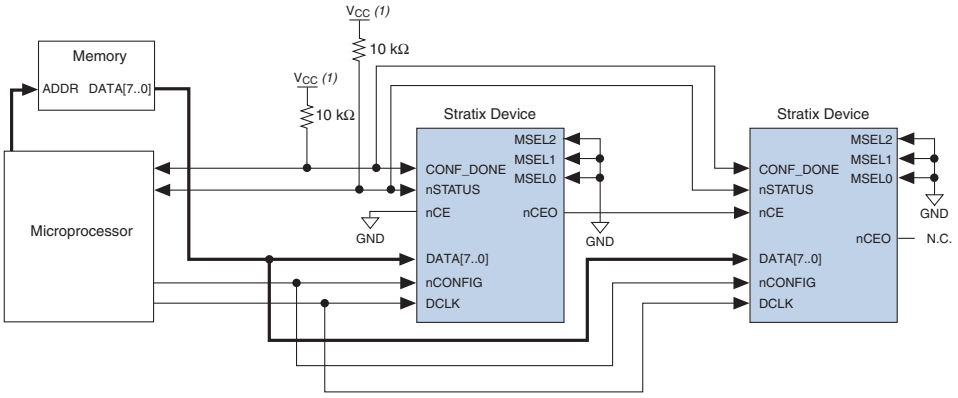


Note to Figure 1–12:
(1) The pull-up resistors should be connected to any V_{CC} that meets the Stratix high-level input voltage (V_{IH}) specification.

For multi-device parallel configuration with a microprocessor, the nCEO pin of the first Stratix or Stratix GX device is cascaded to the second device's nCE pin. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the microprocessor. Because the CONF_DONE pins of the devices are connected together, all devices initialize and enter user mode at the same time.

Because the nSTATUS pins are also tied together, if any of the devices detects an error, the entire chain halts configuration and drives nSTATUS low. The microprocessor can then pulse nCONFIG low to restart configuration. If the **Auto-restart configuration after error** option is on, the Stratix and Stratix GX devices release nSTATUS after a reset time-out period. The microprocessor can then reconfigure the devices once nSTATUS is released. Figure 1–13 shows multi-device configuration using a microprocessor. Figure 1–14 shows multi-device configuration when both Stratix and Stratix GX devices are receiving the same data. In this case, the microprocessor sends the data to both devices simultaneously, and the devices configure simultaneously.

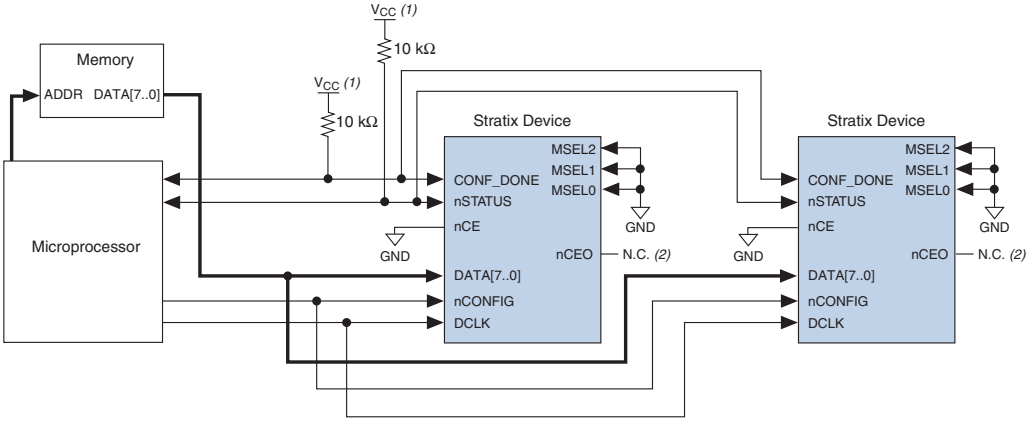
Figure 1–13. Parallel Data Transfer in Serial Configuration with a Microprocessor



Note to Figure 1–13:

(1) You should connect the pull-up resistors to any V_{CC} that meets the Stratix high-level input voltage (V_{IH}) specification.

Figure 1–14. Multiple Device Parallel Configuration with the Same Data Using a Microprocessor



Notes to Figure 1–14:

(1) You should connect the pull-up resistors to any V_{CC} that meets the Stratix high-level input voltage (V_{IH}) specification.

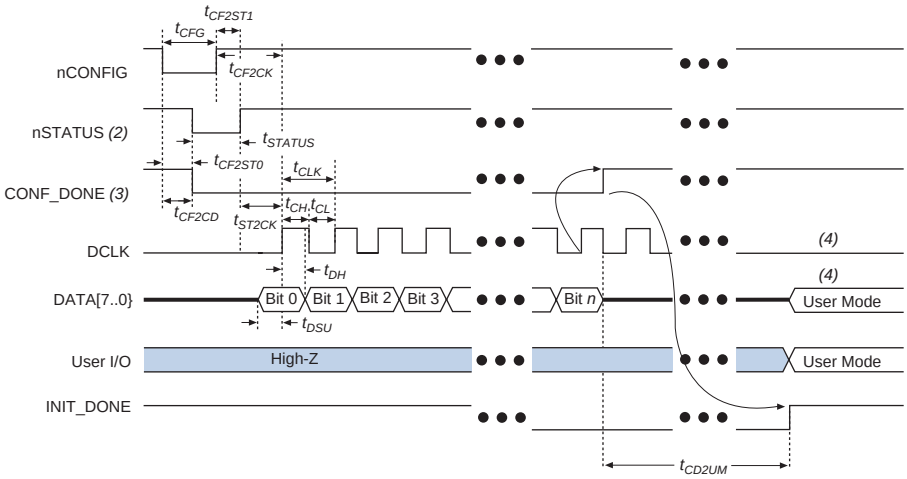
(2) The nCEO pins are left unconnected when configuring the same data into multiple Stratix or Stratix GX devices.

For more information on configuring multiple Altera devices in the same configuration chain, see the *Configuring Mixed Altera FPGA Chains* chapter in the *Configuration Handbook, Volume 2*.

FPP Configuration Timing

Figure 1–15 shows FPP timing waveforms for configuring a Stratix or Stratix GX device in FPP mode. Table 1–9 shows the FPP timing parameters for Stratix or Stratix GX devices.

Figure 1–15. Timing Waveform for Configuring Devices in FPP Mode *Note (1)*



Notes to Figure 1–15:

(1) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(2) Upon power-up, the Stratix II device holds nSTATUS low for the time of the POR delay.
(3) Upon power-up, before and during configuration, CONF_DONE is low.
(4) DCLK should not be left floating after configuration. It should be driven high or low, whichever is convenient. DATA[] is available as user I/Os after configuration and the state of these pins depends on the dual-purpose pin settings.

Table 1–9. FPP Timing Parameters for Stratix & Stratix GX Devices (Part 1 of 2)

Symbol	Parameter	Min	Max	Units
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		µs
t_{DSU}	Data setup time before rising edge on DCLK	7		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CFG}	nCONFIG low pulse width	40		µs
t_{CH}	DCLK high time	4		ns
t_{CL}	DCLK low time	4		ns
t_{CLK}	DCLK period	10		ns

Table 1–9. FPP Timing Parameters for Stratix & Stratix GX Devices (Part 2 of 2)

Symbol	Parameter	Min	Max	Units
f_{MAX}	DCLK frequency		100	MHz
t_{CD2UM}	CONF_DONE high to user mode *(1)*	6	20	µs
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CF2ST1}	nCONFIG high to nSTATUS high		40 *(2)*	µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(2)*	µs
t_{ST2CK}	nSTATUS high to firstrising edge of DCLK	1		µs

Notes to Table 1–9:

(1) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 136 to obtain this value.

(2) This value is obtainable if users do not delay configuration by extending the nSTATUS low pulse width.

PPA Configuration

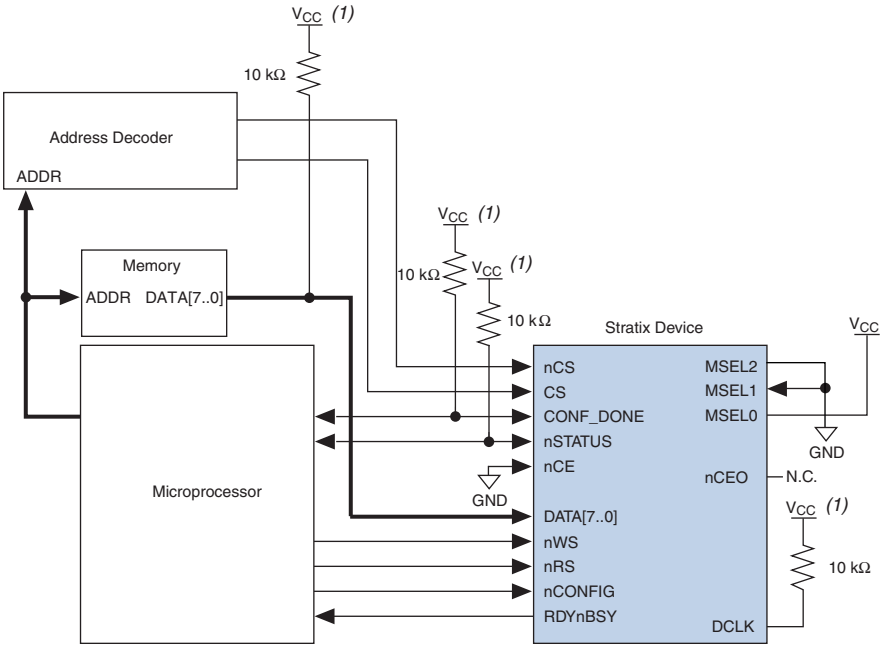
In PPA schemes, a microprocessor drives data to the Stratix or Stratix GX device through a download cable. When using a PPA scheme, use a 1-kΩ pull-up resistor to pull the DCLK pin high to prevent unused configuration pins from floating.

To begin configuration, the microprocessor drives nCONFIG high and then asserts the target device's nCS pin low and CS pin high. Next, the microprocessor places an 8-bit configuration word on the target device's data inputs and pulses nWS low. On the rising edge of nWS, the target device latches a byte of configuration data and then drives its RDYnBSY signal low, indicating that it is processing the byte of configuration data. The microprocessor then performs other system functions while the Stratix or Stratix GX device is processing the byte of configuration data.

Next, the microprocessor checks nSTATUS and CONF_DONE. If nSTATUS is high and CONF_DONE is low, the microprocessor sends the next data byte. If nSTATUS is low, the device is signaling an error and the microprocessor should restart configuration. However, if nSTATUS is high and all the configuration data is received, the device is ready for initialization. At the beginning of initialization, CONF_DONE goes high to indicate that configuration is complete. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. When initialization is complete, the Stratix or Stratix GX device enters user mode.

Figure 1–16 shows the PPA configuration circuit. An optional address decoder controls the device's nCS and CS pins. This decoder allows the microprocessor to select the Stratix or Stratix GX device by accessing a particular address, simplifying the configuration process.

Figure 1–16. PPA Configuration Circuit



Note to Figure 1–16:
(1) The pull-up resistor should be connected to the same supply voltage as the Stratix or Stratix GX device.

The device's nCS or CS pins can be toggled during PPA configuration if the design meets the specifications for t_{CSSU}, t_{WSP}, and t_{CSH} given in Table 1–10 on page 1–36. The microprocessor can also directly control the nCS and CS signals. You can tie one of the nCS or CS signals to its active state (i.e., nCS may be tied low) and toggle the other signal to control configuration.

Stratix and Stratix GX devices can serialize data internally without the microprocessor. When the Stratix or Stratix GX device is ready for the next byte of configuration data, it drives RDYnBSY high. If the microprocessor senses a high signal when it polls RDYnBSY, the microprocessor strobes the next byte of configuration data into the device. Alternatively, the nRS signal can be strobed, causing the RDYnBSY signal to appear on DATA7. Because RDYnBSY does not need to

be monitored, reading the state of the configuration data by strobing nRS low saves a system I/O port. Do not drive data onto the data bus while nRS is low because it causes contention on DATA7. If the nRS pin is not used to monitor configuration, you should tie it high. To simplify configuration, the microprocessor can wait for the total time of t_{BUSY} (max) + t_{RDY2WS} + t_{W2SB} before sending the next data bit.

After configuration, the nCS, CS, nRS, nWS, and RDYnBSY pins act as user I/O pins. However, if the PPA scheme is chosen in the Quartus II software, these I/O pins are tri-stated by default in user mode and should be driven by the microprocessor. To change the default settings in the Quartus II software, select **Device & Pin Option** (Compiler Setting menu).

If the Stratix or Stratix GX device detects an error during configuration, it drives nSTATUS low to alert the microprocessor. The microprocessor can then pulse nCONFIG low to restart the configuration process. Alternatively, if the **Auto-Restart Configuration on Frame Error** option is turned on, the Stratix or Stratix GX device releases nSTATUS after a reset time-out period. After nSTATUS is released, the microprocessor can reconfigure the Stratix or Stratix GX device. At this point, the microprocessor does not need to pulse nCONFIG low.

The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The microprocessor must monitor the nSTATUS pin to detect errors and the CONF_DONE pin to determine when programming completes (CONF_DONE goes high one byte early in parallel mode). If the microprocessor sends all configuration data and starts initialization but CONF_DONE is not asserted, the microprocessor must reconfigure the Stratix or Stratix GX device.

By default, the INIT_DONE is disabled. You can enable the INIT_DONE output by turning on the **Enable INIT_DONE output** option in the Quartus II software. If you do not turn on the **Enable INIT_DONE output** option in the Quartus II software, you are advised to wait for the maximum value of t_{CD2UM} (see Table 1–10) after the CONF_DONE signal goes high to ensure the device has been initialized properly and that it has entered user mode.

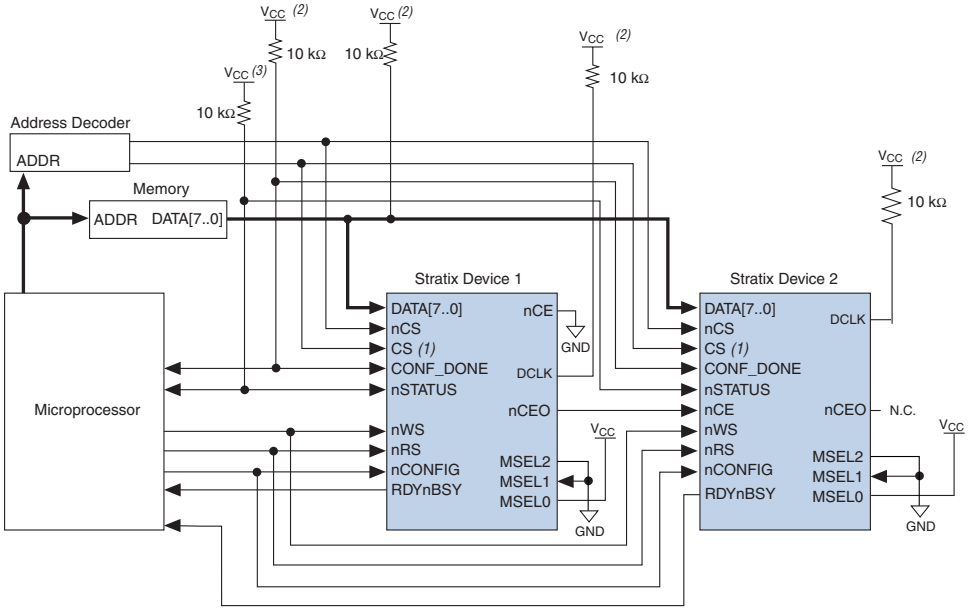
During configuration and initialization, and before the device enters user mode, the microprocessor must not drive the CONF_DONE signal low.



If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

You can also use PPA mode to configure multiple Stratix and Stratix GX devices. Multi-device PPA configuration is similar to single-device PPA configuration, except that the Stratix and Stratix GX devices are cascaded. After you configure the first Stratix or Stratix GX device, nCEO is asserted, which asserts the nCE pin on the second device, initiating configuration. Because the second Stratix or Stratix GX device begins configuration within one write cycle of the first device, the transfer of data destinations is transparent to the microprocessor. All Stratix and Stratix GX device CONF_DONE pins are tied together; therefore, all devices initialize and enter user mode at the same time. See Figure 1–17.

Figure 1–17. PPA Multi-Device Configuration Circuit



Notes to Figure 1–17:

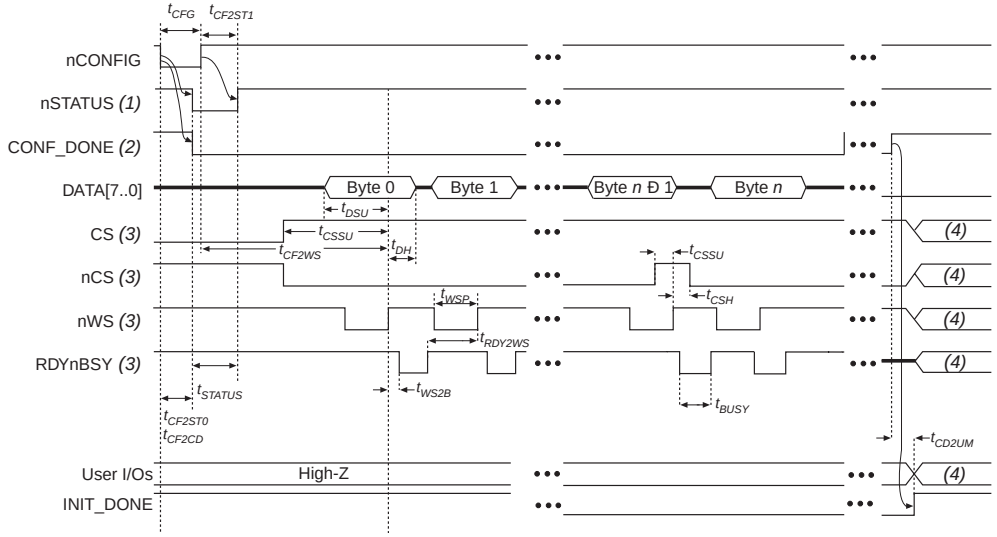
(1) If not used, you can connect the CS pin to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) Connect the pull-up resistor to the same supply voltage as the Stratix or Stratix GX device.

PPA Configuration Timing

Figure 1–18 shows the Stratix and Stratix GX device timing waveforms for PPA configuration.

Figure 1–18. PPA Timing Waveforms for Stratix & Stratix GX Devices

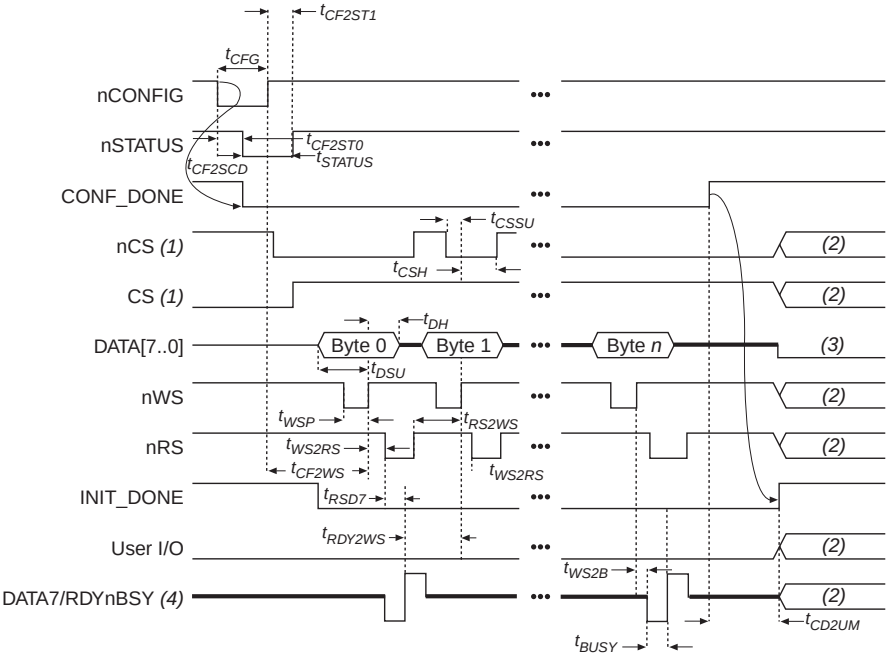


Notes to Figure 1–18:

(1) Upon power-up, nSTATUS is held low for the time of the POR delay.
(2) Upon power-up, before and during configuration, CONF_DONE is low.
(3) After configuration, the state of CS, nCS, nWS, and RDYnBSY depends on the design programmed into the Stratix or Stratix GX device.
(4) Device I/O pins are in user mode.

Figure 1–19 shows the Stratix and Stratix GX timing waveforms when using strobed nRS and nWS signals.

Figure 1–19. PPA Timing Waveforms Using Strobed nRS & nWS Signals



Notes to Figure 1–19:

(1) The user can toggle nCS or CS during configuration if the design meets the specification for t_{CSSU}, t_{WSP}, and t_{CSH}.

(2) Device I/O pins are in user mode.

(3) The DATA[7..0] pins are available as user I/Os after configuration and the state of theses pins depends on the dual-purpose pin settings. Do not leave DATA[7..0] floating. If these pins are not used in user-mode, you should drive them high or low, whichever is more convenient.

(4) DATA7 is a bidirectional pin. It represents an input for data input, but represents an output to show the status of RDYnBSY.

Table 1–10 defines the Stratix and Stratix GX timing parameters for PPA configuration

Table 1–10. PPA Timing Parameters for Stratix & Stratix GX Devices

Symbol	Parameter	Min	Max	Units
t_{CF2WS}	nCONFIG high to first rising edge on nWS	40		µs
t_{DSU}	Data setup time before rising edge on nWS	10		ns
t_{DH}	Data hold time after rising edge on nWS	0		ns
t_{CSSU}	Chip select setup time before rising edge on nWS	10		ns
t_{CSH}	Chip select hold time after rising edge on nWS	0		ns
t_{WSP}	nWS low pulse width	15		ns
t_{CFG}	nCONFIG low pulse width	40		µs
t_{WS2B}	nWS rising edge to RDYnBSY low		20	ns
t_{BUSY}	RDYnBSY low pulse width	7	45	ns
t_{RDY2WS}	RDYnBSY rising edge to nWS rising edge	15		ns
t_{WS2RS}	nWS rising edge to nRS falling edge	15		ns
t_{RS2WS}	nRS rising edge to nWS rising edge	15		ns
t_{RSD7}	nRS falling edge to DATA7 valid with RDYnBSY signal		20	ns
t_{CD2UM}	CONF_DONE high to user mode *(1)*	6	20	µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(2)*	µs
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CF2ST1}	nCONFIG high to nSTATUS high		40 *(2)*	µs

Notes to Table 1–10:

(1) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 136 to obtain this value.

(2) This value is obtained if you do not delay configuration by extending the nstatus to low pulse width.

For information on how to create configuration and programming files for this configuration scheme, see the *Software Settings* section in the *Configuration Handbook, Volume 2*.

JTAG Programming & Configuration

The JTAG has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture offers the capability to efficiently test components on printed circuit boards (PCBs) with tight lead spacing. The BST architecture can test pin connections without using physical test

probes and capture functional data while a device is operating normally. You can also use the JTAG circuitry to shift configuration data into the device.

For more information on JTAG boundary-scan testing, see *AN 39: IEEE 1149.1 (JTAG) Boundary-Scan Testing in Altera Devices*.

To use the SignalTap® II embedded logic analyzer, you need to connect the JTAG pins of your Stratix device to a download cable header on your PCB.

For more information on SignalTap II, see the *Design Debugging Using SignalTap II Embedded Logic Analyzer* chapter in the *Quartus II Handbook, Volume 2*.

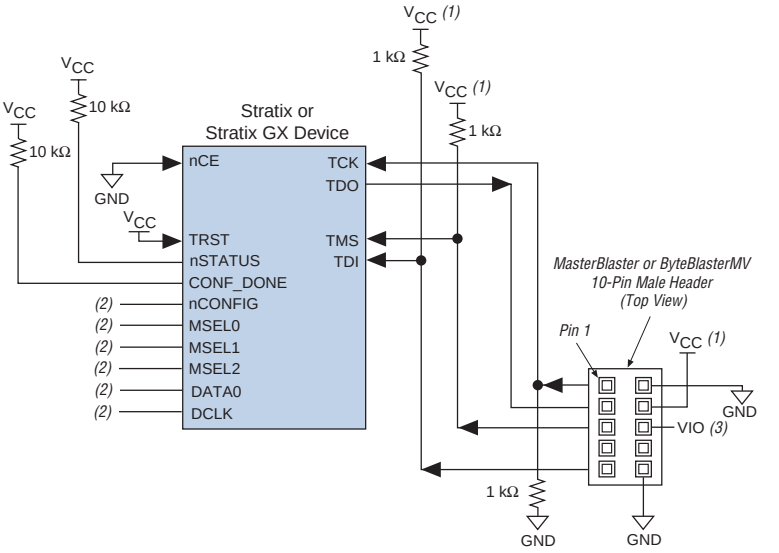
A device operating in JTAG mode uses four required pins, TDI, TDO, TMS, and TCK, and one optional pin, TRST. The four JTAG input pins (TDI, TMS, TCK and TRST) have weak, internal pull-up resistors, whose values range from 20 to 40 kΩ All other pins are tri-stated during JTAG configuration. Do not begin JTAG configuration until all other configuration is complete. Table 1–11 shows each JTAG pin's function.

Table 1–11. JTAG Pin Descriptions

Pin	Description	Function
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. The VCCSEL pin controls the input buffer selection.
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. The high level output voltage is determined by VCCIO.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the Test Access Port (TAP) controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. The VCCSEL pin controls the input buffer selection.
TCK	Test clock input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. The VCCSEL pin controls the input buffer selection.
TRST	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. The VCCSEL pin controls the input buffer selection.

During JTAG configuration, data is downloaded to the device on the PCB through the MasterBlaster or ByteBlasterMV header. Configuring devices through a cable is similar to programming devices in-system. One difference is to connect the TRST pin to V_{CC} to ensure that the TAP controller is not reset. See Figure 1–20.

Figure 1–20. JTAG Configuration of a Single Device



Notes to Figure 1–20:

(1) You should connect the pull-up resistor to the same supply voltage as the download cable.

(2) You should connect the nCONFIG, MSEL0, and MSEL1 pins to support a non-JTAG configuration scheme. If you only use JTAG configuration, connect nCONFIG to V_{CC}, and MSEL0, MSEL1, and MSEL2 to ground. Pull DATA0 and DCLK to high or low.

(3) V_{IO} is a reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. See the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value.

To configure a single device in a JTAG chain, the programming software places all other devices in BYPASS mode. In BYPASS mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later.

Stratix and Stratix GX devices have dedicated JTAG pins. You can perform JTAG testing on Stratix and Stratix GX devices before and after, but not during configuration. The chip-wide reset and output enable pins on Stratix and Stratix GX devices do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration of Stratix and Stratix GX devices, you should consider the regular configuration pins. Table 1–12 shows how you should connect these pins during JTAG configuration.

Table 1–12. Dedicated Configuration Pin Connections During JTAG Configuration

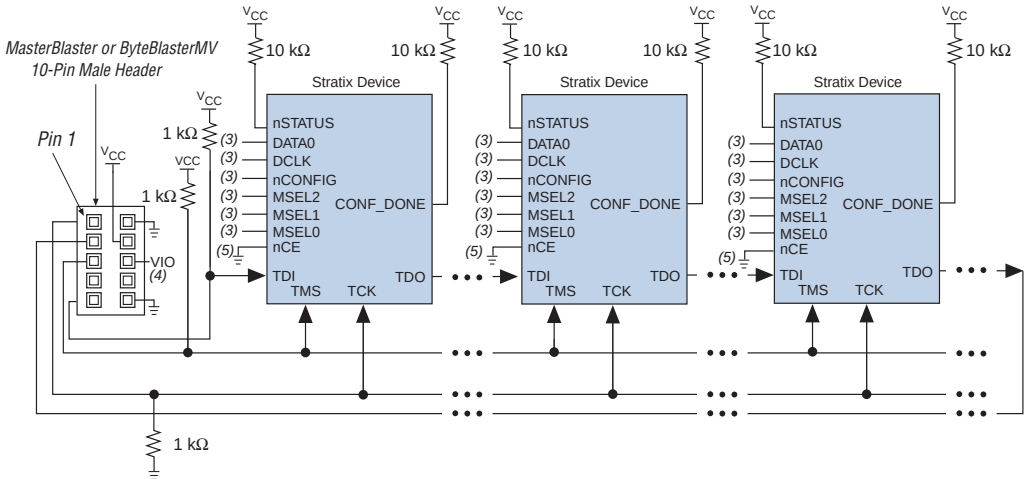
Signal	Description
nCE	On all Stratix and Stratix GX devices in the chain, nCE should be driven low by connecting it to ground, pulling it low via a resistor, or driving it by some control circuitry. For devices that are also in multi-device PS, FPP or PPA configuration chains, the nCE pins should be connected to GND during JTAG configuration or JTAG configured in the same order as the configuration chain.
nCEO	On all Stratix and Stratix GX devices in the chain, nCEO can be left floating or connected to the nCE of the next device. See nCE pin description above.
MSEL	These pins must not be left floating. These pins support whichever non-JTAG configuration is used in production. If only JTAG configuration is used, you should tie both pins to ground.
nCONFIG	nCONFIG must be driven high through the JTAG programming process. Driven high by connecting to V_{CC}, pulling high via a resistor, or driven by some control circuitry.
nSTATUS	Pull to V_{CC} via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin should be pulled up to V_{CC} individually. nSTATUS pulling low in the middle of JTAG configuration indicates that an error has occurred.
CONF_DONE	Pull to V_{CC} via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin should be pulled up to V_{CC} individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Should not be left floating. Drive low or high, whichever is more convenient on your board.
DATA0	Should not be left floating. Drive low or high, whichever is more convenient on your board.

JTAG Programming & Configuration of Multiple Devices

When programming a JTAG device chain, one JTAG-compatible header, such as the ByteBlasterMV header, is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capacity of the download cable. However, when more than five devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

JTAG-chain device programming is ideal when the PCB contains multiple devices, or when testing the PCB using JTAG BST circuitry. Figure 1–21 shows multi-device JTAG configuration.

Figure 1–21. Multi-Device JTAG Configuration *Notes (1), (2)*



Notes to Figure 1–21:

(1) Stratix, Stratix GX, APEX™ II, APEX 20K, Mercury™, ACEX® 1K, and FLEX® 10K devices can be placed within the same JTAG chain for device programming and configuration.

(2) For more information on all configuration pins connected in this mode, see Table 1–11 on page 1–37.

(3) Connect the nCONFIG, MSEL0, MSEL1, and MSEL2 pins to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to V_{CC}, and MSEL0, MSEL1, and MSEL2 to ground. Pull DATA0 and DCLK to either high or low.

(4) V_{IO} is a reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. See the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value.

(5) nCE must be connected to GND or driven low for successful JTAG configuration.

The nCE pin must be connected to GND or driven low during JTAG configuration. In multi-device PS, FPP and PPA configuration chains, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, you should make sure the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multi-device configuration chain, the nCEO of the previous device drives nCE of the next device low when it has successfully been JTAG configured.

The Quartus II software verifies successful JTAG configuration upon completion. The software checks the state of CONF_DONE through the JTAG port. If CONF_DONE is not in the correct state, the Quartus II software indicates that configuration has failed. If CONF_DONE is in the correct state, the software indicates that configuration was successful.

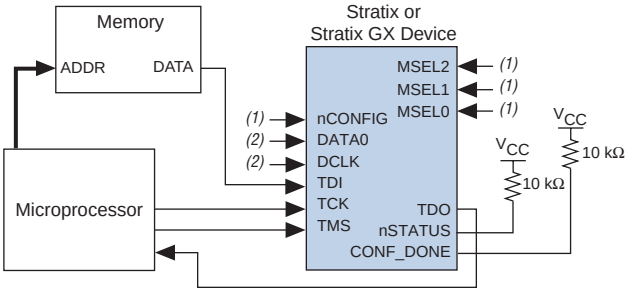


If VCCIO is tied to 3.3 V, both the I/O pins and JTAG TDO port drive at 3.3-V levels.

Do not attempt JTAG and non-JTAG configuration simultaneously. When configuring through JTAG, allow any non-JTAG configuration to complete first.

Figure 1–22 shows the JTAG configuration of a Stratix or Stratix GX device with a microprocessor.

Figure 1–22. JTAG Configuration of Stratix & Stratix GX Devices with a Microprocessor



Notes to Figure 1–22:

(1) Connect the nCONFIG, MSEL2, MSEL1, and MSEL0 pins to support a non-JTAG configuration scheme. If your design only uses JTAG configuration, connect the nCONFIG pin to V_{CC} and the MSEL2, MSEL1, and MSEL0 pins to ground.

(2) Pull DATA0 and DCLK to either high or low.

Configuration with JRunner Software Driver

JRunner is a software driver that allows you to configure Altera FPGAs through the ByteBlasterMV download cable in JTAG mode. The programming input file supported is in Raw Binary File (**.rbf**) format. JRunner also requires a Chain Description File (**.cdf**) generated by the Quartus II software. JRunner is targeted for embedded JTAG configuration. The source code has been developed for the Windows NT operating system. You can customize the code to make it run on other platforms.

For more information on the JRunner software driver, see the *JRunner Software Driver: An Embedded Solution to the JTAG Configuration White Paper* and zip file.

Jam STAPL Programming & Test Language

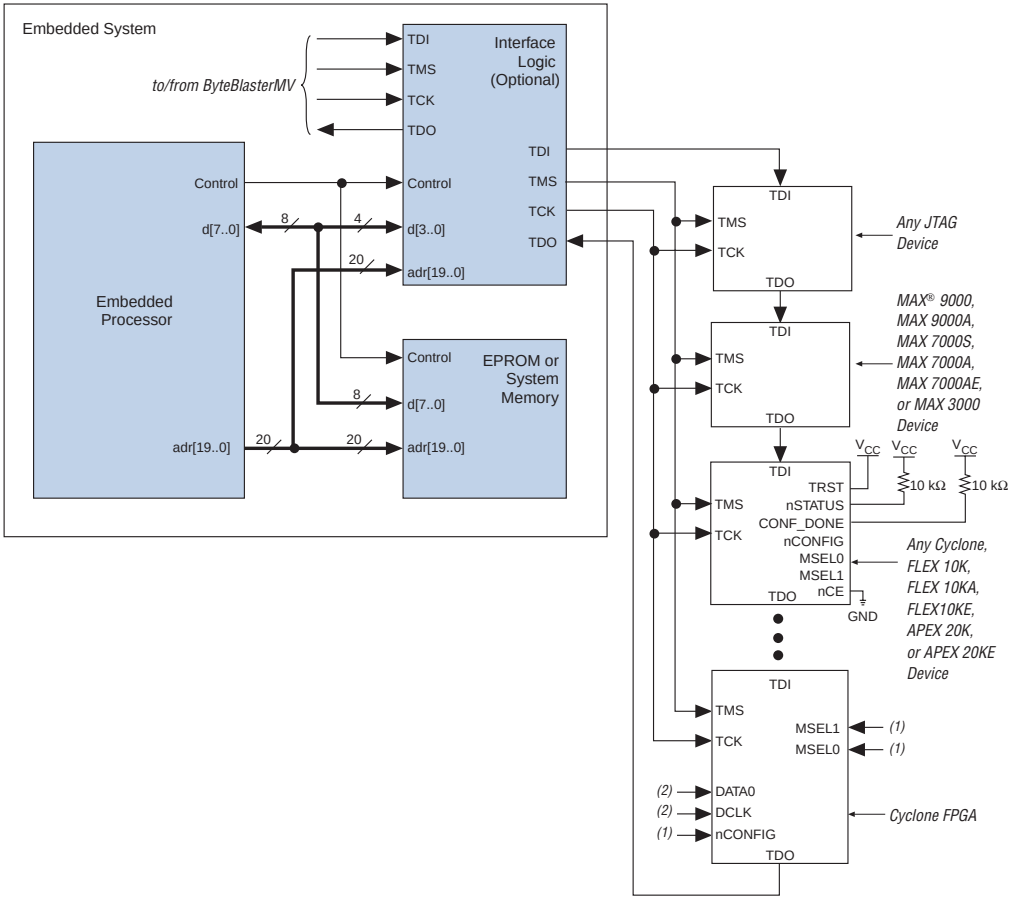
The Jam™ Standard Test and Programming Language (STAPL), JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

Connecting the JTAG Chain to the Embedded Processor

There are two ways to connect the JTAG chain to the embedded processor. The most straightforward method is to connect the embedded processor directly to the JTAG chain. In this method, four of the processor pins are dedicated to the JTAG interface, saving board space but reducing the number of available embedded processor pins.

Figure 1–23 illustrates the second method, which is to connect the JTAG chain to an existing bus through an interface PLD. In this method, the JTAG chain becomes an address on the existing bus. The processor then reads from or writes to the address representing the JTAG chain.

Figure 1–23. Embedded System Block Diagram



Notes to Figure 1–23:

(1) Connect the nCONFIG, MSEL2, MSEL1, and MSEL0 pins to support a non-JTAG configuration scheme. If your design only uses JTAG configuration, connect the nCONFIG pin to V_{CC} and the MSEL2, MSEL1, and MSEL0 pins to ground.

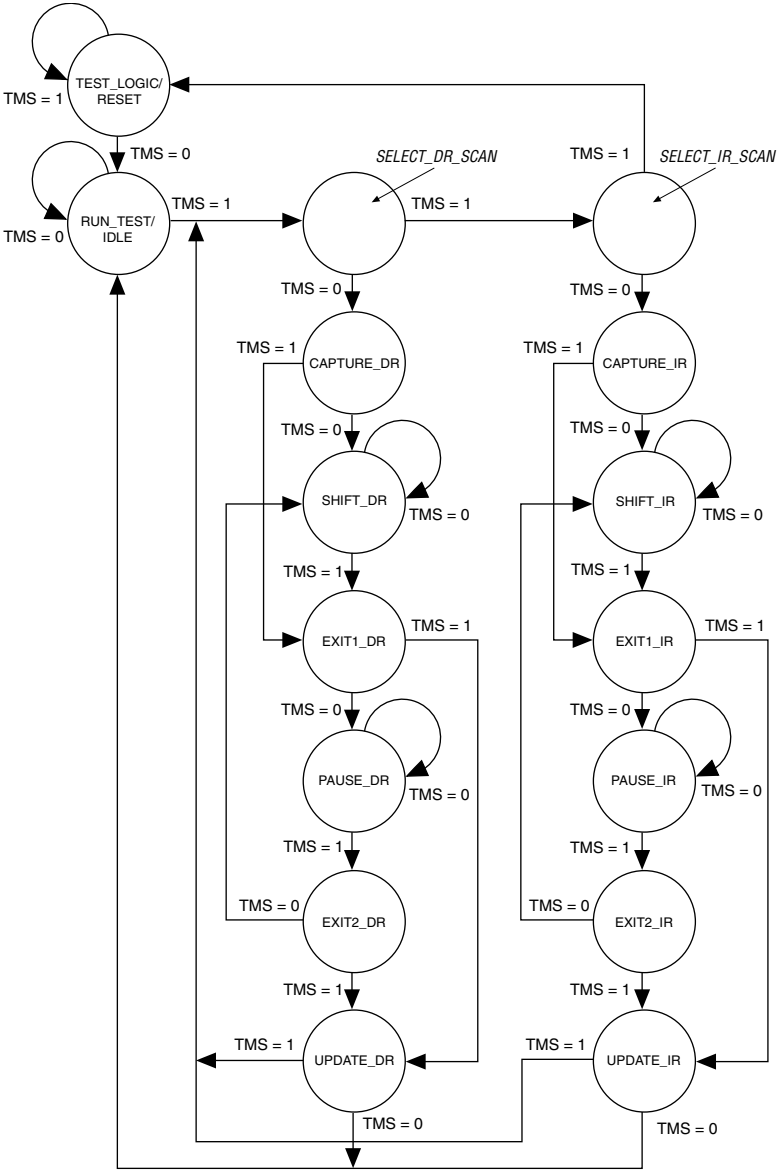
(2) Pull DATA0 and DCLK to either high or low.

Both JTAG connection methods should include space for the MasterBlaster or ByteBlasterMV header connection. The header is useful during prototyping because it allows you to verify or modify the Stratix or Stratix GX device's contents. During production, you can remove the header to save cost.

Program Flow

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine. The TAP controller is a 16-state state machine that is clocked on the rising edge of TCK, and uses the TMS pin to control JTAG operation in a device. Figure 1–24 shows the flow of an IEEE Std. 1149.1 TAP controller state machine.

Figure 1–24. JTAG TAP Controller State Machine



While the Jam Player provides a driver that manipulates the TAP controller, the Jam Byte-Code File (**.jbc**) provides the high-level intelligence needed to program a given device. All Jam instructions that

send JTAG data to the device involve moving the TAP controller through either the data register leg or the instruction register leg of the state machine. For example, loading a JTAG instruction involves moving the TAP controller to the SHIFT_IR state and shifting the instruction into the instruction register through the TDI pin. Next, the TAP controller is moved to the RUN_TEST/IDLE state where a delay is implemented to allow the instruction time to be latched. This process is identical for data register scans, except that the data register leg of the state machine is traversed.

The high-level Jam instructions are the DRSCAN instruction for scanning the JTAG data register, the IRSCAN instruction for scanning the instruction register, and the WAIT command that causes the state machine to sit idle for a specified period of time. Each leg of the TAP controller is scanned repeatedly, according to instructions in the JBC file, until all of the target devices are programmed.

Figure 1–25 illustrates the functional behavior of the Jam Player when it parses the JBC file. When the Jam Player encounters a DRSCAN, IRSCAN, or WAIT instruction, it generates the proper data on TCK, TMS, and TDI to complete the instruction. The flow diagram shows branches for the DRSCAN, IRSCAN, and WAIT instructions. Although the Jam Player supports other instructions, they are omitted from the flow diagram for simplicity.

Figure 1–25. Jam Player Flow Diagram (Part 1 of 2)

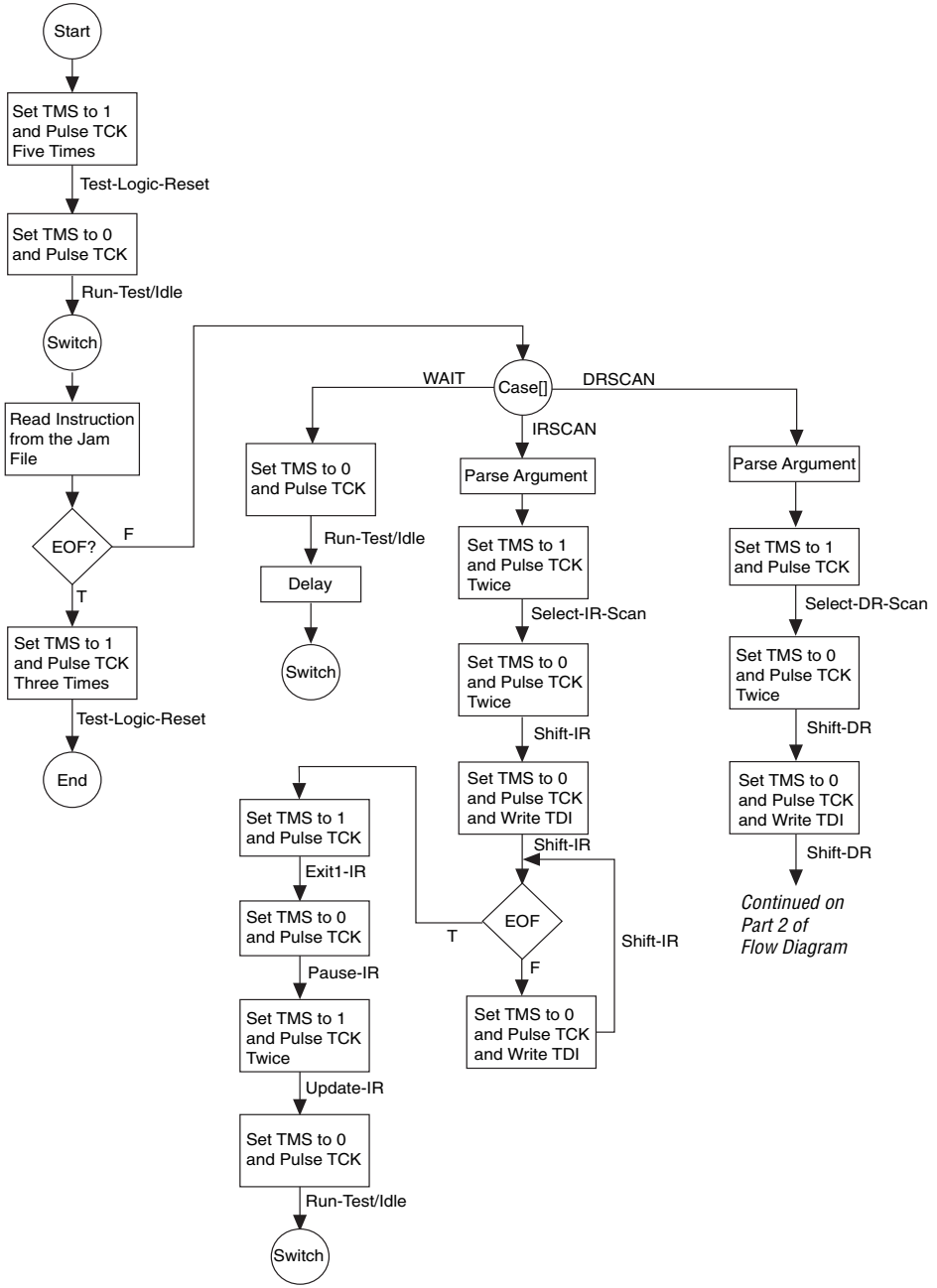
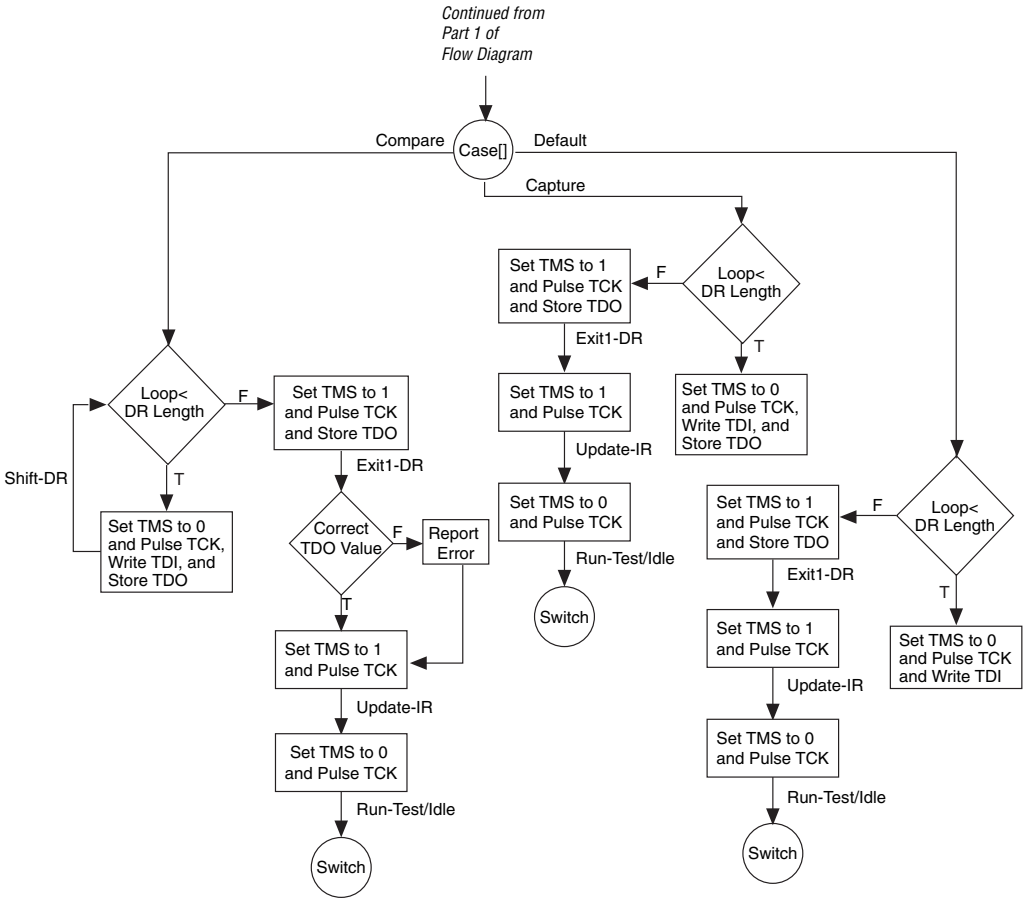


Figure 1–26. Jam Player Flow Diagram (Part 2 of 2)



Execution of a Jam program starts at the beginning of the program. The program flow is controlled using GOTO, CALL/RETURN, and FOR/NEXT structures. The GOTO and CALL statements see labels that are symbolic names for program statements located elsewhere in the Jam program. The language itself enforces almost no constraints on the organizational structure or control flow of a program.

The Jam language does not support linking multiple Jam programs together or including the contents of another file into a Jam program.

Jam Instructions

Each Jam statement begins with one of the instruction names listed in Table 1–13. The instruction names, including the names of the optional instructions, are reserved keywords that you cannot use as variable or label identifiers in a Jam program.

Table 1–13. Instruction Names

BOOLEAN	INTEGER	PREIR
CALL	IRSCAN	PRINT
CRC	IRSTOP	PUSH
DRSCAN	LET	RETURN
DRSTOP	NEXT	STATE
EXIT	NOTE	WAIT
EXPORT	POP	VECTOR *(1)*
FOR	POSTDR	VMAP *(1)*
GOTO	POSTIR	–
IF	PREDR	–

Note to Table 1–13:
(1) This instruction name is an optional language extension.

Table 1–14 shows the state names that are reserved keywords in the Jam language. These keywords correspond to the state names specified in the IEEE Std. 1149.1 JTAG specification.

Table 1–14. Reserved Keywords (Part 1 of 2)

IEEE Std. 1149.1 JTAG State Names	Jam Reserved State Names
Test-Logic-Reset	RESET
Run-Test-Idle	IDLE
Select-DR-Scan	DRSELECT
Capture-DR	DRCAPTURE
Shift-DR	DRSHIFT
Exit1-DR	DREXIT1
Pause-DR	DRPAUSE
Exit2-DR	DREXIT2
Update-DR	DRUPDATE
Select-IR-Scan	IRSELECT

Table 1–14. Reserved Keywords (Part 2 of 2)	
IEEE Std. 1149.1 JTAG State Names	**Jam Reserved State Names**
Capture-IR	IRCAPTURE
Shift-IR	IRSHIFT
Exit1-IR	IREXIT1
Pause-IR	IRPAUSE
Exit2-IR	IREXIT2
Update-IR	IRUPDATE

Example Jam File that Reads the IDCODE

Figure 1–27 illustrates the flexibility and utility of the Jam STAPL. The example reads the IDCODE out of a single device in a JTAG chain.



The array variable, I_IDCODE, is initialized with the IDCODE instruction bits ordered the LSB first (on the left) to most significant bit (MSB) (on the right). This order is important because the array field in the IRSCAN instruction is always interpreted, and sent, MSB to LSB.

Figure 1–27. Example Jam File Reading IDCODE

```
BOOLEAN read_data[32];
BOOLEAN I_IDCODE[10] = BIN 1001101000; 'assumed
BOOLEAN ONES_DATA[32] = HEX FFFFFFFF;
INTEGER i;
'Set up stop state for IRSCAN
IRSTOP IRPAUSE;
'Initialize device
STATE RESET;
IRSCAN 10, I_IDCODE[0..9]; 'LOAD IDCODE INSTRUCTION
STATE IDLE;
WAIT 5 USEC, 3 CYCLES;
DRSCAN 32, ONES_DATA[0..31], CAPTURE
read_data[0..31];
'CAPTURE IDCODE
PRINT "IDCODE:";
FOR i=0 to 31;
PRINT read_data[i];
NEXT i;
EXIT 0;
```

Configuring Using the MicroBlaster Driver

The MicroBlaster™ software driver allows you to configure Altera devices in an embedded environment using PS or FPP mode. The MicroBlaster software driver supports a Raw Binary File (**.rbf**) programming input file. The source code is developed for the Windows NT operating system, although you can customize it to run on other operating systems. For more information on the MicroBlaster software driver, go to the Altera web site (**www.altera.com**).

Device Configuration Pins

The following tables describe the connections and functionality of all the configuration related pins on the Stratix or Stratix GX device. Table 1–15 describes the dedicated configuration pins, which are required to be connected properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 1 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
VCCSEL	N/A	All	Input	Dedicated input that selects which input buffer is used on the configuration input pins; nCONFIG, DCLK, RUnLU, nCE, nWS, nRS, CS, nCS and CLKUSR. The VCCSEL input buffer is powered by V_{CCINT} and has an internal 2.5 kΩ pull-down resistor that is always active. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects the 1.8-V/1.5-V input buffer, and a logic low selects the 3.3-V/2.5-V input buffer. See the "V_{CCSEL} Pins" section for more details.
PORSEL	N/A	All	Input	Dedicated input which selects between a POR time of 2 ms or 100 ms. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects a POR time of about 2 ms and a logic low selects POR time of about 100 ms. The PORSEL input buffer is powered by V_{CCINT} and has an internal 2.5 kΩ pull-down resistor that is always active.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 2 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nIO_PULLUP	N/A	All	Input	Dedicated input that chooses whether the internal pull-ups on the user I/Os and dual-purpose I/Os (DATA[7..0], nWS, nRS, RDYnBSY, nCS, CS, RUnLU, PGM[], CLKUSR, INIT_DONE, DEV_OE, DEV_CLR) are on or off before and during configuration. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) turns off the weak internal pull-ups, while a logic low turns them on. The nIO_PULLUP input buffer is powered by V_{CCINT} and has an internal 2.5 kΩ pull-down resistor that is always active.
MSEL[2..0]	N/A	All	Input	3-bit configuration input that sets the Stratix or Stratix GX device configuration scheme. See Table 1–2 for the appropriate connections. These pins can be connected to V_{CCIO} of the I/O bank they reside in or ground. This pin uses Schmitt trigger input buffers.
nCONFIG	N/A	All	Input	Configuration control input. Pulling this pin low during user-mode causes the FPGA to lose its configuration data, enter a reset state, tri-state all I/O pins. Returning this pin to a logic high level initiates a reconfiguration. If your configuration scheme uses an enhanced configuration device or EPC2 device, nCONFIG can be tied directly to V_{CC} or to the configuration device's nINIT_CONF pin. This pin uses Schmitt trigger input buffers.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 3 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS	N/A	All	Bidirectional open-drain	The device drives nSTATUS low immediately after power-up and releases it after the POR time. Status output. If an error occurs during configuration, nSTATUS is pulled low by the target device. Status input. If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If a configuration device is used, driving nSTATUS low causes the configuration device to attempt to configure the FPGA, but since the FPGA ignores transitions on nSTATUS in user-mode, the FPGA does not reconfigure. To initiate a reconfiguration, nCONFIG must be pulled low. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-kΩ pull-up resistors should not be used on these pins. When using EPC2 devices, only external 10-kΩ pull-up resistors should be used. This pin uses Schmitt trigger input buffers.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 4 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
CONF_DONE	N/A	All	Bidirectional open-drain	Status output. The target FPGA drives the CONF_DONE pin low before and during configuration. Once all configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE. Status input. After all data is received and CONF_DONE goes high, the target device initializes and enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Driving CONF_DONE low after configuration and initialization does not affect the configured device. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-kΩ pull-up resistors should not be used on these pins. When using EPC2 devices, only external 10-kΩ pull-up resistors should be used. This pin uses Schmitt trigger input buffers.
nCE	N/A	All	Input	Active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, it should be tied low. In multi-device configuration, nCE of the first device is tied low while its nCEO pin is connected to nCE of the next device in the chain. The nCE pin must also be held low for successful JTAG programming of the FPGA. This pin uses Schmitt trigger input buffers.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 5 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCEO	N/A	All Multi-Device Schemes	Output	Output that drives low when device configuration is complete. In single device configuration, this pin is left floating. In multi-device configuration, this pin feeds the next device's nCE pin. The nCEO of the last device in the chain is left floating. The voltage levels driven out by this pin are dependent on the V_{CCIO} of the I/O bank it resides in.
DCLK	N/A	Synchronous configuration schemes (PS, FPP)	Input (PS, FPP)	In PS and FPP configuration, DCLK is the clock input used to clock data from an external source into the target device. Data is latched into the FPGA on the rising edge of DCLK. In PPA mode, DCLK should be tied high to V_{CC} to prevent this pin from floating. After configuration, this pin is tri-stated. In schemes that use a configuration device, DCLK is driven low after configuration is done. In schemes that use a control host, DCLK should be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device. This pin uses Schmitt trigger input buffers.
DATA0	I/O	PS, FPP, PPA	Input	Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin. The V_{IH} and V_{IL} levels for this pin are dependent on the V_{CCIO} of the I/O bank that it resides in. After configuration, DATA0 is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings. After configuration, EPC1 and EPC1441 devices tri-state this pin, while enhanced configuration and EPC2 devices drive this pin high.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 6 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DATA[7..1]	I/O	Parallel configuration schemes (FPP and PPA)	Inputs	Data inputs. Byte-wide configuration data is presented to the target device on DATA[7..0]. The V_{IH} and V_{IL} levels for these pins are dependent on the V_{CCIO} of the I/O banks that they reside in. In serial configuration schemes, they function as user I/Os during configuration, which means they are tri-stated. After PPA or FPP configuration, DATA[7..1] are available as a user I/Os and the state of these pin depends on the **Dual-Purpose Pin** settings.
DATA7	I/O	PPA	Bidirectional	In the PPA configuration scheme, the DATA7 pin presents the RDYnBSY signal after the nRS signal has been strobed low. The V_{IL} and V_{IL} levels for this pin are dependent on the V_{CCIO} of the I/O bank that it resides in. In serial configuration schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, DATA7 is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings.
nWS	I/O	PPA	Input	Write strobe input. A low-to-high transition causes the device to latch a byte of data on the DATA[7..0] pins. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nWS is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 7 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nRS	I/O	PPA	Input	Read strobe input. A low input directs the device to drive the RDYnBSY signal on the DATA7 pin. If the nRS pin is not used in PPA mode, it should be tied high. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nRS is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings.
RDYnBSY	I/O	PPA	Output	Ready output. A high output indicates that the target device is ready to accept another data byte. A low output indicates that the target device is busy and not ready to receive another data byte. In PPA configuration schemes, this pin drives out high after power-up, before configuration and after configuration before entering user-mode. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, RDYnBSY is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 8 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCS/CS	I/O	PPA	Input	Chip-select inputs. A low on nCS and a high on CS select the target device for configuration. The nCS and CS pins must be held active during configuration and initialization. During the PPA configuration mode, it is only required to use either the nCS or CS pin. Therefore, if only one chip-select input is used, the other must be tied to the active state. For example, nCS can be tied to GND while CS is toggled to control configuration.In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nCS and CS are available as a user I/Os and the state of these pins depends on the **Dual-Purpose Pin** settings.
RUnLU	N/A if using Remote Configuration; I/O if not	Remote Configuration in FPP, PS or PPA	Input	Input that selects between remote update and local update. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects remote update and a logic low selects local update. When not using remote update or local update configuration modes, this pins is available as general-purpose user I/O pin.
PGM[2..0]	N/A if using Remote Configuration; I/O if not using	Remote Configuration in FPP, PS or PPA	Input	These output pins select one of eight pages in the memory (either flash or enhanced configuration device) when using a remote configuration mode. When not using remote update or local update configuration modes, these pins are available as general-purpose user I/O pins.

Table 1–16 describes the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore during configuration, these pins function as user I/O pins and are tri-stated with weak pull-ups.

Table 1–16. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input. Synchronizes the initialization of one or more devices. This pin is enabled by turning on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software.
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	Status pin. Can be used to indicate when the device has initialized and is in user mode. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 10-kΩ pull-up. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high and the FPGA enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. This pin is enabled by turning on the **Enable INIT_DONE output** option in the Quartus II software.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows the user to override all tri-states on the device. When this pin is driven low, all I/Os are tri-stated. When this pin is driven high, all I/Os behave as programmed. This pin is enabled by turning on the **Enable device-wide output enable (DEV_OE)** option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared. When this pin is driven high, all registers behave as programmed. This pin is enabled by turning on the **Enable device-wide reset (DEV_CLRn)** option in the Quartus II software.

Table 1–17 describes the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. If you plan to use the SignalTap II Embedded Logic Analyzer, you will need to connect the JTAG pins of your device to a JTAG header on your board.

Table 1–17. Dedicated JTAG pins

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}. This pin uses Schmitt trigger input buffers.
TDO	N/A	Output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	N/A	Input	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}. This pin uses Schmitt trigger input buffers.
TCK	N/A	Input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND. This pin uses Schmitt trigger input buffers.
TRST	N/A	Input	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND. This pin uses Schmitt trigger input buffers.

2. Remote System Configuration with Stratix & Stratix GX Devices

S52015-3.1

Introduction

Altera® Stratix® and Stratix GX devices are the first programmable logic devices (PLDs) featuring dedicated support for remote system configuration. Using remote system configuration, a Stratix or Stratix GX device can receive new configuration data from a remote source, update the flash memory content (through enhanced configuration devices or any other storage device), and then reconfigure itself with the new data.

Like all Altera SRAM-based devices, Stratix and Stratix GX devices support standard configuration modes such as passive serial (PS), fast passive parallel (FPP), and passive parallel asynchronous (PPA). You can use the standard configuration modes with remote system configuration.

This chapter discusses remote system configuration of Stratix and Stratix GX devices, and how to interface them with enhanced configuration devices to enable this capability. This document also explains some related remote system configuration topics, such as the watchdog timer, remote system configuration registers, and factory or application configurations files. The Quartus® II software (version 2.1 and later) supports remote system configuration.

Remote Configuration Operation

Remote system configuration has three major parts:

- The Stratix or Stratix GX device receives updated or new data from a remote source over a network (or through any other source that can transfer data). You can implement a Nios™ (16-bit ISA) or Nios® II (32-bit ISA) embedded processor within either a Stratix or Stratix GX device or an external processor to control the read and write functions of configuration files from the remote source to the memory device.
- The new or updated information is stored into the memory device, which can be an enhanced configuration device, industry-standard flash memory device, or any other storage device (see Figure 2–2).
- The Stratix or Stratix GX device updates itself with the new data from the memory.

Figure 2–1 shows the concept of remote system configuration in Stratix and Stratix GX devices.

Figure 2–1. Remote System Configuration with Stratix & Stratix GX Devices

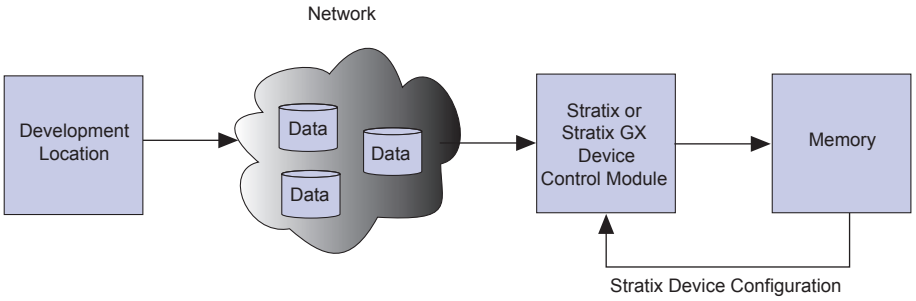
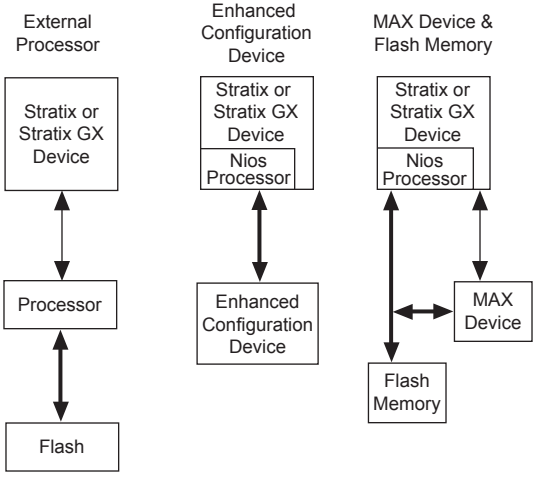


Figure 2–2. Different Options for Remote System Configuration



Remote System Configuration Modes

Stratix and Stratix GX device remote system configuration has two modes: remote configuration mode and local configuration mode. Table 2–1 shows the pin selection settings for each configuration mode.

Table 2–1. Standard, Remote & Local Configuration Options Note (1)

RUnLU (2)	MSEL[2] (3)	MSEL[1..0]	System Configuration Mode	Configuration Mode
–	0	00	Standard	FPP
–	0	01	Standard	PPA
–	0	10	Standard	PS
1	1	00	Remote	FPP
1	1	01	Remote	PPA
1	1	10	Remote	PS
0	1	00	Local	FPP
0	1	01	Local	PPA
0	1	10	Local	PS

Notes to Table 2–1:

(1) For detailed information on standard PS, FPP, and PPA models, see the *Configuring Stratix & Stratix GX Devices* chapter of the *Stratix Device Handbook, Volume 2*.

(2) In Stratix and Stratix GX devices, the RUnLU (remote update/local update) pin, selects between local or remote configuration mode.

(3) The MSEL[2] select mode selects between standard or remote system configuration mode.

Remote Configuration Mode

Using remote configuration mode, you can manage up to seven different application configurations for Stratix and Stratix GX devices. The seven-configuration-file limit is due to the number of pages that the PGM[] pins in the Stratix or Stratix GX device and enhanced configuration devices can select.



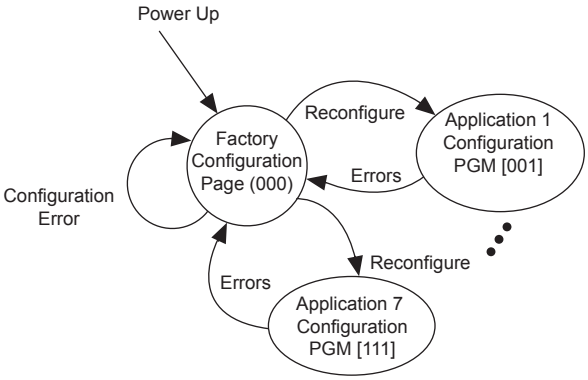
If more than seven files are sent to a system using remote configuration mode, previous files are overwritten.

Stratix and Stratix GX devices support remote configuration mode for PS, FPP, and PPA modes. Specify remote configuration mode by setting the MSEL2 and RUnLU pins to high. (See Table 2–1).

On power-up in remote configuration mode, the Stratix or Stratix GX device loads the user-specified factory configuration file, located in the default page address 000 in the enhanced configuration device. After the device configures, the remote configuration control register points to the

page address of the application configuration that should be loaded into the Stratix or Stratix GX device. If an error occurs during user mode of an application configuration, the device reloads the default factory configuration page. Figure 2–3 shows a diagram of remote configuration mode.

Figure 2–3. Remote Configuration Mode



Local Configuration Mode

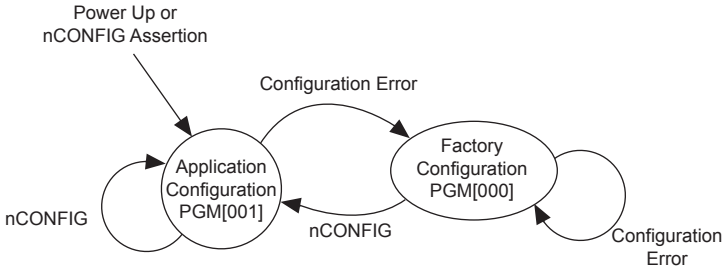
Local configuration mode—a simplified version of remote configuration mode—is suitable for systems that load an application immediately upon power-up. In this mode you can only use one application configuration, which you can update either remotely or locally.

In local configuration mode, upon power-up, or when `nCONFIG` is asserted, the Stratix or Stratix GX device loads the application configuration immediately. Factory configuration loads only if an error occurs during the application configuration's user mode. If you use an enhanced configuration device, page address `001` is the location for the application configuration data, and page address `000` is the location for the factory configuration data.

If the configuration data at page address `001` does not load correctly due to cyclic redundancy code (CRC) failure, or it times-out of the enhanced configuration device, or the external processor times-out, then the factory configuration located at the default page (page address `000`) loads into the Stratix or Stratix GX device.

In local configuration mode (shown in Figure 2–4), the user watchdog timer is disabled. For more information on the watchdog timer, see "Watchdog Timer" on page 2–7.

Figure 2–4. Local Configuration Mode



In local configuration mode, one application configuration is available to the device. For remote or local configuration mode selection, see Table 2–1.

Remote System Configuration Components

The following components are used in Stratix and Stratix GX devices to support remote and local configuration modes:

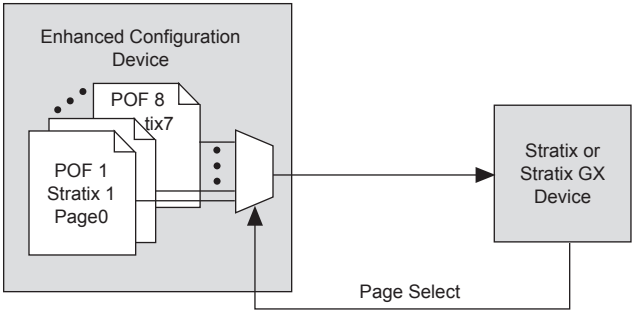
- Page mode feature
- Factory configuration
- Application configuration
- Watchdog timer
- Remote update sub-block
- Remote configuration registers

A description of each component follows.

Page Mode Feature

The page mode feature enables Stratix and Stratix GX devices to select a location to read back data for configuration. The enhanced configuration device can receive and store up to eight different configuration files (one factory and seven application files). Selection of pages to read from is performed through the `PGM[2..0]` pins on the Stratix or Stratix GX device and enhanced configuration devices. These pins in the Stratix or Stratix GX device can be designated user I/O pins during standard configuration mode, but in remote system configuration mode, they are dedicated output pins. Figure 2–5 shows the page mode feature in Stratix or Stratix GX devices and enhanced configuration devices.

Figure 2–5. Page Mode Feature in Stratix or Stratix GX Devices & Enhanced Configuration Devices



Upon power-up in remote configuration mode, the factory configuration (see description below) selects the user-specified page address through the Stratix or Stratix GX PGM[2..0] output pins. These pins drive the PGM[2..0] input pins of the enhanced configuration device and select the requested page in the memory.

If an intelligent host is used instead of an enhanced configuration device, you should create logic in the intelligent host to support page mode settings similar to that in enhanced configuration devices.

Factory Configuration

Factory configuration is the default configuration data setup. In enhanced configuration devices, this default page address is 000. Factory configuration data is written into the memory device only once by the system manufacturer and should not be remotely updated or altered. In remote configuration mode, the factory configuration loads into the Stratix or Stratix GX device upon power-up.

The factory configuration specifications are as follows:

- Receives new configuration data and writes it to the enhanced configuration or other memory devices
- Determines the page address for the next application configuration that should be loaded to the Stratix or Stratix GX device
- Upon an error in the application configuration, the system reverts to the factory configuration
- Determines the reason for any application configuration error
- Determines whether to enable or disable the user watchdog timer for application configurations

- Determines the user watchdog timer's settings if the timer is enabled (remote configuration mode)
- If the user watchdog timer is not reset after a predetermined amount of time, it times-out and the system loads the factory configuration data back to the Stratix or Stratix GX device

If a system encounters an error while loading application configuration data, or if the device re-configures due to `nCONFIG` assertion, the Stratix or Stratix GX device loads the factory configuration. The remote system configuration register determines the reason for factory re-configuration. Based on this information, the factory configuration determines which application configuration needs to be loaded.

Application Configuration

The application configuration is the configuration data received from the remote source and updated into different locations or pages of the memory storage device (excluding the factory default page).

Watchdog Timer

A watchdog timer is a circuit that determines whether another mechanism functions properly. The watchdog timer functions like a time-delay relay that remains in the reset state while an application runs properly. This action periodically sends a reset command from the working application to the watchdog timer. Stratix and Stratix GX devices are equipped with a built-in watchdog timer for remote system configuration.

A user watchdog timer prevents a faulty application configuration from indefinitely stalling the Stratix or Stratix GX device. The timer functions as a counter that counts down from an initial value, which is loaded into the device from the factory configuration. This is a 29-bit counter, but you use only the upper 12 bits to set the value for the watchdog timer. You specify the counter value according to your design needs.

The timer begins counting once the Stratix or Stratix GX device goes into user mode. If the application configuration does not reset the user watchdog timer after the specified time, the timer times-out. At this point, the Stratix or Stratix GX device is re-configured by loading the factory configuration and resetting the user watchdog timer.



The watchdog timer is disabled in local configuration mode.

Remote Update Sub-Block

The remote update sub-block is responsible for administrating the remote configuration feature. This sub-block, which is controlled by a remote configuration state machine, generates the control signals required to control different remote configuration registers.

Remote Configuration Registers

Remote configuration registers are a series of registers required to keep track of page addresses and the cause of configuration errors. Table 2–2 gives descriptions of the registers' functions. You can control both the update and shift registers; the status and control registers are controlled by internal logic, but can be read via the shift register.

Table 2–2. Remote Configuration Registers

Register	Description
Control register	This register contains the current page address, the watchdog timer setting, and one bit specifying if the current configuration is a factory or application configuration. During a capture in an application configuration, this register is read into the shift register.
Update register	This register contains the same data as the control register, except that it is updated by the factory configuration. The factory configuration updates the register with the values to be used in the control register on the next re-configuration. During capture in a factory configuration, this register is read into the shift register.
Shift register	This register is accessible by the core logic and allows the update, status, and control registers to be written and sampled by the user logic. The update register can only be updated by the factory configuration in remote configuration mode.
Status register	This register is written into by the remote configuration block on every re-configuration to record the cause of the re-configuration. This information is used by factory configuration to determine the appropriate action following a re-configuration.

Figure 2–6 shows the control, update, shift, and status registers and the data path used to control remote system configuration.

Figure 2–6. Remote Configuration Registers & Related Data Path

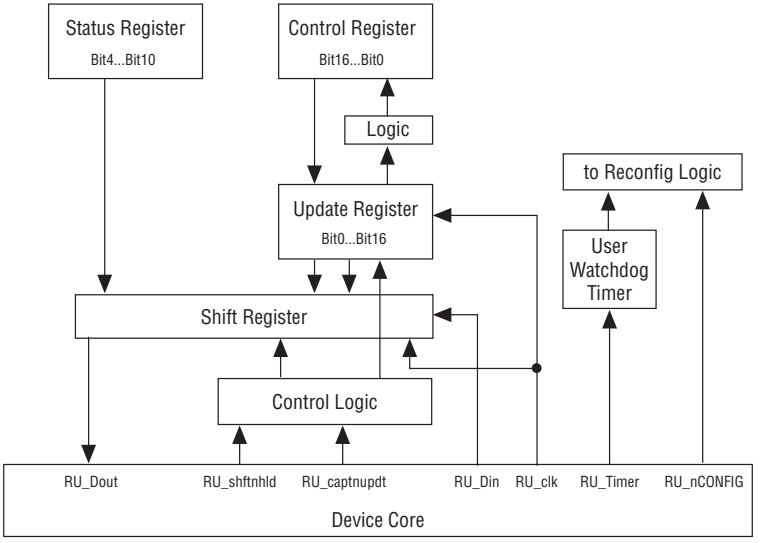


Table 2–3 describes the user configuration signals that are driven to/from the device logic array. The remote configuration logic has one input signal to the device logic array and six output signals from the device logic array.

Table 2–3. User Configuration Signals To/From Device Core (Part 1 of 2)

Signal Name	To/From Device Core	Description
RU_Timer	Output from the core to the remote update block	Request from the application to reset the user watchdog timer with its initial count. A falling edge of this signal triggers a reset of the user watchdog timer.
RU_nCONFIG	Output from the core to the remote update block	When driven low, this signal triggers the device to reconfigure. If requested by the factory configuration, the application configuration specified in the remote update control register is loaded. If requested by the application configuration, the factory configuration is loaded.
RU_Clk	Output from the core to the remote update block	Clocks the remote configuration shift register so that the contents of the status and control registers can be read out, and the contents of update register can be loaded. The shift register latches data on the rising edge of the RU_Clk.

Table 2–3. User Configuration Signals To/From Device Core (Part 2 of 2)

Signal Name	To/From Device Core	Description
RU_shftnhld	Output from the core to the remote update block	If its value is "1", the remote configuration shift register shifts data on the rising edge of RU_Clk. It its value is "0" and RU_captnupdt is "0", the shift register updates the update register. If its value is "0", and RU_captnupdt is "1", the shift register captures the status register and either the control or update register (depending on whether the configuration is factory or application).
RU_captnupdt	Output from the core to the remote update block	When RU_captnupdt is at value "1" and RU_shftnhld is at value"0", the system specifies that the remote configuration shift register should be written with the content of the status register and either the update register (in a factory configuration) or the control register (in an application configuration). This shift register is loaded on the rising edge of RU_Clk. When RU_captnupdt is at value "0" and RU_shftnhld is at value"0", the system specifies that the remote configuration update register should be written with the content of the shift register in a factory configuration. The update register is loaded on the rising edge of RU_Clk. This pin is enabled only for factory configuration in remote configuration mode (it is disabled for the application configuration in remote configuration or for local configuration modes). If RU_shftnhld is at value "1", RU_captnupdt has no function.
RU_Din	Output from the core to the remote update block	Data to be written into the remote configuration shift register on the rising edge of RU_Clk. To load into the shift register, RU_shftnhld must be asserted.
RU_Dout	Input to the core from the remote update block	Output of the remote configuration shift register to be read by core logic. New data arrives on each rising edge of RU_Clk.

All of the seven device core signals (see Figure 2–6), are enabled for both remote and local configuration for both factory and application configuration, except RU_Timer and RU_captnupdt. Figure 2–7 and Table 2–4 specify the content of control register upon power-on reset (POR).

The difference between local configuration and remote configuration is how the control register is updated during a re-configuration and which core signals are enabled.

Figure 2–7. Remote System Configuration Control Register

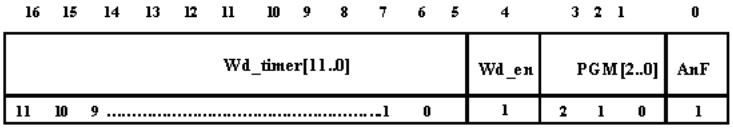


Table 2–4 shows the content of the control register upon POR.

Table 2–4. Control Register Contents

Parameter	Definition	POR Reset Value	Comment
AnF	Current configuration is factory or applications	1 bit '1'	Applications
		1 bit '0'	Factory
PGM[2..0]	Page mode selection	3 bits '001'	Local configuration
		3 bits '000'	Remote configuration
Wd_en	User watchdog timer enable	1 bit '0'	–
Wd_timer [11..0]	User watchdog timer time-out value	12 bits '0'	High order bits of 29 bit counter

The status register specifies the reason why re-configuration has occurred and determines if the re-configuration was due to a CRC error, nSTATUS pulled low due to an error, the device core caused an error, nCONFIG was reset, or the watchdog timer timed-out. Figure 2–8 and Table 2–5 specify the content of the status register.

Figure 2–8. Remote System Configuration Status Register

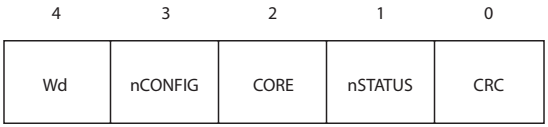


Table 2–5 shows the content of the status register upon POR.

Table 2–5. Status Register Contents

Parameter	Definition	POR Reset Value
CRC (from configuration)	CRC caused re-configuration	1 bit '0'
nSTATUS	nSTATUS caused re-configuration	1 bit '0'
CORE *(1)*	Device core caused re-configuration	1 bit '0'
nCONFIG	NCONFIG caused re-configuration	1 bit '0'
Wd	Watchdog Timer caused re-configuration	1 bit '0'

Note to Table 2–5:

(1) Core re-configuration enforces the system to load the application configuration data into the Stratix or Stratix GX device. This occurs after factory configuration specifies the appropriate application configuration data.

Quartus II Software Support

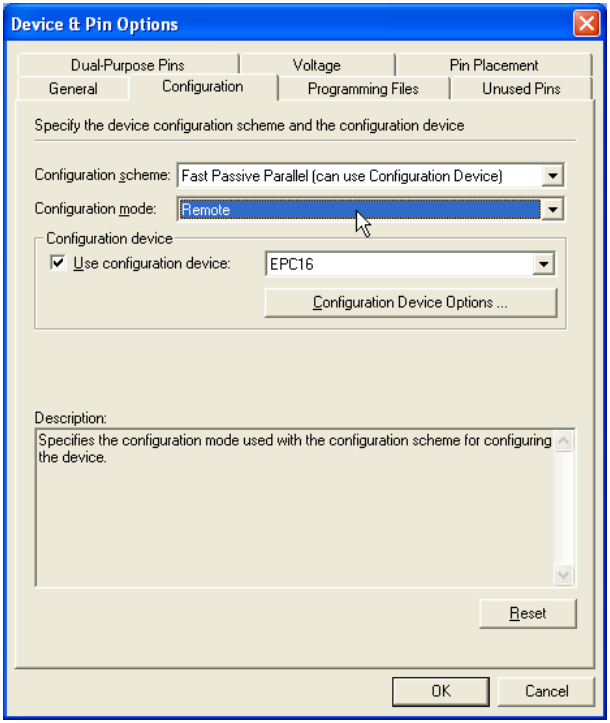
The Quartus II software supports implementation of both remote and local configuration modes in your Stratix or Stratix II device. To include the remote or local configuration feature to your design, select remote or local as the configuration mode under the **Device & Pin Options** compiler settings (prior to compilation). This selection reserves the dual-purpose RUnLU and PGM[2:0] pins for use as dedicated inputs in remote/local configuration modes.

To set the configuration mode as remote or local, follow these steps (See Figure 2–9):

1. Open the **Device & Pin Options** settings window under the **Assignments** menu.
2. Select **Device & Pin Options** dialog box. The **Device & Pin Options** dialog box is displayed.
3. Click the **Configuration** tab.
4. In the **Configuration mode** list, select **Remote** or **Local**.

The Standard mode selection disables the remote system configuration feature. In addition to the mode selection, you can specify the configuration scheme and configuration device (if any) used by your setup.

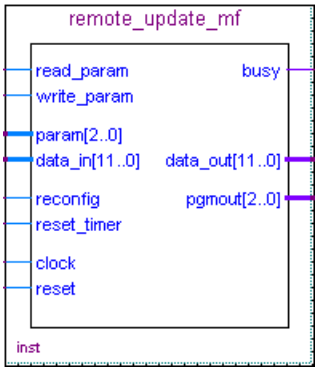
Figure 2–9. Device & Pin Options Dialog Box



Additionally, the remote configuration mode requires you to either instantiate the `altremote_update` megafunction or the WYSIWYG (what-you-see-is-what-you-get) atom into your design. Without this atom or megafunction, you are not be able to access the dedicated remote configuration circuitry or registers within the Stratix or Stratix GX device. See Figure 2–10 for a symbol of the `altremote_update` megafunction.

The local configuration mode, however, can be enabled with only the device **Configuration Options** compiler setting.

Figure 2–10. altremote_update Megafunction Symbol



altremote_update Megafunction

A remote update megafunction, `altremote_update`, is provided in the Quartus II software to provide a memory-like interface to allow for easy control of the remote update parameters. Tables 2–6 and 2–7 describe the input and output ports available on the `altremote_update` megafunction. Table 2–8 shows the `param[2..0]` bit settings.

Table 2–6. Input Ports of the altremote_update Megafunction (Part 1 of 2)

Port Name	Required	Source	Description
clock	Y	Logic Array	Clock input to the `altremote_update` block. All operations are performed with respects to the rising edge of this clock.
reset	Y	Logic Array	Asynchronous reset, which is used to initialize the remote update block. To ensure proper operation, the remote update block must be reset before first accessing the remote update block. This signal is not affected by the busy signal and will reset the remote update block even if busy is logic high. This means that if the reset signal is driven logic high during writing of a parameter, the parameter will not be properly written to the remote update block.
reconfig	Y	Logic Array	When driven logic high, reconfiguration of the device is initiated using the current parameter settings in the remote update block. If busy is asserted, this signal is ignored. This is to ensure all parameters are completely written before reconfiguration begins.
reset_timer	N	Logic Array	This signal is required if you are using the watchdog timer feature. A logic high resets the internal watchdog timer. This signal is not affected by the busy signal and can reset the timer even when the remote update block is busy. If this port is left connected, the default value is 0.

Table 2–6. Input Ports of the altremote_update Megafunction (Part 2 of 2)

Port Name	Required	Source	Description
read_param	N	Logic Array	Once `read_param` is sampled as a logic high, the busy signal is asserted. While the parameter is being read, the busy signal remains asserted, and inputs on `param[]` are ignored. Once the busy signal is deactivated, the next parameter can be read. If this port is left unconnected, the default value is 0.
write_param	N	Logic Array	This signal is required if you intend on writing parameters to the remote update block. When driven logic high, the parameter specified on the `param[]` port should be written to the remote update block with the value on `data_in[]`. The number of valid bits on `data_in[]` is dependent on the parameter type. This signal is sampled on the rising edge of clock and should only be asserted for one clock cycle to prevent the parameter from being re-read on subsequent clock cycles. Once `write_param` is sampled as a logic high, the busy signal is asserted. While the parameter is being written, the busy signal remains asserted, and inputs on `param[]` and `data_in[]` are ignored. Once the busy signal is deactivated, the next parameter can be written. This signal is only valid when the `Current_Configuration` parameter is factory since parameters cannot be written in application configurations. If this port is left unconnected, the default value is 0.
param[2..0]	N	Logic Array	3-bit bus that selects which parameter should be read or written. If this port is left unconnected, the default value is 0.
data_in[11..0]	N	Logic Array	This signal is required if you intend on writing parameters to the remote update block 12-bit bus used when writing parameters, which specifies the parameter value. The parameter value is requested using the `param[]` input and by driving the `write_param` signal logic high, at which point the busy signal goes logic high and the value of the parameter is captured from this bus. For some parameters, not all 12-bits will be used in which case only the least significant bits will be used. This port is ignored if the `Current_Configuration` parameter is set to an application configuration since writing of parameters is only allowed in the factory configuration. If this port is left unconnected, the default values is 0.

Note to Table 2–6:

(1) Logic array source means that you can drive the port from internal logic or any general-purpose I/O pin.

Table 2–7. Output Ports of the altremote_update Megafunction

Port Name	Required	Destination	Description
busy	Y	Logic Array	When this signal is a logic high, the remote update block is busy either reading or writing a parameter. When the remote update block is busy, it ignores its `data_in[]`, `param[]`, and `reconfig` inputs. This signal will go high when `read_param` or `write_param` is asserted and will remain asserted until the operation is complete.
pgm_out[2..0]	Y	PGM[2..0] pins	3-bit bus that specifies the page pointer of the configuration data to be loaded when the device is reconfigured. This port must be connected to the `PGM[]` output pins, which should be connected to the external configuration device
data_out[11..0]	N	Logic Array	12-bit bus used when reading parameters, which reads out the parameter value. The parameter value is requested using the `param[]` input and by driving the `read_param` signal logic high, at which point the busy signal will go logic high. When the busy signal goes low, the value of the parameter will be driven out on this bus. The `data_out[]` port is only valid after a read_param has been issued and once the busy signal is de-asserted. At any other time, its output values are invalid. For example, even though the `data_out[]` port may toggle during a writing of a parameter, these values are not a valid representation of what was actually written to the remote update block. For some parameters, not all 12-bits will be used in which case only the least significant bits will be used.

Note to Table 2–7:
(1) Logic array destination means that you can drive the port to internal logic or any general-purpose I/O pin.

Table 2–8. Parameter Settings for the altremote_update Megafunction (Part 1 of 2)

Selected Parameter	param[2..0] bit setting	width of parameter value	POR Reset Value	Description
Status Register Contents	000	5	5 bit '0	Specifies the reason for re-configuration, which could be caused by a CRC error during configuration, `nSTATUS` being pulled low due to an error, the device core caused an error, `nCONFIG` pulled low, or the watchdog timer timed-out. This parameter can only be read.
Watchdog Timeout Value	010	12	12 bits '0	User watchdog timer time-out value. Writing of this parameter is only allowed when in the factory configuration.
Watchdog Enable	011	1	1 bit '0	User watchdog timer enable. Writing of this parameter is only allowed when in the factory configuration

Table 2–8. Parameter Settings for the altremote_update Megafunction (Part 2 of 2)

Selected Parameter	param[2..0] bit setting	width of parameter value	POR Reset Value	Description
Page select	100	3	3 bit '001' - Local configuration 3 bit '000' - Remote configuration	Page mode selection. Writing of this parameter is only allowed when in the factory configuration.
Current configuration (AnF)	101	1	1 bit '0' - Factory 1 bit '1' - Application	Specifies whether the current configuration is factory or and application configuration. This parameter can only be read.
Illegal values	001			
	110			
	111			

Remote Update WYSIWYG ATOM

An alternative to using the `altremote_update` megafunction is to directly instantiate the remote update WYSIWYG atom. This atom should be included in the factory configuration and any application configuration image to access the remote configuration shift registers.

When implementing the atom, you should consider following:

1. Only one atom can be used in the circuit; more than one gives a no-fit.
2. All signals for the cell must be connected. The clock port (`CLK`) must be connected to a live cell. The others can be constant V_{CC} or GND.
3. The `pgmout` port must be connected and must feed `PGM[2.0]` output pins (it cannot be connected to anything else but output pins).
4. The Quartus II software reserves `RUnLU` as an input pin, and you must connect it to V_{CC}.

The Stratix and Stratix GX remote update atom ports are:

Stratix_rublock <*rublock_name*>
(
.clk(<*clock source*>**),**
.shiftnld(<*shiftnld source*>**),**
.captnupdt(<*shiftnld source*>**),**
.regin(<*regin input source from the core*>**),**
.rsttimer(<*input signal to reset the watchdog timer*>**),**
.config(<*input signal to initiate configuration*>**),**
.regout(<*data output destination to core*>**),**
.pgmout(<*program output destinations to pins*>**)**

Table 2–9 shows the remote update block input and output port names and descriptions.

Table 2–9. Remote Update Block Input & Output Ports

Ports	Definition
<*rublock_name*>	The unique identifier for the instance. This identifier name can be anything as long as it is legal for the given description language (i.e., Verilog, VHDL, AHDL, etc.). This field is required.
.clk(<*clock source*>)	Designates the clock input of this cell. All operation is with respect to the rising edge of this clock. This field is required.
.shiftnld(<*shiftnld source*>)	An input into the remote configuration block. When **.shiftnld** = 1, the data shifts from the internal shift registers to the `regout` port at each rising edge of `clk`, and the data also shifts into the internal shift registers from `regin` port. This field is required.
.captnupdt(<*shiftnld source*>)	An input into the remote configuration block. This controls the protocol of when to read the configuration mode or when to write into the registers that control the configuration. This field is required.
.regin(<*regin input source from the core*>)	An input into the configuration block for all data loading into the core. The data shifts into the internal registers at the rising edge of `clk`. This field is required.
.rsttimer(<*input signal to reset the watchdog timer*>)	An input into the watchdog timer of the remote update block. When this is high, it resets the watchdog timer. This field is required.
.config(<*input signal to initiate configuration*>)	An input into the configuration section of the remote update block. When this signal goes high, the part initiates a re-configuration. This field is required.
.regout(<*data output destination to core*>)	A 1-bit output, which is the output of the internal shift register, and updated every rising edge of `clk`. The data coming out depends on the control signals. This field is required.
.pgmout(<*program output destinations to pins*>)	A 3-bit bus. It should always be connected only to output pins (not `bidir` pins). This bus gives the page address (`000` to `111`) of the configuration data to be loaded when the device is getting configured. This field is required.



For more information on the control signals for the remote block, see Table 2–3 on page 2–9.

Using Enhanced Configuration Devices

This section describes remote system configuration of Stratix and Stratix GX devices with the Nios embedded processor using enhanced configuration devices. Enhanced configuration devices are composed of a standard flash memory and a controller. The flash memory stores configuration data, and the controller reads and writes to the flash memory.

In remote system configuration, only PS and FPP modes are supported using an enhanced configuration device. A Stratix or Stratix GX device running a Nios embedded processor can receive data from a remote source through a network or any other appropriate media. A specific page of the enhanced configuration device stores the received data.

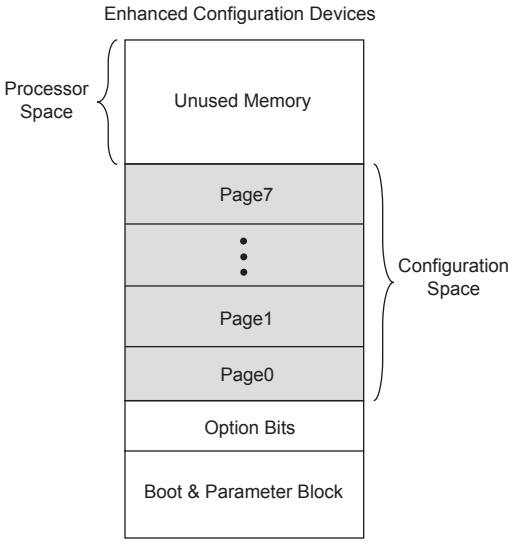
This scheme uses the page mode option in Stratix and Strati GX devices. Up to eight pages can be stored in each enhanced configuration device, each of which can store a configuration file.

In enhanced configuration devices, a page is a section of the flash memory space. Its boundary is determined by the Quartus II software (the page size is programmable). In the software, you can specify which configuration file should be stored in which page within the flash memory. To access the configuration file on each page, set the three input pins (`PGM[2..0]`), which provide access to all eight pages. Because the `PGM[2..0]` pins of an enhanced configuration device connect to the same pins of the Stratix or Stratix GX device, the Stratix or Stratix GX device selects one of the eight memory pages as a target location to read from. Figure 2–11 shows the allocation of different pages in the enhanced configuration device.



For more information on enhanced configuration devices, see the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* and the *Altera Enhanced Configuration Devices* chapter.

Figure 2–11. Memory Map in Enhanced Configuration Device

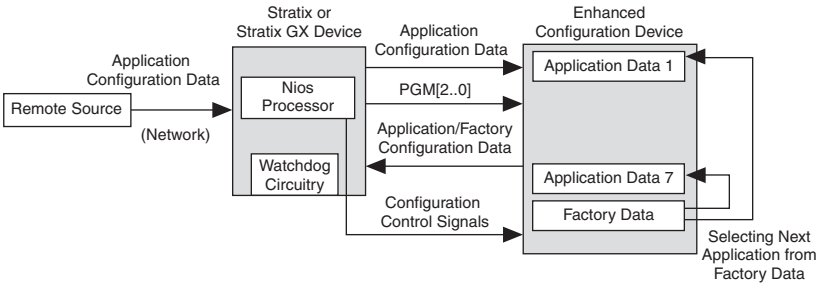


When the Stratix or Stratix GX device powers-up in remote configuration mode, the devices loads configuration data located at page address 000. You should always load the factory default configuration data at this location and make sure this information is not altered.

The factory configuration contains information to determine the next application configuration to load into the Stratix or Stratix GX device. When the Stratix or Stratix GX device successfully loads the application configuration from the page selected by the PGM[2..0] pins, it enters user mode.

In user mode, the Nios embedded processor (or any other logic) assists the Stratix or Stratix GX device in detecting remote system configuration information. In remote system configuration, the Nios embedded processor receives the incoming data from the remote source via the network, writes it to the ECP16 enhanced configuration device, and then initiates loading of the factory configuration into the Stratix or Stratix GX device. Factory configuration reads the remote configuration status register and determines the appropriate application configuration to load into the Stratix or Stratix GX device. Figure 2–12 shows the remote system configuration.

Figure 2–12. Remote System Configuration Using Enhanced Configuration Devices



The user watchdog timer in Stratix and Stratix GX devices ensures that an application configuration has loaded successfully and checks if the application configuration is operating correctly in user mode. The watchdog timer must be continually reset by the user logic. If an error occurs while the application configuration loads, or if the watchdog timer times-out during user mode, the factory configuration is reloaded to prevent the system from halting in an erroneous state. Figure 2–3 on page 2–4 illustrates the remote configuration mode.

Upon power-up in local configuration scheme, the application configuration at page 001 (`PGM[001]` of the enhanced configuration device) loads into the Stratix or Stratix GX device. This application can be remotely or locally updated. If an error occurs during loading of the configuration data, the factory configuration loads automatically (see Figure 2–4 on page 2–5). The rest is identical to remote configuration mode.

Local Update Programming File Generation

This section describes the programming file generation process for performing remote system upgrades. The Quartus II convert programming files (CPF) utility generates the initial and partial programming files for configuration memory within the enhanced configuration devices.

The two pages that local configuration mode uses are a factory configuration stored at page 000, and an application configuration stored at page 001. The factory configuration cannot be updated after initial production programming. However, the application configuration can be erased and reprogrammed after initial system deployment.

In local update mode, you would first create the initial programming file with the factory configuration image and a version of the application configuration. Subsequently, you can generate partial programming files to update the application configuration (stored in page 001). Quartus II CPF can create partial programming files in .**hex** (Hexadecimal file), JAM, **.jbc** (JAM Byte-Code File), and POF formats.

In addition to the two configuration pages, user data or processor code can also be pre-programmed in the bottom boot and main data areas of the enhanced configuration device memory. The CPF utility accepts a HEX input file for the bottom and main data areas, and includes this data in the POF output file. However, this is only supported for initial programming file generation. Partial programming file generation for updating user HEX data is not supported, but can be performed using the enhanced configuration device external flash interface.

Initial Programming File Generation

The initial programming file includes configuration data for both factory and application configuration pages. The enhanced configuration device option's bits are always located between byte addresses `0x00010000` and `0x0001003F`. Also, page 0 always starts at `0x00010040` while its end address is dependent on the size of the factory configuration data.

The two memory allocation options that exist for the application configuration are auto addressing and block addressing. In auto addressing mode, Quartus II automatically allocates memory for the application configuration. All the configuration memory sectors that are not used by the page 0 factory configuration are allocated for page 1. The memory allocated is maximized to allow future versions of the application configuration to grow and have bigger configuration files (when the compression feature is enabled). Processor or user data storage (HEX input file) is only supported by the bottom boot area in auto addressing mode.

The following steps and screen shot (see Figure 2–13) describe initial programming file generation with auto addressing mode.

1. Open the **Convert Programming Files** window from the **File** menu.
2. Select **Programmer Object File (*.pof)** from the drop-down list titled **Programming File Type**.

3. Select the enhanced configuration device used (EPC4, EPC8, EPC16), and the mode used (1-bit Passive Serial or Fast Passive Parallel). Only during the initial programming file generation can you specify the **Options**, **Configuration Device**, or **Mode** settings. While generating the partial programming file, all of these settings are grayed out and inaccessible.

4. In the **Input files to convert** box, highlight **SOF Data** at **Page 0** and click **Add File**. Select input SOF file(s) for this configuration page and insert them.

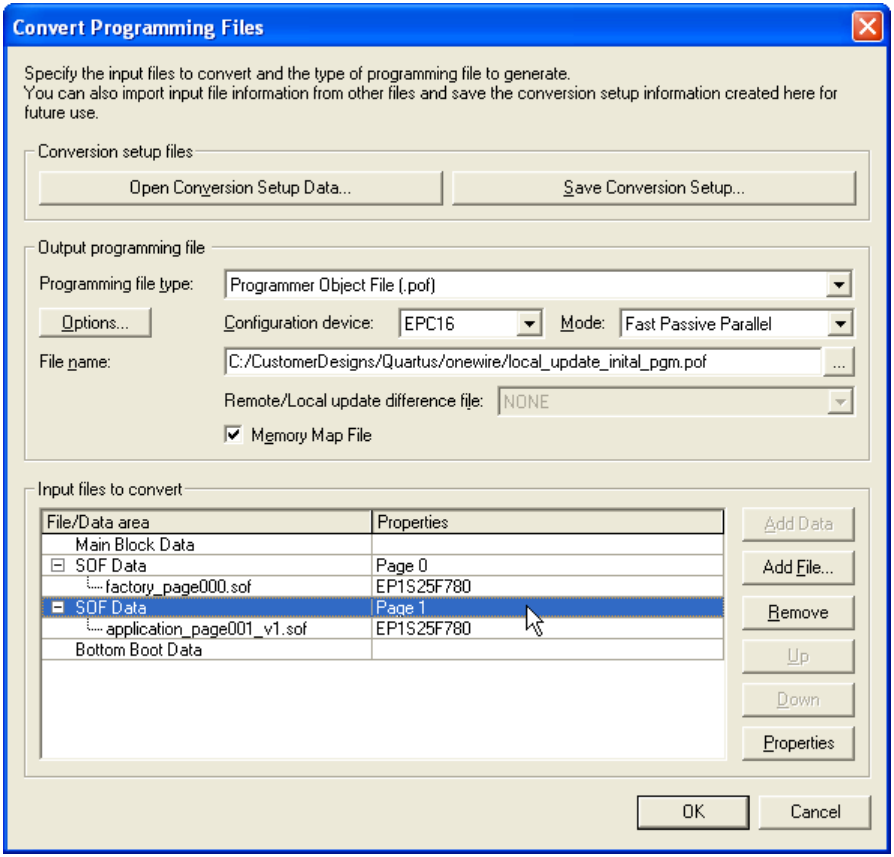
5. Repeat Step 4 for the **Page 1** application configuration page.

6. Check the **Memory Map File** box to generate a memory map output file that specifies the start/end addresses of each configuration page and user data blocks.

7. Save the CPF setup (optionally), by selecting **Save Conversion Setup**… and specifying a name for the **.cof** output file.

8. Click **OK** to generate initial programming and memory map files.

Figure 2–13. CPF Setup for Initial Programming File (Auto Addressing)



A sample memory map output file for the preceding setup is shown below. Configuration option bits and page 0 data occupy main flash sectors 0 through 4. See the *Sharp LHF16J06 Flash memory used in EPC16 devices* Data Sheet at **www.altera.com** to correlate memory addresses to the EPC16 flash sectors. In auto addressing mode, page 1 allocates all unused flash sectors. For this example, this unused area includes main sectors 5 through 30, and all of the bottom boot sectors. While this large portion of memory is allocated for page 1, the real application configuration data is top justified within this region with filler 1'b1 bits in lower memory addresses. Notice that the page 1 configuration data

wraps around the top of the memory and fills up the bottom boot area. The wrap around does not occur if the bottom boot area is used for processor/user HEX data file storage.

Block	Start Address	End Address
OPTION BITS	0x00010000	0x0001003F
PAGE 0	0x00010040	0x00054CC8
PAGE 1	0x001CB372	0x0000FFFD wrapped around

The block addressing mode allows better control of flash memory allocation. You can allocate a specific flash memory region for each application configuration page. This allocation is done by specifying a block starting and block ending address. While selecting the size of the region, you should account for growth in compressed configuration bitstream sizes due to design changes and additions. In local update mode, all configuration data is top justified within this allotted memory. In other words, the last byte of configuration data is stored such that it coincides with the highest byte address location within the allotted space. Lower unused memory address locations within the allotted region are filled with 1's. These filler bits are transmitted during a configuration cycle using page 1, but are ignored by the Stratix device. The memory map output file provides the exact byte address where real configuration data for page 1 begins. Note that any partial update of page 1 should erase all allotted flash sectors before storing new configuration data.

In the block addressing mode, HEX input files can be optionally added to the bottom boot and main flash data areas (one HEX file per area is allowed). The HEX file can be stored with relative addressing or absolute addressing. For more information on relative and absolute addressing, see the *Using Altera Enhanced Configuration Devices* chapter of the *Configuration Handbook*.

Figures 2–14 and 2–15, and the following steps illustrate generating an initial programming file with block addressing for local update mode. This example also illustrates preloading user HEX data into bottom boot and main flash sectors.

1. Open the **Convert Programming Files** window from the **File** menu.

2. Select **Programmer Object File (.pof)** from the drop-down list titled **Programming file type**.

3. Select the enhanced configuration device (EPC4, EPC8, EPC16), and the mode used (1-bit Passive Serial or Fast Passive Parallel). Only during the initial programming file generation can you specify the **Options**, **Configuration device**, or **Mode** settings. While generating the partial programming file, all of these settings are grayed out and inaccessible.

4. In the **Input files to convert** box, highlight **SOF Data** at **Page 0** and click **Add File**. Select input SOF file(s) for this configuration page and insert them.

5. Repeat Step 4 for the Page 1 application configuration page.

6. For enabling block addressing, select the **SOF Data** entry for **Page 1**, and click **Properties**. This opens the **SOF Data Properties** dialog box (see Figure 2–15).

7. Pick **Block** from the **Address Mode** drop down selection, and enter 32-bit Hexadecimal byte address for block **Starting Address** and **Ending Address**. Note that for partial programming support, the block start and end addresses should be aligned to a flash sector boundary. This prevents two configuration pages from overlapping within the same flash boundary. See the flash memory datasheet for data sector boundary information. Click **OK** to save SOF data properties.

8. Check the **Memory Map File** box to generate a memory map output file that specifies the start/end addresses of each configuration page and user data blocks.

9. Save the CPF setup (optionally), by selecting **Save Conversion Setup**… and specifying a name for the COF output file.

10. Click **OK** to generate initial programming and memory map files.

Figure 2–14. CPF Setup for Initial Programming File Generation (Block Addressing)

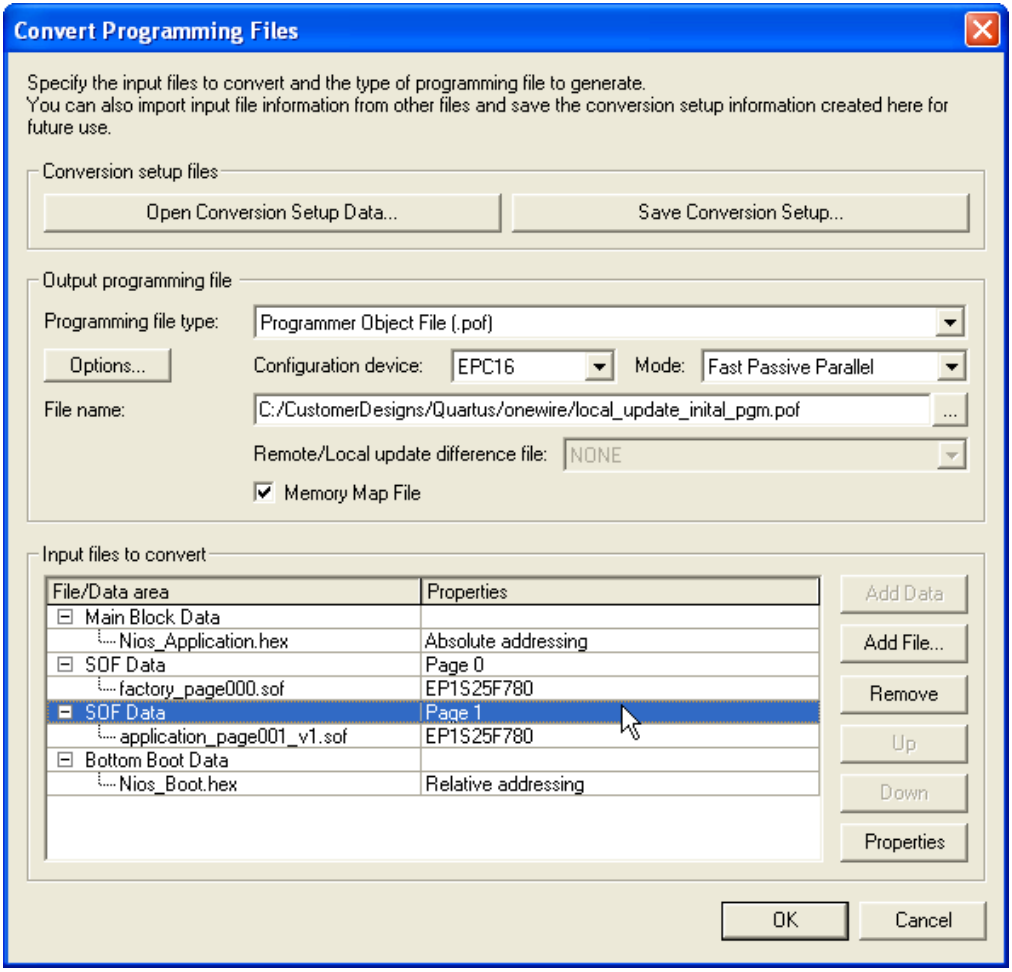
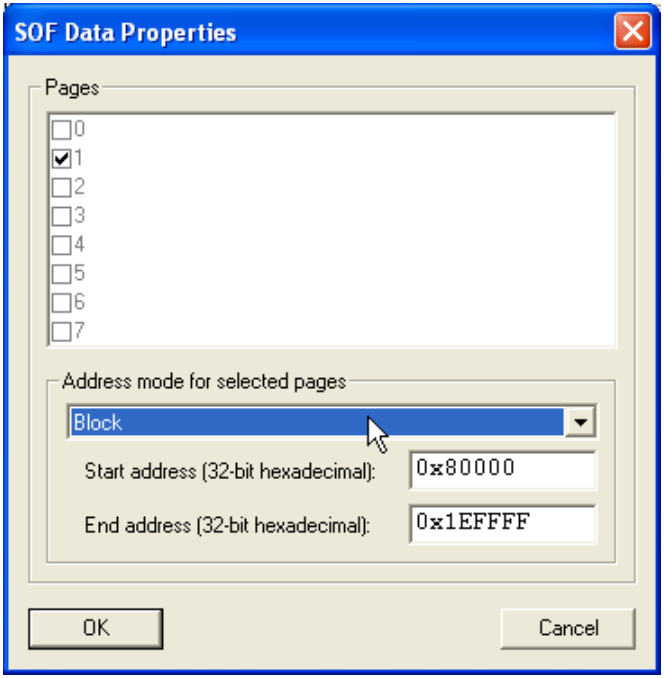


Figure 2–15. Specifying Block Addresses for Application Configuration



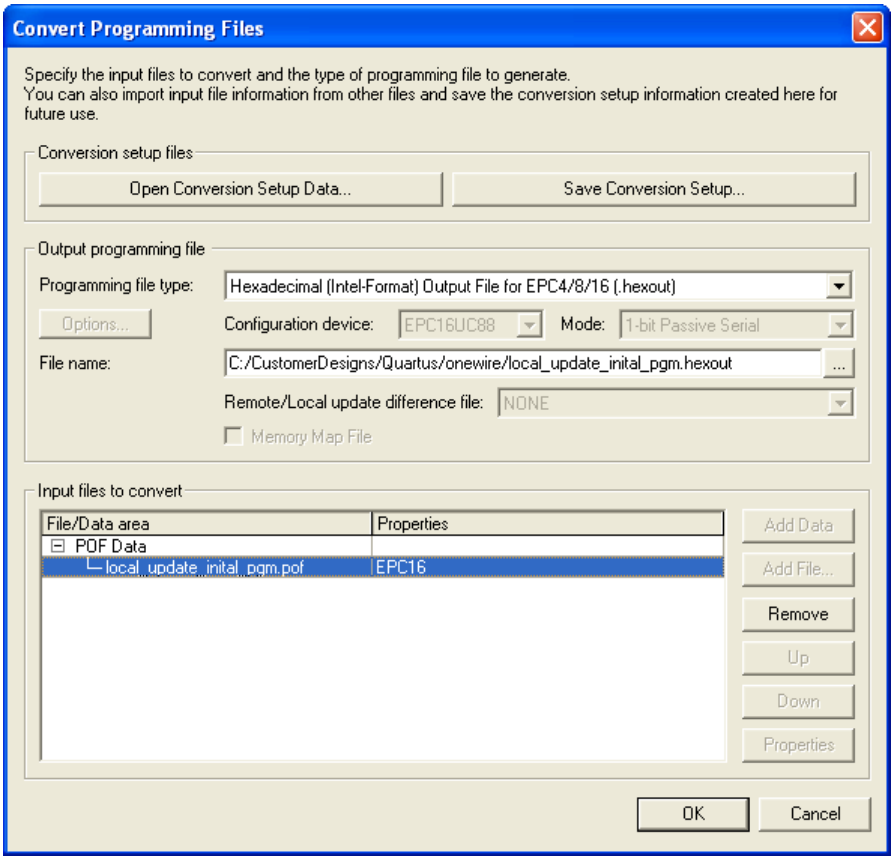
A sample memory map output file for the preceding example is shown below. Note that the allocated memory for page 1 is between 0x00080000 and 0x001EFFFF, while the actual region used by the current application configuration bitstream is between 0x001AB36C and 0x001EFFF7. The configuration data is top justified within the allocated SOF data region.

Block	Start Address	End Address
BOTTOM BOOT	0x00000000	0x000001FF
OPTION BITS	0x00010000	0x0001003F
PAGE 0	0x00010040	0x00054CC8
PAGE 1	0x001AB36C	0x001EFFF7
TOP BOOT/MAIN	0x001F0000	0x001F01FF

Also note that the HEX data stored in the main data area uses absolute addressing. If relative addressing were to be used, the main data contents would be justified with the top (higher address locations) of the memory.

The initial programming file (POF) can be converted to an Intel Hexadecimal format file (*.HEXOUT) using the Quartus II CPF utility. See Figure 2–16.

Figure 2–16. Converting POF Programming File to Intel HEX Format



Partial Programming File Generation

The enhanced Quartus II CPF utility allows an existing application configuration page to be replaced with new data. Partial programming files are generated to perform such configuration data updates.

In order to generate a partial programming file, you have to input the initial programming file (POF) and new configuration data (SOF) to the Quartus II CPF utility. In addition, you have to specify the addressing mode (auto or manual) that was used during initial POF creation. And if

block addressing was used, you should specify the block start and end addresses. With this information, Quartus II ensures that the partial programming file only updates the flash region containing the application configuration. The factory configuration (page 0) and configuration option bits are left unaltered during this process.

Figure 2–17 and the following steps illustrate generation of a partial programming file:

1. Open the **Convert Programming Files** window from the **File** menu.
2. Select **Programmer Object File for Local Update (.pof)** from the drop-down list titled **Programming file type**, and specify an output **File name**.
3. In the **Input files to convert box**, highlight **POF Data** and click **Add File**. Select the initial programming POF file for this design and insert it.
4. In the **Input files to convert box**, highlight **SOF Data** and click **Add File.** Select the new application configuration bitstream (SOF) and insert it.
5. When using block addressing, select the **SOF Data** entry for **Page 1**, and click **Properties**. This opens the **SOF Data Properties** dialog box (see Figure 2–18).
6. Pick **Block** from the **Address Mode** drop down selection, and enter 32-bit Hexadecimal byte address for block **Starting Address** and **Ending Address**. These addresses should be identical to those used to generate the initial programming file. Click **OK** to save SOF data properties.
7. Check the **Memory Map File** box to generate a memory map output file that specifies the start/end addresses of the new application configuration data in page 1.
8. Pick a local update difference file from the **Remote/Local Update Difference File** drop-down menu. You can select between an Intel HEX, JAM, JBC, and POF output file types. The output file name is the same as the POF output file name with a _dif suffix.
9. Save the CPF setup (optionally), by selecting **Save Conversion Setup…** and specifying a name for the COF output file.
10. Click **OK** to generate initial programming and memory map files.

Figure 2–17. Local Update Partial Programming File Generation

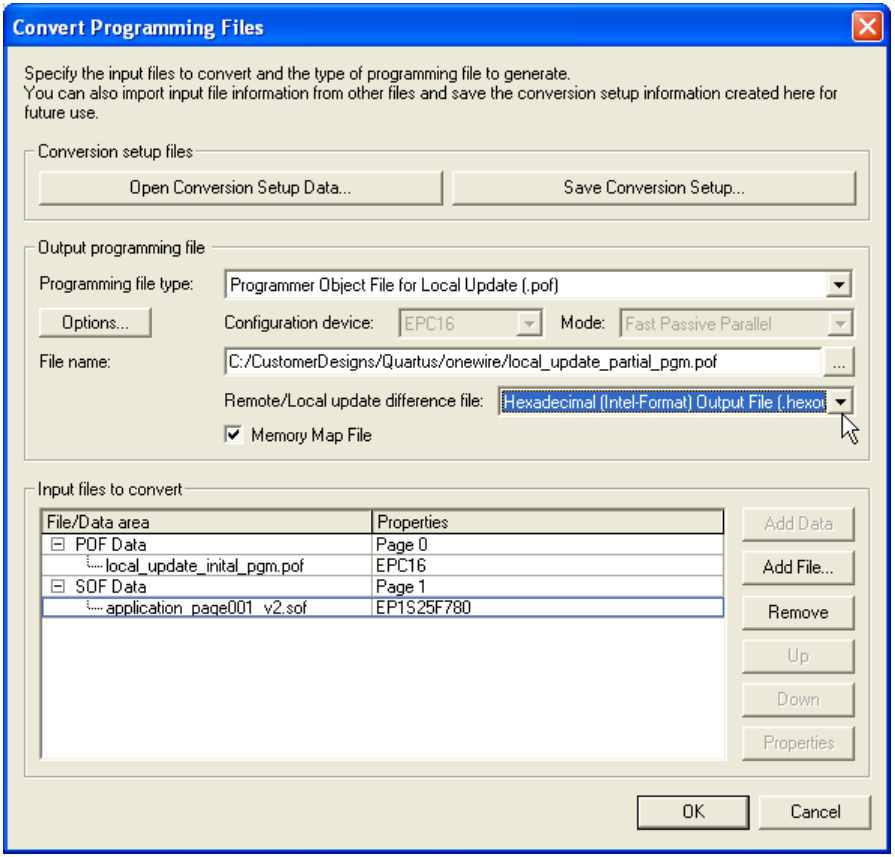
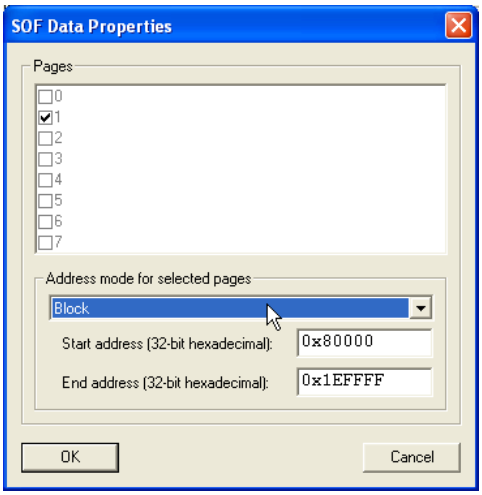


Figure 2–18. Specifying Block Addresses for Application Configuration



Remote Update Programming File Generation

This section describes the programming file generation process for performing remote system upgrades. The Quartus II CPF utility generates the initial and partial programming files for configuration memory within the enhanced configuration devices.

Remote configuration mode uses a factory configuration stored at page 0, and up to seven application configurations stored at pages 1 through 7. The factory configuration cannot be updated after initial production programming. However, the most recent application configuration can be erased and reprogrammed after initial system deployment. Alternatively, a new application configuration can be added provided adequate configuration memory availability.

In remote update mode, you would first create the initial programming file with the factory configuration image and the application configuration(s). Subsequently, you can generate partial programming files to update the most recent application configuration or add a new application configuration. Quartus II CPF can create partial programming files in HEX, JAM, JBC, and POF formats.

In addition to the configuration pages, user data or processor code can also be pre-programmed in the bottom boot and main data areas of the enhanced configuration device memory. The CPF utility accepts a HEX input file for the bottom and main data areas, and includes this data in the

POF output file. However, this is only supported for initial programming file generation. Partial programming file generation for updating user HEX data is not supported, but can be performed using the enhanced configuration device external flash interface.

Initial Programming File Generation

The initial programming file includes configuration data for both factory and application configuration pages. The enhanced configuration device option's bits are always located between byte addresses `0x00010000` and `0x0001003F`. Also, page 0 always starts at `0x00010040` while its end address is dependent on the size of the factory configuration data.

Two memory allocation options exist for application configurations: auto addressing and block addressing. In auto addressing mode, Quartus II packs all application configurations as close together as possible. This maximizes the number of application configurations that can be stored in memory. However, when auto addressing is used you cannot update existing application configurations. Only new application configurations can be added to the memory.

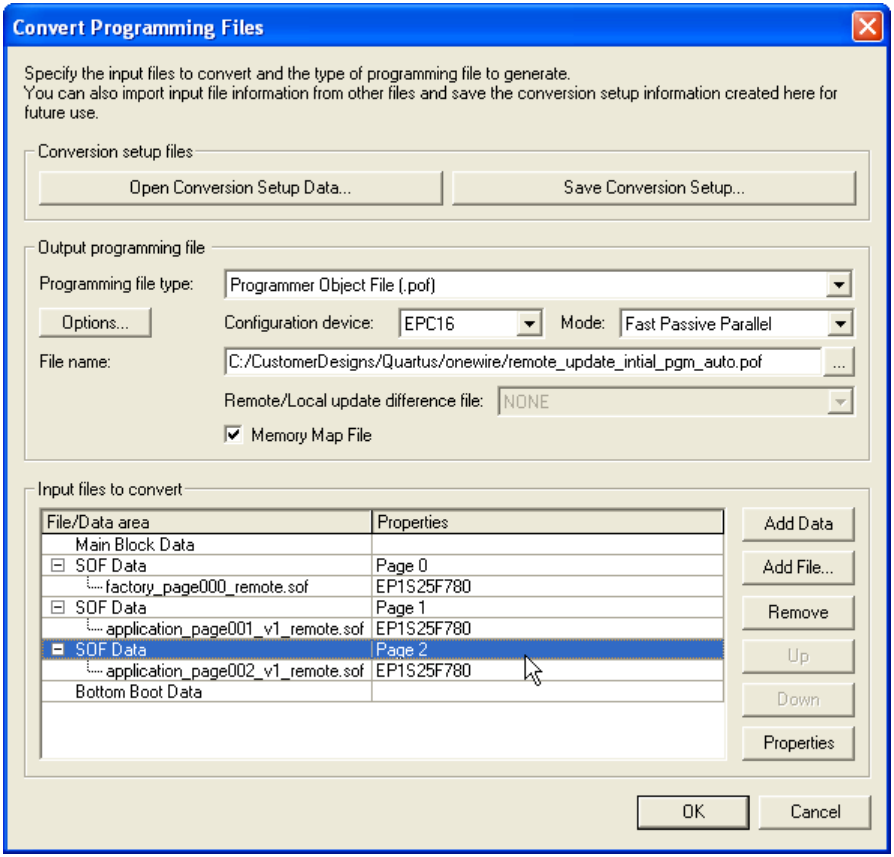
The following steps and screen shot (see Figure 2–19) describe initial programming file generation with auto addressing mode.

1. Open the **Convert Programming Files** window from the **File** menu.
2. Select **Programmer Object File (*.pof)** from the drop-down list titled **Programming file type**.
3. Select the enhanced configuration device used (EPC4, EPC8, EPC16), and the mode used (1-bit Passive Serial or Fast Passive Parallel). Only during the initial programming file generation can you specify the **Options**, **Configuration device**, or **Mode** settings. While generating the partial programming file, all of these settings are grayed out and inaccessible.
4. In the **Input files to convert box**, highlight **SOF Data** at **Page 0** and click **Add File**. Select input SOF file(s) for this configuration page and insert them.
5. Repeat Step 4 for all application configurations (up to 7 maximum).
6. Check the **Memory Map File** box to generate a memory map output file that specifies the start/end addresses of each configuration page and user data blocks.

7. Save the CPF setup (optionally), by selecting **Save Conversion Setup…** and specifying a name for the COF output file.

8. Click **OK** to generate initial programming and memory map files.

Figure 2–19. CPF Setup for Initial Programming File Generation (Auto Addressing)



A sample memory map output file for the preceding setup is shown below. Notice all configuration pages are packed such that two pages can share a flash data sector. This disallows partial programming of application configurations in auto addressing mode.

Block	Start Address	End Address
OPTION BITS	0x00010000	0x0001003F
PAGE 0	0x00010040	0x00054EFA
PAGE 1	0x00054EFC	0x00099DB6
PAGE 2	0x00099DB8	0x000DEC72



See the *Sharp LHF16J06 Data Sheet Flash memory used in EPC16 devices* at **www.altera.com** to correlate memory addresses to the EPC16 flash sectors.

The block addressing mode allows better control of flash memory allocation. You can allocate a specific flash memory region for each application configuration page. This allocation is done by specifying a block starting and block ending address. While selecting the size of the region, you should account for growth in compressed configuration bitstream sizes due to design changes and additions. In remote update mode, all configuration data is top justified within this allotted memory. In other words, the last byte of configuration data is stored such that it coincides with the highest byte address location within the allotted space. Lower unused memory address locations within the allotted region are filled with 1's. These filler bits are transmitted during the application configuration cycle, but are ignored by the Stratix device. The memory map output file provides the exact byte address where real application configuration data for each page begins. Note that any partial update of the most recent application configuration should erase all allotted flash sectors for that page before storing new configuration data.

In the block addressing mode, HEX input files can be optionally added to the bottom boot and main flash data areas (one HEX file per area is allowed). The HEX file can be stored with relative addressing or absolute addressing. For more information on relative and absolute addressing, see the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* chapter of the *Configuration Handbook, Volume 2*.

Figures 2–20 and 2–21, and the following steps illustrate generating an initial programming file with block addressing for remote update mode. This example also illustrates preloading user HEX data into bottom boot and main flash sectors.

1. Open the **Convert Programming Files** window from the **File** menu.

2. Select **Programmer Object File (*.pof)** from the drop-down list titled **Programming file type**.

3. Select the enhanced configuration device used (EPC4, EPC8, EPC16), and the mode used (1-bit Passive Serial or Fast Passive Parallel). Only during the initial programming file generation can you specify the **Options**, **Configuration device**, or **Mode** settings. While generating the partial programming file, all of these settings are grayed out and inaccessible.

4. In the **Input** files to convert box, highlight **SOF Data** at **Page 0** and click **Add File**. Select input SOF file(s) for this configuration page and insert them.

5. Repeat Step 4 for all the application configuration pages (pages 1 and 2 in this example).

6. For enabling block addressing, select the **SOF Data** entry for **Page 1**, and click **Properties**. This opens the **SOF Data Properties** dialog box (see Figure 2–21).

7. Pick **Block** from the **Address Mode** drop down selection, and enter 32-bit Hexadecimal byte address for block **Starting Address** and **Ending Address**. Note that for partial programming support, the block start and end addresses should be aligned to a flash sector boundary. This prevents two configuration pages from overlapping within the same flash boundary. See the flash memory datasheet for data sector boundary information. Click **OK** to save SOF data properties.

8. Check the **Memory Map File** box to generate a memory map output file that specifies the start/end addresses of each configuration page and user data blocks.

9. Save the CPF setup (optionally), by selecting **Save Conversion Setup…** and specifying a name for the COF output file.

10. Click **OK** to generate initial programming and memory map files.

Figure 2–20. CPF Setup for Initial Programming File Generation (Block Addressing)

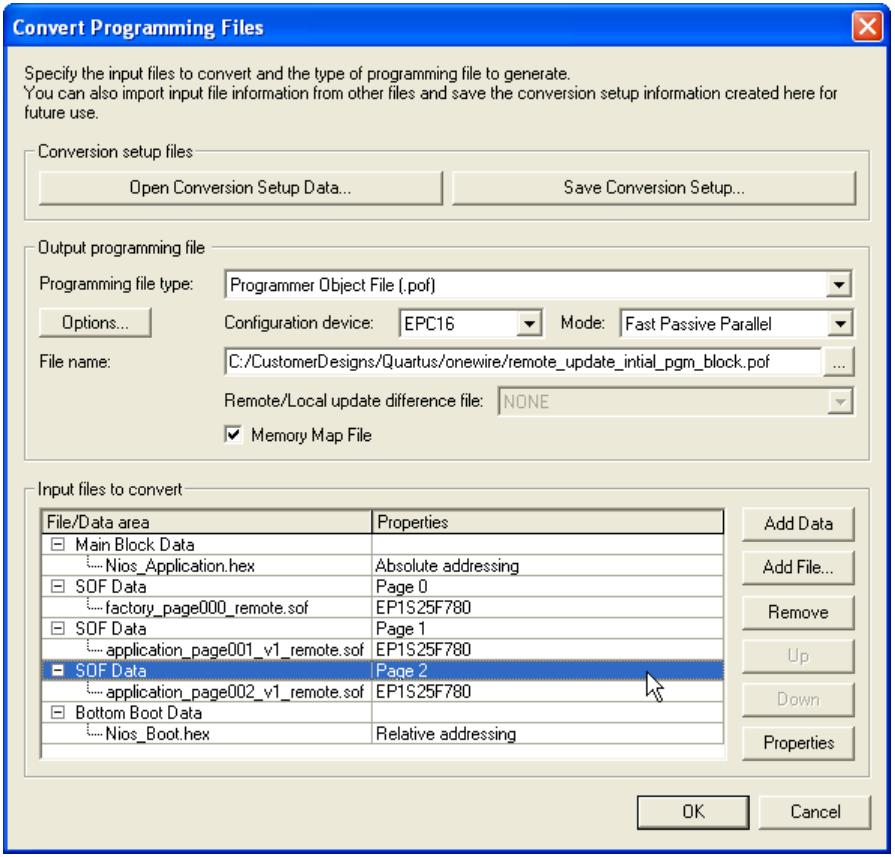
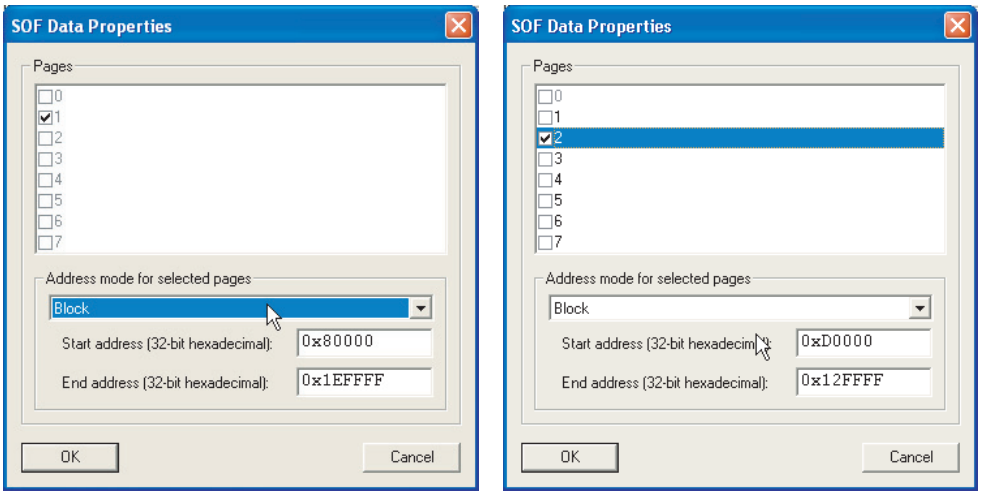


Figure 2–21. Specifying Block Addresses for an Application Configuration



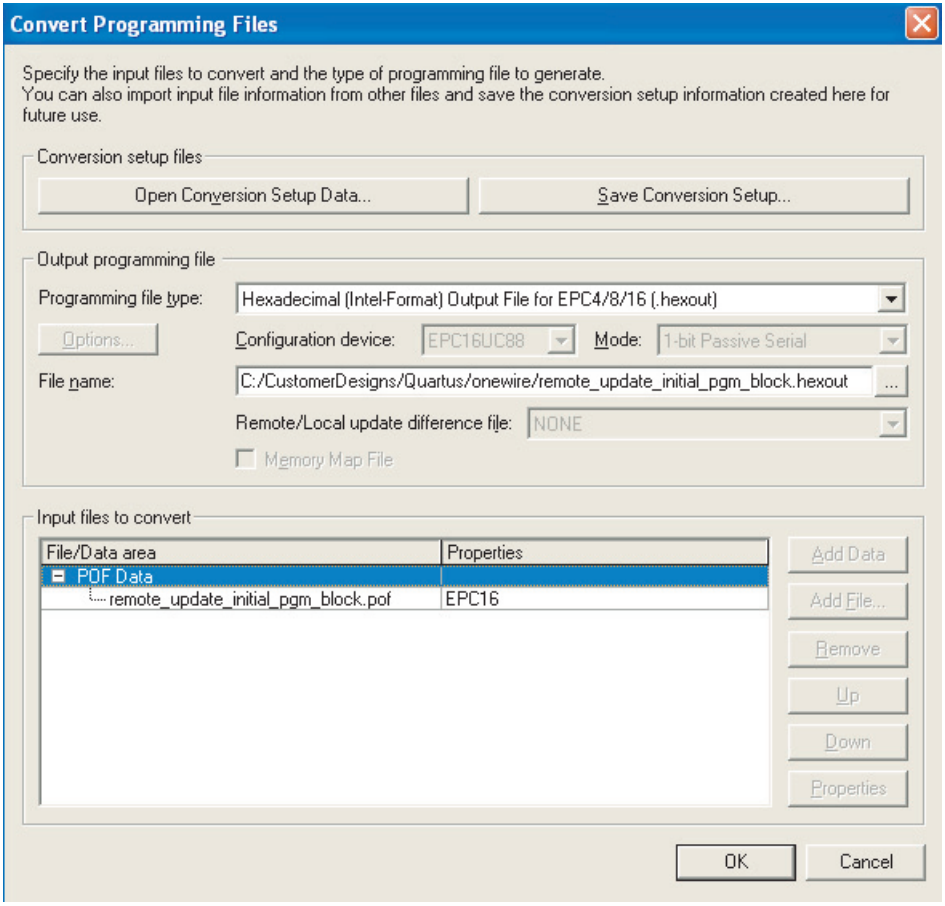
A sample memory map output file for the preceding example is shown below. Note that the allocated memory for page 1 is between `0x00070000` and `0x000BFFFF`, while the actual region used by the current application configuration bitstream is between `0x0007B144` and `0x000BFFFF`. The configuration data is top justified within the allocated SOF data region. Similarly, the allocated memory for page 2 is between `0x000D0000` and `0x0012FFFF`, while the actual region used by the application configuration is between `0x000EB13E` and `0x0012FFF9`.

Block	Start Address	End Address
BOTTOM BOOT	0x00000000	0x000001FF
OPTION BITS	0x00010000	0x0001003F
PAGE 0	0x00010040	0x00054EFA
PAGE 1	0x0007B144	0x000BFFFF
PAGE 2	0x000EB13E	0x0012FFF9
TOP BOOT/MAIN	0x001F0000	0x001F01FF

Also note that the HEX data stored in the main data area uses absolute addressing. If relative addressing were to be used, the main data contents would be justified with the top (higher address locations) of the memory.

The initial POF can be converted to an Intel Hexadecimal format file (*.HEXOUT) using the Quartus II CPF utility. See Figure 2–22.

Figure 2–22. Converting POF Programming File to Intel HEX Format



Partial Programming File Generation

In remote update mode, the Quartus II CPF utility allows an existing application configuration page to be replaced with new data, or a new application configuration to be added. Partial programming files are generated to perform such configuration data updates.

In order to generate a partial programming file, you have to input the initial POF and new configuration data (SOF) to the Quartus II CPF utility. In addition, you have to specify the addressing mode (auto or manual) that was used during initial POF creation. And if block addressing was used, you should specify the block start and end

addresses. With this information, Quartus II ensures that the partial POF only updates the flash region containing the application configuration. The factory configuration (page 0) and configuration option bits are left unaltered during this process. The only exception is when a new application configuration is added, the configuration options bits are updated to include start/end addresses for the new page. All existing page addresses and other configuration options bits remain unchanged.

Figure 2–23 and the following steps illustrate generation of a partial programming file to replace the most recent application configuration. In this example, the initial programming file contained one factory and two application configurations. Hence, the page 2 application configuration is being updated with new data.

1. Open the **Convert Programming Files** window from the **File** menu.
2. Select **Programmer Object File for Remote Update (*.pof)** from the drop-down list titled **Programming file type**, and specify an output file name.
3. In the **Input files to convert** box, highlight **POF Data** and click **Add File**. Select the initial programming POF file for this design and insert it.
4. In the **Input files to convert box**, highlight **SOF Data** and click **Add File**. Select the new application configuration bitstream (SOF) and insert it.
5. When using block addressing, select the **SOF Data** entry for **Page 2**, and click **Properties**. This opens the **SOF Data Properties** dialog box (see Figure 2–24 on page 2–42).
6. Pick **Block** from the **Address Mode** drop down selection, and enter 32-bit Hexadecimal byte address for block **Starting Address** and **Ending Address**. These addresses should be identical to those used to generate the initial programming file. Click **OK** to save SOF data properties.
7. Check the **Memory Map File** box to generate a memory map output file that specifies the start/end addresses of the new application configuration data in page 1.
8. Pick a remote update difference file from the **Remote/Local Update Difference File** drop-down menu. You can select between an Intel HEX, JAM, JBC, and POF output file types. The output file name is the same as the POF output file name with a _dif suffix.

9. Save the CPF setup (optionally), by selecting **Save Conversion Setup…** and specifying a name for the COF output file.

10. Click **OK** to generate initial programming and memory map files.

Figure 2–23. Remote Update Partial Programming File Generation

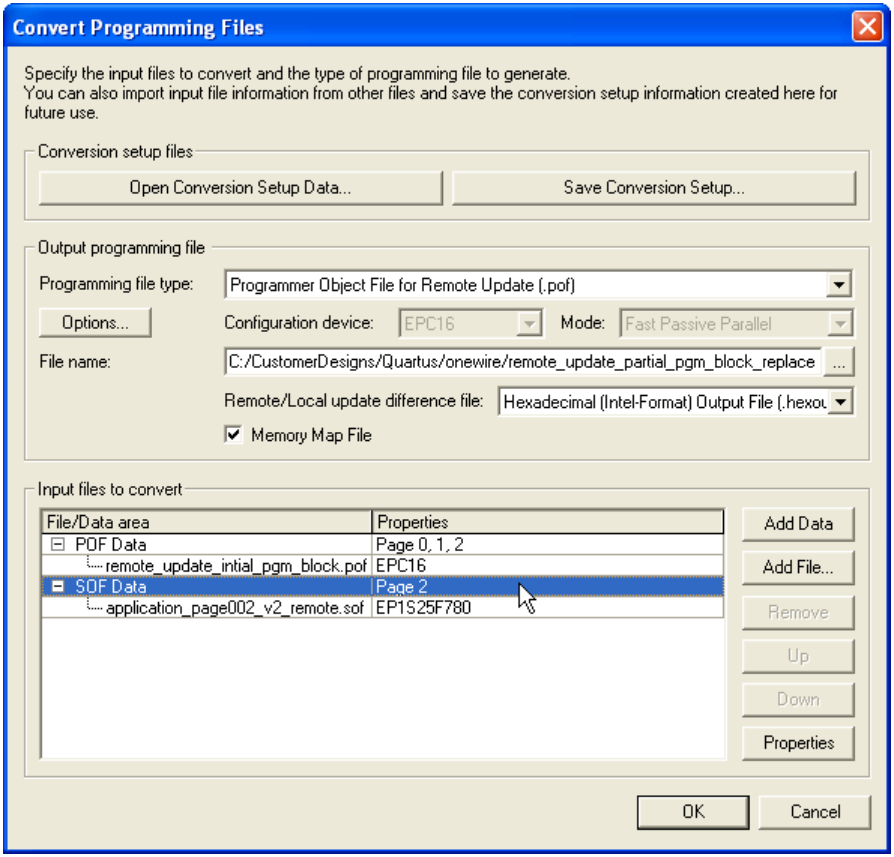
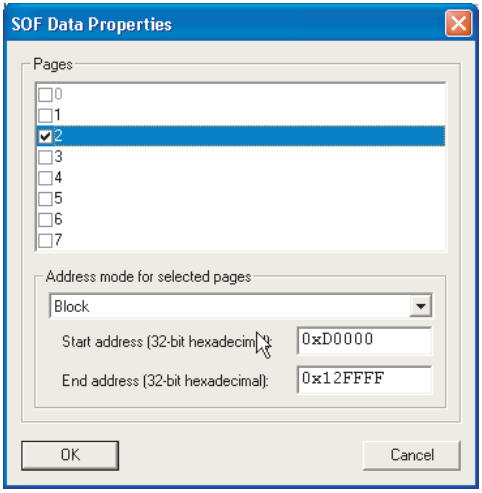


Figure 2–24. Specifying Block Addresses for Application Configuration



For adding a new application configuration, follow the steps listed above with one modification. In Step 5, select **SOF Data** and click on **Properties**. In the **SOF Data Properties** dialog box, select a new page (for example, page 3) and specify the addressing mode information. Continue with steps 7 through 10. When a new page is added, the memory map output file lists the start/end addresses for this page. A sample is shown below:

Block	Start Address	End Address
OPTION BITS	0x00010000	0x0001003F
PAGE 3	0x0012FFFA	0x00174EB4

Combining MAX Devices & Flash Memory

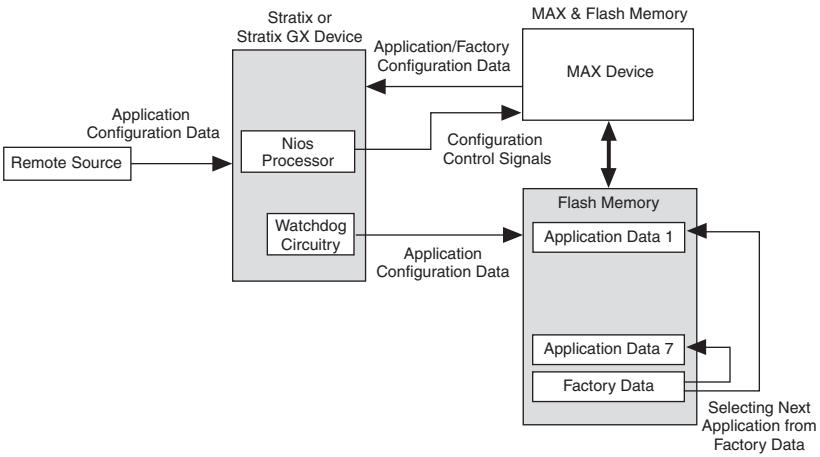
This section describes remote system configuration with the Stratix or Stratix GX device and the Nios embedded processor, using a combination of MAX® devices and flash memory.

You can use MAX 3000 or MAX 7000 devices and an industry-standard flash memory device instead of enhanced configuration devices. In this scheme, flash memory stores configuration data, and the MAX device controls reading and writing to the flash memory, keeping track of address locations.

The MAX device determines which address location and at what length to store configuration data in flash memory. The Nios embedded processor, running in the Stratix or Stratix GX device, receives the

incoming data from the remote source and writes it to the address location in flash memory. The Nios embedded processor initiates loading of factory configuration into the Stratix or Stratix GX device. Figure 2–25 shows remote system configuration using a MAX device and flash memory combination.

Figure 2–25. Remote System Configuration Using a MAX Device & Flash Memory



You can use both remote and local configuration modes in this scheme. You should specify a default page for factory configuration and make sure it is not altered or removed at any time. In remote system configuration mode, PS, FPP, and PPA modes are supported when configuring with MAX and flash devices.

Using an External Processor

This section describes remote system configuration with Stratix or Stratix GX devices and the Nios embedded processor, using an external processor and flash memory devices.

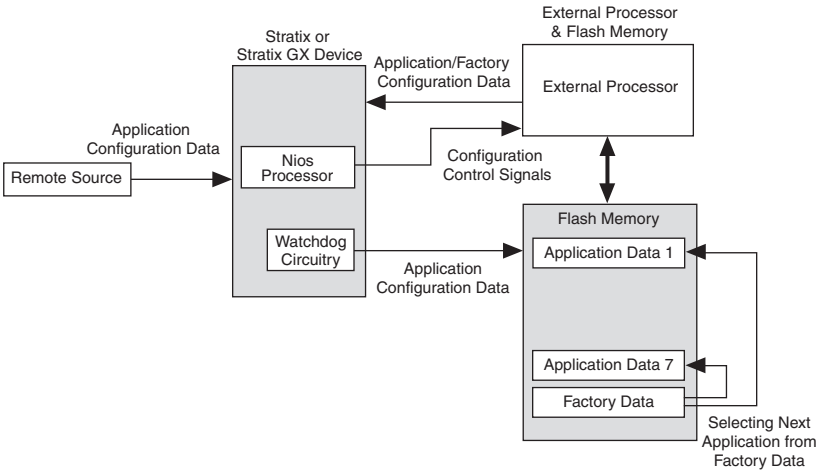
In this scheme, the external processor and flash memory device replace the enhanced configuration device. Flash memory stores configuration data, and the processor controls reading and writing to the flash memory and also keeps track of the address location. This type of remote system configuration supports PS, FPP, and PPA modes.

The processor determines at which address which length to store the configuration data in flash memory. The Nios embedded processor receives the incoming data from a remote source and writes it to the address location in the flash memory, and then initiates loading of factory

configuration data into the Stratix or Stratix GX device. Figure 2–26 shows the remote system configuration using a Nios embedded processor and flash memory.

You can use both remote and local configuration modes in this scheme. You should specify a default page for factory configuration and make sure it is not altered or removed at any time.

Figure 2–26. Remote System Configuration Using External Processor & Flash Memory



Conclusion

Stratix and Stratix GX devices are the first PLDs with dedicated support for remote system configuration. By allowing real-time system upgrades from a remote source, you can use Stratix and Stratix GX devices in a variety of applications that require automatic configuration updates. With the built-in watchdog timer circuitry, Stratix and Stratix GX devices avoid incorrect or erroneous states. Using Stratix and Stratix GX devices with remote system configuration enhances design flexibility and reduces time to market.

Section II. Design Guidelines

This section provides information on design transition, board design guidelines, and Stratix GX device path delay issues.

This section includes the following chapter:

- Chapter 3, Transitioning APEX Designs to Stratix & Stratix GX Devices
- Chapter 4, Stratix GX Board Design Guidelines
- Chapter 5, Quartus II Software Fitter Warnings

Revision History

The table below shows the revision history for Chapters 3 through 5.

Chapter(s)	Date / Version	Changes Made
3	February 2005 v3.0	Added chapter to *Stratix GX Device Handbook*.
4	February 2005 v1.0	Added chapter to *Stratix GX Device Handbook*.
5	March 2005 v1.0	Added chapter to *Stratix GX Device Handbook*.

3. Transitioning APEX Designs to Stratix & Stratix GX Devices

S52012-3.0

Introduction

Stratix® and Stratix GX devices are Altera's next-generation, system-on-a-programmable-chip (SOPC) solution. Stratix and Stratix GX devices simplify the block-based design methodology and bridge the gap between system bandwidth requirements and programmable logic performance.

This chapter highlights the new features in the Stratix and Stratix GX devices and provides assistance when transitioning designs from APEX™ II or APEX 20K devices to the Stratix or Stratix GX architecture. You should be familiar with the APEX II or APEX 20K architecture and available device features before using this chapter. Use this chapter in conjunction with the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* or the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1*.

General Architecture

Stratix and Stratix GX devices offer many new features and architectural enhancements. Enhanced logic elements (LEs) and the MultiTrack™ interconnect structure offer reduced resource utilization and considerable design performance improvement. The MultiTrack interconnect uses DirectDrive™ technology to ensure the availability of deterministic routing resources for any design block, regardless of its placement within the device.

All architectural changes between Stratix and Stratix GX and APEX II or APEX 20K devices described in this section do not require any design changes. However, you must resynthesize your design and recompile in the Quartus® II software to target Stratix and Stratix GX devices.

Logic Elements

Stratix and Stratix GX device LEs include several new, advanced features that improve design performance and reduce logic resource consumption (see Table 3–1). The Quartus II software automatically uses these new LE features to improve device utilization.

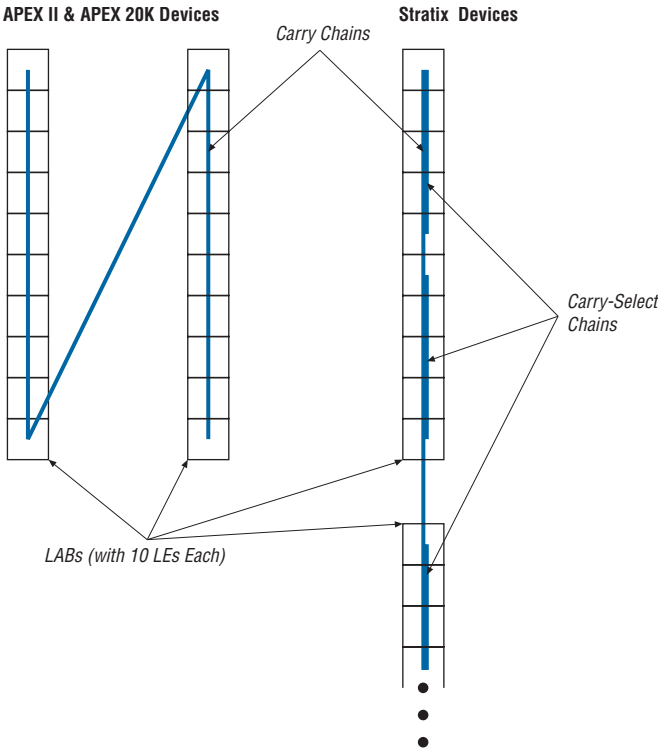
Table 3–1. Stratix & Stratix GX LE Features

Feature	Function	Benefit
Register chain interconnects	Direct path between the register output of an LE and the register input of an adjacent LE within the same logic array block (LAB)	■ Conserves LE resources ■ Provides fast shift register implementation ■ Saves local interconnect routing resources within an LAB
Look-up table (LUT) chain interconnects	Direct path between the combinatorial output of an LE and the fast LUT input of an adjacent LE within the same LAB	■ Allows LUTs within the same LAB to cascade together for high-speed wide fan-in functions, such as wide XOR operations ■ Bypasses local interconnect for faster performance
Register-to-LUT feedback path	Allows the register output to feed back into the LUT of the same LE, such that the register is packed with its own fan-out LUT	■ Enhanced register packing mode ■ Uses resources more efficiently
Dynamic arithmetic mode	Uses one set of LEs for implementing both an adder and subtractor	■ Improves performance for functions that switch between addition and subtraction frequently, such as correlators
Carry-select chain	Calculates outputs for a possible carry-in of 1 or 0 in parallel	■ Gives immediate access to result for both a carry-in of 1 or 0 ■ Increases speed of carry functions for high-speed operations, such as counters, adders, and comparators
Asynchronous clear and asynchronous preset function	Supports direct asynchronous clear and preset functions	■ Conserves LE resources ■ Does not require additional logic resources to implement NOT-gate push-back

In addition to the new LE features described in Table 3–1, there are enhancements to the chains that connect LEs together. Carry chains are implemented vertically in Stratix and Stratix GX devices, instead of horizontally as in APEX II and APEX 20K devices, and continue across rows, instead of across columns, as shown in Figure 3–1. Also note that the Stratix and Stratix GX architectures do not support the cascade primitive. Therefore, the Quartus II Compiler automatically converts

cascade primitives in APEX II and APEX 20K designs to a wire primitive when compiled for Stratix and Stratix GX devices. These architectural changes are transparent to the user and do not require design changes.

Figure 3–1. Carry Chain Implementation in APEX II & APEX 20K Devices vs. Stratix & Stratix GX Devices



MultiTrack Interconnect

Stratix and Stratix GX devices use the MultiTrack interconnect structure to provide a high-speed connection between logic resources using performance-optimized routing channels of different lengths. This feature maximizes overall design performance by placing critical paths on routing lines with greater speed, resulting in minimal propagation delay.

Stratix and Stratix GX device MultiTrack interconnect resources are described in Table 3–2.

Table 3–2. Stratix & Stratix GX Device MultiTrack Interconnect Resources

Routing Type	Interconnect	Span
Row	Direct link	Adjacent LABs and/or blocks
Row	R4	Four LAB units horizontally
Row	R8	Eight LAB units horizontally
Row	R24	Horizontal routing across the width of the device
Column	C4	Four LAB units vertically
Column	C8	Eight LAB units vertically
Column	C16	Vertical routing across the length of the device

Direct link routing saves row routing resources while providing fast communication paths between resource blocks. Direct link interconnects allow an LAB, digital signal processing (DSP) block, or TriMatrix™ memory block to drive data into the local interconnect of its left and right neighbors. LABs, DSP blocks, and TriMatrix memory blocks can also use direct link interconnects to drive data back into themselves for feedback.

The Quartus II software automatically uses these routing resources to enhance design performance.



For more information about LE architecture and the MultiTrack interconnect structure in Stratix and Stratix GX devices, see the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* or the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1*.

DirectDrive Technology

When using APEX II or APE 20K devices, you must place critical paths in the same MegaLAB™ column to improve performance. Additionally, you should place critical paths in the same MegaLAB structure for optimal performance. However, this restriction does not exist in Stratix and Stratix GX devices because they do not contain MegaLAB structures. With the new DirectDrive™ technology in Stratix and Stratix GX devices, the actual distance between the source and destination of a path is the most important criteria for meeting timing performance. DirectDrive technology ensures that the same routing resources are available to each design block, regardless of its location in the device.

Architectural Element Names

The architectural element naming system within Stratix and Stratix GX devices differs from the row-column coordinate system (for example, LC1_A2, LAB_B1) used in previous Altera device families. Stratix and Stratix GX devices uses a new naming system based on the X-Y coordinate system, (X, Y). A number (N) designates the location within the block where the logic resides, such as LEs within an LAB. Because the Stratix and Stratix GX architectures are column-based, this naming simplifies location assignments. Stratix and Stratix GX architectural blocks include:

- LAB: logic array block
- DSP: digital signal processing block
- DSPOUT: adder/subtractor/accumulator or summation block of the DSP block
- M512: 512-bit memory block
- M4K: 4-Kbit memory block
- M-RAM: 512-Kbit memory block

Elements within architectural blocks include:

- LE: logic element
- IOC: I/O element
- PLL: phase-locked loop
- DSPMULT: DSP block multiplier
- SERDESTX: transmitter serializer/deserializer
- SERDESRX: receiver serializer/deserializer

Table 3–3 highlights the new location syntax used for Stratix and Stratix GX devices.

Table 3–3. Stratix & Stratix GX Location Assignment Syntax

Architectural Elements	Element Name	Location Syntax	Example of Location Syntax	
			Location	Description
Blocks	LAB, DSP, DSPOUT, M512, M4K, M-RAM	<*element_name*>_X<*number*>_Y<*number*>	LAB_X1_Y1	Designates the LAB in row 1, column 1
Logic	LE, IOC, PLL, DSPMULT, SERDESTX, SERDESRX	<*element_name*>_X<*number*>_Y<*number*>_N<*number*>	LC_X1_Y1_N0	Designates the first LE, N0, in the LAB located in row 1, column 1
Pins *(1)*	I/O pins	pin_<*pin_label*>	pin_5	Pin 5

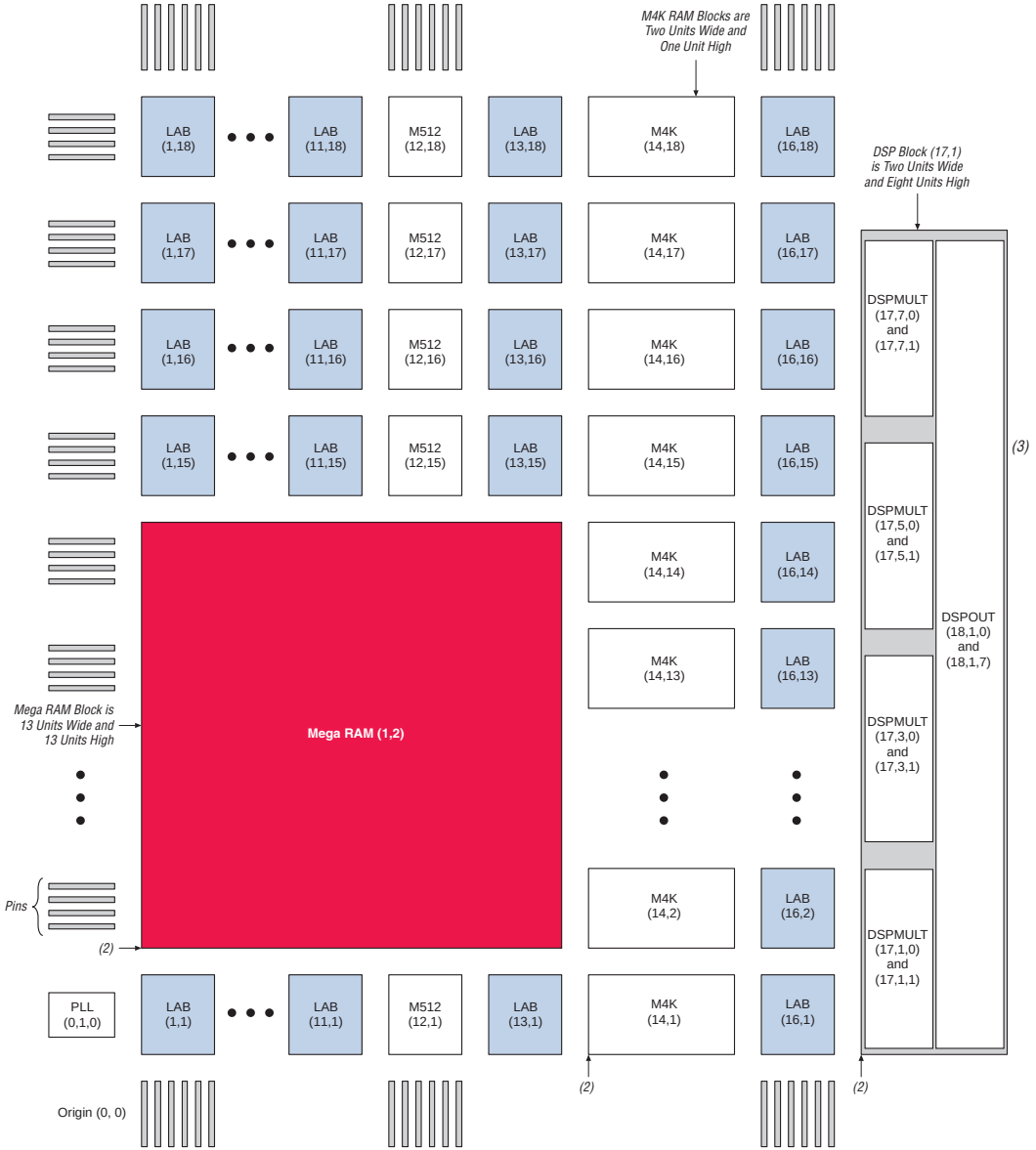
Note to Table 3–3:
(1) You can make assignments to I/O pads using IOC_X<*number*>_Y<*number*>_N<*number*>.

Use the following guidelines with the new naming system:

- The anchor point, or origin, in Stratix and Stratix GX devices is in the bottom-left corner, instead of the top-left corner as in APEX II and APEX 20K devices.
- The anchor point, or origin, of a large block element (e.g., a M-RAM or DSP block) is also the bottom-left corner.
- All numbers are zero-based, meaning the origin at the bottom-left of the device is X0, Y0.
- The I/O pins constitute the first and last rows and columns in the X-Y coordinates. Therefore, the bottom row of pins resides in X<*number*>, Y0, and the first left column of pins resides in X0, Y<*number*>.
- The sub-location of elements, N, numbering begins at the top. Therefore, the LEs in an LAB are still numbered from top to bottom, but start at zero.

Figure 3–2 show the Stratix and Stratix GX architectural element numbering convention. Figure 3–3 displays the floorplan view in the Quartus II software.

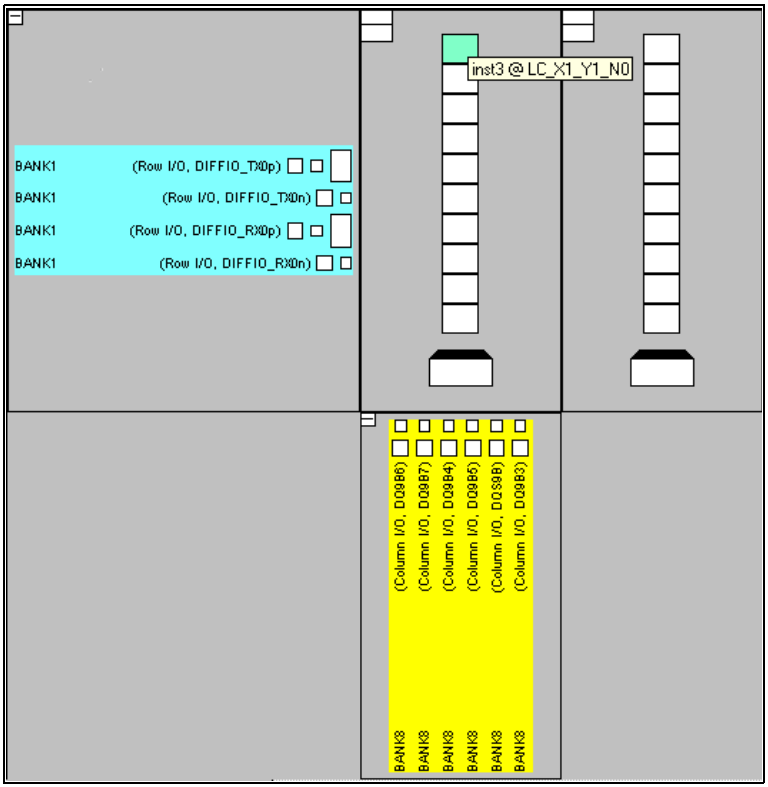
Figure 3–2. Stratix & Stratix GX Architectural Elements Note (1)



Notes to Figure 3–2:

(1) Figure 3–2 shows part of a Stratix and Stratix GX device.
(2) Large block elements use their lower-left corner for the coordinate location.
(3) The Stratix GX architectural elements include transceiver blocks on the right side of the device.

Figure 3–3. LE Numbering as Shown in the Quartus II Software



TriMatrix Memory

TriMatrix memory has three different sizes of memory blocks, each optimized for a different purpose or application. M512 blocks consist of 512 bits plus parity (576 bits), M4K blocks consist of 4K bits plus parity (4,608 bits), and M-RAM blocks consist of 512K bits plus parity (589,824 bits). This new structure differs from APEX II and APEX 20K devices, which feature uniformly sized embedded system blocks (ESBs) either 4 Kbits (APEX II devices) or 2 Kbits (APEX 20K devices) large. Stratix and Stratix GX TriMatrix memory blocks give you advanced control of each memory block, with features such as byte enables, parity bit storage, and shift-register mode, as well as mixed-port width support and true dual-port mode operation.

Table 3–4 compares TriMatrix memory with ESBs.

Table 3–4. Stratix & Stratix GX TriMatrix Memory Blocks vs. APEX II & APEX 20K ESBs

Features	Stratix & Stratix GX			APEX II ESB	APEX 20K ESB
	M512 RAM	M4K RAM	M-RAM		
Size (bits)	576	4,608	589,824	4,096	2,048
Parity bits	Yes	Yes	Yes	No	No
Byte enable	No	Yes	Yes	No	No
True dual-port mode	No	Yes Includes support for mixed width	Yes Includes support for mixed width	Yes Includes support for mixed width	No
Embedded shift register	Yes	Yes	No	No	No
Dedicated content-addressable memory (CAM) support	No	No	No	Yes	Yes
Pre-loadable initialization with a **.mif** *(1)*	Yes	Yes	No	Yes	Yes
Packed mode *(2)*	No	Yes	No	Yes	Yes
Feed-through behavior	Rising edge	Rising edge	Rising edge	Falling edge	Falling edge
Output power-up condition	Powers up cleared even if using a **.mif** *(1)*	Powers up cleared even if using a **.mif** *(1)*	Powers up with unknown state	Powers up cleared or to initialized value, if using a **.mif** *(1)*	Powers up cleared or to initialized value, if using a **.mif** *(1)*

Notes to Table 3–4:
(1) **.mif**: Memory Initialization File.
(2) Packed mode refers to combining two single-port RAM blocks into a single RAM block that is placed into true dual-port mode.

Stratix and Stratix GX TriMatrix memory blocks only support pipelined mode, while APEX II and APEX 20K ESBs support both pipelined and flow-through modes. Since all TriMatrix memory blocks can be pipelined, all input data and address lines are registered, while outputs can be either registered or combinatorial. You can use Stratix and Stratix GX memory block registers to implement input and output registers without utilizing additional resources. You can compile designs containing pipelined memory blocks (inputs registered) for Stratix and Stratix GX devices without any modifications. However, if an APEX II or

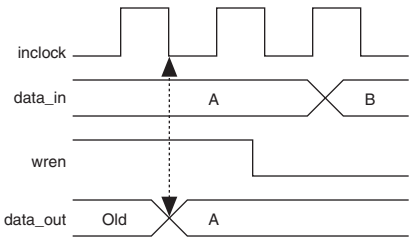
APEX 20K design contains flow-through memory, you must modify the memory modules to target the Stratix and Stratix GX architectures (see "Memory Megafunctions" on page 3–12 for more information).

For more information about TriMatrix memory and converting flow-through memory modules to pipelined, see the *TriMatrix Embedded Memory Blocks in Stratix & Stratix GX Devices* chapter in the *Stratix GX Device Handbook* and *AN 210: Converting Memory from Asynchronous to Synchronous for Stratix & Stratix GX Designs*.

Same-Port Read-During-Write Mode

In same-port read-during-write mode, the RAM block can be in single-port, simple dual-port, or true dual-port mode. One port from the RAM block both reads and writes to the same address location using the same clock. When APEX II or APEX 20K devices perform a same-port read-during-write operation, the new data is available on the falling edge of the clock cycle on which it was written, as shown in Figure 3–4. When Stratix and Stratix GX devices perform a same-port read-during-write operation, the new data is available on the rising edge of the same clock cycle on which it was written, as shown in Figure 3–5. This holds true for all TriMatrix memory blocks.

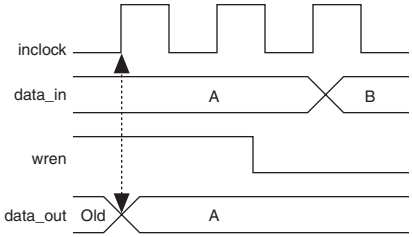
Figure 3–4. Falling Edge Feed-Through Behavior (APEX II & APEX 20K Devices) *Note (1)*



Note to Figure 3–4:

(1) Figures 3–4 and 3–5 assume that the address stays constant throughout and that the outputs are not registered.

Figure 3–5. Rising Edge Feed-Through Behavior (Stratix & Stratix GX Devices) Note (1)

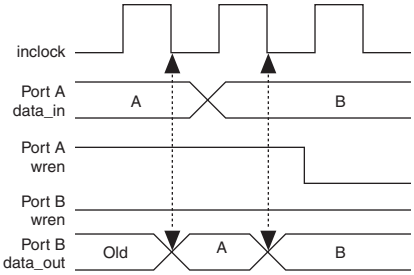


Note to Figure 3–5:
(1) Figures 3–4 and 3–5 assume that the address stays constant throughout and that the outputs are not registered.

Mixed-Port Read-During-Write Mode

Mixed-port read-during-write mode occurs when a RAM block in simple or true dual-port mode has one port reading and the other port writing to the same address location using the same clock. In APEX II and APEX 20K designs, the ESB outputs the old data in the first half of the clock cycle and the new data in the second half of the clock cycle, as indicated by Figure 3–6.

Figure 3–6. Mixed-Port Feed-Through Behavior (APEX II & APEX 20K Devices) Note (1)

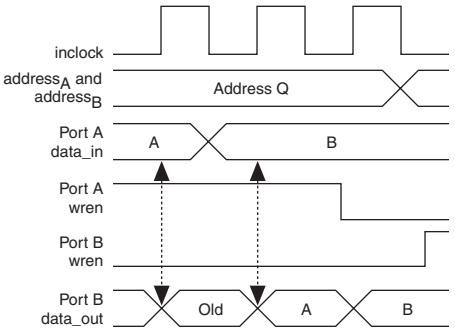


Note to Figure 3–6:
(1) Figure 3–6 assumes that outputs are not registered.

Stratix and Stratix GX device RAM outputs the new data on the rising edge of the clock cycle immediately after the data was written. When you use Stratix and Stratix GX M512 and M4K blocks, you can choose whether to output the old data at the targeted address or output a don't care value during the clock cycle when the new data is written. M-RAM blocks

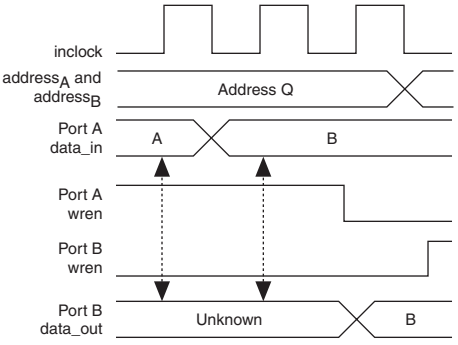
always output a don't care value. Figures 3–7 and 3–8 show the feed-through behavior of the mixed-port mode. You can use the `altsyncram` megafunction to set the output behavior during mixed-port read-during-write mode.

Figure 3–7. Mixed-Port Feed-Through Behavior (OLD_DATA) *Note (1)*



Note to Figure 3–7:
(1) Figures 3–7 and 3–8 assume that the address stays constant throughout and that the outputs are not registered.

Figure 3–8. Mixed-Port Feed-Through Behavior (DONT_CARE) *Note (1)*



Note to Figure 3–8:
(1) Figures 3–7 and 3–8 assume that the address stays constant throughout and that the outputs are not registered.

Memory Megafunctions

To convert RAM and ROM originally targeting the APEX II or APEX 20K architecture to Stratix or Stratix GX memory, specify Stratix or Stratix GX as the target family in the MegaWizard Plug-In Manager. The software

updates the memory module for the Stratix or Stratix GX architecture and instantiates the new synchronous memory megafunction, altsyncram, which supports both RAM and ROM blocks in the Stratix and Stratix GX architectures.

FIFO Conditions

First-in first-out (FIFO) functionality is slightly different in Stratix and Stratix GX devices compared to APEX II and APEX 20K devices. Stratix and Stratix GX devices do not support simultaneous reads and writes from an empty FIFO buffer. Also, Stratix and Stratix GX devices do not support the lpm_showahead parameter when targeting a FIFO buffer because the TriMatrix memory blocks are synchronous. The lpm_showahead parameter for APEX II and APEX 20K devices puts the FIFO buffer in "read-acknowledge" mode so the first data written into the FIFO buffer immediately flows through to the output. Other than these two differences, all APEX II and APEX 20K FIFO functions are fully compatible with the Stratix and Stratix GX architectures.

Design Migration Mode in Quartus II Software

The Quartus II software features a migration mode for simplifying the process of converting APEX II and APEX 20K memory functions to the Stratix or Stratix GX architecture. If the design can use the Stratix or Stratix GX altsyncram megafunction as a replacement for a previous APEX II or APEX 20K memory function while maintaining functionally similar behavior, the Quartus II software automatically converts the memory. The software produces a warning message during compilation reminding you to verify that the design migrated correctly.

For memory blocks with all inputs registered, the existing megafunction is converted to the new altsyncram megafunction. The software generates a warning when the altsyncram megafunction is incompatible. For example, a RAM block with all inputs registered except the read enable compiles with a warning message indicating that the read-enable port is registered.

You can suppress warning messages for the entire project or for individual memory blocks by setting the SUPPRESS_MEMORY_CONVERSION_WARNINGS parameter to "on" as a global parameter by selecting **Assignment Organizer** (Tools menu). In the **Assignment Organizer** window, click **Parameters** in the **Assignment Categories** box. Type SUPPRESS_MEMORY_CONVERSION_WARNINGS in the **Assignment Name** box and type ON in the **Assignment Setting** box. To suppress these warning messages on a per-memory-instance basis, set the SUPPRESS_MEMORY_CONVERSION_WARNINGS parameter in the Assignment Organizer to "on" for the memory instance.

If the functionality of the APEX II or APEX 20K memory megafunction differs from the `altsyncram` functionality and at least one clock feeds the memory megafunction, the Quartus II software converts the APEX II or APEX 20K memory megafunction to the Stratix or Stratix GX `altsyncram` megafunction. This conversion is useful for an initial evaluation of how a design might perform in Stratix or Stratix GX devices and should only be used for evaluation purposes. During this process, the Quartus II software generates a warning that the conversion may be functionally incorrect and timing results may not be accurate. Since the functionality may be incorrect and the compilation is only intended for comparative purposes, the Quartus II software does not generate a programming file. A functionally correct conversion requires manually instantiating the `altsyncram` megafunction and may require additional design changes.

If the previous memory function does not have a clock (fully asynchronous), the fitting-evaluation conversion results in an error message during compilation and does not successfully convert the design.



See *AN 210: Converting Memory from Asynchronous to Synchronous for Stratix & Stratix GX Designs* for more information.

Table 3–5 summarizes the possible scenarios when using design migration mode and the resulting behavior of the Quartus II software.

The most common cases where design-migration mode may have difficulty converting the existing design are when:

- A port is reading from an address that is being written to by another port (mixed-port read-during-write mode). If both ports are using the same clock, the read port in Stratix and Stratix GX devices do not see the new data until the next clock cycle, after the new data was written.

- There are differences in power-up behavior between APEX II, APEX 20K, and Stratix and Stratix GX devices. You should manually account for these differences to maintain desired operation of the system.

Table 3–5. Migration Mode Summary

Memory Configuration	Conditions	Possible Instantiated Megafunctions	Quartus II Warning Message(s)	Programming File Generated

Single-port	All inputs are registered.	`altram`
		`altrom`
		`lpm_ram_dq`
		`lpm_ram_io`
		`lpm_rom`
			Power-up differences. *(1)*	Yes

Multi-port (two-, three-, or four-port functions)	All inputs are registered.	`altdpram`
		`lpm_ram_dp`
		`altqpram`
		`alt3pram`
			Power-up differences.
			Mixed-port read- during-write. *(1)*
				Yes

Dual-port	Read-enable ports are unregistered.
	Other inputs registered.
		`altdpram`
		`lpm_ram_dp`
		`altqpram`
		`alt3pram`
			Power-up differences.
			Mixed-port read- during-write.
			Read enable will be registered. *(1)*
				Yes

Dual-port	Any other unregistered port except read-enable ports.
	Clock available.
		`altdpram`
		`lpm_ram_dp`
		`altqpram`
		`alt3pram`
			Compile for fitting- evaluation purposes.	No

Single-port	At least one registered input.
	Clock available.
		`altram`
		`lpm_ram_dq`
		`lpm_ram_io`
			Compile for fitting- evaluation purposes.	No

No clock	No clock.	`altram`
		`altrom`
		`altdpram`
		`altqpram`
		`alt3pram`
		`altdpram`
		`lpm_ram_dq`
		`lpm_ram_io`
		`lpm_rom`
		`lpm_ram_dp`
		`lpm_ram_dp`
			Error – no conversion possible.	No

Note to Table 3–5:

(1) If the `SUPPRESS_MEMORY_COUNVERSION_WARNINGS` parameter is turned on, the Quartus II software does not issue these warnings.

DSP Block

Stratix and Stratix GX device DSP blocks outperform LE-based implementations for common DSP functions. Each DSP block contains several multipliers that can be configured for widths of 9, 18, or 36 bits. Depending on the mode of operation, these multipliers can optionally feed an adder/subtractor/accumulator or summation unit.

You can configure the DSP block's input registers to efficiently implement shift registers for serial input sharing, eliminating the need for external shift registers in LEs. You can add pipeline registers to the DSP block for accelerated operation. Registers are available at the input and output of the multiplier, and at the output of the adder/subtractor/accumulator or summation block.

DSP blocks have four modes of operation:

- Simple multiplier mode
- Multiply-accumulator mode
- Two-multipliers adder mode
- Four-multipliers adder mode

Associated megafunctions are available in the Quartus II software to implement each mode of the DSP block.

DSP Block Megafunctions

You can use the `lpm_mult` megafunction to configure the DSP block for simple multiplier mode. You can set the `lpm_mult` **Multiplier Implementation** option in the MegaWizard Plug-In Manager to either use the default implementation, ESBs, or the DSP blocks. If you select the **Use Default** option, the compiler first attempts to place the multiplier in the DSP blocks. However, under certain conditions, the compiler may implement the multiplier in LEs. The placement depends on factors such as DSP block resource consumption, the width of the multiplier, whether an operand is a constant, and other options chosen for the megafunction.

Stratix and Stratix GX devices do not support the **Use ESBs** option. If you select this option, the Quartus II software tries to place the multiplier in unused DSP blocks.

You can recompile APEX II or APEX 20K designs using the `lpm_mult` megafunction for Stratix and Stratix GX devices in the Quartus II software without changing the megafunction. This makes converting `lpm_mult` megafunction designs to Stratix or Stratix GX devices straightforward.

APEX II and APEX 20K designs use pipeline stages to improve registered performance of LE-based multipliers at the expense of latency. However, you may not need to use pipeline stages when targeting Stratix and Stratix GX high-speed DSP blocks. The DSP blocks offer three sets of dedicated pipeline registers. Therefore, Altera recommends that you reduce the number of pipeline stages to three or fewer and implement them in the DSP blocks. Additional pipeline stages are implemented in LEs, which add latency without providing any performance benefit.

For example, you can configure a DSP block for 36 × 36-bit multiplication using the `lpm_mult` megafunction. If you specify two pipeline stages, the software uses the DSP block input and pipeline registers. If you specify three pipeline stages, the software places the third pipeline stage in the DSP block output registers. This design yields the same performance with three pipeline stages because the critical path for a 36 × 36-bit operation is within the multiplier. With four or more pipeline stages, the device inefficiently uses LE resources for the additional pipeline stages. Therefore, if multiplier modules in APEX II or APEX 20K designs are converted to Stratix or Stratix GX designs and do not require the same number of pipeline stages, the surrounding circuitry must be modified to preserve the original functionality of the design.

A design with multipliers feeding an accumulator can use the `altmult_accum (MAC)` megafunction to set the DSP block in multiply-accumulator mode. If the APEX II or APEX 20K design already uses LE-based multipliers feeding an accumulator, the Quartus II software does not automatically instantiate the new `altmult_accum (MAC)` megafunction. Therefore, you should use the MegaWizard Plug-In Manager to instantiate the `altmult_accum (MAC)` megafunction. You can also use LeonardoSpectrum™ or Synplify synthesis tools, which have DSP block inference support, to instantiate the megafunction.

Designs that use multipliers feeding into adders can instantiate the new `altmult_add` megafunction to configure the DSP blocks for two-multipliers adder or four-multipliers adder mode. You can also use the `altmult_add` megafunction for stand-alone multipliers to take advantage of the DSP blocks features such as dynamic sign control of the inputs and the input shift register connections. These features are not accessible through the `lpm_mult` megafunction. If your APEX II or APEX 20K designs already use multipliers feeding an adder/subtractor, the Quartus II software does not automatically infer the new `altmult_add` megafunction. Therefore, you should step through the MegaWizard Plug-In Manager to instantiate the new `altmult_add` megafunction or use LeonardoSpectrum or Synplify synthesis tools, which have DSP block inference support.

Furthermore, the altmult_add and altmult_accum (MAC) megafunctions are only available for Stratix and Stratix GX devices because these megafunctions target Stratix and Stratix GX DSP blocks, which are not available in other device families. If you attempt to use these megafunctions in designs that target other Altera device families, the Quartus II software reports an error message. Use lpm_mult and an lpm_add_sub or an altaccumulate megafunction for similar functionality in other device families.

If you use a third-party synthesis tool, you may be able to avoid the megafunction conversion process. LeonardoSpectrum and Synplify provide inference support for lpm_mult, altmult_add, and altmult_accum (MAC) to use the DSP blocks.

If your design does not require you to implement all the multipliers in DSP blocks, you must manually set a global parameter or a parameter for each instance to force the tool to implement the lpm_mult megafunction in LEs. Depending on the synthesis tools, inference of DSP blocks is handled differently.



For more information about using DSP blocks in Stratix and Stratix GX devices, see the *DSP Blocks in Stratix & Stratix GX Devices* chapter of the *Stratix Device Handbook*.

PLLs & Clock Networks

Stratix and Stratix GX devices provide exceptional clock management with a hierarchical clock network and up to four enhanced phase-locked loops (PLLs) and eight fast PLLs versus the four general-purpose PLLs and four True-LVDS™ PLLs in APEX II devices. By providing superior clock interfacing, numerous advanced clocking features, and significant enhancements over APEX II and APEX 20K PLLs, the Stratix and Stratix GX device PLLs increase system performance and bandwidth.

Clock Networks

There are 16 global clock networks available throughout each Stratix or Stratix GX device as well as two fast regional and four regional clock networks per device quadrant, resulting in up to 40 unique clock networks per device. The increased number of dedicated clock resources available in Stratix and Stratix GX devices eliminate the need to use general-purpose I/O pins as clock inputs.

Stratix EP1S25 and smaller devices have 16 dedicated clock pins and EP1S30 and larger devices have four additional clock pins to feed various clocking networks. In comparison, APEX II devices have eight dedicated clock pins and APEX 20KE and APEX 20KC devices have four dedicated clock pins.

The dedicated clock pins in Stratix and Stratix GX devices can feed the PLL clock inputs, the global clock networks, and the regional clock networks. PLL outputs and internally-generated signals can also drive the global clock network. These global clocks are available throughout the entire device to clock all device resources.

Stratix and Stratix GX devices are divided into four quadrants, each equipped with four regional clock networks. The regional clock network can be fed by either the dedicated clock pins or the PLL outputs within its device quadrant. The regional clock network can only feed device resources within its particular device quadrant.

Each Stratix and Stratix GX device provides eight dedicated fast clock I/O pins `FCLK[7..0]` versus four dedicated fast I/O pins in APEX II and APEX 20K devices. The fast regional clock network can be fed by these dedicated `FCLK[7..0]` pins or by the I/O interconnect. The I/O interconnect allows internal logic or any I/O pin to drive the fast regional clock network. The fast regional clock network is available for general-purpose clocking as well as high fan-out control signals such as clear, preset, enable, `TRDY` and `IRDY` for PCI applications, or bidirectional or output pins.

EP1S25 and smaller devices have eight fast regional clock networks, two per device quadrant. The quadrants in EP1S30 and larger devices are divided in half, and each half-quadrant can be clocked by one of the eight fast regional networks. Additionally, each fast regional clock network can drive its neighboring half-quadrant (within the same device quadrant).

PLLs

Table 3–6 highlights Stratix and Stratix GX PLL enhancements to existing APEX II, APEX 20KE and APEX 20KC PLL features.

Table 3–6. Stratix & Stratix GX PLL vs. APEX II, APEX 20KE & APEX 20KC PLL Features (Part 1 of 2)

Feature	Stratix & Stratix GX		APEX II PLLs	APEX 20KE & APEX 20KC PLLs
	Enhanced PLLs	Fast PLLs		
Number of PLLs	Two (EP1S30 and smaller devices); four (EP1S40 and larger devices) *(9)*	Four (EP1S25 and smaller devices); eight (EP1S30 and larger devices) *(10)*	Four general-purpose PLLs and four LVDS PLLs	Up to four general-purpose PLLs. Up to two LVDS PLLs. *(1)*
Minimum input frequency	3 MHz	15 MHz	1.5 MHz	1.5 MHz
Maximum input frequency	250 to 582 MHz *(2)*	644.5 MHz *(11)*	420 MHz	420 MHz

Table 3–6. Stratix & Stratix GX PLL vs. APEX II, APEX 20KE & APEX 20KC PLL Features (Part 2 of 2)

Feature	Stratix & Stratix GX		APEX II PLLs	APEX 20KE & APEX 20KC PLLs
	Enhanced PLLs	Fast PLLs		
Internal clock outputs per PLL	6	3 *(3)*	2	2
External clock outputs per PLL	Four differential/eight singled-ended or one single-ended *(4)*	Yes *(5)*	1	1
Phase Shift	Down to 160-ps increments *(6)*	Down to 125-ps increments *(6)*	500-ps to 1-ns resolution	0.4- to 1-ns resolution
Time shift	250-ps increments for ± 3 ns *(7)*	No	No	No
M counter values	1 to 512	1 to 32	1 to 160	2 to 160
N counter values	1 to 512	N/A	1 to 16	1 to 16
PLL clock input sharing	No	Yes	Yes	Yes
T1/E1 rate conversion *(8)*	No	No	Yes	Yes

Notes to Table 3–6:

(1) EP20K200E and smaller devices only have two general-purpose PLLs. EP20K400E and larger devices have two LVDS PLLs and four general-purpose PLLs. For more information, see *AN 115: Using the ClockLock & ClockBoost PLL Features in APEX Devices*.

(2) The maximum input frequency for Stratix and Stratix GX enhanced PLLs depends on the I/O standard used with that input clock pin. For more information, see the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* or the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1*.

(3) Fast PLLs 1, 2, 3, and 4 have three internal clock output ports per PLL. Fast PLLs 7, 8, 9, and 10 have two internal clock output ports per PLL.

(4) Every Stratix device has two enhanced PLLs with eight single-ended or four differential outputs each. Two additional enhanced PLLs in EP1S80, EP1S60, and EP1S40 devices each have one single-ended output.

(5) Any I/O pin can be driven by the fast PLL global or regional outputs as an external clock output pin.

(6) The smallest phase shift unit is determined by the voltage-controlled oscillator (VCO) period divided by 8.

(7) There is a maximum of 3 ns between any two PLL clock outputs.

(8) The T1 clock frequency is 1.544 MHz and the E1 clock frequency is 2.048 MHz, which violates the minimum clock input frequency requirement of the Stratix PLL.

(9) Stratix GX EP1SGX10 and EP1SGX25 contain two. EP1SGX40 contains four.

(10) Stratix GX EP1SGX10 and EP1SGX25 contain two. EP1SGX40 contains four.

(11) Stratix GX supports clock rates of 1 Gbps using DPA.

Enhanced PLLs

Stratix and Stratix GX devices provide up to four enhanced PLLs with advanced PLL features. In addition to the feature changes mentioned in Table 3–6, Stratix and Stratix GX device PLLs include many new,

advanced features to improve system timing management and performance. Table 3–7 shows some of the new features available in Stratix and Stratix GX enhanced PLLs.

Table 3–7. Stratix & Stratix GX Enhanced PLL Features

Feature	Description
Programmable duty cycle *(1)*	Allows variable duty cycle for each PLL clock output.
PLL clock outputs can feed logic array *(1)*	Allows the PLL clock outputs to feed data ports of registers or combinatorial logic.
PLL locked output can feed the logic array *(1)*	Allows the PLL locked port to feed data ports of registers or combinatorial logic.
Multiplication allowed in zero-delay buffer mode or external feedback mode	The PLL clock outputs can be a multiplied or divided down ratio of the PLL input clock.
Programmable phase shift allowed in zero-delay buffer mode or external feedback mode *(2)*	The PLL clock outputs can be phase shifted. The phase shift is relative to the PLL clock output.
Phase frequency detector (PFD) disable	Allows the VCO to operate at its last set control voltage and frequency with some long term drift.
Clock output disable *(3)*	PLL maintains lock with output clocks disabled. *(4)*
Programmable lock detect & gated lock	Holds the lock signal low for a programmable number of input clock cycles.
Dynamic clock switchover	Enables the PLL to switch between two reference input clocks, either for clock redundancy or dual-clock domain applications.
PLL reconfiguration	Allows the counters and delay elements within the PLL to be reconfigured in real-time without reloading a programmer object file (**.pof**).
Programmable bandwidth	Provides advanced control of the PLL bandwidth by using the programmable control of the PLL loop characteristics.
Spread spectrum	Modulates the target frequency over a frequency range to reduce electromagnetic interference (EMI) emissions.

Notes to Table 3–7:

(1) These features are also available in fast PLLs.
(2) In addition to the delay chains at each counter, you can specify the programmable phase shift for each PLL output at fine and coarse levels.
(3) Each PLL clock output has an associated clock enable signal.
(4) If the PLL is used in external feedback mode, the PLL will need to relock.

Fast PLLs

Stratix and Stratix GX fast PLLs are similar to the APEX II True-LVDS PLLs in that the *W* setting, which governs the relationship between the clock input and the data rate, and the *J* setting, which controls the width

of the high-speed differential I/O data bus, do not have to be equal. Additionally, Stratix and Stratix GX fast PLLs offer up to three clock outputs, two multiplied high-speed PLL clocks to drive the serializer/deserializer (SERDES) block and/or an external pin, and a low-speed clock to drive the logic array. You can use fast PLLs for both high-speed interfacing and for general-purpose PLL applications.

Table 3–8 shows the differences between Stratix and Stratix GX fast PLLs and APEX II and APEX 20K True-LVDS PLLs.

Table 3–8. Stratix & Stratix GX Fast PLL vs. APEX II & APEX 20K True-LVDS PLL

Feature	Stratix & Stratix GX	APEX II	APEX 20KE APEX 20KC
Number of fast PLLs or True-LVDS PLLs *(1)*	Four (EP1S25 and smaller devices) fast PLLs Eight (EP1S30 and larger devices) fast PLLs *(4)*	Four True-LVDS PLLs	Two True-LVDS PLLs *(2)*
Number of channels per transmitter/receiver block	20	18	18
VCO frequency	300 to 840 MHz *(5)*	200 MHz to 1GHz	200 to 840 MHz
Minimum input frequency *M* = 4, 5, 6	300 – *M* MHz	50 MHz	50 MHz *M* = 4 *(3)*
Minimum input frequency *M* = 7, 8, 9, 10	300 – *M* MHz	30 MHz	30 MHz *M* = 7, 8 *(3)*

Notes to Table 3–8:
(1) You can also use Stratix and Stratix GX device fast PLLs for general-purpose PLL applications.
(2) EP20K400E and larger devices have two True-LVDS PLLs.
(3) In APEX 20KE and APEX 20KC devices, *M* = 4, 7, or 8.
(4) Stratix GX EP1SGX10 and EP1SGX25 contain two. EP1SGX10 contains four.
(5) Stratix GX supports a frequency range of 300–1000 MHz (using DPA).

The Stratix and Stratix GX fast PLL VCO frequency range is 300 to 840 MHz, and the APEX II True-LVDS PLL VCO frequency range is 200 MHz to 1 GHz. Therefore, you must update designs that use a data rate of less than 300 megabits per second (Mbps) to use the enhanced PLLs and M512 RAM blocks in SERDES bypass mode. Additionally, you must update designs that use a data rate faster than 840 Mbps.

altpll Megafunction

Altera recommends that you replace instances of the `altclklock` megafunction with the `altpll` megafunction to take advantage of new Stratix and Stratix GX PLL features. Although in most cases you can retarget your APEX II or APEX 20K design to a Stratix or Stratix GX

device with the altclklock megafunction, there are specific cases where you must use the altpll megafunction, as explained in this section.

In the MegaWizard Plug-In Manager, select the altpll megafunction in the I/O directory from the **Available Megafunctions** box (see Figure 3–9). The altclklock megafunction is also available from the Quartus II software for backward compatibility, but instantiates the new altpll megafunction when targeting Stratix or Stratix GX devices. The Quartus II Compiler automatically selects whether the altpll module uses either an enhanced PLL or a fast PLL based on the design's PLL needs and the feature requirements of each PLL.

Figure 3–9. altpll Megafunction Selection in the MegaWizard Plug-In Manager



You can compile APEX II, APEX 20KE, and APEX 20KC designs using the altclklock megafunction in normal mode for Stratix and Stratix GX devices without updating the megafunction. However, you should replace the altclklock megafunction with the altpll megafunction. If the Quartus II software cannot implement the requested clock multiplication and division of the PLL, the compiler reports an error message with the appropriate reason stated.

APEX II, APEX 20KE, and APEX 20KC devices have only one external clock output available per PLL. Therefore, when retargeting an APEX II, APEX 20KE, or APEX 20KC design that uses PLLs in zero delay buffer mode or external feedback mode to a Stratix or Stratix GX device, you should replace instances of the `altclklock` megafunction. If an APEX II, APEX 20KE, or APEX 20KC `altclklock` module only uses one PLL clock output (internal or external) and is compiled to target a Stratix or Stratix GX device, the design compiles successfully with a warning that the design uses the Stratix or Stratix GX PLL external clock output, `extclk0`. However, if the APEX II, APEX 20KE, or APEX 20KC PLL has more than one PLL clock output, you must replace instances of the `altclklock` megafunction with the `altpll` megafunction because the Quartus II Compiler does not know which PLL clock output is fed to an external output pin or fed back to the Stratix or Stratix GX device `fbin` pin. For example, if an APEX II, APEX 20KE, or APEX 20KC design with an `altclklock` megafunction uses the `clock0` output port to feed the external clock output pin and the `clock1` output port to feed the internal logic array, the Quartus II software generates an error during compilation and you must use the MegaWizard Plug-In Manager to instantiate the `altpll` megafunction. By using the `altpll` megafunction, you can choose which of the four external clock outputs to use and take advantage of the new Stratix and Stratix GX PLL features now available in the zero delay buffer mode or external feedback mode.

Timing Analysis

When the Quartus II software performs a timing analysis for APEX II, APEX 20KE, or APEX 20KC designs, PLL clock settings override the project clock settings. However, during timing analysis for Stratix and Stratix GX designs using PLLs, the project clock settings override the PLL input clock frequency and duty cycle settings. The MegaWizard Plug-In Manager does not use the project clock settings to determine the `altpll` parameters. This saves time with designs that use features such as clock switchover or PLL reconfiguration because the Quartus II software can perform a timing analysis without recompiling the design. It is important to note the following:

- A warning during compilation reports that the project clock settings overrides the PLL clock settings.
- The project clock setting overrides the PLL clock settings for timing-driven compilation.
- The compiler will check the lock frequency range of the PLL. If the frequency specified in the project clock settings is outside the lock frequency range, the PLL clock settings will not be overridden.
- Performing a timing analysis without recompiling your design does not change the programming files. You must recompile your design to update the programming files.

- A **Default Required f_{MAX}** setting does not override the PLL clock settings. Only individual clock settings override the PLL clock settings.

Therefore, you can enter different project clock settings corresponding to new PLL settings and accelerate timing analysis by eliminating a full compilation cycle.



For more information about using Stratix and Stratix GX PLLs, see the *General-Purpose PLLs in Stratix & Stratix GX Devices* chapter.

I/O Structure

The Stratix and Stratix GX I/O element (IOE) architecture is similar to the APEX II architecture, with a total of six registers and a latch in each IOE. The registers are organized in three sets: two output registers to drive a single or double-data rate (DDR) output path, two input registers and a latch to support a single or DDR input path, and two output enable registers to enhance clock-to-output enable timing or for DDR SDRAM interfacing. A new synchronous reset signal is available to each of the three sets of registers for preset or clear, or neither. In addition to the advanced IOE architecture, the Stratix and Stratix GX IOE features dedicated circuitry for external RAM interfacing, new I/O standards, differential on-chip termination, and high-speed differential I/O standard support.

External RAM Interfacing

The advanced Stratix and Stratix GX IOE architecture includes dedicated circuitry to interface with external RAM. This circuitry provides enhanced support for external high-speed memory devices such as DDR SDRAM and FCRAM. The DDR SDRAM interface uses a bidirectional signal, DQS, to clock data, DQ, at both the transmitting and receiving device. Stratix and Stratix GX devices transmit the DQS signal with the DQ data signals to minimize clock to data skew.

Stratix and Stratix GX devices include groups of programmable DQS and DQ pins, in the top and bottom I/O banks of the device. Each group consists of a DQS pin that supports a fixed number of DQ pins. The number of DQ pins depends on the DQ bus mode. When using the external RAM interfacing circuitry, the DQS pin drives a dedicated clock network that feeds the DQ pins residing in that bank. The Stratix and Stratix GX IOE has programmable delay chains that can phase shift the DQS signal by 90° or 72° to ensure data is sampled at the appropriate point in time. Therefore, the Stratix and Stratix GX devices make full use of the IOEs, and remove the need to build the input data path in the logic array. You can make these I/O assignments in the Quartus II Assignment Organizer.

For more information on external RAM interfacing, see the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* or the *Stratix GX Device Family Data Sheet* in the *Stratix GX Device Family Handbook, Volume 1*.

I/O Standard Support

The Stratix and Stratix GX devices support all of the I/O standards that APEX II and APEX 20K devices support, including high-speed differential I/O standards such as LVDS, LVPECL, PCML, and HyperTransport™ technology, differential HSTL on input and output clocks, and differential SSTL on output clocks. Stratix and Stratix GX devices also introduce support for SSTL-18 Class I & II. Similar to APEX II devices, Stratix and Stratix GX devices only support certain I/O standards in designated I/O banks. In addition, `vref` pins are dedicated pins in Stratix and Stratix GX devices and now support up to 40 input pins.

For more information about I/O standard support in Stratix and Stratix GX devices, see the *Selectable I/O Standards in Stratix & Stratix GX Devices* chapter.

High-Speed Differential I/O Standards

Stratix and Stratix GX devices support high-speed differential interfaces at speeds up to 840 Mbps using high-speed PLLs that drive a dedicated clock network to the SERDES. Each fast PLL can drive up to 20 high-speed channels. Stratix and Stratix GX devices use enhanced PLLs and M512 RAM blocks to provide up to 420 Mbps performance for SERDES bypass clock interfacing. There is no restriction on the number of channels that can be clocked using this scenario.

Stratix and Stratix GX devices have a different number of differential channels than APEX II devices. Tables 3–9 and 3–10 highlight the number of differential channels supported in Stratix and Stratix GX devices.

Table 3–9. Number of Dedicated DIfferential Channels in Stratix Devices (Part 1 of 2) *Note (1)*

Device	Pin Count	Number of Receiver Channels	Number of Transmitter Channels
EP1S10	672	36	36
	780	44	44
EP1S20	672	50	48
	780	66	66

Table 3–9. Number of Dedicated DIfferential Channels in Stratix Devices (Part 2 of 2) Note (1)

Device	Pin Count	Number of Receiver Channels	Number of Transmitter Channels
EP1S25	672	58	56
	780	66	70
	1,020	78	78
EP1S30	780	66	70
	956	80	80
	1,020	80	80
		2	2
EP1S40	956	80	80
	1,020	80	80
		10	10
	1,508	80	80
		10	10
EP1S60	956	80	80
	1,020	80	80
		10	12
	1,508	80	80
		36	36
EP1S80	956	80	80
		0	40
	1,508	80	80
		56	72

Note to Table 3–9:

(1) For information on channel speeds, see the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* and the *High-Speed Differential I/O Interfaces* chapter in the *Stratix Device Handbook, Volume 2*.

Table 3–10. Number of Dedicated DIfferential Channels in Stratix GX Devices (Part 1 of 2) Note (1)

Device	Pin Count	Number of Transceivers	Number of Source-Synchronous Channels
EP1SGX10 C	672	4	22
EP1SGX10 D	672	8	22

Table 3–10. Number of Dedicated DIfferential Channels in Stratix GX Devices (Part 2 of 2) Note (1)

Device	Pin Count	Number of Transceivers	Number of Source-Synchronous Channels
EP1SGX25 C	672	4	39
EP1SGX25 D	672/1,020	8	39
EP1SGX25 F	1,020	16	39
EP1SGX40 D	1,020	8	45
EP1SGX40 G	1,020	20	45

Note to Table 3–10:

(1) For information on channel speeds, see the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1* and the *High-Speed Source-Synchronous Differential I/O Interfaces in Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2*.

The differential I/O within Stratix GX also provides dynamic phase alignment (DPA). DPA enables the differential I/O to operate up to 1 Gbps per channel. DPA automatically and continuously tracks fluctuations caused by system variations and self-adjusts to eliminate the phase skew between the multiplied clock and the serial data. The block contains a dynamic phase selector for phase detection and selection, a SERDES, a synchronizer, and a data realigner circuit. You can bypass the dynamic phase aligner without affecting the basic source-synchronous operation of the channel by using a separate deserializer.

If you compile an APEX II LVDS design that uses clock-data synchronization (CDS) for a Stratix or Stratix GX device, the Quartus II software issues a warning during compilation that Stratix and Stratix GX devices do not support CDS.

Stratix and Stratix GX devices offer a flexible solution using new byte realignment circuitry to correct for byte misalignment by shifting, or slipping, data bits. Stratix and Stratix GX devices activate the byte realignment circuitry when an external pin (`rx_data_align`) or an internal custom-made state machine asserts the `SYNC` node high.

APEX II, APEX 20KE, and APEX 20KCdevices have a dedicated transmitter clock output pin (`LVDSTXOUTCLK`). In Stratix and Stratix GX devices, a transmitter `dataout` channel with an LVDS clock (fast clock) generates the transmitter clock output. Therefore, you can drive any channel as an output clock to an I/O pin, not just dedicated clock output pins. This solution offers better versatility to address various applications that require more complex clocking schemes.



For more information on differential I/O support, data realignment, and the transmitter clock output in Stratix and Stratix GX devices, see the *High-Speed Differential I/O Interfaces in Stratix Devices* chapter.

altlvds Megafunction

To take full advantage of the high-speed differential I/O standards available in Stratix and Stratix GX devices, you should update each instance of the `altlvds` megafunction in APEX II, APEX 20KE, and APEX 20KC designs. In the MegaWizard Plug-In Manager, choose the `altlvds` megafunction, select Stratix or Stratix GX as the target device family, update the megafunction, and recompile your design.

The `altlvds` megafunction supports new Stratix and Stratix GX parameters that are not available for APEX II, APEX 20KE, and APEX 20KC devices. Tables 3–11 and 3–12 describe the new parameters for the LVDS receiver and LVDS transmitter, respectively.

Table 3–11. New altlvds Parameters for Stratix LVDS Receiver Note (1)

Parameter	Function
`input_data_rate` *(2)*	Specifies the data rate in Mbps. This parameter replaces the multiplication factor *W*.
`inclock_data_alignment`	Indicates the alignment of `rx_inclk` and `rx_in` data.
`rx_data_align`	Drives the data alignment port of the fast PLL and enables byte realignment circuitry.
`registered_data_align_input`	Registers the `rx_data_align` input port to be clocked by `rx_outclock`.
`common_rx_tx_pll` *(3)*	Indicates the fast PLL can be shared between receiver and transmitter applications.

Table 3–12. New altlvds Parameters for Stratix LVDS Transmitter (Part 1 of 2) Note (1)

Parameter	Function
`output_data_rate` *(2)*	Specifies the data rate in Mbps. This parameter replaces the multiplication factor *W*.
`inclock_data_alignment`	Indicates the alignment of `tx_inclk` and `tx_in` data.
`outclock_alignment`	Specifies the alignment of `tx_outclock` and `tx_out` data.
`registered_input`	Specifies the clock source for the input synchronization registers, which can be either `tx_inclock` or `tx_coreclock`. Used only when the **Registered Inputs** option is selected.

Table 3–12. New altlvds Parameters for Stratix LVDS Transmitter (Part 2 of 2) Note (1)	
Parameter	**Function**
common_rx_tx_pll (3)	Indicates the fast PLL can be shared between receiver and transmitter applications.

Notes to Tables 3–11 and 3–12:
(1) You can specify these parameters in the MegaWizard Plug-In Manager.
(2) You must specify a data rate in the MegaWizard Plug-In Manager instead of a *W* factor.
(3) The same fast PLL can be used to clock both the receiver and transmitter only if both are running at the same frequency.

Above the standard I/O offered by APEX II, APEX 20K, and Stratix devices, Stratix GX devices provide up to 20 3.175 Gbps transceivers. The transceivers provide high-speed serial links for chip-to-chip, backplane, and line-side connectivity and support a number of the emerging high-speed protocols. You can find more information in the *Stratix GX Family Data Sheet* in the *Stratix GX Family Handbook, Volume 1*.

Configuration

The Stratix and Stratix GX devices supports all current configuration schemes, including the use of enhanced configuration devices, passive serial (PS), passive parallel asynchronous (PPA), fast passive parallel (FPP), and JTAG. Stratix and Stratix GX devices also provide a number of new configuration enhancements that you can take advantage of when migrating APEX II and APEX 20K designs to Stratix and Stratix GX devices.

Configuration Speed & Schemes

You can configure Stratix and Stratix GX devices at a maximum clock speed of 100 MHz, which is faster than the 66-MHz and 33-MHz maximum configuration speeds for APEX II and APEX 20K devices, respectively. Similar to APEX II devices, you can use 8-bit parallel data to configure Stratix and Stratix GX devices (the target device can receive byte-wide configuration data on each clock cycle) significantly speeding up configuration times.

You can select a configuration scheme based on how the MSEL pins are driven. Stratix and Stratix GX devices have three MSEL pins (APEX II and APEX 20K devices have two MSEL pins) for determining the configuration scheme.



For more information about Stratix and Stratix GX configuration schemes, see the *Configuring Stratix & Stratix GX Devices* chapter.

Remote Update Configuration

The APEX 20K device family introduced the concept of remote update configuration, where you could send the APEX 20K device new configuration files from a remote source and the device would store the files in flash memory and reconfigure itself with the new configuration data. The Stratix and Stratix GX devices enhance support for remote update configuration with new, dedicated circuitry to handle and recover from errors. If an error occurs either during device configuration or in user mode, this new circuitry reconfigures the Stratix or Stratix GX device to a known state. Additionally, the Stratix and Stratix GX devices have a user watchdog timer to ensure the application configuration data executes successfully during user mode. User logic must continually reset this watchdog timer in order to validate that the application configuration data is functioning properly.



For more information about how to use the remote and local update modes, see the *Remote System Configuration with Stratix & Stratix GX Devices* chapter.

JTAG Instruction Support

Stratix and Stratix GX devices support two new JTAG instructions, `PULSE_NCONFIG` and `CONFIG_IO`. The `PULSE_NCONFIG` instruction emulates pulsing the `nCONFIG` signal low to trigger reconfiguration, while the actual `nCONFIG` pin on the device is unaffected. The `CONFIG_IO` instruction allows you to use the JTAG chain to configure I/O standards for all pins. Because this instruction interrupts device configuration, you should reconfigure the Stratix or Stratix GX device after you finish JTAG testing to ensure proper device operation. Table 3–13 compares JTAG instruction support in Stratix and Stratix GX devices versus APEX II and APEX 20K devices. For further information about the supported JTAG instructions, see the appropriate device family data sheet.

Table 3–13. JTAG Instruction Support (Part 1 of 2)

JTAG Instruction	Stratix	APEX II	APEX 20K
SAMPLE/PRELOAD	✓	✓	✓
EXTEST	✓	✓	✓
BYPASS	✓	✓	✓
USERCODE	✓	✓	✓
IDCODE	✓	✓	✓
ICR Instructions	✓	✓	✓

Table 3–13. JTAG Instruction Support (Part 2 of 2)

JTAG Instruction	Stratix	APEX II	APEX 20K
SignalTap™ II Instructions	✓	✓	✓
HIGHZ	✓	✓	
CLAMP	✓	✓	
PULSE_NCONFIG	✓		
CONFIG_IO	✓		

Conclusion

The Stratix and Stratix GX devices extend the advanced features available in the APEX II and APEX 20K device families to deliver a complete system-on-a-programmable-chip (SOPC) solution. By following these guidelines, you can easily transition current APEX II and APEX 20K designs to take advantage of the new features available in Stratix and Stratix GX devices.

4. Stratix GX Board Design Guidelines

SGX53001-1.0

Introduction

Digital board design has become more complicated over the years. Devices have 0.13-µm gate lengths and are powered by 1.5-V power supplies. While the trend toward miniaturization continues, device speeds are increasing. FPGAs now have embedded transceivers that can support serial data transmission at 3.125 Gigabits per second (Gbps). The low-voltage supply results in decreased noise margin; therefore, small voltage noise on the board can cause a bit to flip. Similarly, voltage noise directly increases the clock jitter, which reduces the timing margin. If the jitter is large enough, the bit transition boundaries can become fuzzy, and the current data bit could be confused for the past or future bit. To reduce voltage and timing noise and ensure error-free data transmission, good design techniques are essential.

The following things are critical when designing a successful high-speed digital board:

- Clean power supply design (see "Power Circuitry" on page 4–6)
- Decoupling capacitor selection (see "Decoupling Circuitry Design" on page 4–13)
- Analog ground and power isolation (see "Ground Plane & Island Design" on page 4–30)
- Transmission line termination (see "Transmission Line Termination" on page 4–52)
- Crosstalk minimization (see "Crosstalk" on page 4–72)
- Impedance discontinuity minimization (see "Transmission Line Routing" on page 4–58)
- Proper differential signal routing (see "Transmission Line Routing" on page 4–58)

This document explains high-speed board design concepts and techniques as applied to Stratix® GX devices and explains the major aspects of successful high-speed board design. The Stratix GX device and the Stratix GX development board are used as the platforms for the tests and measurements described in this document. Topics covered include clock circuitry design, power circuitry design, power supply decoupling, transmission line topologies, length matching, AC versus DC coupling, termination techniques, time domain reflectometer (TDR) usage, and S-parameters.

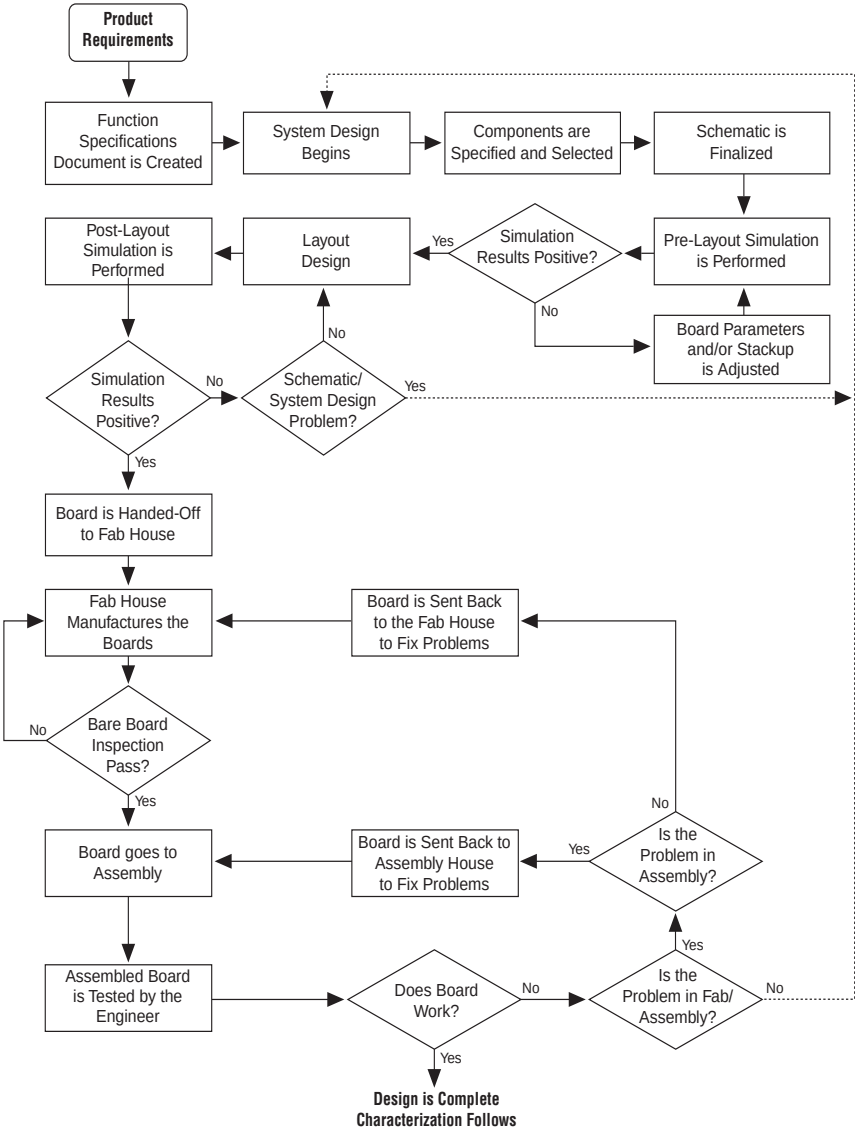
Board Design Overview

A typical high-speed board design begins with a product requirements document, a concept review, and a functional specification. After the specification is finalized, actual system design begins with the selection of key components. At this point, timing analysis for all the interfaces and power analysis—to determine the power requirements—is performed. Based on the power analysis, power supply modules and regulators are chosen, and a decoupling scheme is specified. Based on the timing analysis, length-matching criteria for the buses is established. Upon completion of these tasks, as well as completion of the system design and selection of remaining components, the schematics are created. These are typically reviewed among engineers who make any necessary changes, after which they are given to the layout designer. A layout guidelines document, which specifies how to place components on the board and how to route the traces, is created and given to the layout designer as well.

A pre-layout simulation is performed to determine the ideal stackup, trace widths, spacing, and other routing requirements. Any changes to the schematic, based on the simulation results, are incorporated in the schematic and provided to the layout designer. When the layout is complete, a post-layout simulation is performed on the critical sections of the board to ensure that there are no major signal integrity problems. Based on the results of the post-layout simulation, any changes required are incorporated into the layout, and finally the layout is released to the fabrication house for board manufacturing.

Figure 4–1 shows a typical board design process.

Figure 4–1. Simplified View of a Typical High-Speed Board Design Process



Support Circuitry Design

Support circuitry includes clocks, power, and decoupling. This section contains detailed guidelines for designing these parts of a high-speed board.

Clock Circuitry

Clocks serve as references for digital signals and must be designed very carefully. Any noise on the clock signal reduces the timing margin for digital signals. Typically, on-board clocks are derived from crystal-based oscillators. For the Stratix GX development board, Altera use the standard 3.3-V SM7745DV series CMOS crystals from Pletronics for high-frequency applications. This crystal has ± 50 ppm stability and less than 1 ps RMS jitter from 12 kHz to 20 MHz from the carrier. For low-frequency applications Altera uses the standard 3.3 V SM7745HV series CMOS crystals from Pletronics. The stability of this crystal is ± 50 ppm, but the jitter is not specified. Altera recommends these or equivalent crystals.

For maximum noise immunity, the crystal oscillator's CMOS output typically needs to be converted to a differential standard like LVDS before it is fed to the Stratix GX device. An ICS8545 CMOS to LVDS buffer from Integrated Circuit Systems, Inc. achieves this conversion. A side benefit of using a clock buffer like this one is that it provides multiple outputs. This feature is very helpful because clock lines need to have one source and one destination for best signal integrity.

Other things to remember for optimal clock performance are:

- Place the crystal oscillator close to the driver circuitry for best jitter performance.
- Route the differential clock lines tightly coupled to each other.
- The oscillators and the drivers need clean power supplies.
- Place series termination at the clock output if the trace is too long.
- Place parallel differential termination close to the receiver pins.

Whether a particular trace is too long depends on the edge rate of the clock and on the speed at which the signal propagates on the board. If the electrical length of the trace is small (1/10 or 10%)[1] compared to the rise time, then no termination is needed. Otherwise, use series termination. The propagation speed for microstrip and stripline, two of the commonly used transmission line topologies, is shown in the following equations[2]:

$$Speed(Microstrip) = \frac{1}{85\sqrt{0.475\varepsilon_r + 0.67}} inches / ps$$

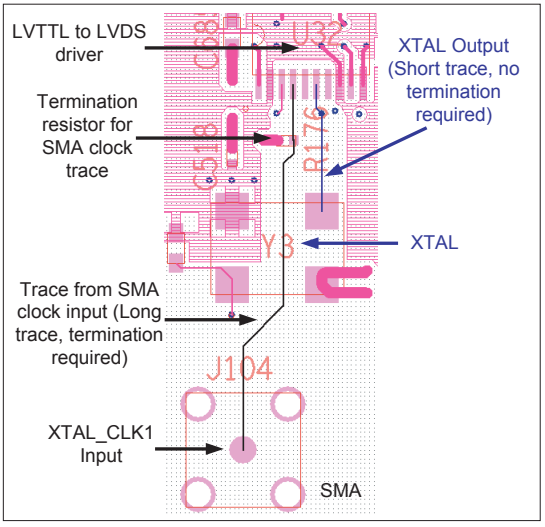
[1] Some designers use 1/6 or 16% as the cut-off point. The important thing to remember is that the length needs to be small.
[2] Refer to *High-Speed Digital Design: A Handbook of Black Magic* by Johnson and Graham, pp186-188.

$$Speed(Stripline) = \frac{1}{85\sqrt{\varepsilon_r}} inches / ps$$

Using ε_r of 4.5 for typical FR-4 material, the propagation speed for microstrip and stripline traces works out to be 0.007021 inches/ps and 0.005546 inches/ps, respectively. Expressed another way, the signals travel 142 ps/inch in microstrip traces and 180 ps/inch in stripline traces.

Case Study: As a case study, the microstrip trace length from the output of the 156.25 MHz crystal to the driver (see Figure 4–2) is about 0.28 in on the Stratix GX development board. The signal takes 142 * 0.28 = 39.76 ps to traverse this distance. The rise time of the crystal output is 1.0 ns (see the *Pletronics SM7745HV* data sheet). The signal transit time is less than 4% of the signal rise time, so no termination is needed. When the clock is being brought to the board through an SMA connector (J104), the trace length from the SMA to the clock driver is about four times the crystal transit time calculated earlier, in addition to the SMA cable length. In this case, a termination is definitely needed. Use R176 for termination, as shown in Figure 4–2.

Figure 4–2. Layout Example for Clock Distribution



The specifications for some I/O standards always require termination regardless of the trace length. These include PCML, LVDS, LVPECL, SSTL-II, and HSTL-II.

Isolated Power & Ground Plane Design

Isolated power and ground planes are created by using ferrite beads. A ferrite bead acts as a DC short but presents an impedance to the passage of high-frequency noise. It can be beneficial in systems that have lots of switching noise. Sources of switching noise include switching power supplies and simultaneous switching outputs. For Stratix GX devices, Altera recommends isolating the high-speed ground (GND_GXB) from the digital ground (GND), so that the noise from the digital ground does not reach the high-speed circuitry. In the Stratix GX development board, there are three ground planes: GND, GND_PLL, and GND_GXB, for characterization purposes. For most high-density designs, Altera recommends only two ground planes: GND and GND_GXB. If the board does not have lots of switching circuitry and is otherwise cleanly designed, one ground plane is acceptable.

Currently, the benefits of split power planes for the transceiver quads are under evaluation. Preliminary tests suggest that for most cases the benefits of isolating power for the quads are minimal. See "Power Plane & Island Design" and "Ground Plane & Island Design" for more information.

Power Circuitry

As gate lengths shrink, the power supply voltages of transistors also decrease. As a result, the noise margin also decreases. The reduced noise margin makes the transistors even more sensitive to noise on the power supply. The features of the board must be understood before the power supply can be designed. Some key characteristics of a high-density board with one or more FPGAs include:

- Hundreds of I/O pins with varying voltage standards switching at several frequencies.
- The core LE use is highly unpredictable and depends on the actual design loaded. The user could load a design that uses close to 100% of the LEs, embedded memory, and hard logic (like transceivers), and run everything at full speed.
- Analog and digital signals on the board. For example, TTL switching. is digital while Stratix GX transceiver outputs are analog signals.
- The potential for noise to couple between different power and ground planes.

Regulator Selection

Altera recommends selecting linear regulators as much as possible, because they are easy to design and their performance is less critical to layout. You must sometimes select switching regulators, because linear

regulators cannot supply the desired amount of power efficiently. In those cases, you must design and layout the switching regulators very carefully. All major switching components and traces must be closely contained on the same layer.

The Stratix GX development board uses Fairchild FAN5066 switching regulators (U36 and U37) for 3.3-V and 1.5-V digital supplies. Refer to "Switching Regulator Layout Example" on page 4–8 for details on the layout of these regulators on the Stratix GX development board.

To support a board with the characteristics listed earlier, the power circuitry must be able to do the following:

- Supply large amounts of current to the core and I/O voltages.
- Supply clean power and ground to sensitive analog components such as PLLs and transceivers.
- Maintain good efficiency to prevent excess heat dissipation on the regulators.

Voltage regulators come in two forms: linear and switching. *Linear* regulators are easy to design and provide very clean output voltage, but often they cannot provide large amounts of current without getting very hot. *Switching* regulators can provide large amounts of currents (over 10 A) with good efficiency, so there is no need for a heat sink. Switching regulators require very good design and layout practices to achieve good noise performance.

Use a switching regulator for the core supply, because current draw on the core can be high, and linear regulators cannot supply the necessary current without requiring a large heat sink. For example, if the core current needs to be 5 A, and the input and output voltages of the linear regulator are 3.3/1.5 volts, the heat dissipation would be approximately 5 * (3.3 – 1.5) = 9 Watts. This much heat requires a large heat sink on the linear regulator, which may not be possible to use on the board because of form factors, costs, or aesthetics.

For sensitive circuitry such as phase-locked loops (PLLs), transceivers, and double-data rate (DDR) memory, linear regulators are preferred. Choose the linear regulator based on the current requirements. Sometimes a small heat sink is required, depending on the current being drawn from the regulator and the voltage drop from the input to the output. Refer to *AN 185: Thermal Management Using Heat Sinks* for more details on regulator and heat sink selection.

If there are hundreds of LVTTL signals on the I/O, all of which can potentially switch at the same time, a switching regulator may be necessary, because the current requirement can be several Amperes.

Switching Regulator Layout Example

Linear regulators are relatively easy to design and lay out. Switching regulators, on the other hand, require a great deal of care. Typically schematics for both the linear and switching regulators can be taken directly from the vendor data sheet or the application note. Some of the linear regulators Altera recommends include Micrel MIC29502BU, National LMS1585 and National LP2995M. Similarly, Altera has used Fairchild FAN5066 for switching regulators with good results.

With switching regulators, component layout is extremely important. To explain this point further, consider an actual example. Figure 4–3 shows a capture plot of the FAN5066 circuit from the Stratix GX development board.

This switching regulator runs from a 5.0-V supply and can generate an output voltage that equals the voltage reference. Set the reference voltage on the board by applying the required voltage on the `CNTRL` pin (pin 16). The 12.0-V supply is used only for biasing and draws very little current. The overall circuitry consists of the FAN5066 switching controller, external FETs (Q3 and Q4) to boost the current output capability, and the associated resistors, diodes, inductors, and capacitors as recommended by the manufacturer.

The switching regulator is essentially a feedback loop consisting of a pulse width modulator (PWM). The feedback loop compares the actual output voltage to the desired voltage (reference voltage). If the actual output is less than desired, then the high drive (HIDRV) output is asserted, which turns the upper field effect transistor (FET) on and charges the output capacitor (C73). The end result is that the output voltage climbs. If the actual output is more than desired, the low drive (LODRV) output is asserted, which turns the lower FET on and discharges the output capacitor (C73). The end result is that the output voltage drops. In steady state the output voltage equals the input reference voltage. Figure 4–4 shows the schematic of the switching regulator used in the SGX development board.

Figure 4–3. Switching Regulator Schematic

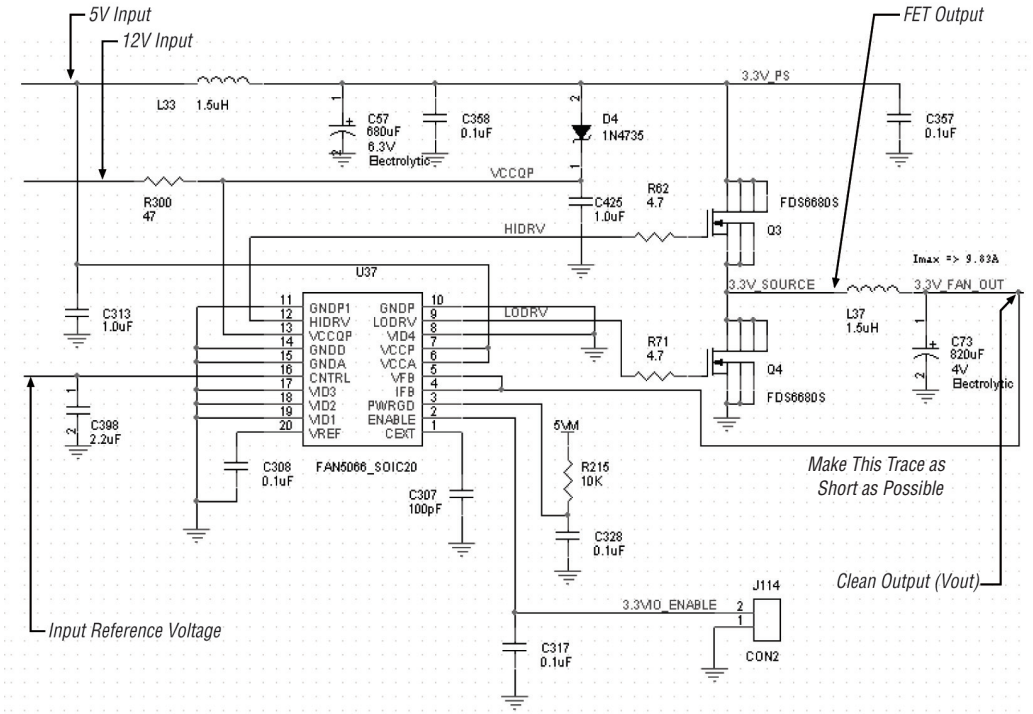
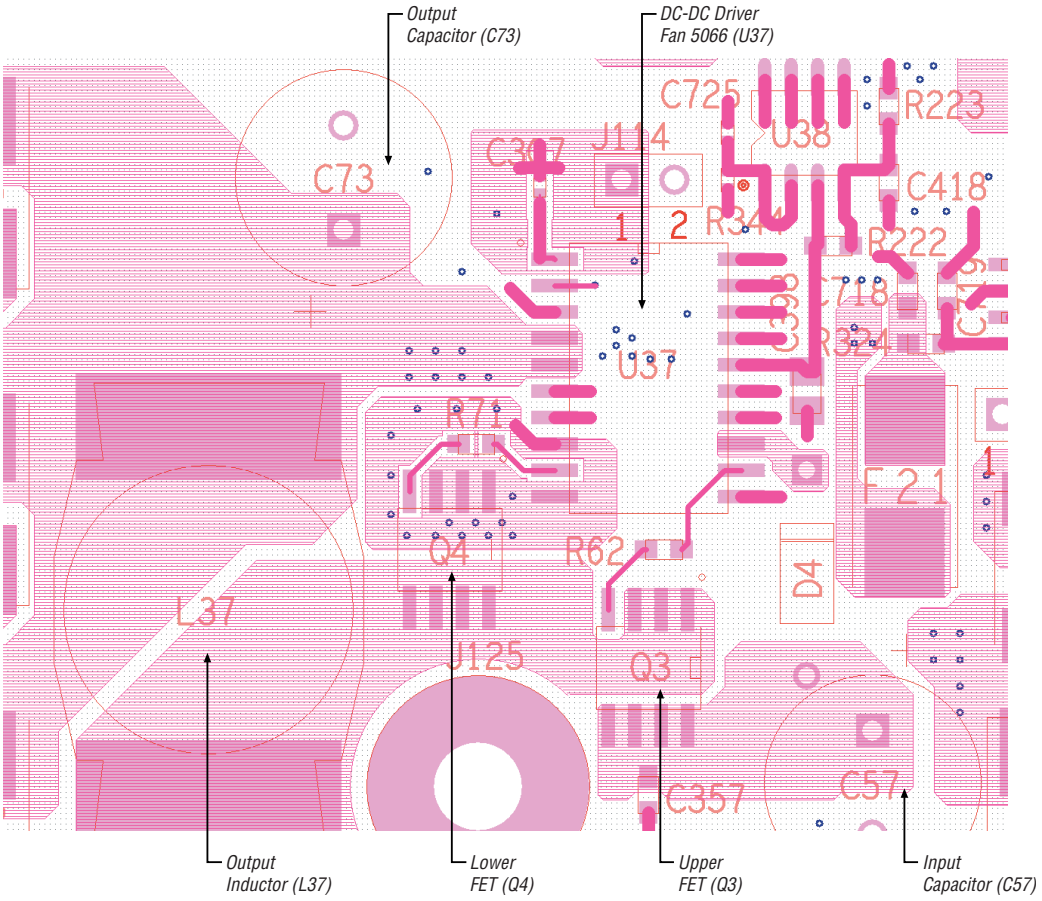


Figure 4–4. Switching Regulator Layout



Figures 4–5 through 4–7 show the waveforms at the gates of the upper and the lower FET. While the lower FET switches from about 0 V to 5 V, the upper FET switches from approximately 0 V to 10 V. These are large magnitudes on a board with 3.3-, 2.5-, or 1.5-volt logic signals. Routing fast switching signals close to sensitive analog and digital signals and across multiple layers can cause the switching signals to radiate energy to other critical traces. Consequently, it is very important to contain the large switching signals compactly on the same layer.

Figure 4–5. Switching Waveforms at the Output of the FET

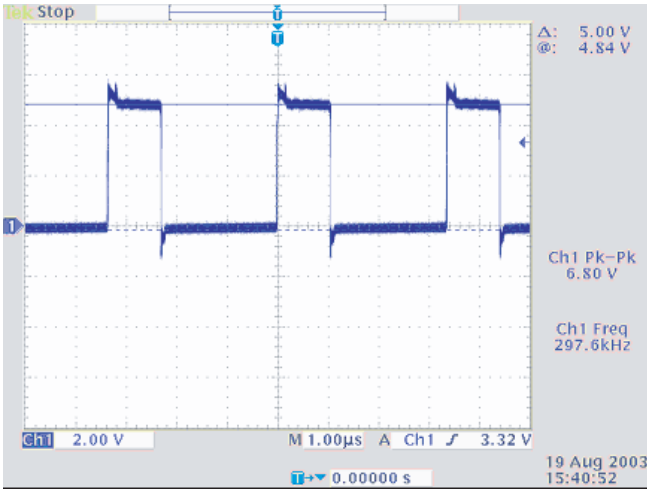


Figure 4–6. Switching Waveforms at the Gate of the Upper FET

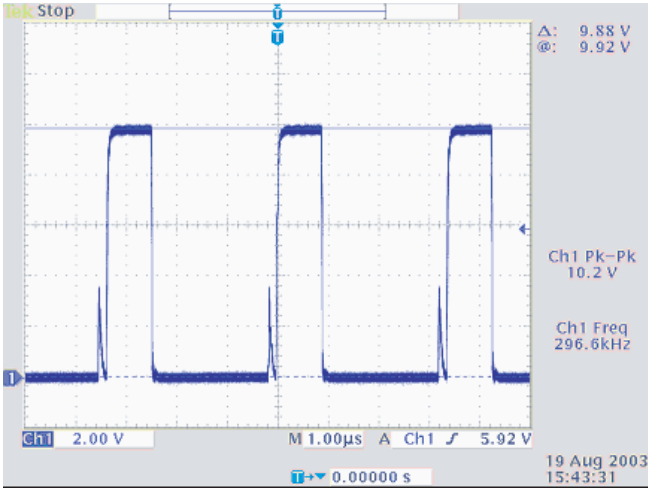
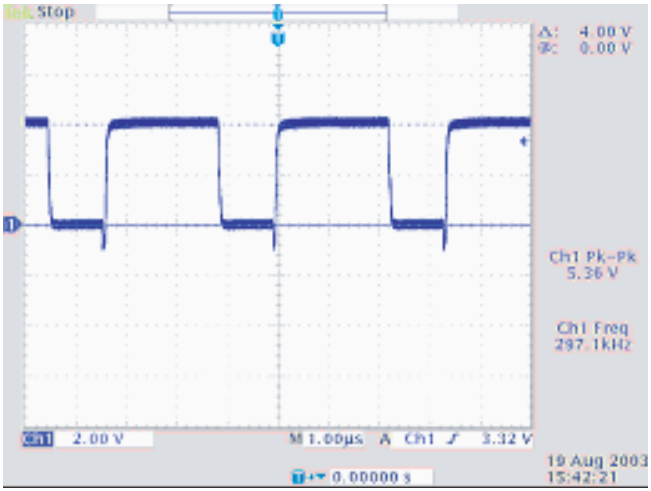
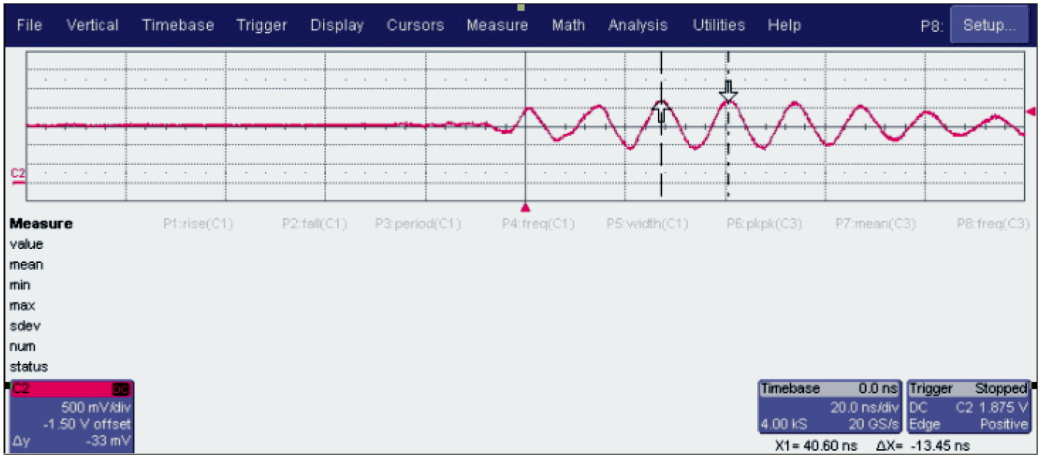


Figure 4–7. Switching Waveforms at the Gate of the Lower FET



Altera has compared the performance improvement of the output ripple when using a good layout versus a poor layout. The results of excessive ripple at the output of a switching regulator because of poor board layout, are shown in Figure 4–8. The output ripple is almost 1.5 V peak to peak. This example may be an extreme case, but the board in this example was designed with an intentionally poor layout. The layout was later improved by reducing the length of the HIDRV and LODRV signals and minimizing via count to determine how much improvement would result. With the improved layout, the noise decreased to about 71 mV peak to peak.

Figure 4–8. Excessive Ripple at the Output of a Switching Regulator



Altera recommends that you follow the switching regulator layout of the Stratix GX development board and review the Fairchild Semiconductor FAN5066 data sheet in detail. The guidelines are summarized here:

- Place all components on the same layer, if possible.
- Minimize the trace lengths for the gate drives (HIDRV and LODRV).
- Place all the 0.1 µF decoupling capacitors close to the pins.
- Place the HIDRV, LODRV, VCCQP and the FET output traces far away from the analog sections of the chip, including VCNTRL (pin 16), VFB, and IFB (feedback pins 4 and 5), and C_{EXT} (pin 1). Do all routing for the switching signals (VCCQP, HIDRV, LODRV, and FET output) on the component layer to avoid radiation to multiple layers through the vias.
- Surround the capacitor connected to pin 1 (C_{EXT}) with a ground guard trace and place a ground plane beneath it.

Decoupling Circuitry Design

You can divide decoupling capacitor circuitry into two categories: bulk decoupling and local decoupling. Bulk decoupling uses large decoupling capacitors (often tens to hundreds of microfarads), usually placed close to the regulators and designed to work for low frequencies. Local decoupling involves smaller capacitors (typically 2.2 µF, 0.1 µF, or 0.01 µF) that are placed close to the chips and are designed to decouple high frequency noise from the power supply.

Determining the number of decoupling capacitors depends on the expected current draw from the power supply, the rate of current change, and the desired impedance between the power plane and the ground plane at the frequency of interest. For switching (or sometimes even linear) regulators, the manufacturers specify a certain minimum bulk decoupling at the output of the regulator for stability and for ensuring a clean output.

Bulk Decoupling

The amount of bulk decoupling needed depends on the number of outputs switching, the load capacitance, the slew rate, and the inductance of the planes and routing traces and vias.

Estimate the amount of bulk decoupling needed as follows:

1. Determine the worst case current draw on the power net. This draw depends on how many gates can switch at the same time and how much load they need to drive.

   For example, assume there are 100 3.3-V LVTTL I/O pins switching, each with 11.5 pF loads, and the switch is completed in 1 ns. (These values are fairly typical for Stratix and Stratix GX devices.) The total worst case current required is:

$$\Delta I = NC\frac{\delta V}{\delta t} = 100 \times 11.5\,pF\,\frac{3.3V}{1ns} = 3.8\,Amps$$

2. Determine the maximum noise margin degradation allowed in the logic circuit. This value depends on the standard being used. For LVTTL, the threshold is shown in Table 4–1.

   The noise margin between V_{OH} and V_{IH} is 2.4 – 1.7 = 0.7 V. The margin between V_{OL} and V_{IL} is 0.7 – 0.45 = 0.25 V. Therefore, the worst case noise margin is 0.25 V, the absolute maximum value of noise allowed on the power supply while still guaranteeing successful data transfer. A good design allows for safety margin, so this example specifies 0.1 V as the maximum value of noise permitted. Call this parameter ΔV.

Table 4–1. LVTTL Parameters (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{CCIO}	Output supply voltage		3.0	3.6	V
V_{IH}	High-level input voltage		1.7	4.1	V

Table 4–1. LVTTL Parameters (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{IL}	Low-level input voltage		–0.5	0.7	V
V_{OH}	High-level output voltage	I_{OH} = –4 to –24 mA	2.4		V
V_{OL}	Low-level output voltage	I_{OH} = 4 to 24 mA		0.45	V

3. Using ΔV from step 2 and ΔI from step 1, determine the maximum impedance allowed on the power supply plane. In the example, it equals Z_{max} = ΔV/ΔI = 0.1/3.8 = 26.3 mW.

4. Next, determine the power supply wiring inductance. This inductance depends on the actual routing path of the power supply to the plane and the device pin. This path includes the vias, inductance of the balls to the actual connection on the die, inductance of the regulator output pin, and inductance of the actual routing from the regulator to the via. Approximate the inductance of the via with the following formula:

$$L_{via} = 5.08h[\ln(\frac{4h}{d}) + 1]$$

Assume that the height (h) of the via is 63 mils and that the diameter is 10 mils. Using these values in the above formula, the via inductance is 1.4 nH. These values are typical for vias on the Stratix GX development board for a signal traversing about 2/3 of the board thickness. Divide the via inductance by the number of vias that are in parallel. For 10 vias, the total via inductance is 0.14 nH. Other factors that add to this result are the ball-to-die connection and the inductance of the traces, which are harder to estimate and depend on the exact layout. Inductance of the plane is very small (less than 1 nH) if the physical dimensions are at least 1" by 1." You can lump all these inductances into a single approximation of 5 nH (L_{total} = 5 nH), which represents the worst case inductance on a typical system, assuming wide traces for power distribution.

Given the value for the power supply inductance, you can now estimate the frequency above which decoupling is required. Use this equation:

$$F_{critical} = \frac{Z_{\max}}{2\pi L_{tot}} = \frac{26.3\times10^{-3}}{2\pi 5\times10^{-9}} = 838kHz$$

5. Finally, use this equation to calculate the value of the decoupling capacitor:

$$C_{bypass} = \frac{1}{2\pi F_{critical} Z_{\max}} = \frac{1}{2.\pi.838\times10^{3}\times26.3\times10^{-3}} = 7.25\mu F$$

Consider the example on the Stratix GX development board. There are about 350 LVTTL signals on the Stratix device (on banks 1, 2, 3, 4, 7, and 8). The calculation above was for 100 LVTTL signals. For 350 signals, the required value of bulk decoupling increases by a factor of $(3.5)^2$, or 12.25. Therefore, bulk decoupling equals 7.25 * 12.25 = 89 µF. The development board uses a 100 µF capacitor on the 3.3-V power supply net (3.3V_S_IO) for bulk decoupling for the Stratix device. This is in addition to the bulk decoupling required by the switching regulators and the associated filters at the output. With this level of decoupling, there is good performance.

Local Decoupling

Local decoupling is designed to work at high frequencies. Unfortunately, at high frequencies, discrete capacitors are rarely effective by themselves. There must be a combination of discrete capacitors and plane capacitance. The analysis is quite involved and its accuracy is not easy to verify. So, experimental data is more helpful for creating design guidelines. Altera has conducted several experiments on this topic. Based on these experiments, Altera recommends about one decoupling capacitor per power pin. The results of simulations using Agilent design system (ADS) software determined the values of capacitors required.

The amount of local decoupling is more difficult to determine, because high frequency effects are more pronounced for local decoupling. Altera recommends using one capacitor per power pin. More detailed guidelines for specific situations will be available in the near future. You should not mix values of capacitors, because doing so can create resonant peaks of impedances. In most cases it is best to choose the highest value of the capacitance available in the package required by the designer.

Simulating Decoupling Circuits

The efficacy of a decoupling scheme is indicated by the impedance between the power plane and the ground plane across the frequency of interest. The goal of a power supply decoupling scheme is to achieve the lowest possible impedance across the frequency of interest. If the impedance is low, then the noise on the power rail is shunted to the ground, whereas if the impedance is high, the noise stays on the power rail and can cause the circuitry to malfunction.

In this simulation, Altera placed 25 decoupling capacitors of different values in parallel and measured the impedance from 1 MHz to 10 GHz. The frequency of interest can be different for each application, so focus on the frequency band of interest for your application. Using 25 decoupling capacitors gives a realistic number of capacitors used in a typical decoupling scheme. The behavior of vias, pads, and traces were captured using the multilayer transmission line models of ADS, which considers the dielectric material and its dissipative losses.

Eight different cases, summarized in Table 4–2, were simulated to get a wide range of capacitor ratios.

Table 4–2. Simulation Cases

Case	Number of Capacitors					Waveform Name
	470 pF	1000 pF	0.01 µF	0.1 µF	2.2µF	
1	10	8	4	2	1	Z_double_decade
2	5	5	5	5	5	Z_equal_ratio
3	1	2	4	8	10	Z_reverse_decade
4	25	0	0	0	0	Z_all_470pf_caps
5	0	25	0	0	0	Z_all_1000pf_caps
6	0	0	25	0	0	Z_all_point_01_caps
7	0	0	0	25	0	Z_all_point_1_caps
8	0	0	0	0	25	Z_all_2_point_2_caps

Figure 4–9 shows the simulation setup with only one capacitor (only one capacitor is shown for clarity), including the parasitics, pads, routing trace to the vias, and the vias themselves. For the cases shown in Table 4–2, the model of the capacitor shown in Figure 4–9 was applied repeatedly in parallel.

Each capacitor is modeled as a series RLC circuit. The parasitic resistance (R) and the parasitic inductance (L) were obtained from the data sheet published by the manufacturer of the capacitor. The capacitor pads on the board were modeled as 42-mil by 32-mil square pads. The vias were modeled as circular vias with pad diameters of 20 mils and hole diameters of 10 mils. The routing distance from the capacitor pad to the via was modeled as a trace, 50 mils long and 25 mils wide, equal to the total distance for routing on both sides of the capacitor. These are typical values and were extracted from the Stratix GX development board layout design.

Figure 4–9. Simulation Setup for One Capacitor

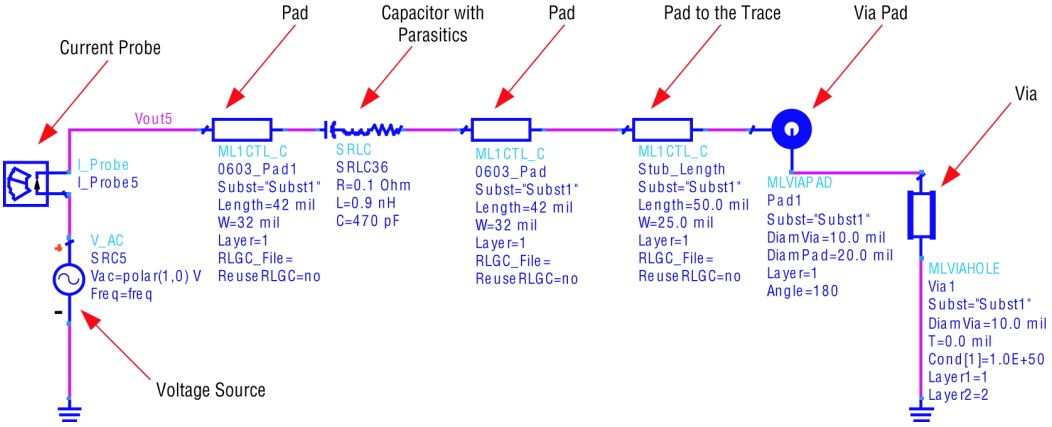


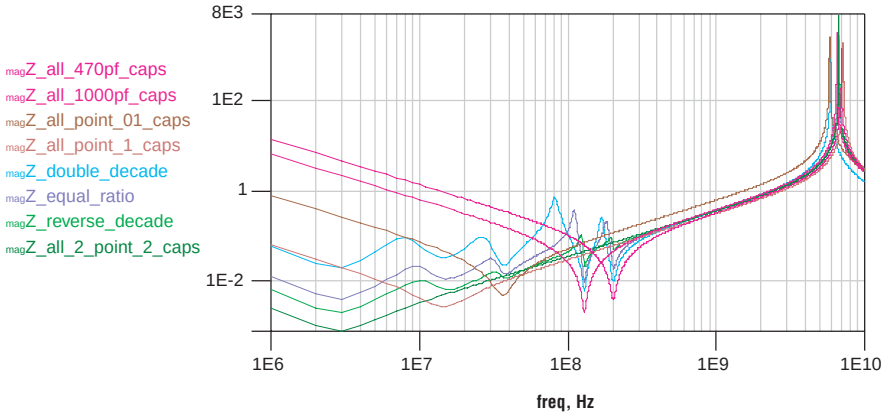
Table 4–3 shows the extracted models of 470 pF, 1000 pF, 0.01 µF, 0.1 µF, and 2.2 µF capacitors from the vendor datasheets (Panasonic and AVX).

Table 4–3. Extracted Models of Capacitor Parasitics

Capacitor Value	Part Number	Vendor	Parasitic L (nH)	Parasitic R (Ω)	SRF (MHz)
0.1 µF	ECJ1VB1C104K	Panasonic	1.50	0.07	13
0.01 µF	ECJ1VB1C103K	Panasonic	0.83	0.10	55
2.2 µF	08056D225KAT2A	AVX/Kyocera	1.00	0.02	3.4
470 pF	ECJ-0EB1E471K	Panasonic	0.86	0.10	250
0.001 µF	ECJ-1VB1H102K	Panasonic	1.13	0.05	150

Figure 4–10 shows the simulated impedance profile for the eight cases listed in Table 4–2.

Figure 4–10. Impedance Simulation Results From Table 4–2



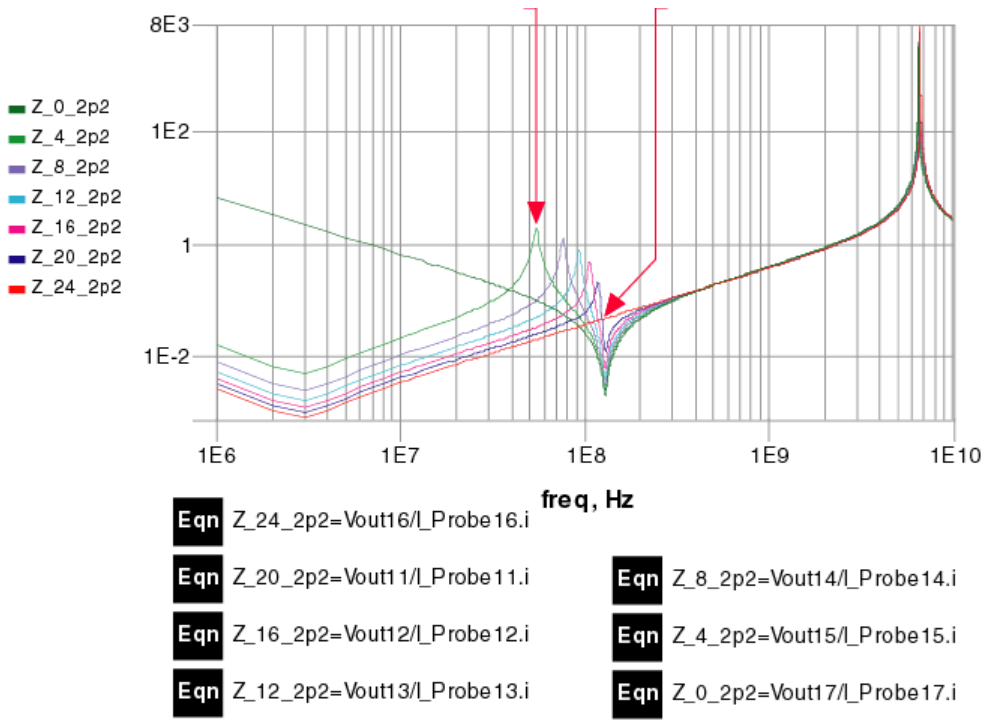
The following observations were made:

- Decoupling at very high frequencies cannot be achieved by discrete capacitors. It must be done with a sandwiched power and ground plane.
- Traditionally, the number of capacitors is doubled for every lower decade in value. For example, beginning with one 0.1 µF capacitor, you would use two 0.01 µF capacitors. Case 1 in Table 4–2 on page 4–17 approximates this behavior. However, this approach often performs worse than when using equal numbers of capacitors of equal values, or even reversing the ratio (number of capacitors halved for each decade lower in value).
- Using a very large number of values can cause resonance spikes at multiple frequencies, so use as few values as possible to contain the magnitude of the resonance spike.

- Using all 2.2 µF capacitors gives the lowest impedance profile without any resonance spikes. Although 1000 pF capacitors (when used in conjunction with 2.2 µF caps) do give the low impedance dip at higher frequencies, they also result in the corresponding resonance spike.

To study how the ratio of 2.2 µF versus 1000 pF capacitors affects the impedance profile, Altera performed further simulations. Figure 4–11 shows the impedance profiles when using all 24 2.2-µF caps versus using a mix of 2.2 µF and 1000 pF. Only 24 capacitors were used in this simulation as opposed to 25, allowing for easy ratios. This difference is not significant. The lower red graph in Figure 4–11 shows the case when all 24 capacitors are 2.2 µF. It has the smoothest impedance profile and is lower in impedance than other cases, except for a dip at 129 MHz. If a particular board has significant noise at the frequency of the dip (approximately 129 MHz), a 1000 pF capacitor is helpful. However, you must ensure that there is negligible noise at the resonant peaks. The resonant peaks occur at 54 MHz, 75 MHz, 92 MHz, 106 MHz, and 118 MHz, depending on the ratio of 1000 pF caps to 2.2 µF caps.

Figure 4–11. Impedance Profile With a Combination of 2.2 µF & 0.001 µF Capacitors



Based on the simulation results, it is important to select the highest value possible in a particular package, because these capacitors have the lowest impedance for the largest band of frequency. Examples include 2.2 µF, 1.0 µF, and 0.1 µF. Do not go lower than that unless there is a compelling reason to do so, such as the presence of a strong noise spike at the exact frequency where the lower value capacitor resonates. Also, do not mix capacitor values (for example using 10 of 0.1 µF and 10 of 0.001 µF). This mixing can create undesired resonances.

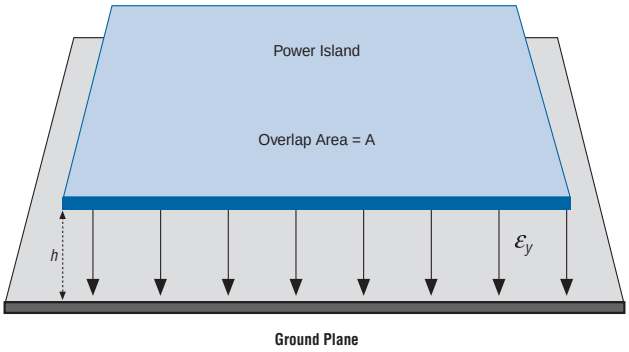
The Stratix GX development board uses many mixed capacitor values for decoupling, not because it is the best approach but because it allows the flexibility to test several approaches and values.

Plane Capacitance

For very high frequencies (above 300 MHz), decoupling using discrete capacitors is less effective. Use power plane capacitance for decoupling noise at these frequencies.

You can understand the concept of plane capacitance by studying the classic parallel plate capacitor, shown in Figure 4–12.

Figure 4–12. Parallel Plane Capacitance



An electric field is created when there is a power plane next to a ground plane. The upper area in Figure 4–12 shows the power island or plane, the lower area shows the ground plane, and the arrows represent the electric field lines. This electric field gives rise to a capacitance, the magnitude of which is shown by:

$$C = \frac{\varepsilon_0 \varepsilon_r A}{h}$$

where

ε_0 = permittivity of free space

ε_r = relative permittivity of the dielectric used

A = area of overlap

h = separation of the planes.

If there are ground planes on both sides of the power island, then the capacitance needs to be calculated for each side and added to determine the total capacitance.

Plane capacitance is the primary way of decoupling at high frequencies, so it must be an integral part of any high speed design. At high frequencies (above 300 MHz), the discrete capacitors are not very effective.

As an example, consider the following.

Example: Determine the parallel plate capacitance for 1 square inch of area overlap in an FR-4 dielectric (ε_r = 4.5) and a separation of 4 mils.

Solution:

h = 4 mils = $1.016 * 10^{-4}$ m

ε_0 = permittivity of free space = $8.85 * 10^{-12}$ F/m

A = 1 sq inch = $6.4516 * 10^{-4}$ m^2

ε_r = 4.5

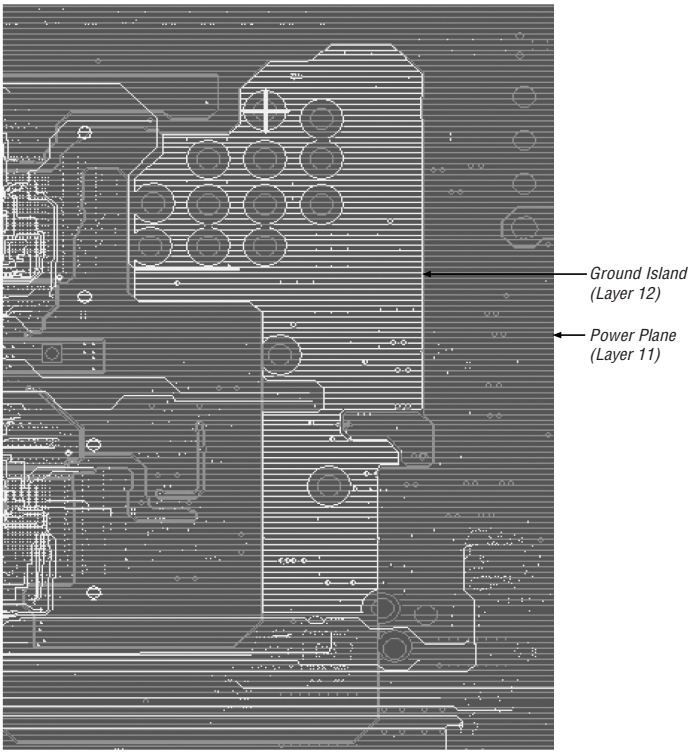
Applying these numbers to the equation on page 4–21 yields C = 253 pF. Therefore, there is about 253 pF per square inch of area overlap on a typical FR-4 board with 4 mils of separation. The value scales inverse linearly with separation and linearly with area.

Altera has successfully used plane capacitance in several of its boards.

Case Study: Altera designed two boards with the Stratix GX test chip. On the first board, the separation between the ground and the 1.5-V transceiver power plane was about 16 mils. On the second board, ground

islands were added (with approximate areas of 0.13 square inches) next to the 1.5 V transceiver plane. The islands were on the next layer and the separation was 4 mils. Figure 4–13 shows the islands as they appear on the board layout.

Figure 4–13. Ground Island Used With the 1.5V_XCVR Plane to Increase Decoupling at High Frequencies



Figures 4–14 and 4–15 show the reduction in noise after adding the islands. The addition of the islands reduced the noise from 70 mV p-p to less than 50 mV p-p under the same experimental conditions.

Figure 4–14. Noise on the 1.5V_XCVR Plane With the Ground Plane 16 mils Away

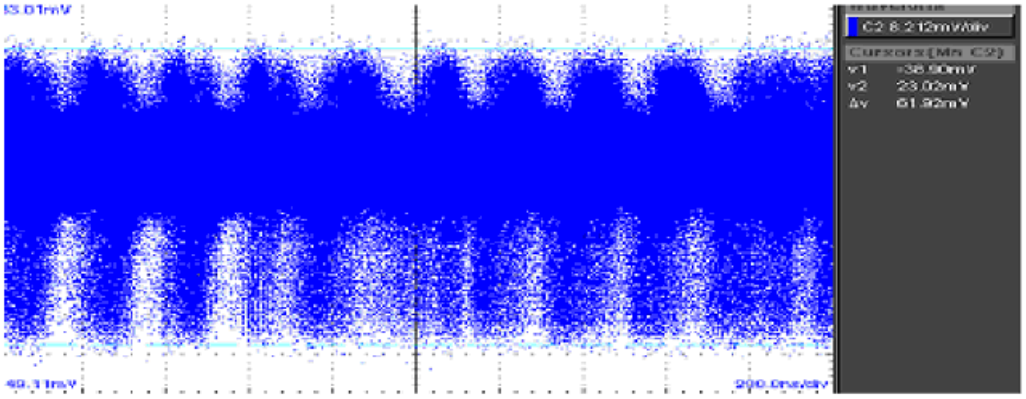
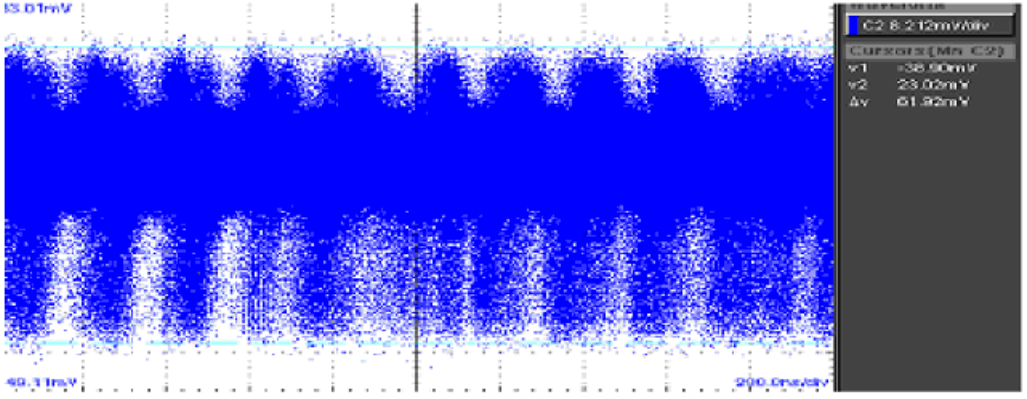


Figure 4–15. Noise on the 1.5V_XCVR Plane With the 0.13 Square Inch Island 4 mils Away



When using split ground planes, it is best to pair the correct ground plane with the correct power plane. For example, 1.5V_XCVR and GND_GXB planes and islands should be used rather than 1.5V_XCVR and GND.

Plane & Island Design

The following sections discuss various issues related to the design of power planes, ground planes, and power and ground islands.

Power Plane & Island Design

Stratix GX devices have many power and ground pins, and it is important to know how to connect them on the board. Table 4–4 shows the list of power and ground pins and how they connect on the Stratix GX development board. For the actual pin numbers, refer to the pin tables at **www.altera.com**.

Table 4–4. Stratix GX Power & Ground Pins (Part 1 of 2)

Pin Name	Description	Voltage
VCCP_B[17..13]	Digital power for Stratix GX transceiver quads (banks) 13 through 17	1.5 V (linear)
VCCT_B[17..13]	Transmitter power for Stratix GX transceiver quads (banks) 13 through 17	1.5 V (linear)
VCCR_B[17..13]	Receiver power for Stratix GX transceiver quads (banks) 13 through 17	1.5 V (linear)
VCCG_B[17..13]	Power for Stratix GX transceiver quads (banks) 13 through 17 guard rings.	1.5 V (linear)
VCCA_B[17..13]	Analog power for Stratix GX transceiver quads (banks) 13 through 17.	3.3 V (linear)
VCCG_PLL[12,11,8,6,5,2,1]	Power supply for the guard ring of PLLs	1.5 V (linear)
VCCA_PLL[12,11,8,6,5,2,1]	Analog power supply for PLLs	1.5 V (linear)
VCC_PLL5_OUTA	PLL output buffer supply for PLL5 outputs [1..0]	3.3 V, 2.5 V or 1.5 V *(1)* (linear)
VCC_PLL5_OUTB	PLL output buffer supply for PLL5 outputs [3..2]	3.3 V, 2.5 V or 1.5 V *(1)* (linear)
VCC_PLL6_OUTA	PLL output buffer supply for PLL6 outputs [1..0]	3.3 V, 2.5 V or 1.5 V *(1)* (linear)
VCC_PLL6_OUTB	PLL output buffer supply for PLL6 outputs [3..2]	3.3 V, 2.5 V or 1.5 V *(1)* (linear)
VCCINT	Digital power supply for device internal power	1.5 V (linear or switching)
VCCIO[8..7, 4..1]	Power supply for I/Os in the banks 8, 7, 4..1	Variable *(2)* (linear or switching)

Table 4–4. Stratix GX Power & Ground Pins (Part 2 of 2)		
Pin Name	**Description**	**Voltage**
GNDG_PLL[12,11,8,5,2,1]	Ground for the guard rings of PLLs 1, 2, 5, 6, 8, 11, and 12	0 V
GNDA_PLL[12,11,8,6,5,2,1]	Analog ground for PLLs 1, 2, 5, 6, 8, 11, and 12	0 V
GND_GXB	Ground for the high-speed circuitry in the 3.125 Gbps transceivers	0 V
GND	General-purpose ground	0 V

Notes for Table 4–4:

(1) Connect this pin to the desired voltage depending on the I/O standard for the PLL outputs chosen. For example, in the Stratix GX development board, VCC_PLL6_OUTA and VCC_PLL6_OUTB connect to 2.5 V to allow an SSTL-II clock (2.5V I/O) on the PLL6 output for the double data rate (DDR) memory application.

(2) Connect this pin to the I/O standard of the signals on the bank. This value is 3.3 V for LVTTL, 2.5 V for SSTL-II, and so on. On the Stratix GX development board, VCCIO7 connects to 2.5 V because the board uses the SSTL_II I/O standard on that bank. On other banks, it connects to 3.3 V.

Table 4–4 also shows the recommended regulator (linear or switching). The circuitry pertaining to PLLs and transceivers should be connected to voltages generated from linear power supplies. General purpose I/O and core supplies can be switching or linear. It is always better to use linear if possible, but for some I/Os and cores, the current draw might be too large.

The transceivers have many types of power supply pins. Altera recommends that you isolate the receive (VCCR_B[17..13]) and transmit (VCCT_B[17..13]) power supplies of each quad with a ferrite bead. The reason behind the isolation is to prevent noise from one quad leaking to the other quads. The digital power supply (VCCP_B[17..13]) can be shared among all the quads because it is less sensitive to noise. The guard power supply VCCG_B[17..13] can also be shared among quads. Use the islands shown in Table 4–5:

Table 4–5. Islands Used for Stratix GX Transceiver Quads (Part 1 of 2)	
Islands	**Quads**
Island 1	Use for quad 1 VCCT and VCCR pins (VCCT_B13 and VCCR_B13)
Island 2	Use for quad 2 VCCT and VCCR pins (VCCT_B14 and VCCR_B14)
Island 3	Use for quad 3 VCCT and VCCR pins (VCCT_B15 and VCCR_B15)

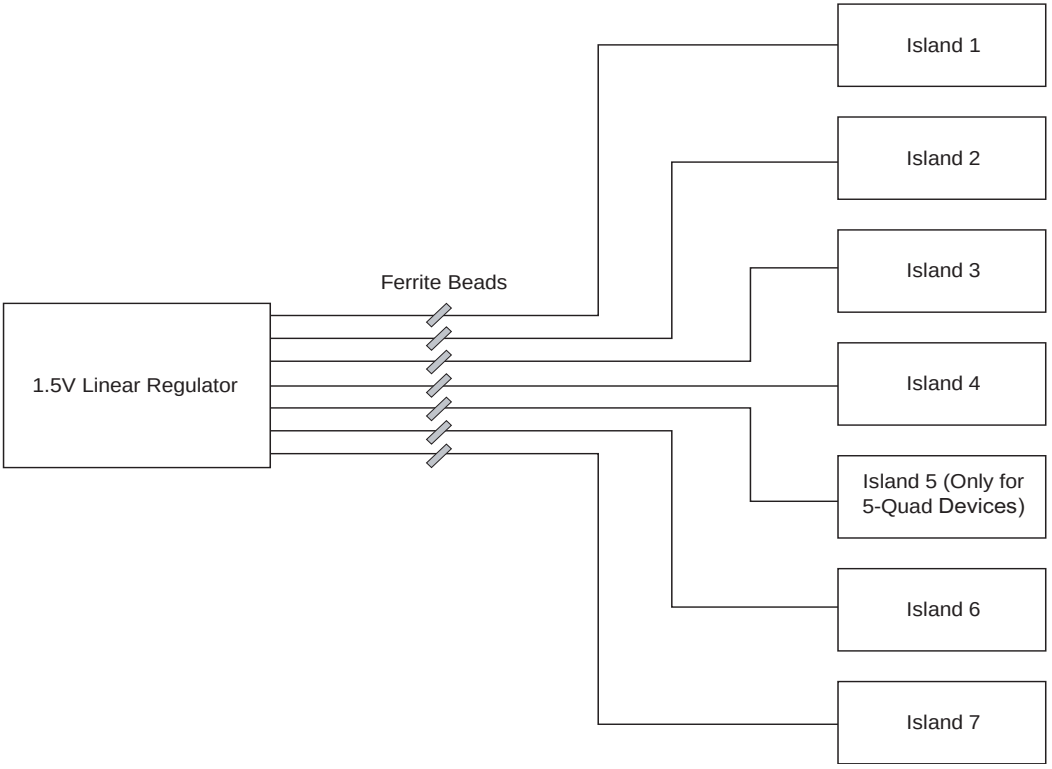
Table 4–5. Islands Used for Stratix GX Transceiver Quads (Part 2 of 2)

Islands	Quads
Island 4	Use for quad 4 VCCT and VCCR pins (VCCT_B16 and VCCR_B16)
Island 5	Use for quad 5 VCCT and VCCR pins (VCCT_B17 and VCCR_B17)
Island 6	Use for `VCCP_B[17..13]`, for example, digital supply for all quads
Island 7	Use for `VCCG_B[17..13]`, for example, guard supply for all quads

Note to Table 4–5:
(1) Island 5 is required only for the 5-quad devices.

Figure 4–16 shows a block diagram of the islands.

Figure 4–16. Block Diagram Representation for Quad Isolation



Note to Figure 4–16:
(1) Island 5 is required only for the 5-quad devices.

Altera has successfully used the island approach to powering the Stratix GX transceivers on the Stratix GX development board. Figure 4–17 shows the section of the schematic that shows the islands. The net named 1.5V_XCVR is the output of a linear regulator that is split into seven islands, namely: 1.5V_XCVR1, 1.5V_XCVR2, 1.5V_XCVR3, 1.5V_XCVR4, 1.5V_XCVR5, 1.5V_VCCP, and 1.5V_VCCG.

Altera is currently evaluating the degree to which the islands help in reducing jitter.

Figure 4–17. Section of Schematic Showing the Islands for the Transceiver Quads

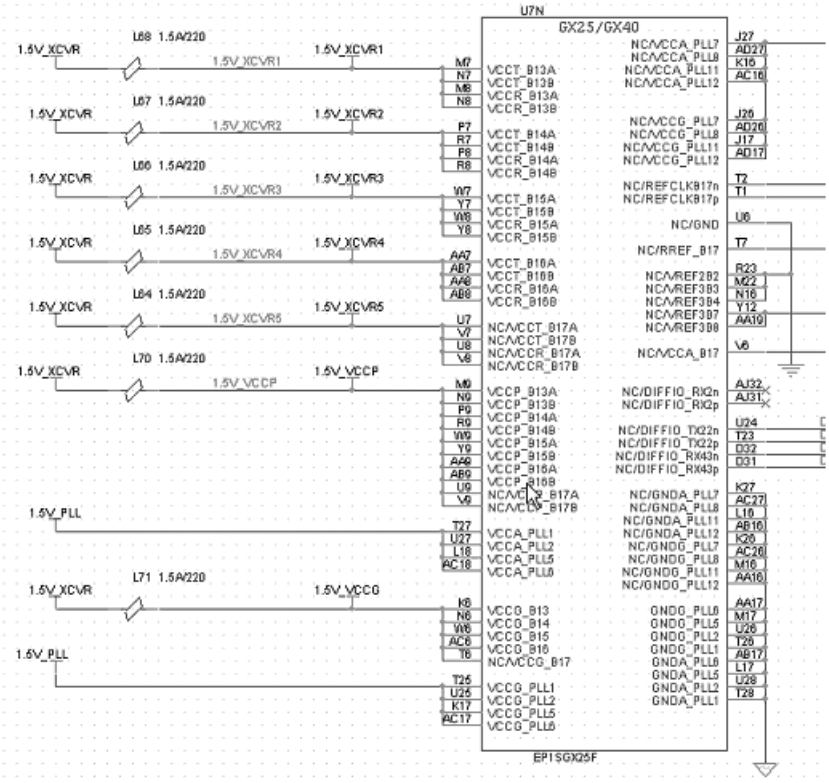
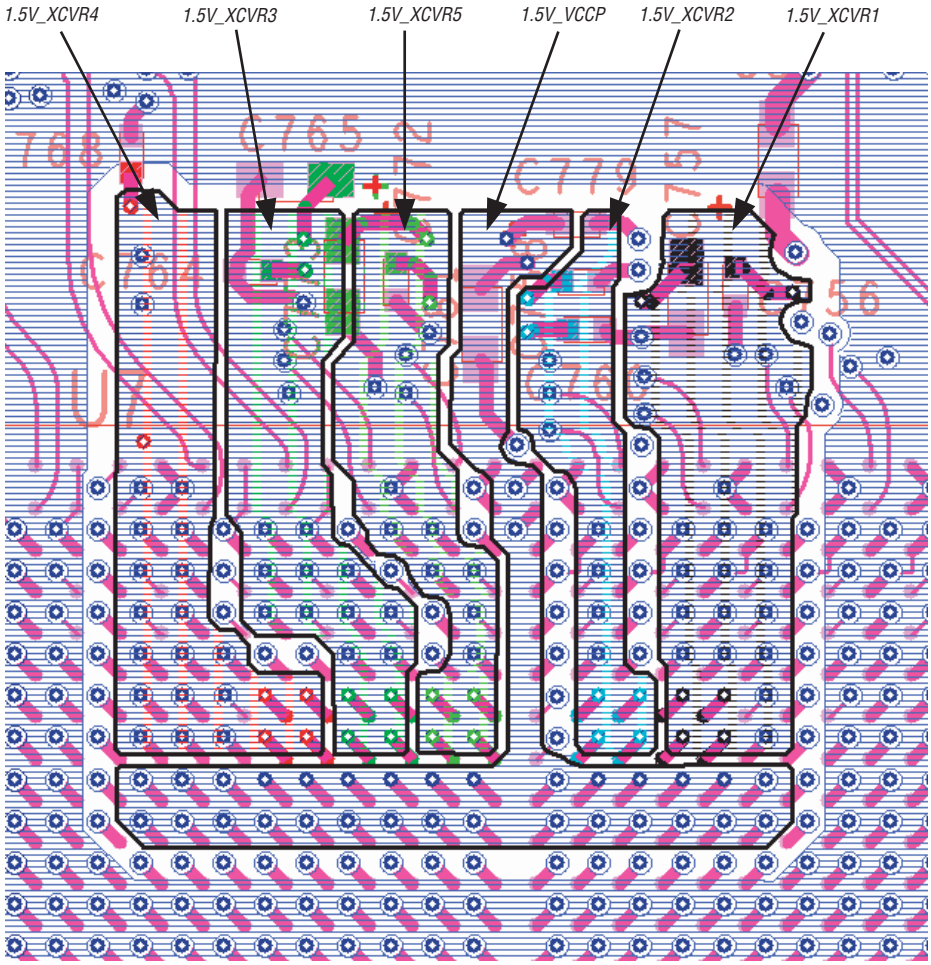


Figure 4–18 shows the layout of six of the islands. The 1.5V_VCCG (guard) island is on a different layer because of layout constraints.

Figure 4–18. Layout of the Islands for the Transceiver Quads



Ground Plane & Island Design

The Stratix GX development board has three types of ground pins: GND, GND_GXB and GND_PLL. GND_GXB is used for 3.125 Gbps transceiver ground, GND_PLL is used for PLL ground, and GND is used for general purpose ground on the board. In the die, GND_GXB and GND are separate planes, but they are connected by a trace at the package level. GND_PLL does not have a separate plane and is connected to the GND plane at the package level.

When designing your board you must decide whether to connect all the ground pins to a single ground plane on the board or create separate ground planes for GND_GXB, GND, and GND_PLL and connect them by using ferrite beads. This is a difficult trade-off, and there are two schools of thought in the industry regarding splitting ground planes.

Proponents of isolating ground planes say that it is good for systems that can generate a lot of noise because of switching I/Os or power supplies. The idea is that the ferrite beads attenuate the noise leaking from the system ground (which can be noisy) to the clean ground. Noise-sensitive analog subsystems like the transceivers require clean power and ground, and the isolation provides better performance. The ferrite beads must be carefully selected to allow for plenty of current handling, low DC voltage loss, and high AC impedance. See "Resistors, Capacitors, Inductors & Ferrite Beads" on page 4–77 for more details.

Supporters of the solid ground plane approach claim that there is no benefit to isolating ground planes, because noise cannot simply disappear from the ground plane and reaches the analog circuitry by some path or other. The layout is also complicated by having isolated ground planes because more layers might be needed. If the same plane is split into two, instead of adding a new layer, the layout designer must ensure that no critical signals cross over the split, because the split causes impedance discontinuity.

There is no one answer on this topic that applies to all systems. If the system is relatively simple without too many fast switching I/Os, and if there are no switching regulators, a solid ground plane is unlikely to cause problems. However, if the system has many fast, high-amplitude digital signals switching and noisy external components such as switching regulators, Altera recommends isolating ground planes.

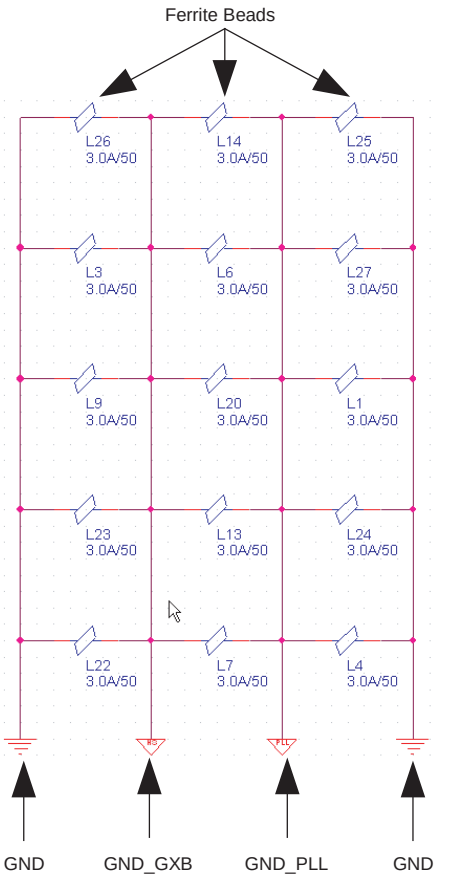
The Stratix GX development board uses an isolated ground approach. There are three distinct ground nets: GND, GND_PLL, and GND_GXB, separated by several ferrite beads. Figure 4–19 shows a section of the Stratix GX development board schematic highlighting the isolation of the ground planes. Many ferrite beads were used in parallel to facilitate testing with several possible combinations.

Altera tested the performance of this system with the ferrite beads intact, as shown in Figure 4–19, and with them shorted to a wire. The Stratix GX EP1SGX40 device had about 72% of its LEs used with a design that had Gray code counters. The Stratix GX device had a DDR interface running at 200 MHz and 17 channels of SPI-4.2 data running at 1 Gbps per channel. The LE usage was about 46%. The Stratix GX device also had 19 channels of transceivers running at PRBS 2^5 - 1 data at 3.125 Gbps in an external loopback. The transmitter output of the 20th channel was sent to

an oscilloscope to observe the jitter and the eye diagram. This setup should result in a fairly heavy load for the devices and should generate a good deal of noise on the power planes. The test setup was identical for both tests, except for the shorted ferrite beads.

The tests showed a jitter improvement of approximately 10% when isolating the ground planes. If the devices are loaded even more fully, the improvement is even greater. If the devices are lightly loaded, the improvement is less.

Figure 4–19. Isolation of GND, GND_GXB & GND_PLL



Based on the test results and the benefit of isolating ground planes for noisy boards, Altera® recommends isolating the GND_GXB and GND planes with ferrite beads. To determine how many ferrite beads to use,

consider the following. The power consumption per quad is about 450 mW, so the current draw is about 450 mW/1.5 = 300 mA. Therefore, the total expected current draw is about 1.5 Amps. Each ferrite bead has 25 mW of DC resistance. So, if there is one ferrite bead, the voltage drop is 37.5 mV, which is 2.5% of the 1.5-V supply. To reduce this drop, use multiple ferrite beads in parallel. On the Stratix GX development board, there are five beads in parallel, so that the voltage drop is reduced to 0.5% of the supply voltage.

For the GND_PLL plane, it is not crucial to use a separate ground plane. At the device package level, this plane shares the ground plane with the general-purpose logic ground. There is a separate plane on the development board for GND_PLL for testing flexibility.

Altera recommends two ground planes, GND and GND_GXB, isolated by ferrite beads. These planes are especially important for complex boards with many noisy signals and switching regulators. If the board does not have switching regulators or switching signals, and it is otherwise very cleanly designed, one ground plane is acceptable. Altera recommends connecting the PLL ground pins directly to the system GND.

Transmission Lines

This section discusses transmission line designs for high-speed digital boards.

Transmission Line Topologies

This section provides a brief overview of the main transmission line topologies commonly used in high-speed digital boards. Refer to any transmission line design book, such as Brian Wadell's *Transmission Line Design Handbook*, for impedance equations and design techniques.

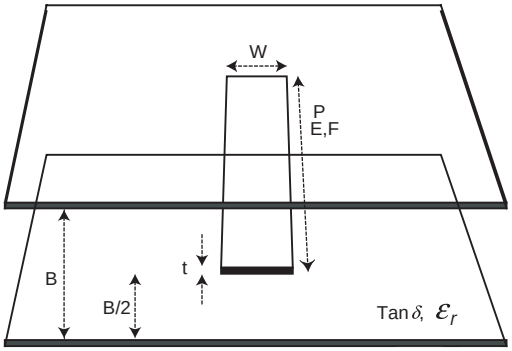
Striplines

The following sections detail the variations on stripline design topologies.

Simple Stripline

Simple stripline is a planar type transmission line well suited for multilayer PCB design. Figure 4–20 shows the basic structure of this transmission line. A thin, centered conductive strip with width W placed between two conductive planes separated by a dielectric with thickness B is the most basic stripline structure.

Figure 4–20. Simple Stripline



Figures 4–21 and 4–22 show the frequency domain simulation setup and results for a 50 Ω simple stripline, respectively.

Figure 4–21. Simulation Setup for a Simple Stripline

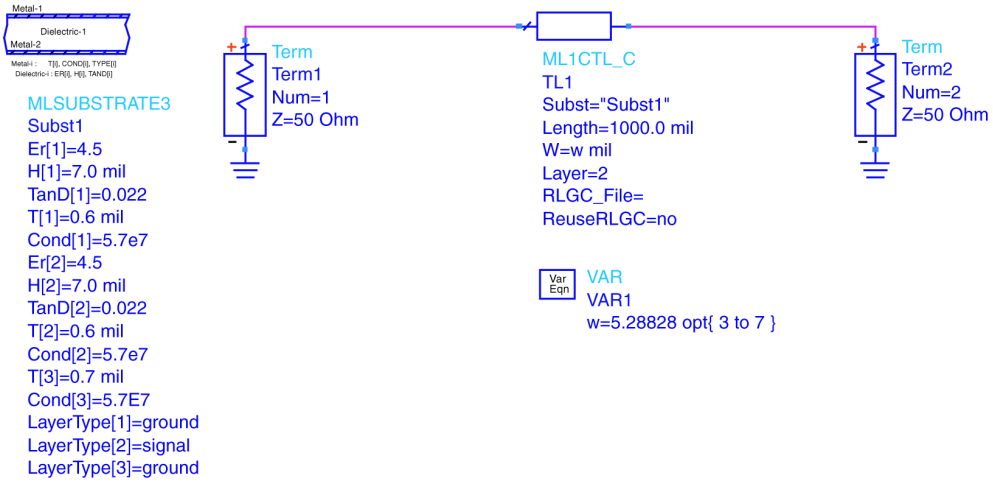
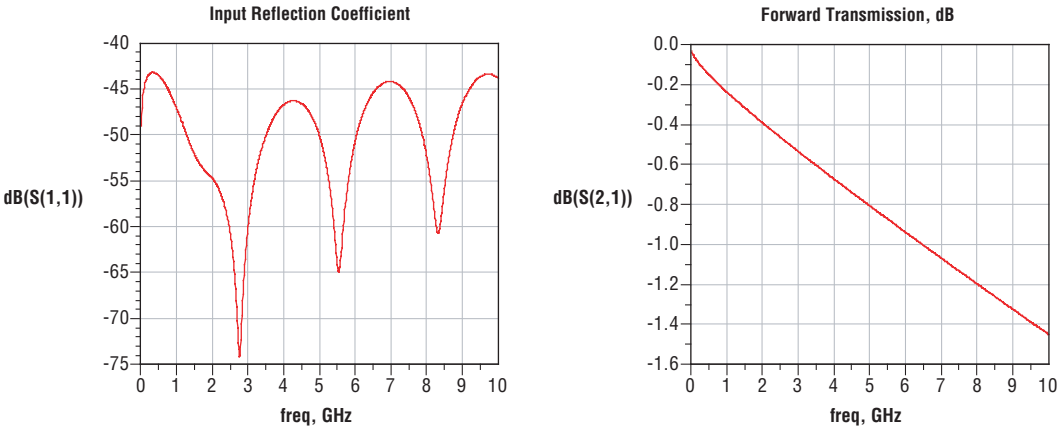


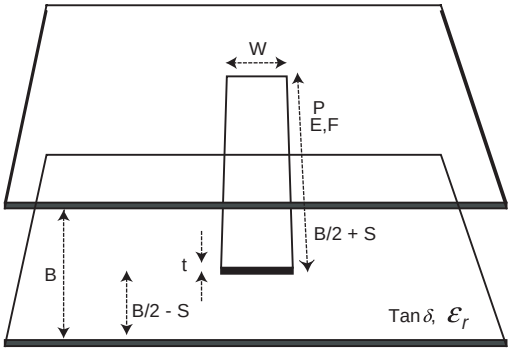
Figure 4–22. Simulation Results for a Simple Stripline



Offset Stripline

The offset stripline configuration is shown in Figure 4–23. The only difference between the structure of the offset stripline and the normal stripline is that the conductor strip is not placed at the center (B/2 distance from any of the planes). The relative position of the conductor strip is S from B/2. This transmission line is useful for single-ended signals routed between a ground plane and a power plane. The strip is placed closer to the ground plane, which is less noisy than the power plane. In this way there is less noise power coupled to the signal and an improved S/N ratio without the need for an extra ground plane. You must use this structure when the dielectric between the planes is constructed of three layers of dielectrics (two cores, one prepeg, or vice versa). In the Stratix GX development board there are two cores and one prepeg. This arrangement of dielectrics produces better results because the cores have lower permittivity and thickness tolerances than prepegs.

Figure 4–23. Offset Stripline



Figures 4–24 and 4–25 show the frequency domain simulation setup and results for a 50 Ω offset stripline, respectively.

Figure 4–24. Simulation Setup for an Offset Stripline

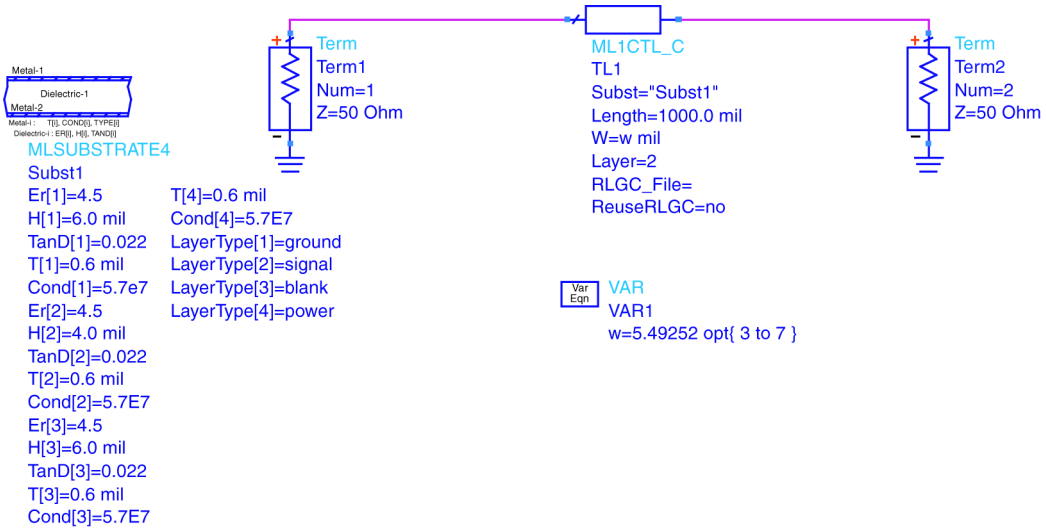
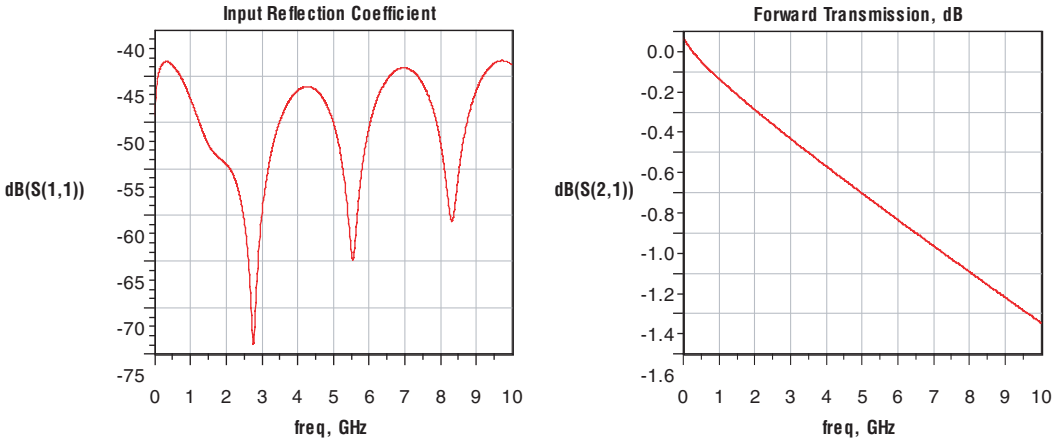


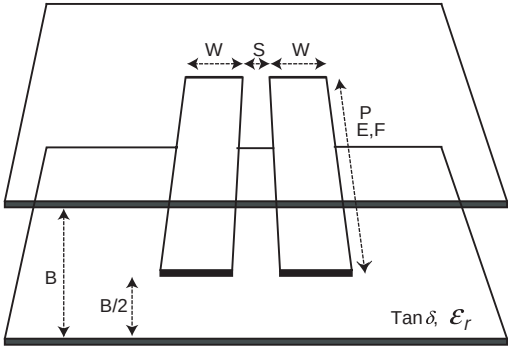
Figure 4–25. Simulation Results for an Offset Stripline



Edge-Coupled Differential Stripline

The edge-coupled differential stripline is shown in Figure 4–26. Altera recommends using differential transmission lines in noisy environments where the common mode noise is suppressed in the receiver. This type of transmission line requires some degree of coupling between the two conductive strips. The coupling is directly proportional to the distance S. The differential transmission lines have two operation modes. Even mode, in which both traces are excited with equal amplitude and phase, and odd mode, in which the traces are excited with equal amplitude but one is 180° out of phase. This structure must be analyzed as one unit and with the proper excitation. For differential signaling, the odd mode is the desired operation mode for canceling common mode noise.

Figure 4–26. Edge-Coupled Differential Stripline



Figures 4–27 and 4–28 show the frequency domain simulation setup and results for a 100-Ω edge-coupled differential stripline, respectively.

The differential impedance is:

$$Z_{DIFF} = 2\ (Zoo)$$

If you excite one port and terminate all others (three) with its characteristic impedance *Zo* (50 Ω) the input impedance is:

$$Zo = \sqrt{Zoe * Zoo}$$

where:

Z_{oe} is the even mode characteristic impedance

Z_{oo} is the odd mode characteristic impedance

Z_{DIFF} is the differential characteristic impedance

Z_o is the characteristic impedance

Figure 4–27. Simulation Setup for an Edge-Coupled Differential Stripline

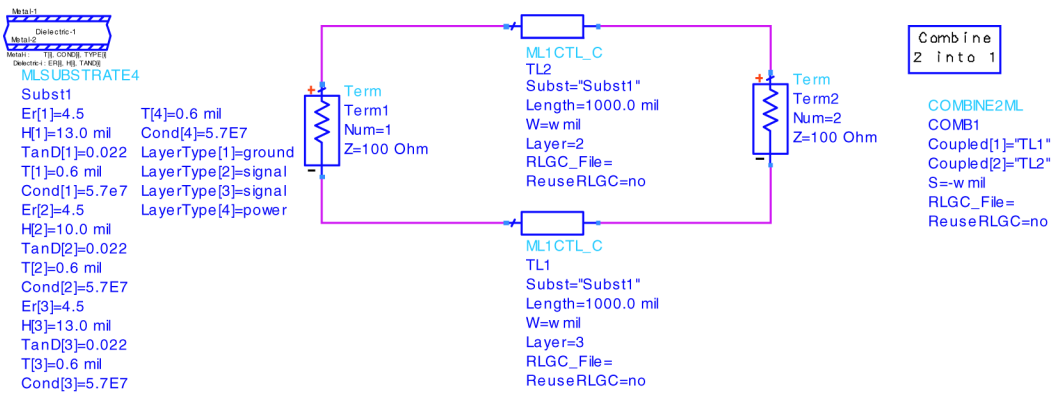
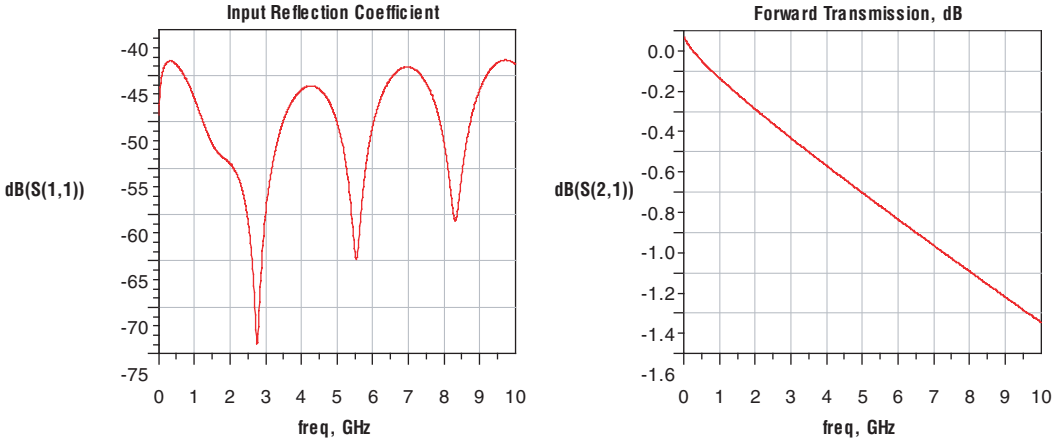


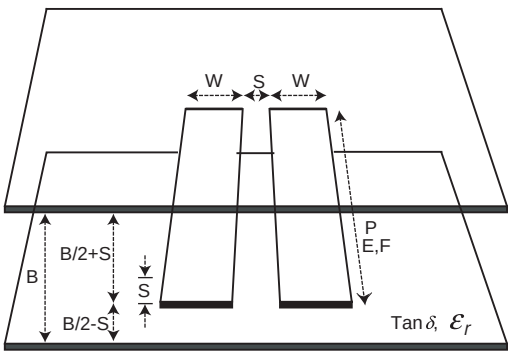
Figure 4–28. Simulation Results for an Edge-Coupled Differential Stripline



Edge-Coupled Differential Offset Stripline

The edge-coupled differential offset stripline topology, shown in Figure 4–29, is used when an odd number of dielectric layers exists between the planes. The signals in this dielectric arrangement are routed closer to one of the planes. This transmission line offers all the advantages of the differential transmission lines, plus the flexibility to use wider traces without the penalty on the total board thickness.

Figure 4–29. Edge-Coupled Differential Offset Stripline



Figures 4–30 and 4–31 show the frequency domain simulation setup and results for a 100-Ω edge-coupled differential offset stripline, respectively. The Stratix GX development board contains many edge-coupled

differential striplines. Examples from the board are discussed in "Crosstalk" on page 4–72. This topology is also known as *dual stripline*, because there are two signal layers between the two power planes.

Figure 4–30. Simulation Setup for an Edge-Coupled Differential Offset Stripline

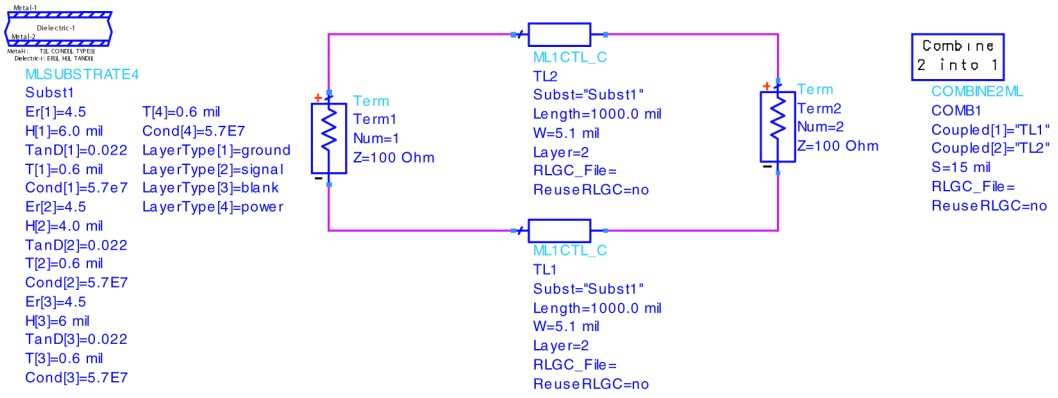
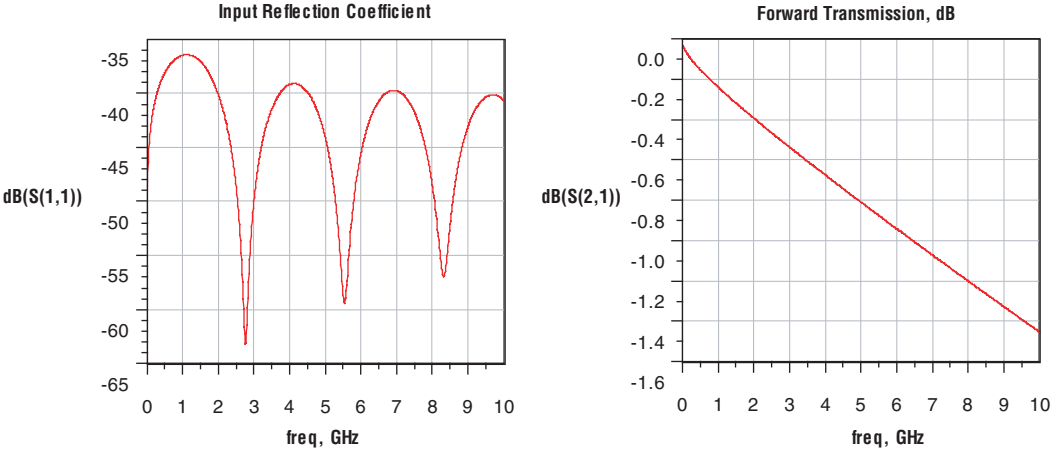


Figure 4–31. Simulation Results for an Edge-Coupled Differential Offset Stripline



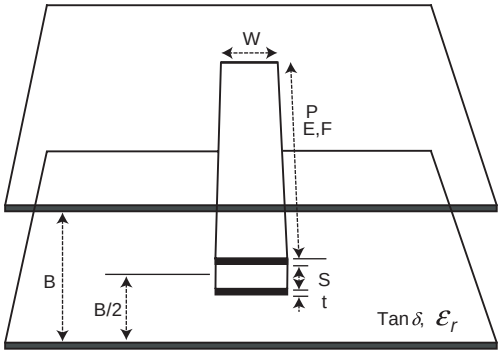
Broadside-Coupled Differential Stripline

Broadside-coupled differential stripline, shown in Figure 4–32, has some advantages over the edge-coupled differential stripline, including higher coupling and easier routing. The disadvantages are the possibility of the

traces becoming too narrow for the manufacturability of the board and the board becoming too thick to compensate for the trace-width thickness ratio.

This configuration is helpful when differential high-speed lines reside in the inner rows of the BGA, and it is impossible to route two lines (one differential pair) between vias or pads.

Figure 4–32. Broadside-Coupled Differential Stripline



Figures 4–33 and 4–34 show the frequency domain simulation setup and results for a 100-Ω broadside-coupled differential stripline, respectively.

Figure 4–33. Simulation Setup for a Broadside-Coupled Differential Stripline

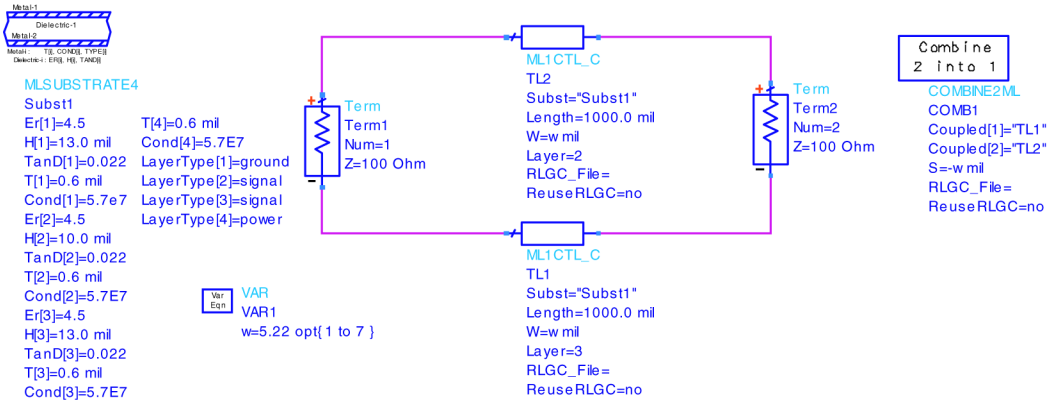
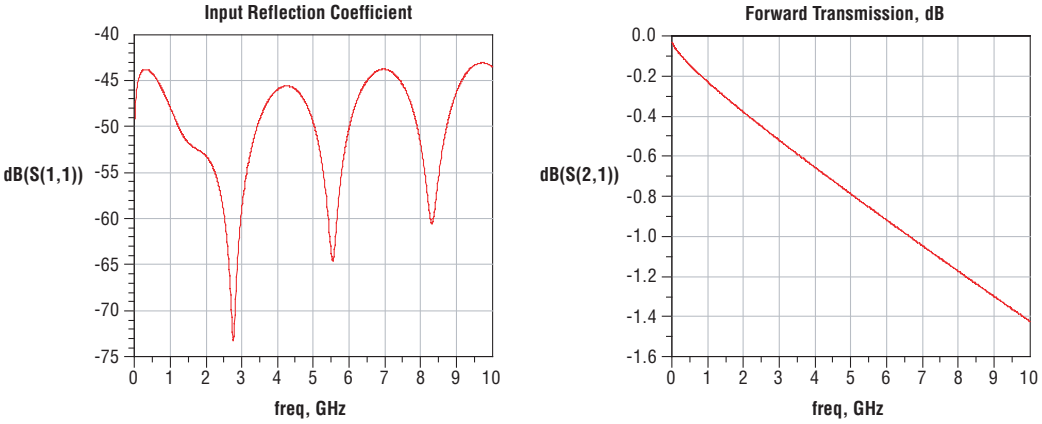


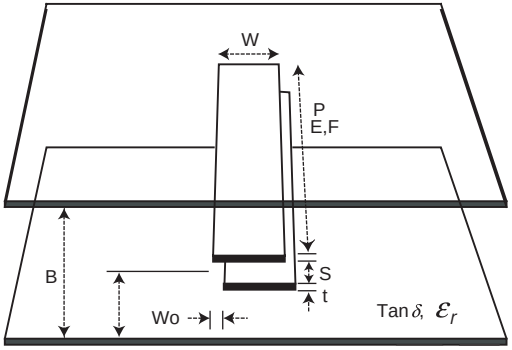
Figure 4–34. Simulation Results for a Broadside-Coupled Differential Stripline



Broadside-Coupled Differential Offset Stripline

The broadside-coupled differential offset stripline structure, shown in Figure 4–35, has the advantages of the broadside-coupled differential stripline with the added flexibility that allows you to adjust the differential impedance by offsetting the two conductor strips. The differential impedance increases as the offset dimension *Wo* increases.

Figure 4–35. Broadside-Coupled Differential Offset Stripline



Figures 4–36 and 4–37 show the frequency domain simulation setup and results for a 100-Ω broadside-coupled differential offset stripline, respectively.

The Stratix GX development board uses edge coupling instead of broadside coupling, with loose coupling between the positive and negative traces. Although tight coupling provides greater noise immunity, it is necessary to loosen the coupling because of the mechanical placement requirements of SMA connectors. With loose coupling, if the routing is broadside coupled, the thickness of the board increases. With loose coupling, edge-coupled routing is preferred unless there is the flexibility to increase board thickness.

Figure 4–36. Simulation Setup for a Broadside-Coupled Differential Offset Stripline

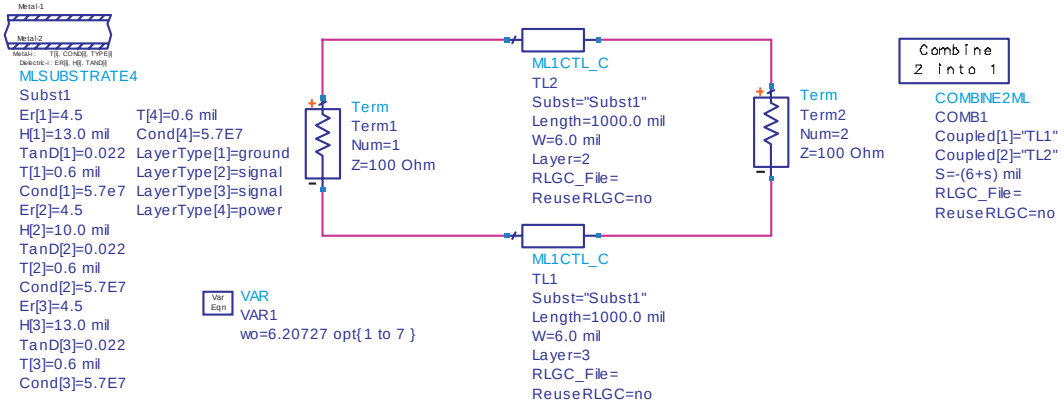
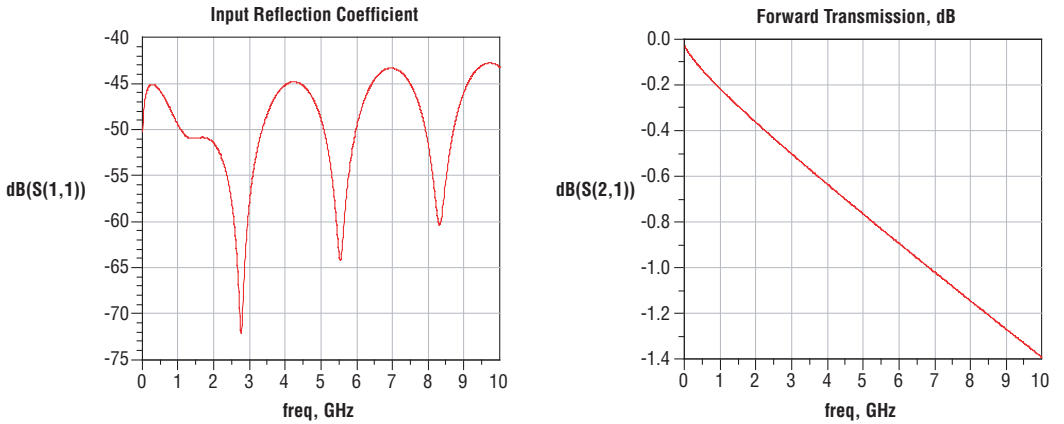


Figure 4–37. Simulation Results for a Broadside-Coupled Differential Offset Stripline



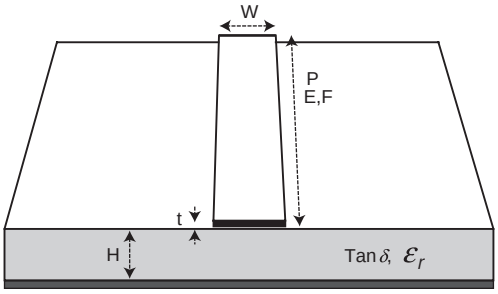
Microstrips

Microstrip topologies can be either simple or differential. Microstrips are the most popular planar transmission lines because of their low cost and easy integration with other passive and active components on PCBs.

Simple Microstrip

Figure 4–38 shows the simple microstrip topology. A microstrip is a conductive strip of width W and thickness t, placed on top of a dielectric material backed with a ground plane.

Figure 4–38. Simple Microstrip



Figures 4–39 and 4–40 show the frequency domain simulation setup and results for a 50-W simple microstrip, respectively.

Figure 4–39. Simulation Setup for a Simple Microstrip

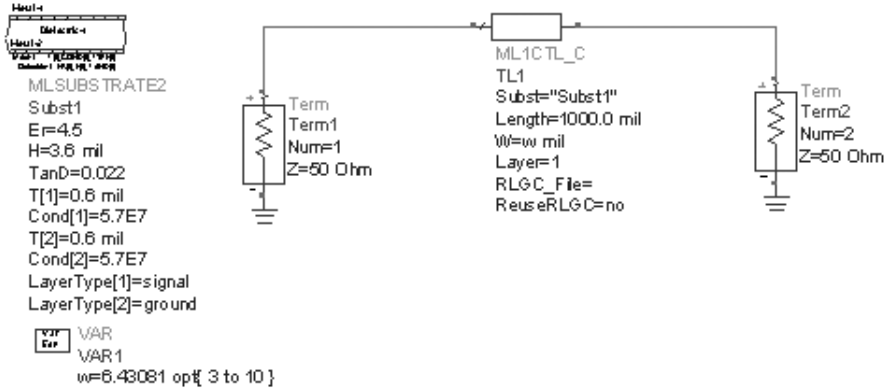
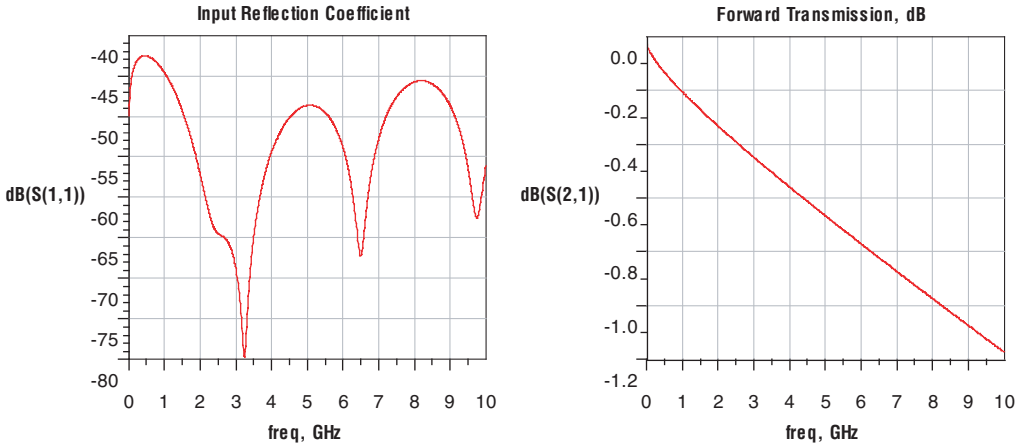


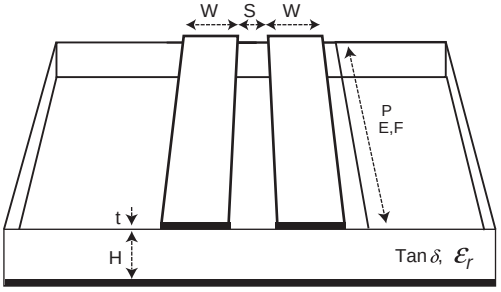
Figure 4–40. Simulation Results for a Simple Microstrip



Differential Microstrip

The differential microstrip topology, shown in Figure 4–41, has the advantages of the simple microstrip, but also provides common mode noise cancellation. The main problem with differential microstrips is that the coupling between the positive and negative traces is low.

Figure 4–41. Differential Microstrip



Figures 4–42 and 4–43 show two variants of the differential microstrip extracted from the Stratix GX development board.

Figure 4–42. Loosely Coupled Differential Microstrip

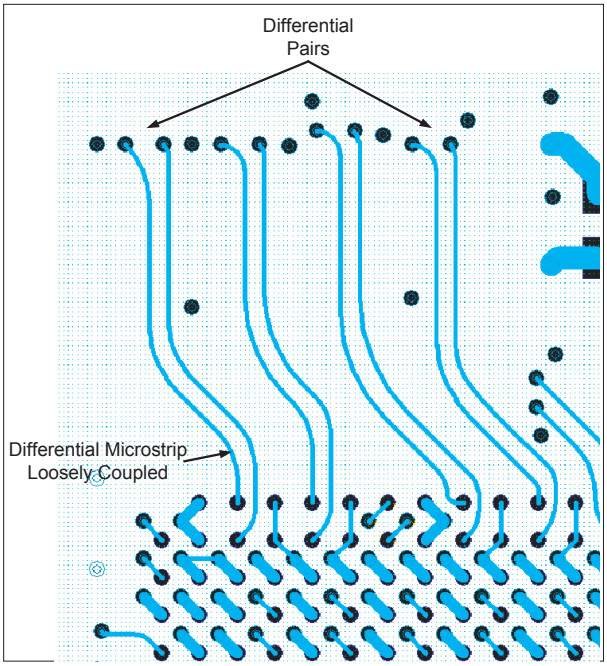
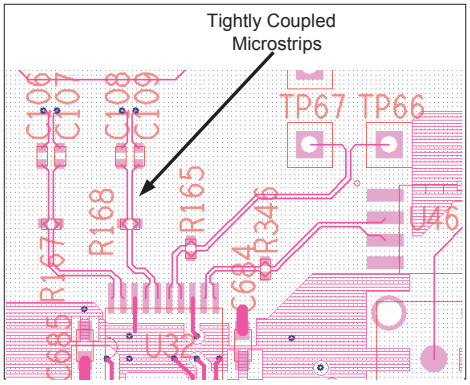


Figure 4–43. Tightly Coupled Differential Microstrip



Figures 4–44 and 4–45 show the frequency domain simulation setup and results for a 100-Ω tightly coupled differential microstrip, respectively.

Figure 4–44. Simulation Setup for a Tightly Coupled Differential Microstrip

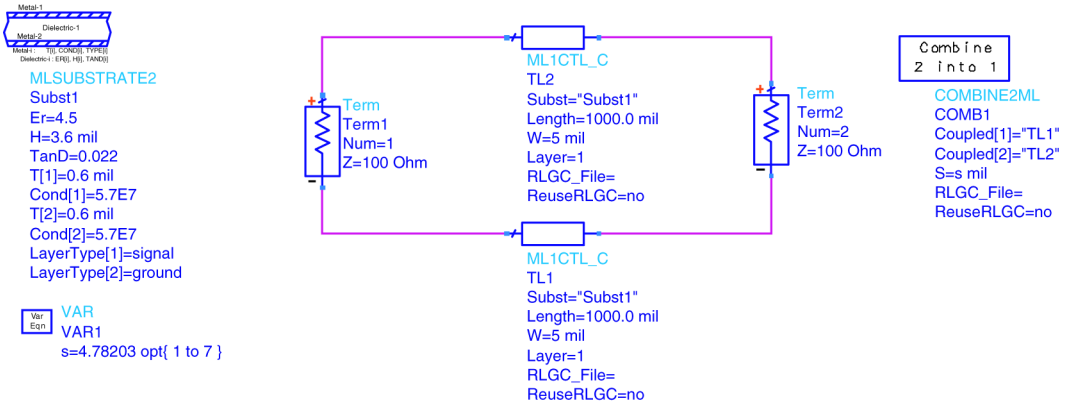
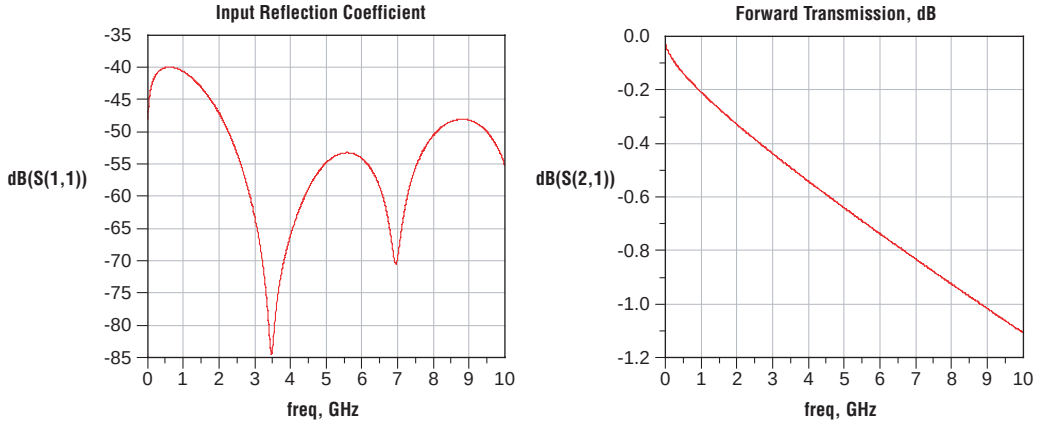


Figure 4–45. Simulation Results for a Tightly Coupled Differential Microstrip



Coplanar Wave Guides

There are three types of coplanar wave guides: simple, grounded, and grounded differential.

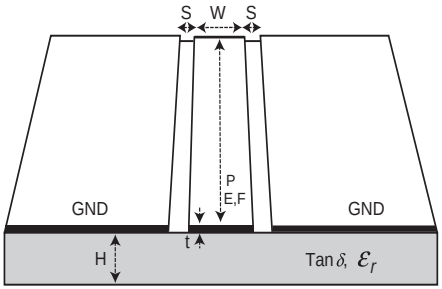
Simple Coplanar Wave Guide

The simple coplanar wave guide, shown in Figure 4–46, is used primarily in microwave systems. This structure does not require vias, and you can easily mount passive or active devices in the signal path, resulting in a

low-loss, high-speed transmission line. The simple coplanar wave guide requires that the substrate thickness (H) be "infinite," so that the fields remain outside the dielectric.

You cannot use this structure on multilayer boards because it cannot have a second layer.

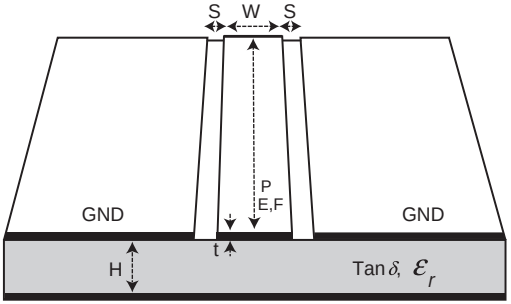
Figure 4–46. Simple Coplanar Wave Guide



Grounded Coplanar Wave Guide

The grounded coplanar wave guide is used extensively in communications systems that integrate different technologies such as surface mount technology, multichip modules, and multilayer boards. The dielectric thickness (H) must be at least five times the spacing (S) to be considered a grounded coplanar wave guide; otherwise, it is considered a microstrip with a ground shield. Figure 4–47 shows the grounded coplanar wave guide topology.

Figure 4–47. Grounded Coplanar Wave Guide



Figures 4–48 and 4–49 show the frequency domain simulation setup and results for a 50-Ω grounded coplanar wave guide, respectively.

Figure 4–48. Simulation Setup for a Grounded Coplanar Wave Guide

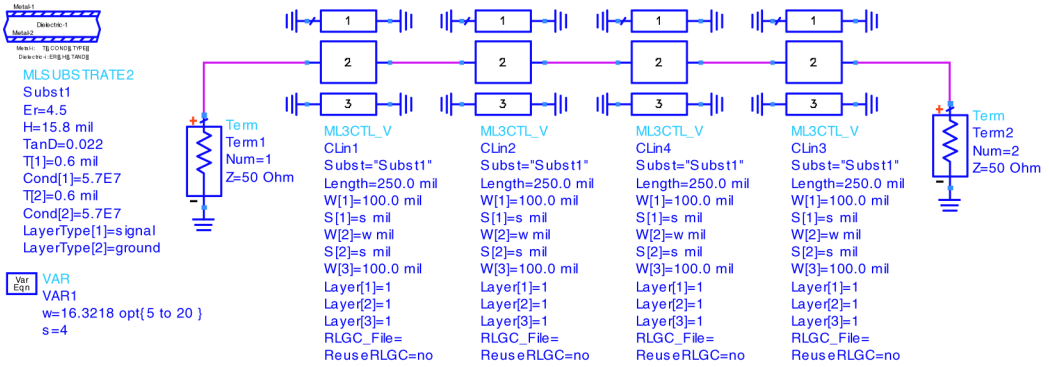


Figure 4–49. Simulation Results for a Grounded Coplanar Wave Guide

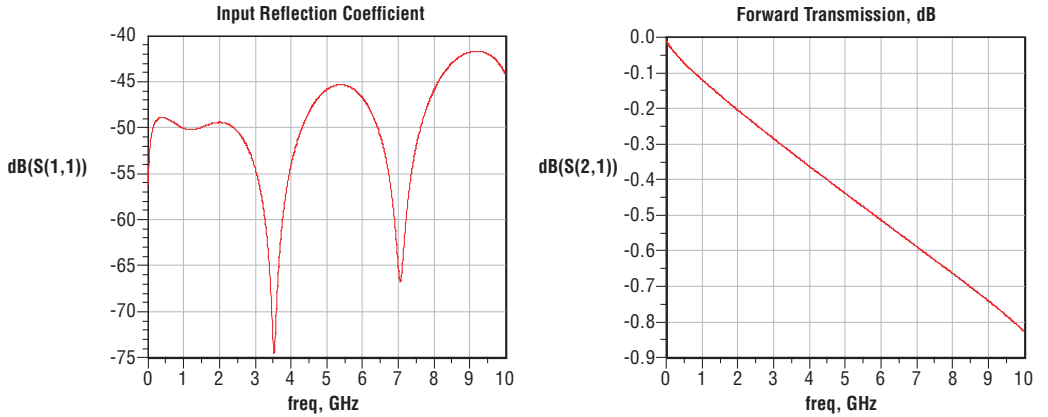
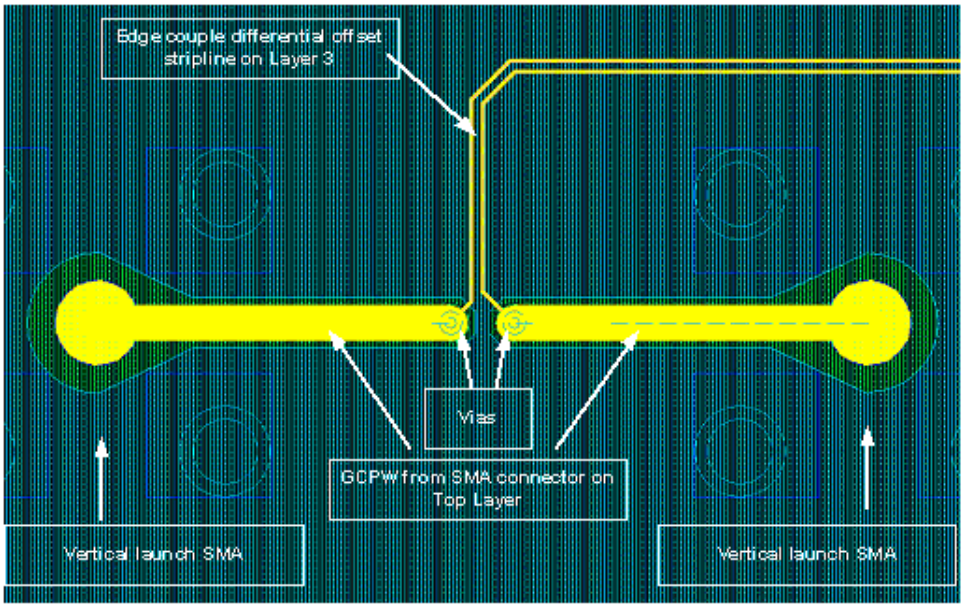


Figure 4–50 shows the Stratix GX development board's grounded coplanar wave guide and its edge-coupled differential offset stripline layout capture plot. This particular differential pair was used for the 1.0-Gbps source synchronous lines.

Figure 4–50. Layout Capture Plot for the Grounded Coplanar Wave Guide & Edge-Coupled Differential Offset Stripline

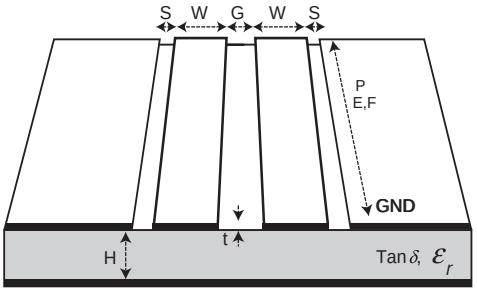


Grounded Differential Coplanar Wave Guide

The grounded differential coplanar wave guide is the differential version of the grounded coplanar wave guide and is used in high-speed digital systems that require maximum noise immunity. Figure 4–51 shows the topology. The same limitations for S and G apply in this topology as for the grounded coplanar wave guide.

This topology does not appear on the Stratix GX development board because of the loose coupling requirements. See "Broadside-Coupled Differential Offset Stripline" on page 4–42 for more information.

Figure 4–51. Grounded Differential Coplanar Wave Guide



Figures 4–52 and 4–53 show the frequency domain simulation setup and results for a 100-Ω grounded differential coplanar wave guide, respectively.

Figure 4–52. Simulation Setup for a Grounded Differential Coplanar Wave Guide

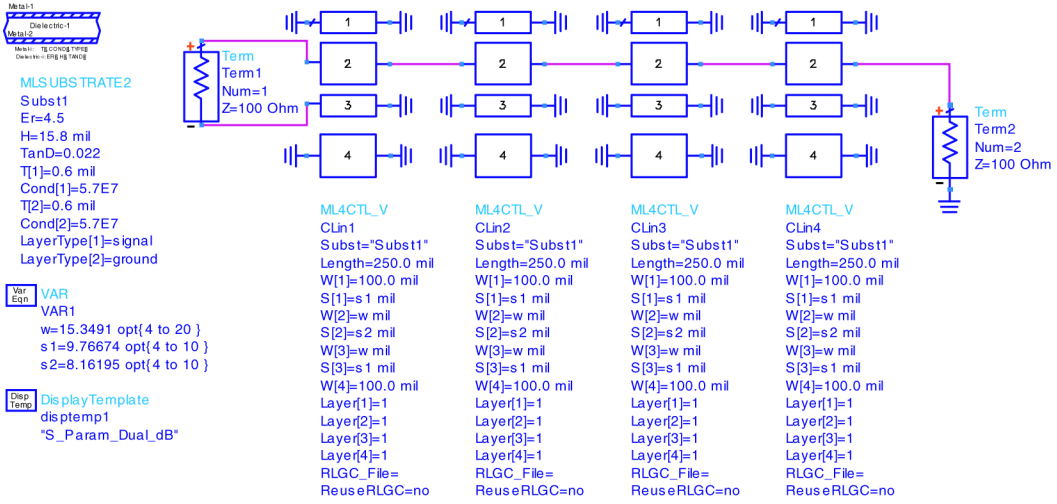
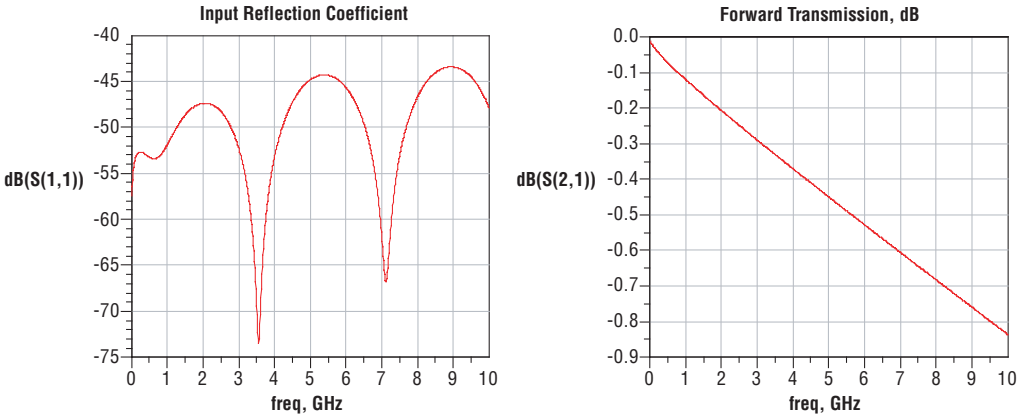


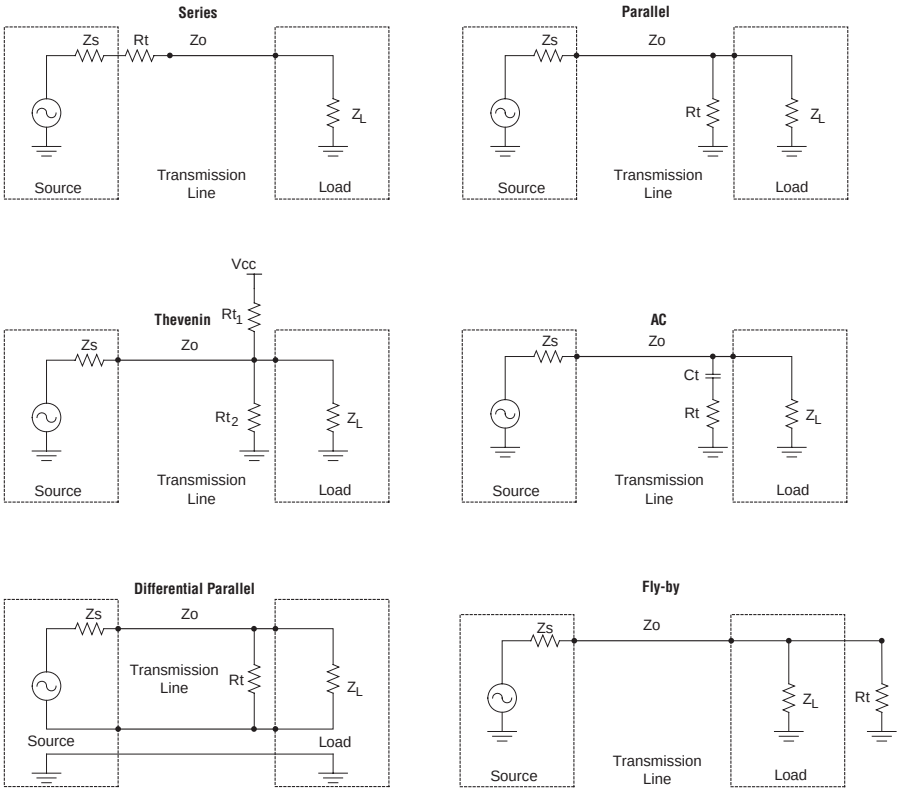
Figure 4–53. Simulation Results for a Grounded Differential Coplanar Wave Guide



Transmission Line Termination

High-speed signals require termination at the beginning, end, or both sides of the transmission line to prevent the reflected signal from the ends from corrupting the original signal. This section briefly describes several termination techniques. For more details about each technique, refer to *AN315: Guidelines for Designing High-Speed FPGA PCBs*. Figure 4–54 shows the different termination techniques.

Figure 4–54. Termination Techniques



Series Termination

With series termination, you place the termination resistor in series with the transmission line at the source end of the transmission line. The value of the resistor (R_s) and the value of the output impedance of the buffer must add up to 50 Ω for optimal performance. This is not always possible in cases when the buffer output impedance depends on the current setting or the voltage swing. In most cases, using a 22-Ω resistor in series works well for typical CMOS buffers such as those in Stratix GX devices. Series termination minimizes the DC power dissipation, but perfect impedance matching is not always possible, because the buffer output impedance varies.

Parallel Termination

With parallel termination, you place a resistor whose value is equal to the impedance of the trace at the end of the line on the receiver side. Ideally, the resistor should be directly on the pin. Parallel termination is better than series termination when the output impedance of the buffer is unknown. With parallel termination, you get an almost perfect impedance match between the transmission line and the termination resistor. The disadvantage with parallel termination is that it increases the DC power dissipation because of the DC path to ground.

Thévenin Termination

Thévenin termination is identical to parallel termination, except that you use two resistors to achieve the impedance match. The two resistors can also provide a DC level. Two criteria determine the value of the resistors. First, the two resistors in parallel must equal the line impedance. Second, the resistors must form a voltage divider so that the restored DC voltage equals the desired DC level. With parallel termination, you can achieve an almost perfect impedance match between the transmission line and the termination resistors. You can also restore DC voltage to AC-coupled signals. The main disadvantages to thévenin termination are the DC power dissipation and the added elements.

AC Termination

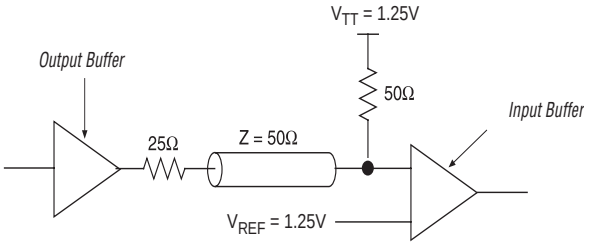
AC termination is identical to parallel termination except that the termination impedance consists of a resistor and a capacitor. The capacitor acts as a DC block, reducing static power dissipation. The capacitor must be chosen so that the RC time constant is about one rise time of the signal. Perform simulations to choose the best value of the capacitor. AC termination provides the advantages of parallel termination without the DC power dissipation. The main disadvantages are the rise and fall time degradation and the added elements.

Differential Parallel Termination

With differential parallel termination, you place the termination resistor between the positive and negative signals, close to the receiver pins. This type of termination is a subset of parallel termination.

Specific I/O standards, for example, SSTL-II for the DDR interface, sometimes require both series and parallel termination. Figure 4–55 shows the termination shown in this particular standard.

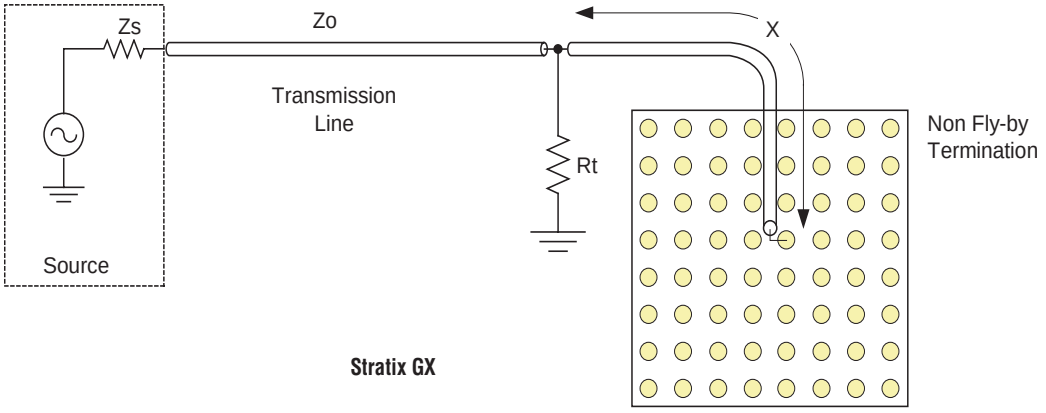
Figure 4–55. Class I Termination for DDR Memory Interfaces



Fly-By Termination

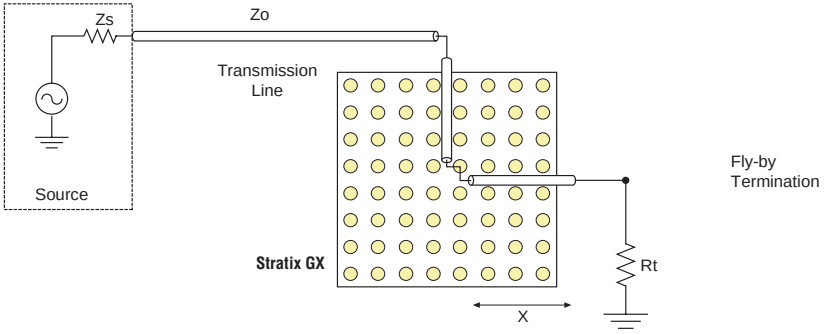
Fly-by termination is not precisely a termination technique, but a placement technique for termination resistors. To understand fly-by, consider the parallel (non fly-by) termination shown in Figure 4–56. Layout constraints might prevent the termination resistor, R_t, from being placed close to the receiver pin (the load). If the pin is in the middle of the device and there is only an inch of trace length between the termination point and the receiver pin, the setup degrades the signal quality, because the extra stub length acts as a capacitive load.

Figure 4–56. Normal Termination Resistor Placement



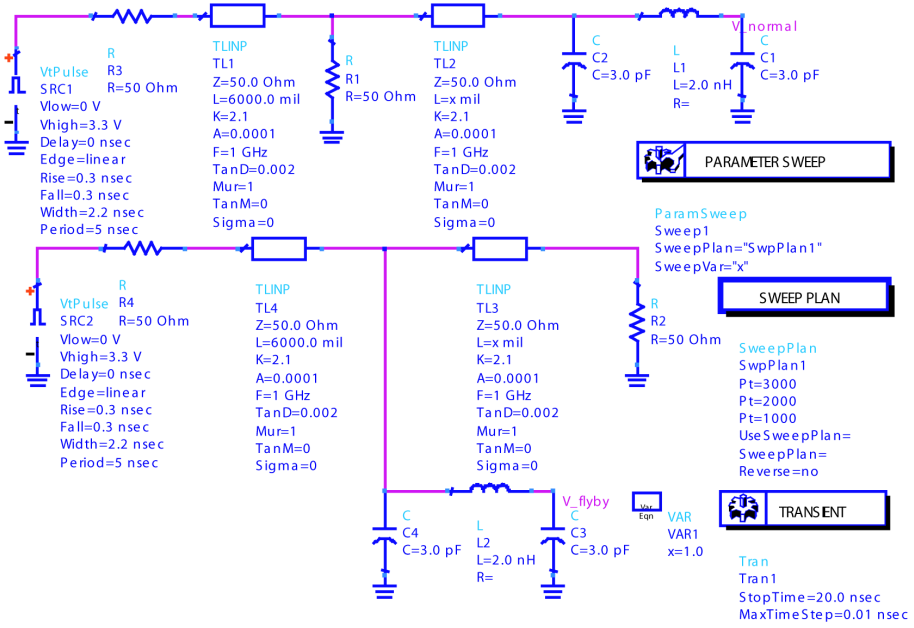
To prevent this degradation, place the termination resistor after the load, as shown in Figure 4–57. Even if the resistor is a couple of inches away, the termination occurs at the end of the transmission line, and there are no significant reflections. The trace from the transmission line to the receiver pin can be kept small.

Figure 4–57. Fly-By Termination Resistor Placement Technique



The simulation setup for fly-by termination is shown in Figure 4–58. The upper circuit in this diagram is for fly-by termination resistor placement, and the lower is for normal placement. Figure 4–59 shows some simulation results that show how the fly-by termination technique helps improve signal integrity.

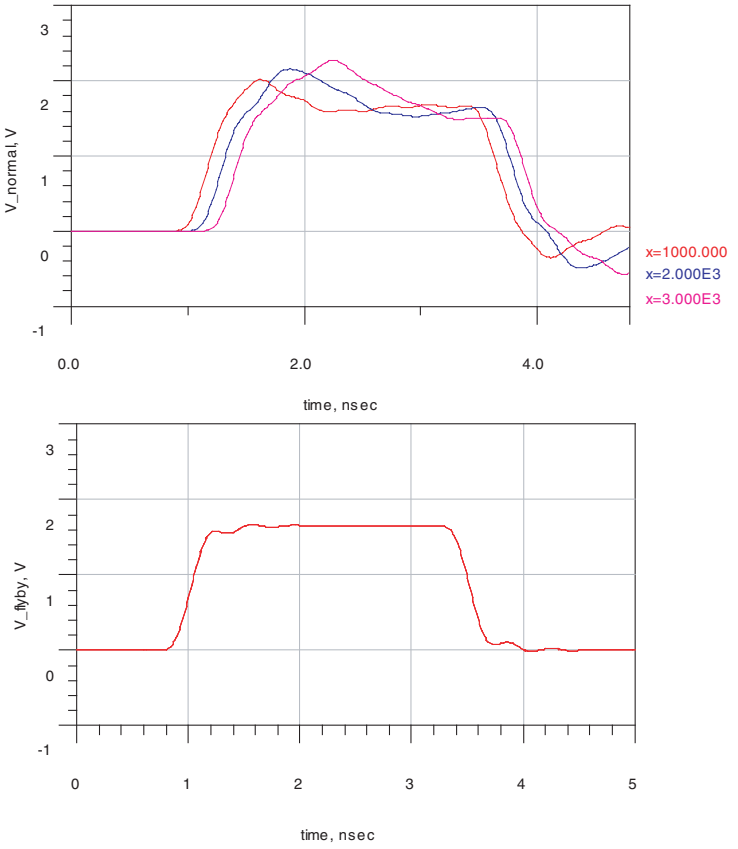
Figure 4–58. ADS Simulation Setup for Normal Versus Fly-By Termination



The simulation results are shown in Figure 4–59. The signal quality remains the same regardless of how far away the resistor is, provided that it is placed in fly-by mode.

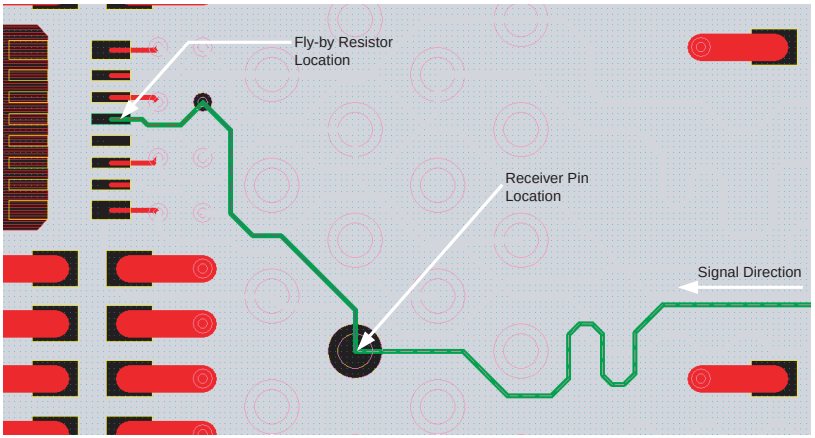
For the non fly-by case, the signal has an overshoot, which worsens when you place the resistor farther away.

Figure 4–59. Signal Quality at the Receiver Device Pins



Altera uses the fly-by resistor placement technique extensively on the Stratix GX development board. Figure 4–60 shows an example capture plot.

Figure 4–60. Fly-By Termination Example



Transmission Line Routing

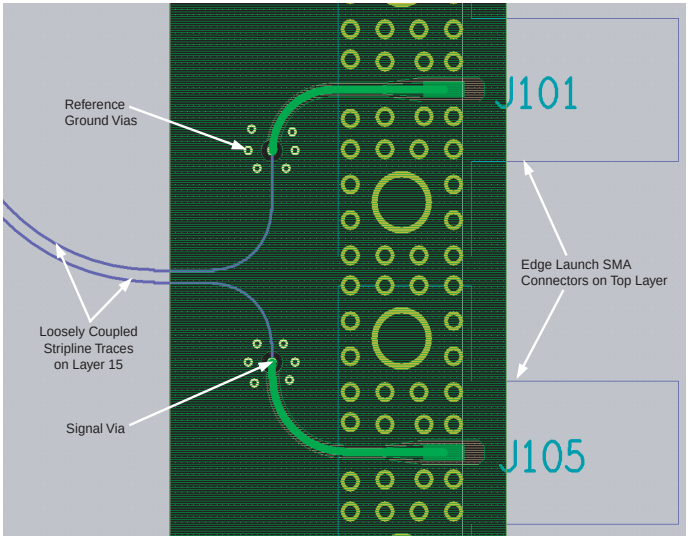
This section describes the things to keep in mind when designing transmission line routing for your high-speed board.

General Guidelines

This section provides a summary of routing guidelines.

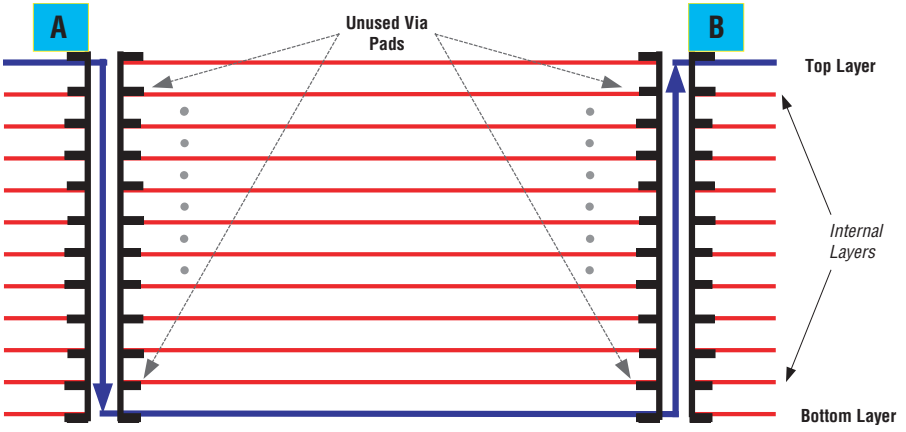
- Use tight coupling for differential traces as much as possible. If it is not possible to maintain tight coupling throughout the trace length, then you must use loose coupling for the entire trace. Figure 4–61 shows 3.125-Gbps differential transceiver traces, which are loosely coupled on layer 15 and the top layer of the Stratix GX development board. Tight coupling is not possible on the top layer because of the minimum separation required (because of the mechanical constraints of the SMA connectors). If you used tight coupling on layer 15, the signal would need to transition from tight to loose coupling, which would introduce impedance discontinuities.

Figure 4–61. Loose Coupling & Ground Reference Example



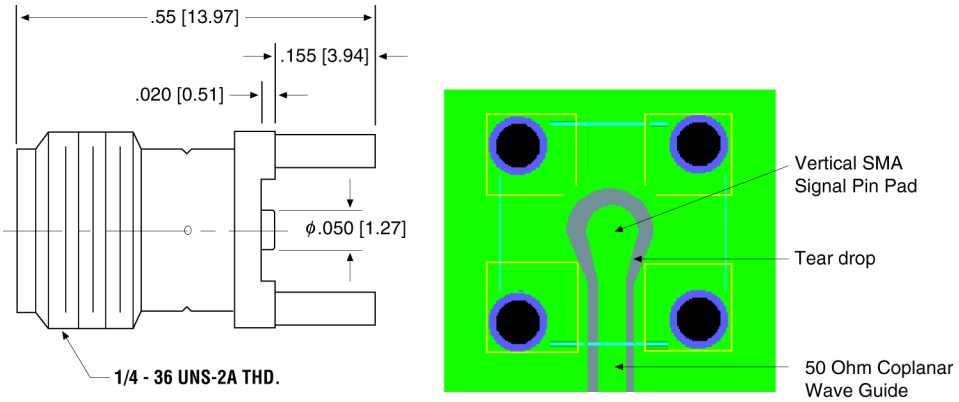
- Microstrip and stripline transmission lines are both fine for routing. Stripline tends to have more loss than microstrip, but it also shields the signals from possible noise from other sources. The constraints during layout design often dictate on which layer to do the routing.
- Use rounded corners while routing. Do not use 90° bends, which introduce impedance discontinuities. A 45° bend provides a compromise, but rounded corners offer the best performance.
- Both broadside and edge coupling are fine. Generally, broadside coupling makes it easy to route out of the BGA, but requires more layers. Edge coupling makes it harder to route out of the BGA, but requires fewer layers. Both provide acceptable signal integrity performance.
- Do not allow high-speed signals to cross over plane splits. A crossover causes signals to see a longer return path, which increases trace inductance. The increased inductance changes the impedance of the line, causing signal integrity problems.
- Remove all unused via pads during the manufacturing process. Some fabrication houses refer to unused via pads as "nonfunctional via pads." Unused via pads add additional capacitance on the signal path. Figure 4–62 shows examples of unused via pads.

Figure 4–62. Unused Via Pads



- If the signal must be routed through a via, route it so that the via stub length is minimized. The option shown in Figure 4–64 minimizes the via stub length and is preferred over the option shown in Figure 4–65. Altera used this technique in the Stratix GX development board. The high-speed traces were taken to layer 16 from the top layer through the vias to use as much of the via length as possible and to minimize the stubs.
- For dual stripline traces (Power 1 – Signal 1 – Signal 2 - Power 2 configuration), ensure that the signals on the Signal 1 layer are orthogonal to the signals on the Signal 2 layer if they cross each other. If they do not cross each other, ensure that they are at least 4W away, where W is the width of the traces.
- If a connector has a section that does not have controlled impedance, try to minimize the length of the connector. For example, for SMA connectors, reduce the center conductor length as much as you can (less than 20 mils if possible). This way the parasitic inductance is reduced.
- Use "teardropping" to reduce impedance discontinuity when going from a wide pin and trace to a narrow pin or trace. For example, when you interface an SMA connector to a trace, use teardropping. Figure 4–63 shows the dimensions of an SMA connector and a screen capture of how Altera used teardropping on the Stratix GX development board. The SMA connector has a 50-mil diameter center pin, but the board traces might be 5-mils wide. Teardropping helps minimize the impedance discontinuity.

Figure 4–63. SMA Teardropping



- Avoid routing a high-speed signal over many layers, because this type of routing introduces via discontinuities.
- Use reference ground vias wherever signal vias are necessary. The reference ground vias should be placed close to the signal vias. Figure 4–61 on page 4–59 shows an example of using ground reference vias near the signal vias where a high-speed trace changes layers. This example is from the edge-launch SMA connectors of the 3.125-Gbps transceivers on the Stratix GX development board.
- Avoid having long stubs on the via, using most of the via length for routing. For example, if you have a signal that originates on the top layer and you have to use an internal layer for routing, use the one closest to the bottom layer. This approach avoids excess via length. The option shown in Figure 4–64 on page 4–63 is preferred over the option shown in Figure 4–65 on page 4–63.
- Avoid reference plane changes for high-speed signals. Reference plane changes occur when you reference part of the trace to a ground plane and the other part to another ground plane or a power plane.
- Avoid using power planes for references.

Length Matching

Source-synchronous interfaces require length matching, except when using dynamic phase alignment. You must ensure that this requirement is clearly stated in the layout guidelines. How closely the lengths must be matched depends on the data rate and the available timing margin. For example, consider a source synchronous link at 840 Mbps where the bit unit interval equals 1.19 ns. In this example, assume that you allocate a length mismatch of 2% for the maximum timing margin loss. At 1.19 ns

unit intervals, this value equals approximately 24 ps. Using the microstrip equation on page 4–4, for a microstrip line with FR-4 dielectric, the signal travels 1 inch in 142 ps. In 24 ps the signal can travel 24/142 = 0.17 inches, or 170 mils. Therefore, you must match the lengths of all the signals (including data, clocks, addresses, and any controls signals) to within ± 85 mils.

Examples of source-synchronous interfaces include the 840-Mbps interface available on Stratix and Stratix GX devices, the SSTL interface for DDR memory, and the HSTL interface for Quad Data Rate (QDR) interface.

A notable exception to the length matching requirement is the source-synchronous interface with DPA available in Stratix GX devices, which allows 1-Gbps data transfer without requiring length matching. The data and clock lines can literally be inches longer or shorter than each other and the interface still operates robustly.

You must account for the length of every via a signal traverses through and include the accumulated value in the length report (a file generated by the layout tool). Do not include the entire length of the via, just the portion the signal traverses (ignore the stub length). For example, Figures 4–64 and 4–65 show two possibilities for signal routing on a board with 12 signal layers. For both options, the signal originates at point A and ends at point B on the top layer. In Figure 4–65, the signal is routed on layer 3; in Figure 4–64, the signal is routed on the bottom layer. The total length the signal travels, for the respective options, is:

$$L_{routing,option1} = (z' + y + z') = (2z' + y)$$

$$L_{routing,option2} = (z + y + z) = (2z + y)$$

Figure 4–64. Optimal Routing Path With Through-Hole Vias

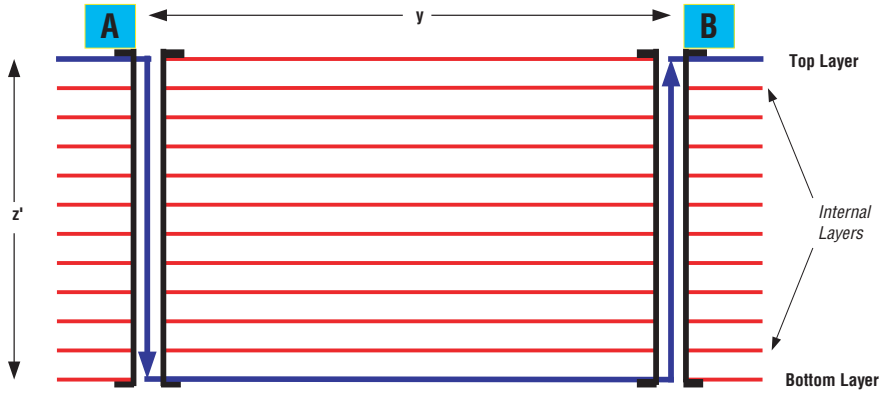
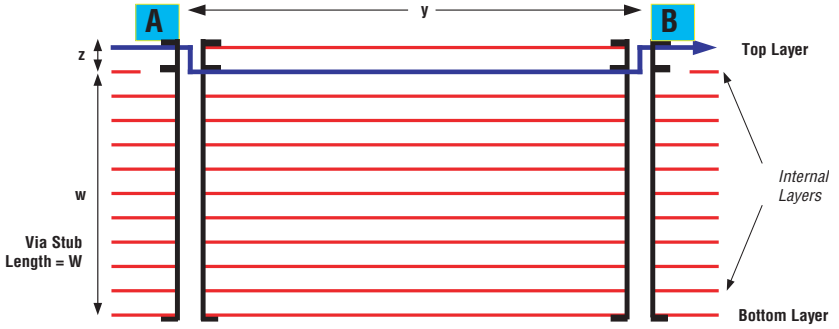


Figure 4–65. Sub-Optimal Routing Path With Through-Hole Vias



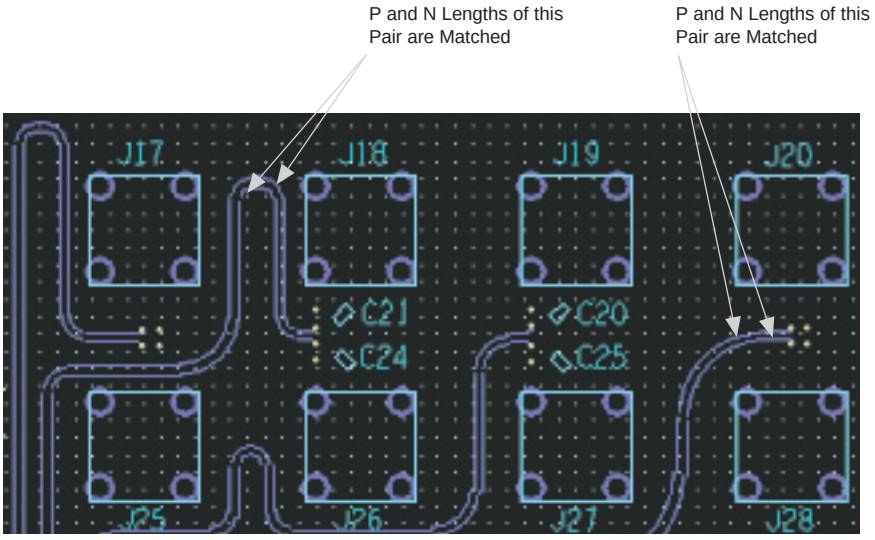
In Figure 4–65, W represents the extra stub that you need to consider in length calculations only if you use layer 16 for routing.

In the equations on page 4–62, the total difference in via lengths between these two routing options is 2z'-2z = 2w. This distance can be over 100 mils for thick boards.

Because this kind of calculation can quickly become intractable, it is best to use as few vias as possible on the board and to tighten the length-matching specification. For instance, in the example above, you can tighten the specification from ± 85 mils to ± 50 mils.

For differential traces, it is important to match the length of the positive and negative signals. A mismatch in lengths causes duty cycle distortion, which can decrease the timing margin. Altera recommends that the lengths be matched to ± 50 mils. If the signal goes through a turn and the outer trace is longer than the inner trace, you can use a second turn in the opposite direction as an offset. Figure 4–66 shows an example of high-speed transceiver routing to achieve length matching on the Stratix GX development board.

Figure 4–66. Transceiver Signal Routing for Length Matching



Other Transmission Line Issues

This section describes transmission line issues other than routing.

Stackup

The stackup design is an iterative process in which the relative permittivity (ε_r), trace width, layer count, and transmission line structures are set by engineers to achieve the best system performance at the minimum cost. These parameters are selected based on the

engineering team's knowledge of the system, the fabrication house, and the simulations. Table 4–6 shows the stackup for the Stratix GX development board.

Table 4–6. Stratix GX Stack-Up

Layer Number	Type	Cu Weight (oz)	Foil Thickness (in.)	Dielectric Thickness (in.)	Construction	ε_r
1	Signal	0.5	0.0007	0.0036	1 × 1080	3.76
2	Plane	1.0	0.0014	0.0040	1080 + 2113	4.00
3	Signal	0.5	0.0007	0.0061	2116	3.86
4	Signal	0.5	0.0007	0.0040	1080 + 2113	4.00
5	Plane	1.0	0.0014	0.0045	1080 + 2113	3.86
6	Signal	0.5	0.0007	0.005	2116	3.63
7	Signal	0.5	0.0007	0.0045	1080 + 2113	3.86
8	Plane	1.0	0.0014	0.0040	1080 + 2113	4.00
9	Signal	0.5	0.0007	0.0061	2116	3.86
10	Signal	0.5	0.0007	0.0045	1080 + 2113	4.00
11	Plane	1.0	0.0014	0.005	1080 + 2113	4.00
12	Signal	0.5	0.0007	0.0045	2116	3.86
13	Signal	0.5	0.0007	0.005	1080 + 2113	4.00
14	Plane	1.0	0.0014	0.0045	1080 + 2113	4.00
15	Signal	0.5	0.0007	0.0061	2116	3.86
16	Signal	0.5	0.0007	0.0040	1080 + 2113	4.00
17	Plane	1.0	0.0014	0.0036	1 × 1080	3.76
18	Signal	0.5	0.0007			
Total Thickness Cu to Cu (in.)		0.093 ± 0.006				

Because of the large number of power and ground planes needed on the Stratix GX development board, Altera uses a symmetrical stackup that is optimized for striplines. The dielectric thickness, permittivity, and trace width were chosen so that the total board thickness was approximately 0.093 in. A minimum trace width of 0.004 in keeps costs low and reduces the effects of fabrication deviations. The layer count is set based on the density of the board.

The development board material is NELCO FR-4 4000-6, which is an enhanced multifunctional, high temperature resin system that shows a relative permittivity of ~3.86. and tanδ of 0.022. The stackup consists of 6 solid planes and 12 signal layers arranged symmetrically from the center.

For transmission line topologies, Altera uses differential microstrip, grounded coplanar wave guide, and differential dual stripline (offset edge-coupled differential stripline). The impedance is 50 Ω for single-ended traces and 100 Ω (differential) for differential traces.

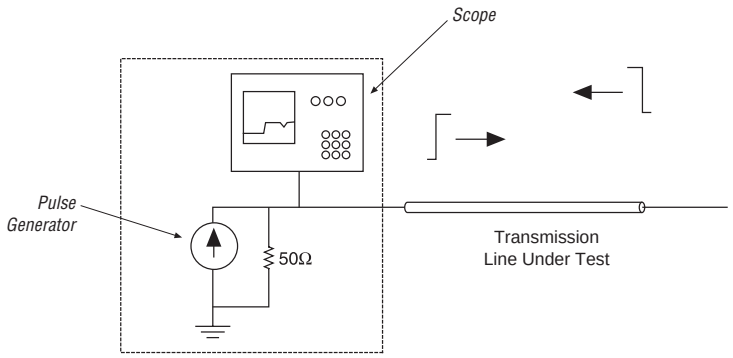
In the stackup design process, the fabrication house has considerable influence during the fine-tuning stage. Typically, the engineer provides a preliminary stackup, and the fabrication house responds with a modified stackup based on their process tolerances. They typically guarantee the impedance required on the traces. Typical tolerances are ± 10%, although ± 5% is fairly common for high-speed boards. Altera uses ± 5% for better system performance.

Discontinuities

You can characterize discontinuities in the transmission path using either a frequency domain tool or a time domain tool. In the frequency domain, you can use a network analyzer for characterization. A commonly used time domain tool is the time domain reflectometer (TDR). For digital systems that are commonly described in time domain, the TDR is often easier and more intuitive to use. This section summarizes the principles of a TDR and how to interpret the results.

A TDR is an oscilloscope with the capability of sending a very fast signal pulse. The pulse travels down the transmission line and is reflected wherever it sees a discontinuity. Figure 4–67 shows a conceptual diagram of a TDR. The reflected wave takes $2T$ time units to be displayed in the scope, because it takes T time units for the pulse to reach the discontinuity and another T time units for it to return to the scope.

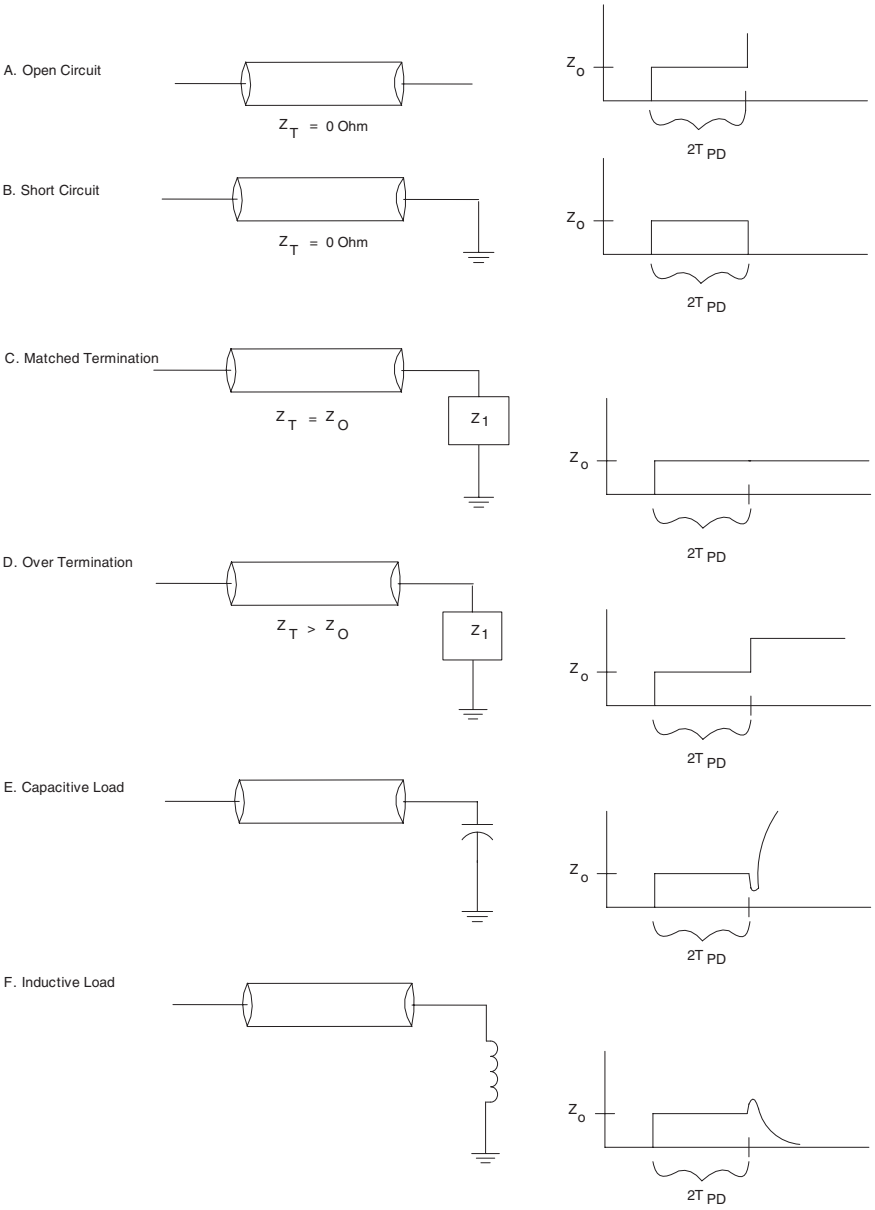
Figure 4–67. Conceptual Diagram of a TDR System



Using a TDR, resistive discontinuities show a staircase profile, capacitive discontinuities cause a dip, and inductive discontinuities cause a bump. The reason for this is that initially the capacitor acts as a short-circuit for a high-frequency pulse, and the inductor acts like an open circuit. Based on the location of the impedance discontinuity on a TDR plot, you can trace it to the exact board location.

Figure 4–68 shows the different types of terminations and the associated TDR responses.

Figure 4–68. TDR Responses to Different Terminations & Discontinuities



To determine the value of the resistive discontinuity, look at the size of the step in the staircase waveform. To determine the value of the capacitive or inductive discontinuity, look at the rise time of the associated response. Then use the following formulas:

Resistive Termination

$$R = Z_0 \frac{1+\Gamma}{1-\Gamma}$$

where Γ is the reflection coefficient. For a 1-V initial signal, Γ equals the step of the staircase. For a different initial step, you need to scale Γ linearly.

Capacitive Termination

$$tau = 2.2t_r = RC$$

for first order R-C circuits.

Inductive Termination

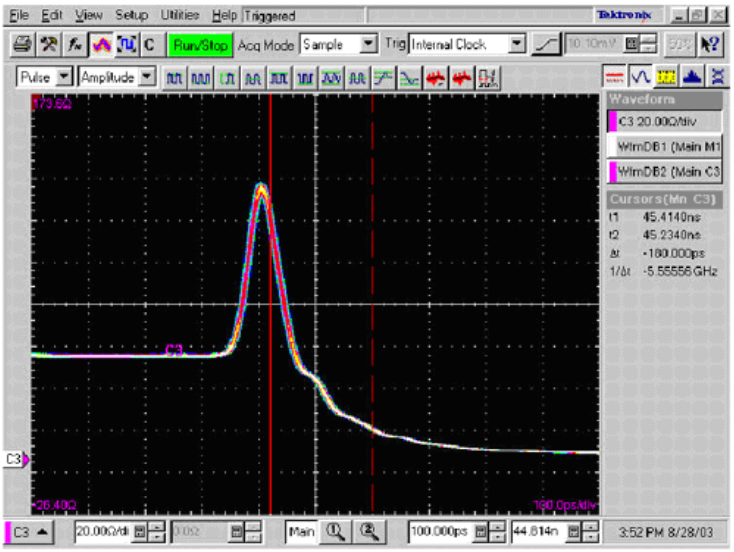
$$tau = 2.2t_r = L / R$$

for first order L-R circuits.

In the previous equations, R is the effective resistance and impedance in the circuit. The rise time (t_r) is determined from the TDR response. The only variables remaining are L and C, either of which can be computed.

Example: Find the value of the inductance on the 0.1-in pitch header pin connector from the TDR response shown in Figure 4–69.

Figure 4–69. TDR Plot of a Header Connected to a 50-Ω Cable



Solution: From the plot of the header discontinuity, the signal fall time is about 180 ps, and the cable's impedance is 50 Ω Therefore,
$L = 2.2 \times t_r \times R = 2.2 \times 180 \times 50 = 19{,}800$ pH = 19.8 nH.



Real-world TDR responses are often complicated and hard to interpret because of multiple reflections and a combination of discontinuities (capacitive, inductive, and resistive) that could be present on the board. One approach is to isolate each part and measure the TDR by soldering the part at the end of an SMA cable so that each part can be characterized separately.

As a real-world example of a TDR response, consider the following measurements, which were taken to characterize SMA connectors being evaluated during the design of the Stratix GX development board. The SMA connectors are shown in Figure 4–70, and the results are shown in Figure 4–71.

The first measurement Figure 4–71 shows the TDR response of the SMA cable used. The second shows the response of the SMA cable with the connector connected at the end. You can see quite a bit of ringing, signifying the presence of parasitics. The third shows the response of the SMA cable and connector assembly, with the connector's center conductor stripped down to about 10 mils in length. This response is almost identical to the response of the SMA cable only, except for the

presence of a fixed delay. Comparing these plots shows that the center conductor of the SMA connector adds quite a few unwanted parasitics, but stripping down the conductor greatly reduces the parasitics.

Figure 4–70. Top & Side Views of Original & Modified SMA

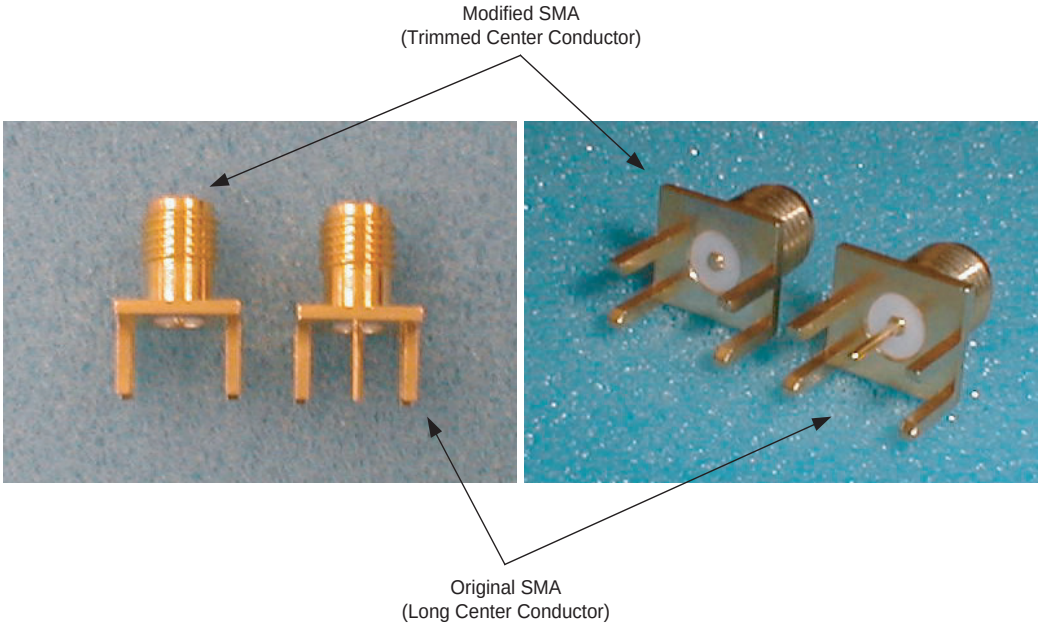
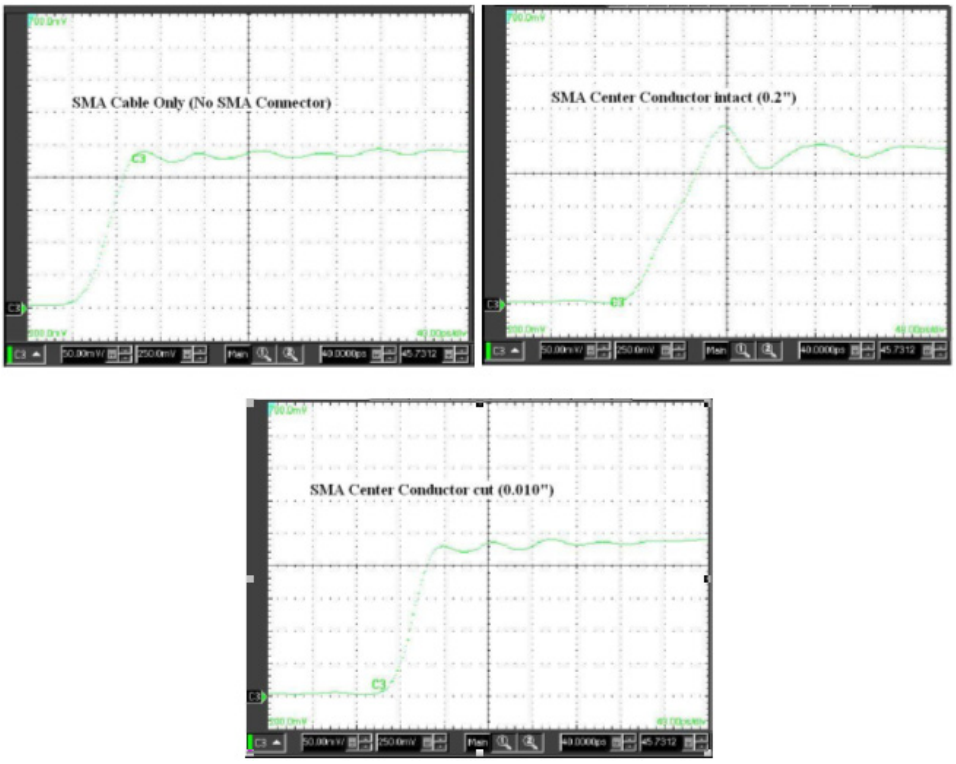


Figure 4–71. Impedance Profiles of SMA Connectors, Measured by TDR



Crosstalk

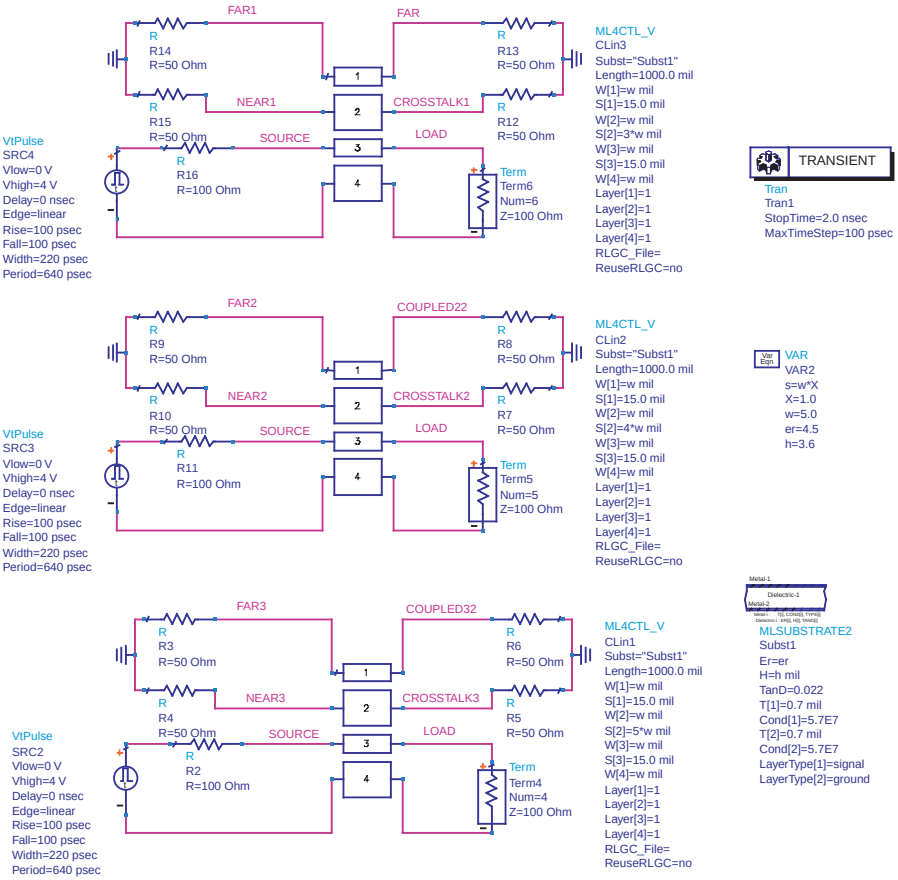
If two traces are close to each other, the signal switching on one trace can excite a voltage on the other, resulting in crosstalk. Crosstalk control must be a key objective of any board design. With high-speed designs it is especially important because of the reduced noise margins.

The following crosstalk case studies are simulations that vary the spacing between the aggressor net and the victim net. Both simulations used differential traces.

Crosstalk Case Study 1 – Loosely Coupled Microstrip Traces
This case study used two pairs of 1-inch long microstrip differential traces. The width (W) of all the traces is 5 mils. The spacing between the positive and negative loosely coupled traces is 15 mils. The aggressor trace had the same configuration, but was placed at varying distances to

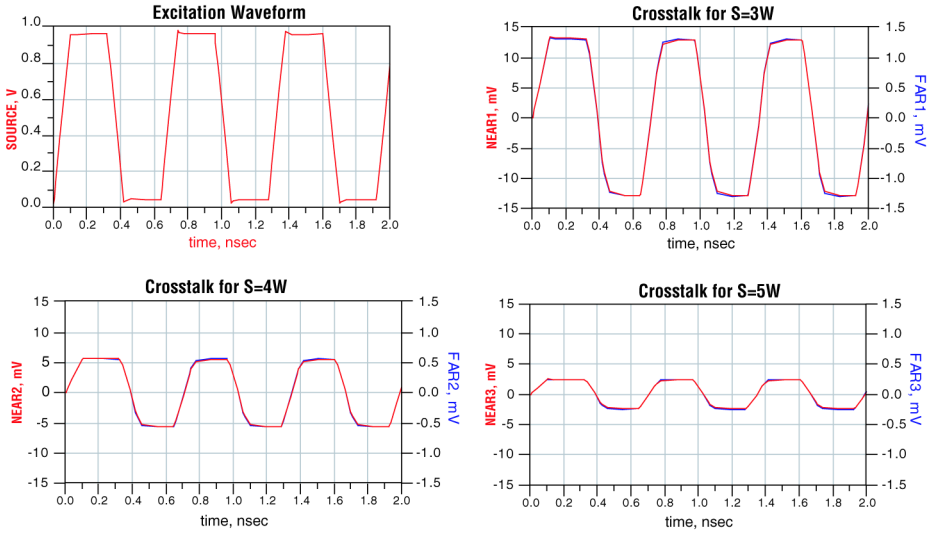
see the effect of spacing on crosstalk. This test setup is based on the microstrip lines routed on the top layer of the Stratix GX development board. The simulation setup is shown in Figure 4–72.

Figure 4–72. ADS Test Setup for Loosely Coupled Microstrip Crosstalk Simulation



The simulation results are shown in Figure 4–73. The input waveform is at the top left. Crosstalk on the near and far victim lines for S = 3W appears in the top right. The bottom left shows crosstalk for S = 4W, and the bottom right shows crosstalk for S = 5W.

Figure 4–73. Crosstalk Results for Loosely Coupled Microstrip



When the aggressor trace is S = 3W away, the crosstalk on the nearest victim line is approximately 13.5 mV, and the crosstalk on the farthest victim line is approximately 0.9 mV. When the spacing is increased to S = 4W, the crosstalk on the nearest victim line decreases to 2.5 mV, and the crosstalk on the farthest victim line decreases to 0.6 mV. The decreases on the farthest line are not significant because the lines are loosely coupled: increasing the spacing by 1W does not significantly increase the overall distance to the farthest line. When the spacing is increased to S = 5W, the crosstalk on the nearest victim line decreases to 1.5 mV and the crosstalk on the farthest victim line decreases to 0.6 mV. These three situations allow you to determine what level of noise you can tolerate on your victim lines. Altera recommends maintaining a minimum S = 4W, which is the guideline followed for the Stratix GX development board.

Crosstalk Case Study 2 – Tightly Coupled Stripline Traces

This case study used two pairs of 1-inch-long tightly coupled stripline traces. This setup replicates the tightly coupled stripline traces of the Stratix GX development board. The width of all traces is 3.7 mils, and the spacing between positive and negative traces is 5.3 mils. The aggressor trace used the same configuration but was placed at varying distances to see the effect of spacing on crosstalk. Figure 4–74 shows the simulation setup, and Figure 4–75 shows the results. In Figure 4–75, the input waveform is on the top left. Crosstalk on the near and far victim lines for S = 3W is shown in the top right. The bottom left shows crosstalk for S = 4W, and the bottom right shows crosstalk for S = 5W.

Figure 4–74. ADS Test Setup for Tightly Coupled Stripline Crosstalk Simulation

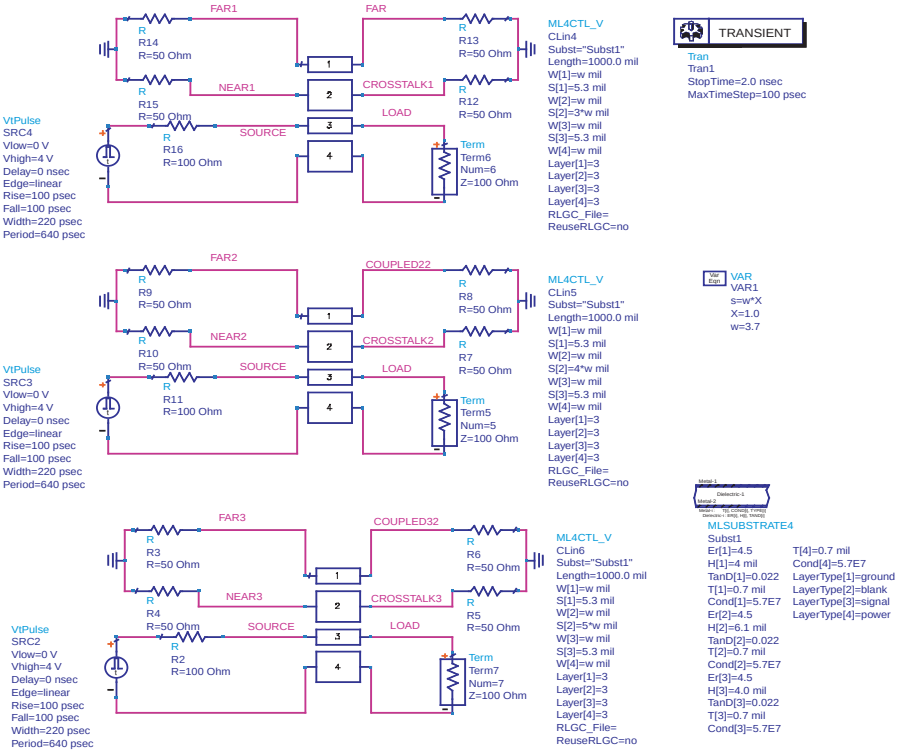
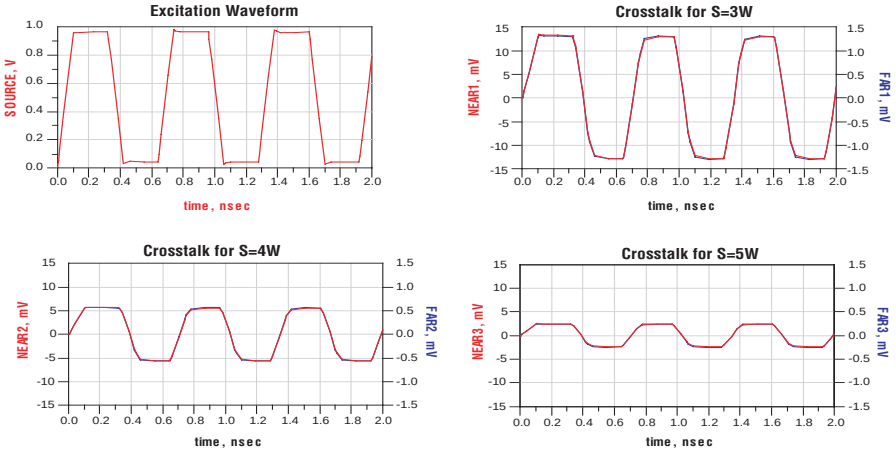


Figure 4–75. Crosstalk Results for Tightly Coupled Stripline



When the aggressor trace is S = 3W away, the crosstalk on the nearest victim line is approximately 13.5 mV, and the crosstalk on the farthest victim line is approximately 1.35 mV. When the spacing is increased to S = 4W, the crosstalk on the nearest victim line decreases to 5.5 mV, and the crosstalk on the farthest victim line decreases to 0.55 mV. The decreases on the farthest line are not significant because the lines are tightly coupled: increasing the spacing by 1W does not significantly increase the overall distance to the farthest line. When the spacing is increased to S = 5W, the crosstalk on the nearest victim line decreases to 2.5 mV, and the crosstalk on the farthest victim line decreases to 0.25 mV. These three situations allow you to determine what level of noise you can tolerate on your victim lines. Altera recommends maintaining a minimum S = 4W, which is the guideline it follows for the Stratix GX development board.

Miscellaneous

This section describes miscellaneous topics, including component selection, S-parameters, the Smith Chart, AC and DC coupling, unused pin connections, and power trace thickness.

Component Selection for High-Speed Design

Deciding which components to use is an important part of board design. This section provides information and guidelines for selecting discrete components for the PCB.

Resistors, Capacitors, Inductors & Ferrite Beads

You should choose the smallest footprint available when selecting discrete components such as resistors and capacitors. The small footprint means that the pad on the board can be small and that the parasitic capacitance and inductance will also be small. Altera typically uses 40 mil × 20 mil (0402) package components for the high-speed signals.

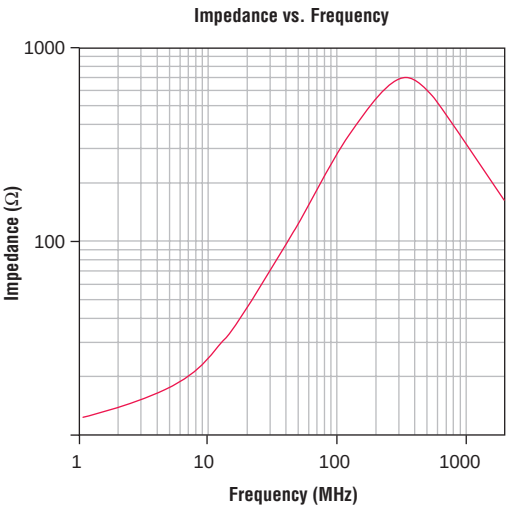
Inductors typically require bigger footprints because they are often used for power supply filtering and must be bigger to support high currents without saturating.

The three parameters of interest when choosing ferrite beads are:

- DC resistance
- AC impedance
- Current handling capability

A good ferrite bead has low DC resistance, high AC impedance, and high current handling capability. However, as the current handling capability increases, the AC impedance tends to drop; consequently there is a trade-off involved. The impedance versus frequency plot of a typical ferrite bead is shown in Figure 4–76.

Figure 4–76. Typical Impedance Profile of a Ferrite Bead



In Figure 4–76, the impedance at 2 GHz is more than 100 Ω The ratio between that impedance and the power supply impedance, which is often lower than 1 Ω, is more than 100. As a result, most of the noise is blocked by the ferrite bead and is shunted to ground instead.

Altera recommends Steward MI0805M221R-00 ferrite beads for transceiver power and ground planes. The DC resistance for this part is lower than 50 mΩ, and it can handle 2.5 A of current. The impedance is over 200 Ω at 1 GHz. Altera also recommends the Murata BLM31PG500SN1 ferrite bead. It has 25 mΩ of DC resistance, 3 A of current, and has 75 Ω of AC impedance at 1 GHz. Two Steward ferrite beads connected in parallel can provide 5 A of current capability with 25 mΩ of DC resistance and over 100 Ω of AC impedance. This performance level is adequate for most applications.

SMA Connectors

SMA connectors are typically used for high-speed signals because of their controlled impedance, mechanical robustness, and good signal integrity. These connectors come in the form of edge launch or vertical launch. For edge launch, the connector is connected to the board edge, the central conductor stays flush with the board, and the board is sandwiched between the ground conductors.

The vertical launch type is through-hole or surface mount. For the through-hole type, all the legs and the center conductor go through the board. For a surface mount type, the center conductor barely touches the top layer of the board.

The type of launch technique does not affect the signal quality very much. However, a long center conductor adds inductance to the transmission path and degrades the signal.

With vertical launch surface mount, you can often strip the center conductor down to below 20 mils. This stripping might be slightly harder to achieve in edge launch configuration, but the choice is dependent on the capability of the SMA manufacturers. Altera has stripped the center conductor down to less than 20 mils short in its designs. Altera recommends using Lighthorse Technologies and Northrop Grumman for SMA connector requirements.

SMA Cables

SMA cables have 18-GHz bandwidth or more, which is sufficient for 3.125-Gbps applications. However, be careful that the cable is crimped securely at both ends to avoid unpredictable behavior.

Probes

Use a good differential probe with short tips for differential signals. Differential probes make the measurement immune to common mode noise and pickup on the probes.

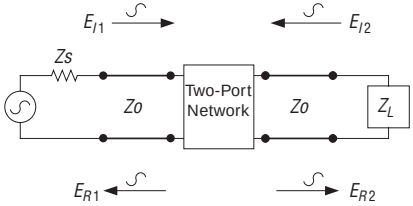
S-Parameters

It is difficult to define voltages or currents in transmission lines and to measure them at microwave frequencies, because direct measurements usually involve the magnitude (inferred from power) and phase of a wave traveling in a given direction, or the standing wave. Thus, equivalent voltages and currents, and the related impedance and admittance matrices, become somewhat of an abstraction when dealing with high-frequency networks.

A representation more consistent with direct measurements, and with the ideas of incident, reflected, and transmitted waves, is provided by the scattering matrix (S-parameters). S-parameters are often used to quantify the impedance, total losses, input return loss, insertion loss, and isolation and crosstalk for the different transmission lines structures. These concepts are most useful at those frequencies where you must consider distributed rather than lumped parameters.

To understand S-parameters, consider the two-port network shown in Figure 4–77. The network can contain a transmission line or any linear time invariant component. The incident voltages at port 1 and 2 are E_{I1} and E_{I2}; and the reflected voltages are E_{R1} and E_{R2}. Z_o is the characteristic impedance of the transmission line. Z_S and Z_L are the source and the load impedances.

Figure 4–77. S-Parameters



Taking the incident and reflected voltage waves on each side of the two-port network and dividing them by the square root of the characteristic impedance, Z_o, results in new variables (a_1, a_2, b_1 and b_2), which are the normalized voltage wave amplitudes at each port.

The square of the magnitude of a_1 ($|a_1|^2$) represents the incident power in port 1, and ($|b_1|^2$) represents the reflected power from this port. The same relations apply to port 2.

$$a_1 = \frac{E_{I1}}{\sqrt{Zo}} \quad a2 = \frac{E_{I2}}{\sqrt{Zo}}$$

$$b_1 = \frac{E_{R1}}{\sqrt{Zo}} \quad b_2 = \frac{E_{I2}}{\sqrt{Zo}}$$

The resulting parameters relate the scattered wave (reflected wave) from the network to the incidents waves. These parameters are called S-parameters and are defined by:

$$b_1 = S_{11}a_1 + S_{12}a_2$$

$$b_2 = S_{21}a_1 + S_{22}a_2$$

where S_{11} is the input reflection coefficient, S_{21} is the forward transmission through the network, S_{12} is the reverse transmission through the network, and S_{22} is the output reflection coefficient.

The measurement setup for S-parameters is shown in Figure 4–78 and Figure 4–79. Apply the following conditions for these measurements:

$$S_{11} = \left.\frac{a_1}{b_1}\right|_{a_2=0}$$

$$S_{21} = \left.\frac{b_2}{a_1}\right|_{a_2=0}$$

$$S_{12} = \left.\frac{b_1}{a_2}\right|_{a_1=0}$$

$$S_{22} = \left.\frac{b_2}{a_2}\right|_{a_1=0}$$

Figure 4–78. Measurement Setup for S_{11} & S_{21}

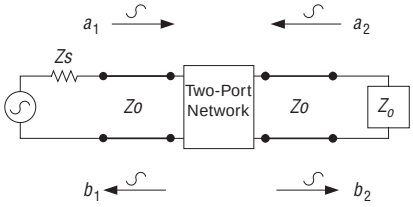
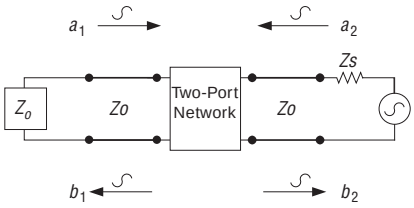


Figure 4–79. Measurement Setup for S_{22} & S_{12}



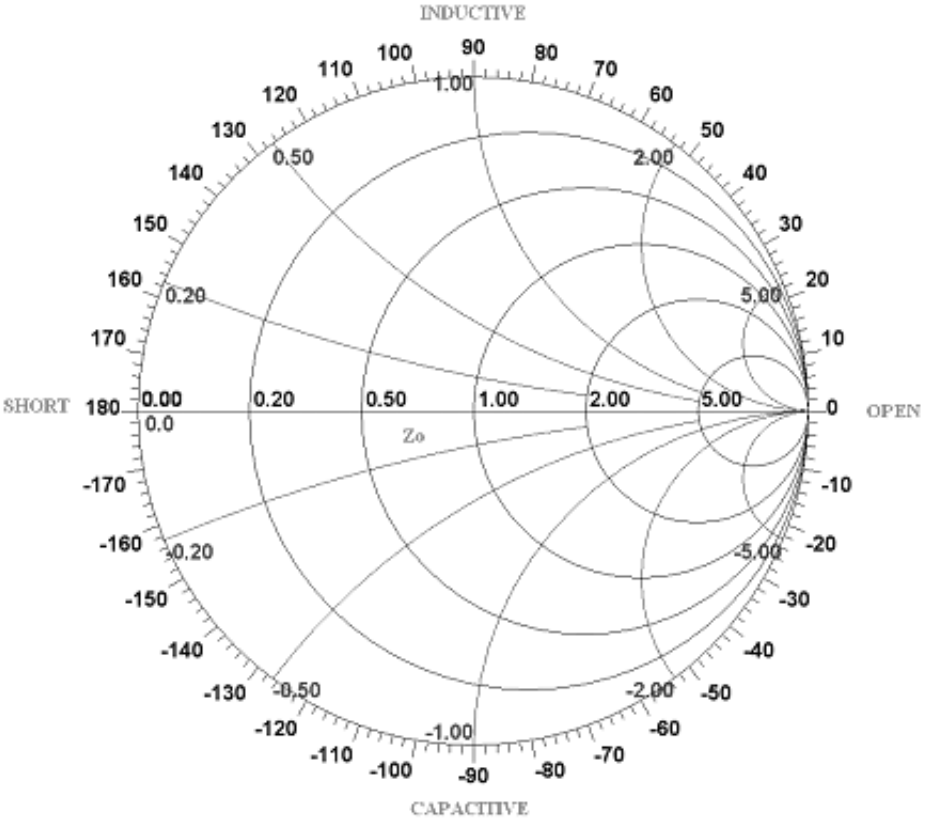
For example, to measure S_{11}, you have to ensure that $a_2 = 0$. The way to do so is by making sure that there is no reflection from the load (in other words, set $Z_L = Z_o$).

S-parameters are typically measured using network analyzers.

Smith Chart

The Smith Chart is a graphical representation of the impedances in a complex plane. You can use this tool to analyze transmission line impedance matching networks and capacitive or inductive behavior of loads. A typical Smith Chart is shown in Figure 4–80. If the impedance is capacitive, it shows up in the top half of the circle; if it is inductive, it shows up in the bottom half. The far-right point in the middle represents an open circuit; the far left represents a short circuit. The middle point (label 1.0) represents a perfect load match to the characteristic impedance of the line.

Figure 4–80. Smith Chart



AC Versus DC Coupling

AC coupling refers to the use of a series capacitor on a signal to block the DC signals from going through. DC coupling refers to the case where this capacitor is not present and the signal passes through without any interruption. In AC coupling, a DC restore circuit is generally required after the capacitor to ensure that the common mode voltage requirements of the receiver are met. Sometimes, as in the Stratix GX transceiver inputs, the DC restore circuitry is built into the device. In that case, external DC restore circuitry is not necessary. DC coupling works only in cases where the output common mode voltage of the transmitter is in the required range of the input common mode voltage of the receiver.

The advantage of AC coupling is that it allows chips with different common mode voltages to interface with each other. The disadvantage is that it requires an extra capacitor, which can add some jitter or other degradation if not properly selected.

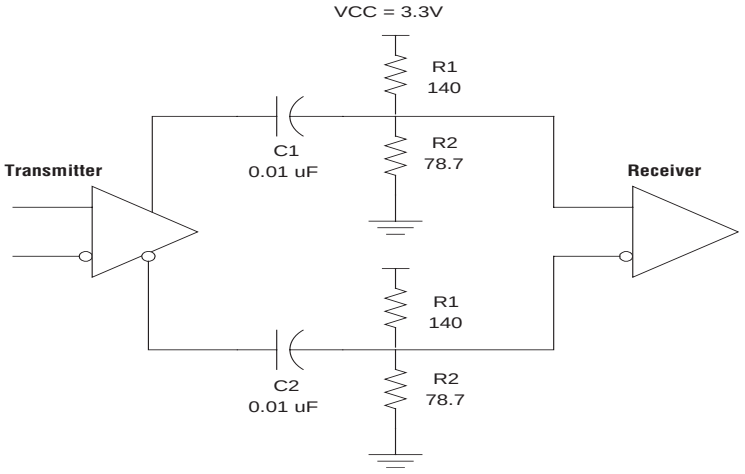
If you are certain that the common mode voltage requirements of the receiver are a subset of the common mode voltage output of the transmitter, use DC coupling. If you are in a borderline case, or if the requirements are not satisfied, then use AC coupling.

In choosing the value of the coupling capacitor, consider what happens if the capacitor is too big or too small. If the capacitor is too big, it can significantly slow the signal down and also responds poorly to fast changing input signals due to the long charge and discharge times. If the capacitor is too small, it presents a fair deal of impedance and can increase attenuation and change the characteristic impedance of the path. A good balance between these two conflicting requirements is a 0.01 µF capacitor, which Altera uses for its 3.125 Gbps transceiver designs.

When selecting components, use the smallest size possible, because smaller size components have smaller size pads, which reduce the discontinuity. Altera has used 0402 components (40 mils by 20 mils) in its designs.

You can design the DC restore circuitry in a variety of ways. Altera typically uses a simple resistive voltage divider (see Figure 4–81). Be sure to use precision resistors (0.1% or 1%) for differential signals, so that the restored DC levels on the positive and negative signals are very closely matched. In Figure 4–81, the DC restore circuitry restores the DC level to 3.3 * 78.7/(140 + 78.7) = 1.1875 Volts.

Figure 4–81. AC Coupling



Stratix GX devices have DC biasing on the high-speed transceivers inputs and reference clock inputs (REFCLKB[17..13]n and REFCLKB[17..13]p) designed for the 1.5-V PCML standard, so AC coupling is not required. This saves components and board space. If you are using other I/O standards such as LVPECL or LVDS, then you need to AC-couple them, because their common mode voltage is different from the 1.5-V PCML common mode voltage. External biasing networks are not needed, because the common mode is generated internally in the device. Altera used external biasing network on the Stratix GX development board for evaluation purpose only.

Unused Pin Connections

The unused I/O pins should be driven to ground. The unused clock inputs should be grounded as well, but they can be left floating. Similarly, the VCC/GND pins for unused PLLs should be tied to their respective supplies and grounds. Refer to the pin tables of the particular devices for more details.

Power Trace Thickness

Power traces must be carefully specified to ensure that they can deliver the required currents to the destination with minimal voltage drop and minimal temperature rise. This requires the traces to have a certain thickness. Calculators are available to calculate the required trace width to support a certain amount of current at a specified temperature rise. You can find one such calculator at

http://www.geocities.com/CapeCanaveral/Lab/9643/TraceWidth.htm. A good layout designer should also have charts and calculators for this purpose.

5. Quartus II Software Fitter Warnings

SGX53002-1.0

Introduction

The Stratix® GX device does not have delay information for the following paths:

- Transmitter PLL input clock to `coreclk_out`
- Reference clock pin (`inclk`) to the transmitter PLL
- Reference clock pin (`rx_cruclk`) to `rx_clkout` (recovered clock)
- Reference clock pin (`rx_cruclk`) to the receiver PLL
- Interquad (IQ) lines (in a multi-transceiver block application, if the clocking scheme utilizes interquad lines for clock distribution or routing, IQ lines are not modeled)

The Fitter in the Quartus® II software generates warning messages for paths that involve these clocks or IQ lines. The warning messages state clearly that the connections are not recommended because accurate delay information is not available.

There are three starting points for the checks:

- Transmitter PLL `coreclk_out`
- Receiver PLL `rx_clkout`
- IQ lines used for clocking the reference clock of transceivers (`refclkb` pins)

The warnings are specific to any path with one of the three starting points. The warnings appear in the Fitter's processing log and are associated with unreliable timing paths that contain the following:

- Incorrect clock to output time (t_{CO})
- Incorrect propagation delay (t_{PD})
- Incorrect register-to-register timing

The warnings generated by the Fitter fall into four categories and are associated with the following:

- Signals that originate from the GXB connected to an IO pin or a signal that originates from the GXB that feeds through a register to an IO pin
- Data that originates from a pin that is clocked by a clock that originates from the GXB

- A clock from one of the starting points feeds a register that is involved in a register-to-register transfer and the destination register is being clocked by a clock related to the starting point
- IQ delay line warnings because the design uses REFCLKB pins for reference clocks when using multiple transceivers in the device



Altera® strongly recommends that you review and resolve the warnings generated by the Fitter in the Quartus II software.

If the starting point is the transmitter PLL, the Fitter in the Quartus II software generates the warning messages for the following configurations:

- t_{PD} warning messages
  The starting point (GXB Transmitter PLL coreclk) feeds an I/O pin directly. This configuration causes the Fitter to incorrectly report t_{PD}. Figure 5–1 shows the t_{PD} (propagation delay) from inclk to the transmitter PLL through coreclk_out to the IO pin. Some example warnings are provided below.

  Warning: Clock connectivity to I/O pins from a GXB transmitter PLL coreclk is not recommended

  Warning: I/O pin coreclk_out is fed by GXB transmitter PLL<*design-specific path*>

- t_{CO} warning messages
  The starting point feeds an I/O register directly. This configuration causes the Fitter to incorrectly report t_{CO}. Clock connectivity to I/O pins through registers originating from a GXB transmitter PLL coreclk is not recommended (see Figure 5–1). In this case, coreclk_out from the transmitter PLL clocks a register (inst3) and the output of that register is connected directly to an output pin. Because coreclk_out is related to inclk and there is no delay information from inclk through the transmitter PLL to coreclk_out, the Fitter reports incorrect t_{CO}.

  The Fitter flags a warning if any similar type of connectivity occurs. An example warning is shown below.

  Warning: Register inst3 clocked by GXB transmitter PLL<*design-specific path*>

- t_{SU} and t_H messages:
  The starting point feeds a register that is involved in a register-to-register transfer from a register that is clocked by a clock related to the starting point. In this configuration, the Fitter performs

incorrect register-to-register analysis. This could result in the Fitter providing incorrect t_{SU} (setup time) and t_H (hold time). Register-to-register transfer between different clock domains is not recommended if one of the clocks is from either the GXB transmitter PLL (`coreclk`) or from the receiver PLL (`rx_clkout`). In the example warning shown below, there is a clock domain crossing between `inclk` (the starting point) and `coreclk_out` (the clock related to the starting point)

Warning: Data is transferred from source register inst4 (clocked by clock `inclk`) to destination register inst3 (clocked by the GXB transmitter PLL<*design-specific path*>.

Figure 5–1 highlights the paths that trigger warning messages from the Fitter. The Fitter reports an incorrect path delay for these paths.

t_{CO} delays are also incorrect for these paths (any register that is being clocked by `coreclk_out`).

The Fitter generates register-to-register warning messages because of the clock domain crossing from `inclk` (input clock to the transmitter PLL) to `coreclk_out` (output of the transmitter PLL).

Figure 5–1. Transmit PLL coreclk_out Showing t_{PD} & t_{CO} Registers

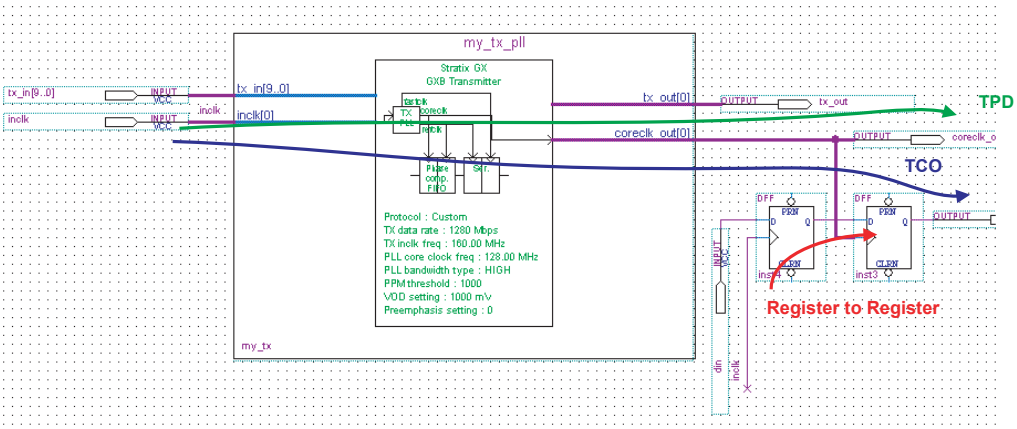


Figure 5–2 shows the t_{PD} (propagation delay) from `rx_cruclk` through the receiver PLL, through `rx_clkout`, and to the I/O pin. The Fitter reports an incorrect path delay for this path.

t_{CO} delays are also incorrect for this path (any register that is clocked by `rx_clkout` (inst3 in Figure 5–2)).

The `rx_clkout` is related to the `rx_cruclk` (receive input reference clock).

The Fitter generates register-to-register warning messages because of the clock domain crossing from `rx_cruclk` (input clock to the receiver PLL) to `rx_clkout` (output of the receiver PLL). This configuration is shown in Figure 5–2, with data transfer from register 1 (inst4) to register 2 (inst3).

Figure 5–2. Receiver PLL rx_clkout Showing t_{PD} & t_{CO} Registers

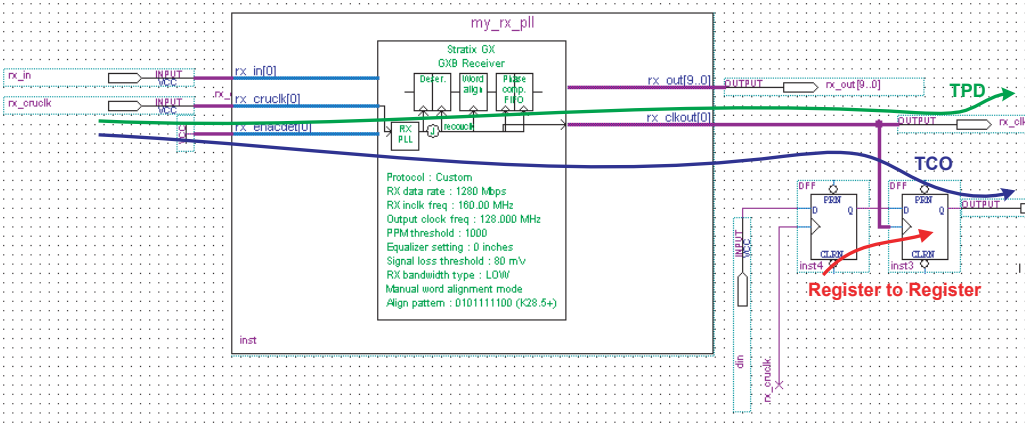
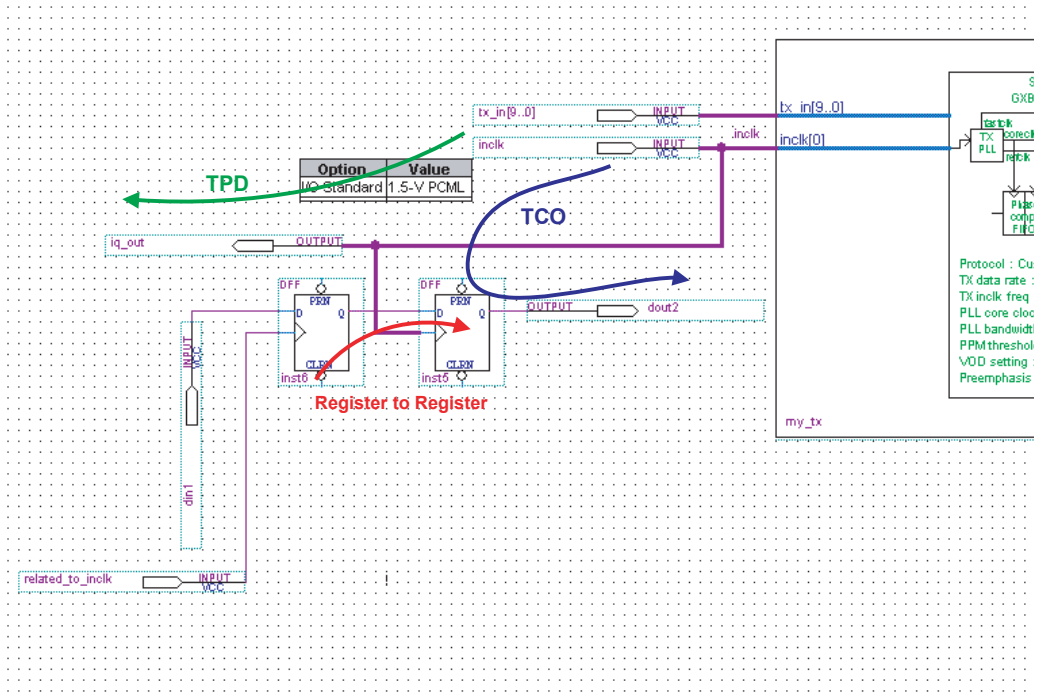


Figure 5–3 shows the usage of IQ lines and multi-transceiver blocks. It shows the t_{PD} (propagation delay) from `inclk` to the `iq_out` output pin. The Fitter reports an incorrect path delay for this path.

The Fitter also reports incorrect t_{CO} (for the path using the same `inclk` as the clock that drives an output pin).

The Fitter also issues a register-register warning because of the clock domain crossing from the clock related to `inclk` to `inclk`.

Figure 5–3. Multi-Transceiver Block Clocking Using IQ Lines Showing t_{PD} & t_{CO} Registers



Suppressing Fitter Warnings

Altera strongly recommends that you review the paths that generate the warning messages. If you decide on this connectivity for design purposes after evaluating the warnings, you can suppress the Fitter warnings.

Use the **Cut** command (in the Assignment Editor for the specific path) to suppress these warnings. The command suppresses the messages in the Fitter and instructs the Timing Analyzer to not perform timing analysis on these paths.

Design Suggestions

This section offers some possible design solutions to help you avoid the paths that cause errors and generate warnings.

Register-to-Register Warnings

If the warning messages generated by the Fitter are of type register-to-register, you should revisit the clock scheme and data path to effectively decouple different clock domains. You do this by clock-decoupling logic and FIFOs as required.

Registers Originating From GXB Connected to IO Pins

Avoid this connectivity as much as possible. If the clocking scheme cannot be changed because of design requirements, then you must resolve this issue at the pin level of the receive device. The receiver on the other end must include some type of dynamic phase adjustment of the clock with respect to the data bus.

In some cases, you might monitor certain GXB signals for test and debug purposes. If you feed these signals to an I/O pin for monitoring, you can suppress the warnings by using the **Cut** command.

Using REFCLKB Pins for Input Reference Clock

If you assign a REFCLKB pin as the input reference clock, the Fitter automatically routes the input using IQ lines. The *Stratix GX Analog Description* chapter of the *Stratix GX Device Handbook, Volume 2* provides information on how IQ lines help in clocking multiple transceivers with one REFCLKB pin and its connectivity across multiple quads. Because of the lack of accurate delay information on IQ lines (used to route the clocks to different transceivers in the device), you should use global clocks for the input reference clock to the transceivers as much as possible (try to avoid using REFCLKB pins). Applications that are sensitive to jitter typically use IQ lines because they are exclusive routing resources for transceivers.

It is not necessary to avoid using REFCLKB pins entirely, but be aware of the issues of using IQ lines for routing input clocks.



Refer to the *REFCKLB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* before using REFCLKB pins for system clocking.



Refer to the *Stratix GX Transceiver User Guide* section of the *Stratix GX Handbook, Volume 2* for details on the various clocks and functional modes of operation of the Stratix GX device.

You can use decoupling FIFOs to address the lack of delay information. The transmit clock can be an external clock (tx_coreclk) from the PLD or an internal clock (output from the transmitter PLL). On the receive side, it can be either rx_coreclk fed from the PLD or various internal

clocks clocking out the data. In a synchronous system, the decoupling FIFOs must be able to handle phase differences between the clock domains of GXB output clocks (`rx_clkout`) and the PLD system clock. In asynchronous systems, the decoupling FIFOs must also compensate for frequency variations between the recovered clock (`rx_clkout`) and the PLD system clock. See the *Stratix GX Transceiver User Guide* section of the *Stratix GX Device Handbook, Volume 2* for information on possible clocking configurations.

PLD-to-Transceiver Clocking

Figure 5–4 shows a system-level clocking scheme you can use to work around the issues described in this chapter. This clocking scheme may not be the most optimal for designs that have tight jitter requirements because of cascading PLLs.

Figure 5–4. PLD & Transceiver Clocking Scheme

