Arria II GX Device Handbook

Volume 1

101 Innovation Drive
San Jose, CA 95134
www.altera.com

AIIGX5V1-1.0

Contents

Chapter Revision Dates

The chapters in this book, *Arria II 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	Arria II GX Device Family Overview	
	Revised:	*February 2009*
	Part Number:	*AIIGX51001-1.0*
Chapter 2	Logic Array Blocks and Adaptive Logic Modules in Arria II GX Devices	
	Revised:	*February 2009*
	Part Number:	*AIIGX51002-1.0*
Chapter 3	Memory Blocks in Arria II GX Devices	
	Revised:	*February 2009*
	Part Number:	*AIIGX51003-1.0*
Chapter 4	DSP Blocks in Arria II GX Devices	
	Revised:	*February 2009*
	Part Number:	*AIIGX51004-1.0*
Chapter 5	Clock Networks and PLLs in Arria II GX Devices	
	Revised:	*February 2009*
	Part Number:	*AIIGX51005-1.0*
Chapter 6	I/O Features in Arria II GX Devices	
	Revised:	*February 2009*
	Part Number:	*AIIGX51006-1.0*
Chapter 7	External Memory Interfaces in Arria II GX Devices	
	Revised:	*February 2009*
	Part Number:	*AIIGX51007-1.0*
Chapter 8	High-Speed Differential I/O Interfaces and DPA in Arria II GX Devices	
	Revised:	*February 2009*
	Part Number:	*AIIGX51008-1.0*
Chapter 9	Configuration, Design Security, and Remote System Upgrades in Arria II GX Devices	
	Revised:	*February 2009*
	Part Number:	*AIIGX51009-1.0*
Chapter 10	SEU Mitigation in Arria II GX Devices	
	Revised:	*February 2009*
	Part Number:	*AIIGX51010-1.0*
Chapter 11	JTAG Boundary-Scan Testing	
	Revised:	*February 2009*
	Part Number:	*AIIGX51011-1.0*
Chapter 12	Power Requirements for Arria II GX Devices	

Revised:	*February 2009*
Part Number:	*AIIGX51012-1.0*

Section I. Device Core

This section provides a complete overview of all features relating to the Arria® II GX device family, the industry's first cost-optimized 40 nm FPGA family. This section includes the following chapters:

Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in this volume.

1. Arria II GX Device Family Overview

AIIGX51001-1.0

Introduction

The Arria® II GX device family is designed specifically for ease-of-use. The cost-optimized, 40-nm device family architecture features a low-power, programmable logic engine, and streamlined transceivers and I/Os. Common interfaces, such as the PCI Express (PIPE), Ethernet, and DDR-2 memory is easily implemented in your design with the Quartus® II software, SOPC Builder design software, Altera's broad library of hard-IP and soft-IP solutions. The Arria II GX device family makes designing for applications requiring transceivers operating at up to 3.75 Gbps fast and easy.

This chapter contains the following sections:

- "Highlights" on page 1–1
- "Arria II GX Device Architecture" on page 1–4
- "Reference and Ordering Information" on page 1–11

Highlights

The Arria II GX device features consist of the following highlights:

- 40 nm, low-power FPGA engine
  - Adaptive logic module (ALM) offers the highest logic efficiency in the industry
  - Eight-input fracturable look-up table (LUT)
  - Memory logic array blocks (MLABs) for efficient implementation of small FIFOs
- High-performance digital signal processing (DSP) blocks up to 350 MHz
  - Configurable as 9 × 9-bit, 12 × 12-bit, 18 × 18-bit, and 36 × 36-bit full precision multipliers
  - Hardcoded adders, subtractors, accumulators, and summation functions
  - Fully integrated design flow with MatLab and Altera's DSP Builder software
- Maximum system bandwidth
  - Up to 16 full-duplex clock data recovery (CDR)-based transceivers supporting rates between 155 Mbps and 3.75 Gbps
  - Dedicated circuitry to support physical layer functionality for popular serial protocols, including PCI Express (PIPE) Gen 1, Gigabit Ethernet, Serial RapidIO, Common Public Radio Interface (CPRI), Open Base Station Architecture Initiative (OBSAI), SD/HD/3G SDI, XAUI, HiGig/HiGig+, and SONET/SDH
- Complete PCI Express (PIPE) protocol solution with an embedded hard IP block that provides PHY-MAC layer, Data Link layer, and Transaction layer functionality

- Optimized for high-bandwidth system interfaces
  - Up to 612 user I/O pins arranged in up to 12 modular I/O banks that support a wide range of single-ended and differential I/O standards
  - High-speed LVDS I/O support with serializer/deserializer (SERDES) and dynamic phase alignment (DPA) circuitry at data rates from 150 Mbps to 1 Gbps
- Low power
  - Patented architectural power reduction techniques
  - Per-channel transceiver power consumption is approximately 100 mW under typical conditions at 3.125 Gbps
  - Power optimizations integrated into the Quartus II development software
- Advanced usability and security features
  - Parallel and serial configuration options
  - On-chip series and differential I/O termination
  - 256-bit advanced encryption standard (AES) programming file encryption for design security with volatile and non-volatile key storage options
  - Robust portfolio of IP for processing, serial protocols, and memory interfaces
  - Low cost, easy-to-use development kits featuring high-speed mezzanine connectors (HSMC)

Table 1–1 shows Arria II GX device features.

Table 1–1. Arria II GX Device Features (Part 1 of 2)

Feature	EP2AGX20	EP2AGX30	EP2AGX45	EP2AGX65	EP2AGX95	EP2AGX125	EP2AGX190	EP2AGX260
ALMs	6,380	10,800	18,050	25,300	37,470	49,640	76,120	102,600
LEs	15,950	27,000	45,125	63,250	93,675	124,100	190,300	256,500
PCI Express hard IP Blocks	1	1	1	1	1	1	1	1
M9K Blocks	87	144	319	495	612	730	840	950
Total Embedded Memory in M9K Blocks (Kbits)	783	1296	2871	4455	5508	6570	7560	8550
Total On-Chip Memory (M9K + MLABs) (Kbits)	982	1,634	3,435	5,246	6,679	8,121	9,939	11,756
Embedded Multipliers (18 × 18)	56	128	232	312	448	576	656	736
General Purpose PLLs	4	4	4	4	6	6	6	6
Transceiver Tx PLLs *(2)*	2	2	2 or 4 *(1)*	2 or 4 *(1)*	4 or 6 *(1)*	4 or 6 *(1)*	6 or 8 *(1)*	6 or 8 *(1)*

Table 1–1. Arria II GX Device Features (Part 2 of 2)

Feature	EP2AGX20	EP2AGX30	EP2AGX45	EP2AGX65	EP2AGX95	EP2AGX125	EP2AGX190	EP2AGX260
User I/O Banks	6	6	6	6	8	8	12	12

Notes to Table 1–1:

(1) The number of PLLs depends on package. Transceiver Tx PLL count = (number of transceiver blocks) × 2.
(2) The FPGA fabric can use these PLLs if they are not being used by the transceiver.

Table 1–2 lists Arria II GX device package options and user I/O pin counts, high-speed LVDS channel counts, and transceiver channel counts for micro BGA and FineLine BGA devices.

Table 1–2. Arria II GX Device Package Options and I/O Information *(Note 1)*, *(2)*

	358-Pin Flip Chip UBGA 17 mm × 17 mm			572-Pin Flip Chip FBGA 25 mm × 25 mm			780-Pin Flip Chip FBGA 29 mm × 29 mm			1152-Pin Flip Chip FBGA 35 mm × 35 mm		
Device	**I/O**	**LVDS *(3)***	**XCVRs**	**I/O**	**LVDS *(3)***	**XCVRs**	**I/O**	**LVDS *(3)***	**XCVRs**	**I/O**	**LVDS *(3)***	**XCVRs**
EP2AGX20	↑156	33Rx + 32Tx	4	↑252	57Rx + 56Tx	4	—	—	—	—	—	—
EP2AGX30	156	33Rx + 32Tx	4	252	57Rx + 56Tx	4	—	—	—	—	—	—
EP2AGX45	156	33Rx + 32Tx	4	252	57Rx + 56Tx	8	↑364	85Rx + 84Tx	8	—	—	—
EP2AGX65	↓156	33Rx + 32Tx	4	252	57Rx + 56Tx	8	364	85Rx + 84Tx	8	—	—	—
EP2AGX95	—	—	—	260	57Rx + 56Tx	8	372	85Rx + 84Tx	12	↑452	105Rx + 104Tx	12
EP2AGX125	—	—	—	↓260	57Rx + 56Tx	8	372	85Rx + 84Tx	12	452	105Rx + 104Tx	12
EP2AGX190	—	—	—	—	—	—	372	85Rx + 84Tx	12	612	145Rx + 144Tx	16
EP2AGX260	—	—	—	—	—	—	↓372	85Rx + 84Tx	12	↓612	145Rx + 144Tx	16

Notes to Table 1–2:

(1) The user I/O counts include clock pins.
(2) The arrows indicate packages vertical migration capability. Vertical migration allows you to migrate to devices whose dedicated pins, configuration pins, and power pins are the same for a given package across device densities.
(3) Rx denotes LVDS input with OCT RD support or pseudo LVDS output. Tx denotes LVDS I/O without OCT RD support or pseudo LVDS output.

Arria II GX devices are available in up to three speed grades: –4 (fastest), –5, and –6 (slowest). Table 1–3 shows Arria II GX devices speed grades.

Table 1–3. Arria II GX FPGA Devices Speed Grades (Part 1 of 2)

Device	358-Pin Flip Chip UBGA	572-Pin Flip Chip FBGA	780-Pin Flip Chip FBGA	1152-Pin Flip Chip FBGA
EP2AGX20	C4, C5, C6, I5	C4, C5, C6, I5	—	—
EP2AGX30	C4, C5, C6, I5	C4, C5, C6, I5	—	—
EP2AGX45	C4, C5, C6, I5	C4, C5, C6, I5	C4, C5, C6, I5	—
EP2AGX65	C4, C5, C6, I5	C4, C5, C6, I5	C4, C5, C6, I5	—

Table 1–3. Arria II GX FPGA Devices Speed Grades (Part 2 of 2)

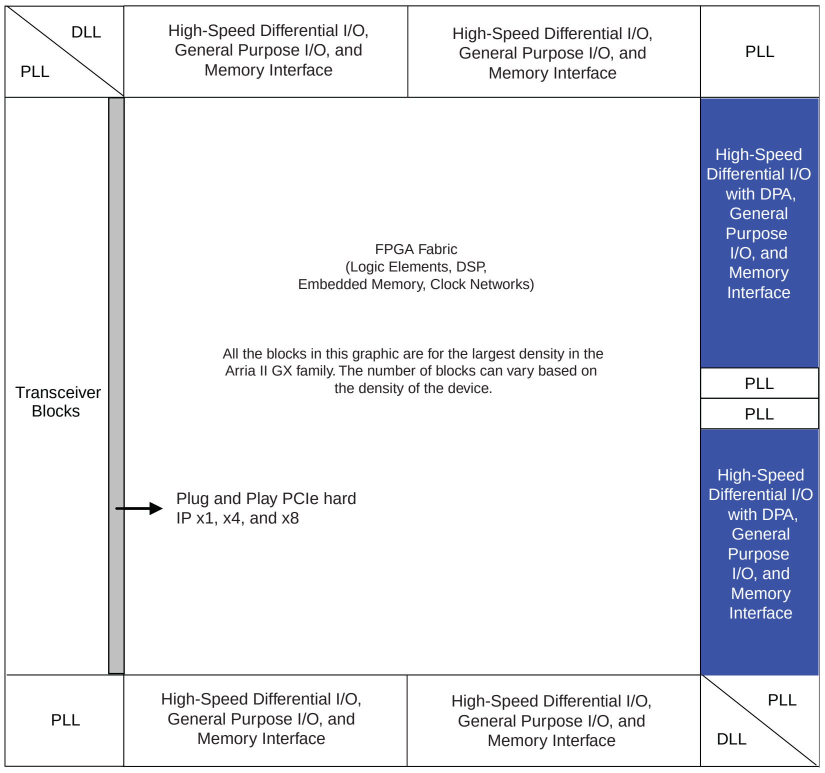
Device	358-Pin Flip Chip UBGA	572-Pin Flip Chip FBGA	780-Pin Flip Chip FBGA	1152-Pin Flip Chip FBGA
EP2AGX95	—	C4, C5, C6, I5	C4, C5, C6, I5	C4, C5, C6, I5
EP2AGX125	—	C4, C5, C6, I5	C4, C5, C6, I5	C4, C5, C6, I5
EP2AGX190	—	—	C4, C5, C6, I5	C4, C5, C6, I5
EP2AGX260	—	—	C4, C5, C6, I5	C4, C5, C6, I5

Arria II GX Device Architecture

Arria II GX FPGAs include a customer-defined feature set optimized for cost-sensitive applications and offer a wide range of density, memory, embedded multiplier, I/O, and packaging options. Arria II GX FPGAs support external memory interfaces and I/O protocols required by wireless, wireline, broadcast, computer, storage, and military markets. They inherit the 8-input advanced logic module, M9K embedded RAM block, and high-performance DSP blocks from the Stratix® IV device family with a cost-optimized I/O cell and a transceiver optimized for 3.75 Gbps speeds.

Figure 1–1 shows an overview of the Arria II GX device architecture.

Figure 1–1. Arria II GX Device Architecture Overview


DLL
PLL
High-Speed Differential I/O, General Purpose I/O, and Memory Interface
High-Speed Differential I/O, General Purpose I/O, and Memory Interface
PLL
High-Speed Differential I/O with DPA, General Purpose I/O, and Memory Interface
FPGA Fabric (Logic Elements, DSP, Embedded Memory, Clock Networks)
All the blocks in this graphic are for the largest density in the Arria II GX family. The number of blocks can vary based on the density of the device.
PLL
PLL
Transceiver Blocks
Plug and Play PCIe hard IP x1, x4, and x8
High-Speed Differential I/O with DPA, General Purpose I/O, and Memory Interface
PLL
High-Speed Differential I/O, General Purpose I/O, and Memory Interface
High-Speed Differential I/O, General Purpose I/O, and Memory Interface
PLL
DLL

High-Speed Transceiver Features

Arria II GX devices integrate up to 16 transceivers on a single device. The transceiver block is optimized for cost and power consumption. Arria II GX transceivers support the following features:

- Configurable pre-emphasis and equalization, and adjustable output differential voltage
- Flexible and easy-to-configure transceiver data path to implement proprietary protocols
- Signal Integrity Features
  - Programmable transmitter pre-emphasis to compensate for inter-symbol interference (ISI)
  - User-controlled five-stage receiver equalization with up to 7 dB of high-frequency gain
  - On-die power supply regulators for transmitter and receiver PLL charge pump and voltage-controlled oscillator (VCO) for superior noise immunity
  - Calibration circuitry for transmitter and receiver on-chip termination (OCT) resistors
- Diagnostic Features
  - Serial loopback from the transmitter serializer to the receiver CDR for transceiver PCS and PMA diagnostics
  - Parallel loopback from the transmitter PCS to the receiver PCS with built-in self test (BIST) pattern generator and verifier
  - Reverse serial loopback pre- and post-CDR to transmitter buffer for physical link diagnostics
  - Loopback master and slave capability in PCI Express hard IP blocks
  - Support for protocol features such as MSB-to-LSB transmission in SONET/SDH configuration and spread-spectrum clocking in PCI Express (PIPE) configuration

Table 1–4 shows some common protocols and the Arria II GX dedicated circuitry and features for implementing these protocols.

Table 1–4. Sample of supported Protocols and Feature Descriptions

Supported Protocols	Feature Descriptions
PCI Express (PIPE)	■ Complete PCI Express (PIPE) Gen1 protocol stack solution compliant to PCI Express Base Specification 1.1 that includes PHY-MAC, Data Link, and Transaction layer circuitry embedded in PCI Express hard IP blocks ■ ×1, ×4, and ×8 lane configurations ■ Built-in circuitry for electrical idle generation and detection, receiver detect, power state transitions, lane reversal, and polarity inversion ■ 8B/10B encoder and decoder, receiver synchronization state machine, and ±300 parts per million (ppm) clock compensation circuitry ■ Options to use: ■ Hard IP Data Link Layer and Transaction Layer ■ Hard IP Data Link Layer and custom Soft IP Transaction Layer
XAUI/HiGig/HiGig+	■ Compliant to IEEE P802.3ae specification ■ Embedded state machine circuitry to convert XGMII idle code groups (||I||) to and from idle ordered sets (||A||, ||K||, ||R||) at the transmitter and receiver, respectively ■ 8B/10B encoder and decoder, receiver synchronization state machine, lane deskew, and ±100 ppm clock compensation circuitry
GIGE	■ Compliant to IEEE 802.3 specification ■ Automatic idle ordered set (/I1/, /I2/) generation at the transmitter, depending on the current running disparity ■ 8B/10B encoder and decoder, receiver synchronization state machine, and 100 ppm clock compensation circuitry
CPRI/OBSAI	■ Reverse bit slipper eliminates latency uncertainty to comply with CPRI/OBSAI specifications ■ Optimized for power and cost for remote radio heads and RF modules

☞ For other protocols supported by Arria II GX devices, such as SONET/SDH, SDI and Serial RapidIO, refer to the *Arria II GX Transceiver Architecture* chapter in volume 2 of the *Arria II GX Device Handbook*.

The following sections provide an overview of the various features of the Arria II GX FPGA.

PCI Express (PIPE) hard IP

Every Arria II GX device includes an integrated hard IP block which implements PCI Express (PIPE) PHY-MAC, data link, and transaction layers. This PCI Express hard IP block is highly configurable to meet the requirements of the majority of PCI Express applications. PCI Express (PIPE) hard IP makes implementing a PCI Express (PIPE) Gen 1 solution in your Arria II GX design simple and easy.

PCI Express hard IP is instantiated using the PCI Compiler MegaWizard™ Plug-In Manager, similar to soft IP functions, but does not consume core FPGA resources or require placement, routing, and timing analysis to insure correct operation of the core. The Arria II GX hard IP for PCI Express (PIPE) includes support for:

- ×1, ×4, and ×8 lane configurations
- Root port and endpoint configurations
- 512 byte payload
- Compliant to PCI Express (PIPE) 1.1 at 2.5 Gbps

Logic Array Block and Adaptive Logic Modules

- Logic array blocks (LABs) consists of 10 adaptive logic modules (ALMs), carry chains, shared arithmetic chains, LAB control signals, local interconnect, and register chain connection lines
- ALMs expand the traditional four-input LUT architecture to eight-inputs, increasing performance by reducing LEs, logic levels, and associated routing
- LABs have a derivative called memory LAB (MLAB), that adds SRAM-memory capability to the LAB
- MLAB and LAB blocks always coexist as pairs, allowing up to 50% of the logic (LABs) to be traded for memory (MLABs)

Embedded Memory Blocks

- M9K embedded memory blocks provide up to 8,550 Kbits of on-chip memory capable of up to 390-MHz performance. The embedded memory structure consists of columns of M9K memory blocks that you can configure as RAM, FIFO buffers, and ROM
- Optimized for applications such as high-throughput packet processing, high-definition (HD) line buffers for video processing functions, and embedded processor program and data storage
- Quartus II software allows you to take advantage of M9K memory blocks by instantiating memory using a dedicated megafunction wizard or by inferring memory directly from VHDL or Verilog source code

Table 1–5 shows Arria II GX device memory modes.

Table 1–5. Arria II GX Memory Modes

Port Mode	Port Width Configuration
Single Port	×1, ×2, ×4, ×8, ×9, ×16, ×18, ×32, and ×36
Simple Dual Port	×1, ×2, ×4, ×8, ×9, ×16, ×18, ×32, and ×36
True Dual Port	×1, ×2, ×4, ×8, ×9, ×16, and ×18

DSP Resources

- Fulfills the digital signal processing requirements of 3G and LTE wireless infrastructure applications, video processing applications, and voice processing applications
- DSP block input registers efficiently implement shift registers for finite impulse response (FIR) filter applications
- Quartus II software includes megafunctions that are used to control the mode of operation of the DSP blocks based on user-parameter settings
- Multipliers can also be inferred directly from VHDL or Verilog HDL source code

I/O Features

- Contains up to 12 modular I/O banks
- All I/O banks support a wide range of single-ended and differential I/O standards, as listed in Table 1–6

Table 1–6. Arria II GX FPGA I/O Standards Support

Type	I/O Standard
Single-Ended I/O	LVTTL, LVCMOS, SSTL, HSTL, PCI, and PCI-X
Differential I/O	SSTL, HSTL, LVPECL, LVDS, mini-LVDS, and RSDS

- Supports programmable bus hold, programmable weak pull-up resistors, and programmable slew rate control
- Calibrates OCT or driver impedance matching for single-ended I/O standards with one OCT calibration block on the top-left, top-right, and bottom-left corners of the device
- Dedicated configuration banks at Bank 3C and 8C which support dedicated configuration pins and dual-function pins with a configuration scheme at 1.8, 2.5, 3.0, and 3.3 V
- Dedicated V_{REF} pin per I/O bank to allow voltage-referenced I/O standards. Each I/O bank can operate at independent V_{CCIO} and V_{REF} levels

High-Speed LVDS I/O with DPA and Soft CDR

- Dedicated circuitry for implementing LVDS interfaces at speeds from 150 Mbps to 1 Gbps
- On-chip differential termination for high-speed LVDS interfacing
- DPA circuitry and soft-CDR circuitry at the receiver automatically compensates for channel-to-channel and channel-to-clock skew in source-synchronous interfaces and allows for implementation of asynchronous serial interfaces with embedded clocks at data rates from 150 Mbps to 1 Gbps

Clock Management

- Provides dedicated global clock networks (GCLKs), regional clock networks (RCLKs), and periphery clock networks (PCLKs) that are organized into a hierarchical structure that provides up to 148 unique clock domains
- Up to six PLLs with seven outputs per PLL to provide robust clock management and synthesis
  - Independently programmable PLL outputs, creating a unique and customizable clock frequency with no fixed relation to any other clock
  - Inherent jitter filtration and fine granularity control over multiply and divide ratios
  - Supports spread-spectrum input clocking and counter cascading with PLL input clock frequencies ranging from 5 to 500 MHz to support both low-cost and high-end clock performance
- Unused transceiver PLLs can be used by the FPGA fabric to provide more flexibility

Auto-Calibrating External Memory Interfaces

- I/O structure enhanced to provide flexible and cost-effective support for different types of memory interfaces
- Contains features such as on-chip termination and DQS/DQ pin groupings to enable rapid and robust implementation of different memory standards
- An auto-calibrating megafunction is available in the Quartus II software for DDR SDRAM, DDR2 SDRAM, and DDR3 SDRAM memory interface PHYs; the megafunction takes advantage of the PLL dynamic reconfiguration feature to calibrate based on the changes of process, voltage, and temperature

Table 1–7 lists preliminary external memory support. Memory maximum performance is pending device characterization.

Table 1–7. Arria II GX Device External Memory Interface Maximum Performance

Memory Type	Maximum Performance
DDR SDRAM	200 MHz
DDR2 SDRAM	300 MHz
DDR3 SDRAM	300 MHz
QDRII SRAM	250 MHz

Arria II GX devices feature I/O capable of electrical support for QDRII SRAM, but Altera does not currently supply a controller or PHY megafunction for QDRII SRAM interfaces.

For more information regarding the external memory interfaces support, refer to the *External Memory Interfaces in Arria II GX Devices* chapter in volume 1 in the *Arria II GX Device Handbook*.

Nios II

- Arria II GX devices support all variants of the NIOS II processor
- Nios II processors are supported by an array of software tools from Altera and leading embedded partners and are used by more designers than any other configurable processor

Configuration Features

- Configuration
  - Meets the 200 ms wake-up time requirement for PCI Express (PIPE)
- Design Security
  - Supports programming file encryption using 256-bit volatile and non-volatile security keys to protect designs from copying, reverse engineering, and tampering in fast passive parallel (FPP) configuration mode with an external host (such as a MAX® II device or microprocessor), or when using fast active serial (AS) or passive serial (PS) configuration scheme
  - Decrypts an encrypted configuration bitstream using the AES algorithm, an industry standard encryption algorithm that is FIPS-197 certified and requires a 256-bit security key
- Remote System Upgrade
  - Allows error-free deployment of system upgrades from a remote location securely and reliably without an external controller
  - Soft logic (either the Nios II embedded processor or user logic) implementation in the device helps download a new configuration image from a remote location, store it in configuration memory, and direct the dedicated remote system upgrade circuitry to start a reconfiguration cycle
  - Dedicated circuitry in the remote system upgrade helps to avoid system down time by performing error detection during and after the configuration process, and recover from an error condition by reverting back to a safe configuration image, and provides error status information

SEU Mitigation

- Offers built-in error detection circuitry to detect data corruption due to soft errors in the configuration random access memory (CRAM) cells
- Allows all CRAM contents to be read and verified to match a configuration-computed cyclical redundancy check (CRC) value
- The bit location and the type of soft error can be identified and read-out through the Joint Test Action Group (JTAG) or the core interface

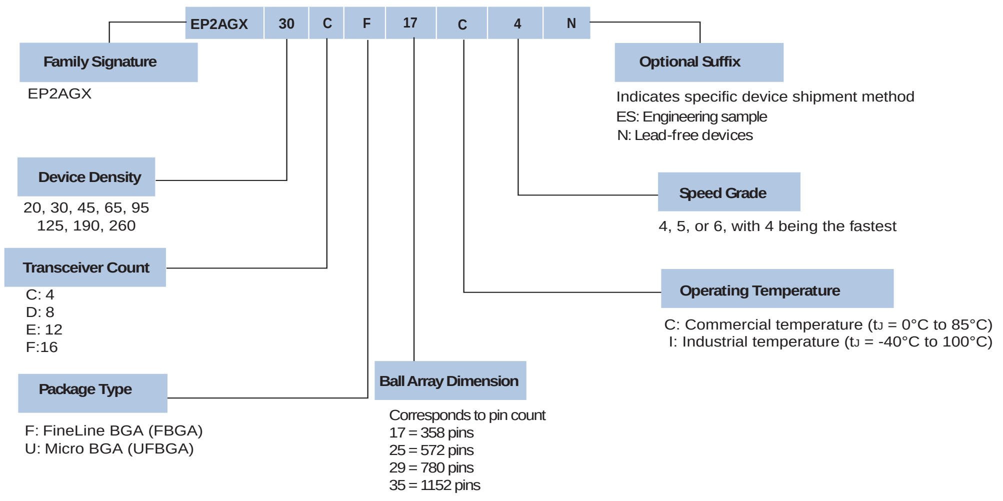
JTAG Boundary Scan Testing

- Supports JTAG IEEE Std. 1149.1 and IEEE Std. 1149.6 specifications
- IEEE Std. 1149.6 supports high-speed serial interface (HSSI) transceivers and performs boundary scan on alternating current (AC)-coupled transceiver channels
- Boundary-scan test (BST) architecture offers the capability to test pin connections without using physical test probes and capture functional data while a device is operating normally

Reference and Ordering Information

Figure 1–2 describes the ordering codes for Arria II GX devices.

Figure 1–2. Arria II GX Device Packaging Ordering Information



Document Revision History

Table 1–8 shows the revision history for this document.

Table 1–8. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial Release.	—

2. Logic Array Blocks and Adaptive Logic Modules in Arria II GX Devices

AIIGX51002-1.0

Introduction

This chapter describes the features of the logic array block (LAB) in the Arria® II GX core fabric. The logic array block is composed of basic building blocks known as adaptive logic modules (ALMs) that you can configure to implement logic functions, arithmetic functions, and register functions.

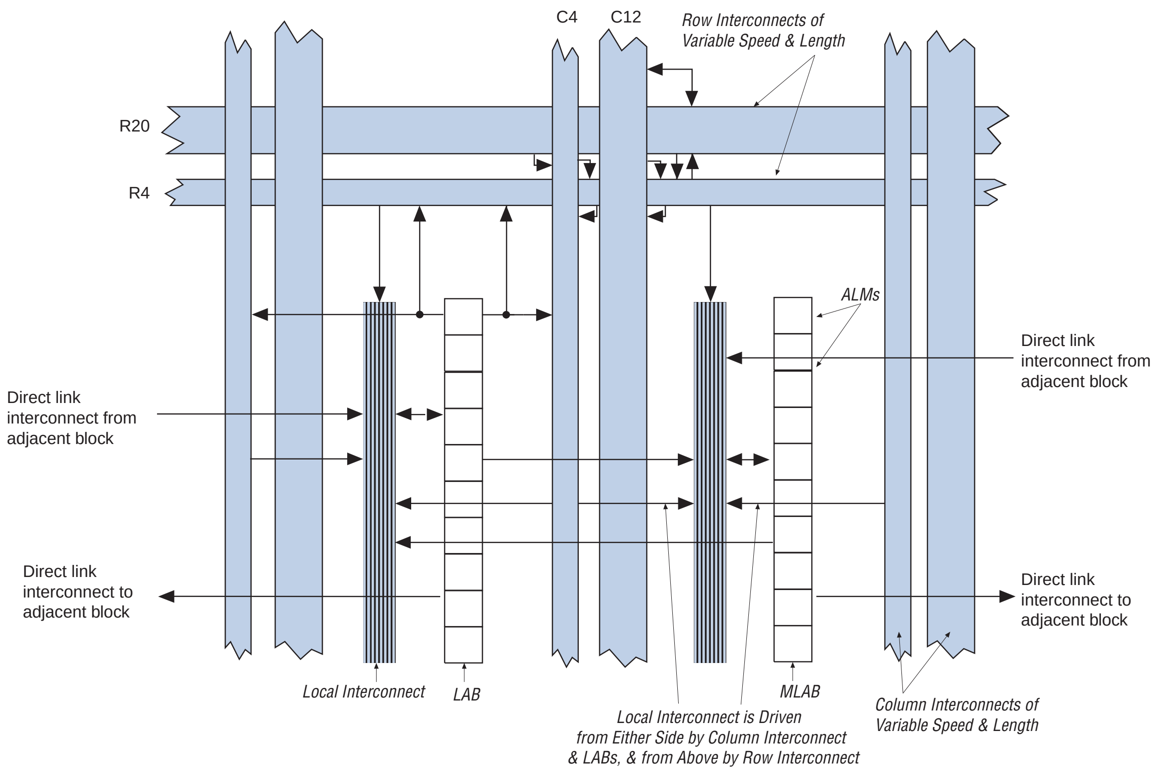
This chapter contains the following sections:

- "Logic Array Blocks" on page 2–1
- "Adaptive Logic Modules" on page 2–4

Logic Array Blocks

Figure 2–1 shows the Arria II GX LAB structure and the LAB interconnects.

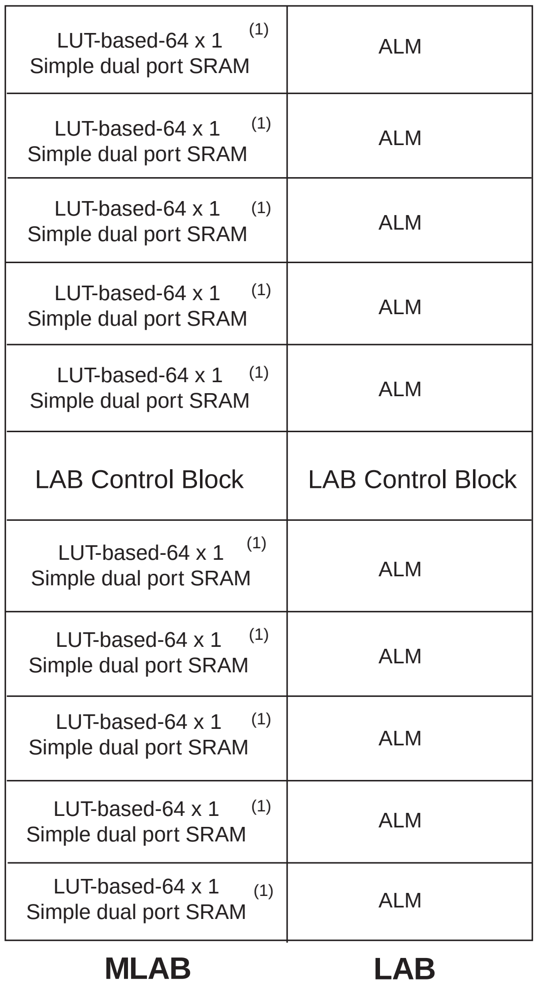
Figure 2–1. Arria II GX LAB Structure


C4
C12
Row Interconnects of Variable Speed & Length
R20
R4
ALMs
Direct link interconnect from adjacent block
Direct link interconnect from adjacent block
Direct link interconnect to adjacent block
Direct link interconnect to adjacent block
Local Interconnect
LAB
MLAB
Local Interconnect is Driven from Either Side by Column Interconnect & LABs, & from Above by Row Interconnect
Column Interconnects of Variable Speed & Length

Half of the available LABs in Arria II GX devices can be used as a memory LAB (MLAB). The MLAB supports a maximum of 640 bits of simple dual-port static random access memory (SRAM). You can configure each ALM in an MLAB as either a 64 × 1 or 32 × 2 block, resulting in a configuration of 64 × 10 or 32 × 20 simple dual-port SRAM blocks. MLAB and LAB blocks always coexist as pairs in all Arria II GX device families. MLAB is a superset of the LAB and includes all LAB features. Figure 2–2 shows an overview of LAB and MLAB topology.

MLAB is described in detail in the *Memory Blocks in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

Figure 2–2. Arria II GX LAB and MLAB Structure


LUT-based-64 x 1 Simple dual port SRAM (1)
ALM
LUT-based-64 x 1 Simple dual port SRAM (1)
ALM
LUT-based-64 x 1 Simple dual port SRAM (1)
ALM
LUT-based-64 x 1 Simple dual port SRAM (1)
ALM
LUT-based-64 x 1 Simple dual port SRAM (1)
ALM
LAB Control Block
LAB Control Block
LUT-based-64 x 1 Simple dual port SRAM (1)
ALM
LUT-based-64 x 1 Simple dual port SRAM (1)
ALM
LUT-based-64 x 1 Simple dual port SRAM (1)
ALM
LUT-based-64 x 1 Simple dual port SRAM (1)
ALM
LUT-based-64 x 1 Simple dual port SRAM (1)
ALM
MLAB
LAB

Note to Figure 2–2:

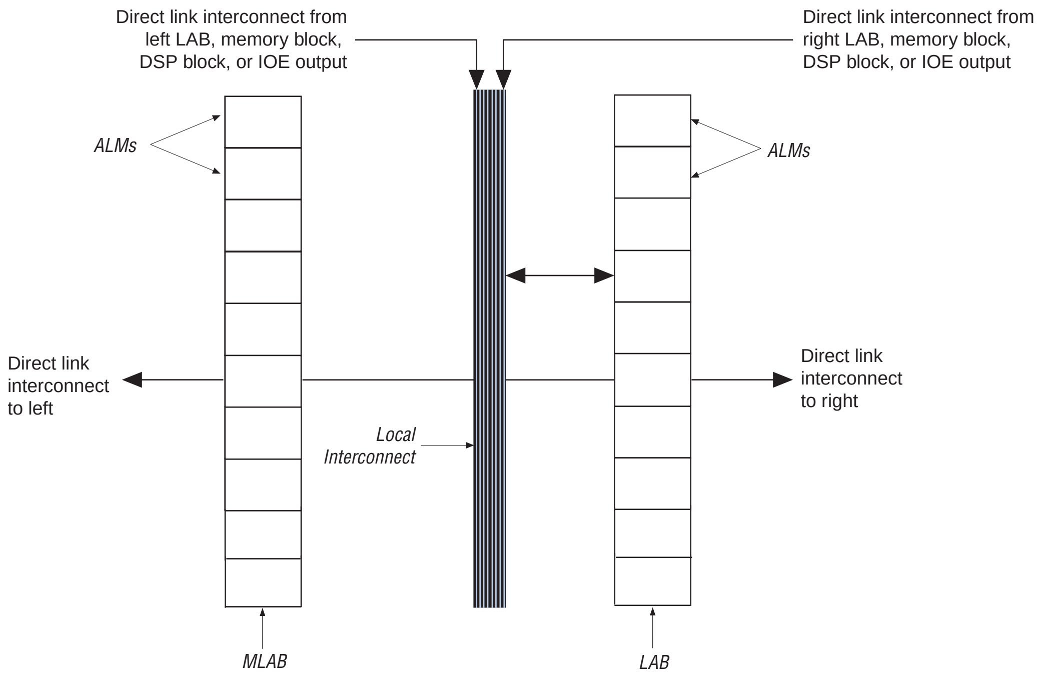
(1) You can use an MLAB ALM as a regular LAB ALM or configure it as a dual-port SRAM, as shown.

LAB Interconnects

The LAB local interconnect is used to drive ALMs in the same LAB. Each LAB can drive 30 ALMs through fast local and direct link interconnects. Ten ALMs are in any given LAB and ten ALMs are in each of the adjacent LABs.

Figure 2–3 shows the direct link connection, which connects adjacent LABs, memory blocks, DSP blocks, or I/O element (IOE) outputs.

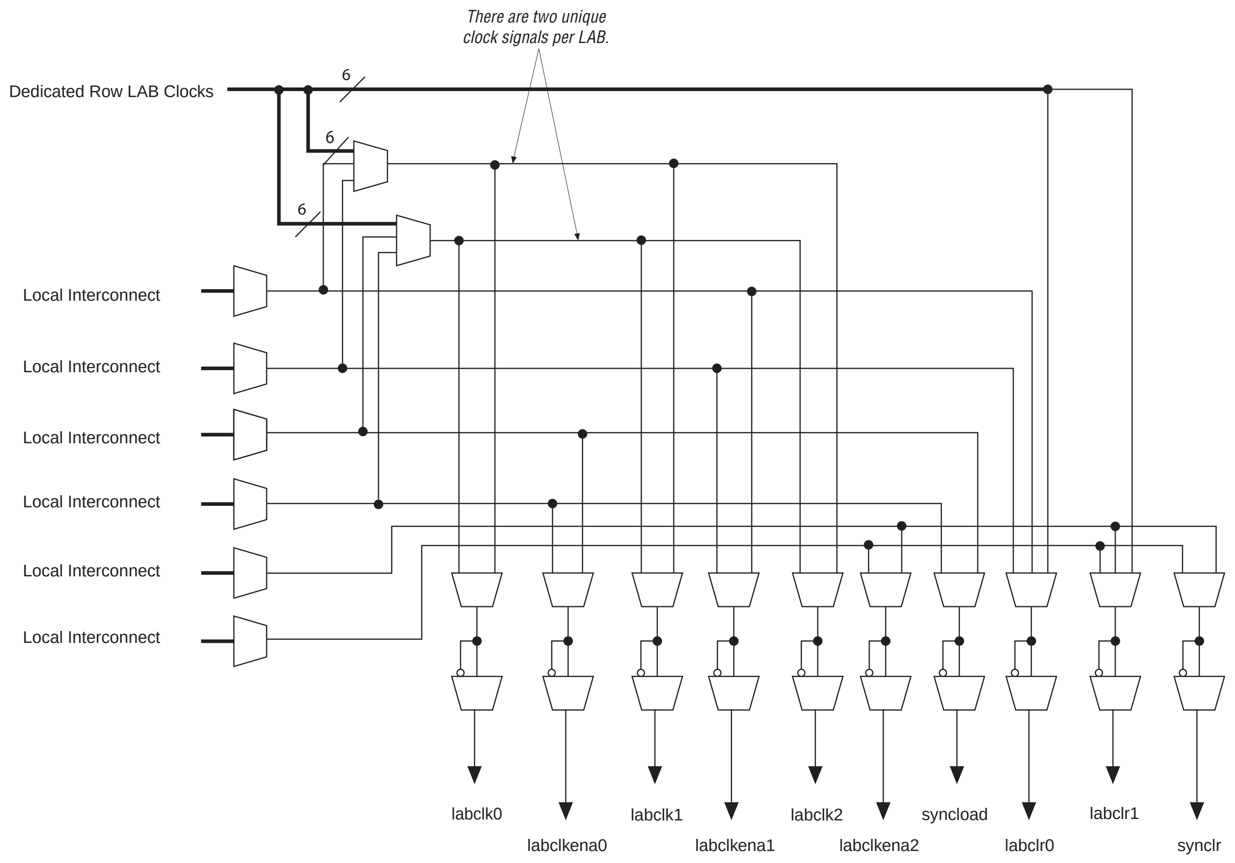
Figure 2–3. Direct Link Connection


Direct link interconnect from left LAB, memory block, DSP block, or IOE output
Direct link interconnect from right LAB, memory block, DSP block, or IOE output
ALMs
ALMs
Direct link interconnect to left
Direct link interconnect to right
Local Interconnect
MLAB
LAB

LAB Control Signals

Each LAB contains dedicated logic for driving control signals to its ALMs, and has two unique clock sources and three clock enable signals, as shown in Figure 2–4. The LAB control block can generate up to three clocks using the two clock sources and three clock enable signals. Each LAB's clock and clock enable signals are linked. De-asserting the clock enable signal turns off the corresponding LAB-wide clock.

Figure 2–4. LAB-Wide Control Signals



Adaptive Logic Modules

The ALM is the basic building block of logic in the Arria II GX architecture, providing advanced features with efficient logic utilization. One ALM can implement any function of up to six inputs and certain seven-input functions.

Each ALM drives all types of interconnects: local, row, column, carry chain, shared arithmetic chain, register chain, and direct link interconnects. Figure 2–5 shows a high-level block diagram of the Arria II GX ALM.

Figure 2–5. High-Level Block Diagram of the Arria II GX ALM

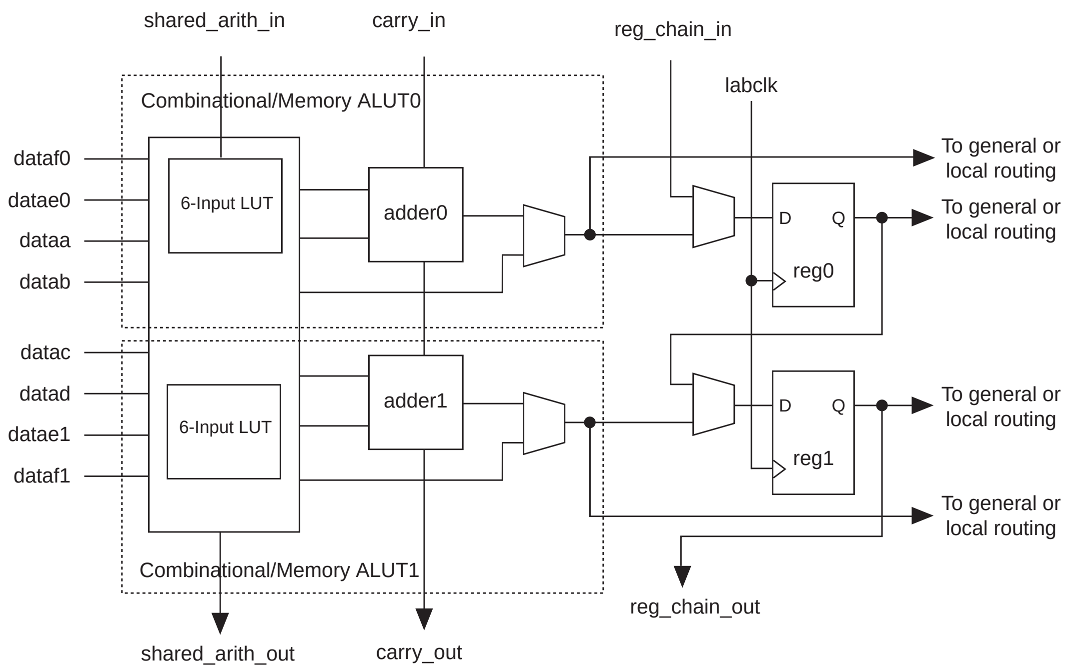
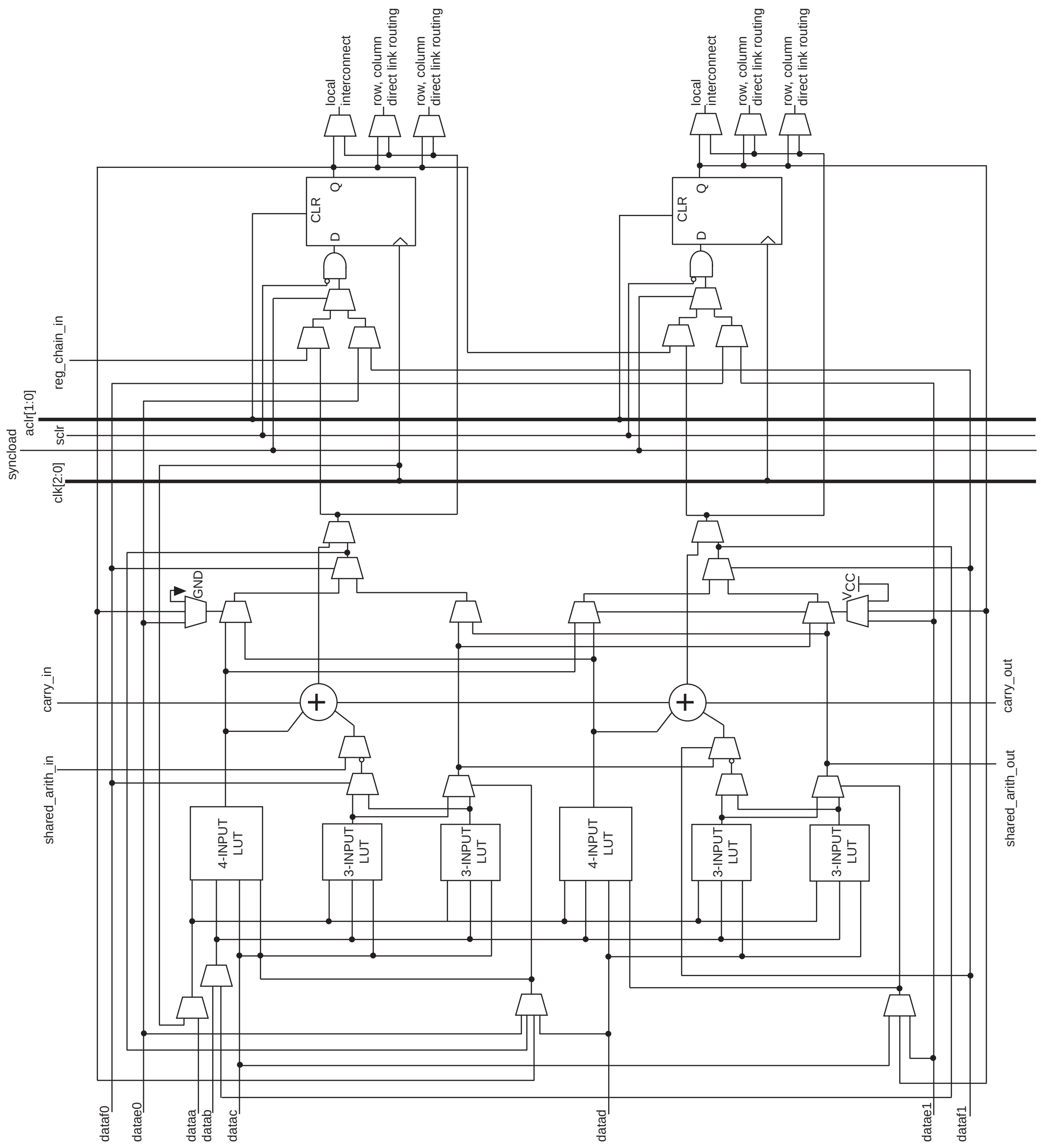


Figure 2–6 shows a detailed view of all the connections in an ALM.

Figure 2–6. Arria II GX ALM Details



The clock and clear control signals of an ALM's register can be driven by global signals, general-purpose I/O pins, or any internal logic. General-purpose I/O pins or internal logic can drive the clock enable. For combinational functions, the register is bypassed and the output of the look-up table (LUT) drives directly to the outputs of an ALM.

Each ALM has two sets of outputs that drive the local, row, and column routing resources. The LUT, adder, or register output can drive these outputs (refer to Figure 2–6). For each set of output drivers, two ALM outputs can drive column, row, or direct link routing connections, and one of these ALM outputs can also drive local interconnect resources. This allows the LUT or adder to drive one output while the register drives another output.

This feature, called register packing, improves device utilization by allowing the device to use the register and combinational logic for unrelated functions. Another mechanism to improve fitting is to allow the register output to feed back into the LUT of the same ALM so that the register is packed with its own fan-out LUT. The ALM can also drive out registered and unregistered versions of the LUT or adder output.

The Quartus® II software automatically configures the ALMs for optimized performance.

ALM Operating Modes

The Arria II GX ALM can operate in any of the following modes:

- Normal
- Extended LUT
- Arithmetic
- Shared Arithmetic
- LUT-Register

The Quartus II software and supported third-party synthesis tools, in conjunction with parameterized functions such as the library of parameterized modules (LPM) functions, automatically choose the appropriate mode for common functions such as counters, adders, subtractors, and arithmetic functions.

Normal Mode

Normal mode is suitable for general logic applications and combinational functions. In this mode, up to eight data inputs from the LAB local interconnect are inputs to the combinational logic. Normal mode allows two functions to be implemented in one Arria II GX ALM, or an ALM to implement a single function of up to six inputs. The ALM can support certain combinations of completely independent functions and various combinations of functions that have common inputs.

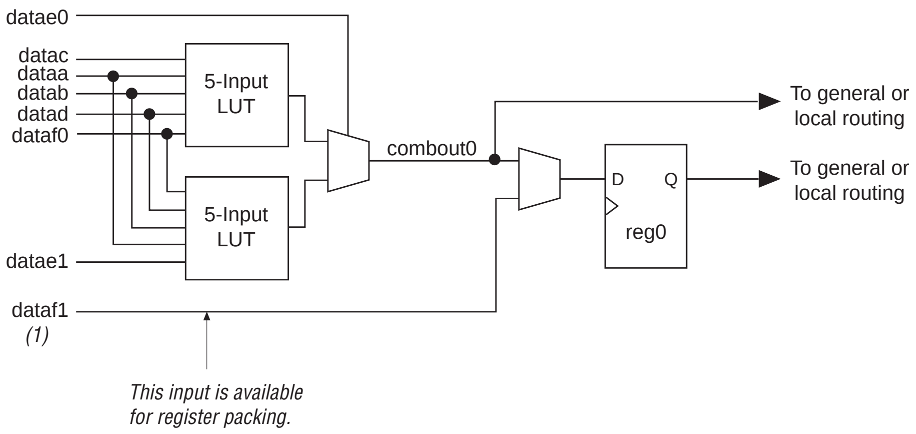
The Quartus II Compiler automatically searches for functions of common inputs or completely independent functions to be placed into one ALM and to make efficient use of the device resources.

Extended LUT Mode

Use extended LUT mode to implement a specific set of seven-input functions. The set must be a 2-to-1 multiplexer fed by two arbitrary five-input functions sharing four inputs. Figure 2–7 shows the template of supported seven-input functions utilizing extended LUT mode. In this mode, if the seven-input function is unregistered, the unused eighth input is available for register packing.

Functions that fit into the template shown in Figure 2–7 often appear in designs as “if-else” statements in Verilog HDL or VHDL code.

Figure 2–7. Template for Supported Seven-Input Functions in Extended LUT Mode


datae0
datac
dataa
datab
datad
dataf0
5-Input LUT
5-Input LUT
datae1
dataf1
(1)
combout0
D
Q
reg0
To general or local routing
To general or local routing
This input is available for register packing.

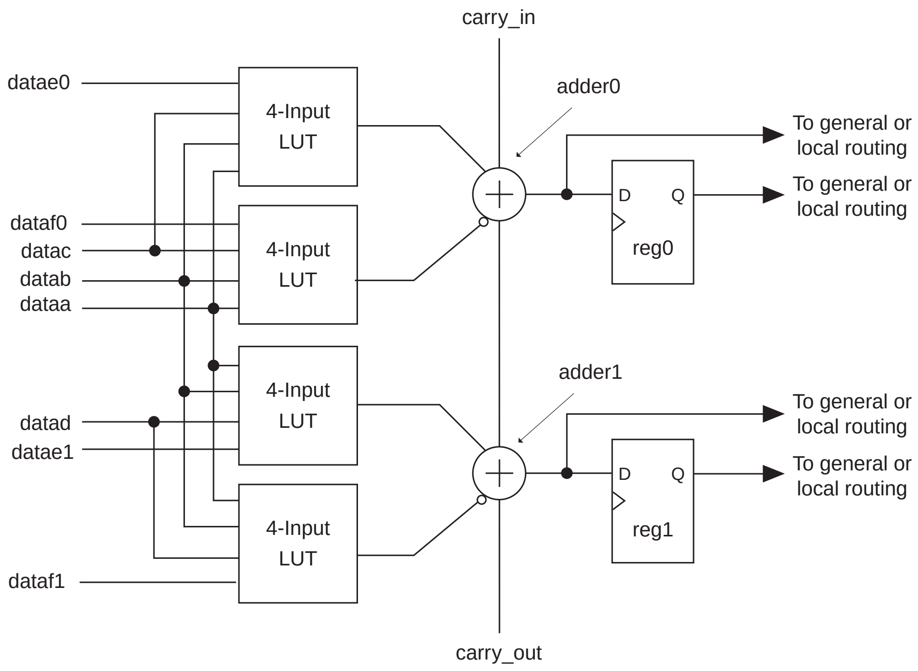
Note to Figure 2–7:

(1) If the seven-input function is unregistered, the unused eighth input is available for register packing. The second register, `reg1`, is not available.

Arithmetic Mode

Arithmetic mode is ideal for implementing adders, counters, accumulators, wide parity functions, and comparators. The ALM in arithmetic mode uses two sets of 2 four-input LUTs along with two dedicated full adders. The dedicated adders allow the LUTs to be available to perform pre-adder logic; therefore, each adder can add the output of 2 four-input functions. Figure 2–8 shows an ALM in arithmetic mode.

Figure 2–8. ALM in Arithmetic Mode



In arithmetic mode, the ALM support simultaneous use of the adder's carry output along with combinational logic outputs. In this operation, the adder output is ignored. This usage of the adder with the combinational logic output provides resource savings of up to 50%.

Arithmetic mode also offers clock enable, counter enable, synchronous up and down control, add and subtract control, synchronous clear, and synchronous load. The LAB local interconnect data inputs generate the clock enable, counter enable, synchronous up and down, and add and subtract control signals. These control signals are good candidates for the inputs that are shared between the four LUTs in the ALM. The synchronous clear and synchronous load options are LAB-wide signals that affect all registers in the LAB. These signals can also be individually disabled or enabled per register. The Quartus II software automatically places any registers that are not used by the counter into other LABs.

Carry Chain

The carry chain provides a fast carry function between the dedicated adders in arithmetic or shared arithmetic mode. The two-bit carry select feature in Arria II GX devices halves the propagation delay of carry chains within the ALM. Carry chains can begin in either the first ALM or the fifth ALM in a LAB. The final carry-out signal is routed to an ALM, 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 library of parameterized modules (LPM) automatically take advantage of carry chains for the appropriate functions.

The Quartus II Compiler creates carry chains longer than 20 ALMs (10 ALMs in arithmetic or shared arithmetic mode) 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.

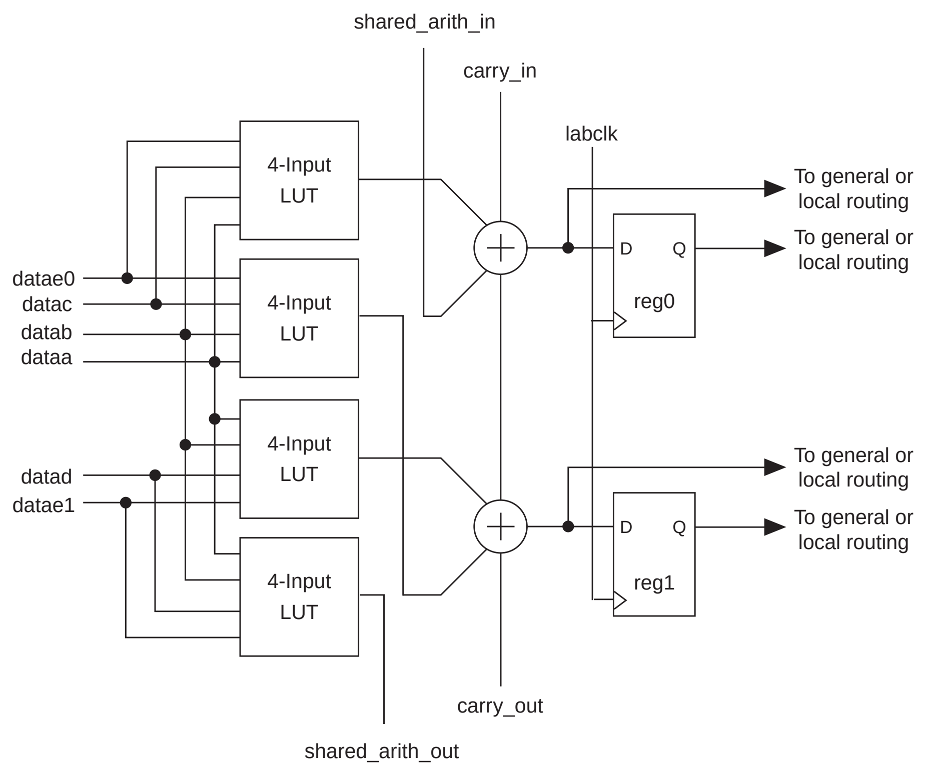
To avoid routing congestion in one small area of the device when a high fan-in arithmetic function is implemented, the LAB can support carry chains that only use either the top half or bottom half of the LAB before connecting to the next LAB. This leaves the other half of the ALMs in the LAB available for implementing narrower fan-in functions in normal mode. Carry chains that use the top five ALMs in the first LAB carry into the top half of the ALMs in the next LAB in the column. Carry chains that use the bottom five ALMs in the first LAB carry into the bottom half of the ALMs in the next LAB within the column. In every alternate LAB column, the top half can be bypassed; in the other MLAB columns, the bottom half can be bypassed.

☞ For more information on carry chain interconnect, refer to "ALM Interconnects" on page 2–13.

Shared Arithmetic Mode

In shared arithmetic mode, the ALM can implement a three-input add in an ALM. In this mode, the ALM is configured with 4 four-input LUTs. Each LUT either computes the sum of three inputs or the carry of three inputs. The output of the carry computation is fed to the next adder using a dedicated connection called the shared arithmetic chain. This shared arithmetic chain can significantly improve the performance of an adder tree by reducing the number of summation stages required to implement an adder tree. Figure 2–9 shows the ALM using this feature.

Figure 2–9. ALM in Shared Arithmetic Mode



Shared Arithmetic Chain

The shared arithmetic chain available in enhanced arithmetic mode allows the ALM to implement a three-input add. This significantly reduces the resources necessary to implement large adder trees or correlator functions.

The shared arithmetic chains can begin in either the first or sixth ALM in an LAB. The Quartus II Compiler creates shared arithmetic chains longer than 20 ALMs (10 ALMs in arithmetic or shared arithmetic mode) by linking LABs together automatically. For enhanced fitting, a long shared arithmetic chain runs vertically, allowing fast horizontal connections to memory and DSP blocks. A shared arithmetic chain can continue as far as a full column.

Similar to the carry chains, the top and bottom half of shared arithmetic chains in alternate LAB columns can be bypassed. This capability allows the shared arithmetic chain to cascade through half of the ALMs in an LAB while leaving the other half available for narrower fan-in functionality. Every other LAB column is top-half bypassable, while the other LAB columns are bottom-half bypassable.

For more information on shared arithmetic chain interconnect, refer to "ALM Interconnects" on page 2–13.

LUT-Register Mode

LUT-Register mode allows third register capability in an ALM. Two internal feedback loops allow combinational `ALUT1` to implement the master latch and combinational `ALUT0` to implement the slave latch needed for the third register. The LUT register shares its clock, clock enable, and asynchronous clear sources with the top dedicated register. Figure 2–10 shows the register constructed using two combinational blocks in the ALM.

Figure 2–10. LUT Register from Two Combinational Blocks

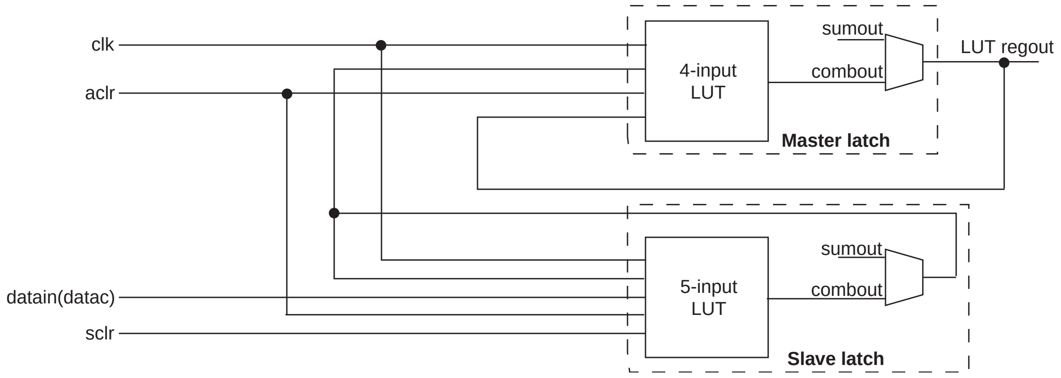
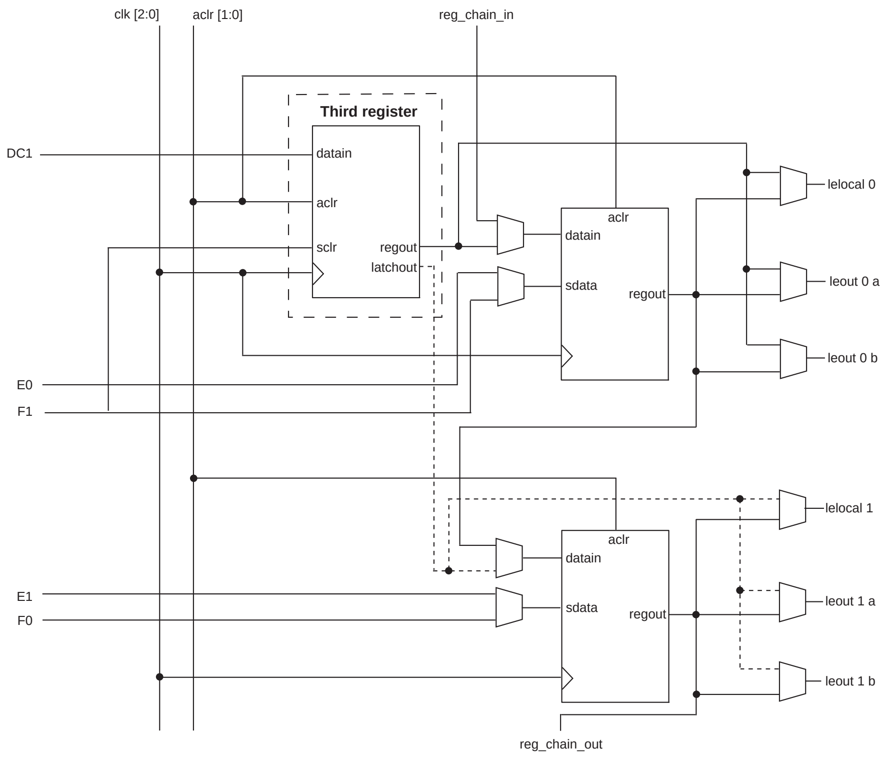


Figure 2–11 shows the ALM in LUT-Register mode.

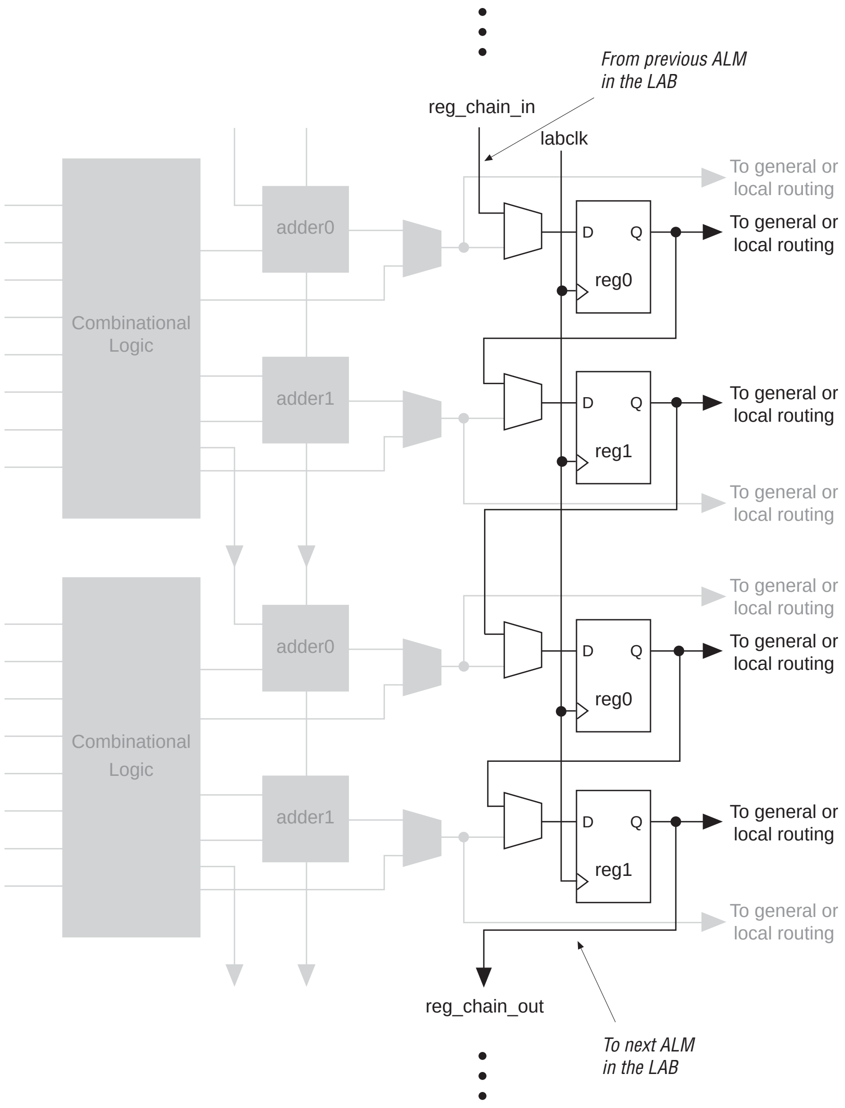
Figure 2–11. ALM in LUT-Register Mode with 3-Register Capability



Register Chain

In addition to general routing outputs, the ALMs in any given LAB have register chain outputs. This allows registers in the same LAB to be cascaded together. The register chain interconnect allows a LAB to use LUTs for a single combinational function and the registers to be used for an unrelated shift register implementation. These resources speed up connections between ALMs while saving local interconnect resources (refer to Figure 2–12). The Quartus II Compiler automatically takes advantage of these resources to improve utilization and performance.

Figure 2–12. Register Chain in an LAB *(Note 1)*



Note to Figure 2–12:

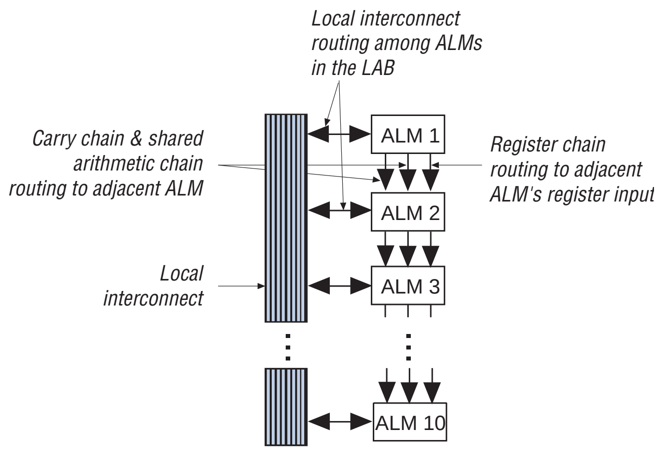
(1) You can use the combinational or adder logic to implement an unrelated, un-registered function.

For more information about register chain interconnect, refer to "ALM Interconnects" on page 2–13.

ALM Interconnects

There are three dedicated paths between ALMs: Register Cascade, Carry-chain, and Shared Arithmetic chain. Arria II GX devices include an enhanced interconnect structure in LABs for routing shared arithmetic chains and carry chains for efficient arithmetic functions. The register chain connection allows the register output of one ALM to connect directly to the register input of the next ALM in the LAB for fast shift registers. These ALM-to-ALM connections bypass the local interconnect. Figure 2–13 shows the shared arithmetic chain, carry chain, and register chain interconnects.

Figure 2–13. Shared Arithmetic Chain, Carry Chain, and Register Chain Interconnects


Local interconnect routing among ALMs in the LAB
Carry chain & shared arithmetic chain routing to adjacent ALM
Register chain routing to adjacent ALM's register input
Local interconnect
ALM 1
ALM 2
ALM 3
ALM 10

Clear and Preset Logic Control

The ALM directly supports an asynchronous clear function. You can achieve the register preset through the Quartus II software's NOT-gate push-back logic option. Each LAB supports up to two clears.

Arria II GX devices provide a device-wide reset pin (`DEV_CLRn`) that resets all registers in the device. An option set before compilation in the Quartus II software enables this pin. This device-wide reset overrides all other control signals.

Document Revision History

Table 2–1 shows the revision history for this document.

Table 2–1. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial Release.	—

3. Memory Blocks in Arria II GX Devices

AIIGX51003-1.0

Introduction

Arria® II GX memory blocks include 640-bit memory logic array blocks (MLABs) and 9-Kbit M9K blocks. You can configure each embedded memory block independently to be a single- or dual-port RAM, FIFO, ROM, or shift register using the Quartus® II MegaWizard™ Plug-In Manager. You can stitch together multiple blocks of the same type to produce larger memories with minimal timing penalty. This chapter describes memory blocks, modes, features, and design considerations.

This chapter contains the following sections:

- "Memory Features" on page 3–1
- "Memory Modes" on page 3–7
- "Clocking Modes" on page 3–14
- "Design Considerations" on page 3–15

Memory Features

Table 3–1 summarizes the features supported by the memory blocks.

Table 3–1. Summary of Memory Features (Part 1 of 2)

Feature	MLABs	M9K Blocks
Maximum performance	360 MHz *(1)*	390 MHz *(1)*
Total RAM bits (including parity bits)	640	9,216
Configurations (depth × width)	64 × 8 64 × 9 64 × 10 32 × 16 32 × 18 32 × 20	8K × 1 4K × 2 2K × 4 1K × 8 1K ×9 512 × 16 512 × 18 256 × 32 256 × 36
Parity bits	✓	✓
Byte enable	✓	✓
Packed mode	—	✓
Address clock enable	✓	✓
Single-port memory	✓	✓
Simple dual-port memory	✓	✓
True dual-port memory	—	✓

Table 3–1. Summary of Memory Features (Part 2 of 2)

Feature	MLABs	M9K Blocks
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 if registered, otherwise reads memory contents.	Outputs cleared
Register clears	Output registers	Output registers
Write/Read operation triggering	Write: Falling clock edges Read: Rising clock edges	Write and Read: Rising clock edges
Same-port read-during-write	Outputs set to old data or don't care	Outputs set to old data, new data, or don't care
Mixed-port read-during-write	Outputs set to old data or don't care	Outputs set to old data or don't care
ECC Support	Soft IP support using Quartus II software	Soft IP support using Quartus II software

Note to Table 3–1:

(1) These numbers are preliminary until characterization is final.

Table 3–2 shows the capacity and distribution of the memory blocks in each Arria II GX family member.

Table 3–2. Memory Capacity and Distribution in Arria II GX Devices

Device	MLABs	M9K Blocks	Total RAM Bits (including MLABs) (Kbits)
EP2AGX20	319	87	982
EP2AGX30	540	144	1,634
EP2AGX45	903	319	3,435
EP2AGX65	1,265	495	5,246
EP2AGX95	1,874	612	6,679
EP2AGX125	2,482	730	8,121
EP2AGX190	3,806	840	9,939
EP2AGX260	5,130	950	11,756

Memory Block Types

M9K memory blocks are dedicated resource; MLABs are dual-purpose blocks. They can be configured as regular LABs or as MLABs. Ten ALMs make up one MLAB. You can configure each ALM in an MLAB as either a 64 × 1 or a 32 × 2 block, resulting in a 64 × 10 or 32 × 20 simple dual-port SRAM block in a single MLAB.

Parity Bit Support

All memory blocks have built-in parity bit support. The ninth bit associated with each byte can store a parity bit or serve as an additional data bit. No parity function is actually performed on the ninth bit.

Byte Enable Support

All memory blocks support byte enables that mask the input data so that only specific bytes of data are written. The unwritten bytes retain the previous written value. The write enable (`wren`) signals, along with the byte enable (`byteena`) signals, control the RAM blocks' write operations.

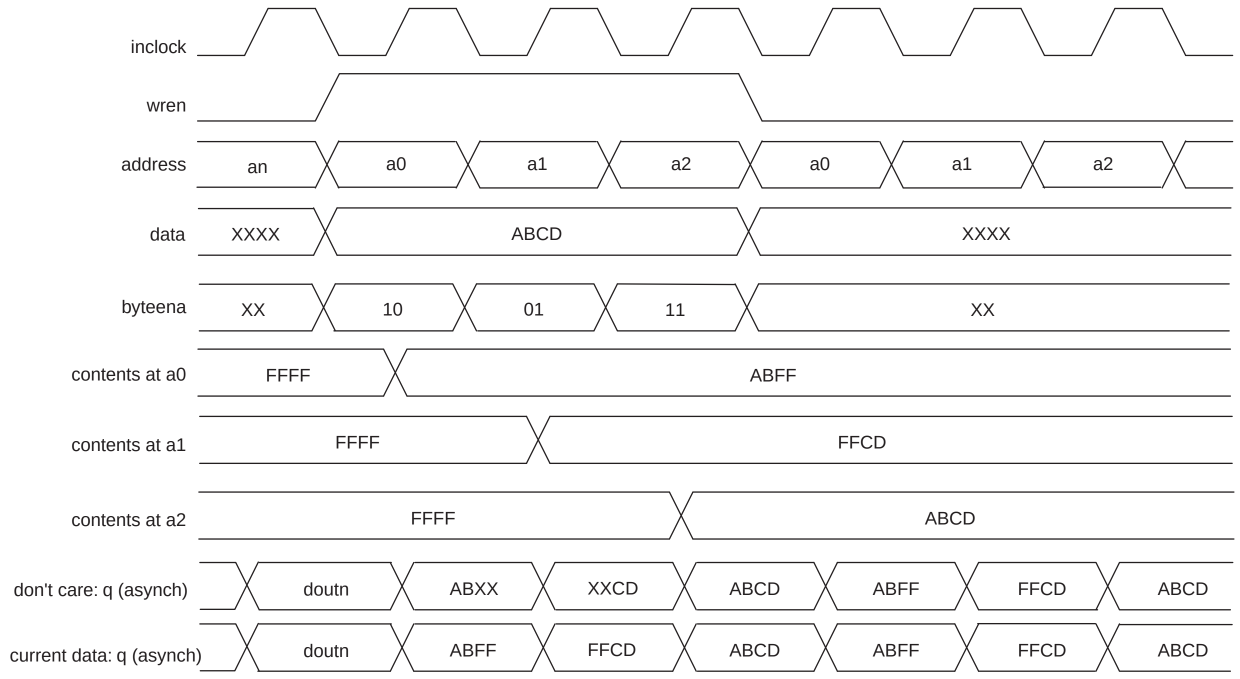
The default value for the byte enable signals is high (enabled), in which case writing is controlled only by the write enable signals. The byte enable registers have no clear port. When using parity bits on the M9K blocks, the byte enable controls all nine bits (eight bits of data plus one parity bit). When using parity bits on the MLAB, the byte-enable controls all 10 bits in the widest mode.

Byte enables operate in a one-hot fashion, with the LSB of the `byteena` signal corresponding to the LSB of the data bus. For example, if using a RAM block in ×18 mode, `byteena = 01`, `data[8..0]` is enabled and `data[17..9]` is disabled. Similarly, if `byteena = 11`, both `data[8..0]` and `data[17..9]` are enabled. Byte enables are active high.

Figure 3–1 shows how the write enable (`wren`) and byte enable (`byteena`) signals control the operations of the RAM.

When a byte-enable bit is de-asserted during a write cycle, the corresponding data byte output can appear as either a "don't care" value or the current data at that location. The output value for the masked byte is controllable using the Quartus II software. When a byte-enable bit is asserted during a write cycle, the corresponding data byte output also depends on the setting chosen in the Quartus II software.

Figure 3–1. Arria II GX Byte Enable Functional Waveform



Packed Mode Support

Arria II GX M9K blocks support packed mode. The packed mode feature packs two independent single-port RAMs into one memory block. The Quartus II software automatically implements packed mode where appropriate by placing the physical RAM block into true dual-port mode and using the MSB of the address to distinguish between the two logical RAMs. The size of each independent single-port RAM must not exceed half of the target block size.

Address Clock Enable Support

All Arria II GX memory blocks support address clock enable, which holds the previous address value for as long as the signal is enabled (`addressstall` = `1`). When you configure the memory blocks in dual-port mode, each port has its own independent address clock enable. The default value for the address clock enable signals is low (disabled).

Figure 3–2 shows an address clock enable block diagram. The address clock enable is referred to by the port name `addressstall`.

Figure 3–2. Arria II GX Address Clock Enable Block Diagram

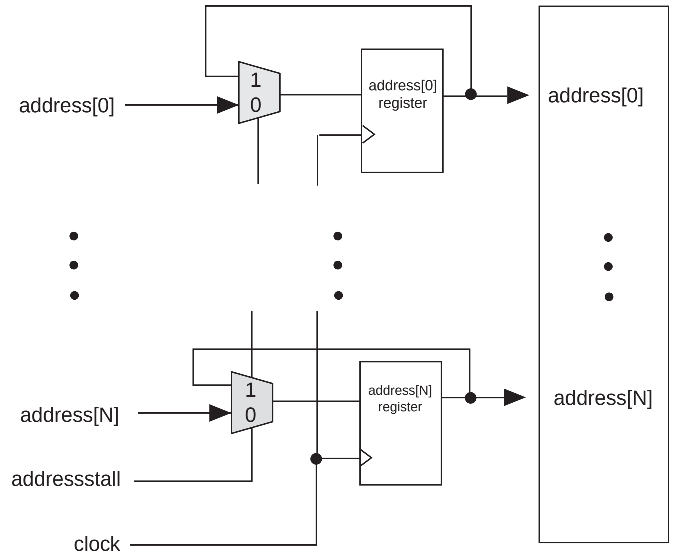


Figure 3–3 shows the address clock enable waveform during the read cycle.

Figure 3–3. Arria II GX Address Clock Enable during Read Cycle Waveform

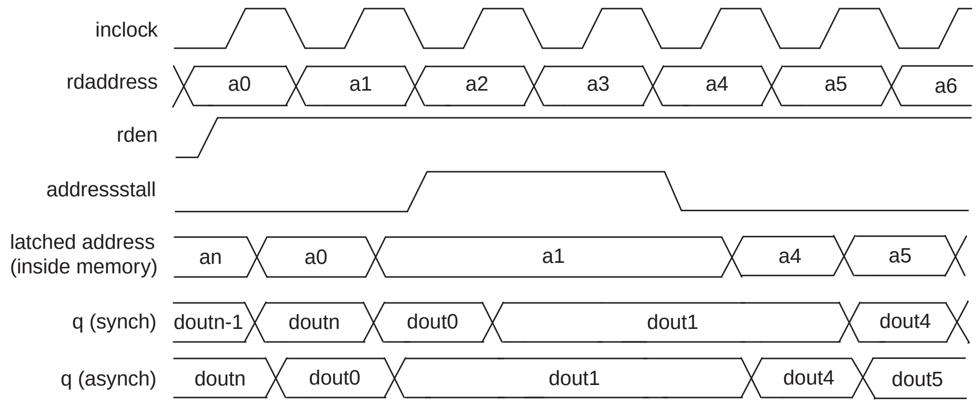
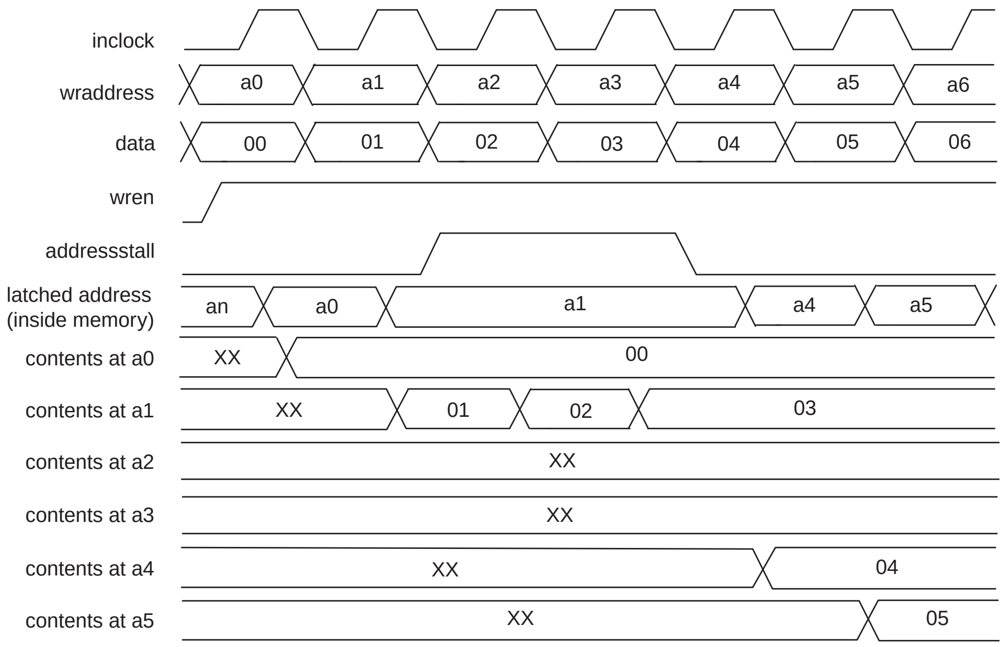


Figure 3–4 shows the address clock enable waveform during write cycle.

Figure 3–4. Arria II GX Address Clock Enable during Write Cycle Waveform



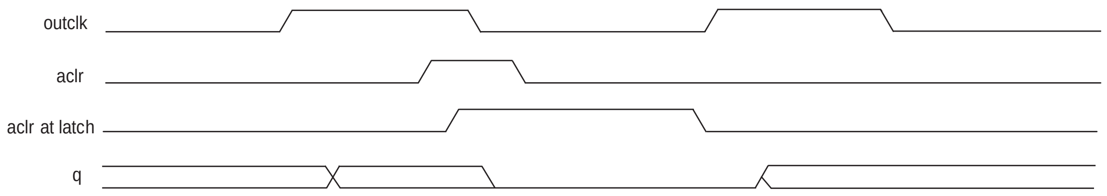
Mixed Width Support

M9K memory blocks support mixed data widths inherently. MLABs can support mixed data widths through emulation using the Quartus II software. When using simple dual-port, true dual-port, or FIFO modes, mixed width support allows you to read and write different data widths to a memory block. For more information about the different widths supported per memory mode, refer to "Memory Modes" on page 3–7.

Asynchronous Clear

Arria II GX memory blocks support asynchronous clears on the output latches and output registers. Therefore, if your RAM is not using output registers, you can still clear the RAM outputs using the output latch asynchronous clear. Figure 3–5 shows a functional waveform showing this functionality.

Figure 3–5. Output Latch Asynchronous Clear Waveform



You can selectively enable asynchronous clears per logical memory using the Quartus II RAM MegaWizard Plug-In Manager.

For more information about the RAM MegaWizard Plug-In Manager, refer to the *RAM Megafunction User Guide*.

Memory Modes

Arria II GX memory blocks allow you to implement fully synchronous SRAM memory in multiple modes of operation. M9K blocks do not support asynchronous memory (unregistered inputs). MLABs support asynchronous (flow-through) read operations.

Depending on which memory block you target, you can use the following modes:

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

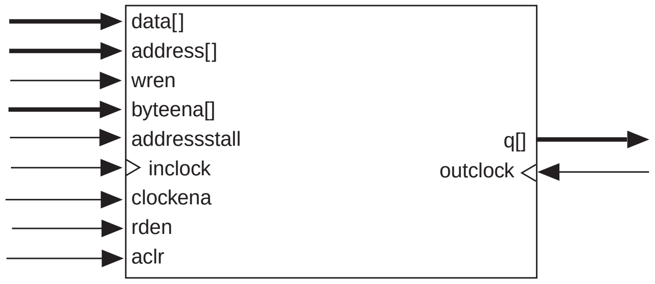
To choose the desired read-during-write behavior, set the read-during-write behavior to either **new data**, **old data**, or **don't care** in the RAM MegaWizard Plug-In Manager in the Quartus II software. For more information about this behavior, refer to "Read-During-Write" on page 3–15.

When using the memory blocks in ROM, single-port, simple dual-port, or true dual-port mode, you can corrupt the memory contents if you violate the setup or hold time on any of the memory block input registers. This applies to both read and write operations.

Single-Port RAM

All memory blocks support single-port mode. Single-port mode allows you to do either one-read or one-write operation at a time. Simultaneous reads and writes are not supported in single-port mode. Figure 3–6 shows the single-port RAM configuration.

Figure 3–6. Single-Port Memory *(Note 1)*



Note to Figure 3–6:

(1) You can implement two single-port memory blocks in a single M9K block. For more information, refer to "Packed Mode Support" on page 3–4.

During a write operation, behavior of the RAM outputs is configurable. If you use the read-enable signal and perform a write operation with the read enable de-activated, the RAM outputs retain the values they held during the most recent active read enable. If you activate read enable during a write operation, or if you are not using the read-enable signal at all, the RAM outputs either show the new data being written, the old data at that address, or a don't care value.

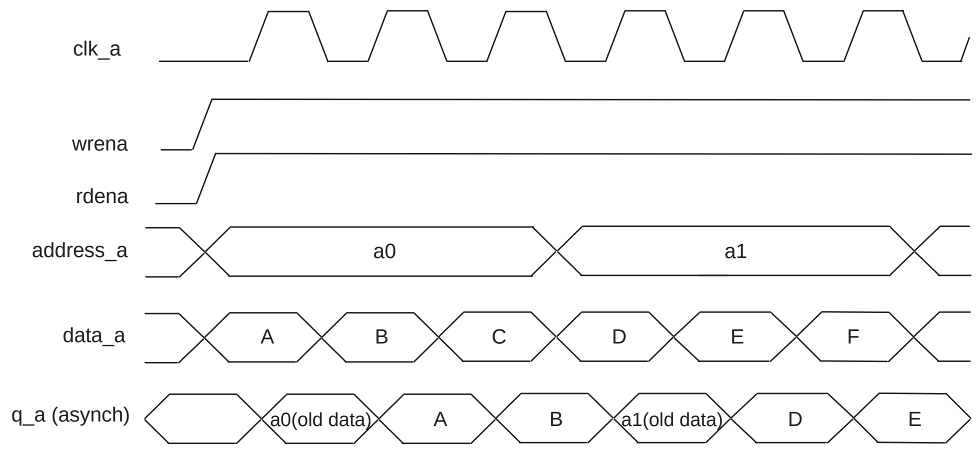
Table 3–3 shows the possible port width configurations for memory blocks in single-port mode.

Table 3–3. Arria II GX Port Width Configurations for MLABs and M9K Blocks

	MLABs	M9K Blocks
Port Width Configurations	64 × 8	8K × 1
	64 × 9	4K × 2
	64 × 10	2K × 4
	32 × 16	1K × 8
	32 × 18	1K × 9
	32 × 20	512 × 16
		512 × 18
		256 × 32
		256 × 36

Figure 3–7 shows timing waveforms for read and write operations in single-port mode with unregistered outputs. Registering the RAM's outputs would simply delay the q output by one clock cycle.

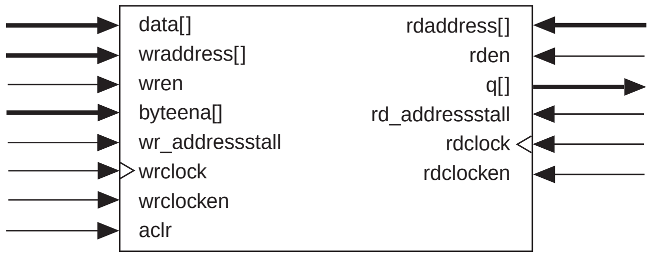
Figure 3–7. Timing Waveform for Read-Write Operations (Single-Port Mode)



Simple Dual-Port Mode

All memory blocks support simple dual-port mode. Simple dual-port mode allows you to perform one-read and one-write operation to different locations at the same time. Figure 3–8 shows a simple dual-port configuration.

Figure 3–8. Arria II GX Simple Dual-Port Memory *(Note 1)*



Note to Figure 3–8:

(1) Simple dual-port RAM supports input and output clock mode in addition to the read and write clock mode shown.

Simple dual-port mode supports different read and write data widths (mixed width support). Table 3–4 shows the mixed width configurations for the M9K blocks in simple dual-port mode. MLABs do not have native support for mixed width operation. The Quartus II software can implement mixed width memories in MLABs with more than one MLAB.

Table 3–4. Arria II GX M9K Block Mixed-Width Configurations (Part 1 of 2)

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

Table 3–4. Arria II GX M9K Block Mixed-Width Configurations (Part 2 of 2)

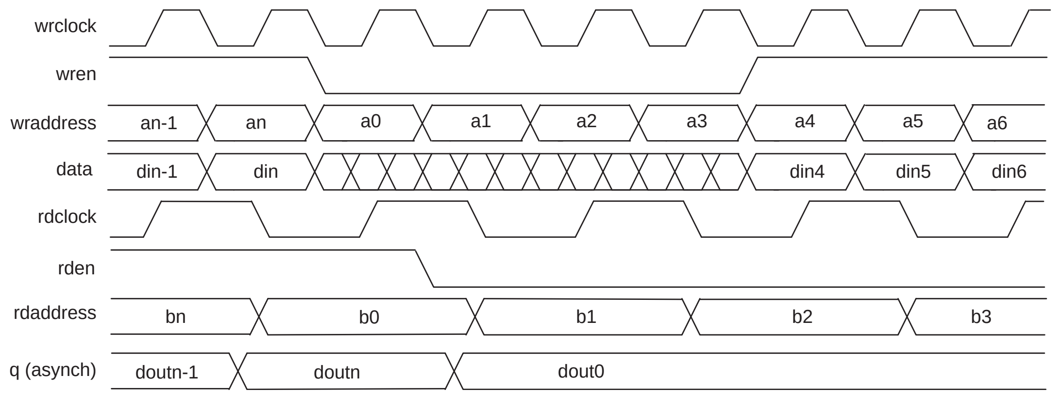
Read Port	Write Port								
	8K×1	**4K×2**	**2K×4**	**1K×8**	**512×16**	**256×32**	**1K×9**	**512×18**	**256×36**
2K×4	✓	✓	✓	✓	✓	✓	—	—	—
1K×8	✓	✓	✓	✓	✓	✓	—	—	—
512×16	✓	✓	✓	✓	✓	✓	—	—	—
256×32	✓	✓	✓	✓	✓	✓	—	—	—
1K×9	—	—	—	—	—	—	✓	✓	✓
512×18	—	—	—	—	—	—	✓	✓	✓
256×36	—	—	—	—	—	—	✓	✓	✓

In simple dual-port mode, M9K blocks support separate write-enable and read-enable signals. Read-during-write operations to the same address can either output a don't care value or old data.

MLABs only support a write-enable signal. Read-during-write behavior for the MLABs can be either don't care, new data, or old data. The available choices depend on the configuration of the MLAB.

Figure 3–9 shows timing waveforms for read and write operations in simple dual-port mode with unregistered outputs. Registering the RAM's outputs would simply delay the q output by one clock cycle.

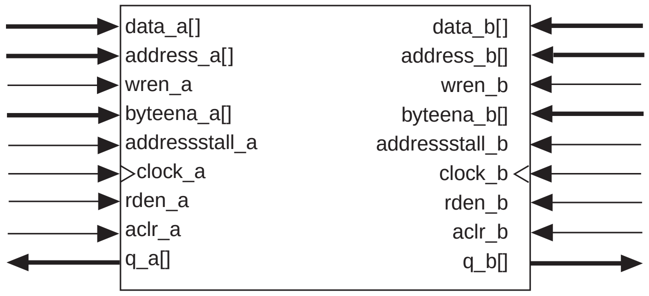
Figure 3–9. Arria II GX Simple Dual-Port Timing Waveforms



True Dual-Port Mode

Arria II GX M9K blocks support true dual-port mode. Sometimes called bi-directional dual-port, this mode allows you to perform any combination of two port operations: two reads, two writes, or one read and one write at two different clock frequencies. Figure 3–10 shows the true dual-port RAM configuration.

Figure 3–10. Arria II GX True Dual-Port Memory *(Note 1)*



Note to Figure 3–10:

(1) True dual-port memory supports input and output clock mode in addition to the independent clock mode shown.

The widest bit configuration of the M9K blocks in true dual-port mode is 512 × 16-bit (×18-bit with parity).

Wider configurations 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, its maximum width equals half of the total number of output drivers. Table 3–5 lists the possible M9K block mixed-port width configurations in true dual-port mode.

Table 3–5. Arria II GX M9K Block Mixed-Width Configuration

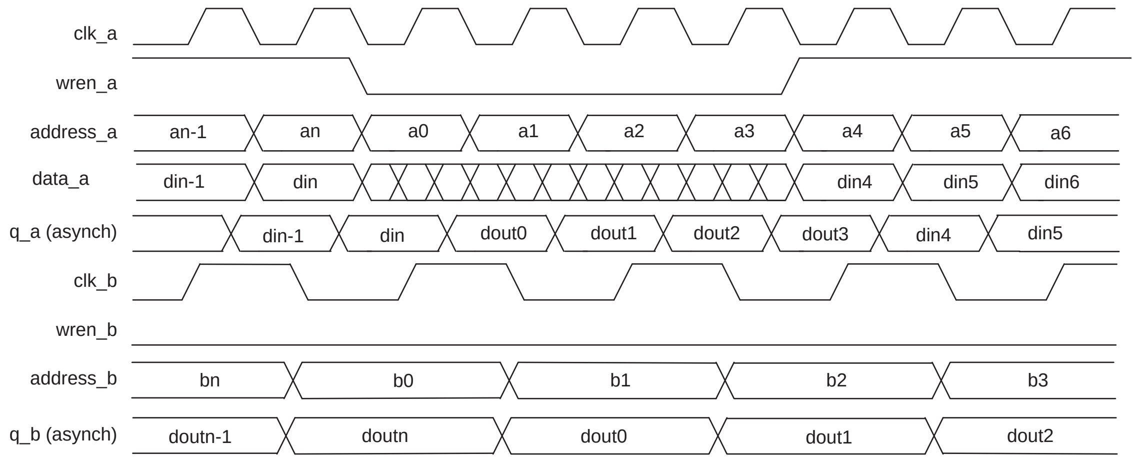
Read Port	Write Port						
	8K×1	4K×2	2K×4	1K×8	512×16	1K×9	512×18
8K×1	✓	✓	✓	✓	✓	—	—
4K×2	✓	✓	✓	✓	✓	—	—
2K×4	✓	✓	✓	✓	✓	—	—
1K×8	✓	✓	✓	✓	✓	—	—
512×16	✓	✓	✓	✓	✓	—	—
1K×9	—	—	—	—	—	✓	✓
512×18	—	—	—	—	—	✓	✓

In true dual-port mode, M9K blocks support separate write-enable and read-enable signals. Read-during-write operations to the same address can either output new data at that location or old data.

In true dual-port mode, you can access any memory location at any time from either port. When accessing the same memory location from both ports, you must avoid possible write conflicts. A write conflict happens when you attempt to write to the same address location from both ports at the same time. This results in unknown data being stored to that address location. No conflict resolution circuitry is built into the Arria II GX memory blocks. You must handle address conflicts external to the RAM block.

Figure 3–11 shows true dual-port timing waveforms for the write operation at port A and read operation at port B with the **Read-During-Write** behavior set to **new data**. Registering the RAM's outputs would simply delay the q outputs by one clock cycle.

Figure 3–11. Arria II GX True Dual-Port Timing Waveform



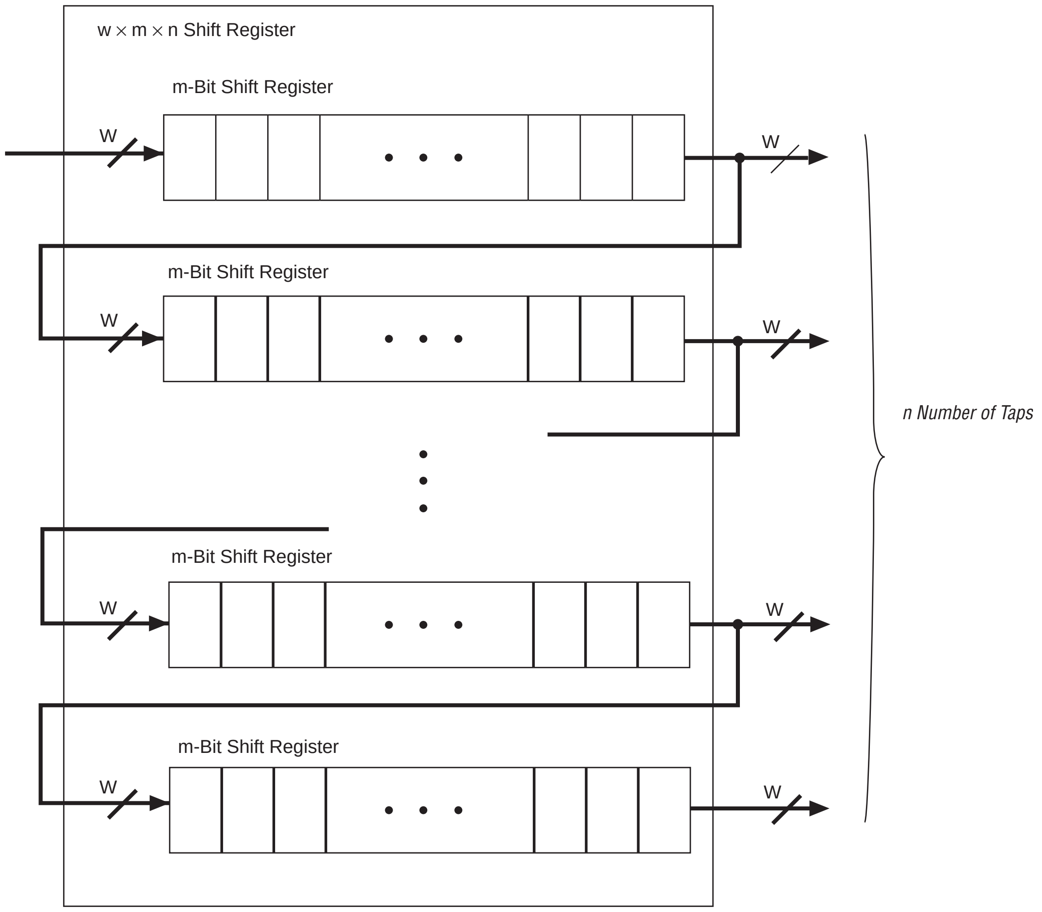
Shift-Register Mode

All Arria II GX memory blocks support 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- and cross-correlation functions. These and other DSP applications require local data storage, traditionally implemented with standard flipflops that quickly exhaust 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.

The size of a shift register (w × m × n) is determined by the input data width (w), the length of the taps (m), and the number of taps (n). You can cascade memory blocks to implement larger shift registers.

Figure 3–12 shows the memory block in shift-register mode.

Figure 3–12. Arria II GX Shift-Register Memory Configuration



ROM Mode

All Arria II GX memory blocks support ROM mode. A memory initialization file (**.mif**) initializes the ROM contents of these blocks. The address lines of the ROM are registered on M9K blocks, but can be unregistered on MLABs. The outputs can be registered or unregistered. Output registers can be asynchronously cleared. The ROM read operation is identical to the read operation in the single-port RAM configuration.

FIFO Mode

All memory blocks support FIFO mode. MLABs are ideal for designs with many small, shallow FIFO buffers. To implement FIFO buffers in your design, you can use the FIFO MegaWizard Plug-In Manager in the Quartus II software. Both single- and dual-clock (asynchronous) FIFOs are supported.

For more information about implementing FIFO buffers, refer to the *Single- and Dual-Clock FIFO Megafunctions User Guide*.

Clocking Modes

Arria II GX memory blocks support the following clocking modes:

- Independent
- Input and output
- Read and write
- Single clock

Violating the setup or hold time on the memory block address registers could corrupt the memory contents. This applies to both read and write operations.

Table 3–6 shows the clocking mode versus memory mode support matrix.

Table 3–6. Arria II GX Memory Clock Modes

Clocking Mode	True Dual-Port Mode	Simple Dual-Port Mode	Single-Port Mode	ROM Mode	FIFO Mode
Independent	✓	—	—	✓	—
Input and output	✓	✓	✓	✓	—
Read and write	—	✓	—	—	✓
Single clock	✓	✓	✓	✓	✓

Independent Clock Mode

Arria II GX memory blocks can implement independent clock mode for true dual-port memories. 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 for port A and port B registers. Asynchronous clears are supported only for output latches and output registers on both ports.

Input and Output Clock Mode

Arria II GX memory blocks can implement input and output clock mode for true and simple dual-port memories. In this mode, an input clock controls all registers related to the data input to the memory block including data, address, byte enables, read enables, and write enables. An output clock controls the data output registers. Asynchronous clears are available on output latches and output registers only.

Read and Write Clock Mode

Arria II GX memory blocks can implement read and write clock mode for simple dual-port memories. In this mode, a write clock controls the data-input, write-address, and write-enable registers. Similarly, a read clock controls the data-output, read-address, and read-enable registers. The memory blocks support independent clock enables for both the read and write clocks. Asynchronous clears are available on data output latches and registers only.

Single Clock Mode

Arria II GX memory blocks can implement single-clock mode for true dual-port, simple dual-port, and single-port memories. In this mode, a single clock, together with a clock enable, is used to control all registers of the memory block. Asynchronous clears are available on output latches and output registers only.

Design Considerations

This section describes guidelines for designing with memory blocks.

Memory Block Selection

The Quartus II software automatically partitions user-defined memory into embedded memory blocks by taking into account both speed and size constraints placed on your design. For example, the Quartus II software may spread out memory across multiple memory blocks when resources are available to increase the performance of your design. You can manually assign memory to a specific block size using the RAM MegaWizard Plug-In Manager.

MLABs can implement single-port SRAM through emulation using the Quartus II software. Emulation results in minimal additional logic resources being used. Because of the dual-purpose architecture of the MLAB, it only has data input registers and output registers in the block. MLABs gain input address registers and additional optional data output registers from adjacent ALMs with register packing.

For more information about register packing, refer to the *Logic Array Blocks and Adaptive Logic Modules in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

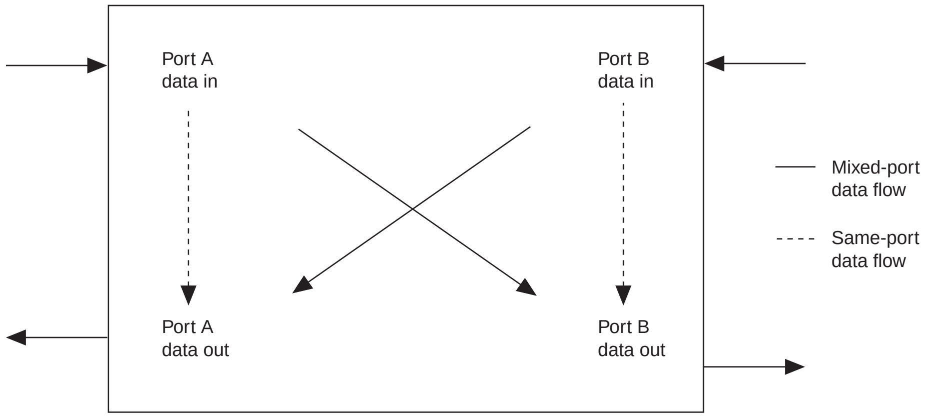
Conflict Resolution

When using the memory blocks in true dual-port mode, it is possible to attempt two write operations to the same memory location (address). Because there is no conflict resolution circuitry built into the memory blocks, this results in unknown data being written to that location. Therefore, you must implement conflict resolution logic external to the memory block to avoid address conflicts.

Read-During-Write

You can customize the read-during-write behavior of the Arria II GX memory blocks to suit your design needs. Two types of read-during-write operations are available: same port and mixed port. Figure 3–13 shows the difference between the two types.

Figure 3–13. Arria II GX Read-During-Write Data Flow



Same-Port Read-During-Write Mode

This mode applies to either a single-port RAM or the same port of a true dual-port RAM. In same-port read-during-write mode, three output choices are available: new data mode (or flow-through), old data mode, or don't care mode. In new data mode, the new data is available on the rising edge of the same clock cycle on which it was written. In old data mode, the RAM outputs reflect the old data at that address before the write operation proceeds. In don't care mode, the RAM outputs don't care values for a read-during-write operation.

Figure 3–14 shows sample functional waveforms of same-port read-during-write behavior in new data mode.

Figure 3–14. Same Port Read-During Write: New Data Mode

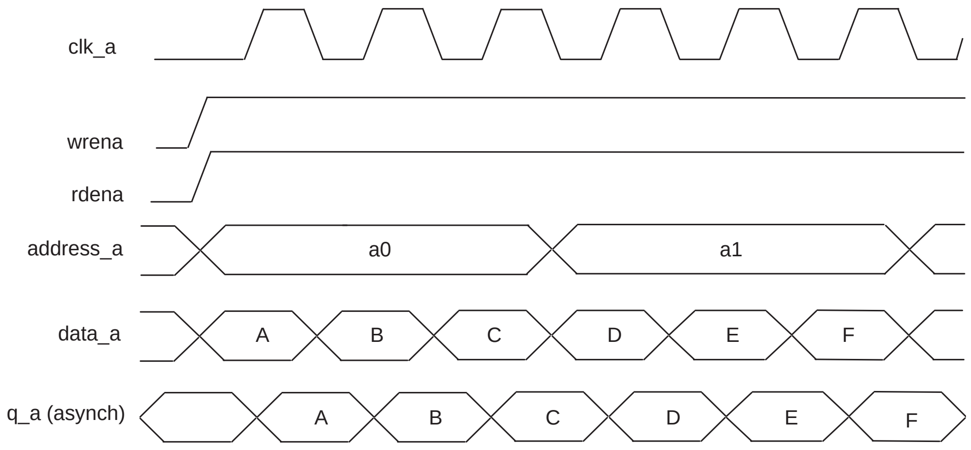
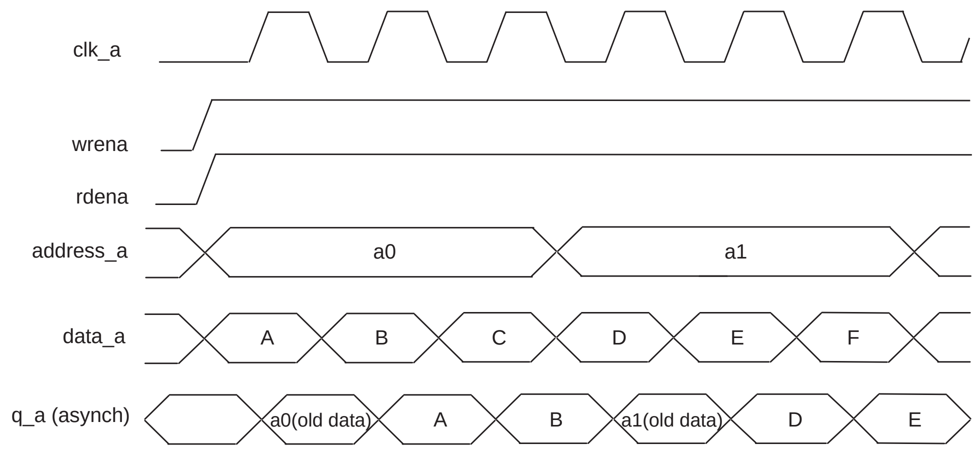


Figure 3–15 shows sample functional waveforms of same-port read-during-write behavior in old data mode.

Figure 3–15. Same Port Read-During-Write: Old Data Mode



Mixed-Port Read-During-Write Mode

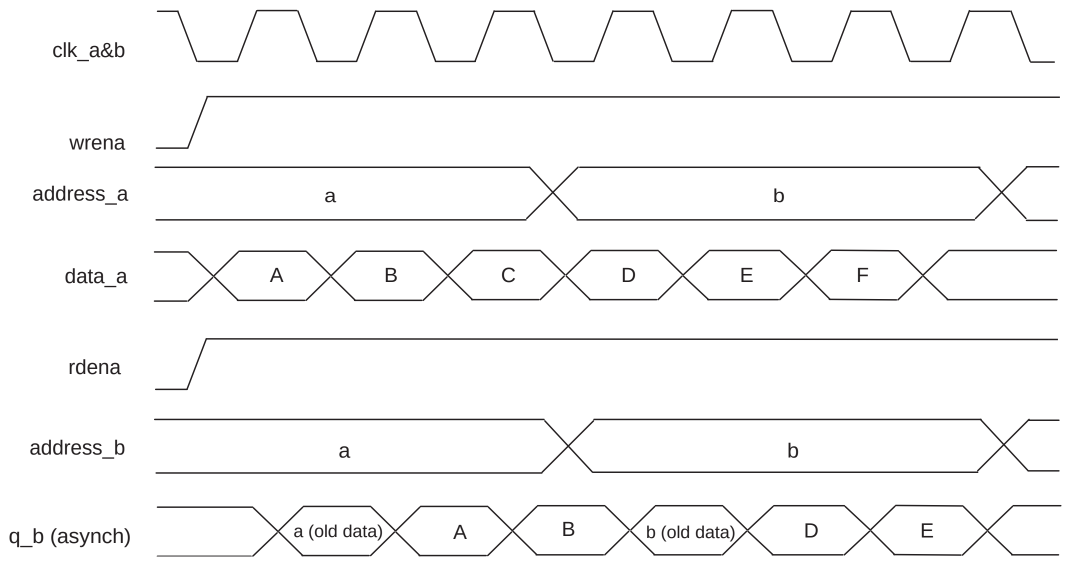
This mode applies to a RAM in simple or true dual-port mode which has one port reading and the other port writing to the same address location with the same clock.

In this mode, you also have two output choices: old data or don't care. In old data mode, a read-during-write operation to different ports causes the RAM outputs to reflect the old data at that address location. In don't care mode, the same operation results in a "don't care" or "unknown" value on the RAM outputs.

Read-during-write behavior is controlled using the RAM MegaWizard Plug-In Manager. For more information about how to implement the desired behavior, refer to the *RAM Megafunction User Guide*.

Figure 3–16 shows a sample functional waveform of mixed-port read-during-write behavior in old data mode. In don't care mode, the old data shown in the figure is simply replaced with "don't care".

Figure 3–16. Mixed Port Read During Write: Old Data Mode



Mixed-port read-during-write is not supported when two different clocks are used in a dual-port RAM. The output value is unknown during a dual-clock mixed-port read-during-write operation.

Power-Up Conditions and Memory Initialization

M9K memory block outputs power up to zero (cleared), regardless of whether the output registers are used or bypassed. MLABs power up to zero if output registers are used and power up reading the memory contents if output registers are not used. The Quartus II software initializes the RAM cells to zero unless there is an MIF file (**.mif**) specified.

All memory blocks support initialization using a **.mif** file. You can create **.mif** files in the Quartus II software and specify their use with the RAM MegaWizard Plug-In Manager when instantiating a memory in your design. Even if a memory is pre-initialized (for example, using a **.mif** file), it still powers up with its outputs cleared.

For more information about **.mif** files, refer to the *RAM Megafunction User Guide* and the *Quartus II Handbook*.

Power Management

Arria II GX memory block clock-enables allow you to control clocking of each memory block to reduce AC-power consumption. Use the read-enable signal to ensure that read operations only occur when you need them to. If your design does not require read-during-write, you can reduce your power consumption by de-asserting the read-enable signal during write operations or any period when no memory operations occur.

The Quartus II software automatically places any unused memory blocks in low power mode to reduce static power.

Document Revision History

Table 3–7 shows the revision history for this chapter.

Table 3–7. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial Release.	—

4. DSP Blocks in Arria II GX Devices

AIIGX51004-1.0

Introduction

The Arria® II GX family of devices have dedicated high-performance digital signal processing (DSP) blocks optimized for DSP applications. These DSP blocks are the fourth generation of hardwired, fixed-function silicon blocks dedicated to maximizing signal processing capability and ease-of-use at the lowest silicon cost.

Many complex systems, such as WiMAX, 3GPP WCDMA, high-performance computing (HPC), voice over Internet protocol (VoIP), H.264 video compression, medical imaging, and HDTV, use sophisticated DSP techniques. Arria II GX devices are ideally suited as the DSP blocks consist of a combination of dedicated elements that perform multiplication, addition, subtraction, accumulation, summation, and dynamic shift operations.

This chapter contains the following sections:

- "DSP Block Overview" on page 4–1
- "Simplified DSP Operation" on page 4–3
- "Operational Modes Overview" on page 4–5
- "DSP Block Resource Descriptions" on page 4–6
- "Operational Mode Descriptions" on page 4–13
- "Software Support for Arria II GX Devices" on page 4–31

DSP Block Overview

Each Arria II GX device has two to four columns of DSP blocks that implement multiplication, multiply-add, multiply-accumulate (MAC), and dynamic shift functions. Architectural highlights of the Arria II GX DSP block include:

- High-performance, power-optimized, fully registered and pipelined multiplication operations
- High-performance DSP blocks up to 350 MHz
- Natively supported 9-bit, 12-bit, 18-bit, 36-bit word lengths
- Natively supported 18-bit complex multiplications
- Efficiently supported floating-point arithmetic formats (24-bits for single precision and 53-bits for double precision)
- Signed and unsigned input support
- Built-in addition, subtraction, and accumulation units to combine multiplication results efficiently
- Cascading 18-bit input bus to form tap-delay line for filtering applications
- Cascading 44-bit output bus to propagate output results from one block to the next block without external logic support

- Rich and flexible arithmetic rounding and saturation units
- Efficient barrel shifter support
- Loopback capability to support adaptive filtering

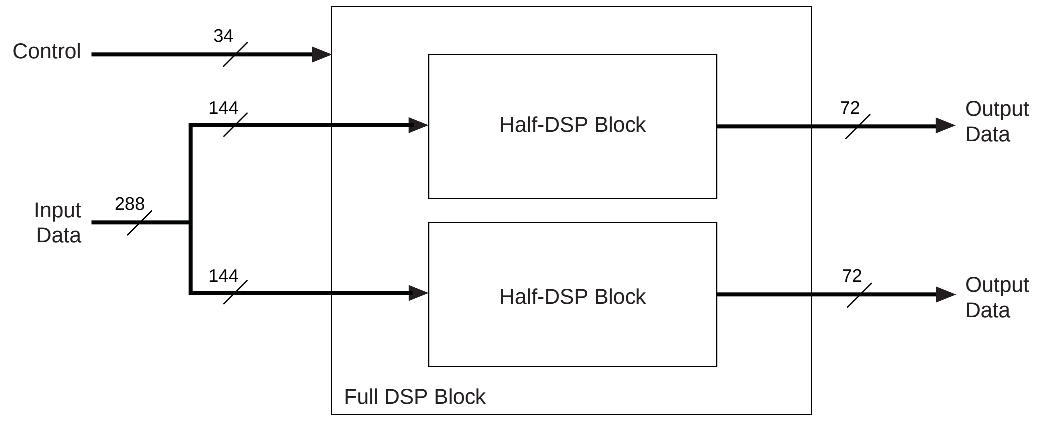
Table 4–1 shows the number of DSP blocks in Arria II GX devices.

Table 4–1. Number of DSP Blocks in Arria II GX Devices

Device	DSP Blocks	Independent Input and Output Multiplication Operators					High Precision Multiplier Adder Mode	Four Multiplier Adder Mode
		9 x 9 Multipliers	12 x 12 Multipliers	18 x 18 Multipliers	18 x 18 Complex	36 x 36 Multipliers	18 x 36	18 x 18
2AGX20	7	56	42	28	14	14	28	56
2AGX30	16	128	96	64	32	32	64	128
2AGX45	29	232	174	116	58	58	116	232
2AGX65	39	312	234	156	78	78	156	312
2AGX95	56	448	336	224	112	112	224	448
2AGX125	72	576	432	288	144	144	288	576
2AGX190	82	656	492	328	164	164	328	656
2AGX260	92	736	552	368	184	184	368	736

Each DSP block occupies four logic array blocks (LABs) in height and can be divided further into two half-blocks that share some common clocks. Figure 4–1 shows the layout of each block.

Figure 4–1. Overview of DSP Block Signals



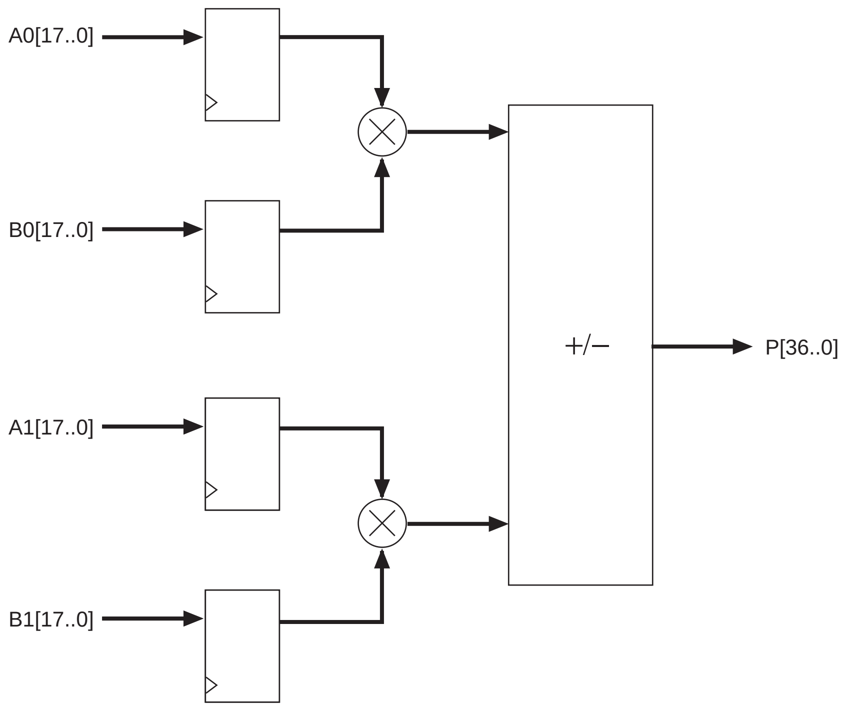
Simplified DSP Operation

The fundamental building block of Arria II GX devices is a pair of 18 × 18-bit multipliers followed by a first-stage 37-bit addition/subtraction unit, as shown in Equation 4–1 and Figure 4–2. For all signed numbers, input and output data is represented in 2's complement format only.

Equation 4–1. Multiplier Equation

$$P[36..0] = A_0[17..0] \times B_0[17..0] \pm A_1[17..0] \times B_1[17..0]$$

Figure 4–2. Basic Two-Multiplier Adder Building Block



The structure shown in Figure 4–2 is useful for building more complex structures, such as complex multipliers and 36 × 36 multipliers, as described in later sections.

Each Arria II GX DSP block contains four two-multiplier adder units (2 two-multiplier adder units per half-block). Therefore, there are eight 18 × 18 multiplier functionalities per DSP block. For a detailed diagram of the DSP block, refer to Figure 4–4 on page 4–7.

Following the two-multiplier adder units are the pipeline registers, the second-stage adders, and an output register stage. You can configure the second-stage adders to provide the alternative functions shown in Equation 4–1 and Equation 4–2 per half-block.

Equation 4–2. Four-Multiplier Adder Equation

$$Z[37..0] = P_0[36..0] + P_1[36..0]$$

Equation 4–3. Four-Multiplier Adder Equation (44-Bit Accumulation)

$$W_n[43..0] = W_{n-1}[43..0] \pm Z_n[37..0]$$

In these equations, *n* denotes sample time and P[36..0] are the results from the two-multiplier adder units.

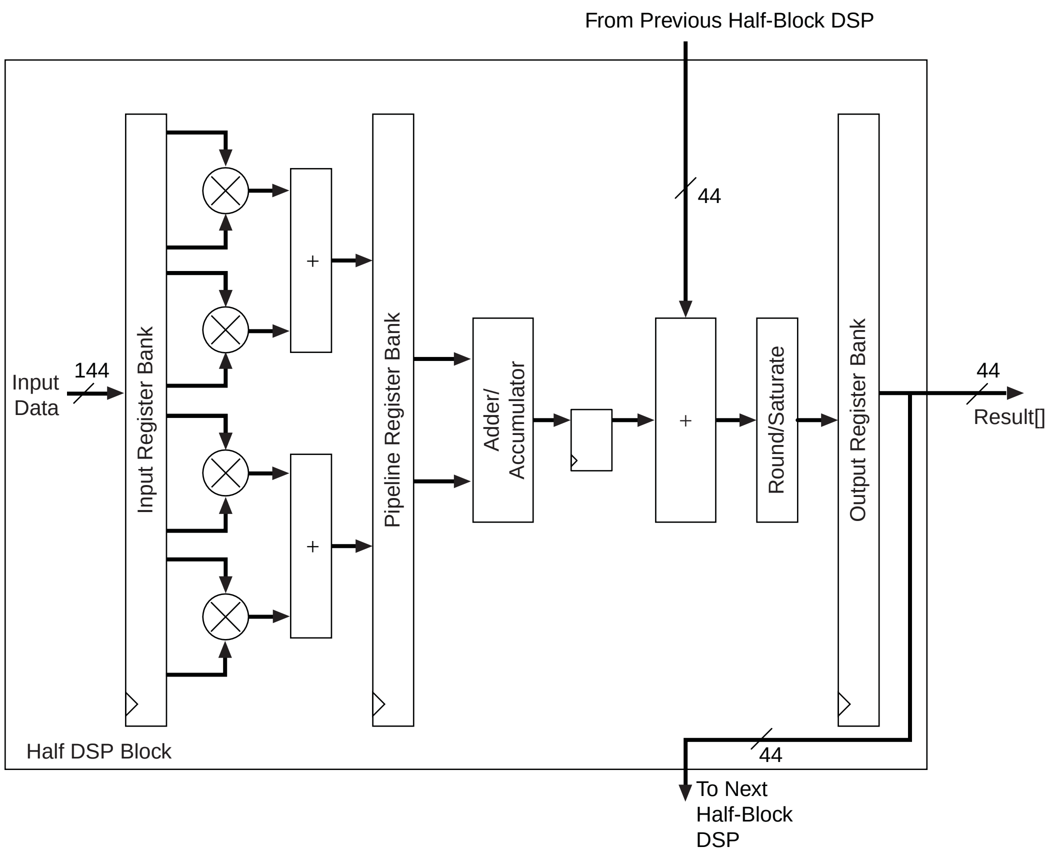
Equation 4–2 provides a sum of four 18 × 18-bit multiplication operations (four-multiplier adder), and Equation 4–3 provides a four 18 × 18-bit multiplication operation, but with a maximum of a 44-bit accumulation capability by feeding the output from the output register bank back to the adder/accumulator block. For the output register bank and adder/accumulator block, refer to Figure 4–3.

You can bypass all register stages depending on which mode you select.

To support finite impulse response (FIR)-like structures efficiently, a major addition to the DSP block in Arria II GX devices is the ability to propagate the result of one half-block to the next half-block completely in the DSP block without additional soft logic overhead. This is achieved by the inclusion of a dedicated addition unit and routing that adds the 44-bit result of a previous half-block with the 44-bit result of the current block. The 44-bit result is either fed to the next half-block or out of the DSP block using the output register stage. This is shown in Figure 4–3. Detailed examples are described in later sections.

To support single-channel type FIR filters efficiently, you can configure one of the multiplier input's registers to form a tap delay line input, saving resources and providing higher system performance.

Figure 4–3. Output Cascading Feature for FIR Structures



Operational Modes Overview

You can use each Arria II GX DSP block in one of five basic operational modes. Table 4–2 shows the five basic operational modes and the number of multipliers that you can implement in a single DSP block, depending on the mode.

Table 4–2. Arria II GX DSP Block Operation Modes (Part 1 of 2)

Mode	Multiplier in Width	Number of Multiplier	# per Block	Signed or Unsigned	RND, SAT	In Shift Register	Chainout Adder	1st Stage Add/Sub	2nd Stage Add/Acc
Independent Multiplier	9-bits	1	8	Both	No	No	No	—	—
	12-bits	1	6	Both	No	No	No	—	—
	18-bits	1	4	Both	Yes	Yes	No	—	—
	36-bits	1	2	Both	No	No	No	—	—
	Double	1	2	Both	No	No	No	—	—
Two-Multiplier Adder *(1)*	18-bits	2	4	Signed *(4)*	Yes	No	No	Both	—
Four-Multiplier Adder	18-bits	4	2	Both	Yes	Yes	Yes	Both	Add Only
Multiply Accumulate	18-bits	4	2	Both	Yes	Yes	Yes	Both	Both

Table 4–2. Arria II GX DSP Block Operation Modes (Part 2 of 2)

Mode	Multiplier in Width	Number of Multiplier	# per Block	Signed or Unsigned	RND, SAT	In Shift Register	Chainout Adder	1st Stage Add/Sub	2nd Stage Add/Acc
Shift *(2)*	36-bits *(3)*	1	2	Both	No	No	—	—	—
High Precision Multiplier Adder	18x36	2	2	Both	No	No	No	—	Add Only

Notes to Table 4–2:

(1) This mode also supports loopback mode. In loopback mode, the number of loopback multipliers per DSP block is two. You can use the remaining multipliers in regular two-multiplier adder mode.

(2) Dynamic shift mode supports arithmetic shift left, arithmetic shift right, logical shift left, logical shift right, and rotation operation.

(3) Dynamic shift mode operates on a 32-bit input vector but the multiplier width is configured as 36 bits.

(4) Unsigned value is also supported but you must ensure that the result can be contained in 36 bits.

The DSP block consists of two identical halves: top-half and bottom-half. Each half has four 18 × 18 multipliers.

Arria II GX DSP blocks can operate in different modes simultaneously. Each half-block is fully independent except for the sharing of the four `clock`, `ena`, and the `aclr` signals. For example, you can break down a single DSP block to operate a 9 × 9 multiplier in one half-block and an 18 × 18 two-multiplier adder in the other half-block. This increases DSP block resource efficiency and allows you to implement more multipliers in an Arria II GX device. The Quartus II software automatically places multipliers that can share the same DSP block resources in the same block.

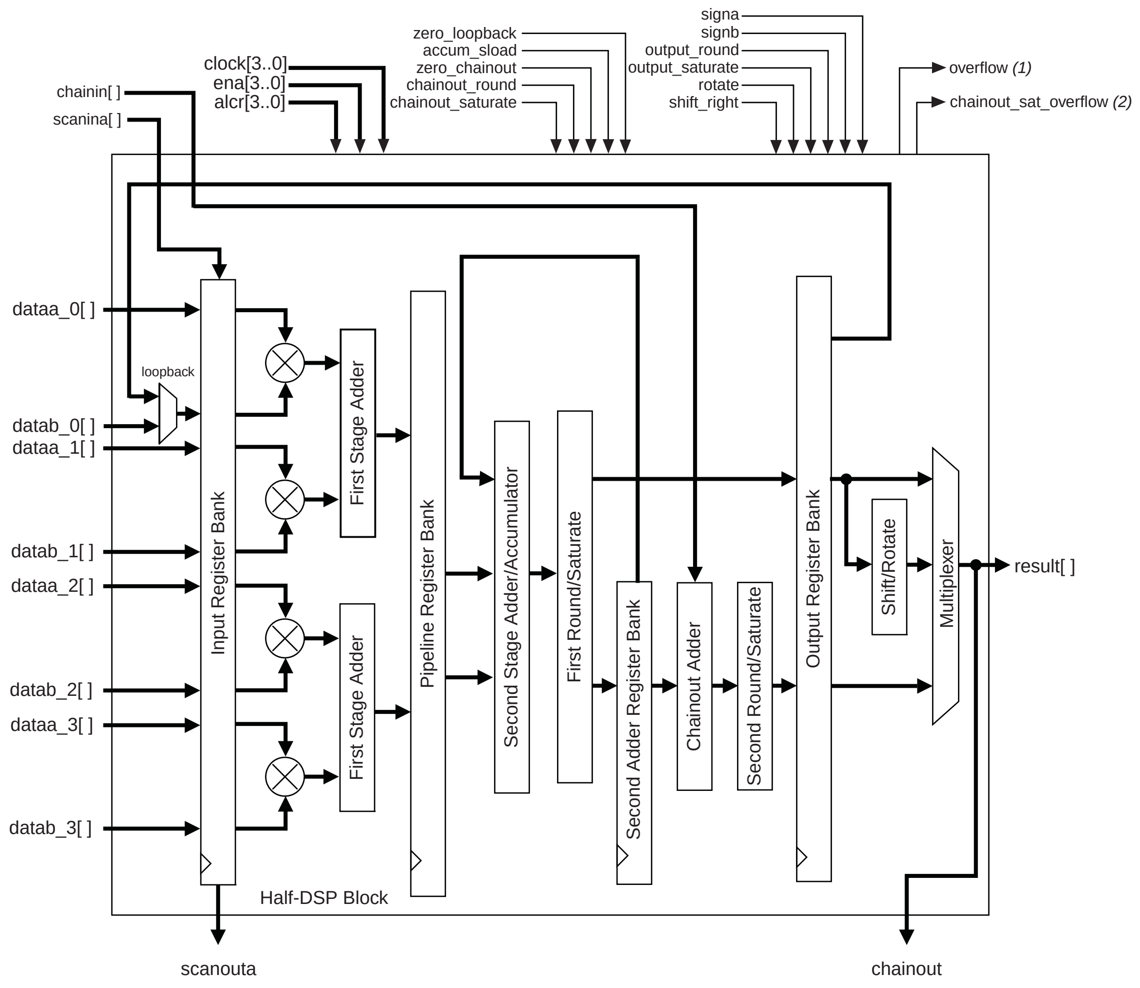
DSP Block Resource Descriptions

The DSP block consists of the following elements:

- Input register bank
- Four two-multiplier adders
- Pipeline register bank
- Two second-stage adders
- Four round and saturation logic units
- Second adder register and output register bank

Figure 4–4 shows a detailed overall architecture of the top half of the DSP block. Table 4–9 on page 4–30 shows a list of DSP block dynamic signals.

Figure 4–4. Half-DSP Block Architecture



Notes to Figure 4–4:

(1) Block output for accumulator overflow and saturate overflow.

(2) Block output for saturation overflow of `chainout`.

(3) When the `chainout` adder is not in use, the second adder register banks are known as output register banks.

Input Registers

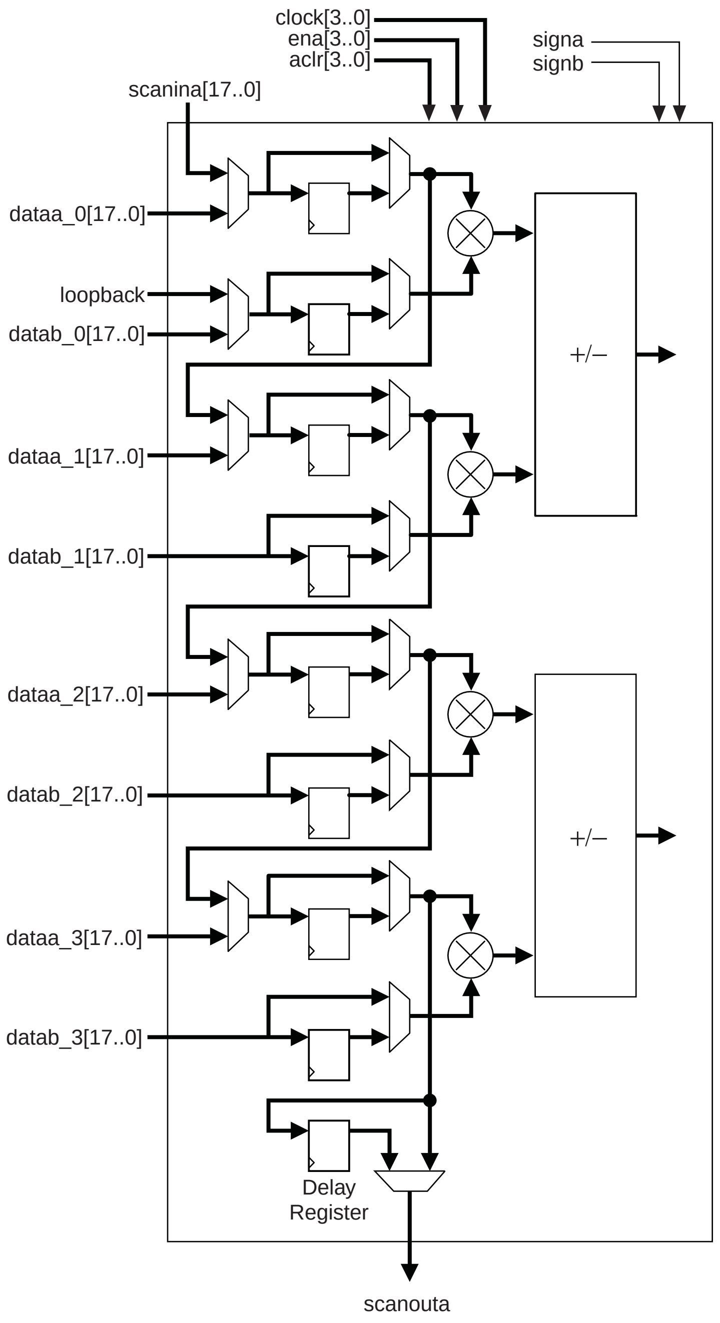
All DSP block registers are triggered by the positive edge of the clock signal and are cleared upon power up. Each multiplier operand can feed an input register or feed directly to the multiplier, bypassing the input registers. The following DSP block signals control the input registers in the DSP block:

- `clock[3..0]`
- `ena[3..0]`
- `aclr[3..0]`

Every DSP block has nine 18-bit data input register banks per half DSP block. Every half DSP block has the option to use the eight data register banks as inputs to the four multipliers. The special ninth register bank is a delay register required by modes that use both the cascade and chainout features of the DSP block for balancing the latency requirements.

A feature of the input register bank is to support a tap delay line. Therefore, the top leg of the multiplier input (A) can be driven from general routing or from the cascade chain, as shown in Figure 4–5. Table 4–9 on page 4–30 shows a list of DSP block dynamic signals.

Figure 4–5. Input Register of Half-DSP Block *(Note 1)*



Note to Figure 4–5:

(1) The `scanina` signal originates from the previous DSP block, while the `scanouta` signal goes to the next DSP block.

You must select the incoming data for multiplier input (A) from either general routing or from the cascade chain at compile time. In cascade mode, the dedicated shift outputs from one multiplier block directly feeds input registers of the adjacent multiplier below it (in the same half DSP block) or the first multiplier in the next half DSP block, to form an 8-tap shift register chain per DSP Block. The DSP block can increase the length of the shift register chain by cascading to the lower DSP blocks. The dedicated shift register chain spans a single column, but you can implement longer shift register chains requiring multiple columns using the regular FPGA routing resources.

Shift registers are useful in DSP functions such as FIR filters. When implementing 18 × 18 or smaller width multipliers, you do not require external logic to create the shift register chain because the input shift registers are internal to the DSP block. This implementation significantly reduces the logical element (LE) resources required, avoids routing congestion, and results in predictable timing.

The first multiplier in every half DSP block (top- and bottom-half) in Arria II GX devices has a multiplexer for the first multiplier B-input (lower-leg input) register to select between general routing and loopback, as shown in Figure 4–4 on page 4–7. In loopback mode, the most significant 18-bit registered outputs are connected as feedback to the multiplier input of the first top multiplier in each half DSP block. Loopback modes are used by recursive filters where the previous output is required to compute the current output.

Loopback mode is described in detail in "Two-Multiplier Adder Sum Mode" on page 4–20.

Table 4–3 shows the summary of input register modes for the DSP block.

Table 4–3. Input Register Modes

Register Input Mode *(1)*	9 × 9	12 × 12	18 × 18	36 × 36	Double
Parallel input	✓	✓	✓	✓	✓
Shift register input *(2)*	—	—	✓	—	—
Loopback input *(3)*	—	—	✓	—	—

Notes to Table 4–3:

(1) The multiplier operand input wordlengths are statically configured at compile time.

(2) Available only on the A-operand.

(3) Only one loopback input is allowed per half-block. For details, refer to Figure 4–12 on page 4–21.

Multiplier and First-Stage Adder

The multiplier stage supports 9 × 9, 12 × 12, 18 × 18, or 36 × 36 multipliers. Other wordlengths are padded up to the nearest appropriate native wordlength; for example, 16 × 16 would be padded up to use 18 × 18. For more information, refer to "Independent Multiplier Modes" on page 4–13. Depending on the data width of the multiplier, a single DSP block can perform many multiplications in parallel.

Each multiplier operand can be a unique signed or unsigned number. Two dynamic signals, `signa` and `signb`, control the representation of each operand, respectively. A `logic 1` value on the `signa/signb` signal indicates that `data A/dataB` is a signed number; a `logic 0` value indicates an unsigned number. Table 4–4 shows the sign of the multiplication result for the various operand sign representations. The result of the multiplication is signed if any one of the operands is a signed value.

Table 4–4. Multiplier Sign Representation

Data A (signa Value)	Data B (signb Value)	Result
Unsigned (logic 0)	Unsigned (logic 0)	Unsigned
Unsigned (logic 0)	Signed (logic 1)	Signed
Signed (logic 1)	Unsigned (logic 0)	Signed
Signed (logic 1)	Signed (logic 1)	Signed

Each half-block has its own `signa` and `signb` signal. Therefore, all `data A` inputs feeding the same DSP half-block must have the same sign representation. Similarly, all `data B` inputs feeding the same DSP half-block must have the same sign representation. The multiplier offers full precision regardless of the sign representation in all operational modes except for full precision 18 x 18 loopback and two-multiplier adder modes. For more information, refer to "Two-Multiplier Adder Sum Mode" on page 4–20.

When signals `signa` and `signb` are unused, the Quartus II software sets the multiplier to perform unsigned multiplication by default.

The outputs of the multipliers are the only outputs that can feed into the first-stage adder, as shown in Figure 4–4 on page 4–7. There are four first-stage adders in a DSP block (two adders per half DSP block). The first-stage adder block has the ability to perform addition and subtraction. The control signal for addition or subtraction is static and has to be configured upon compile time. The first-stage adders are used by the sum modes to compute the sum of two multipliers, 18 × 18-complex multipliers, and to perform the first stage of a 36 × 36 multiply and shift operation.

Depending on your specifications, the output of the first-stage adder has the option to feed into the pipeline registers, second-stage adder, round and saturation unit, or the output registers.

Pipeline Register Stage

The output from the first-stage adder can either feed or bypass the pipeline registers, as shown in Figure 4–4 on page 4–7. Pipeline registers increase the DSP block's maximum performance (at the expense of extra cycles of latency), especially when using the subsequent DSP block stages. Pipeline registers split up the long signal path between the input-registers/multiplier/first-stage adder and the second-stage adder/round-and-saturation/output-registers, creating two shorter paths.

Second-Stage Adder

There are four individual 44-bit second-stage adders per DSP block (two adders per half DSP block). You can configure the second-stage adders as follows:

- The final stage of a 36-bit multiplier
- A sum of four (18 × 18)
- An accumulator (44-bits maximum)
- A chained output summation (44-bits maximum)

☞ You can use the chained-output adder at the same time as a second-level adder in chained output summation mode.

The output of the second-stage adder has the option to go into the round and saturation logic unit or the output register.

☞ You cannot use the second-stage adder independently from the multiplier and first-stage adder.

Round and Saturation Stage

Round and saturation logic units are located at the output of the 44-bit second-stage adder (the round logic unit followed by the saturation logic unit). There are two round and saturation logic units per half DSP block. The input to the round and saturation logic unit can come from one of the following stages:

- Output of the multiplier (independent multiply mode in 18 × 18)
- Output of the first-stage adder (two-multiplier adder)
- Output of the pipeline registers
- Output of the second-stage adder (four-multiplier adder, multiply-accumulate mode in 18 × 18)

These stages are described in detail in "Operational Mode Descriptions" on page 4–13.

The round and saturation logic unit is controlled by the dynamic round and saturate signals, respectively. A `logic 1` value on the round and/or saturate enables the round and/or saturate logic unit, respectively.

☞ You can use the round and saturation logic units together or independently.

Second Adder and Output Registers

The second adder register and output register banks are two banks of 44-bit registers that can also be combined to form larger 72-bit banks to support 36 × 36 output results.

The outputs of the different stages in the Arria II GX devices are routed to the output registers through an output selection unit. Depending on the operational mode of the DSP block, the output selection unit selects whether the outputs of the DSP blocks comes from the outputs of the multiplier block, first-stage adder, pipeline registers, second-stage adder, or the round and saturation logic unit. The output selection unit is set automatically by the software, based on the DSP block operational mode you specified, and has the option to either drive or bypass the output registers. The exception is when the block is used in shift mode, in which case you dynamically controls the output-select multiplexer directly.

When the DSP block is configured in chained cascaded output mode, both of the second-stage adders are used. The first adder is used for performing four-multiplier adder and the second is used for the chainout adder. The outputs of the four-multiplier adder are routed to the second-stage adder registers before it enters the chainout adder. The output of the chainout adder goes to the regular output register bank. Depending on the configuration, the chainout results can be routed to the input of the next half-block's chainout adder input or to the general fabric (functioning as regular output registers).

The second-stage and output registers are triggered by the positive edge of the clock signal and are cleared on power up. The following DSP block signals control the output registers in the DSP block:

- `clock[3..0]`
- `ena[3..0]`
- `aclr[3..0]`

Operational Mode Descriptions

This section describes the operation modes of Arria II GX devices.

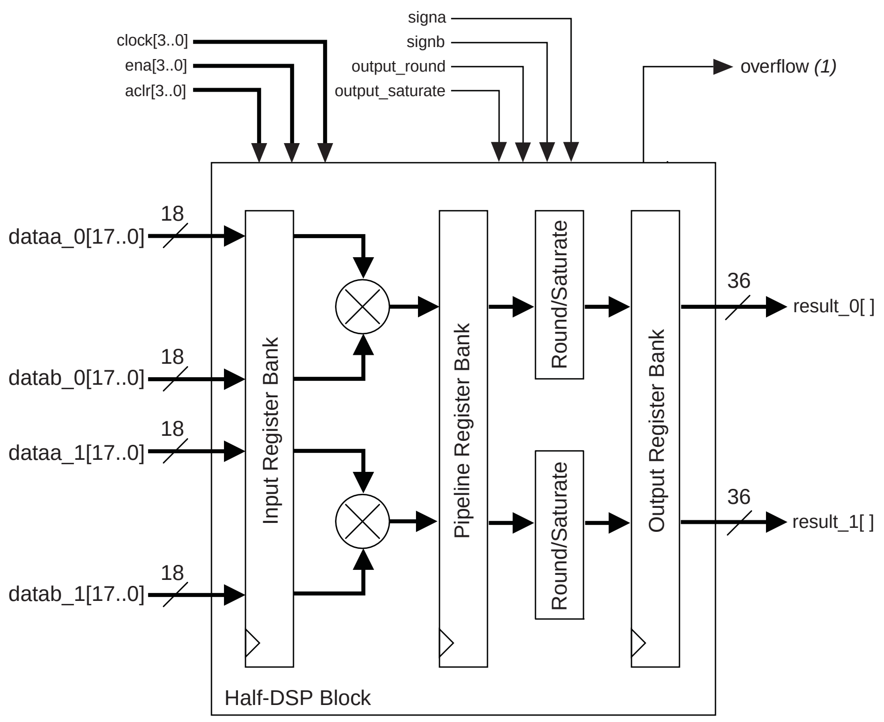
Independent Multiplier Modes

In independent input and output multiplier mode, the DSP block performs individual multiplication operations for general-purpose multipliers.

9-, 12-, and 18-Bit Multiplier

You can configure each DSP block multiplier for 9-, 12-, or 18-bit multiplication. A single DSP block can support up to eight individual 9 × 9 multipliers, six 12 × 12 multipliers, or up to four individual 18 × 18 multipliers. For operand widths up to 9-bits, a 9 × 9 multiplier is implemented. For operand widths from 10 to 12 bits, a 12 × 12 multiplier is implemented and for operand widths from 13 to 18 bits, an 18 × 18 multiplier is implemented. This is done by the Quartus II software by zero-padding the LSBs. Figure 4–6, Figure 4–7, and Figure 4–8 show the DSP block in the independent multiplier operation mode. A list of DSP block dynamic signals is shown in Table 4–9 on page 4–30.

Figure 4–6. 18-Bit Independent Multiplier Mode Shown for Half-DSP Block



Note to Figure 4–6:

(1) Block output for accumulator overflow and saturate overflow.

Figure 4–7. 12-Bit Independent Multiplier Mode Shown for Half-DSP Block

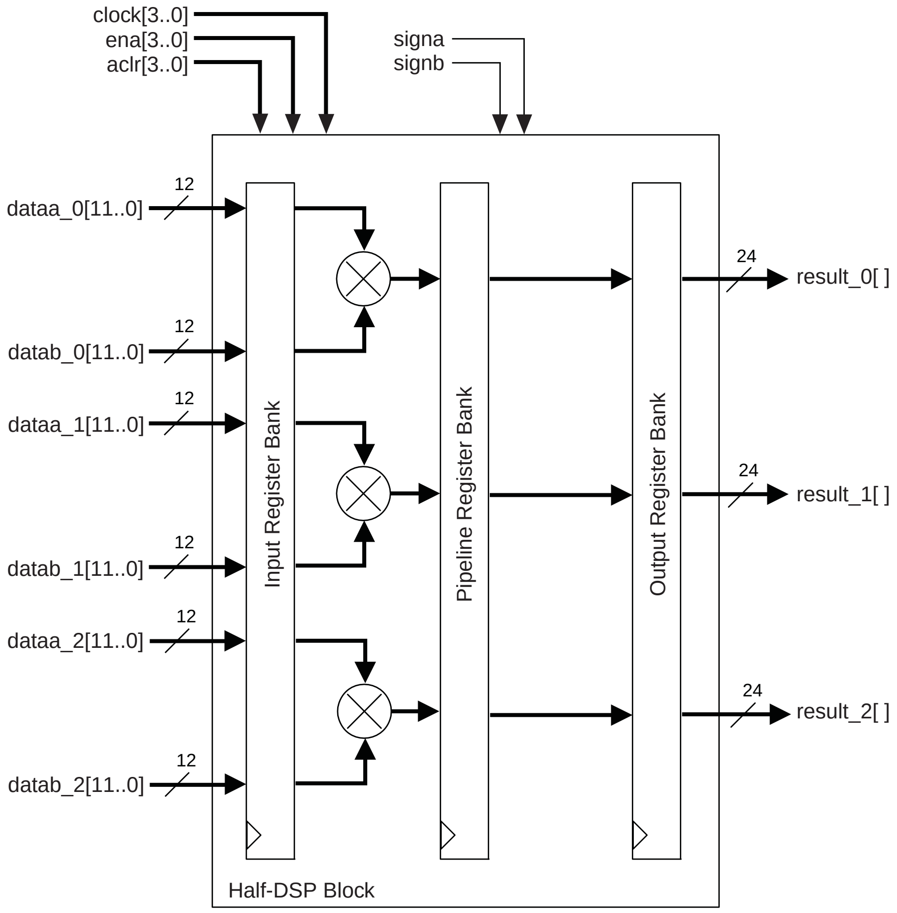
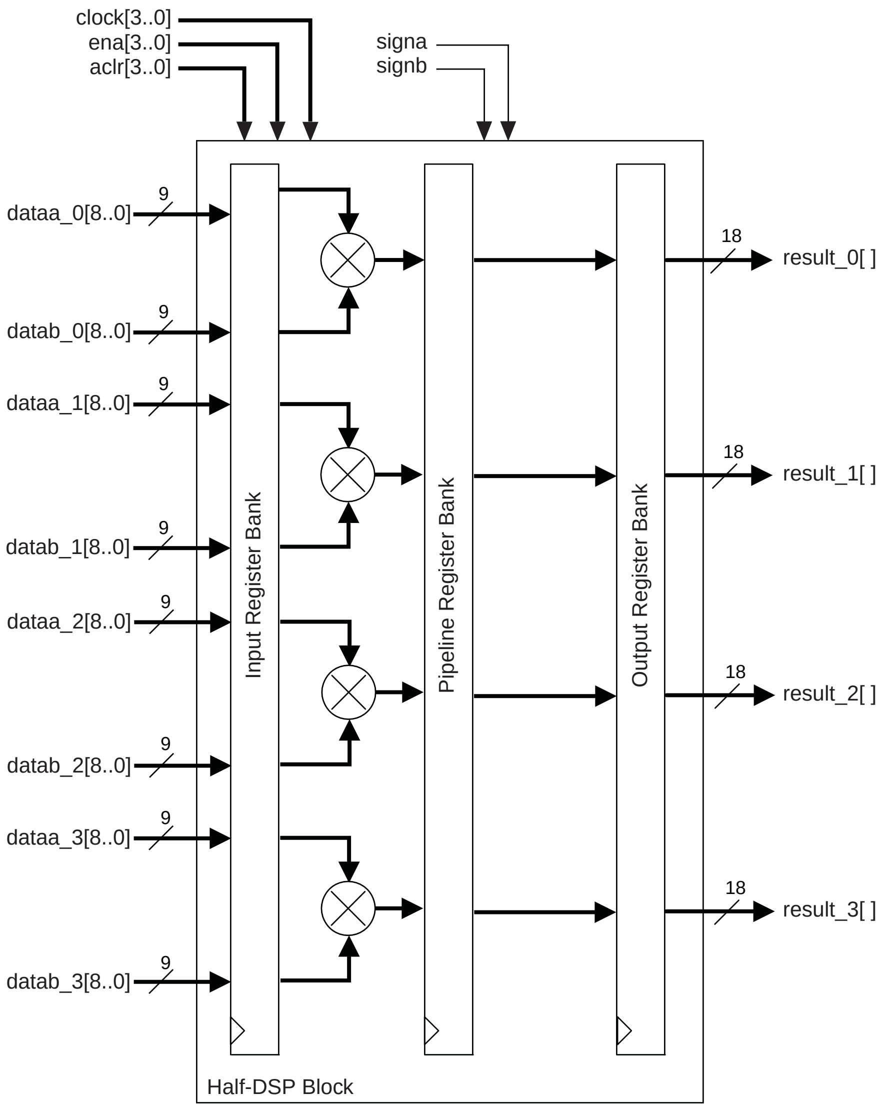


Figure 4–8. 9-Bit Independent Multiplier Mode Shown for Half-Block



The multiplier operands can accept signed integers, unsigned integers, or a combination of both. You can change the `signa` and `signb` signals dynamically and can be registered in the DSP block. Additionally, the multiplier inputs and result can be registered independently. You can use the pipeline registers in the DSP block to pipeline the multiplier result, increasing the performance of the DSP block.

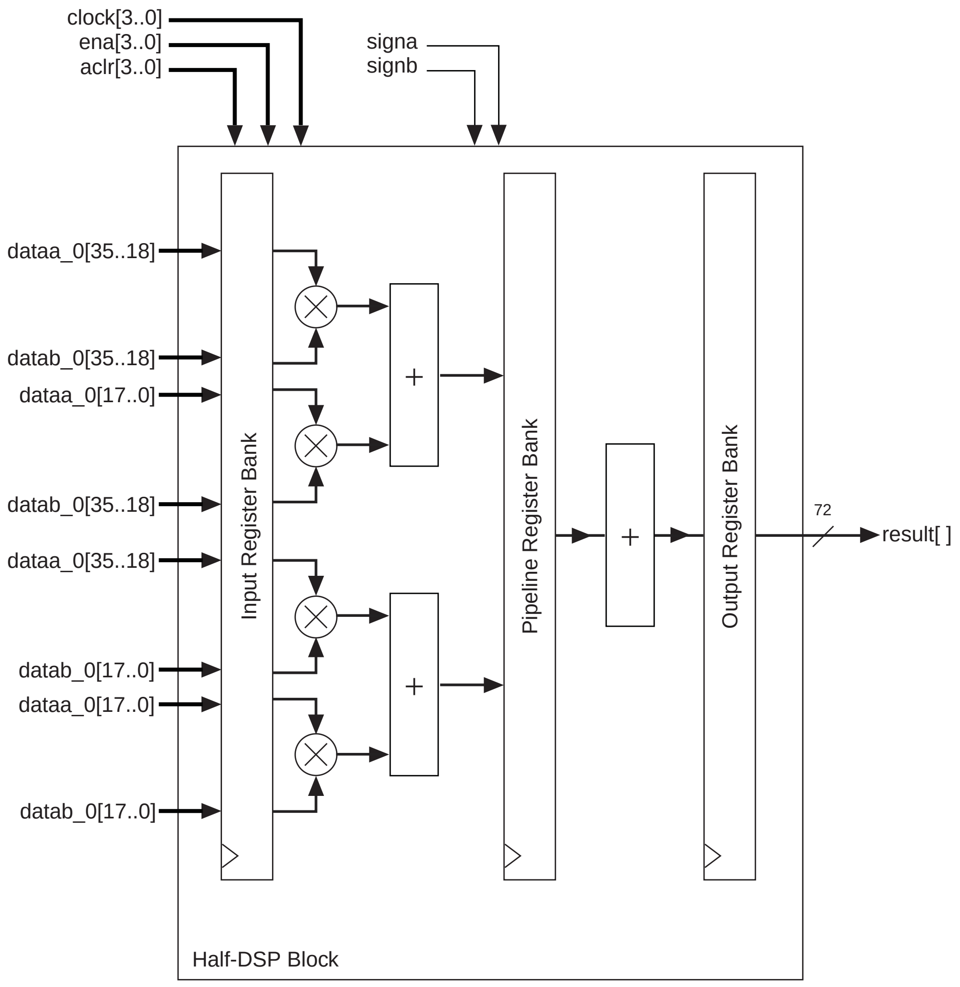
The round and saturation logic unit is supported for 18-bit independent multiplier mode only.

36-Bit Multiplier

You can construct a 36 × 36 multiplier using four 18 × 18 multipliers. This simplification fits into one half-DSP block and is implemented in the DSP block automatically by selecting 36 × 36 mode. The 36-bit multiplier is also under the independent multiplier mode but uses the entire half DSP block, including the dedicated hardware logic after the pipeline registers to implement the 36 × 36-bit multiplication operation, as shown in Figure 4–9.

The 36-bit multiplier is useful for applications requiring more than 18-bit precision; for example, for the mantissa multiplication portion of single precision and extended single precision floating-point arithmetic applications.

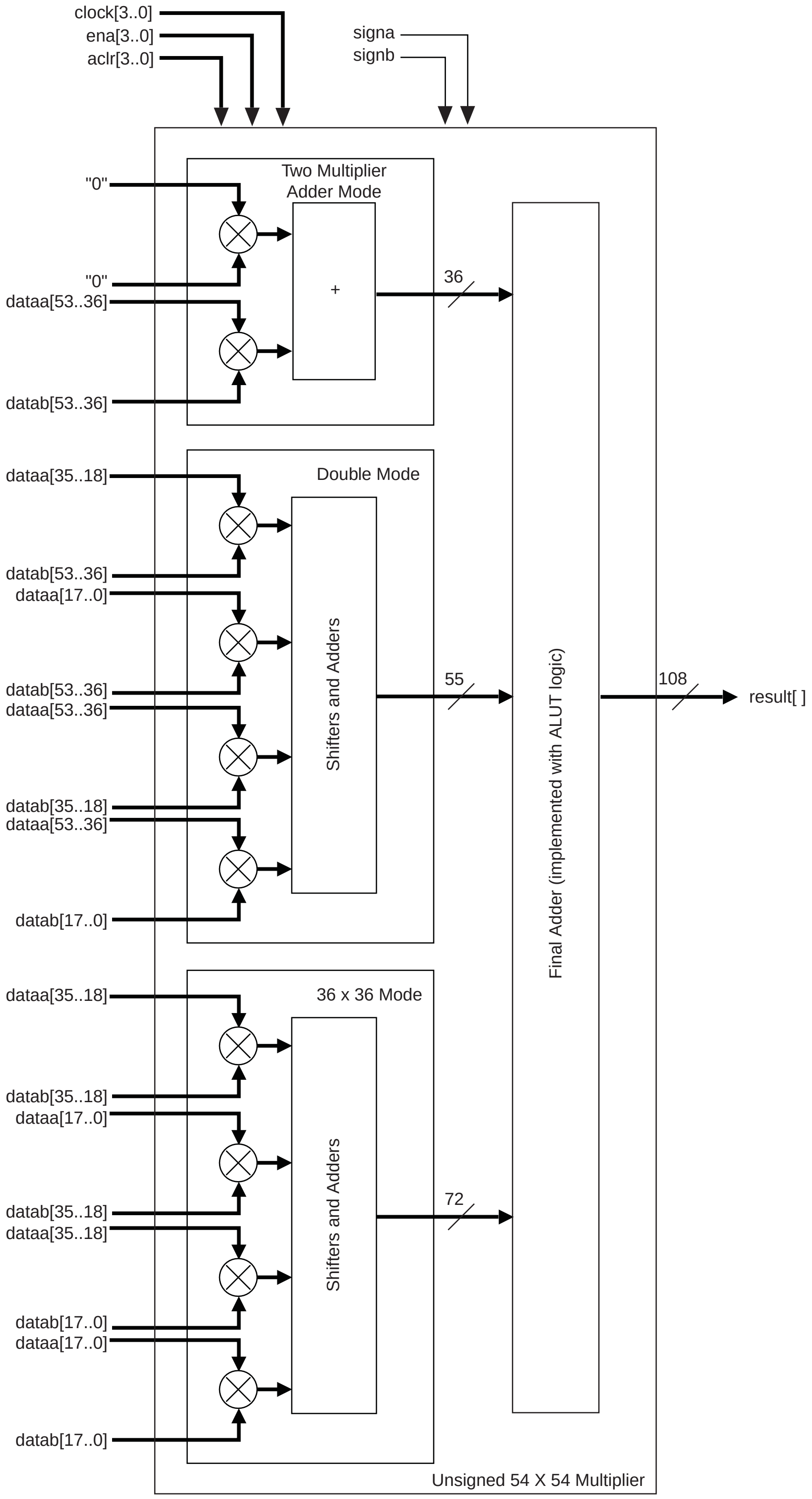
Figure 4–9. 36-Bit Independent Multiplier Mode Shown for Half-DSP Block



Double Multiplier

You can configure the Arria II GX DSP block to support an unsigned 54 × 54-bit multiplier that is required to compute the mantissa portion of an IEEE double precision floating point multiplication. You can build a 54 × 54-bit multiplier using basic 18 × 18 multipliers, shifters, and adders. To efficiently use the Arria II GX DSP block's built in shifters and adders, a special double mode (partial 54 × 54 multiplier) is available that is a slight modification to the basic 36 × 36 multiplier mode. Figure 4–10 shows a 54 × 54-bit multiplier which include the special double mode multiplier.

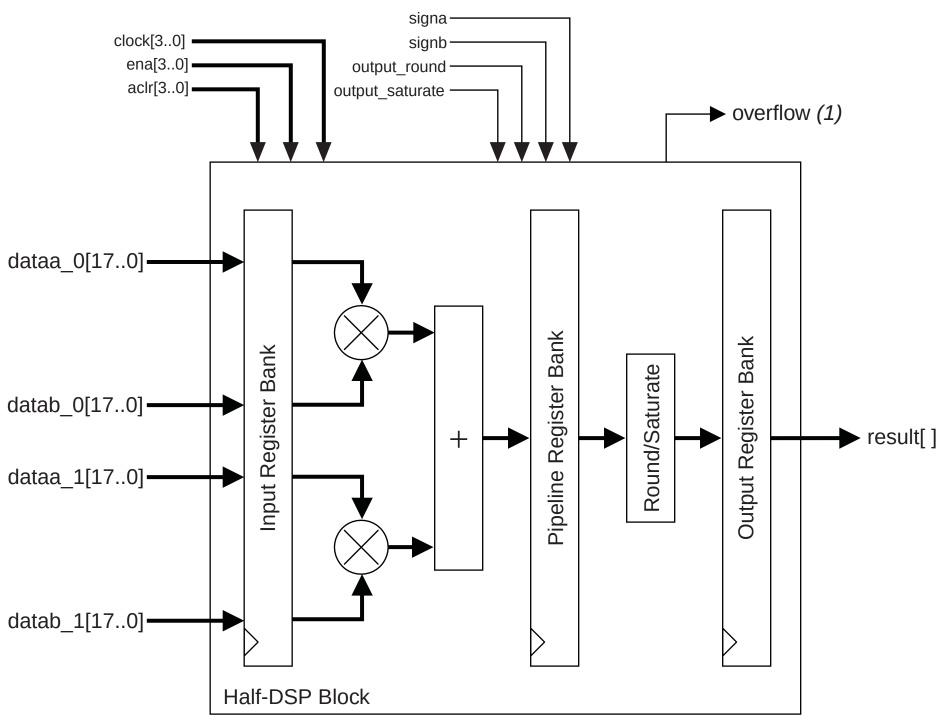
Figure 4–10. Unsigned 54 × 54-Bit Multiplier



Two-Multiplier Adder Sum Mode

In the two-multiplier adder configuration, the DSP block can implement four 18-bit two-multiplier adders (2 two-multiplier adders per half DSP block). You can configure the adders to take the sum or difference of two multiplier outputs. Summation or subtraction has to be selected at compile time. The two-multiplier adder function is useful for applications such as FFTs, complex FIR, and IIR filters. Figure 4–11 shows the DSP block configured in the two-multiplier adder mode.

Figure 4–11. Two-Multiplier Adder Mode Shown for Half-DSP Block *(Note 2)*



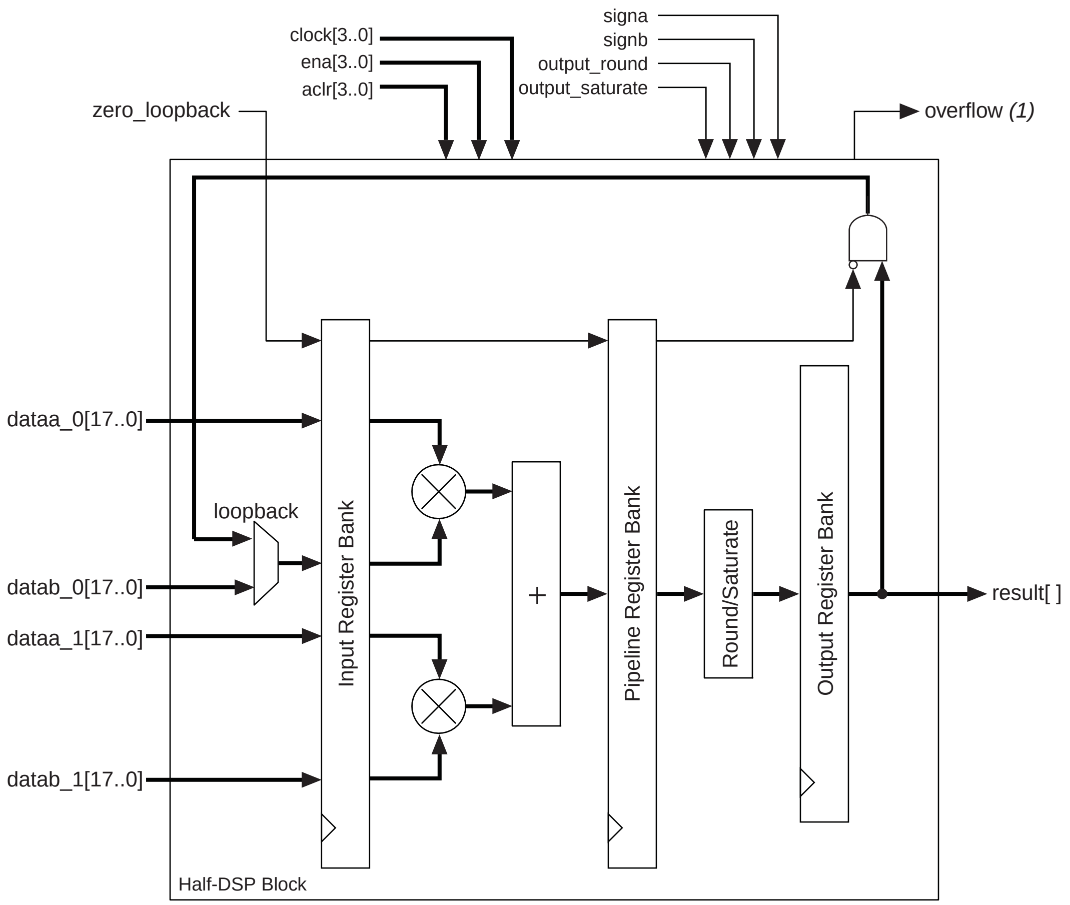
Notes to Figure 4–11:

(1) Block output for accumulator overflow and saturate overflow.
(2) In a half-DSP block, you can implement 2 two-multiplier adders.

The loopback mode is a sub-feature of the two-multiplier adder mode. Figure 4–12 shows the DSP block configured in the loopback mode. This mode takes the 36-bit summation result of the two multipliers and feeds back the most significant 18-bits to the input. The lower 18-bits are discarded. You have the option to disable or zero-out the loopback data with the dynamic `zero_loopback` signal. A `logic 1` value on the `zero_loopback` signal selects the `zeroed` data or disables the looped back data, while a `logic 0` selects the looped back data.

☞ The option to use the loopback mode or the general two-multiplier adder mode must be selected at compile time.

Figure 4–12. Loopback Mode for Half-DSP Block



Note to Figure 4–12:

(1) Block output for accumulator overflow and saturate overflow.

For two-multiplier adder mode, if all the inputs are full 18-bit and unsigned, the result requires 37 bits. As the output data width in two-multiplier adder mode is limited to 36 bits, this 37-bit output requirement is not allowed. Any other combination that does not violate the 36-bit maximum result is permitted; for example, two 16 × 16 signed two-multiplier adders is valid.

Two-multiplier adder mode supports the round and saturation logic unit. You can use pipeline registers and output registers in the DSP block to pipeline the multiplier-adder result, increasing the performance of the DSP block.

18 × 18 Complex Multiply

You can configure the DSP block to implement complex multipliers using the two-multiplier adder mode. A single-half DSP block can implement one 18-bit complex multiplier.

A complex multiplication can be written as shown in Equation 4–4.

Equation 4–4. Complex Multiplication Equation

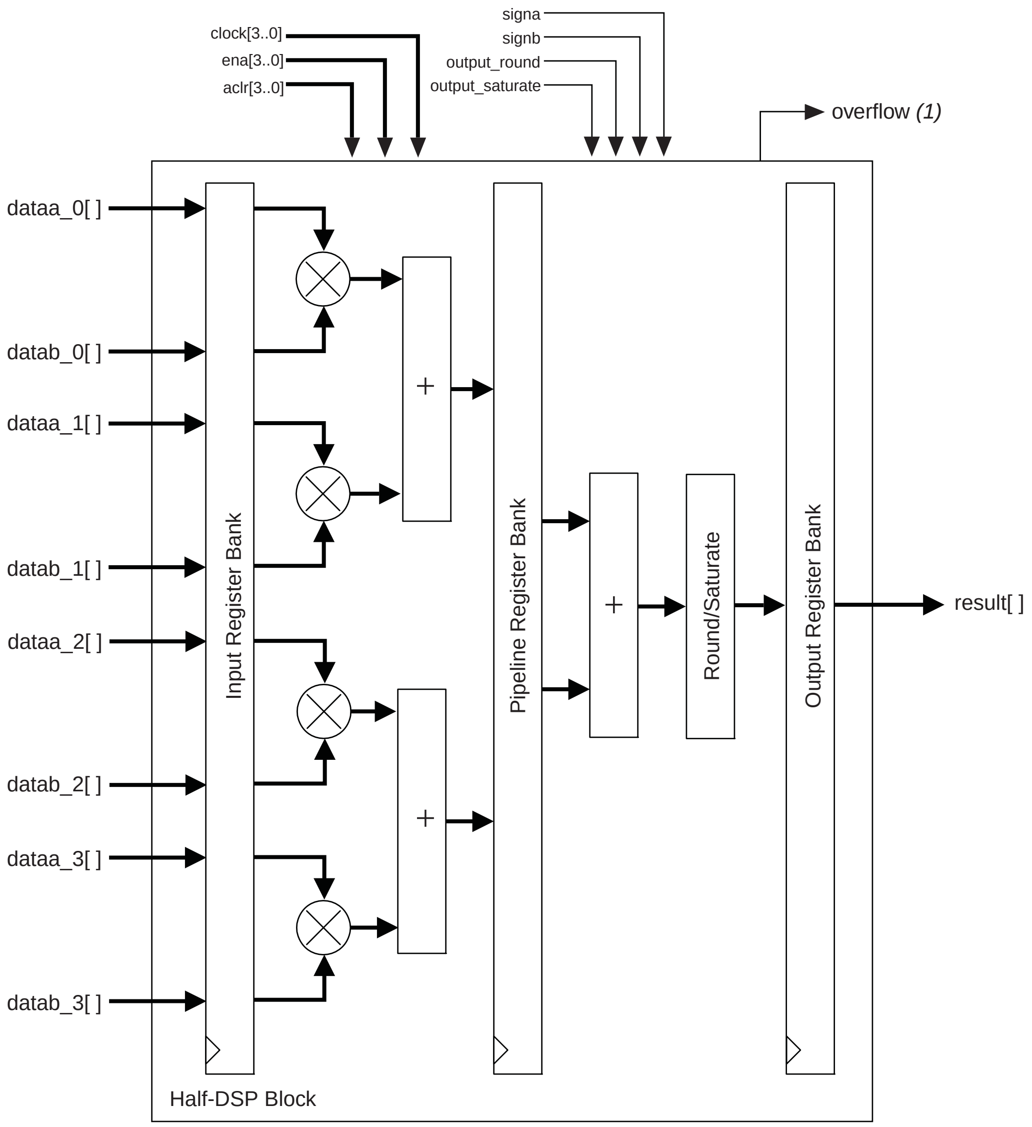
$$(a + jb) \times (c + jd) = ((a \times c)) - (b \times d)) + j((a \times d) + (b \times c))$$

To implement this complex multiplication in the DSP block, the real part $((a \times c) - (b \times d))$ is implemented using two multipliers feeding one subtractor block while the imaginary part $((a \times d) + (b \times c))$ is implemented using another two multipliers feeding an adder block. This mode automatically assumes all inputs are using signed numbers.

Four-Multiplier Adder

In the four-multiplier adder configuration shown in Figure 4–13, the DSP block can implement 2 four-multiplier adders (1 four-multiplier adder per half DSP block). These modes are useful for implementing one-dimensional and two-dimensional filtering applications. The four-multiplier adder is performed in two addition stages. The outputs of two of the four multipliers are initially summed in the two first-stage adder blocks. The results of these two adder blocks are then summed in the second-stage adder block to produce the final four-multiplier adder result, as shown in Equation 4–2 on page 4–3 and Equation 4–3 on page 4–4.

Figure 4–13. Four-Multiplier Adder Mode Shown for Half-DSP Block



Note to Figure 4–13:

(1) Block output for accumulator overflow and saturate overflow.

High-Precision Multiplier Adder Mode

In the high-precision multiplier adder, the DSP block can implement 2 two-multiplier adders, with multiplier precision of 18 × 36 (one two-multiplier adder per DSP half block). This mode is useful in filtering or fast Fourier transform (FFT) applications where a data path greater than 18 bits is required, yet 18 bits is sufficient for coefficient precision. This can occur in cases where that data has a high dynamic range. If the coefficients are fixed, as in FFT and most filter applications, the precision of 18 bits provides a dynamic range over 100 dB, if the largest coefficient is normalized to the maximum 18-bit representation.

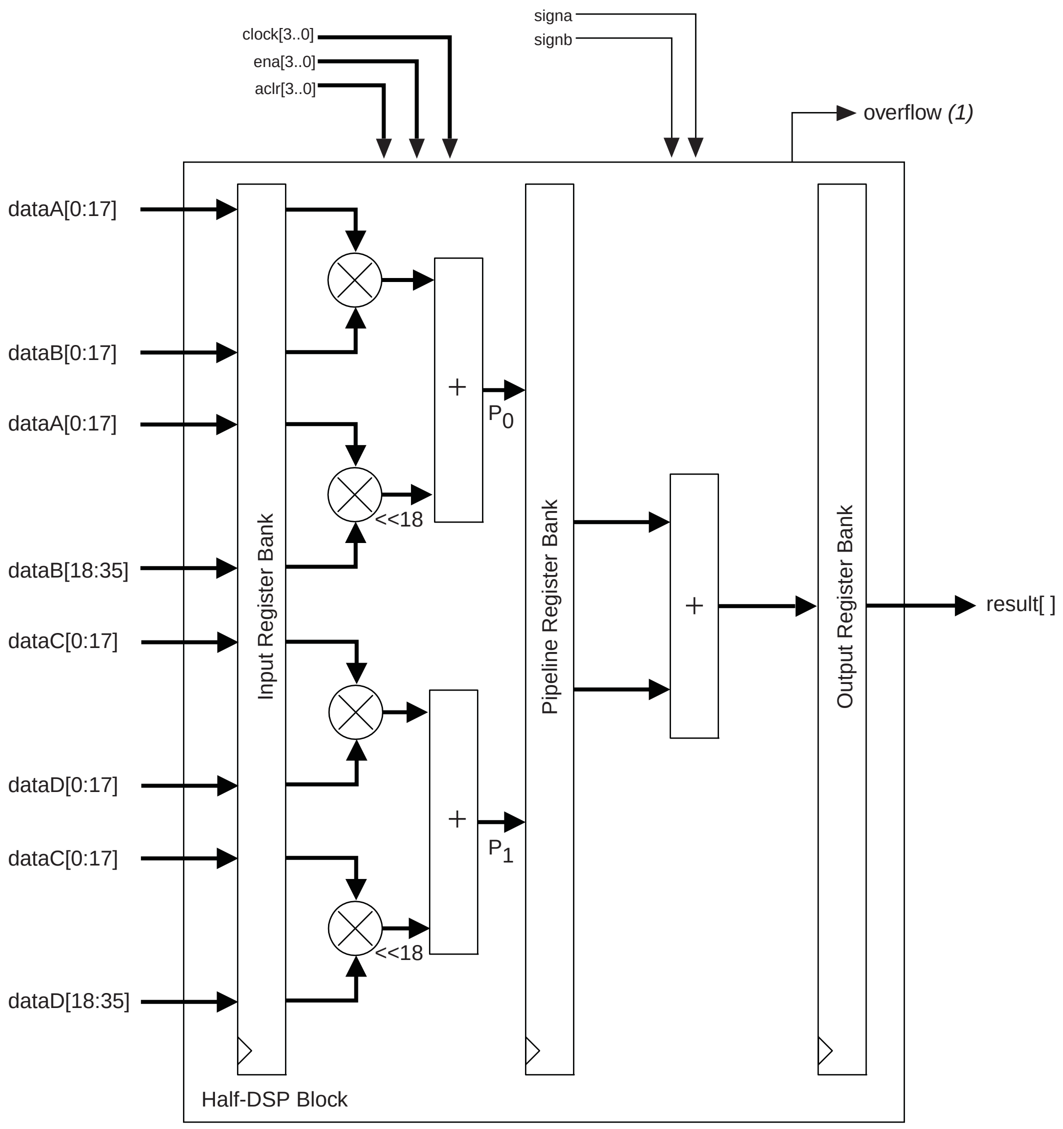
In these situations, the data path can be up to 36 bits, allowing sufficient capacity for bit growth or gain changes in the signal source without loss of precision, which is useful in single precision block floating point applications. As shown in Figure 4–14, the high-precision multiplier is performed in two stages. The sum of the results of the two adders produce the final result:

$Z[54..0] = P_0[53..0] + P_1[53..0]$

where:

$P_0 = A[17..0] \times B[35..0]$ and $P_1 = C[17..0] \times D[35..0]$

Figure 4–14. High-Precision Multiplier Adder Configuration for Half-DSP Block



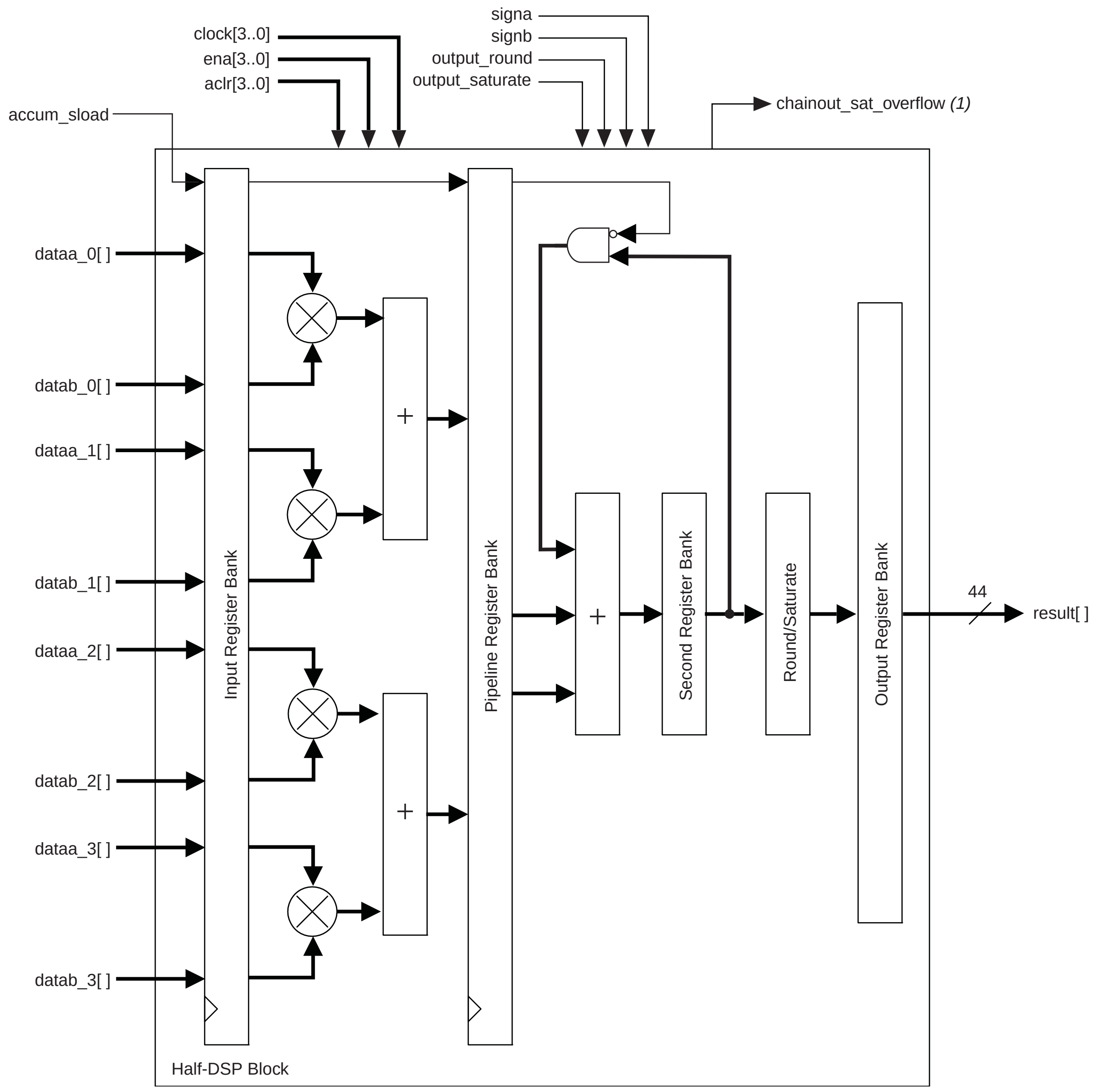
Note to Figure 4–14:

(1) Block output for accumulator overflow and saturate overflow.

Multiply Accumulate Mode

In multiply accumulate mode, the second-stage adder is configured as a 44-bit accumulator or subtractor. The output of the DSP block is looped back to the second-stage adder and added or subtracted with the two outputs of the first-stage adder block according to Equation 4–3 on page 4–4. Figure 4–15 shows the DSP block configured to operate in multiply accumulate mode.

Figure 4–15. Multiply Accumulate Mode Shown for Half-DSP Block



Note to Figure 4–15:

(1) Block output for saturation overflow of chainout.

A single DSP block can implement up to two independent 44-bit accumulators.

The dynamic `accum_sload` control signal is used to clear the accumulation. A `logic 1` value on the `accum_sload` signal synchronously loads the accumulator with the multiplier result only, while a `logic 0` enables accumulation by adding or subtracting the output of the DSP block (accumulator feedback) to the output of the multiplier and first-stage adder.

The control signal for the accumulator and subtractor is static and therefore has to be configured at compile time.

This mode supports the round and saturation logic unit as it is configured as an 18-bit multiplier accumulator.

Shift Modes

Arria II GX devices support the following shift modes for 32-bit input only:

- Arithmetic shift left, `ASL[N]`
- Arithmetic shift right, `ASR[32-N]`
- Logical shift left, `LSL[N]`
- Logical shift right, `LSR[32-N]`
- 32-bit rotator or Barrel shifter, `ROT[N]`

You can switch the shift mode between these modes using the dynamic rotate and shift control signals.

Shift mode in an Arria II GX device can be easily used by a soft embedded processor such as the Nios® II processor to perform the dynamic shift and rotate operation. Figure 4–16 shows the shift mode configuration.

Shift mode makes use of the available multipliers to logically or arithmetically shift left, right, or rotate the desired 32-bit data. The DSP block is configured like the independent 36-bit multiplier mode to perform the shift mode operations.

The arithmetic shift right requires a signed input vector. During arithmetic shift right, the sign is extended to fill the MSB of the 32-bit vector. The logical shift right uses an unsigned input vector. During logical shift right, zeros are padded in the most significant bits shifting the 32-bit vector to the right. The barrel shifter uses an unsigned input vector and implements a rotation function on a 32-bit word length.

Two control signals, `rotate` and `shift_right`, together with the `signa` and `signb` signals, determine the shifting operation. Examples of shift operations are shown in Table 4–5.

Figure 4–16. Shift Operation Mode Shown for Half-DSP Block

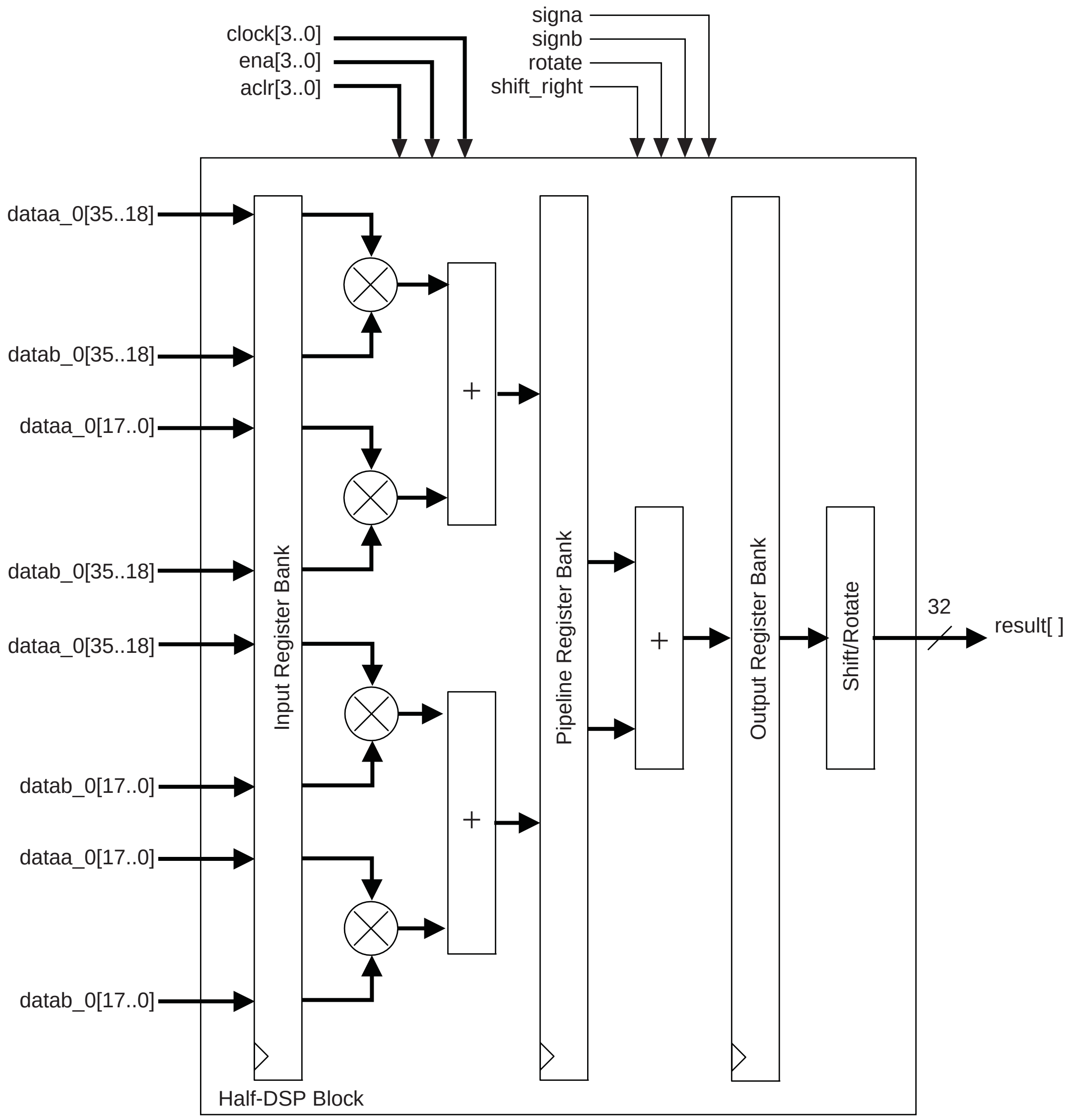


Table 4–5. Examples of Shift Operations

Example	Signa	Signb	Shift	Rotate	A-input	B-input	Result
Logical Shift Left `LSL[N]`	Unsigned	Unsigned	0	0	0×AABBCCDD	0×0000100	0×BBCCDD00
Logical Shift Right `LSR[32-N]`	Unsigned	Unsigned	1	0	0×AABBCCDD	0×0000100	0×000000AA
Arithmetic Shift Left `ASL[N]`	Signed	Unsigned	0	0	0×AABBCCDD	0×0000100	0×BBCCDD00
Arithmetic Shift Right `ASR[32-N]`	Signed	Unsigned	1	0	0×AABBCCDD	0×0000100	0×FFFFFFAA
Rotation `ROT[N]`	Unsigned	Unsigned	0	1	0×AABBCCDD	0×0000100	0×BBCCDDAA

Rounding and Saturation Mode

Round and saturation functions are often required in DSP arithmetic. Rounding is used to limit bit growth and its side effects; saturation is used to reduce overflow and underflow side effects.

Two rounding modes are supported in Arria II GX devices:

- Round-to-nearest-integer mode
- Round-to-nearest-even mode

You must select one of the two options at compile time.

Round-to-nearest-integer provides the biased rounding support and is the simplest form of rounding commonly used in DSP arithmetic. The round-to-nearest-even method provides unbiased rounding support and is used where DC offsets are a concern. Table 4–6 shows how round-to-nearest-even works. Examples of the difference between the two modes are shown in Table 4–7. In this example, a 6-bit input is rounded to 4 bits. You can observe from Table 4–7 that the main difference between the two rounding options is when the residue bits are exactly half way between its nearest two integers and the LSB is zero (even).

Table 4–6. Example of Round-To-Nearest-Even Mode

6- to 4-bits Rounding	Odd/Even (Integer)	Fractional	Add to Integer	Result
010111	×	> 0.5 (11)	1	0110
001101	×	< 0.5 (01)	0	0011
001010	Even (0010)	= 0.5 (10)	0	0010
001110	Odd (0011)	= 0.5 (10)	1	0100
110111	×	> 0.5 (11)	1	1110
101101	×	< 0.5 (01)	0	1011
110110	Odd (1101)	= 0.5 (10)	1	1110
110010	Even (1100)	= 0.5 (10)	0	1100

Table 4–7. Comparison of Round-to-Nearest-Integer and Round-to-Nearest-Even

Round-To-Nearest-Integer	Round-To-Nearest-Even
010111 ⇨ 0110	010111 ⇨ 0110
001101 ⇨ 0011	001101 ⇨ 0011
001010 ⇨ 0011	001010 ⇨ 0010
001110 ⇨ 0100	001110 ⇨ 0100
110111 ⇨ 1110	110111 ⇨ 1110
101101 ⇨ 1011	101101 ⇨ 1011
110110 ⇨ 1110	110110 ⇨ 1110
110010 ⇨ 1101	110010 ⇨ 1100

Two saturation modes are supported in Arria II GX devices:

- Asymmetric saturation mode
- Symmetric saturation mode

You must select one of the two options at compile time.

In 2's complement format, the maximum negative number that can be represented is $-2^{(n-1)}$,while the maximum positive number is $2^{(n-1)} - 1$. Symmetrical saturation limits the maximum negative number to $-2^{(n-1)} + 1$. For example, for 32 bits:

- Asymmetric 32-bit saturation: Max = 0×7FFFFFFF, Min = 0×80000000
- Symmetric 32-bit saturation: Max = 0×7FFFFFFF, Min = 0×80000001

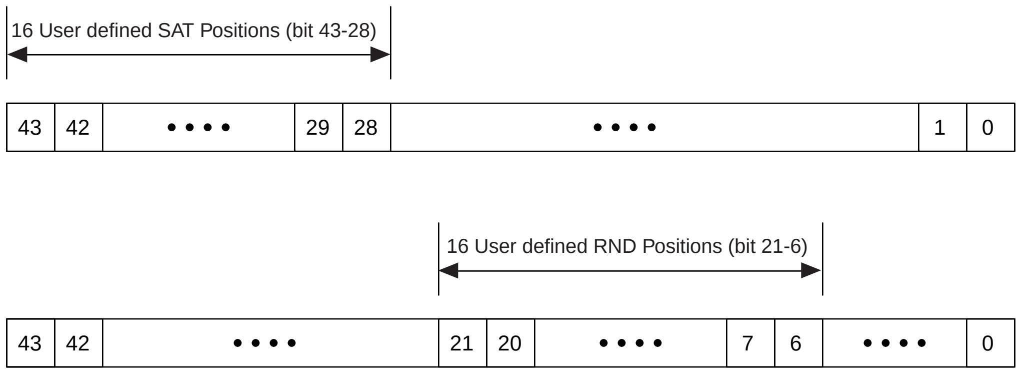
Table 4–8 shows how the saturation works. In this example, a 44-bit input is saturated to 36-bits.

Table 4–8. Examples of Saturation

44 to 36 Bits Saturation	Symmetric SAT Result	Asymmetric SAT Result
5926AC01342h	7FFFFFFFFh	7FFFFFFFFh
ADA38D2210h	800000001h	800000000h

Arria II GX devices have up to 16 configurable bit positions out of the 44-bit bus (`[43:0]`) for the round and saturate logic unit providing higher flexibility. You must select the 16 configurable bit positions at compile time. These 16-bit positions are located at bits `[21:6]` for rounding and `[43:28]` for saturation, as shown in Figure 4–17.

Figure 4–17. Round and Saturation Locations


16 User defined SAT Positions (bit 43-28)
43
42
• • • •
29
28
• • • •
1
0
16 User defined RND Positions (bit 21-6)
43
42
• • • •
21
20
• • • •
7
6
• • • •
0

For symmetric saturation, the RND bit position is also used to determine where the LSP for the saturated data is located.

You can use the rounding and saturation function as described in regular supported multiplication operations shown in Table 4–2 on page 4–5. However, for accumulation type operations, the following convention is used.

The functionality of the round logic unit is in the format of:

Result = RND[$\Sigma(A \times B)$], when used for an accumulation type of operation.

Likewise, the functionality of the saturation logic unit is in the format of:

Result = SAT[$\Sigma(A \times B)$], when used for an accumulation type of operation.

If both the round and saturation logic units are used for an accumulation type of operation, the format is:

Result = SAT[RND[$\Sigma(A \times B)$]]

DSP Block Control Signals

The Arria II GX DSP block is configured using a set of static and dynamic signals. At run time, you can configure the DSP block dynamic signals to toggled or not. Table 4–9 and Table 4–10 show a list of dynamic signals for the DSP block.

Table 4–9. DSP Block Dynamic Signals per Half-DSP Block (Part 1 of 2)

Signal Name	Function	Count
■ `signa` ■ `signb`	Signed/unsigned control for all multipliers and adders. `signa` for "multiplicand" input bus to `dataa[17:0]` each multiplier. `signb` for "multiplier" input bus `datab[17:0]` to each multiplier. `signa` = 1, `signb` = 1 for signed-signed multiplication `signa` = 1, `signb` = 0 for signed-unsigned multiplication `signa` = 0, `signb` = 1 for unsigned-signed multiplication `signa` = 0, `signb` = 0 for unsigned-unsigned multiplication	2
`output_round`	Round control for first stage round/saturation block. `output_round` = 1 for rounding on multiply output `output_round` = 0 for normal multiply output	1
`chainout_round`	Round control for second stage round/saturation block. `chainout_round` = 1 for rounding on multiply output `chainout_round` = 0 for normal multiply output	1
`output_saturate`	Saturation control for first stage round/saturation block for Q-format multiply. If both rounding and saturation is enabled, saturation is done on the rounded result. `output_saturate` = 1 for saturation support `output_saturate` = 0 for no saturation support	1
`chainout_saturate`	Saturation control for second stage round/saturation block for Q-format multiply. If both rounding and saturation's enabled, saturation is done on the rounded result. `chainout_saturate` = 1 for saturation support `chainout_saturate` = 0 for no saturation support	1

Table 4–9. DSP Block Dynamic Signals per Half-DSP Block (Part 2 of 2)

Signal Name	Function	Count
`accum_sload`	Dynamically specifies whether the accumulator value is zero. `accum_sload` = 0, accumulation input is from the output registers `accum_sload` = 1, accumulation input is set to be zero	1
`zero_chainout`	Dynamically specifies whether the chainout value is zero.	1
`zero_loopback`	Dynamically specifies whether the loopback value is zero.	1
`rotate`	`rotation` = 1, rotation feature is enabled	1
`shift_right`	`shift_right` = 1, shift right feature is enabled	1
—	Total Signals per Half-block	11

Table 4–10. DSP Block Dynamic Signals per Full-DSP Block

Signal Name	Function	Count

`clock0`
`clock1`
`clock2`
`clock3`
	DSP-block-wide clock signals	4

`ena0`
`ena1`
`ena2`
`ena3`
	Input and Pipeline Register enable signals	4

`aclr0`
`aclr1`
`aclr2`
`aclr3`
	DSP block-wide asynchronous clear signals (active low).	4

—	Total Count per Full Block	34

Software Support for Arria II GX Devices

Altera provides two distinct methods for implementing various modes of the DSP block in a design: instantiation and inference. Both methods use the following Quartus II megafunctions:

- LPM_MULT
- ALTMULT_ADD
- ALTMULT_ACCUM
- ALTFP_MULT

You can instantiate the megafunctions in the Quartus II software to use the DSP block. Alternatively, with inference, you can create an HDL design and synthesize it using a third-party synthesis tool (such as LeonardoSpectrum, Synplify, or Quartus II Native Synthesis) that infers the appropriate megafunction by recognizing multipliers, multiplier adders, multiplier accumulators, and shift functions. Using either method, the Quartus II software maps the functionality to the DSP blocks during compilation.

Refer to the Quartus II Software Help for instructions about using the megafunctions and the MegaWizard Plug-In Manager.

For more information, refer to *Section III: Synthesis* in volume 1 of the *Quartus II Handbook*.

Document Revision History

Table 4–11 shows the revision history for this chapter.

Table 4–11. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial Release.	—

5. Clock Networks and PLLs in Arria II GX Devices

AIIGX51005-1.0

Introduction

Arria® II GX devices provide a hierarchical clock structure and multiple PLLs with advanced features. Arria II GX devices provide dedicated global clock networks (GCLKs), regional clock networks (RCLKs), and periphery clock networks (PCLKs).

This chapter contains the following sections:

Clock Networks in Arria II GX Devices

The GCLKs, RCLKs, and PCLKs available in Arria II GX devices are organized into hierarchical clock structures that provide up to 148 unique clock domains (16 GCLK + 48 RCLK + 84 PCLK) in the Arria II GX device and allows up to 52 unique GCLK, RCLK, and PCLK clock sources (16 GCLK + 12 RCLK + 24 PCLK) per device quadrant. Table 5–1 shows the clock resources available in Arria II GX devices.

Table 5–1. Clock Resources in Arria II GX Devices

Clock Resource	Number of Resources Available	Source of Clock Resource
Clock input pins	12 Single-ended (6 Differential)	`CLK[4..15] DIFFCLK_[0..5]p/n` pins
Global clock networks	16	`CLK[4..15]` pins, PLL clock outputs, PLD-transceiver interface clocks, and logic array
Regional clock networks	48	`CLK[4..15]` pins, PLL clock outputs, PLD-transceiver interface clocks, and logic array
Periphery clock networks	84 (24 per device quadrant) *(1)*	DPA clock outputs, PLD-transceiver interface clocks, horizontal I/O pins, and logic array
GCLKs/RCLKs per quadrant	28	16 GCLKs + 12 RCLKs
GCLKs/RCLKs per device	64	16 GCLKs + 48 RCLKs

Note to Table 5–1:

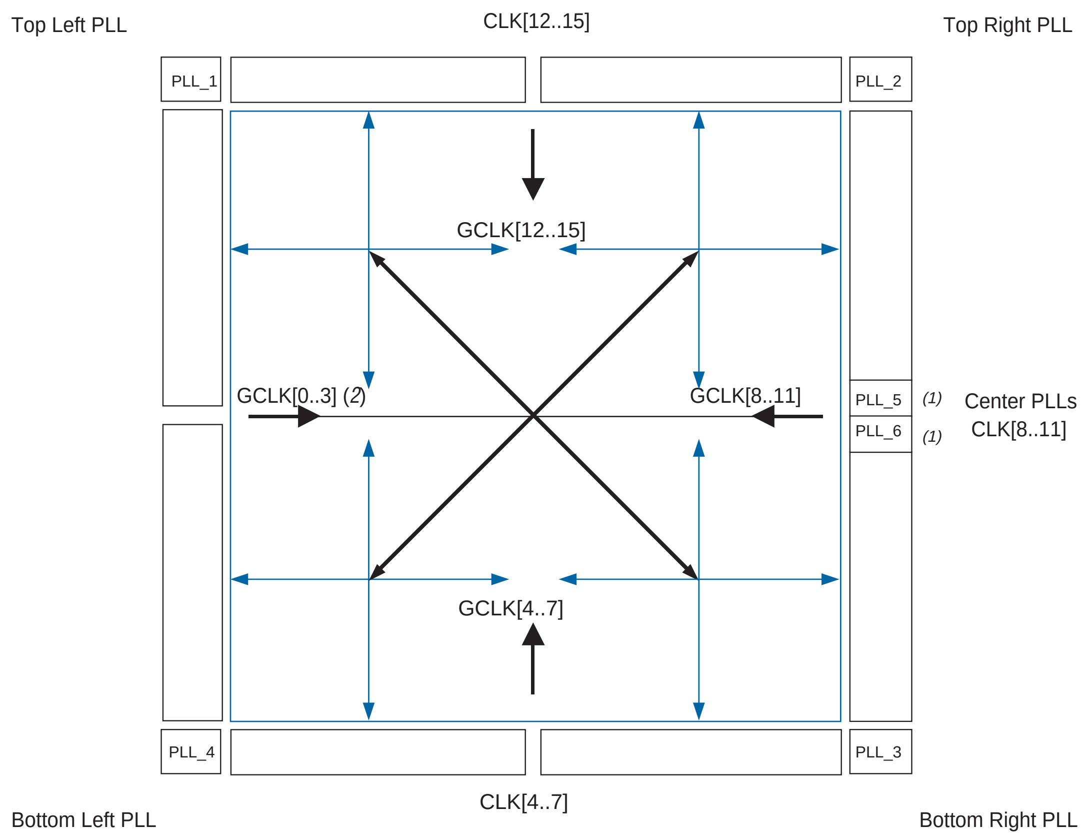
(1) There are 33 PCLKs in the EP2AGX20 and EP2AGX30 devices, where 9 are on the left side and 24 on the right side. There are 50 PCLKs in the EP2AGX45 and EP2AGX65 devices, where 18 are on the left side and 32 on the right side. There are 59 PCLKs in the EP2AGX95 and EP2AGX125 device, where 27 are on the left side and 32 on the right side. There are 84 PCLKS in the EP2AGX190 and EP2AGX260 devices, where 36 are on the left side and 48 on the right side.

Arria II GX devices have up to 12 dedicated single-ended clock pins or 6 dedicated differential clock pins (`DIFFCLK_[0..5]p` and `DIFFCLK_[0..5]n`) that can drive either the GCLK or RCLK networks. These clock pins are arranged on the three sides (top, bottom, and right sides) of the Arria II GX device, as shown in Figure 5–1 and Figure 5–2.

Global Clock Networks

Arria II GX devices provide up to 16 GCLKs that can drive throughout the entire device, serving as low-skew clock sources for functional blocks like adaptive logic modules (ALMs), digital signal processing (DSP) blocks, embedded memory blocks, and PLLs. Arria II GX I/O elements (IOEs) and internal logic can also drive GCLKs to create internally generated global clocks and other high fan-out control signals; for example, synchronous or asynchronous clears and clock enables. Figure 5–1 shows CLK pins and PLLs that can drive GCLK networks in Arria II GX devices.

Figure 5–1. Global Clock Networks



Notes to Figure 5–1:

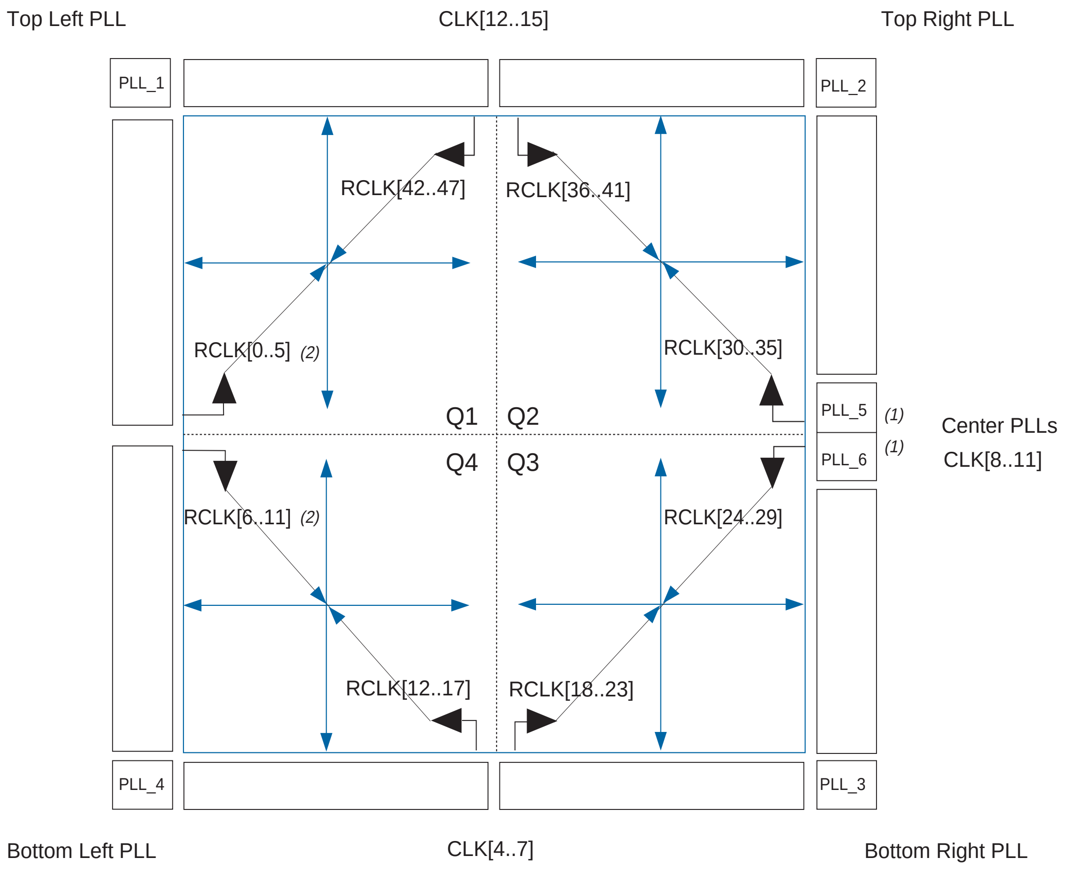
(1) `PLL_5` and `PLL_6` are only available in EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260 devices.

(2) Because there are no dedicated clock pins on the left side of an Arria II GX device, `GCLK[0..3]` are not driven by any clock pins.

Regional Clock Networks

The regional clock (RCLK) networks only pertain to the quadrant they drive into. Arria II GX devices contain 48 RCLK networks that provide the lowest clock delay and skew for logic contained in a single device quadrant. Arria II GX I/O elements and internal logic in a given quadrant can also drive RCLKs to create internally generated regional clocks and other high fan-out control signals; for example, synchronous or asynchronous clears and clock enables. Figure 5–2 shows CLK pins and PLLs that can drive RCLK networks in Arria II GX devices.

Figure 5–2. Regional Clock Networks in Arria II GX Devices



Notes to Figure 5–2:

(1) `PLL_5` and `PLL_6` are only available in EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260 devices.

(2) `RCLK[0..11]` is not driven by any clock pins because there are no dedicated clock pins on the left side of the Arria II GX devices.

Periphery Clock Networks

Periphery clock (PCLK) networks are a collection of individual clock networks driven from the periphery of the Arria II GX device. Clock outputs from the DPA block, PLD-transceiver interface clocks, horizontal I/O pins, and internal logic can drive the PCLK networks. The EP2AGX20 and EP2AGX30 devices contain 33 PCLKs; the EP2AGX45 and EP2AGX65 devices contain 50 PCLKs; the EP2AGX95 and EP2AGX125 devices contain 59 PCLKs; the EP2AGX190 and EP2AGX260 devices contain 84 PCLKs. These PCLKs have higher skew compared to GCLK and RCLK networks and can be used instead of general purpose routing to drive signals into the Arria II GX device.

The legal clock sources for PCLK networks are clock outputs from the DPA block, PLD-transceiver interface clocks, horizontal I/O pins, and internal logic.

Clocking Regions

Arria II GX devices provide up to 64 distinct clock domains (16 GCLKs + 48 RCLKs) in the entire device. You can use these clock resources to form the following four different types of clock regions:

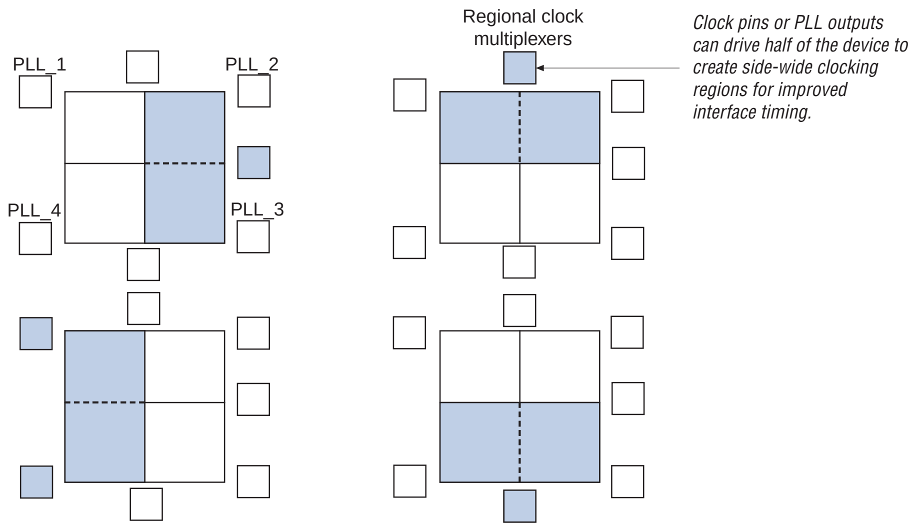
- Entire device clock region
- Regional clock region
- Dual-regional clock region
- Sub-regional clock region (only available in the EP2AGX190 and EP2AGX260 devices)

To form the entire device clock region, a source (not necessarily a clock signal) drives a global clock network that can be routed through the entire device. This clock region has a higher skew compared to other clock regions, but allows the signal to reach every destination in the device. This is a good option for routing global reset/clear signals or routing clocks throughout the device.

To form a regional clock region, a source drives a single-quadrant of the device. This clock region provides the lowest skew in a quadrant and is a good option if all destinations are in a single device quadrant.

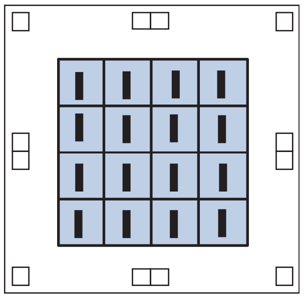
To form a dual-regional clock region, a single source (a clock pin or PLL output) generates a dual-regional clock by driving two regional clock networks (one from each quadrant). This technique allows destinations across two device quadrants to use the same low-skew clock. The routing of this signal on an entire side has approximately the same delay as in a regional clock region. Internal logic can also drive a dual-regional clock network. Corner PLL outputs generate a dual-regional clock network through clock multiplexers that serve the two immediate quadrants of the device. Figure 5–3 shows the dual-regional clock region.

Figure 5–3. Arria II GX Device Dual-Regional Clock Region


Regional clock multiplexers
Clock pins or PLL outputs can drive half of the device to create side-wide clocking regions for improved interface timing.
PLL_1
PLL_2
PLL_4
PLL_3

The sub-regional clock scheme allows the formation of independent sub-regional clock regions for optimal and efficient use of global and regional clock resources. You can partition the device into a maximum of 16 sub-regional clock regions. Each region is driven by a global or regional clock or by an adjacent ALM. This technique allows the formation of optimally sized synchronous clock regions for the best utilization of clock network resources. Figure 5–4, Figure 5–5, and Figure 5–6 show that you can divide the device into 16, 8, or 12 independent sub-regions.

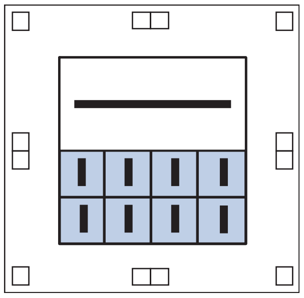
Figure 5–4. Sixteen Independent Sub-Regional Clock Regions *(Note 1)*



Note to Figure 5–4:

(1) The sub-regional clock region is only applicable to the EP2AGX190 and EP2AGX260 devices.

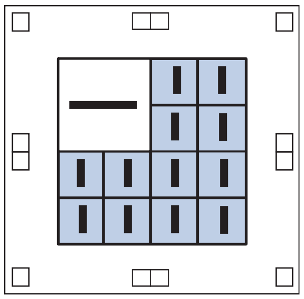
Figure 5–5. Eight Independent Sub-Regional + One Dual-Regional Clock Region *(Note 1)*



Note to Figure 5–5:

(1) The sub-regional clock region is only applicable to the EP2AGX190 and EP2AGX260 devices.

Figure 5–6. Twelve Independent Sub-Regional + One Regional Clock Region *(Note 1)*



Note to Figure 5–6:

(1) The sub-regional clock region is only applicable to the EP2AGX190 and EP2AGX260 devices.

Clock Network Sources

In Arria II GX devices, clock input pins, internal logic, transceiver clocks, and PLL outputs can drive the global and regional clock networks. See Table 5–2 to Table 5–3 for the connectivity between dedicated `CLK[4..15]` pins and the global and regional clock networks.

Dedicated Clock Inputs Pins

The `CLK` pins can either be differential clocks or single-ended clocks. Arria II GX devices supports 6 differential clock inputs or 12 single-ended clock inputs. You can also use the dedicated clock input pins `CLK[4..15]` for high fan-out control signals such as asynchronous clears, presets, and clock enables for protocol signals such as `TRDY` and `IRDY` for PCI through global or regional clock networks.

Logic Array Blocks (LABs)

You can drive up to four signals into each global and regional clock network using LAB-routing to enable internal logic to drive a high fan-out, low-skew signal.

Arria II GX PLLs cannot be driven by internally generated GCLKs or RCLKs. The input clock to the PLL has to come from dedicated clock input pins or pin/PLL-fed GCLKs or RCLKs only.

PLL Clock Outputs

Arria II GX PLLs can drive both GCLK and RCLK networks, as shown in Table 5–5 and Table 5–6.

Table 5–2 shows the connection between the dedicated clock input pins and GCLKs.

Table 5–2. Clock Input Pin Connectivity to Global Clock Networks

Clock Resources	CLK (Pins)											
	4	5	6	7	8	9	10	11	12	13	14	15
`GCLK[0..3]` *(1)*	—	—	—	—	—	—	—	—	—	—	—	—
`GCLK[4..7]`	✓	✓	✓	✓	—	—	—	—	—	—	—	—
`GCLK[8..11]`	—	—	—	—	✓	✓	✓	✓	—	—	—	—
`GCLK[12..15]`	—	—	—	—	—	—	—	—	✓	✓	✓	✓

Note to Table 5–2:

(1) `GCLK[0..3]` is not driven by any clock pins because there are no dedicated clock pins on the left side of the Arria II GX devices.

Table 5–3 shows the connectivity between the dedicated clock input pins and RCLKs in devices. A given clock input pin can drive two adjacent regional clock networks to create a dual-regional clock network.

Table 5–3. Clock Input Pin Connectivity to Regional Clock Networks

Clock Resource	CLK (Pins)											
	4	5	6	7	8	9	10	11	12	13	14	15
`RCLK [12, 14, 16, 18, 20, 22]`	✓	—	✓	—	—	—	—	—		—	—	—
`RCLK [13, 15, 17, 19, 21, 23]`	—	✓	—	✓	—	—	—	—	—	—	—	—
`RCLK [24..35]`	—	—	—	—	✓	✓	✓	✓	—	—	—	—
`RCLK [36, 38, 40, 42, 44, 46]`	—	—	—	—	—	—	—	—	✓	—	✓	—
`RCLK [37, 39, 41, 43, 45, 47]`	—	—	—	—	—	—	—	—	—	✓	—	✓

Clock Input Connections to PLLs

Dedicated clock input pin connectivity to Arria II GX PLLs is shown in Table 5–4.

Table 5–4. Arria II GX Device PLLs and PLL Clock Pin Drivers *(Note 1)*

Dedicated Clock Input Pin (CLK pins)	PLL Number					
	1	2	3	4	5	6
`CLK[4..7]`	—	—	✓	✓	—	—
`CLK[8..11]`	—	✓	✓	—	✓	✓
`CLK[12..15]`	✓	✓	—	—	—	—

Note to Table 5–4:

(1) `PLL_5` and `PLL_6` are connected directly to `CLK[8..11]`. `PLL_1`, `PLL_2`, `PLL_3` and `PLL_4` are driven by the clock input pins through a 4:1 multiplexer.

Clock Output Connections

PLLs in Arria II GX devices can drive up to 24 regional clock networks and 8 global clock networks. Refer to Table 5–5 for Arria II GX PLL connectivity to GCLK networks. The Quartus II® software automatically assigns PLL clock outputs to regional or global clock networks.

Table 5–5 shows how the PLL clock outputs connect to GCLK networks.

Table 5–5. Arria II GX PLL Connectivity to GCLKs

Clock Network	PLL Number					
	1	2	3	4	5	6
`GCLK[0..3]`	✓	—	—	✓	—	—
`GCLK[4..7]`	—	—	✓	✓	—	—
`GCLK[8..11]`	—	✓	✓	—	✓	✓
`GCLK[12..15]`	✓	✓	—	—	—	—

Table 5–6 shows how the PLL clock outputs connect to RCLK networks.

Table 5–6. Arria II GX Regional Clock Outputs From PLLs

Clock Resource	PLL Number					
	1	2	3	4	5	6
`RCLK[0..11]`	✓	—	—	✓	—	—
`RCLK[12..23]`	—	—	✓	✓	—	—
`RCLK[24..35]`	—	✓	✓	—	✓	✓
`RCLK[36..47]`	✓	✓	—	—	—	—

Clock Control Block

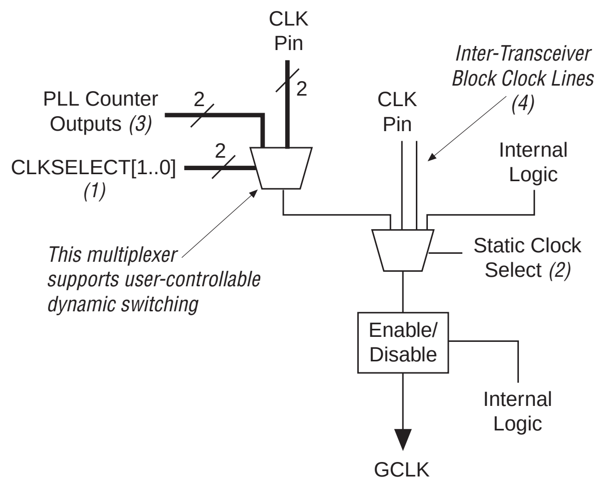
Every global and regional clock network has its own clock control block. The control block provides the following features:

- Clock source selection (dynamic selection for global clocks)
- Global clock multiplexing
- Clock power down (static or dynamic clock enable or disable)

Figure 5–7 and Figure 5–8 show the global clock and regional clock select blocks, respectively.

Select the clock source for the global clock select block either statically or dynamically. You can either statically select the clock source using a setting in the Quartus II software, or you can dynamically select the clock source using internal logic to drive the multiplexer select inputs. When selecting the clock source dynamically, you can either select two PLL outputs (such as `C0` or `C1`), or a combination of clock pins or PLL outputs.

Figure 5–7. Arria II GX Global Clock Control Block



Notes to Figure 5–7:

(1) These clock select signals can only be dynamically controlled through internal logic when the device is operating in user mode.

(2) These clock select signals can only be set through a configuration file (**.sof** or **.pof**) and cannot be dynamically controlled during user mode operation.

(3) The left side of the Arria II GX device only allows PLL counter outputs as the dynamic clock source selection to the global clock network.

(4) This is only available on the left side of the Arria II GX device.

The mapping between input clock pins, PLL counter outputs, and clock control block inputs is shown in Table 5–7.

Table 5–7. Mapping between Input Clock Pins, PLL Counter Outputs, and Clock Control Block Inputs

Clock Control Block Inputs	Description
`inclk[0]`, `inclk[1]` *(1)*	Can be fed by any of the 4 dedicated clock pins on the same side
`inclk[2]`	Can be fed by PLL counters C0 and C2 from the two corner PLLs on the same side of the Arria II GX device
`inclk[3]`	Can be fed by PLL counters C1 and C3 from the two corner PLLs on the same side of the Arria II GX device

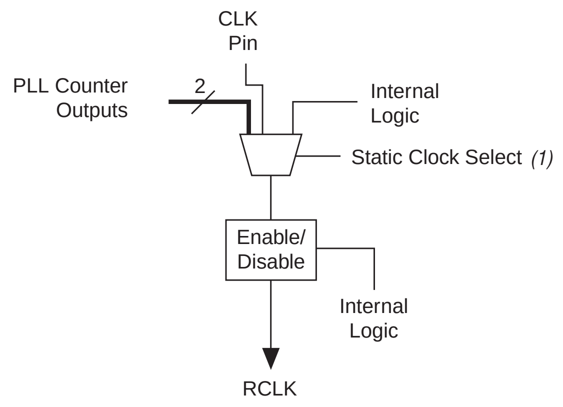
Note to Table 5–7:

(1) The left side of the Arria II GX device only allows PLL counter outputs as the dynamic clock source selection to the global clock network. Hence, `inclk[0]` can be fed by PLL counters C4 or C6 while `inclk[1]` can only be fed by PLL counter C5.

When combining the PLL outputs and clock pins in the same clock control block, you need to ensure that these clock sources are implemented on the same side of the device.

For all possible legal `inclk` sources for each global and regional clock network, refer to Table 5–2 on page 5–6 through Table 5–6 on page 5–8.

Figure 5–8. Regional Clock Control Block



Note to Figure 5–8:

(1) This clock select signal can only be statically controlled through a configuration file (**.sof** or **.pof**) and cannot be dynamically controlled during user mode operation.

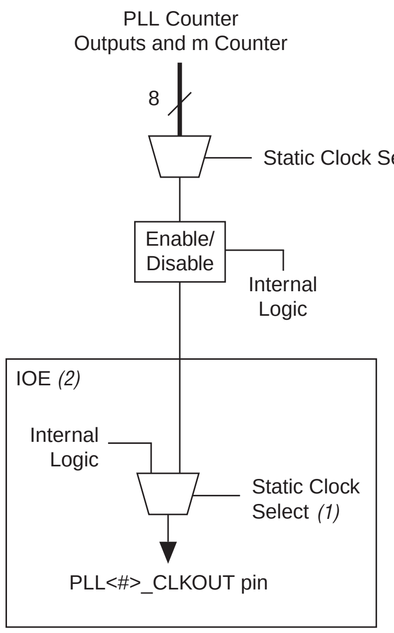
The clock source selection for the regional clock select block can only be controlled statically using configuration bit settings in the configuration file (**.sof** or **.pof**) generated by the Quartus II software.

The Arria II GX clock networks can be powered down by both static and dynamic approaches. When a clock net is powered down, all the logic fed by the clock net is in an off-state, thereby reducing the overall power consumption of the device. The unused global and regional clock networks are automatically powered down through configuration bit settings in the configuration file (**.sof** or **.pof**) generated by the Quartus II software. The dynamic clock enable or disable feature allows the internal logic to control power-up or power-down synchronously on GCLK and RCLK networks, including dual-regional clock regions. This function is independent of the PLL and is applied directly on the clock network, as shown in Figure 5–7 and Figure 5–8.

You can set the input clock sources and the `clkena` signals for the global and regional clock network multiplexers through the Quartus II software using the ALTCLKCTRL megafunction. You can also enable or disable the dedicated external clock output pins using the ALTCLKCTRL megafunction. Figure 5–9 shows the external PLL output clock control block.

When using the ALTCLKCTRL megafunction to implement dynamic clock source selection in the Arria II GX devices, the inputs from the clock pins, except for the left side of device, feed the `inclk[0..1]` ports of the multiplexer, while the PLL outputs feed the `inclk[2..3]` ports. You can choose from among these inputs using the `CLKSELECT[1..0]` signal. Refer to Table 5–7 for the connections between the PLL counter outputs to the clock control block on the left side of the Arria II GX device.

Figure 5–9. Arria II GX External PLL Output Clock Control Block



Notes to Figure 5–9:

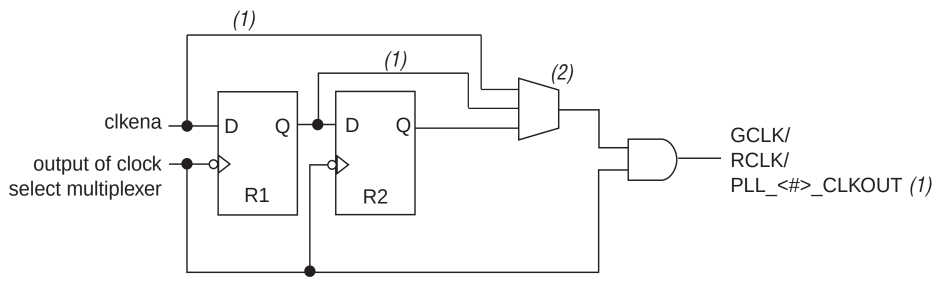
(1) This clock select signal can only be set through a configuration file (**.sof** or **.pof**) and cannot be dynamically controlled during user mode operation.

(2) The clock control block feeds to a multiplexer in the `PLL<#>_CLKOUT` pin's IOE. The `PLL<#>_CLKOUT` pin is a dual-purpose pin. Therefore, this multiplexer selects either an internal signal or the output of the clock control block.

Clock Enable Signals

Figure 5–10 shows how the clock enable/disable circuit of the clock control block is implemented in Arria II GX devices.

Figure 5–10. clkena Implementation



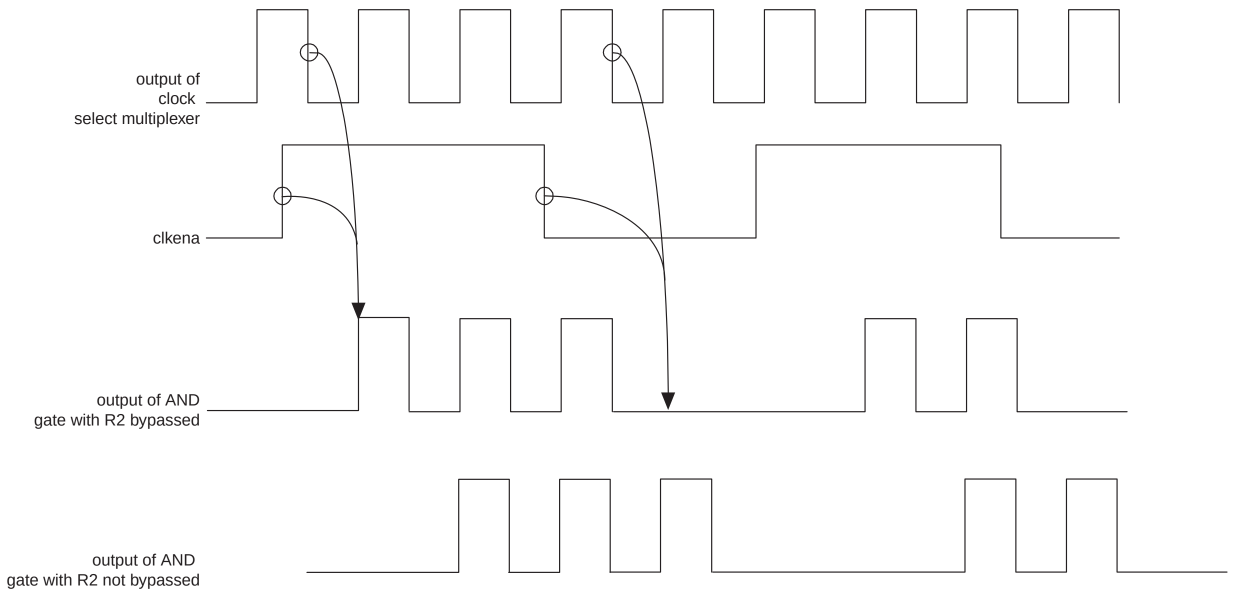
Notes to Figure 5–10:

(1) The R1 and R2 bypass paths are not available for PLL external clock outputs.
(2) The select line is statically controlled by a bit setting in the configuration file (**.sof** or **.pof**).

In Arria II GX devices, the `clkena` signals are supported at the clock network level instead of at the PLL output counter level. This allows you to gate off the clock even when a PLL is not being used. You can also use the `clkena` signals to control the dedicated external clocks from the PLLs. Figure 5–11 shows the waveform example for a clock output enable. The `clkena` signal is synchronous to the falling edge of the clock output.

Arria II GX devices also have an additional metastability register that aids in asynchronous enable/disable of the GCLK/RCLK networks. This register can be optionally bypassed in the Quartus II software.

Figure 5–11. clkena Signals


output of clock select multiplexer
clkena
output of AND gate with R2 bypassed
output of AND gate with R2 not bypassed

Note to Figure 5–11:

(1) You can use the `clkena` signals to enable or disable the global and regional networks or the `PLL<#>_CLKOUT` pins.

The PLL can remain locked independent of the `clkena` signals because the loop-related counters are not affected. This feature is useful for applications that require a low power or sleep mode. The `clkena` signal can also disable clock outputs if the system is not tolerant of frequency over-shoot during resynchronization.

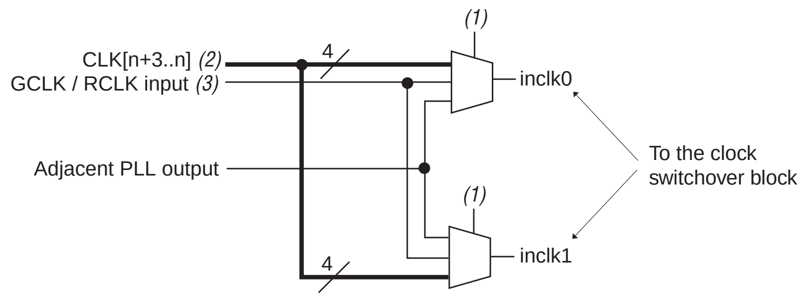
Clock Source Control for PLLs

The clock input to Arria II GX PLLs comes from clock input multiplexers. The clock multiplexer inputs come from dedicated clock input pins, PLLs through the GCLK and RCLK networks, or from dedicated connections between adjacent corner and center PLLs. The clock input sources to corner (`PLL_1`, `PLL_2`, `PLL_3`, `PLL_4`) and center PLLs (`PLL_5` and `PLL_6`) are shown in Figure 5–12.

The multiplexer select lines are set in the configuration file (SRAM object file [**.sof**] or programmer object file [**.pof**]) only. Once configured, this block cannot be changed without loading a new**.sof** or **.pof**. The Quartus II software automatically sets the multiplexer select signals depending on the clock sources selected in your design.

For more information on the clock control block and its supported features in the Quartus II software, refer to the *ALTCLKCTRL Megafunction User Guide*.

Figure 5–12. Clock Input Multiplexer Logic for Arria II GX PLLs



Notes to Figure 5–12:

(1) The input clock multiplexing is controlled through a configuration file (**.sof** or **.pof**) only and cannot be dynamically controlled in user mode operation.

(2) Dedicated clock input pins to PLLs. n=4 for `PLL_4`; n=4 or 8 for `PLL_3`; and n=8 or 12 for `PLL_2`; and n=12 for `PLL_1`.

(3) The global (GCLK) or regional (RCLK) clock input can be driven by an output from another PLL, a pin-driven global or regional clock, or through a clock control block provided the clock control block is fed by an output from another PLL or a pin-driven dedicated global or regional clock. An internally generated global signal or general purpose I/O pin cannot drive the PLL.

Cascading PLLs

The corner and center PLLs can be cascaded through the GCLK/RCLK networks. In addition, where two PLLs exist next to each other, there is a direct connection between them that does not require the GCLK/RCLK network. Using this path reduces clock jitter when cascading PLLs. The direct PLL cascading feature is available in `PLL_5` and `PLL_6` on the right side of the EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260 devices. Arria II GX devices allow cascading of `PLL_1` and `PLL_4` to transceiver PLLs (CMU PLLs and receiver CDRs).

For more information, refer to the "FPGA Fabric PLLs -Transceiver PLLs Cascading" section in the *Arria II GX Transceiver Clocking* chapter in volume 2 of the *Arria II GX Device Handbook*.

PLLs in Arria II GX Devices

Arria II GX devices offer up to 6 PLLs per device and 7 outputs per PLL that provide robust clock management and synthesis for device clock management, external system clock management, and high-speed I/O interfaces. The nomenclature for the PLLs follows their geographical location in the device floor plan. Refer to Figure 5–1 and Figure 5–2 for the location and number of PLLs in Arria II GX devices.

Depending on package, Arria II GX devices offer up to 8 transceiver TX PLLs per device that can be used by the FPGA fabric if they are not being used by the transceiver.

For more details on the number of general-purpose and transceiver TX PLLs in each device density, refer to the *Arria II GX Device Family Overview* chapter in volume 1 of the *Arria II GX Device Handbook*. For more information on the usage of the transceiver TX PLLs in the transceiver block, refer to the *Arria II GX Transceiver Clocking* chapter in volume 2 of the *Arria II GX Device Handbook*.

All Arria II GX PLLs have the same core analog structure and support features. Table 5–8 highlights the features of PLLs in Arria II GX devices.

Table 5–8. Arria II GX PLL Features (Part 1 of 2)

Feature	Arria II GX PLLs
C (output) counters	7
M, N, C counter sizes	1 to 512
Dedicated clock outputs *(1)*	1 single-ended or 1 differential pair 3 single-ended or 3 differential pairs *(2)*
Clock input pins	4 single-ended or 2 differential pin pairs
External feedback input pin	No
Spread-spectrum input clock tracking	Yes *(3)*
PLL cascading	Through GCLK and RCLK and dedicated path between adjacent PLLs. Cascading between the general-purpose PLL and transceiver PLL is supported in `PLL_1` and `PLL_4`.
Compensation modes	All except external feedback mode when using differential I/Os
PLL drives `DIFFCLK` and `LOADEN`	Yes
VCO output drives DPA clock	Yes
Phase shift resolution	Down to 96.125 ps *(4)*
Programmable duty cycle	Yes
Output counter cascading	Yes

Table 5–8. Arria II GX PLL Features (Part 2 of 2)

Feature	Arria II GX PLLs
Input clock switchover	Yes

Notes to Table 5–8:

(1) `PLL_5` and `PLL_6` do not have dedicated clock outputs.

(2) The same PLL clock output drives 3 single-ended or 3 differential I/O pairs. This is only supported in `PLL_1` and `PLL_3` of the EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260 devices.

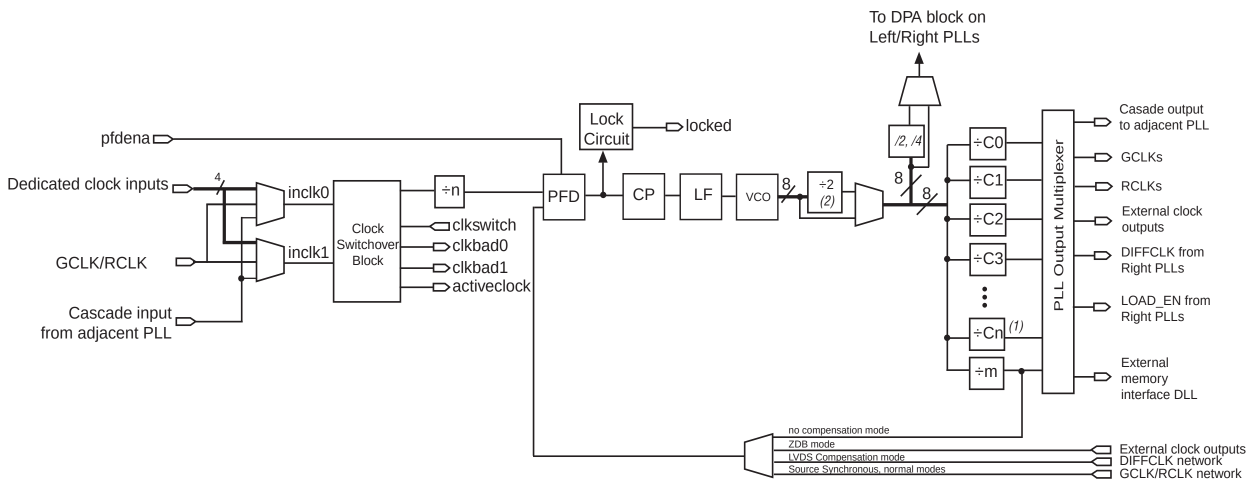
(3) This is applicable only if input clock jitter is in input jitter tolerance specifications.

(4) The smallest phase shift is determined by the voltage-controlled oscillator (VCO) period divided by eight. For degree increments, the Arria II GX device can shift all output frequencies in increments of at least 45°. Smaller degree increments are possible depending on the frequency and C counter value.

Arria II GX PLL Hardware Overview

Figure 5–13 shows a simplified block diagram of the major components of the Arria II GX PLL.

Figure 5–13. Arria II GX PLL Block Diagram



Notes to Figure 5–13:

(1) There are 7 PLL output counters in Arria II GX devices.

(2) This is the VCO post-scale counter K.

The global (GCLK) or regional (RCLK) clock input can be driven by an output from another PLL, a pin-driven global or regional clock, or through a clock control block, provided the clock control block is fed by an output from another PLL or a pin driven dedicated global or regional clock. An internally generated global signal or general purpose I/O pin cannot drive the PLL.

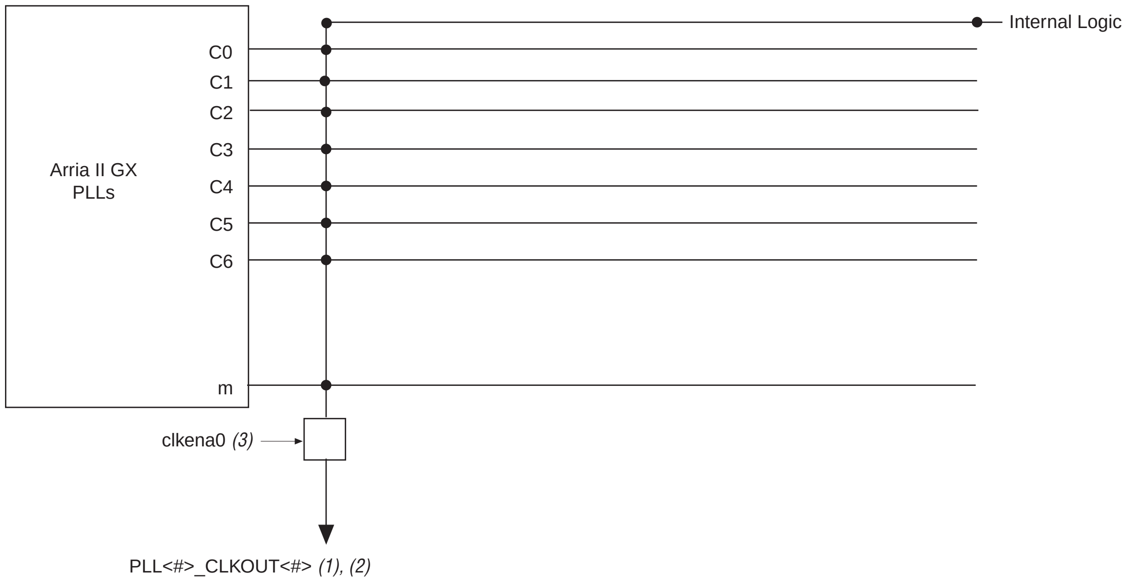
PLL Clock I/O Pins

Each PLL supports one of the following clock I/O pin configurations:

- 1 single-ended I/O or 1 differential I/O pair.
- 3 single-ended I/O pairs or 3 differential I/O pairs (this is only supported in `PLL_1` and `PLL_3` of the EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260 devices).

Figure 5–14 shows the clock I/O pins associated with Arria II GX PLLs.

Figure 5–14. External Clock Outputs for Arria II GX PLLs



Notes to Figure 5–14:

(1) These clock output pins can be fed by any one of the C[6..0], m counters.

(2) The `PLL<#>_CLKOUT<#>p` and `PLL<#>_CLKOUT<#>n` pins can be either single-ended or differential clock outputs. The Arria II GX PLL only routes single-ended I/Os to `PLL<#>CLKOUT<#>p` pins, while `PLL<#>_CLKOUT<#>n` pins can be used as user I/Os.

(3) These external clock enable signals are available only when using the ALTCLKCTRL megafunction.

Any of the output counters (`C[6..0]` or the `M` counter can feed the dedicated external clock outputs, as shown in Figure 5–15. Therefore, one counter or frequency can drive all output pins available from a given PLL.

Each pin of a single-ended output pair can either be in-phase or 180-degrees out-of-phase. The Quartus II software places the NOT gate in your 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, as well as LVDS, LVPECL, differential HSTL, and differential SSTL.

To determine which I/O standards are supported by the PLL clock input and output pins, refer to the *I/O Features in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

Arria II GX PLLs can also drive out to any regular I/O pin through the global or regional clock network. You can also use the external clock output pins as user I/O pins if external PLL clocking is not needed.

PLL Control Signals

You can use the three signals, `pfdena`, `areset`, and `locked`, to observe and control the PLL operation and resynchronization.

pfdena

Use the `pfdena` signal to maintain the most recent locked frequency to enable your system to store its current settings before shutting down. The `pfdena` signal controls the PFD output with a programmable gate. If you disable the PFD, the VCO operates at its most recent set value of control voltage and frequency with some long-term drift to a lower frequency.

areset

The `areset` signal is the reset or resynchronization input for each PLL. The device input pins or internal logic can drive these input signals. When `areset` is driven high, the PLL counters reset, clearing the PLL output and placing the PLL out-of-lock. The VCO is then set back to its nominal setting. When `areset` is driven low again, the PLL resynchronizes to its input as it re-locks.

You should include the `areset` signal in designs if any of the following conditions are true:

- PLL reconfiguration or clock switchover is enabled in your design.
- Phase relationships between the PLL input and output clocks need to be maintained after a loss-of-lock condition

If the input clock to the PLL is not toggling or is unstable upon power up, assert the `areset` signal after the input clock is stable and in specifications.

locked

The locked output of the PLL indicates that the PLL has locked onto the reference clock and the PLL clock outputs are operating at the desired phase and frequency set in the Quartus II software MegaWizard™ Plug-In Manager.

Altera recommends that you use the `areset` and `locked` signals in your designs to control and observe the status of your PLL.

For more information about the PLL control signals, refer to the *ALTPLL Megafunction User Guide*.

Clock Feedback Modes

Arria II GX PLLs support up to five different clock feedback modes. Each mode allows clock multiplication and division, phase shifting, and programmable duty cycle. Table 5–9 shows the clock feedback modes supported by Arria II GX PLLs.

Table 5–9. Clock Feedback Mode Availability (Part 1 of 2)

Clock Feedback Mode	Availability in Arria II GX Devices
Source-synchronous mode	Yes
No-compensation mode	Yes
Normal mode	Yes
Zero-delay buffer (ZDB) mode *(1)*	Yes

Table 5–9. Clock Feedback Mode Availability (Part 2 of 2)

Clock Feedback Mode	Availability in Arria II GX Devices
LVDS compensation	Yes *(2)*

Notes to Table 5–9:

(1) The ZDB mode is using 8 ns delay for compensation in Arria II GX devices.

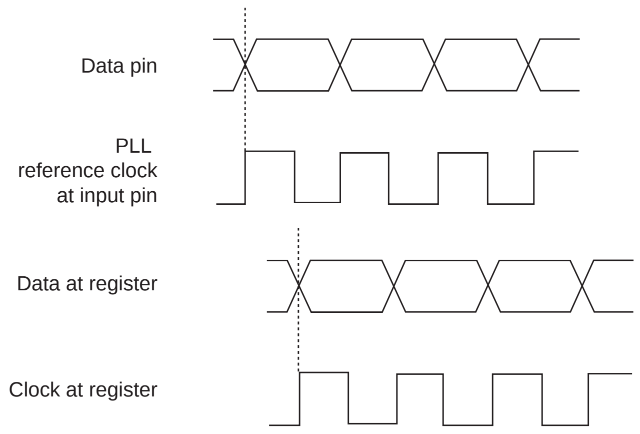
(2) LVDS compensation mode is only supported on the PLL_2, PLL_3, PLL_5, and PLL_6.

☞ The input and output delays are fully compensated by a PLL only when using the dedicated clock input pins associated with a given PLL as the clock sources. For example, when using `PLL_1` in normal mode, the clock delays from the input pin to the PLL clock output-to-destination register are fully compensated, provided the clock input pin is one of the following four pins: `CLK12`, `CLK13`, `CLK14`, or `CLK15`. When an RCLK or GCLK network drives the PLL, the input and output delays may not be fully compensated in the Quartus II software. Another example is when `PLL_1` is configured in zero delay buffer mode and the PLL input is driven by a dedicated clock input pin, a fully compensated clock path results in zero delay between the clock input and one of the output clocks from the PLL. If the PLL input is instead fed by a non-dedicated input (using the GCLK network), then the output clock may not be perfectly aligned with the input clock.

Source Synchronous Mode

If data and clock arrive at the same time on the input pins, the same phase relationship is maintained at the clock and data ports of any IOE input register. Figure 5–15 shows an example waveform of the clock and data in this mode. This mode is recommended for source-synchronous data transfers. Data and clock signals at the IOE experience similar buffer delays as long as you use the same I/O standard.

Figure 5–15. Phase Relationship Between Clock and Data in Source-Synchronous Mode



The source-synchronous mode compensates for the delay of the clock network used plus any difference in the delay between these two paths:

- Data pin to IOE register input
- Clock input pin to the PLL PFD input

You can use the **PLL Compensation** assignment in the Quartus II software Assignment Editor to select which input pins will be used as the PLL compensation targets. You can include your entire data bus, provided the input registers are clocked by the same output of a source-synchronous compensated PLL. In order for the clock delay to be properly compensated, all input pins need to be on the same side of the device. The PLL will compensate for the input pin with the longest pad-to-register delay among all input pins in the compensated bus.

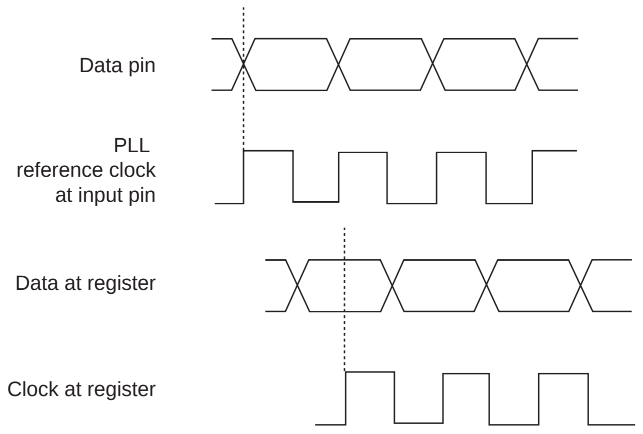
If you do not assign the **PLL Compensation** assignment, the Quartus II software will automatically select all pins driven by the compensated output of the PLL as the compensation target.

Source-Synchronous Mode for LVDS Compensation

The goal of this mode is to maintain the same data and clock timing relationship seen at the pins at the internal SERDES capture register, except that the clock is inverted (180° phase shift), as shown in Figure 5–16. Thus, this mode ideally compensates for the delay of the LVDS clock network plus any difference in delay between these two paths:

- Data pin-to-SERDES capture register
- Clock input pin-to-SERDES capture register. In addition, the output counter needs to provide the 180° phase shift.

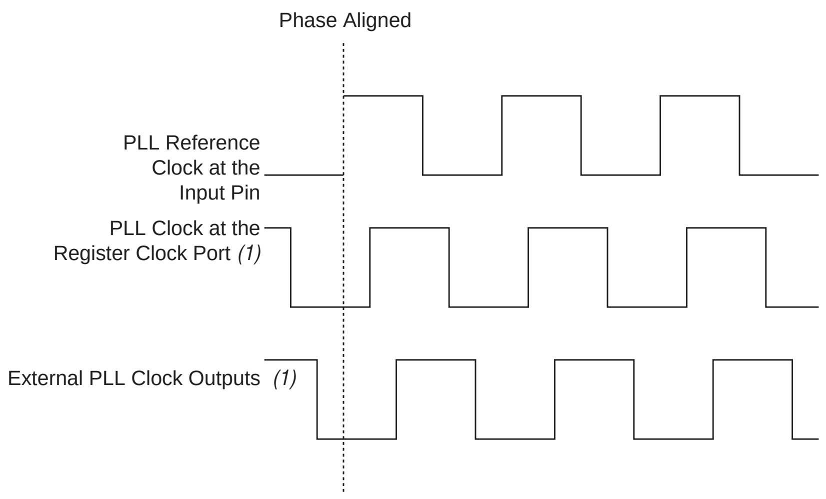
Figure 5–16. Source-Synchronous Mode for LVDS Compensation



No-Compensation Mode

In the no-compensation mode, the PLL does not compensate for any clock networks. This mode provides better jitter performance because the clock feedback into the PFD passes through less circuitry. Both the PLL internal- and external-clock outputs are phase-shifted with respect to the PLL clock input. Figure 5–17 shows an example waveform of the PLL clocks' phase relationship in this mode.

Figure 5–17. Phase Relationship Between PLL Clocks in No Compensation Mode


Phase Aligned
PLL Reference Clock at the Input Pin
PLL Clock at the Register Clock Port (1)
External PLL Clock Outputs (1)

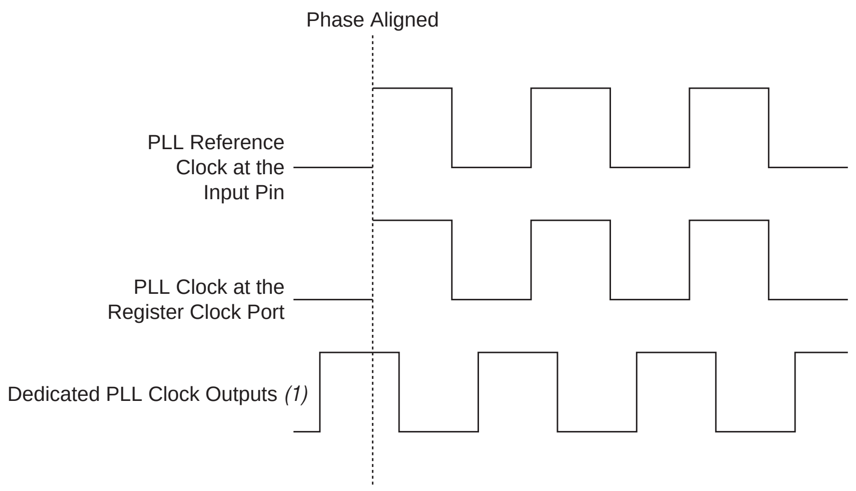
Note to Figure 5–17:

(1) The PLL clock outputs will lag the PLL input clocks depending on routine delays.

Normal Mode

An internal clock in normal mode 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. The Quartus II software timing analyzer reports any phase difference between the two. In normal mode, the delay introduced by the GCLK or RCLK network is fully compensated. Figure 5–18 shows an example waveform of the PLL clocks' phase relationship in this mode.

Figure 5–18. Phase Relationship Between PLL Clocks in Normal Mode


Phase Aligned
PLL Reference Clock at the Input Pin
PLL Clock at the Register Clock Port
Dedicated PLL Clock Outputs (1)

Note to Figure 5–18:

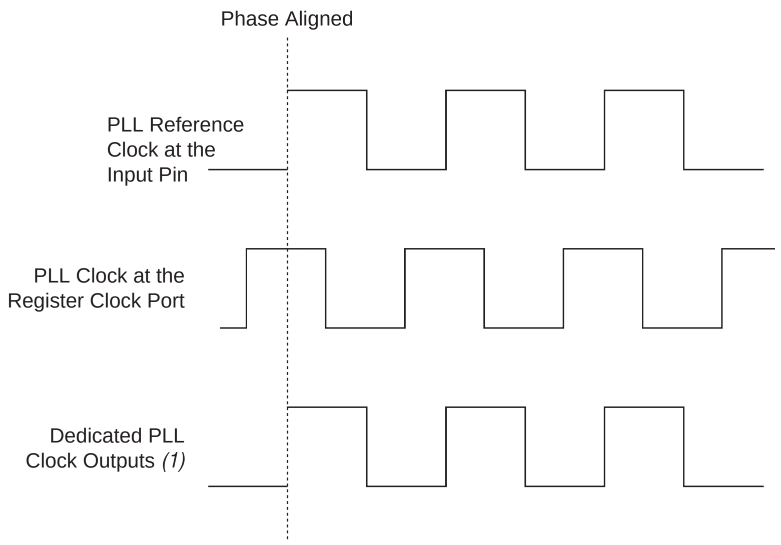
(1) The external clock output can lead or lag the PLL internal clock signals.

Zero-Delay Buffer Mode

In zero-delay buffer (ZDB) mode, the external clock output pin is phase-aligned with the clock input pin for zero delay through the device. When using this mode, you must use the same I/O standard on the input clocks and output clocks to guarantee clock alignment at the input and output pins. This mode is supported on all Arria II GX PLLs.

Figure 5–19 shows an example waveform of the PLL clocks' phase relationship in ZDB mode.

Figure 5–19. Phase Relationship Between PLL Clocks in Zero Delay Buffer Mode


Phase Aligned
PLL Reference Clock at the Input Pin
PLL Clock at the Register Clock Port
Dedicated PLL Clock Outputs *(1)*

Note to Figure 5–19:

(1) The internal PLL clock output can lead or lag the external PLL clock outputs.

Clock Multiplication and Division

Each Arria II GX PLL provides clock synthesis for PLL output ports using $m/(n^*$ post-scale counter) scaling factors. The input clock is divided by a pre-scale factor, n, and is then multiplied by the m feedback factor. The control loop drives the VCO to match f_{in} (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. For example, if output frequencies required from one PLL are 33 and 66 MHz, then the Quartus II software sets the VCO to 660 MHz (the least common multiple of 33 and 66 MHz in the VCO range). Then the post-scale counters scale down the VCO frequency for each output port.

The VCO frequency reported by the Quartus II software is the value after the post-scale counter divider, k.

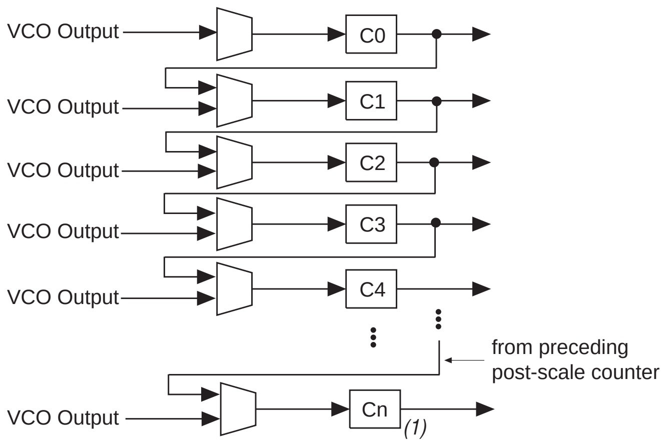
Each PLL has one pre-scale counter, `n`, and one multiply counter, `m`, with a range of 1 to 512 for both `m` and `n`. The `n` counter does not use duty-cycle control because the only purpose of this counter is to calculate frequency division. There are seven generic post-scale counters in each PLL that can feed GCLKs, RCLKs, or external clock outputs. These post-scale counters range from 1 to 512 with a 50% duty cycle setting. The high- and low-count values for each counter range from 1 to 256. The sum of the high- and low-count values chosen for a design selects the divide value for a given counter.

The Quartus II software automatically chooses the appropriate scaling factors according to the input frequency, multiplication, and division values entered into the ALTPLL megafunction.

Post-Scale Counter Cascading

The Arria II GX PLLs support post-scale counter cascading to create counters larger than 512. This is automatically implemented in the Quartus II software by feeding the output of one `C` counter into the input of the next `C` counter as shown in Figure 5–20.

Figure 5–20. Counter Cascading



Note to Figure 5–20:

(1) n = 6

When cascading post-scale counters to implement a larger division of the high-frequency VCO clock, the cascaded counters behave as one counter with the product of the individual counter settings. For example, if C0 = 40 and C1 = 20, then the cascaded value is C0*C1 = 800.

☞ Post-scale counter cascading is set in the configuration file. It cannot be done using PLL reconfiguration.

Programmable Duty Cycle

The programmable duty cycle allows PLLs to generate clock outputs with a variable duty cycle. This feature is supported on the PLL post-scale counters. 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 post-scale counter value determines the precision of the duty cycle. The precision is defined by 50% divided by the post-scale counter value. For example, if the `C0` counter is 10, then steps of 5% are possible for duty-cycle choices between 5% to 90%.

Combining the programmable duty cycle with programmable phase shift allows the generation of precise non-overlapping clocks.

Programmable Phase-Shift

Phase shift is used to implement a robust solution for clock delays in Arria II GX devices. Phase shift is implemented with a combination of the VCO phase output and the counter starting time. The VCO phase output and counter starting time is the most accurate method of inserting delays, because it is purely based on counter settings, which are independent of process, voltage, and temperature.

You can phase-shift the output clocks from the Arria II GX PLLs in either of these two resolutions:

- Fine resolution using VCO phase taps
- Coarse resolution using counter starting time

Fine-resolution phase shifts are implemented by allowing any of the output counters (`C[n..0]`) or the `m` counter to use any of the eight phases of the VCO as the reference clock. This allows you to adjust the delay time with a fine resolution. The minimum delay time that you can insert using this method is defined by:

Equation 5–1. Fine-Resolution Phase Shifts

$$\Phi_{fine} = \frac{1}{8}T_{VCO} = \frac{1}{8f_{VCO}} = \frac{N}{8Mf_{REF}}$$

where f_{REF} is the input reference clock frequency.

For example, if f_{REF} is 100 MHz, n is 1, and m is 8, then f_{VCO} is 800 MHz and Φ_{fine} equals 156.25 ps. This phase shift is defined by the PLL operating frequency, which is governed by the reference clock frequency and the counter settings.

Coarse-resolution phase shifts are implemented by delaying the start of the counters for a predetermined number of counter clocks. You can express coarse phase shift as:

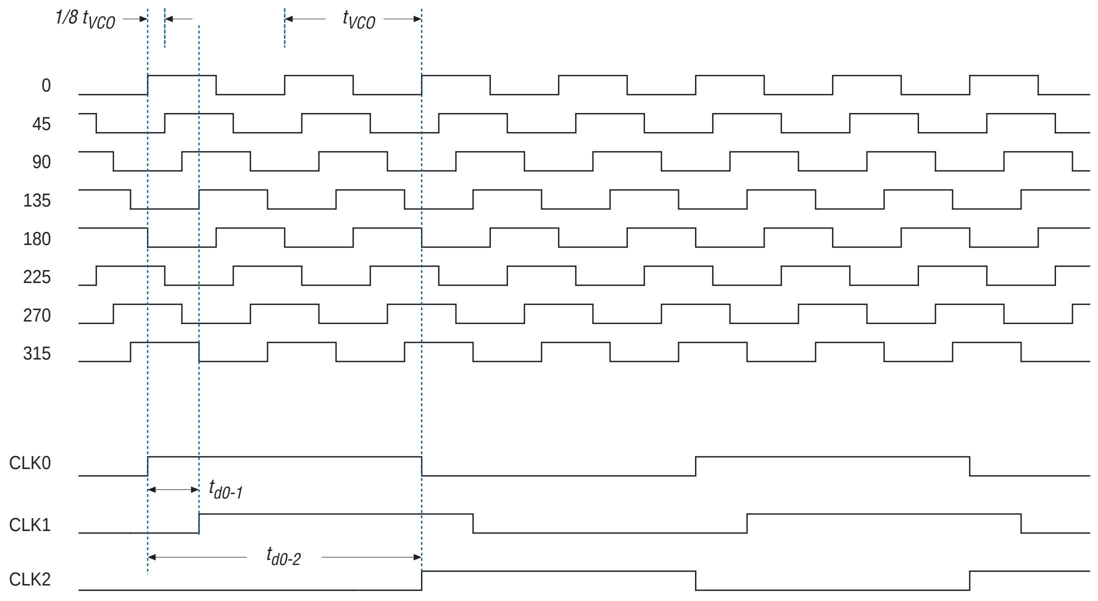
Equation 5–2. Coarse-Resolution Phase Shifts

$$\Phi_{coarse} = \frac{C-1}{f_{VCO}} = \frac{(C-1)N}{Mf_{REF}}$$

where C is the count value set for the counter delay time, (this is the initial setting in the PLL usage section of the compilation report in the Quartus II software). If the initial value is 1, C – 1 = 0° phase shift.

Figure 5–21 shows an example of phase-shift insertion with the fine resolution using the VCO phase taps method. The eight phases from the VCO are shown and labeled for reference. For this example, `CLK0` is based off the `0phase` from the VCO and has the C value for the counter set to one. The `CLK1` signal is divided by four, two VCO clocks for high time and two VCO clocks for low time. `CLK1` is based off the 135× phase tap from the VCO and also has the C value for the counter set to one. The `CLK1` signal is also divided by 4. In this case, the two clocks are offset by 3 Φ_{fine}. `CLK2` is based off the `0phase` from the VCO but has the C value for the counter set to three. This arrangement creates a delay of 2 Φ_{COARSE} (two complete VCO periods).

Figure 5–21. Delay Insertion Using VCO Phase Output and Counter Delay Time



You can use the coarse- and fine-phase shifts to implement clock delays in Arria II GX devices. The ALTPLL megafunction allows you to enter the desired VCO phase taps and initial counter value settings through the Quartus II Megawizard Plug-In Manager.

Arria II GX devices support dynamic phase-shifting of VCO phase taps only. The phase shift is reconfigurable any number of times, and each phase shift takes about one `scanclk` cycle, allowing you to implement large phase shifts quickly.

Programmable Bandwidth

PLL bandwidth is the measure of the PLL's ability to track the input clock and its associated jitter. Arria II GX PLLs provide advanced control of the PLL bandwidth using the PLL loop's programmable characteristics, including loop filter and charge pump. The closed-loop gain 3-dB frequency in the PLL determines the PLL bandwidth. The bandwidth is approximately the unity gain point for open loop PLL response.

Spread-Spectrum Tracking

Arria II 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 PLL. Arria II GX PLLs can track a spread-spectrum input clock as long as the input jitter is in the PLL input jitter tolerance specification. Arria II GX devices cannot internally generate spread-spectrum clocks.

Clock Switchover

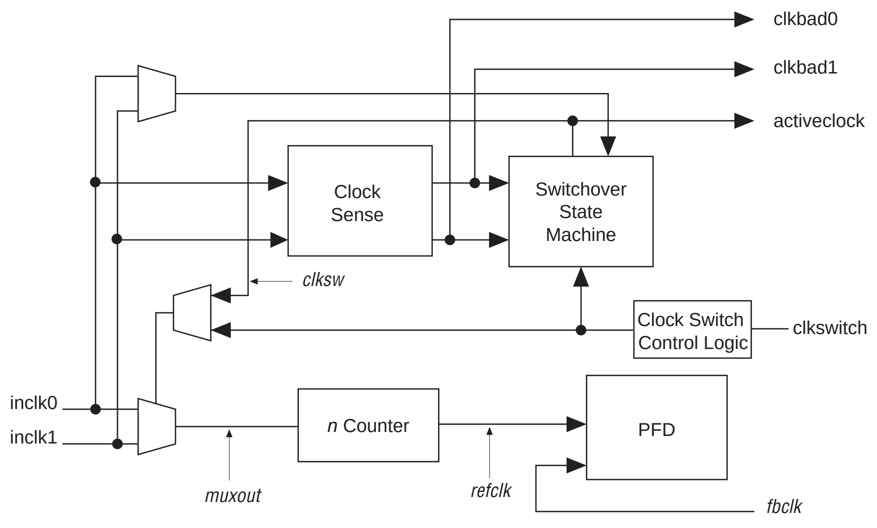
The clock switchover feature allows the PLL to switch between two reference input clocks. Use this feature for clock redundancy or for a dual-clock domain application such as in a system that turns on the redundant clock if the previous clock stops running. Your design can perform clock switchover automatically, when the clock is no longer toggling or based on a user control signal, `clkswitch`.

The following clock switchover modes are supported in Arria II GX PLLs:

- Automatic switchover: The clock sense circuit monitors the current reference clock and if it stops toggling, automatically switches to the other clock `inclk0` or `inclk1`.
- Manual clock switchover: Clock switchover is controlled using the `clkswitch` signal in this mode. When the `clkswitch` signal goes from logic low to logic high, and stays high for at least three clock cycles, the reference clock to the PLL is switched from `inclk0` to `inclk1`, or vice-versa.
- Automatic switchover with manual override: This mode combines modes 1 and 2. When the `clkswitch` signal goes high, it overrides automatic clock switchover mode.

Arria II GX PLLs support a fully configurable clock switchover capability. Figure 5–22 shows the block diagram of the switchover circuit built into the PLL. When the current reference clock is not present, the clock sense block automatically switches to the backup clock for PLL reference. The clock switchover circuit also sends out three status signals—`clkbad[0]`, `clkbad[1]`, and `activeclock`—from the PLL to implement a custom switchover circuit in the logic array. You can select a clock source as the backup clock by connecting it to the `inclk1` port of the PLL in your design.

Figure 5–22. Automatic Clock Switchover Circuit Block Diagram



Automatic Clock Switchover

Use the switchover circuitry to automatically switch between `inclk0` and `inclk1` when the current reference clock to the PLL stops toggling. For example, in applications that require a redundant clock with the same frequency as the reference clock, the switchover state machine generates a signal (`clksw`) that controls the multiplexer select input as shown in Figure 5–22. In this case, `inclk1` becomes the reference clock for the PLL. When using the automatic switchover mode, you can switch back and forth between the `inclk0` and `inclk1` clocks any number of times, when one of the two clocks fails and the other clock is available.

When using the automatic clock switchover mode, the following requirements need to be satisfied:

- Both clock inputs need to be running.
- The period of the two clock inputs can differ by no more than 100% (2×).

If the current clock input stops toggling while the other clock is also not toggling, switchover will not be initiated and the `clkbad[0:1]` signals will not be valid. Also, if both clock inputs are not the same frequency, but their period difference is in 100%, the clock sense block will detect when a clock stops toggling, but the PLL may lose lock after the switchover is completed and need time to relock.

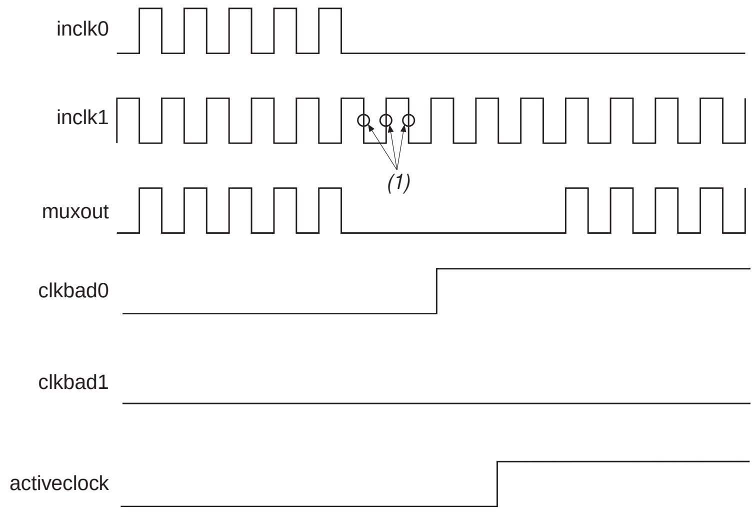
Altera recommends resetting the PLL using the `areset` signal to maintain the phase relationships between the PLL input and output clocks when using clock switchover.

When using automatic switchover mode, the `clkbad[0]` and `clkbad[1]` signals indicate the status of the two clock inputs. When they are asserted, the clock sense block has detected that the corresponding clock input has stopped toggling. These two signals are not valid if the frequency difference between `inclk0` and `inclk1` is greater than 20%.

The `activeclock` signal indicates which of the two clock inputs (`inclk0` or `inclk1`) is being selected as the reference clock to the PLL. When the frequency difference between the two clock inputs is more than 20%, the `activeclock` signal is the only valid status signal.

Figure 5–23 shows an example waveform of the switchover feature when using the automatic switchover mode. In this example, the `inclk0` signal is stuck low. After the `inclk0` signal is stuck at low for approximately two clock cycles, the clock sense circuitry drives the `clkbad[0]` signal high. Also, because the reference clock signal is not toggling, the switchover state machine controls the multiplexer through the `clksw` signal to switch to the backup clock, `inclk1`.

Figure 5–23. Automatic Switchover Upon Loss of Clock Detection



Note to Figure 5–23:

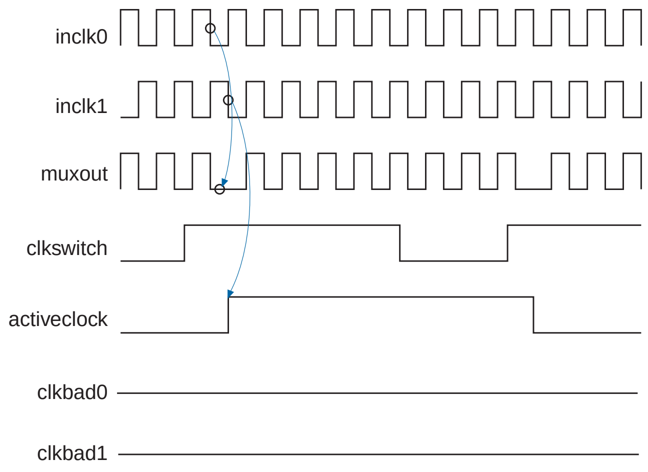
(1) Switchover is enabled on the falling edge of `inclk0` or `inclk1`, depending on which clock is available. In this figure, switchover is enabled on the falling edge of `inclk1`.

Manual Override

In the automatic switchover with manual override mode, you can use the `clkswitch` input for user- or system-controlled switch conditions. You can use this mode for same-frequency switchover or to switch between inputs of different frequencies. For example, if `inclk0` is 66 MHz and `inclk1` is 200 MHz, you must control the switchover using `clkswitch` because the automatic clock-sense circuitry cannot monitor clock input (`inclk0`, `inclk1`) frequencies with a frequency difference of more than 100% (2×). This feature is useful when the clock sources originate from multiple cards on the backplane, requiring a system-controlled switchover between the frequencies of operation. You should choose the backup clock frequency and set the `m`, `n`, `c`, and `k` counters accordingly so the VCO operates in the recommended operating frequency range of 600 to 1,300 MHz. The ALTPLL Megawizard Plug-In Manager notifies you if a given combination of `inclk0 and inclk1` frequencies cannot meet this requirement.

Figure 5–24 shows an example of a waveform illustrating the switchover feature when controlled by `clkswitch`. In this case, both clock sources are functional and **`inclk0`** is selected as the reference clock. The `clkswitch` signal goes high, which starts the switchover sequence. On the falling edge of `inclk0`, the counter's reference clock, `muxout`, is gated off to prevent any clock glitching. On the falling edge of `inclk1`, the reference clock multiplexer switches from `inclk0` to `inclk1` as the PLL reference, and the `activeclock` signal changes to indicate which clock is currently feeding the PLL.

Figure 5–24. Clock Switchover Using the clkswitch (Manual) Control *(Note 1)*



Note to Figure 5–24:

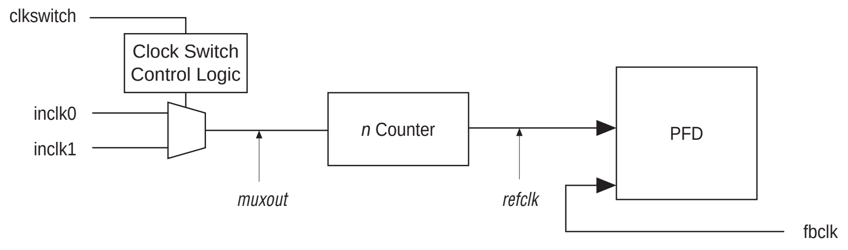
(1) Both `inckl0` and `inclk1` must be running when the `clkswitch` signal goes high to start a manual clock switchover event.

In this mode, the `activeclock` signal mirrors the `clkswitch` signal. As both clocks are still functional during the manual switch, neither `clkbad` signal goes high. Because the switchover circuit is positive-edge sensitive, the falling edge of the `clkswitch` signal does not cause the circuit to switch back from `inclk1` to `inclk0`. When the `clkswitch` signal goes high again, the process repeats. The `clkswitch` signal and automatic switch only work if the clock being switched to is available. If the clock is not available, the state machine waits until the clock is available.

Manual Clock Switchover

In manual clock switchover mode, the `clkswitch` signal controls whether `inclk0` or `inclk1` is selected as the input clock to the PLL. By default, `inclk0` is selected. A low-to-high transition on `clkswitch` and `clkswitch` being held high for at least three `inclk` cycles begins a clock switchover event. You must bring `clkswitch` back low again to perform another switchover event in the future. If you do not require another switchover event in the future, you can leave `clkswitch` in a logic high state after the initial switch. Pulsing `clkswitch` high for at least three `inclk` cycles performs another switchover event. If `inclk0` and `inclk1` are different frequencies and are always running, the `clkswitch` minimum high time must be greater than or equal to three of the slower frequency `inclk0/inclk1` cycles. Figure 5–25 shows the block diagram of the manual switchover circuit.

Figure 5–25. Manual Clock Switchover Circuitry in Arria II GX PLLs



For more information about PLL software support in the Quartus II software, refer to the *ALTPLL Megafunction User Guide*.

Guidelines

Use the following guidelines when implementing clock switchover in Arria II GX PLLs.

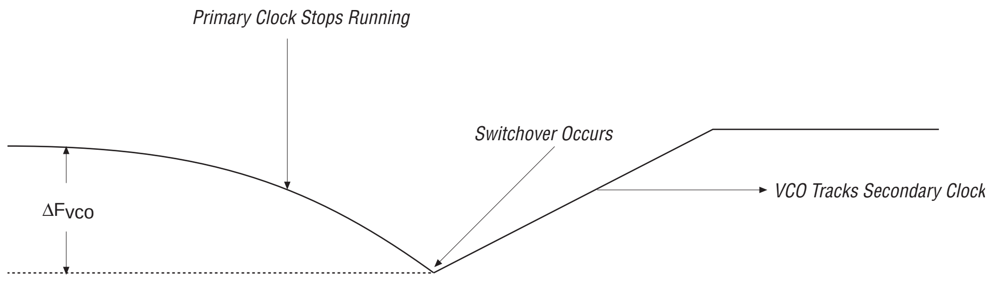
- Automatic clock switchover requires that the `inclk0` and `inclk1` frequencies be in 100% (2×) of each other. Failing to meet this requirement causes the `clkbad[0]` and `clkbad[1]` signals to not function properly.
- When using manual clock switchover, the difference between `inclk0` and `inclk1` can be more than 100% (2×). However, differences in frequency and/or phase of the two clock sources will likely cause the PLL to lose lock. Resetting the PLL ensures that the correct phase relationships are maintained between the input and output clocks.

  Both `inclk0` and `inclk1` must be running when the `clkswitch` signal goes high to start the manual clock switchover event. Failing to meet this requirement causes the clock switchover to not function properly.

- Applications that require a clock switchover feature and a small frequency drift should use a low-bandwidth PLL. The low bandwidth PLL reacts more slowly than a high-bandwidth PLL to reference input clock changes. When the switchover happens, a low bandwidth PLL propagates the stopping of the clock to the output more slowly than a high-bandwidth PLL. However, be aware that the low-bandwidth PLL also increases lock time.
- After a switchover occurs, there may be a finite resynchronization period for the PLL to lock onto a new clock. The exact amount of time it takes for the PLL to re-lock depends on the PLL configuration.
- If the phase relationship between the input clock to the PLL and the output clock from the PLL is important in your design, assert `areset` for at least 10 ns after performing a clock switchover.
- To prevent clock glitches from propagating through your design during PLL resynchronization or after `areset` is applied, use the clock enable feature of the clock control block to disable the clock network. Wait for the locked signal to assert and be stable before re-enabling the output clocks from the PLL at the clock control block.

- Figure 5–26 shows how the VCO frequency gradually decreases when the current clock is lost and then increases as the VCO locks on to the backup clock.

Figure 5–26. VCO Switchover Operating Frequency



- Disable the system during clock switchover if it is not tolerant of frequency variations during the PLL resynchronization period. You can use the `clkbad[0]` and `clkbad[1]` status signals to turn off the PFD (`PFDENA` = `0`) so the VCO maintains its most recent frequency. You can also use the state machine to switch over to the secondary clock. When the PFD is re-enabled, output clock-enable signals (`clkena`) can disable clock outputs during the switchover and resynchronization period. Once the lock indication is stable, the system can re-enable the output clocks.

PLL Reconfiguration

Phase-locked loops (PLLs) use several divide counters and different voltage-controlled oscillator (VCO) phase taps to perform frequency synthesis and phase shifts. In Arria II GX PLLs, you can reconfigure both the counter settings and phase-shift the PLL output clock in real time. You can also change the charge pump and loop-filter components, which dynamically affect the PLL bandwidth. You can use these PLL components to update the output-clock frequency and the PLL bandwidth and to phase-shift in real time, without reconfiguring the entire Arria II GX device.

The ability to reconfigure the PLL in real time is useful in applications that operate at multiple frequencies. It is also useful in prototyping environments, allowing you to sweep PLL output frequencies and adjust the output-clock phase dynamically. For instance, a system generating test patterns is required to generate and transmit patterns at 75 or 150 MHz, depending on the requirements of the device under test. Reconfiguring the PLL components in real time allows you to switch between two such output frequencies in a few microseconds. You can also use this feature to adjust clock-to-out (t_{CO}) delays in real time by changing the PLL output clock phase shift. This approach eliminates the need to regenerate a configuration file with the new PLL settings.

PLL Reconfiguration Hardware Implementation

The following PLL components are reconfigurable in real time:

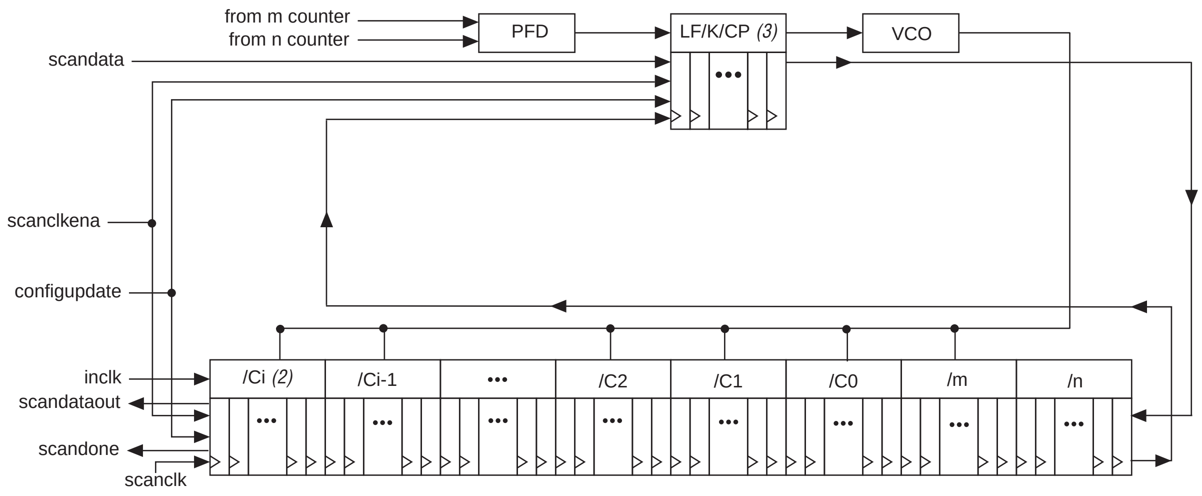
- Pre-scale counter (`n`)
- Feedback counter (`m`)
- Post-scale output counters (`C0` - `C6`)

- Post VCO divider (`K`)
- Dynamically adjust the charge-pump current (`Icp`) and loop-filter components (`R`, `C`) to facilitate reconfiguration of the PLL bandwidth

Figure 5–27 shows how PLL counter settings can be dynamically adjusted by shifting their new settings into a serial shift-register chain or scan chain. Serial data is input to the scan chain using the `scandataport` and shift registers are clocked by `scanclk`. The maximum `scanclk` frequency is 100 MHz. Serial data is shifted through the scan chain as long as the `scanclkena` signal stays asserted. After the last bit of data is clocked, asserting the `configupdate` signal for at least one `scanclk` clock cycle causes the PLL configuration bits to be synchronously updated with the data in the scan registers.

For more information about the PLL reconfiguration port signals, refer to the *ALTPLL_RECONFIG Megafunction User Guide*.

Figure 5–27. PLL Reconfiguration Scan Chain



Notes to Figure 5–27:

(1) The Arria II GX PLLs support `C0` - `C6` counters.

(2) i = 6

(3) This figure shows the corresponding scan register for the K counter in between the scan registers for the charge pump and loop filter. The K counter is physically located after the VCO.

The counter settings are updated synchronously to the clock frequency of the individual counters. Therefore, all counters are not updated simultaneously.

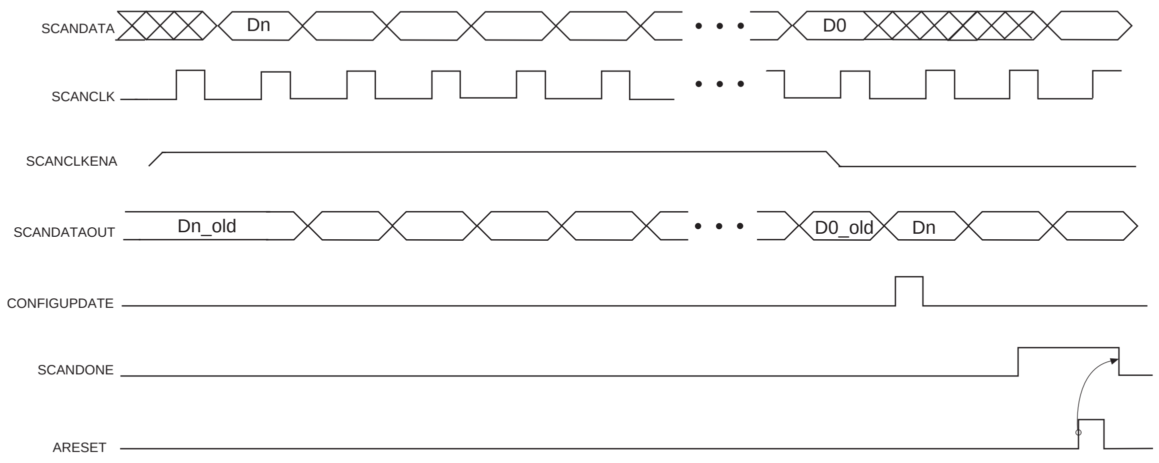
The procedure to reconfigure the PLL counters is shown below:

1. The `scanclkena` signal is asserted at least one `scanclk` cycle prior to shifting in the first bit of `scandata` (`Dn`).
2. Serial data (`scandata`) is shifted into the scan chain on the 2nd rising edge of `scanclk`.
3. After all 180 bits are scanned into the scan chain, the `scanclkena` signal is de-asserted to prevent inadvertent shifting of bits in the scan chain.

4. The `configupdate` signal is asserted for one `scanclk` cycle to update the PLL counters with the contents of the scan chain.
5. The `scandone` signal goes high indicating the PLL is being reconfigured. A falling edge indicates the PLL counters are updated with new settings.
6. Reset the PLL using the `areset` signal if you make any changes to the `M` or `N` counters or the `Icp`, `R`, or `C` settings.
7. Steps 1-5 can be repeated to reconfigure the PLL any number of times.

Figure 5–28 shows a functional simulation of the PLL reconfiguration feature.

Figure 5–28. PLL Reconfiguration Waveform



When you reconfigure the counter clock frequency, you cannot reconfigure the corresponding counter phase shift settings using the same interface. Instead, reconfigure the phase shifts in real time using the dynamic phase shift reconfiguration interface. If you reconfigure the counter frequency, but wish to keep the same non-zero phase shift setting (for example, 90°) on the clock output, you must reconfigure the phase shift immediately after reconfiguring the counter clock frequency.

Post-Scale Counters (C0 to C6)

The multiply or divide values and duty cycle of post-scale counters can be reconfigured in real time. Each counter has an 8-bit high-time setting and an 8-bit low-time setting. The duty cycle is the ratio of output high- or low-time to the total cycle time, which is the sum of the two. Additionally, these counters have two control bits, `rbypass`, for bypassing the counter, and `rselodd`, to select the output clock duty cycle.

When the `rbypass` bit is set to 1, it bypasses the counter, resulting in a divide by 1. When this bit is set to 0, the high- and low-time counters are added to compute the effective division of the VCO output frequency. For example, if the post-scale divide factor is 10, the high- and low-count values could be set to 5 and 5, respectively, to achieve a 50-50% duty cycle. The PLL implements this duty cycle by transitioning the output clock from high to low on the rising edge of the VCO output clock. However, a 4 and 6 setting for the high- and low-count values, respectively, would produce an output clock with a 40-60% duty cycle.

The `rselodd` bit indicates an odd divide factor for the VCO output frequency along with a 50% duty cycle. For example, if the post-scale divide factor is 3, the high- and low-time count values could be set to 2 and 1, respectively, to achieve this division. This implies a 67%-33% duty cycle. If you need a 50%-50% duty cycle, you can set the `rselodd` control bit to 1 to achieve this duty cycle despite an odd division factor. The PLL implements this duty cycle by transitioning the output clock from high to low on a falling edge of the VCO output clock. When you set `rselodd` = 1, you subtract 0.5 cycles from the high time and you add 0.5 cycles to the low time. For example:

- High-time count = 2 cycles
- Low-time count = 1 cycle
- `rselodd` = 1 effectively equals:
  - High-time count = 1.5 cycles
  - Low-time count = 1.5 cycles
  - Duty cycle = (1.5/3) % high-time count and (1.5/3) % low-time count

Scan Chain Description

Arria II GX PLLs have a 180-bit scan chain. Table 5–10 shows the number of bits for each component of an Arria II GX PLL.

Table 5–10. Arria II GX PLL Reprogramming Bits

Block Name	Number of Bits		Total
	Counter	Other *(1)*	
C6 *(2)*	16	2	18
C5	16	2	18
C4	16	2	18
C3	16	2	18
C2	16	2	18
C1	16	2	18
C0	16	2	18
M	16	2	18
N	16	2	18
Charge Pump Current	0	3	3
VCO Post-Scale divider (K)	1	0	1
Loop Filter Capacitor *(3)*	0	2	2
Loop Filter Resistor	0	5	5
Unused CP/LF	0	7	7
Total number of bits	—	—	180

Notes to Table 5–10:

(1) Includes two control bits, `rbypass`, for bypassing the counter, and `rselodd`, to select the output clock duty cycle.

(2) LSB bit for C6 low-count value is the first bit shifted into the scan chain for Left/Right PLLs.

(3) MSB bit for loop filter is the last bit shifted into the scan chain.

Figure 5–29 shows the scan chain order of Arria II GX PLL components which have 7 post-scale counters. The reconfiguration bits start with the C6 post-scale counter.

Figure 5–29. Scan-Chain Order of PLL Components for Arria II GX PLLs

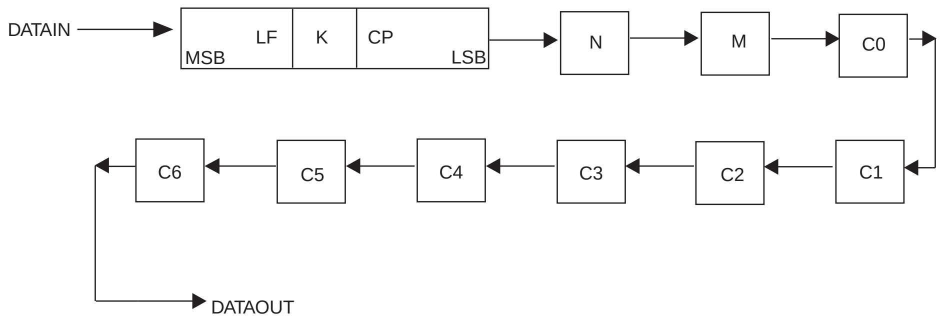
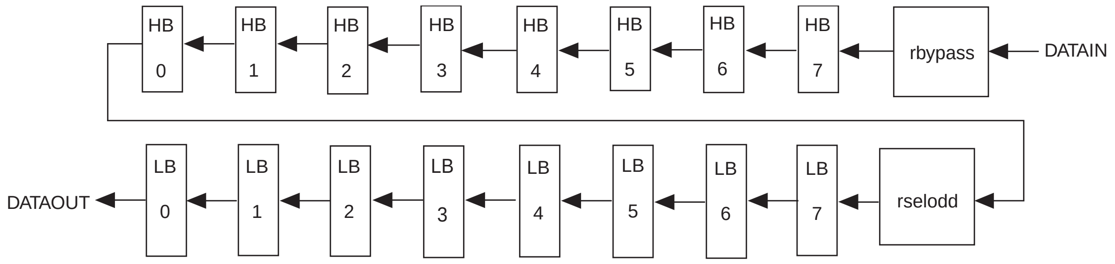


Figure 5–30 shows the scan-chain bit-order sequence for post-scale counters in all Arria II GX PLLs.

Figure 5–30. Scan-Chain Bit-Order Sequence for Post-Scale Counters in Arria II GX PLLs



Charge Pump and Loop Filter

You can reconfigure the charge-pump and loop-filter settings to update the PLL bandwidth in real time. Table 5–11 through Table 5–13 show the possible settings for charge pump current (`Icp`), loop-filter resistor (`R`), and capacitor (`C`) values for Arria II GX PLLs.

Table 5–11. charge_pump_current Bit Settings

CP[2]	CP[1]	CP[0]	Decimal Value for Setting
0	0	0	0
0	0	1	1
0	1	1	3
1	1	1	7

Table 5–12. loop_filter_r Bit Settings (Part 1 of 2)

LFR[4]	LFR[3]	LFR[2]	LFR[1]	LFR[0]	Decimal Value for Setting
0	0	0	0	0	0
0	0	0	1	1	3
0	0	1	0	0	4

Table 5–12. loop_filter_r Bit Settings (Part 2 of 2)

LFR[4]	LFR[3]	LFR[2]	LFR[1]	LFR[0]	Decimal Value for Setting
0	1	0	0	0	8
1	0	0	0	0	16
1	0	0	1	1	19
1	0	1	0	0	20
1	1	0	0	0	24
1	1	0	1	1	27
1	1	1	0	0	28
1	1	1	1	0	30

Table 5–13. loop_filter_c Bit Settings

LFC[1]	LFC[0]	Decimal Value for Setting
0	0	0
0	1	1
1	1	3

Bypassing PLL

Bypassing a PLL counter results in a multiply (`m` counter) or a divide (`n` and `C0` to `C6` counters) factor of one.

Table 5–14 shows the settings for bypassing the counters in Arria II GX PLLs.

Table 5–14. PLL Counter Settings

PLL Scan Chain Bits [0..8] Settings									
LSB *(2)*								**MSB** *(1)*	**Description**
0	X	X	X	X	X	X	X	1 *(3)*	PLL counter bypassed
X	X	X	X	X	X	X	X	0 *(3)*	PLL counter not bypassed because bit 8 (`MSB`) is set to 0

Notes to Table 5–14:

(1) Most significant bit (MSB)

(2) Least significant bit (LSB).

(3) Counter-bypass bit.

For more information on how to use the PLL scan chain bit settings, refer to the *ALTPLL_RECONFIG Megafunction User Guide*.

To bypass any of the PLL counters, set the bypass bit to 1, causing the values on the other bits to be ignored. To bypass the VCO post-scale counter (`K`), set the corresponding bit to 0.

Dynamic Phase-Shifting

The dynamic phase-shifting feature allows the output phases of individual PLL outputs to be dynamically adjusted relative to each other and to the reference clock without the need to send serial data through the scan chain of the corresponding PLL. This feature simplifies the interface and allows you to quickly adjust clock-to-out (t_{CO}) delays by changing the output clock phase-shift in real time. This adjustment is achieved by incrementing or decrementing the VCO phase-tap selection to a given C counter or to the M counter. The phase is shifted by 1/8 of the VCO frequency at a time. The output clocks are active during this phase-reconfiguration process.

Table 5–15 shows the control signals that are used for dynamic phase-shifting.

Table 5–15. Dynamic Phase-Shifting Control Signals

Signal Name	Description	Source	Destination
PHASECOUNTER SELECT[3:0]	Counter select. Four bits decoded to select either the M or one of the C counters for phase adjustment. One address maps to select all C counters. This signal is registered in the PLL on the rising edge of scanclk.	Logic array or I/O pins	PLL reconfiguration circuit
PHASEUPDOWN	Selects dynamic phase shift direction; 1= UP; 0= DOWN. Signal is registered in the PLL on the rising edge of scanclk.	Logic array or I/O pin	PLL reconfiguration circuit
PHASESTEP	Logic high enables dynamic phase shifting.	Logic array or I/O pin	PLL reconfiguration circuit
SCANCLK	Free running clock from core used in combination with PHASESTEP to enable/disable dynamic phase shifting. Shared with scanclk for dynamic reconfiguration.	GCLK/RCLK or I/O pin	PLL reconfiguration circuit
PHASEDONE	When asserted, it indicates to core-logic that the phase adjustment is complete and PLL is ready to act on a possible second adjustment pulse. Asserts based on internal PLL timing. De-asserts on rising edge of scanclk.	PLL reconfiguration circuit	Logic array or I/O pins

Table 5–16 shows the PLL counter selection based on the corresponding PHASECOUNTERSELECT setting.

Table 5–16. Phase Counter Select Mapping

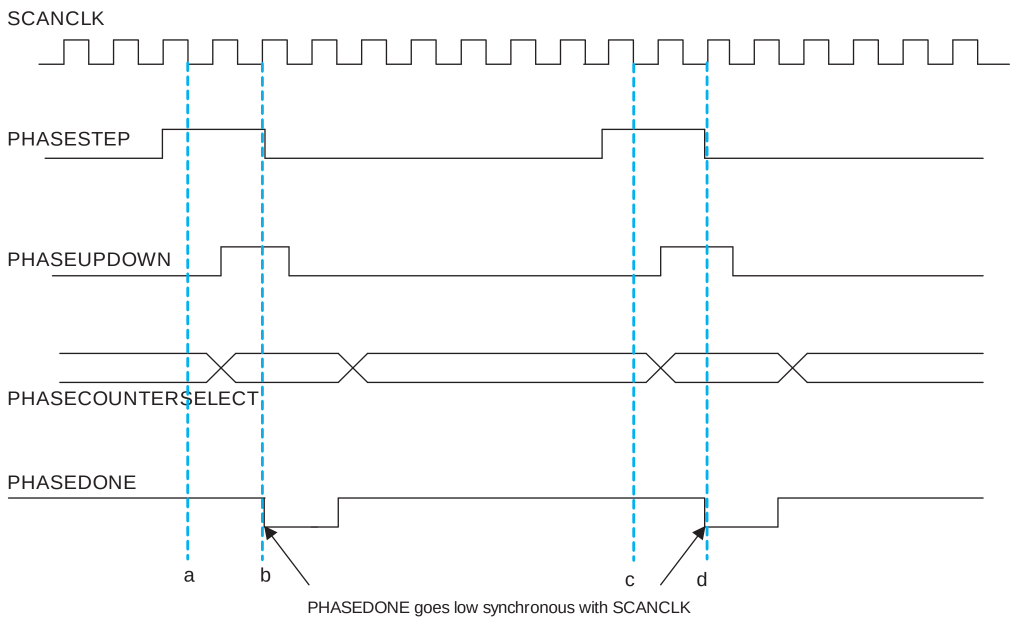
PHASECOUNTERSELECT[3]	[2]	[1]	[0]	Selects
0	0	0	0	All Output Counters
0	0	0	1	M Counter
0	0	1	0	C0 Counter
0	0	1	1	C1 Counter
0	1	0	0	C2 Counter
0	1	0	1	C3 Counter
0	1	1	0	C4 Counter
0	1	1	1	C5 Counter
1	0	0	0	C6 Counter

The procedure to perform one dynamic phase-shift step is as follows:

1. Set `phaseupdown` and `phasecounterselect` as required.
2. Assert `phasestep` for at least two `scanclk` cycles. Each `phasestep` pulse enables one phase shift.
3. De-assert `phasestep`.
4. Wait for `phasedone` to go high.
5. Repeat steps 1-4 as many times as required to perform multiple phase-shifts.

All signals are synchronous to `scanclk` and must meet tsu/th requirements with respect to `scanclk` edges. They are latched on `scanclk` edges and must meet tsu/th requirements with respect to `scanclk` edges.

Figure 5–31. Dynamic Phase Shifting Waveform



Dynamic phase-shifting can be repeated indefinitely. For example, in a design where the VCO frequency is set to 1000 MHz and the output clock frequency is 100 Mhz, performing 40 dynamic phase shifts (each one yields 125 ps phase shift) results in shifting the output clock by 180°, in other words, a phase shift of 5 ns.

The `phasestep` signal is latched on the negative edge of `scanclk`. In Figure 5–31, this is shown by the second `scanclk` falling edge. `phasestep` must stay high for at least two `scanclk` cycles. On the second `scanclk` rising edge after `phasestep` is latched (the fourth `scanclk` rising edge in Figure 5–31), the values of `phaseupdown` and `phasecounterselect` are latched and the PLL starts dynamic phase-shifting for the specified counters and in the indicated direction. On the fourth `scanclk` rising edge, `phasedone` goes high to low and remains low until the PLL finishes dynamic phase-shifting. You can perform another dynamic phase-shift after the `phasedone` signal goes from low to high.

Depending on the VCO and `scanclk` frequencies, `phasedone` low time may be greater than or less than one `scanclk` cycle.

For details about the ALTPLL_RECONFIG Megawizard Plug-In Manager, refer to the *ALTPLL_RECONFIG Megafunction User Guide*.

PLL Specifications

Refer to the *Arria II GX Device Data Sheet* chapter in volume 3 of the *Arria II GX Device Handbook* for information about PLL timing specifications.

Document Revision History

Table 5–17 shows the revision history for this document.

Table 5–17. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial Release.	—

Section II. I/O Interfaces

This section provides information on Arria® II GX device I/O features, external memory interfaces, and high-speed differential interfaces with DPA. This section includes the following chapters:

- Chapter 6, I/O Features in Arria II GX Devices
- Chapter 7, External Memory Interfaces in Arria II GX Devices
- Chapter 8, High-Speed Differential I/O Interfaces and DPA in Arria II GX Devices

Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in this volume.

6. I/O Features in Arria II GX Devices

AIIGX51006-1.0

This chapter provides guidelines for using industry I/O standards in Arria® II GX devices, including I/O features, I/O standards and structure, I/O banks, and design considerations.

This chapter includes the following sections:

Introduction

This chapter contains feature definitions of Arria II GX I/O elements (IOEs). It provides details about how an IOE works and its features. Arria II GX I/Os support a wide range of features:

- Up to 16 full-duplex CDR-based transceivers supporting data rates between 155 Mbps and 3.75 Gbps
- Dedicated circuitry to support physical layer functionality for popular serial protocols, such as PCI Express (PIPE) Gen1, Serial RapidIO®, SATA/SAS, SD/HD/3G SDI, ASI, CPRI/OBSAI, Gigabit Ethernet, XAUI/HiGig/HiGig+, SONET/SDH, GPON and Seriallite II
- Complete PIPE protocol solution with embedded PCI Express hard IP blocks that implement PHY-MAC layer, data link layer, and transaction layer functionality
- Single-ended, non-voltage-referenced and voltage-referenced I/O standards
- Low-voltage differential signaling (LVDS), reduced swing differential signal (RSDS), mini-LVDS, high-speed transceiver logic (HSTL), and SSTL
- Hard DPA block with serializer/deserializer (SERDES)
- Programmable output current strength
- Programmable slew rate
- Programmable bus-hold
- Programmable pull-up resistor
- Open-drain output
- Serial on-chip termination (OCT)
- Differential OCT

- Programmable pre-emphasis
- Programmable voltage output differential (V_{OD})

Arria II GX I/O Standards Support

Table 6–1 shows the supported I/O standards for Arria II GX devices and the typical values for input and output V_{CCIO}, V_{CCPD}, V_{REF}, and board V_{TT}.

Table 6–1. Arria II GX I/O Standards and Voltage Levels *(Note 1)*, *(2)*

I/O Standard	Standard Support	V_{CCIO} (V)		V_{CCPD} (V)	V_{REF} (V)	V_{TT} (V)
		Input Operation	Output Operation			
3.3-V LVTTL/3.3-V LVCMOS	JESD8-B	3.3/3.0/2.5	3.3	3.3	—	—
3.0-V LVTTL/3.0-V LVCMOS	JESD8-B	3.3/3.0/2.5	3.0	3.0	—	—
2.5-V LVTTL/LVCMOS	JESD8-5	3.3/3.0/2.5	2.5	2.5	—	—
1.8-V LVTTL/LVCMOS	JESD8-7	1.8/1.5	1.8	2.5	—	—
1.5-V LVCMOS	JESD8-11	1.8/1.5	1.5	2.5	—	—
1.2-V LVCMOS	JESD8-12	1.2	1.2	2.5	—	—
3.0-V PCI	PCI Rev 2.2	3.0	3.0	3.0	—	—
3.0-V PCI-X *(1)*	PCI-X Rev 1.0	3.0	3.0	3.0	—	—
SSTL-2 Class I and Class II	JESD8-9B	*(2)*	2.5	2.5	1.25	1.25
SSTL-18 Class I and Class II	JESD8-15	*(2)*	1.8	2.5	0.90	0.90
SSTL-15 Class I	—	*(2)*	1.5	2.5	0.75	0.75
HSTL-18 Class I and Class II	JESD8-6	*(2)*	1.8	2.5	0.90	0.90
HSTL-15 Class I and Class II	JESD8-6	*(2)*	1.5	2.5	0.75	0.75
HSTL-12 Class I and Class II	JESD8-16A	*(2)*	1.2	2.5	0.6	0.6
Differential SSTL-2	JESD8-9B	*(2)*, *(3)*	2.5	2.5	—	1.25
Differential SSTL-18	JESD8-15	*(2)*, *(3)*	1.8	2.5	—	0.90
Differential SSTL-15	—	*(2)*, *(3)*	1.5	2.5	—	0.75
Differential HSTL-18	JESD8-6	*(2)*, *(3)*	1.8	2.5	—	0.90
Differential HSTL-15	JESD8-6	*(2)*, *(3)*	1.5	2.5	—	0.75
Differential HSTL-12	JESD8-16A	*(2)*, *(3)*	1.2	2.5	—	0.60
LVDS	ANSI/TIA/ EIA-644	*(2)*	2.5	2.5	—	—
RSDS and mini-LVDS	—	—	2.5	2.5	—	—
LVPECL	—	2.5	—	2.5	—	—
BLVDS	—	2.5	2.5	2.5	—	—

Notes to Table 6–1:

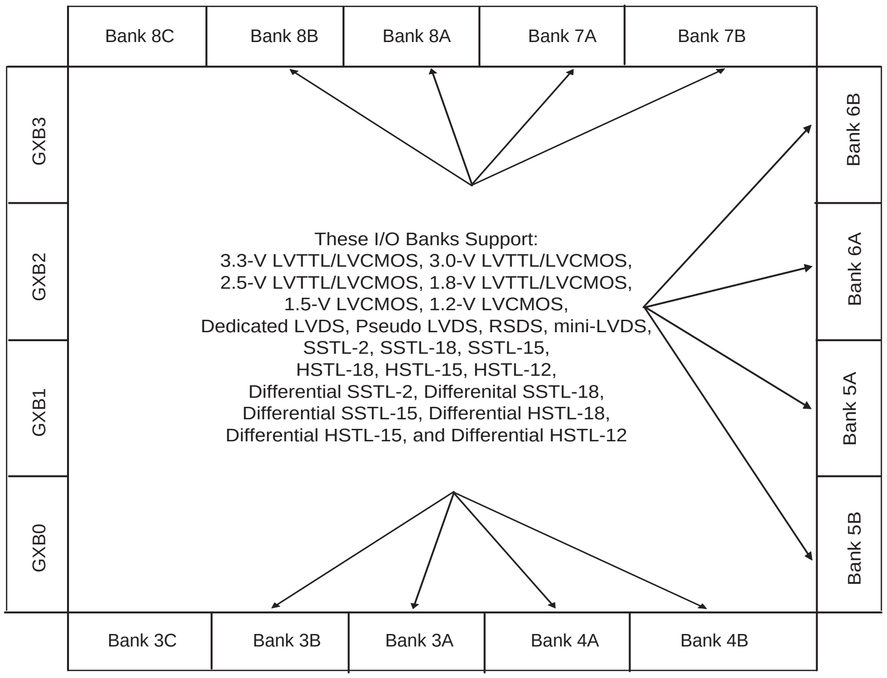
(1) PCI-X does not meet the PCI-X IV curve requirement at the linear region.
(2) Single-ended SSTL/HSTL, differential SSTL/HSTL, and LVDS input buffers are powered by V_{CCPD}.
(3) Differential SSTL/HSTL inputs use LVDS differential input buffers without on-chip R_D support.

For detailed electrical characteristics of each I/O standard, refer to the *Device Data Sheet* chapter in volume 3 of the *Arria II GX Device Handbook*.

Arria II GX I/O Banks

Arria II GX devices contain up to 16 I/O banks, as shown in Figure 6–1. The left side I/O banks contain high-speed transceiver banks, with dedicated configuration banks at banks 3C and 8C. The rest of the banks are user I/O banks. All user I/O banks support all single-ended and differential I/O standards.

Figure 6–1. Arria II GX Devices I/O Banks *(Note 1), (2), (3), (4), (5), (6)*


Bank 8C
Bank 8B
Bank 8A
Bank 7A
Bank 7B
GXB3
GXB2
GXB1
GXB0
Bank 6B
Bank 6A
Bank 5A
Bank 5B
These I/O Banks Support:
3.3-V LVTTL/LVCMOS, 3.0-V LVTTL/LVCMOS,
2.5-V LVTTL/LVCMOS, 1.8-V LVTTL/LVCMOS,
1.5-V LVCMOS, 1.2-V LVCMOS,
Dedicated LVDS, Pseudo LVDS, RSDS, mini-LVDS,
SSTL-2, SSTL-18, SSTL-15,
HSTL-18, HSTL-15, HSTL-12,
Differential SSTL-2, Differenital SSTL-18,
Differential SSTL-15, Differential HSTL-18,
Differential HSTL-15, and Differential HSTL-12
Bank 3C
Bank 3B
Bank 3A
Bank 4A
Bank 4B

Notes to Figure 6–1:

(1) Banks GXB0, GXB1, GXB2, and GXB3 are dedicated banks for high-speed transceiver I/Os.
(2) Banks 3C and 8C are dedicated configuration banks and do not have user I/O pins.
(3) LVDS with DPA is supported at banks 5A, 5B, 6A, and 6B.
(4) Differential HSTL and SSTL inputs use LVDS differential input buffers without differential OCT support.
(5) Differential HSTL and SSTL outputs are not true differential outputs. They use two single-ended outputs with the second output programmed as inverted.
(6) Figure 6–1 is a top view of the silicon die that corresponds to a reverse view for flip chip packages. It is a graphical representation only.

Modular I/O Banks

The I/O pins in Arria II GX devices are arranged in groups called modular I/O banks. Depending on device densities, the number of I/O banks range from 6 to 12, while the number of transceiver banks range from 1 to 4. Table 6–2 shows the number of I/O pins available in each I/O bank.

Table 6–2. Arria II GX Available I/O Pins in Each I/O Bank *(Note 1)*

Package	Device	Bank												Total
		3A	3B	4A	4B	5A	5B	6A	6B	7A	7B	8A	8B	
358-pin Flip Chip UBGA	2AGX20	22	—	38	—	18	—	18	—	38	—	22	—	156
	2AGX30	22	—	38	—	18	—	18	—	38	—	22	—	156
	2AGX45	22	—	38	—	18	—	18	—	38	—	22	—	156
	2AGX65	22	—	38	—	18	—	18	—	38	—	22	—	156
572-pin Flip Chip FBGA	2AGX20	38	—	38	—	50	—	50	—	38	—	38	—	252
	2AGX30	38	—	38	—	50	—	50	—	38	—	38	—	252
	2AGX45	38	—	38	—	50	—	50	—	38	—	38	—	252
	2AGX65	38	—	38	—	50	—	50	—	38	—	38	—	252
	2AGX95	38	—	42	—	50	—	50	—	38	—	42	—	260
	2AGX125	38	—	42	—	50	—	50	—	38	—	42	—	260
780-pin Flip Chip FBGA	2AGX45	54	—	70	—	66	—	50	—	70	—	54	—	364
	2AGX65	54	—	70	—	66	—	50	—	70	—	54	—	364
	2AGX95	54	—	74	—	66	—	50	—	70	—	58	—	372
	2AGX125	54	—	74	—	66	—	50	—	70	—	58	—	372
	2AGX190	54	—	74	—	66	—	50	—	70	—	58	—	372
	2AGX260	54	—	74	—	66	—	50	—	70	—	58	—	372
1152-pin Flip Chip FBGA	2AGX95	70	—	74	16	66	—	66	—	70	16	74	—	452
	2AGX125	70	—	74	16	66	—	66	—	70	16	74	—	452
	2AGX190	70	32	74	32	66	32	66	32	70	32	74	32	612
	2AGX260	70	32	74	32	66	32	66	32	70	32	74	32	612

Note to Table 6–2:

(1) The number of I/O pins do not include transceiver pins.

In Arria II GX devices, the maximum number of I/O banks per side is four, excluding the configuration banks. All Arria II GX devices support migration across device density and package. When migrating between devices with a different number of I/O banks per side, it is the "B" bank which is removed or inserted. For example, when moving from a 12-bank device to an 8-bank device, the banks that are dropped are "B" banks, namely: 3B, 5B, 6B, and 8B. Similarly, when moving from an 8-bank device to a 12-bank device, the banks that are added are "B" banks, namely: 3B, 5B, 6B, and 8B.

During migration from a smaller device to a larger device, the bank size increases or remains the same but never decreases. Figure 6–3 shows pin migration across device densities and packages.

Table 6–3. Arria II GX Pin Migration across Densities

Package	**Pin Type**	**Device**							
		2AGX20	**2AGX30**	**2AGX45**	**2AGX65**	**2AGX95**	**2AGX125**	**2AGX190**	**2AGX260**
358-pin Flip Chip UBGA	I/O	144	144	144	144	—	—	—	—
	Clock	12	12	12	12	—	—	—	—
	XCVR channel	4	4	4	4	—	—	—	—
572-pin Flip Chip FBGA	I/O	240	240	240	240	248	248	—	—
	Clock	12	12	12	12	12	12	—	—
	XCVR channel	4	4	8	8	8	8	—	—
780-pin Flip Chip FBGA	I/O	—	—	352	352	360	360	360	360
	Clock	—	—	12	12	12	12	12	12
	XCVR channel	—	—	8	8	12	12	12	12
1152-pin Flip Chip FBGA	I/O	—	—	—	—	440	440	600	600
	Clock	—	—	—	—	12	12	12	12
	XCVR channel	—	—	—	—	12	12	16	16

Note to Table 6–3:

(1) Each transceiver channel consists of two Tx pins, two Rx pins and a transceiver clock pin.

Arria II GX I/O Structure

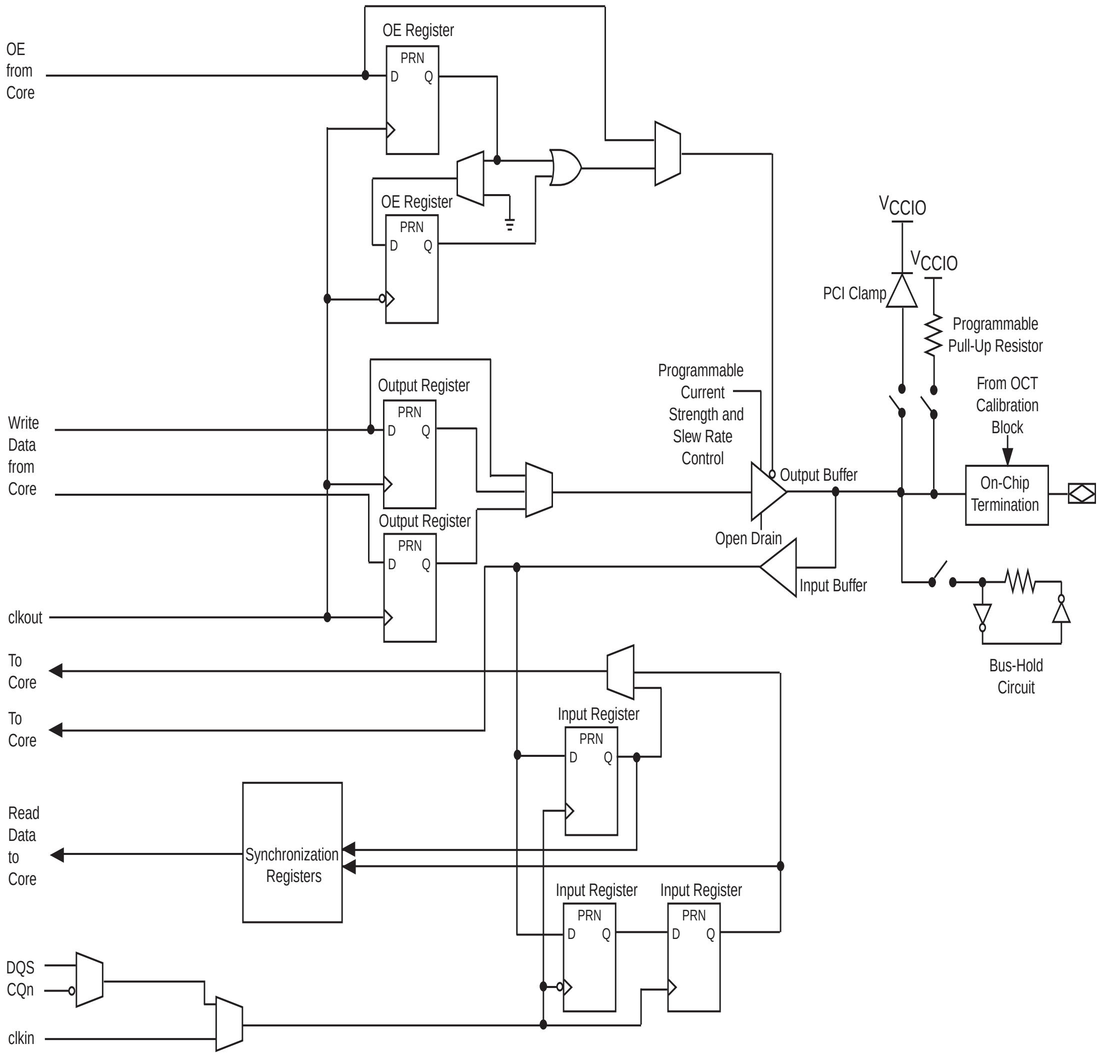
The I/O element in Arria II GX devices contains a bidirectional I/O buffer and I/O registers to support a completely embedded bidirectional SDR or DDR transfer. The IOEs are located in I/O blocks around the periphery of the Arria II GX device. There are up to four IOEs per row I/O block and four IOEs per column I/O block.

The Arria II GX bidirectional IOE supports these features:

- Programmable input delay
- Programmable output current strength
- Programmable slew rate
- Programmable bus-hold
- Programmable pull-up resistor
- Open-drain output
- On-chip series termination with or without calibration
- On-chip differential termination
- PCI clamping diode

Figure 6–2 shows the Arria II GX IOE structure.

Figure 6–2. Arria II GX IOE Structure



For more information about I/O registers and how they are used for memory applications, refer to the *External Memory Interfaces in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

3.3-V I/O Interface

Arria II GX I/O buffers are fully compatible with 3.3-V I/O standards. You can use them as transmitters or receivers in your system. The output high voltage (V_{OH}), output low voltage (V_{OL}), input high voltage (V_{IH}), and input low voltage (V_{IL}) levels meet the 3.3-V I/O standard specifications defined by EIA/JEDEC Standard JESD8-B with margin when the Arria II GX V_{CCIO} voltage is powered by 3.3 V or 3.0 V.

To ensure device reliability and proper operation when interfacing with a 3.3-V I/O system using Arria II GX devices, it is important to ensure that the absolute maximum ratings are not violated. Altera recommends performing IBIS simulation to determine that the overshoot and undershoot voltages are in the guidelines. There are several techniques that you can use to limit overshoot and undershoot voltages, though none are required.

When using the Arria II GX device as a transmitter, techniques to limit overshoot and undershoot at the I/O pins include using slow slew rate and series termination. Transmission line effects that cause large voltage deviations at the receiver are associated with an impedance mismatch between the driver and transmission line. By matching the impedance of the driver to the characteristic impedance of the transmission line, you can significantly reduce overshoot voltage. You can use a series termination resistor placed physically close to the driver to match the total driver impedance to transmission line impedance. Other than 3.3-V LVTTL and 3.3-V LVCMOS I/O standards, Arria II GX devices support series on-chip termination for all LVTTL/LVCMOS I/O standards in all I/O banks.

When using the Arria II GX device as a receiver, a technique you can use to limit overshoot, though not required, is using a clamping diode (on-chip or off-chip termination). Arria II GX devices provide an optional on-chip PCI clamp diode for I/O pins. You can use this diode to protect I/O pins against overshoot voltage.

Another method for limiting overshoot is reducing the bank supply voltage (V_{CCIO}) to 3.0 V. In this method, the clamp diode (on-chip or off-chip termination), though not required, can sufficiently clamp overshoot voltage to in the DC- and AC-input voltage specification. The clamped voltage can be expressed as the sum of the supply voltage (V_{CCIO}) and the diode forward voltage. By lowering V_{CCIO} to 3.0 V, you can reduce overshoot and undershoot for all I/O standards, including 3.3-V LVTTL/LVCMOS, 3.0-V LVTTL/LVCMOS, and 3.0-V PCI/PCI-X. Additionally, lowering V_{CCIO} to 3.0 V reduces power consumption.

For more information about absolute maximum rating and maximum allowed overshoot during transitions, refer to the *Device Data Sheet* chapter in volume 3 of the *Arria II GX Device Handbook*.

External Memory Interfaces

In addition to I/O registers in each IOE, Arria II GX devices also have dedicated registers and phase-shift circuitry on all I/O banks for interfacing with external memory interfaces.

For more information about external memory interfaces, refer to the *External Memory Interfaces in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

High-Speed Differential I/O with DPA Support

Arria II GX devices have the following dedicated circuitry for high-speed differential I/O support:

- Differential I/O buffer
- Transmitter serializer
- Receiver deserializer

- Data realignment circuitry
- Dynamic phase aligner (DPA)
- Synchronizer (FIFO buffer)
- Phase-locked loops (PLLs)

For more information about DPA support, refer to the *High-Speed Differential I/O Interfaces with DPA in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

Programmable Current Strength

The output buffer for each Arria II GX device I/O pin has a programmable current-strength control for certain I/O standards. You can use programmable current strength to mitigate the effects of high signal attenuation due to a long transmission line or a legacy backplane. The LVTTL, LVCMOS, SSTL, and HSTL standards have several levels of current strength that you can control. Table 6–4 lists the programmable current strength settings.

Table 6–4. Programmable Current Strength Settings *(Note 1)*

I/O Standard	I_{OL} / I_{OH} Current Strength Setting (mA) for Top, Bottom, and Right I/O Pins
3.3-V LVTTL *(2)*	12, [8], 4
3.3-V LVCMOS *(2)*	[2]
3.0-V LVTTL	16, 12, 8, 4
3.0-V LVCMOS	16, 12, 8, 4
2.5-V LVTTL/LVCMOS	16, 12, 8, 4
1.8-V LVTTL/LVCMOS	16, 12, 10, 8, 6, 4, 2
1.5-V LVCMOS	16, 12, 10, 8, 6, 4, 2
1.2-V LVCMOS	12, 10, 8, 6, 4, 2
SSTL-2 Class I	12, 8
SSTL-2 Class II	16
SSTL-18 Class I	12, 10, 8
SSTL-18 Class II	16, 12
SSTL-15 Class I	12, 10, 8
HSTL-18 Class I	12, 10, 8
HSTL-18 Class II	16
HSTL-15 Class I	12, 10, 8
HSTL-15 Class II	16
HSTL-12 Class I	12, 10, 8
HSTL-12 Class II	16

Notes to Table 6–4:

(1) The default current strength setting in the Quartus® II software is 50-Ω OCT R_S without calibration for all non-voltage reference and HSTL/SSTL Class I I/O standards. The default setting is 25-Ω OCT R_S without calibration for HSTL/SSTL Class II I/O standards.

(2) The default current strength setting in the Quartus II software is the current strength, shown in brackets [].

☞ Altera recommends performing IBIS or SPICE simulations to determine the right current strength setting for your specific application.

Programmable Slew Rate Control

The output buffer for each Arria II GX device regular- and dual-function I/O pin has a programmable output slew rate control that you can configure for low-noise or high-speed performance. A faster slew rate provides high-speed transitions for high-performance systems. A slow slew rate can help reduce system noise, but adds a nominal delay to the 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.

☞ You cannot use the programmable slew rate feature when using OCT R_S.

The Quartus II software allows two settings for programmable slew rate control: fast and slow. In the Quartus II Assignment Editor, setting 2 = fast, and 0 = slow. Programmable slew rate is available for 8 mA current strength and above.

You can use faster slew rates to improve the available timing margin in memory-interface applications or when the output pin has high-capacitive loading. Altera recommends performing IBIS or SPICE simulations to determine the right slew rate setting for your specific application.

Open-Drain Output

Arria II GX devices provide an optional open-drain output (equivalent to an open collector output) for each I/O pin. When configured as open drain, the logic value of the output is either high-Z or 0. Typically, an external pull-up resistor is needed to provide logic high.

Bus Hold

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

Bus-hold circuitry also pulls non-driven 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 over-driving signals. If you enable the bus-hold feature, you cannot use the programmable pull-up option. The bus-hold feature is disabled if the I/O pin is configured for differential signals.

Bus-hold circuitry uses a resistor with a nominal resistance to weakly pull the last-driven state.

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 Arria II GX device I/O pin provides an optional programmable pull-up resistor during user mode. If you enable this feature for an I/O pin, the pull-up resistor weakly holds the I/O to the V_{CCIO} level.

Programmable pull-up resistors are only supported on user I/O pins and are not supported on dedicated configuration pins, JTAG pins, or dedicated clock pins. If you enable the programmable pull-up option, you cannot use the bus-hold feature.

Programmable Pre-Emphasis

Arria II GX IV LVDS transmitters support programmable pre-emphasis to compensate the frequency dependent attenuation of the transmission line. The Quartus II software allows two settings for programmable pre-emphasis control—0 and 1—where 0 is pre-emphasis off and 1 is pre-emphasis on. The default setting is 1.

For more information about programmable pre-emphasis, refer to the *High-Speed Differential I/O Interfaces with DPA in the Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

Programmable Differential Output Voltage

Arria II GX LVDS transmitters support programmable V_{OD}. Programmable V_{OD} settings allow you to adjust output eye height to optimize trace length and power consumption. A higher V_{OD} swing improves voltage margins at the receiver end, while a smaller V_{OD} swing reduces power consumption. The Quartus II software allows three settings for programmable V_{OD}: low, medium, and high.

For more information about programmable V_{OD}, refer to the *High-Speed Differential I/O Interfaces with DPA in the Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

MultiVolt I/O Interface

Arria II GX architecture supports the MultiVolt I/O interface feature that allows Arria II GX devices in all packages to interface with systems of different supply voltages.

You can connect the `VCCIO` pins to a 1.2-, 1.5-, 1.8-, 2.5-, 3.0-, 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).

Arria II GX `VCCPD` power pins must be connected to a 2.5-, 3.0-, or 3.3-V power supply. Using these power pins to supply pre-driver power to the output buffers increases the performance of the output pins. Table 6–5 summarizes Arria II GX MultiVolt I/O support.

Table 6–5. Arria II GX MultiVolt I/O Support *(Note 1)*, *(2)*

VCCIO (V)	Input Signal (V)						Output Signal (V)					
	1.2	1.5	1.8	2.5	3.0	3.3	1.2	1.5	1.8	2.5	3.0	3.3
1.2	✓	—	—	—	—	—	✓	—	—	—	—	—
1.5	—	✓	✓	—	—	—	—	✓	—	—	—	—
1.8	—	✓	✓	—	—	—	—	—	✓	—	—	—
2.5	—	—	—	✓	✓	—	—	—	—	✓	—	—
3.0	—	—	—	✓	✓	—	—	—	—	—	✓	—
3.3	—	—	—	✓	✓	✓	—	—	—	—	—	✓

Notes to Table 6–5:

(1) The pin current may be slightly higher than the default value. You must verify that the driving device's V_{OL} maximum and V_{OH} minimum voltages do not violate the applicable Arria II GX V_{IL} maximum and V_{IH} minimum voltage specifications.

(2) Altera recommends that you use an external clamp diode on the column I/O pins when the input signal is 3.0 V or 3.3 V.

Arria II GX OCT Support

Arria II GX devices feature series on-chip termination to provide I/O impedance matching and termination capabilities. On-chip termination maintains signal quality, saves board space, and reduces external component costs.

Arria II GX devices support on-chip series (R_S) with or without calibration and on-chip differential termination (R_D) for differential LVDS I/O standards. Arria II GX devices support OCT in all user I/O banks by selecting one of the OCT I/O standards.

Arria II GX devices support OCT R_S in the same I/O bank with different I/O standards if they use the same V_{CCIO} supply voltage. You can independently configure each I/O in an I/O bank to support OCT R_S or programmable current strength.

You cannot configure both OCT R_S and programmable current strength for the same I/O buffer.

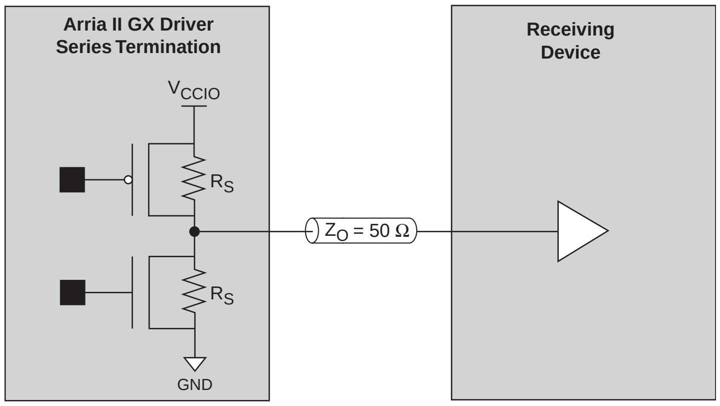
A pair of R_{UP} and R_{DN} pins are available in a given I/O bank for series calibrated termination. R_{UP} and R_{DN} pins share the same V_{CCIO} and GND, respectively, with the I/O bank where they are located. R_{UP} and R_{DN} pins are dual-purpose I/Os, and function as regular I/Os if you do not use the calibration circuit. When used for calibration, R_{UP} and R_{DN} pins are connected to V_{CCIO} or GND, respectively, through an external 25-Ω ±1% or 50-Ω ±1% resistor for an on-chip series termination value of 25 Ω or 50 Ω, respectively.

On-Chip Series (R_S) Termination without Calibration

Arria II GX devices support driver-impedance matching to provide the I/O driver with controlled output impedance that closely matches the impedance of the transmission line. As a result, you can significantly reduce reflections. Arria II GX devices support on-chip series termination for single-ended I/O standards (see Figure 6–3).

The R_S shown in Figure 6–3 is the intrinsic impedance of output transistors. The typical R_S values are 25 Ω and 50 Ω.

Figure 6–3. Arria II GX On-Chip Series Termination without Calibration



To use on-chip termination for:

- SSTL Class I standard, you must select the **50-Ω on-chip series termination** setting, thus eliminating the external 25-Ω R_S (to match the 50-Ω transmission line).
- SSTL Class II standard, you must select the 2**5-Ω on-chip series termination** setting (to match the 50-Ω transmission line and the near-end external 50-Ω pull-up to V_{TT}).

On-Chip Series Termination with Calibration

Arria II GX devices support on-chip series termination with calibration in all banks. The on-chip series termination calibration circuit compares the total impedance of the I/O buffer to the external 25-Ω ±1% or 50-Ω ±1% resistors connected to the R_{UP} and R_{DN} pins, and dynamically enables or disables the transistors until they match.

The R_S shown in Figure 6–4 is the intrinsic impedance of transistors. Calibration occurs at the end of device configuration. When the calibration circuit finds the correct impedance, it powers down and stops changing the characteristics of the drivers.

Figure 6–4. Arria II GX On-Chip Series Termination with Calibration

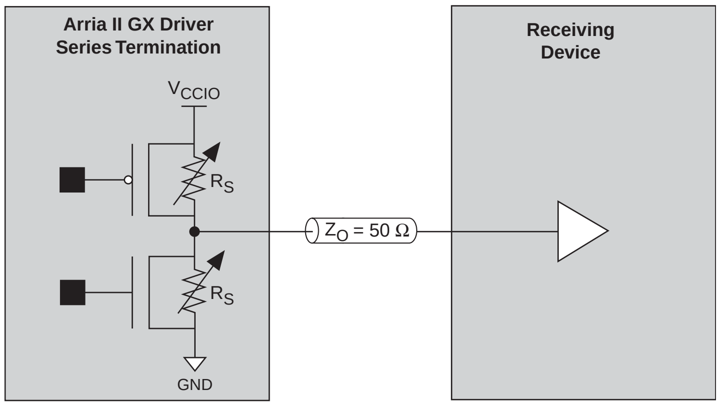


Table 6–6 lists the I/O standards that support on-chip series termination with calibration.

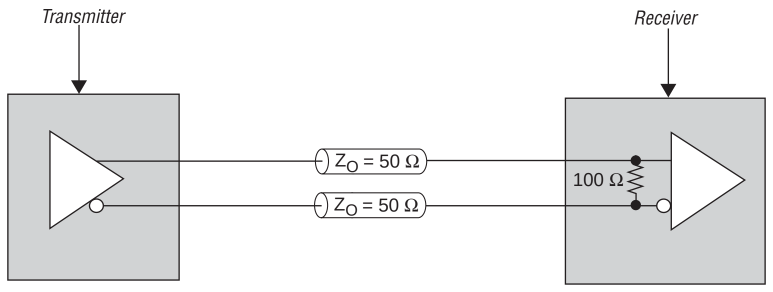
Table 6–6. Selectable I/O Standards with On-Chip Series Termination with Calibration

I/O Standard	On-Chip Series Termination Setting		
	Right I/O	**Top and Bottom I/O**	**Unit**
3.0-V LVTTL/LVCMOS	50	50	Ω
	25	25	Ω
2.5-V LVTTL/LVCMOS	50	50	Ω
	25	25	Ω
1.8-V LVTTL/LVCMOS	50	50	Ω
	25	25	Ω
1.5-V LVCMOS	50	50	Ω
	25	25	Ω
1.2-V LVCMOS	50	50	Ω
	25	25	Ω
SSTL-2 Class I	50	50	Ω
SSTL-2 Class II	25	25	Ω
SSTL-18 Class I	50	50	Ω
SSTL-18 Class II	25	25	Ω
SSTL-15 Class I	50	50	Ω
HSTL-18 Class I	50	50	Ω
HSTL-18 Class II	25	25	Ω
HSTL-15 Class I	50	50	Ω
HSTL-15 Class II	25	25	Ω
HSTL-12 Class I	50	50	Ω
HSTL-12 Class II	25	25	Ω

LVDS Input On-Chip Termination (R_D)

All I/O banks in Arria II GX devices support input differential on-chip termination R_D with a nominal resistance value of 100 Ω, as shown in Figure 6–5. However, not all input differential pins support on-chip termination R_D.

Figure 6–5. Differential Input On-Chip Termination



For more information about differential on-chip termination, refer to the *High-Speed Differential I/O Interfaces with DPA in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook.*

Arria II GX OCT Calibration

Arria II GX devices support calibrated on-chip series termination (R_S) on all I/O pins. You can calibrate the Arria II GX I/O bank with any of the three OCT calibration blocks (CB) available in the devices.

OCT Calibration Block

The three OCT calibration blocks reside in the top-left, top-right, and bottom-left corners of the device.

An OCT calibration block has the same V_{CCIO} as the I/O bank that contains the block. OCT R_S calibration is supported on all user I/O banks with different V_{CCIO} voltage standards, up to the number of available OCT calibration blocks. You can configure I/O banks to receive calibrated codes from any OCT calibration block with the same V_{CCIO}. All I/O banks with the same V_{CCIO} can share one OCT calibration block, even if that particular I/O bank has an OCT calibration block.

Arria II GX Termination Schemes for I/O Standards

The following section describes the different termination schemes for I/O standards used in Arria II GX devices.

Single-Ended I/O Standards Termination

Voltage-referenced I/O standards require both an input reference voltage, V_{REF}, and a termination voltage, V_{TT}. The reference voltage of the receiving device tracks the termination voltage of the transmitting device. Figure 6–6 shows the details of SSTL I/O termination on Arria II GX devices.

Figure 6–6. Arria II GX SSTL I/O Standard Termination

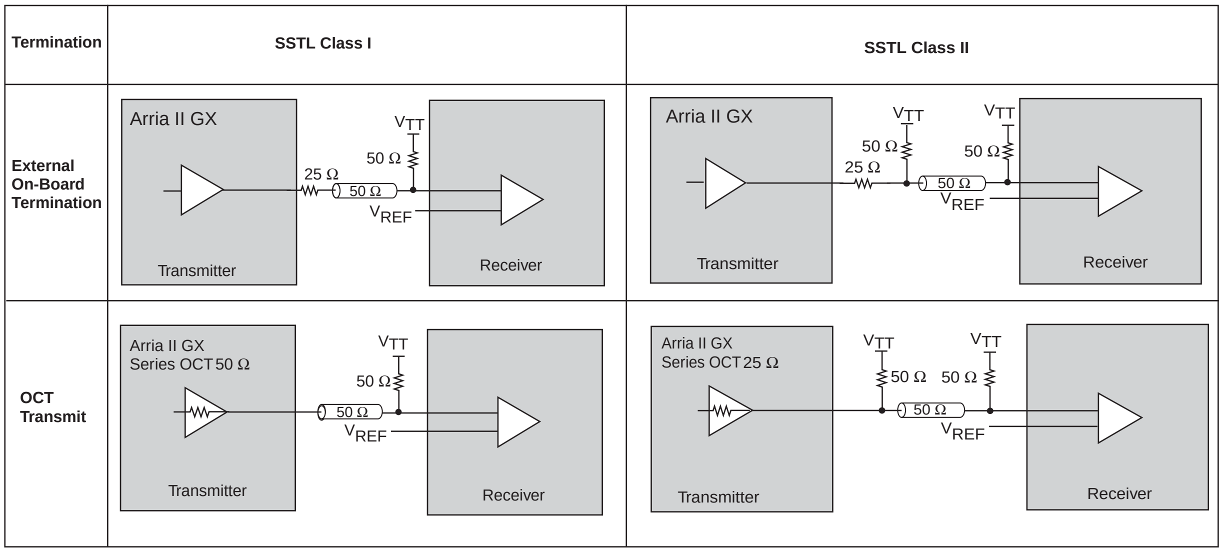
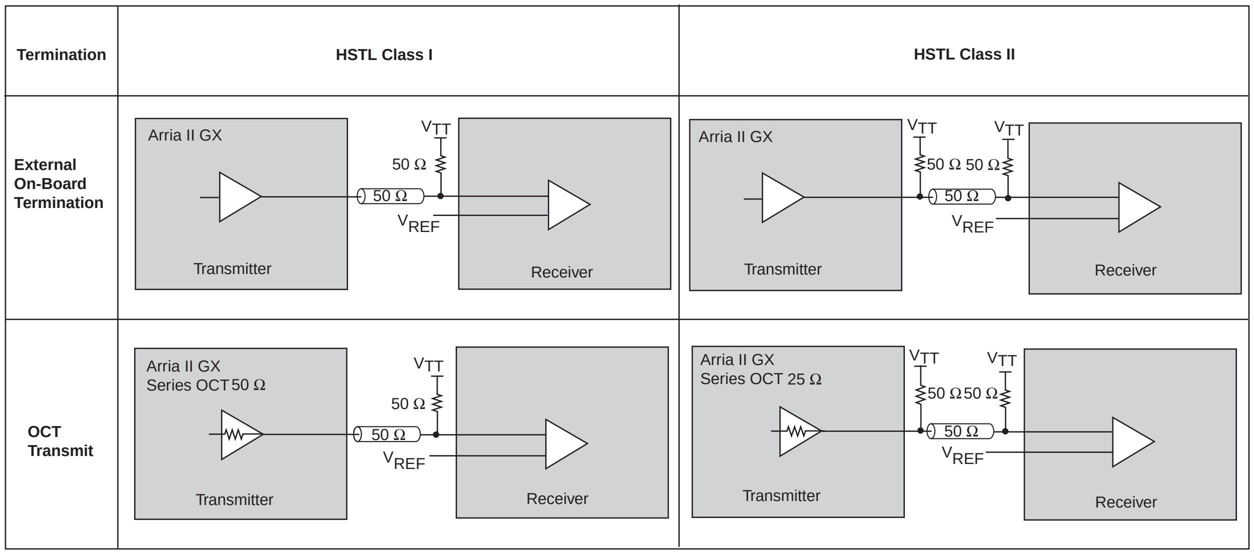


Figure 6–7 shows the details of HSTL I/O termination on Arria II GX devices.

Figure 6–7. Arria II GX HSTL I/O Standard Termination



Differential I/O Standards Termination

Arria II GX devices support differential SSTL-2 and SSTL-18, differential HSTL-18, HSTL-15, HSTL-12, LVDS, LVPECL, RSDS, and mini-LVDS. Figure 6–8 through Figure 6–14 show the details of various differential I/O terminations on Arria II GX devices.

Figure 6–8 shows the details of differential SSTL I/O standard termination on Arria II GX devices.

Figure 6–8. Arria II GX Differential SSTL I/O Standard Termination

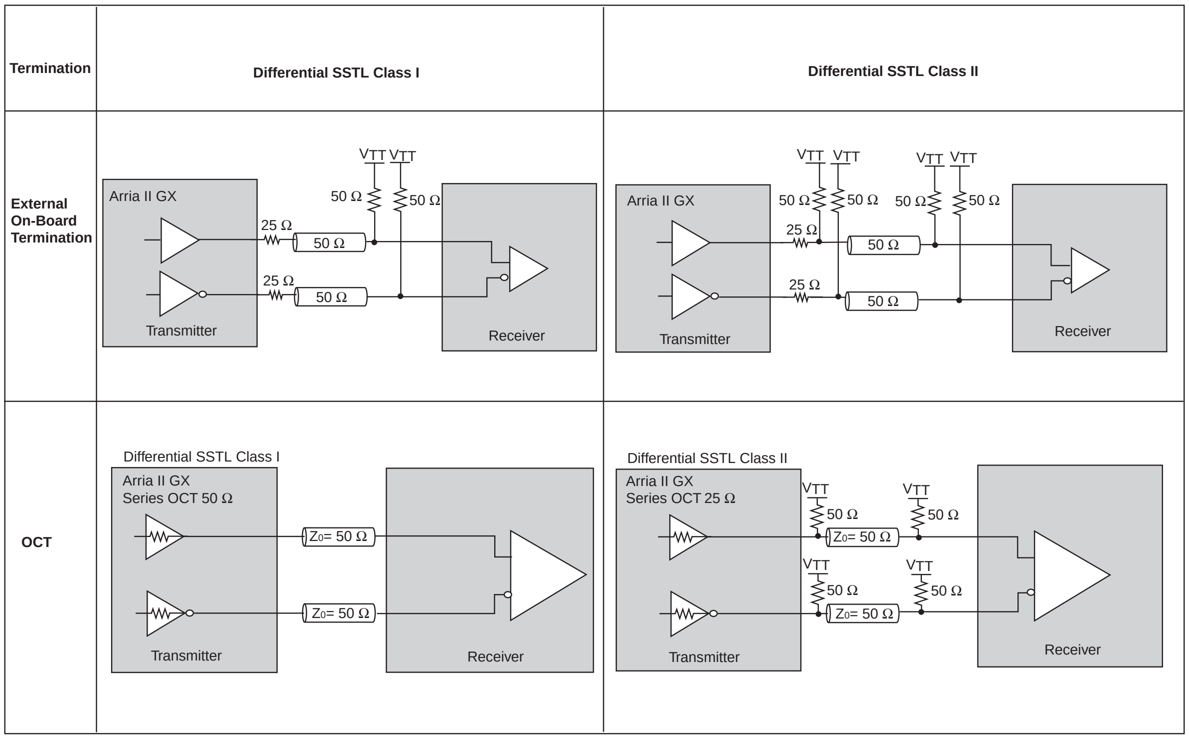
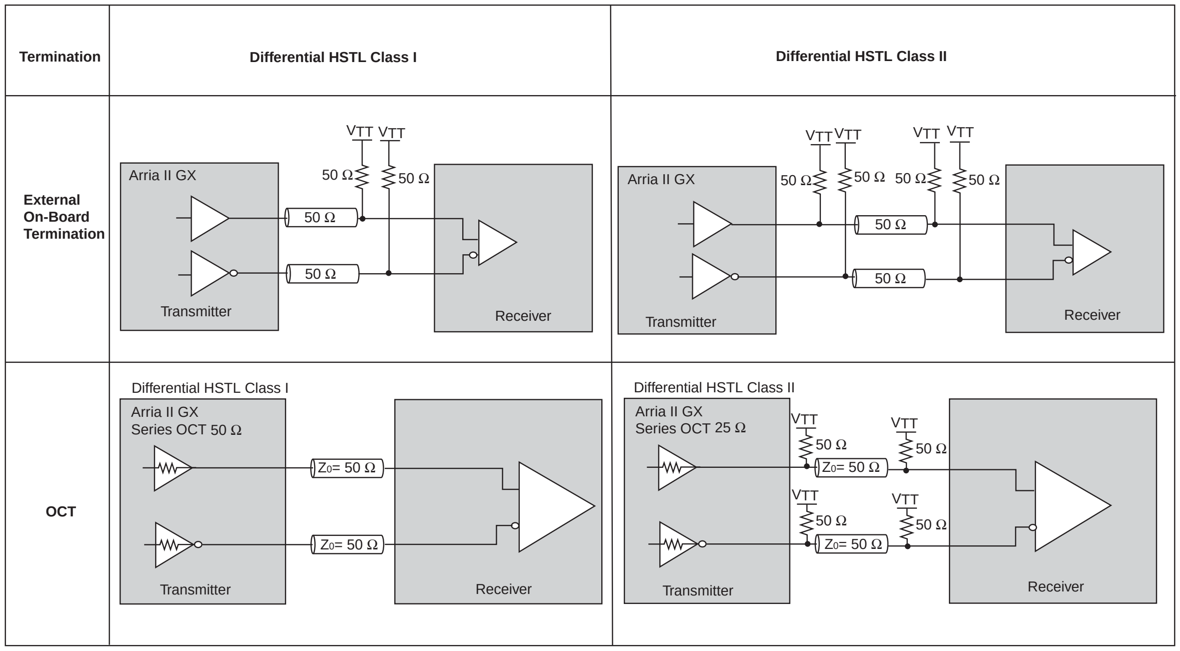

Termination
Differential SSTL Class I
Differential SSTL Class II
External On-Board Termination
OCT
Arria II GX
Transmitter
Receiver
VTT
50 Ω
25 Ω
Differential SSTL Class I
Arria II GX Series OCT 50 Ω
Z0= 50 Ω
Differential SSTL Class II
Arria II GX Series OCT 25 Ω

Figure 6–9 shows the details of differential HSTL I/O standard termination on Arria II GX devices.

Figure 6–9. Arria II GX Differential HSTL I/O Standard Termination

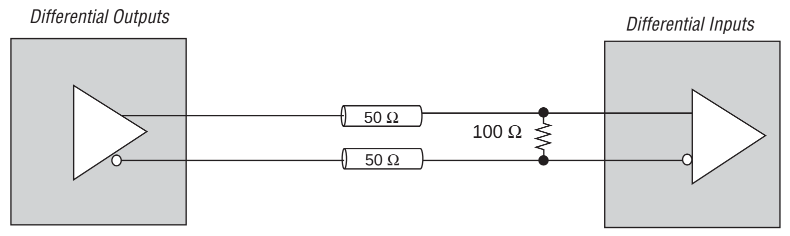
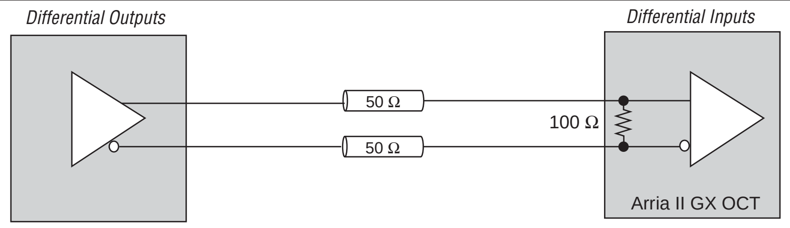
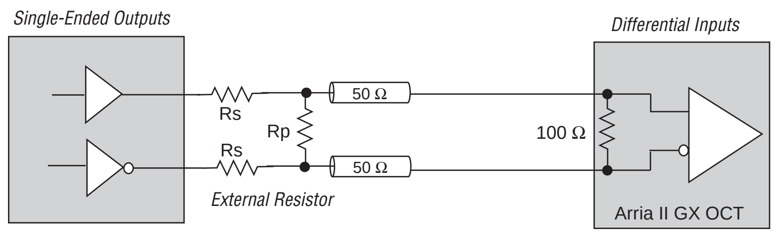

Termination
Differential HSTL Class I
Differential HSTL Class II
External On-Board Termination
VTT
50 Ω
Arria II GX
Transmitter
Receiver
OCT
Arria II GX Series OCT 50 Ω
Arria II GX Series OCT 25 Ω
Z0= 50 Ω

LVDS

The LVDS I/O standard is a differential high-speed, low-voltage swing, low-power, general-purpose I/O interface standard. In Arria II GX devices, the LVDS I/O standard requires a 2.5-V V_{CCIO} level. The LVDS input buffer requires 2.5-V V_{CCPD}. LVDS requires a 100-Ω termination resistor between the two signals at the input buffer. Arria II GX devices provide an optional 100-Ω differential termination resistor in the device using on-chip differential termination. The external-resistor topology is for a data rate of up to 700 Mbps.

Figure 6–10 shows the details of LVDS termination in Arria II GX devices.

Figure 6–10. Arria II GX LVDS I/O Standard Termination *(Note 1)*

Termination	LVDS
External On-Board Termination	 Differential Outputs Differential Inputs 50 Ω 50 Ω 100 Ω
OCT Receive (Dedicated LVDS Output)	 Differential Outputs Differential Inputs 50 Ω 50 Ω 100 Ω Arria II GX OCT
OCT Receive (Single-Ended LVDS Output with Three Resistor Network, LVDS_E_3R) *(1)*	 Single-Ended Outputs Differential Inputs Rs Rp Rs 50 Ω 50 Ω 100 Ω External Resistor Arria II GX OCT

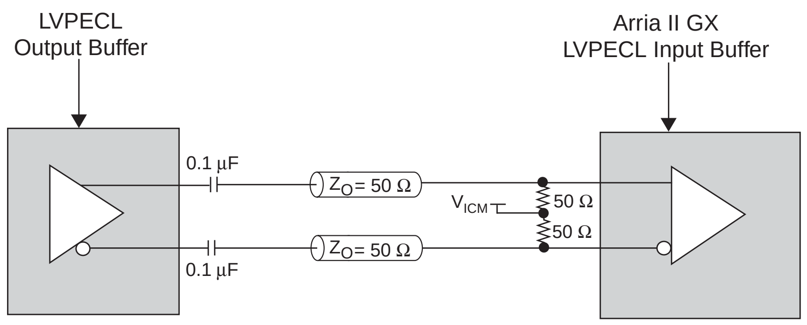
Note to Figure 6–10:

(1) $R_p = 170\ \Omega$ and $R_s = 120\ \Omega$ for LVDS_E_3R.

Differential LVPECL

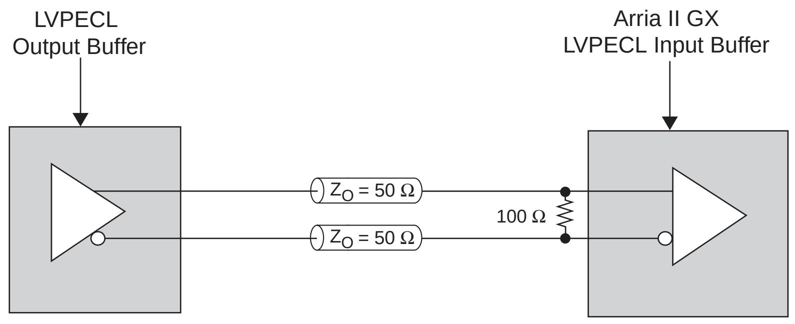
Arria II GX devices support the LVPECL I/O standard on input clock pins only. LVPECL output operation is not supported. LVDS input buffers are used to support LVPECL input operation. AC-coupling is required when the LVPECL common mode voltage of the output buffer is higher than Arria II GX LVPECL input common mode voltage. Figure 6–11 shows the AC-coupled termination scheme. The 50-Ω resistors used at the receiver end are external to the device.

Figure 6–11. LVPECL AC-Coupled Termination



Arria II GX devices support DC-coupled LVPECL if the LVPECL output common mode voltage is in the Arria II GX LVPECL input buffer specification (see Figure 6–12).

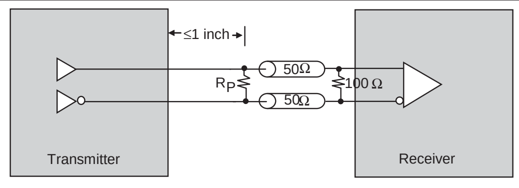
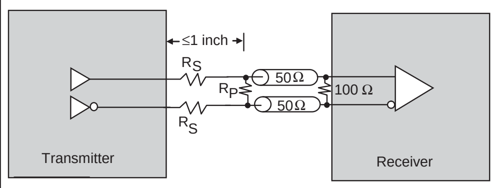
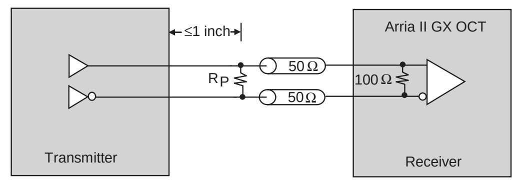
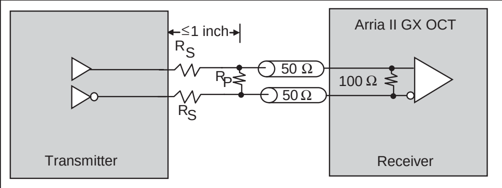
Figure 6–12. LVPECL DC-Coupled Termination



RSDS

Arria II GX devices support the RSDS output standard with a data rate of up to 360 Mbps using LVDS output buffer types. Arria II GX devices supports dedicated RSDS, RSDS with a one-resistor network, and RSDS with a three-resistor network. Two single-ended output buffers are used for external one- or three-resistor networks, as shown in Figure 6–13.

Figure 6–13. Arria II GX RSDS I/O Standard Termination *(Note 1)*

Termination	One-Resistor Network (RSDS_E_1R)	Three-Resistor Network (RSDS_E_3R)
External On-Board Termination		
OCT		

Note to Figure 6–13:

(1) The R_S and R_P values are pending characterization.

A resistor network is required to attenuate the LVDS output-voltage swing to meet RSDS specifications. You can modify the three-resistor network values to reduce power or improve the noise margin. The resistor values chosen should satisfy the equation shown in Equation 6–1.

Equation 6–1.

$$\frac{R_S \times \frac{R_P}{2}}{R_S + \frac{R_P}{2}} = 50\ \Omega$$

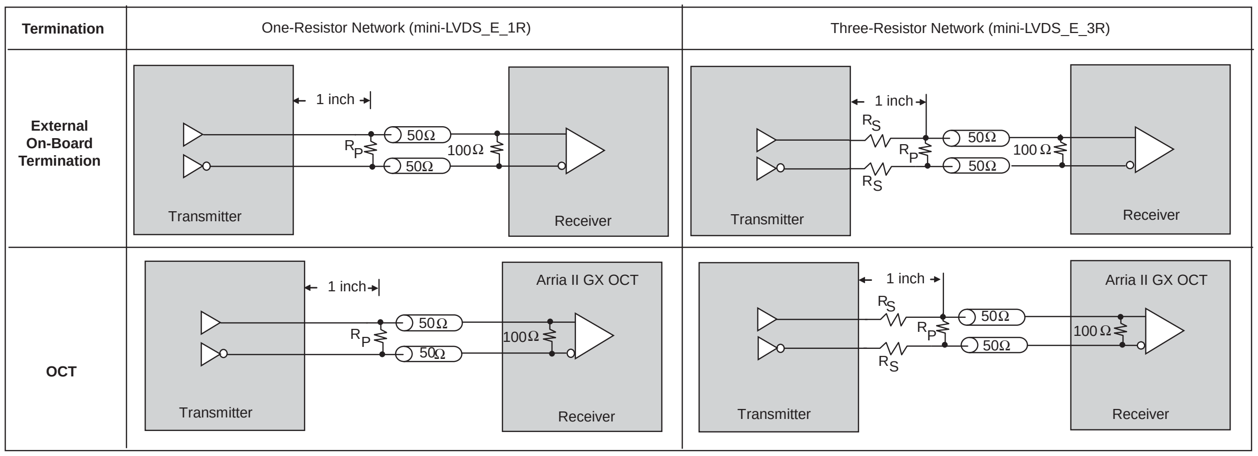
Altera recommends that you perform additional simulations using IBIS models to validate that custom resistor values meet the RSDS requirements.

For more information about the RSDS I/O standard, refer to the *RSDS Specification* from the National Semiconductor website at www.national.com.

Mini-LVDS

Arria II GX devices support the mini-LVDS output standard with a data rate up to 400 Mbps using LVDS-type output buffers. Arria II GX devices support dedicated RSDS as well as RSDS with a one-resistor network or a three-resistor network. Two single-ended output buffers are used for external one- or three-resistor networks, as shown in Figure 6–14.

Figure 6–14. Arria II GX mini-LVDS I/O Standard Termination *(Note 1)*


Termination
One-Resistor Network (mini-LVDS_E_1R)
Three-Resistor Network (mini-LVDS_E_3R)
External On-Board Termination
OCT
1 inch
R_P
R_S
50Ω
100Ω
Transmitter
Receiver
Arria II GX OCT

Note to Figure 6–14:

(1) The R_S and R_P values are pending characterization.

A resistor network is required to attenuate the LVDS output voltage swing to meet mini-LVDS specifications. You can modify the three-resistor network values to reduce power or improve noise margin. The resistor values chosen should satisfy the equation shown in Equation 6–2.

Equation 6–2.

$$\frac{R_S \times \frac{R_P}{2}}{R_S + \frac{R_P}{2}} = 50\ \Omega$$

Altera recommends that you perform additional simulations using IBIS models to validate that custom resistor values meet the RSDS requirements.

For more information about the mini-LVDS I/O standard, see the *mini-LVDS Specification* from the Texas Instruments website at www.ti.com.

Arria II GX Design Considerations

While Arria II GX devices feature various I/O capabilities for high-performance and high-speed system designs, the following items require attention to ensure the success of these designs:

- "I/O Termination" on page 6–21
- "I/O Bank Restrictions" on page 6–22
- "I/O Placement Guidelines" on page 6–22

I/O Termination

This section describes I/O termination requirements for single-ended and differential I/O standards.

Single-Ended I/O Standards

Although single-ended, non-voltage-referenced I/O standards do not require termination, impedance matching is necessary to reduce reflections and improve signal integrity.

Voltage-referenced I/O standards require both an input reference voltage, V_{REF} and a termination voltage, V_{TT}. The reference voltage of the receiving device tracks the termination voltage of the transmitting device. Each voltage-referenced I/O standard requires a specific termination setup. For example, a proper resistive signal termination scheme is critical in SSTL2 standards to produce a reliable DDR memory system with superior noise margin.

Arria II GX on-chip series termination provides the convenience of no external components. When optimizing OCT for use in typical transmission line environments, the R_S impedance must be equal to or less than the transmission line impedance for optimal performance. In ideal applications, setting the OCT impedance to match the transmission line impedance avoids reflections. Alternatively, you can use external pull-up resistors to terminate the voltage-referenced I/O standards such as SSTL and HSTL.

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 signal line. Arria II GX devices provide an optional differential on-chip resistor when using LVDS.

For PCB layout guidelines, refer to *AN 224: High-Speed Board Layout Guidelines* and *AN 315: Guidelines for Designing High Speed FPGA PCBs*.

I/O Bank Restrictions

Each I/O bank can simultaneously support multiple I/O standards. The following sections provide guidelines for mixing non-voltage-referenced and voltage-referenced I/O standards in Arria II GX devices.

Non-Voltage-Referenced Standards

Each Arria II GX device I/O bank has its own `VCCIO` pins and supports only one V_{CCIO}, either 1.2, 1.5, 1.8, 2.5, 3.0, or 3.3 V. An I/O bank can simultaneously support any number of input signals with different I/O standard assignments, as shown in Table 6–1 on page 6–2.

For output signals, a single I/O bank supports non-voltage-referenced output signals that drive at the same voltage as V_{CCIO}. Because an I/O bank can only have one V_{CCIO} value, it can only drive out the 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 standard inputs and outputs and 3.0-V LVCMOS inputs (but not output or bidirectional pins).

Voltage-Referenced Standards

To accommodate voltage-referenced I/O standards, each Arria II GX device's user I/O bank has a dedicated V_{REF} pin. Each bank can only have a single V_{CCIO} voltage level and a single V_{REF} voltage level at a given time.

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.

Voltage-referenced bidirectional and output signals must be the same as the I/O bank's V_{CCIO} voltage. For example, you can only place SSTL-2 output pins in an I/O bank with a 2.5-V V_{CCIO}.

Mixing Voltage-Referenced and Non-Voltage-Referenced Standards

An I/O bank can support both non-voltage-referenced and voltage-referenced pins by applying each of the rule sets individually. For example, an I/O bank can support SSTL-18 inputs and 1.8-V inputs and outputs with a 1.8-V V_{CCIO} and a 0.9-V V_{REF}. Similarly, an I/O bank can support 1.5-V standards, 1.8-V inputs (but not outputs), and HSTL and HSTL-15 I/O standards with a 1.5-V V_{CCIO} and 0.75-V V_{REF}.

I/O Placement Guidelines

This section provides I/O placement guidelines for the programmable I/O standards supported by Arria II GX devices and includes essential information for designing systems using an Arria II GX device's selectable I/O capabilities.

3.3-V, 3.0-V, and 2.5-V LVTTL/LVCMOS Tolerance Guidelines

Altera recommends the following techniques when you use 3.3-, 3.0-, and 2.5-V I/O standards to limit overshoot and undershoot at I/O pins:

- Low drive strength or series termination—The I/O driver's impedance must be equal to or greater than the board trace impedance to minimize overshoot and undershoot at the un-terminated receiver end. If high driver strength (lower driver impedance) is needed, Altera recommends series termination at the driver end (on-chip or off-chip).

- Output slew rate—Arria II GX devices have two levels of slew rate control for single-ended output buffers. Slow slew rate can significantly reduce the overshoot and undershoot in the system at the cost of slightly slower performance.
- Input clamping diodes—Arria II GX I/Os have on-chip clamping diodes.
- When you use clamping diodes, the floating well of the I/O is clamped to V_{CCN}. As a result, the Arria II GX device might draw extra input leakage current from the external input driver. This may violate the hot-socket DC- and AC-current specification and increase power consumption. With the clamping diode enabled, the Arria II GX device supports a maximum DC current of 8 mA.

Pin Placement Guideline

You are advised to create a Quartus II design, enter your device I/O assignments, and compile your design to validate your pin placement. The Quartus II software checks your pin connections with respect to I/O assignment and placement rules to ensure proper device operation. These rules are dependent on device density, package, I/O assignments, voltage assignments, and other factors that are not described in this chapter.

For information about pin placement with respect to V_{REF}, LVDS, mini-LVDS, and RSDS, refer to the *I/O Management* chapter in volume 2 of the *Quartus II Development Software Handbook*.

Document Revision History

Table 6–7 shows the revision history for this document.

Table 6–7. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial release.	—

7. External Memory Interfaces in Arria II GX Devices

AIIGX51007-1.0

Introduction

Altera's Arria® II GX FPGAs provide an efficient architecture to quickly and easily fit wide external memory interfaces with their small modular I/O bank structure. The I/Os are designed to provide flexible and high-performance support for existing and emerging external double data rate (DDR) memory standards, such as DDR3, DDR2, DDR SDRAM, and QDRII SRAM. The Arria II GX FPGA supports DDR external memory on the top, bottom, and right I/O banks.

The high-performance memory interface solution includes a self-calibrating megafunction (ALTMEMPHY), optimized to take advantage of the Arria II GX I/O structure and the Quartus® II TimeQuest timing analyzer. The ALTMEMPHY megafunction provides the total solution for the highest reliable frequency of operation across process, voltage, and temperature (PVT) variations.

This chapter includes the following sections:

- "Arria II GX Memory Interfaces Pin Support" on page 7–3
- "Arria II GX External Memory Interface Features" on page 7–14

Table 7–1 summarizes the maximum clock rate Arria II GX devices support with external memory devices.

Table 7–1. Arria II GX Maximum Clock Rate Support for External Memory Interfaces with Half-Rate PHY *(Note 1)*, *(2)*

Memory Standards	C4 Speed Grade (MHz)	C5 Speed Grade (MHz)	C6 Speed Grade (MHz)
DDR3 SDRAM *(3)*	300	N/A	N/A
DDR2 SDRAM *(4)*	300 *(5)*	267 *(6)*	200
DDR SDRAM *(4)*	200	200	200
QDRII SRAM *(7)*, *(8)*	250	250	200

Notes to Table 7–1:

(1) These numbers are preliminary until characterization is final. The supported operating frequencies listed here are memory interface maximums for the FPGA device family. Your design's actual achievable performance is based on design- and system-specific factors, as well as static timing analysis of the completed design.

(2) The maximum clock rates are applicable to Class I termination, for interfaces using either row I/Os or column I/Os. The maximum clock rates are lower for Class II termination or interfaces using a combination of row and column I/Os.

(3) Arria II GX devices support DDR3 SDRAM components only as read and write leveling capability is not available in Arria II GX devices, hence DDR3 DIMMs cannot be supported. Interface with multiple DDR3 SDRAM components requires component arrangement according to the DDR2 DIMM tree topology.

(4) This applies to interfaces with both components and single-rank, unbuffered modules.

(5) The 300-MHz DDR2 interface requires the use of 400-MHz DDR2 SDRAM modules or components.

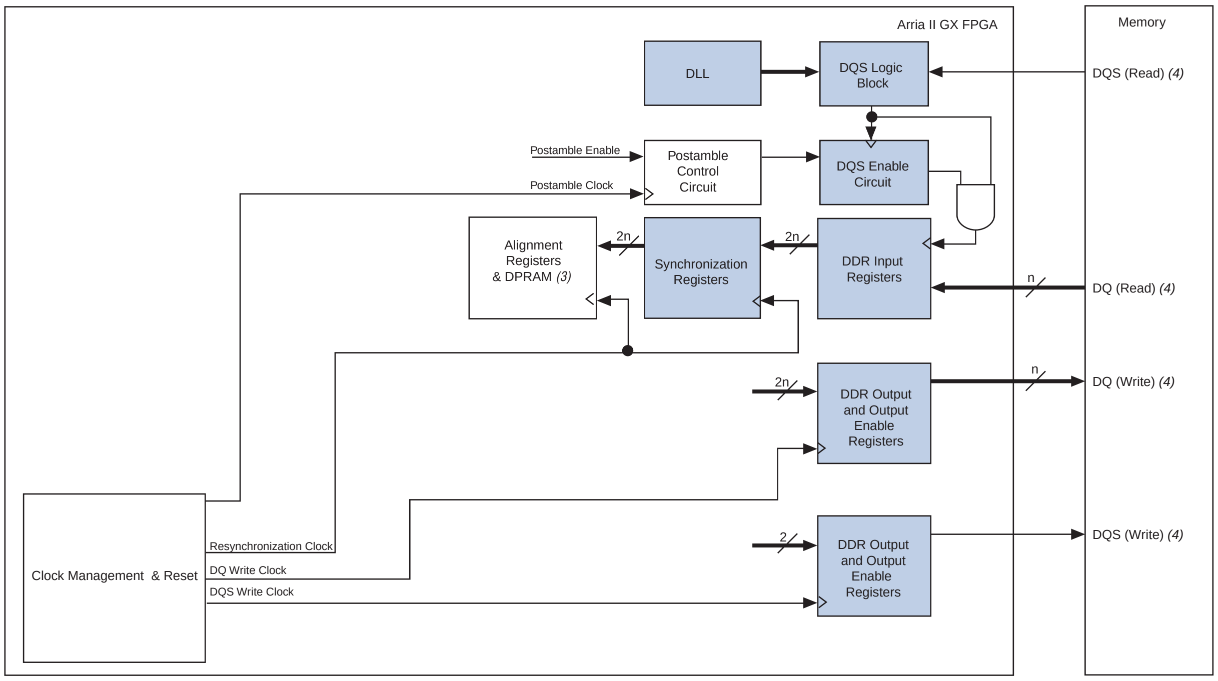
(6) The 267-MHz DDR2 interface requires the use of 333-MHz DDR2 SDRAM modules or components.

(7) QDRII SRAM supports 1.8-V and 1.5-V HSTL I/O standards. However, Altera recommends using the 1.8-V HSTL I/O standard for maximum performance because of the higher I/O drive strength.

(8) Arria II GX devices feature I/Os capable of electrical support for QDRII. However, Altera does not currently supply a controller or PHY megafunction for QDRII interfaces.

Figure 7–1 shows a memory interface data path overview.

Figure 7–1. External Memory Interface Data Path Overview *(Note 1)*, *(2)*



Notes to Figure 7–1:

(1) You can bypass each register block.
(2) Shaded blocks are implemented in the I/O element (IOE).
(3) The memory blocks used for each memory interface may differ slightly. The alignment register is implemented in the core logic.
(4) These signals may be bidirectional or unidirectional, depending on the memory standard. When bidirectional, the signal is active during both read and write operations.

This chapter describes the hardware features in Arria II GX devices that facilitate high-speed memory interfacing for the DDR memory standard including delay-locked loops (DLLs).

Memory interfaces also use I/O features such as on-chip termination (OCT), programmable input delay chains, programmable output delay, slew rate adjustment, and programmable drive strength.

For more information about any of the above-mentioned features, refer to the *I/O Features in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

The ALTMEMPHY megafunction instantiates a phase-locked loop (PLL) and PLL reconfiguration logic to adjust the phase shift based on VT variation.

For more information about the ALTMEMPHY support for Arria II GX devices, refer to the *External Memory PHY Interface Megafunction User Guide (ALTMEMPHY)*.

For more information about the Arria II GX PLL, refer to the *Clock Networks and PLLs in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

Arria II GX Memory Interfaces Pin Support

A typical memory interface requires data (D, Q, or DQ), data strobe (DQS/CQ and DQSn/CQn), address, command, and clock pins. Some memory interfaces use data mask (DM or BWSn) pins to enable write masking. This section describes how Arria II GX devices support all these pins.

Table 7–2 summarizes the pin connections between an Arria II GX device and an external memory device.

Table 7–2. Arria II GX Memory Interfaces Pin Utilization

Pin Description	Memory Standard	Arria II GX Pin Utilization
Read Data	All	DQ
Write Data	All	DQ *(1)*
Parity, DM, BWSn, ECC	All	DQ *(1)*, *(2)*
Read Data Strobes/Clocks	DDR3 SDRAM DDR2 SDRAM (with differential DQS signaling) *(3)*	Differential DQS/DQSn (also used as write data clock)
	DDR2 SDRAM (with single-ended DQS signaling) *(3)* DDR SDRAM	Single-ended DQS (also used as write data clock)
	QDRII SRAM	Complementary CQ/CQn
Write Data Clocks	QDRII SRAM *(4)*	Any DQS and DQSn pin pairs associated with the DQ groups used for the write data pins *(1)*
Memory Clocks (for Address and Commands) *(5)*	DDR3 SDRAM DDR2 SDRAM DDR SDRAM	Any unused DQ or DQS pins with `DIFFOUT` capability for the `mem_clk[n:0]` and `mem_clk_n[n:0]` signals.

Notes to Table 7–2:

(1) If the write data signals are unidirectional, connect them, including the data mask pins, to a separate DQS/DQ group other than the read DQS/DQ group. Connect the write clock to the DQS and DQSn pin-pair associated with that DQS/DQ group. Do not use the CQ and CQn pin-pair as write clocks.

(2) The BWSn and DM pins need to be part of the write DQS/DQ group, while parity and ECC pins need to be part of the read DQS/DQ group.

(3) DDR2 SDRAM supports either single-ended or differential DQS signaling.

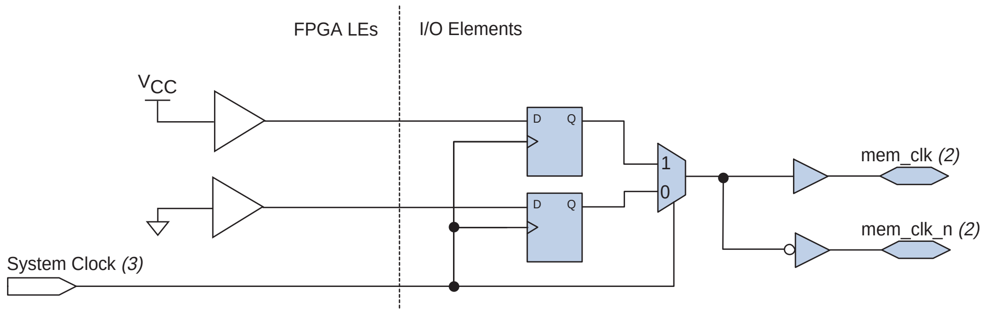
(4) QDRII SRAM devices use the K/K# clock pin-pair to latch write data, address, and command signals. The clocks must be part of the DQS/DQ group and follow the write data clock rules in this case.

(5) ALTMEMPHY megafunction implementation for a DDR3, DDR2, or DDR SDRAM interface requires that you place all memory clock pin-pairs in a single DQ group of adequate width to minimize skew. For example, DIMMs requiring three memory clock pin-pairs need to use a ×4 DQS/DQ group.

DDR3, DDR2, and DDR SDRAM devices use CK and CK# signals to capture the address and command signals. Generate these signals to mimic the write-data strobe using Arria II GX DDR I/O registers (DDIOs) to ensure that timing relationships between CK/CK# and DQS signals (`tDQSS`, `tDSS`, and `tDSH` in DDR3, DDR2, and DDR SDRAM devices) are met. QDRII SRAM devices use the same clock (K/K#) to capture the write data, address, and command signals.

Memory clock pins in Arria II GX devices are generated using a DDIO register going to differential output pins (see Figure 7–2), marked in the pin table with `DIFFOUT`, `DIFFIO_TX`, or `DIFFIO_RX` prefixes. For more information about which pins to use for memory clock pins, refer to Table 7–2 on page 7–3.

Figure 7–2. Memory Clock Generation *(Note 1)*, *(2)*, *(3)*



Notes to Figure 7–2:

(1) Refer to Table 7–2 on page 7–3 for the pin location requirements for these pins.

(2) The `mem_clk[0]` and `mem_clk_n[0]` pins for DDR3, DDR2, and DDR SDRAM interfaces use the I/O input buffer for feedback; therefore, bidirectional I/O buffers are used for these pins. For memory interfaces using a differential DQS input, the input feedback buffer is configured as differential input; for memory interfaces using a single-ended DQS input, the input buffer is configured as a single-ended input. Using a single-ended input feedback buffer requires that the I/O standard's V_{REF} voltage is provided to that I/O bank's V_{REF} pins.

(3) Global or regional clock networks are required for memory output clock generation to minimize jitter.

Arria II GX devices offer differential input buffers for differential read-data strobe and clock operations. In addition, Arria II GX devices also provide an independent DQS logic block for each CQn pin for complementary read-data strobe and clock operations. In the Arria II GX pin tables, the differential DQS pin pairs are denoted as DQS and DQSn pins, while the complementary CQ signals are denoted as CQ and CQn pins. DQSn and CQn pins are marked separately in the pin table. Each CQn pin connects to a DQS logic block and the shifted CQn signals go to the negative-edge input registers in the DQ I/O element registers.

DQ pins can be bidirectional signals, as in DDR3, DDR2, and DDR SDRAM, or unidirectional signals, as in QDRII SRAM devices. Connect the unidirectional read-data signals to Arria II GX DQ pins and the unidirectional write-data signals to a different DQS/DQ group than the read DQS/DQ group. Furthermore, the write clocks must be assigned to the DQS/DQSn pins associated to this write DQS/DQ group. Do not use the CQ/CQn pin-pair for write clocks.

Using a DQS/DQ group for the write-data signals minimizes output skew and allows vertical migration.

The DQS and DQ pin locations are fixed in the pin table. Memory interface circuitry is available in every Arria II GX I/O bank that does not support transceivers. All memory interface pins support the I/O standards required to support DDR3, DDR2, DDR SDRAM, and QDRII SRAM devices.

The Arria II GX device supports DQS and DQ signals with DQ bus modes of ×4, ×8/×9, ×16/×18, or ×32/×36. The DDR, DDR2, and DDR3 interfaces use one DQS pin for each x8 group; for example, an interface with a ×72 DDR2 DIMM needs nine DQS pins. When any of these pins are not used for memory interfacing, you can use them as user I/Os. In addition, you can use any DQSn or CQn pins not used for clocking as DQ (data) pins. Table 7–3 lists pin support per DQS/DQ bus mode, including the DQS/CQ and DQSn/CQn pin pair.

Table 7–3. Arria II GX DQS/DQ Bus Mode Pins

Mode	DQSn Support	CQn Support	Parity or DM (Optional)	Typical Number of Data Pins per Group	Maximum Number of Data Pins per Group *(1)*
×4	Yes	No	No *(5)*	4	5
×8/×9 *(2)*	Yes	Yes	Yes	8 or 9	11
×16/×18 *(3)*	Yes	Yes	Yes	16 or 18	23
×32/×36 *(4)*	Yes	Yes	Yes	32 or 36	47

Notes to Table 7–3:

(1) This represents the maximum number of DQ pins (including parity and data mask pins) connected to the DQS bus network with single-ended DQS signaling. When you use differential or complementary DQS signaling, the maximum number of data-per-group decreases by one. This number may vary per DQS/DQ group in a particular device. For the DDR, DDR2, and DDR3 interface, the number of pins is further reduced for interfaces larger than ×8 due to the need of one DQS pin for each ×8 group. Check with the pin table for the accurate number per group.

(2) Two ×4 DQS/DQ groups are stitched together to make a ×8/×9 group, so there are a total of 12 pins in this group.

(3) Four ×4 DQS/DQ groups are stitched together to make a ×16/×18 group.

(4) Eight ×4 DQS/DQ groups are stitched together to make a ×32/×36 group.

(5) The DM pin can be supported if the differential DQS is not used and the group does not have additional signals.

Figure 7–4 through Figure 7–9 show the maximum number of DQS/DQ groups per side of the Arria II GX device. These figures represent the die-top view of the Arria II GX device.

Table 7–4 shows the number of I/O modules and DQS/DQ groups per side of the Arria II GX device.

Table 7–4. Number of DQS/DQ Groups and I/O Modules in Arria II GX Devices per Side (Part 1 of 2)

Device	Package	Side	I/O Module *(1)*	DQS/DQ Groups			
				×4	×8/×9	×16/×18	×32/×36
EP2AGX20 EP2AGX30 EP2AGX45 EP2AGX65	358-Pin Ultra FineLine BGA	Top/Bottom	3	6	3	1	0
		Right	2	4	2	2	0
EP2AGX20 EP2AGX30 EP2AGX45 EP2AGX65 EP2AGX95 EP2AGX125	572-Pin FineLine BGA	Top/Bottom	4	8	4	2	0
		Right	6	12	6	2	0
EP2AGX45 EP2AGX65 EP2AGX95 EP2AGX125 EP2AGX190 EP2AGX260	780-Pin FineLIne BGA	Top/Bottom/ Right	7	14	7	3	1
EP2AGX95 EP2AGX125	1152-Pin FineLine BGA	Top/Bottom	9	18	9	4	2
		Right	8	16	8	4	2

Table 7–4. Number of DQS/DQ Groups and I/O Modules in Arria II GX Devices per Side (Part 2 of 2)

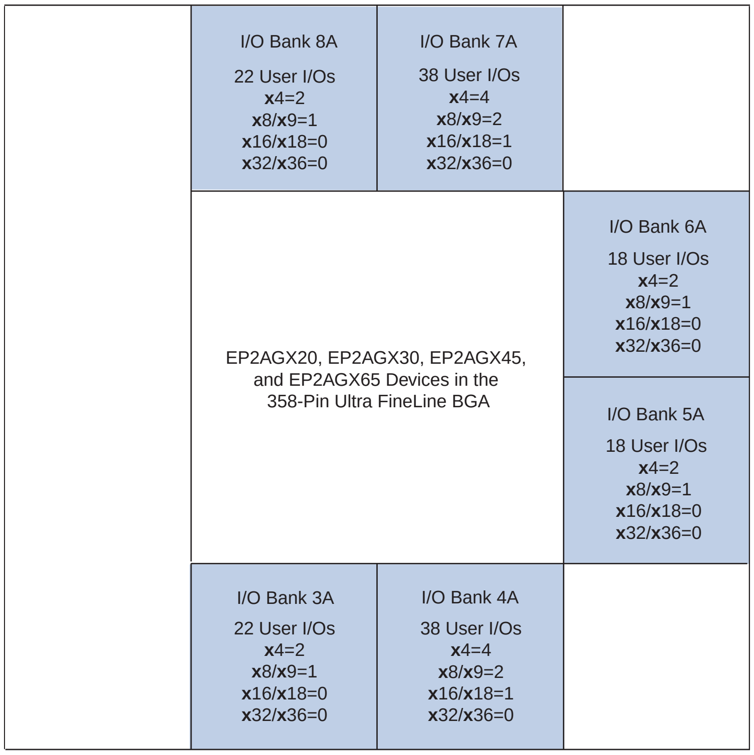
Device	Package	Side	I/O Module *(1)*	DQS/DQ Groups			
				×4	×8/×9	×16/×18	×32/×36
EP2AGX190 EP2AGX260	1152-Pin FineLine BGA	Top/Bottom/ Right	12	24	12	6	2

Note to Table 7–4:

(1) Each I/O module consists of 16 I/O pins. Twelve of the 16 pins are DQ/DSQ pins.

Figure 7–3 shows the number of DQS/DQ groups per bank in EP2AGX20, EP2AGX30, EP2AGX45, and EP2AGX65 devices in the 358-pin Ultra FineLine BGA package.

Figure 7–3. Number of DQS/DQ Groups per Bank in EP2AGX20, EP2AGX30, EP2AGX45, and EP2AGX65 Devices in the 358-Pin Ultra Fineline BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*

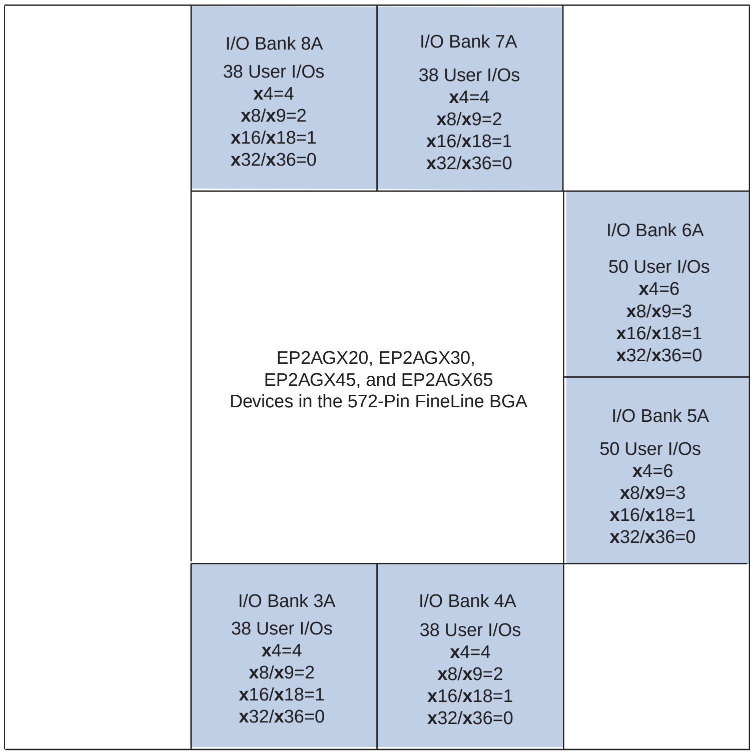


Notes to Figure 7–3:

(1) These numbers are preliminary until the devices are available.
(2) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.
(3) Several configuration pins in Bank 6A are shared with DQS/DQ pins. You cannot use a x4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups you have chosen are not also used for configuration.
(4) Arria II GX devices in the 358-pin Ultra FineLine BGA Package do not support x36 QDRII SRAM interface because x36 emulation mode (combining two x18 DQS/DQ groups to form a single x36 DQS/DQ group) is not supported.

Figure 7–4 shows the number of DQS/DQ groups per bank in Arria II GX EP2AGX20, EP2AGX30, EP2AGX45, and EP2AGX65 devices in the 572-pin FineLine BGA package.

Figure 7–4. Number of DQS/DQ Groups per Bank in EP2AGX20, EP2AGX30, EP2AGX45, and EP2AGX65 Devices in the 572-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*



Notes to Figure 7–4:

(1) These numbers are preliminary until the devices are available.

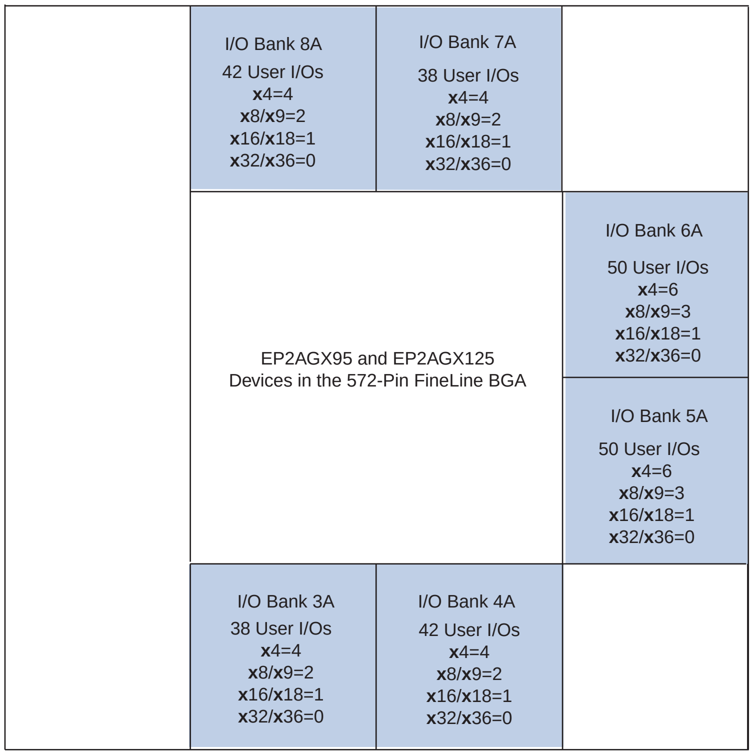
(2) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(3) Several configuration pins in Bank 6A are shared with DQS/DQ pins. You cannot use a x4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups you have chosen are not also used for configuration.

(4) Arria II GX devices in the 572-pin FineLine BGA Package do not support x36 QDRII SRAM interface because x36 emulation mode (combining two x18 DQS/DQ groups to form a single x36 DQS/DQ group) is not supported.

Figure 7–5 shows the number of DQS/DQ groups per bank in Arria II GX EP2AGX95 and EP2AGX125 devices in the 572-pin FineLine BGA package.

Figure 7–5. Number of DQS/DQ Groups per Bank in EP2AGX95 and EP2AGX125 Devices in the 572-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*



Notes to Figure 7–5:

(1) These numbers are preliminary until the devices are available.

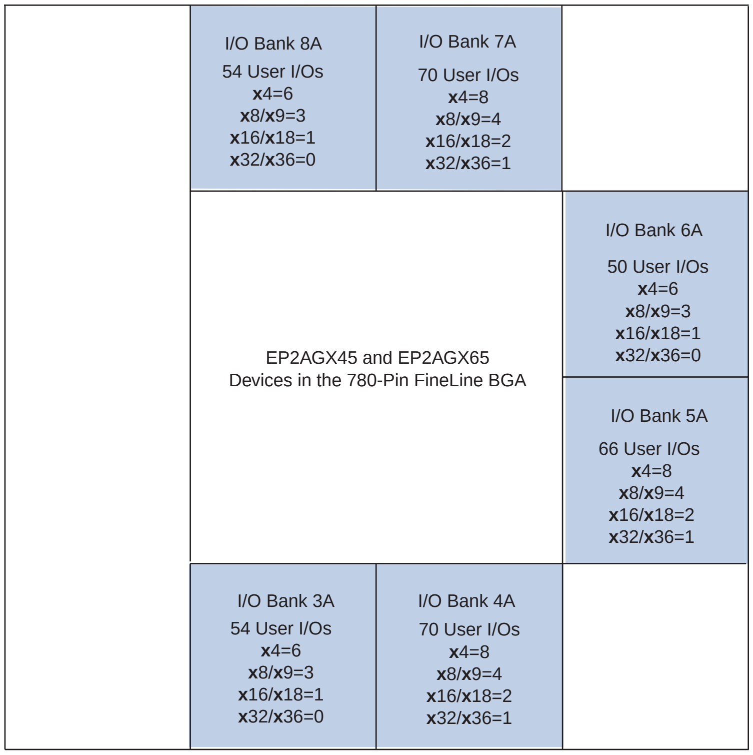
(2) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(3) Several configuration pins in Bank 6A are shared with DQS/DQ pins. You cannot use a x4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups you have chosen are not also used for configuration.

(4) Arria II GX devices in the 572-pin FineLine BGA Package do not support x36 QDRII SRAM interface because x36 emulation mode (combining two x18 DQS/DQ groups to form a single x36 DQS/DQ group) is not supporte

Figure 7–6 shows the number of DQS/DQ groups per bank in Arria II GX EP2AGX45 and EP2AGX65 devices in the 780-pin FineLine BGA package.

Figure 7–6. Number of DQS/DQ Groups per Bank in EP2AGX45 and EP2AGX65 Devices in the 780-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*



Notes to Figure 7–6:

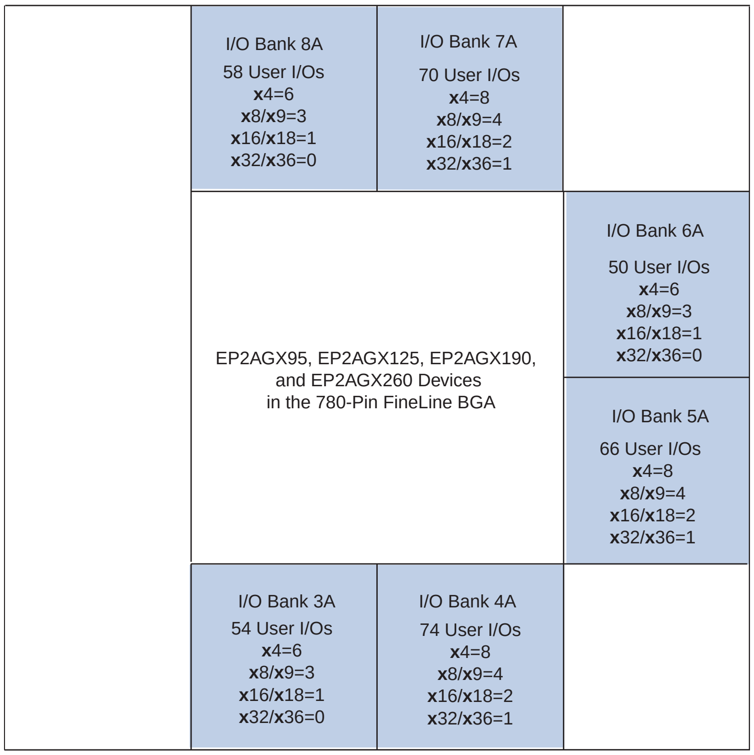
(1) These numbers are preliminary until the devices are available.

(2) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(3) Several configuration pins in Bank 6A are shared with DQS/DQ pins. You cannot use a x4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups you have chosen are not also used for configuration.

Figure 7–7 shows the number of DQS/DQ groups per bank in Arria II GX EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260 devices in the 780-pin FineLine BGA package.

Figure 7–7. Number of DQS/DQ Groups per Bank in EP2AGX95, EP2AGX125, EP2AGX190 and EP2AGX260 Devices in the 780-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*



Notes to Figure 7–7:

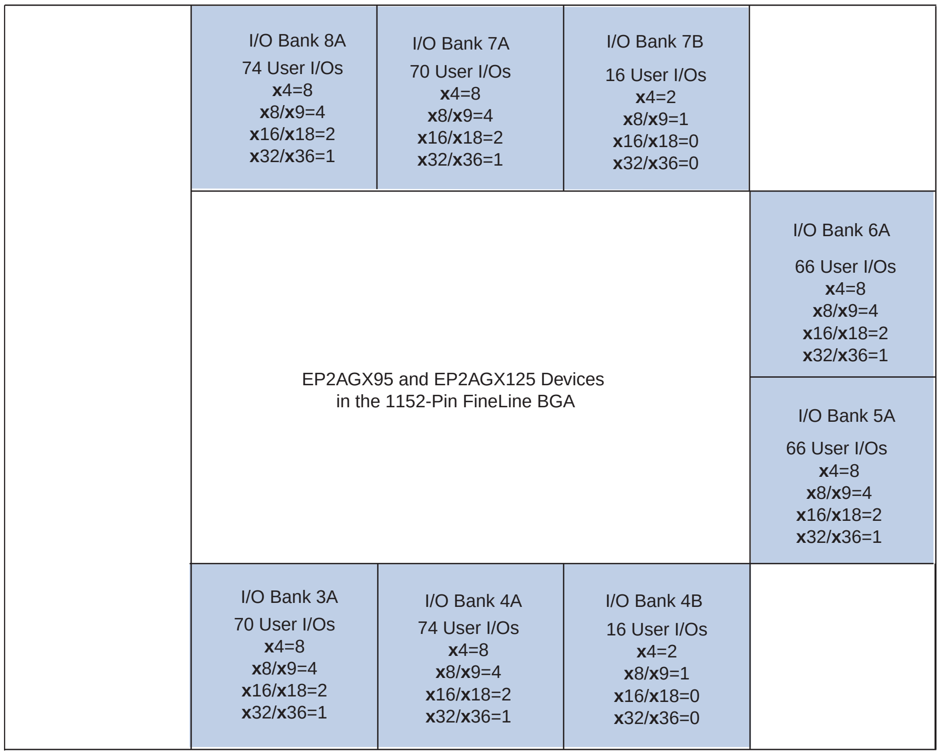
(1) These numbers are preliminary until the devices are available.

(2) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(3) Several configuration pins in Bank 6A are shared with DQS/DQ pins. You cannot use a x4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups you have chosen are not also used for configuration.

Figure 7–8 shows the number of DQS/DQ groups per bank in Arria II GX EP2AGX95 and EP2AGX125 devices in the 1152-pin FineLine BGA package.

Figure 7–8. Number of DQS/DQ Groups per Bank in EP2AGX95 and EP2AGX125 Devices in the 1152-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*



Notes to Figure 7–8:

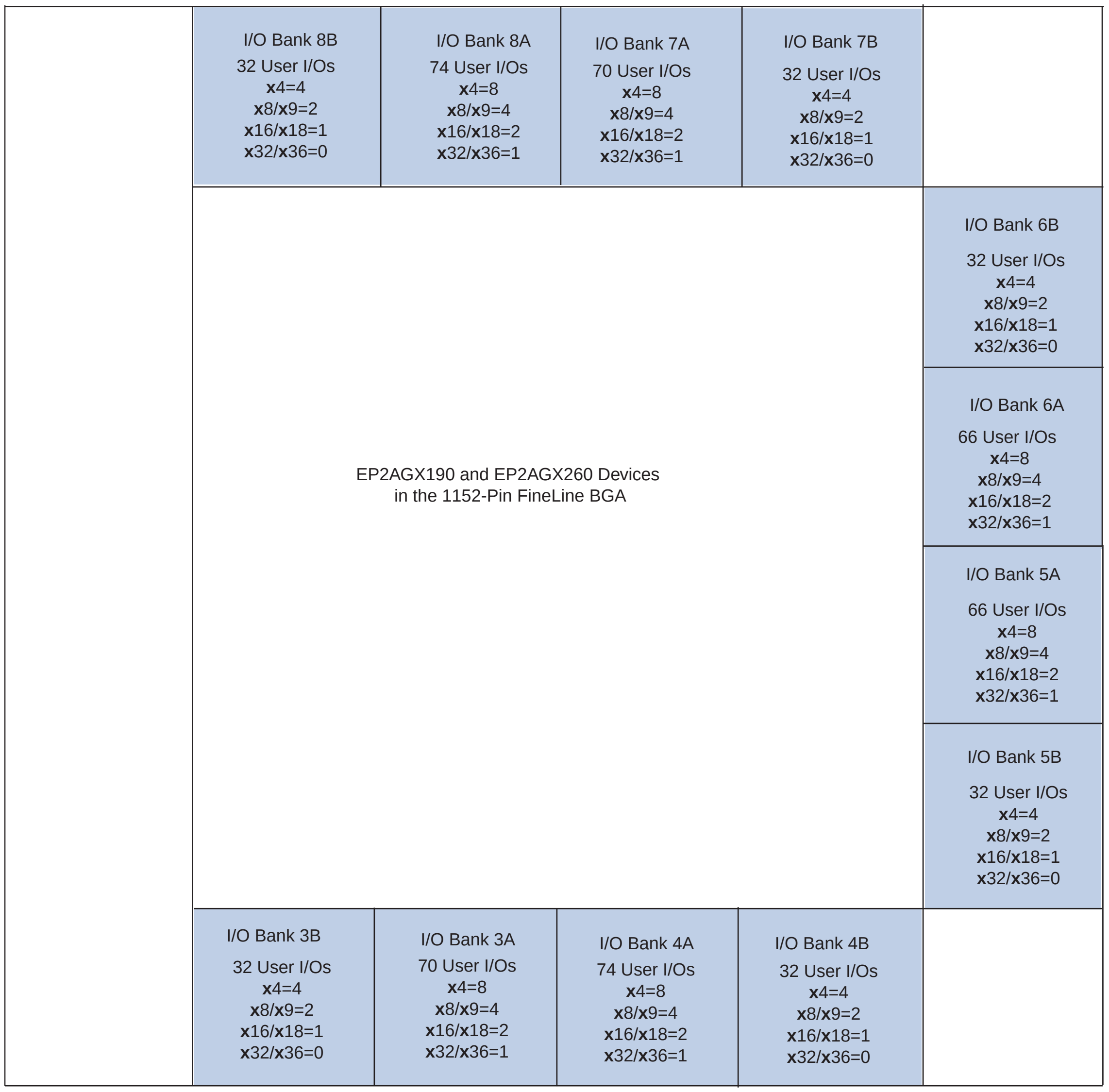
(1) These numbers are preliminary until the devices are available.

(2) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(3) Several configuration pins in Bank 6A are shared with DQS/DQ pins. You cannot use a x4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups you have chosen are not also used for configuration.

Figure 7–9 shows the number of DQS/DQ groups per bank in Arria II GX EP2AGX190 and EP2AGX260 devices in the 1152-pin FineLine BGA package.

Figure 7–9. Number of DQS/DQ Groups per Bank in EP2AGX190 and EP2AGX260 Devices in the 1152-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*



Notes to Figure 7–9:

(1) These numbers are preliminary until the devices are available.

(2) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(3) Several configuration pins in Bank 6A are shared with DQS/DQ pins. You cannot use a x4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups you have chosen are not also used for configuration.

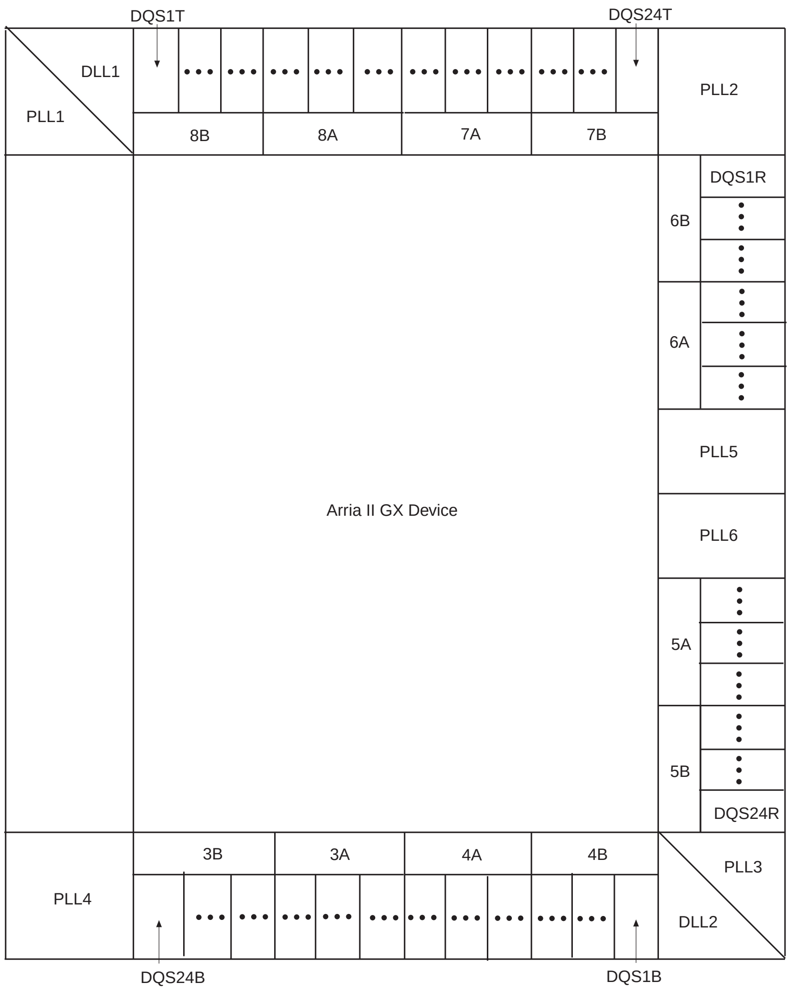
The DQS and DQSn pins are listed in the Arria II GX pin tables as `DQSXY` and `DQSnXY`, respectively, where `X` denotes the DQS/DQ grouping number and `Y` denotes whether the group is located on the top (T), bottom (B), or right (R) side of the device. The DQS/DQ pin numbering is based on ×4 mode.

The corresponding DQ pins are marked as DQXY, where X indicates to which DQS group the pins belong to and Y indicates whether the group is located on the top (T), bottom (B), or right (R) side of the device. For example, DQS3B indicates a DQS pin that is located on the bottom side of the device. The DQ pins belonging to that group are shown as DQ3B in the pin table. See Figure 7–10 for illustrations.

☞ The parity, DM, BWSn, and ECC pins are shown as DQ pins in the pin table.

The numbering scheme starts from the top-left side of the device going clockwise in a die-top view. Figure 7–10 shows how the DQS/DQ groups are numbered in a die-top view of the largest Arria II GX device.

Figure 7–10. DQS Pins in Arria II GX I/O Banks for EP2AGX260 with the F1152 Package



Arria II GX External Memory Interface Features

Arria II GX devices are rich with features that allow robust high-performance external memory interfacing. The ALTMEMPHY megafunction allows you to use these external memory interface features and helps set up the physical interface (PHY) best suited for your system. This section describes each Arria II GX device feature that is used in external memory interfaces from the DQS phase-shift circuitry and DQS logic block.

When you use the Altera memory controller MegaCore functions, the ALTMEMPHY megafunction is instantiated for you.

Table 7–5 shows the Quartus II megafunction/MegaCore support for the Arria II GX external memory interface.

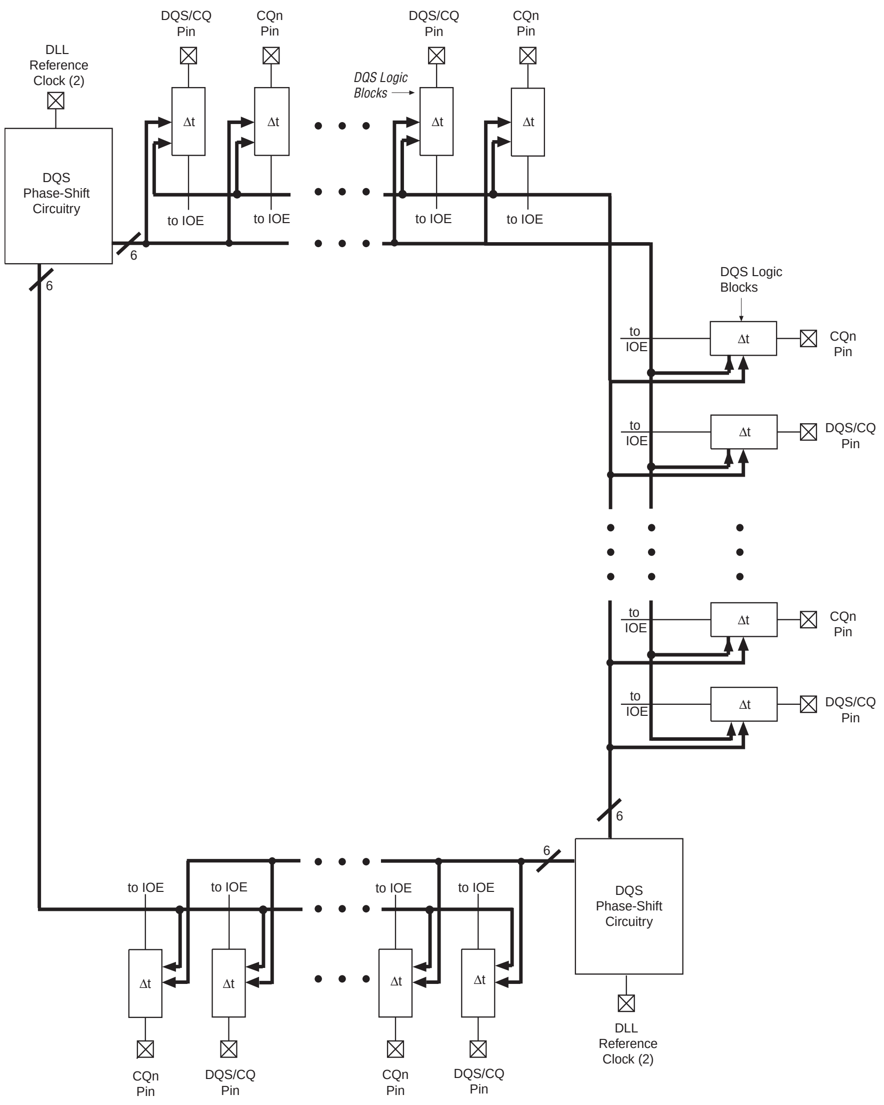
Table 7–5. Megafunction/MegaCore Support for the Arria II GX External Memory Interface

Memory Standards	Megafunction/IP Support				
	High-Performance Controller Full Data Rate	High-Performance Controller Half Data Rate	ALTMEMPHY Full Data Rate	ALTMEMPHY Half Data Rate	ALTDLL and ALTDQ_DQS Megafunctions
DDR SDRAM	✓	✓	✓	✓	✓
DDR2 SDRAM	✓	✓	✓	✓	✓
DDR3 SDRAM	—	✓	—	✓	✓
QDRII SRAM	—	—	—	—	✓

DQS Phase-Shift Circuitry

Arria II GX phase-shift circuitry provides phase shift to the DQS/CQ and CQn pins on read transactions when the DQS/CQ and CQn pins are acting as input clocks or strobes to the FPGA. DQS phase-shift circuitry consists of DLLs that are shared between multiple DQS pins and the phase-offset control module to further fine-tune the DQS phase shift for different sides of the device. Figure 7–11 shows how the DQS phase-shift circuitry is connected to the DQS/CQ and CQn pins in the device where memory interfaces are supported on the top, bottom, and right sides of the Arria II GX device.

Figure 7–11. DQS/CQ and CQn Pins and DQS Phase-Shift Circuitry *(Note 1)*



Notes to Figure 7–11:

(1) Refer to Table 7–6 on page 7–17 for possible reference input clock pins for each DLL.

(2) You can configure each DQS/CQ and CQn pin with a phase shift based on one of two possible DLL output settings.

DQS phase-shift circuitry is connected to DQS logic blocks that control each DQS/CQ or CQn pin. The DQS logic blocks allow the DQS delay settings to be updated concurrently at every DQS/CQ or CQn pin.

DLL

DQS phase-shift circuitry uses a DLL to dynamically control the clock delay needed by the DQS/CQ and CQn pins. The DLL, in turn, uses a frequency reference to dynamically generate control signals for the delay chains in each of the DQS/CQ and CQn pins, allowing it to compensate for PVT variations. The DQS delay settings are Gray-coded to reduce jitter when the DLL updates the settings. Phase-shift circuitry needs a maximum of 1280 clock cycles to lock and calculate the correct input clock period. Do not send data during these clock cycles because there is no guarantee that it will be captured properly. As the settings from the DLL may not be stable until this lock period has elapsed, be aware that anything using these settings may be unstable during this period.

You can still use the DQS phase-shift circuitry for any memory interfaces that are operating at less than 100 MHz. However, the DQS signal may not shift over 2.5 ns. At less than 100 MHz, while the DQS phase shift may not be exactly centered to the data valid window, sufficient margin needs to still exist for reliable operation.

There are two DLLs in an Arria II GX device, located in the top-left and bottom-right corners of the device. These two DLLs can support a maximum of two unique frequencies, with each DLL running at one frequency. Each DLL can have two outputs with different phase offsets, which allows one Arria II GX device to have four different DLL phase-shift settings.

Each DLL can access the top, bottom, and right side of the device. This means that each I/O bank is accessible by two DLLs, giving more flexibility to create multiple frequencies and multiple-type interfaces. The DLL outputs the same DQS delay settings for the different sides of the device.

Interfaces that span two sides of the device are not recommended for high-performance memory interface applications. However, Arria II GX devices support interfaces with DQ/DQS groups wrapping over column and row I/Os from adjacent sides of the devices. Interfaces with DQ/DQS groups from both top and bottom I/O banks are not supported.

Each bank can use settings from either one or both DLLs. For example, `DQS1R` can get its phase-shift settings from DLL1, while `DQS2R` can get its phase-shift settings from DLL2.

The reference clock for each DLL might come from PLL output clocks or dedicated clock input pins, as specified in Table 7–6.

When you have a dedicated PLL that only generates the DLL input reference clock, set the PLL mode to **No Compensation** or the Quartus II software changes it automatically. Because the PLL does not use any other outputs, it does not need to compensate for any clock paths.

Arria II GX devices support PLL cascading. When cascading PLLs, it is best to use PLLs adjacent to each other (for example, PLL5 and PLL6) so that the dedicated path between the two PLLs is used instead of using a GCLK or RCLK clock network that might be subjected to core noise. The TimeQuest timing analyzer takes PLL cascading into consideration for timing analysis.

Table 7–6. DLL Reference Clock Input *(Note 1)*

DLL	CLKIN (Top/Bottom)	CLKIN (Right)	PLL

DLL1	`CLK12`
	`CLK13`
	`CLK14`
	`CLK15`
		Not Available	PLL1

DLL2	`CLK4`
	`CLK5`
	`CLK6`
	`CLK7`
		`CLK8`
		`CLK9`
		`CLK10`
		`CLK11`
			PLL3

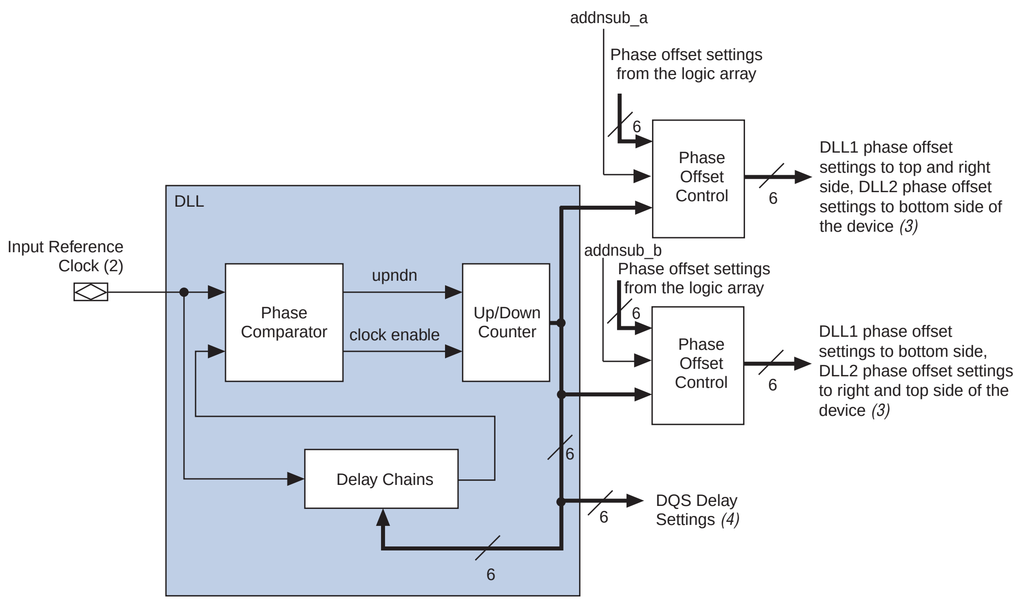
Note to Table 7–6:

(1) `CLK4` to `CLK7` are located on the bottom side, `CLK8` to `CLK11` are located on the right side, and `CLK12` to `CLK15` are located on the top side of the device.

When using the ALTMEMPHY megafunction, use the dedicated PLL input pin for the PLL reference clock.

Figure 7–12 shows a simple block diagram of the DQS Phase-Shift Circuitry. The input reference clock goes into the DLL to a chain of up to 16 delay elements. The phase comparator compares the signal coming out of the end of the delay chain block to the input reference clock. The phase comparator then issues the `upndn` signal to the Gray-coded counter. This signal increments or decrements a 6-bit delay setting (DQS delay settings) 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.

Figure 7–12. Simplified Diagram of the DQS Phase-Shift Circuitry *(Note 1)*


addnsub_a
Phase offset settings from the logic array
6
Phase Offset Control
6
DLL1 phase offset settings to top and right side, DLL2 phase offset settings to bottom side of the device (3)
DLL
Input Reference Clock (2)
Phase Comparator
upndn
clock enable
Up/Down Counter
addnsub_b
Phase offset settings from the logic array
6
Phase Offset Control
6
DLL1 phase offset settings to bottom side, DLL2 phase offset settings to right and top side of the device (3)
Delay Chains
6
6
DQS Delay Settings (4)
6

Notes to Figure 7–12:

(1) All features of the DQS phase-shift circuitry are accessible from the ALTMEMPHY megafunction in the Quartus II software.

(2) The input reference clock for the DQS phase-shift circuitry can come from a PLL output clock or an input clock pin. Refer to Table 7–6 and Table 7–7 for the exact PLL and input clock pin.

(3) Phase offset settings can only go to the DQS logic blocks.

(4) DQS delay settings can go to the logic array and DQS logic block.

You can reset the DLL from either the logic array or a user I/O pin. Each time the DLL is reset, you must wait for 1280 clock cycles for the DLL to lock before you can capture the data properly.

Depending on the DLL frequency mode, the DLL can shift the incoming DQS signals by 0°, 22.5°, 30°, 36°, 45°, 60°, 67.5°, 72°, 90°, 108°, 120°, 135°, 144°, or 180°. The shifted DQS signal is then used as the clock for the DQ IOE input registers.

All DQS/CQ and CQn pins, referenced to the same DLL, can have their input signal phase shifted by a different degree amount but all must be referenced at one particular frequency. For example, you can have a 90° phase shift on `DQS1T` and a 60° phase shift on `DQS2T`, referenced from a 200-MHz clock. Not all phase-shift combinations are supported. The phase shifts on the DQS pins referenced by the same DLL must all be a multiple of 22.5° (up to 90°), 30° (up to 120°), 36° (up to 144°), or 45° (up to 180°).

There are six different frequency modes for the Arria II GX DLL, as shown in Table 7–7. Each frequency mode provides different phase-shift selections. In frequency mode 0, 1, 2, and 3, the 6-bit DQS delay settings vary with PVT to implement the phase-shift delay. In frequency modes 4 and 5, only 5 bits of the DQS delay settings vary with PVT to implement the phase-shift delay; the most significant bit of the DQS delay setting is set to **0**.

Table 7–7. Arria II GX DLL Frequency Modes

Frequency Mode	Available Phase Shift	Number of Delay Chains
0	22.5, 45, 67.5, 90	16
1	30, 60, 90, 120	12
2	36, 72, 108, 144	10
3	45, 90, 135, 180	8
4	30, 60, 90, 120	12
5	36, 72, 108, 144	10

For the frequency range of each mode, refer to the *Device Data Sheet* chapter in volume 3 of the *Arria II GX Device Handbook*.

For 0° shift, the DQS/CQ signal bypasses both the DLL and DQS logic blocks. The Quartus II software automatically sets the DQ input delay chains so that the skew between the DQ and DQS/CQ pin at the DQ IOE registers is negligible when the 0° shift is implemented. You can feed the DQS delay settings to the DQS logic block and the logic array.

The shifted DQS/CQ signal goes to the DQS bus to clock the IOE input registers of the DQ pins. The signal can also go into the logic array for resynchronization if you are not using the IOE resynchronization registers. The shifted CQn signal can go to the negative-edge input register in the DQ IOE or the logic array and is only used for QDRII SRAM interfaces.

Phase Offset Control

Each DLL has two phase offset modules and can provide two separate DQS delay settings with independent offset; one offset goes clockwise half-way around the chip and the other goes counter-clockwise half-way around the chip. Even though you have independent phase offset control, the frequency of the interface using the same DLL has to be the same. Use the phase offset control module for making small shifts to the input signal and use the DQS phase-shift circuitry for larger signal shifts. For example, if the DLL only offers a multiple of 30° phase shift, but your interface needs a 67.5° phase shift on the DQS signal, you can use two delay chains in the DQS logic blocks to give you 60° phase shift and use the phase offset control feature to implement the extra 7.5° phase shift.

You can either use a static phase offset or a dynamic phase offset to implement the additional phase shift. The available additional phase shift is implemented in 2s: complement in Gray-code between settings –64 to +63 for frequency mode 0, 1, 2, and 3, and between settings –32 to +31 for frequency modes 4, 5, and 6. An additional bit indicates whether the setting has a positive or negative value. The settings are linear, each phase offset setting adds a delay amount.

For more information about the specified phase-shift settings, refer to the *Device Data Sheet* chapter in volume 3 of the *Arria II GX Device Handbook*.

The DQS phase shift is the sum of the DLL delay settings and the user-selected phase offset settings whose top setting is 64 for frequency modes 0, 1, 2, and 3; and 32 for frequency modes 4, 5, and 6. Therefore, the actual physical offset setting range is 64 or 32 subtracted by the DQS delay settings from the DLL.

☞ When you use this feature, monitor the DQS delay settings to know how many offsets you can add and subtract in the system. Note that the DQS delay settings output by the DLL are also Gray-coded

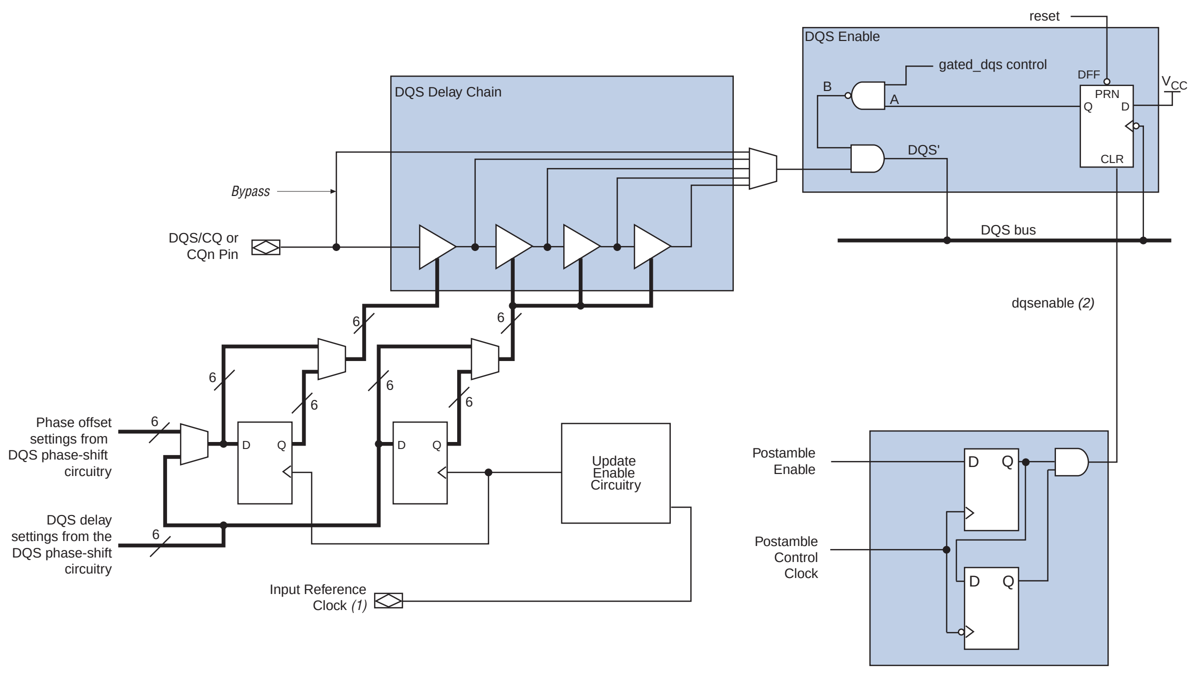
For example, if the DLL determines that DQS delay settings of 28 are needed to achieve a 30° phase shift in DLL frequency mode 1, you can subtract up to 28 phase offset settings and you can add up to 35 phase offset settings to achieve the optimal delay that you need. However, if the same DQS delay settings of 28 is needed to achieve 30° phase shift in DLL frequency mode 4, you can still subtract up to 28 phase offset settings, but you can only add up to 3 phase offset settings before the DQS delay settings reach their maximum settings because DLL frequency mode 4 only uses 5-bit DLL delay settings.

For more information about the value for each step, refer to the *Device Data Sheet* chapter in volume 3 of the *Arria II GX Device Handbook*.

DQS Logic Block

Each DQS/CQ and CQn pin is connected to a separate DQS logic block, which consists of DQS delay chains, update enable circuitry, and DQS postamble circuitry (see Figure 7–13).

Figure 7–13. Arria II GX DQS Logic Block



Notes to Figure 7–13:

(1) The input reference clock for the DQS phase-shift circuitry can come from a PLL output clock or an input clock pin. Refer to Table 7–6 and Table 7–7 for the exact PLL and input clock pin.

(2) The `dqsenable` signal can also come from the Arria II GX FPGA fabric.

DQS Delay Chain

DQS delay chains consist of a set of variable delay elements to allow the input DQS/CQ and CQn signals to be shifted by the amount specified by the DQS phase-shift circuitry or the logic array. There are four delay elements in the DQS delay chain; the first delay chain closest to the DQS/CQ or CQn pin can either be shifted by the DQS delay settings or by the sum of the DQS/CQ delay setting and the phase-offset setting. The number of delay chains required is transparent because the ALTMEMPHY megafunction automatically sets it when you choose the operating frequency. The DQS delay settings can come from the DQS phase-shift circuitry on either end of the I/O banks or from the logic array.

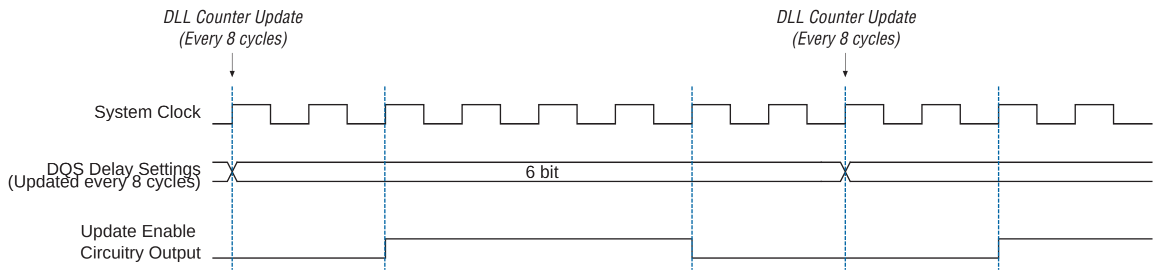
The delay elements in the DQS logic block have the same characteristics as the delay elements in the DLL. When the DLL is not used to control the DQS delay chains, you can input your own Gray-coded 6-bit or 5-bit settings using the `dqs_delayctrlin[5..0]` signals available in the ALTMEMPHY megafunction. These settings control 1, 2, 3, or all 4 delay elements in the DQS delay chains. The ALTMEMPHY megafunction can also dynamically choose the number of DQS delay chains needed for the system. The amount of delay is equal to the sum of the delay element's intrinsic delay and the product of the number of delay steps and the value of the delay steps.

You can also bypass the DQS delay chain to achieve 0° phase shift.

Update Enable Circuitry

Both the DQS delay settings and the phase-offset settings pass through a register before going into the DQS delay chains. The registers are controlled by the update enable circuitry to allow enough time for any changes in the DQS delay setting bits to arrive at all the delay elements. This allows them to be adjusted at the same time. The update enable circuitry enables the registers to allow enough time for the DQS delay settings to travel from the DQS phase-shift circuitry or core logic to all the DQS logic blocks before the next change. It uses the input reference clock or a user clock from the core to generate the update enable output. The ALTMEMPHY megafunction uses this circuit by default. Figure 7–14 shows an example waveform of the update enable circuitry output.

Figure 7–14. DQS Update Enable Waveform

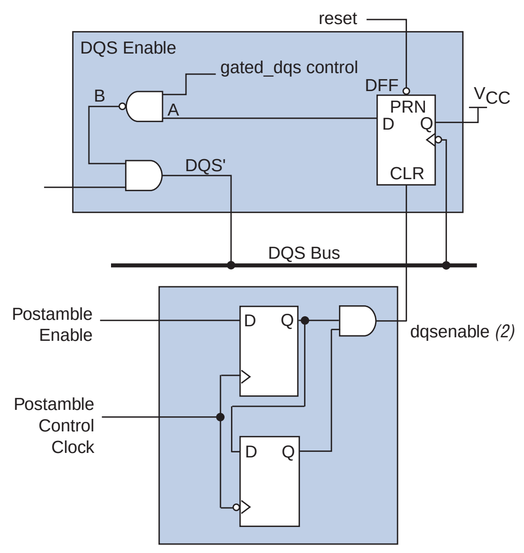


DQS Postamble Circuitry

For external memory interfaces that use a bidirectional read strobe such as in DDR3, DDR2, and DDR SDRAM, the DQS signal is low before going to or coming from a high-impedance state. The state in which DQS is low, just after a high-impedance state, is called the preamble; the state in which DQS is low, just before it returns to a high-impedance state, is called the postamble. There are preamble and postamble specifications for both read and write operations in DDR3, DDR2, and DDR SDRAM. The DQS postamble circuitry, featured in Figure 7–15, ensures that data is not lost if there is noise on the DQS line at the end of a read postamble time.

Arria II GX devices have dedicated postamble registers that can be controlled to ground the shifted DQS signal used to clock the DQ input registers at the end of a read operation. This ensures that any glitches on the DQS input signals at the end of the read postamble time do not affect the DQ IOE registers.

Figure 7–15. Arria II GX DQS Postamble Circuitry *(Note 1)*



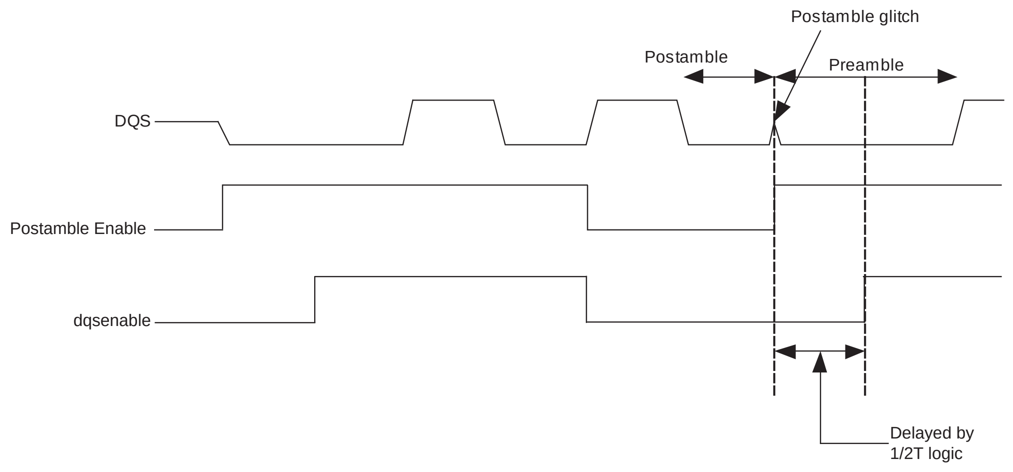
Notes to Figure 7–15:

(1) The postamble clock can come from any of the delayed resynchronization clock taps although it is not necessarily of the same phase as the resynchronization clock.

(2) The `dqsenable` signal can also come from the Arria II GX FPGA fabric.

There is an AND gate after the postamble register outputs that is used to avoid postamble glitches from a previous read burst on a non-consecutive read burst. This scheme allows a half-a-clock cycle latency for `dqsenable` assertion and zero latency for `dqsenable` de-assertion, as shown in Figure 7–16.

Figure 7–16. Avoiding Glitch on a Non-Consecutive Read Burst Waveform

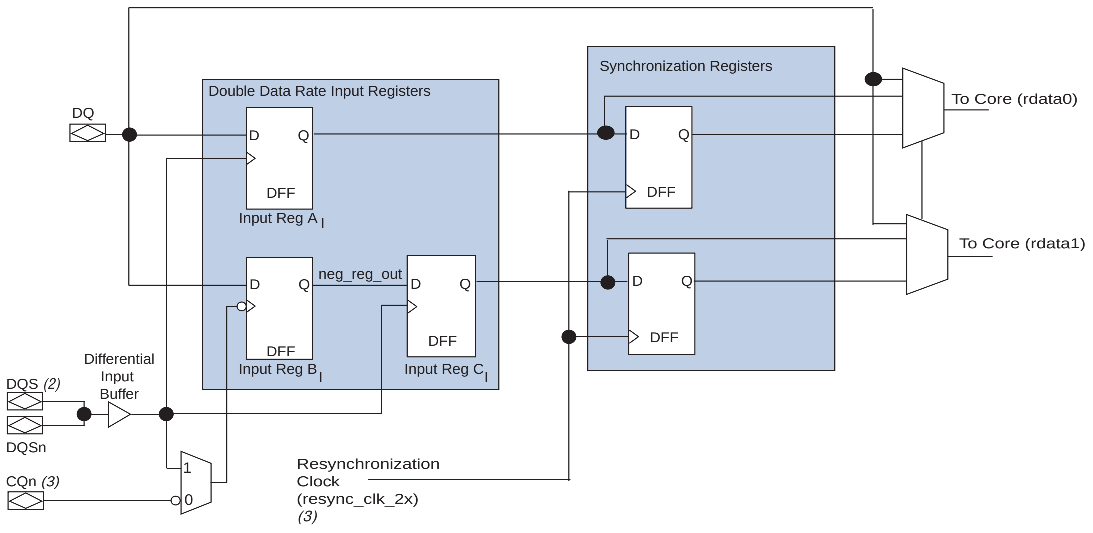


I/O Element Registers

IOE registers are expanded to allow source-synchronous systems to have faster register-to-register transfers and resynchronization. Both top, bottom, and right IOEs have the same capability. Right IOEs have extra features to support LVDS data transfer.

Figure 7–17 shows the registers available in the Arria II GX input path. The input path consists of DDR input registers and resynchronization registers. You can bypass each block of the input path.

Figure 7–17. Arria II GX IOE Input Registers *(Note 1)*


Synchronization Registers
Double Data Rate Input Registers
DQ
D
Q
DFF
Input Reg A_I
To Core (rdata0)
To Core (rdata1)
neg_reg_out
Input Reg B_I
Input Reg C_I
Differential Input Buffer
DQS (2)
DQSn
CQn (3)
1
0
Resynchronization Clock (resync_clk_2x) (3)

Notes to Figure 7–17:

(1) You can bypass each register block in this path.

(2) The input clock can be from the DQS logic block (whether the postamble circuitry is bypassed or not) or from a global clock line.

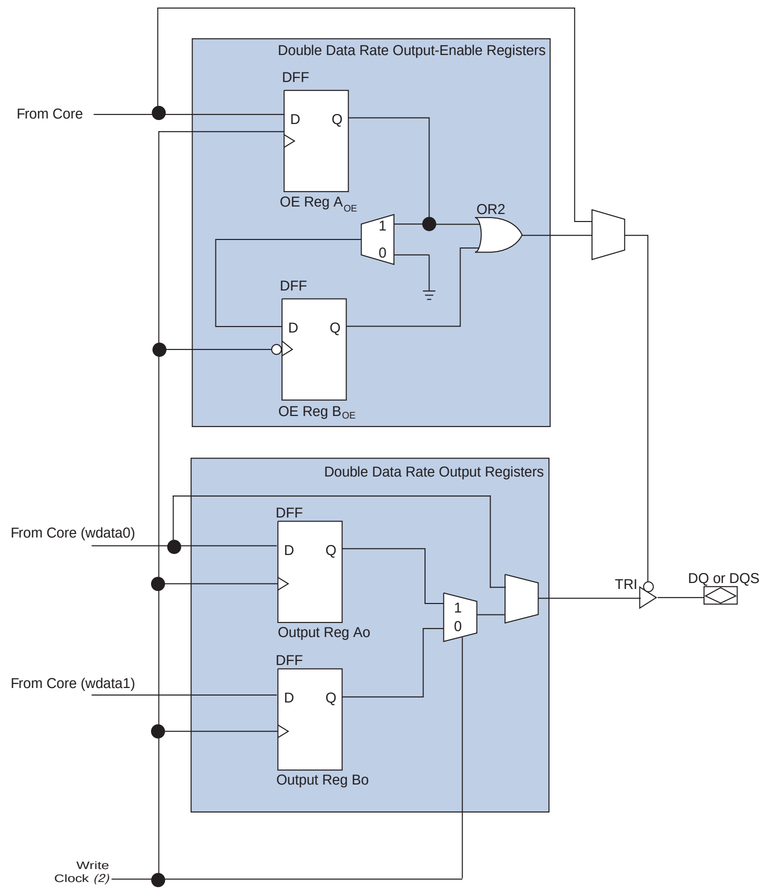
(3) This input clock comes from the CQn logic block.

There are three registers in the DDR input registers block. Two registers capture data on the positive and negative edges of the clock, while the third register aligns the captured data. You can choose to use the same clock for the positive edge and negative edge registers, or two complementary clocks (DQS/CQ for positive-edge register and DQSn/CQn for negative-edge register). The third register that aligns the captured data uses the same clock as the positive edge registers.

The resynchronization registers resynchronize the data to the resynchronization clock domain. These registers are clocked by the resynchronization clock that is generated by the PLL. The outputs of the resynchronization registers go straight to the core.

Figure 7–18 shows the registers available in the Arria II GX output and output-enable paths. The device can bypass each block of the output and output-enable path.

Figure 7–18. Arria II GX IOE Output and Output-Enable Path Registers *(Note 1)*



Notes to Figure 7–18:

(1) You can bypass each register block of the output and output-enable paths.

(2) The write clock comes from the PLL. The DQ write clock and DQS write clock have a 90° offset between them.

The output path is designed to route combinatorial or registered SDR outputs and DDR outputs from the FPGA core.

The output-enable path has a structure similar to the output path. You can have a combinatorial or registered output in SDR applications.

Revision History

Table 7–8 shows the revision history for this document.

Table 7–8. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial release.	—

8. High-Speed Differential I/O Interfaces and DPA in Arria II GX Devices

AIIGX51008-1.0

Introduction

This chapter describes the high-speed differential I/O features and resources, and the functionality of the serializer/deserializer (SERDES) and dynamic phase alignment (DPA) circuitry in Arria® II GX devices. The new modular I/O architecture in Arria II GX devices allows for high-speed LVDS interface on the top, bottom, and right sides of the device. The left side of the device is occupied by high-speed transceiver blocks. Dedicated SERDES and DPA circuitry are implemented on the right side of the device to further enhance LVDS interface performance in the device.

This chapter contains the following sections:

The Arria II GX device family has the following dedicated circuitry for high-speed differential I/O support:

- Differential I/O buffer
- Transmitter serializer
- Receiver deserializer
- Data realignment
- DPA
- Synchronizer (FIFO buffer)
- Phase-locked loops (PLLs)

The Arria II GX family supports the following differential I/O standards:

- LVDS
- mini-LVDS
- Reduced swing differential signaling (RSDS)
- Low-voltage positive emitter-coupled logic (LVPECL)
- Bus LVDS (BLVDS)

Dedicated mini-LVDS and RSDS inputs are not supported. The LVPECL I/O standard is only for PLL clock inputs in differential mode.

Table 8–1 shows the data rate range supported by LVDS, mini-LVDS, and RSDS.

Table 8–1. Supported Data Rate Range *(Note 1)*

I/O Standards	LVDS	mini-LVDS	RSDS
Data Rate Range (Mbps)	150-1000 *(2)*	150-400	150-360

Note to Table 8–1:

(1) The maximum data rate supported subject to silicon characterization.

(2) Dedicated SERDES and DPA circuitry on the right side of the device are required to achieve LVDS data rate at 1000 Mbps.

For specifications and features of the differential I/O standards supported in Arria II GX devices, refer to the *I/O Features in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

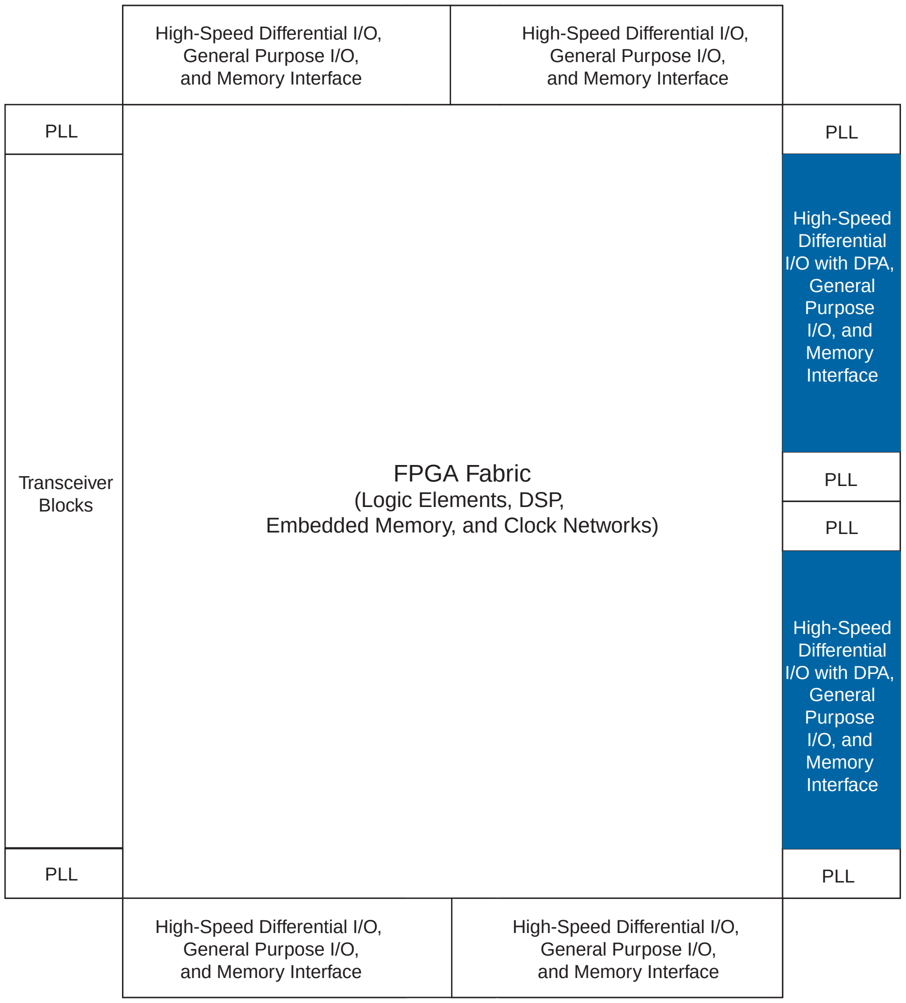
For specifications of the differential I/O standards supported in Arria II GX devices, refer to the *Device Data Sheet* chapter in volume 3 of the *Arria II GX Device Handbook*.

LVDS Channels

In Arria II GX devices, there are dedicated LVDS input buffers and LVDS I/O buffers at the top, bottom, and right side of the device. The LVDS input buffers have 100-Ω differential on-chip termination (OCT R_D) support. You can configure the LVDS I/O buffers as either LVDS input (without OCT R_D) or dedicated LVDS output buffers. Alternatively, you can configure the LVDS pins on the top, bottom, and right sides of the device, as pseudo-LVDS output buffers, which use two single-ended output buffers with an external resistor network to support LVDS, mini-LVDS and RSDS standards. Figure 8–1 shows a high-level chip overview of Arria II GX devices. The left side of the device is occupied by high-speed transceiver blocks.

For more details about I/O bank information, refer to the *I/O Features in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

Figure 8–1. High-Speed Differential I/Os with DPA Locations in an Arria II GX Device *(Note 1)*, *(2)*, *(3)*


High-Speed Differential I/O, General Purpose I/O, and Memory Interface
High-Speed Differential I/O, General Purpose I/O, and Memory Interface
PLL
PLL
High-Speed Differential I/O with DPA, General Purpose I/O, and Memory Interface
Transceiver Blocks
FPGA Fabric (Logic Elements, DSP, Embedded Memory, and Clock Networks)
PLL
PLL
High-Speed Differential I/O with DPA, General Purpose I/O, and Memory Interface
PLL
PLL
High-Speed Differential I/O, General Purpose I/O, and Memory Interface
High-Speed Differential I/O, General Purpose I/O, and Memory Interface

Notes to Figure 8–1:

(1) This is a top view of the silicon die, which corresponds to a reverse view for flip chip packages. It is a graphical representation only.

(2) Applicable to EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260 devices.

(3) There are no center PLLs on the right I/O banks for EP2AGX20, EP2AGX30, EP2AGX45, and EP2AGX65 devices.

Table 8–2 and Table 8–3 show the maximum number of row and column LVDS I/Os supported in Arria II GX devices. You can design the LVDS I/Os as dedicated LVDS input, output buffers or pseudo-LVDS output buffers, as long as the combination does not exceed the maximum count. For example, there are a total of 56 LVDS pairs of I/Os in 780-pin EP2AGX45 device row (refer to Table 8–2). You can design up to a maximum of:

- 28 dedicated LVDS input buffers with OCT R_D and 28 dedicated LVDS output buffers, or
- 56 LVDS input buffers of which 28 are dedicated LVDS input buffers with OCT RD and 28 need external 100-Ω termination, or
- 28 dedicated LVDS output buffers and 28 emulated LVDS output buffers, or
- 56 emulated LVDS output buffers.

Table 8–2. LVDS Channels Supported in Arria II GX Device Row I/O Banks *(Note 1)*, *(2)*, *(3)*, *(4)*

Device	358-Pin FlipChip UBGA	572-Pin FlipChip FBGA	780-Pin FlipChip FBGA	1152-Pin FlipChip FBGA
EP2AGX20	8Rx or pTx + 8Tx or pTx	24Rx or pTx + 24Tx or pTx	—	—
EP2AGX30	8Rx or pTx + 8Tx or pTx	24Rx or pTx + 24Tx or pTx	—	—
EP2AGX45	8Rx or pTx + 8Tx or pTx	24Rx or pTx + 24Tx or pTx	28Rx or pTx + 28Tx or pTx	—
EP2AGX65	8Rx or pTx + 8Tx or pTx	24Rx or pTx + 24Tx or pTx	28Rx or pTx + 28Tx or pTx	—
EP2AGX95	—	24Rx or pTx + 24Tx or pTx	28Rx or pTx + 28Tx or pTx	32Rx or pTx + 32Tx or pTx
EP2AGX125	—	24Rx or pTx + 24Tx or pTx	28Rx or pTx + 28Tx or pTx	32Rx or pTx + 32Tx or pTx
EP2AGX190	—	—	28Rx or pTx + 28Tx or pTx	48Rx or pTx + 48Tx or pTx
EP2AGX260	—	—	28Rx or pTx + 28Tx or pTx	48Rx or pTx + 48Tx or pTx

Notes to Table 8–2:

(1) Rx = dedicated LVDS input buffers with OCT R_D support.
(2) Tx = dedicated LVDS output buffers or LVDS input buffers without OCT R_D support.
(3) pTx = pseudo-LVDS output buffers, either LVDS_E_3R or LVDS_E_1R.
(4) The LVDS channel count does not include dedicated clock input pins and PLL clock output pins.

Table 8–3. LVDS Channels Supported in Arria II GX Device Column I/O Banks *(Note 1)*, *(2)*, *(3)*, *(4)*

Device	358-Pin FlipChip UBGA	572-Pin FlipChip FBGA	780-Pin FlipChip FBGA	1152-Pin FlipChip FBGA
EP2AGX20	25Rx or pTx + 24Tx or pTx	33Rx or pTx + 32Tx or pTx	—	—
EP2AGX30	25Rx or pTx + 24Tx or pTx	33Rx or pTx + 32Tx or pTx	—	—
EP2AGX45	25Rx or pTx + 24Tx or pTx	33Rx or pTx + 32Tx or pTx	57Rx or pTx + 56Tx or pTx	—
EP2AGX65	25Rx or pTx + 24Tx or pTx	33Rx or pTx + 32Tx or pTx	57Rx or pTx + 56Tx or pTx	—
EP2AGX95	—	33Rx or pTx + 32Tx or pTx	57Rx or pTx + 56Tx or pTx	73Rx or pTx + 72Tx or pTx
EP2AGX125	—	33Rx or pTx + 32Tx or pTx	57Rx or pTx + 56Tx or pTx	73Rx or pTx + 72Tx or pTx
EP2AGX190	—	—	57Rx or pTx + 56Tx or pTx	97Rx or pTx + 96Tx or pTx
EP2AGX260	—	—	57Rx or pTx + 56Tx or pTx	97Rx or pTx + 96Tx or pTx

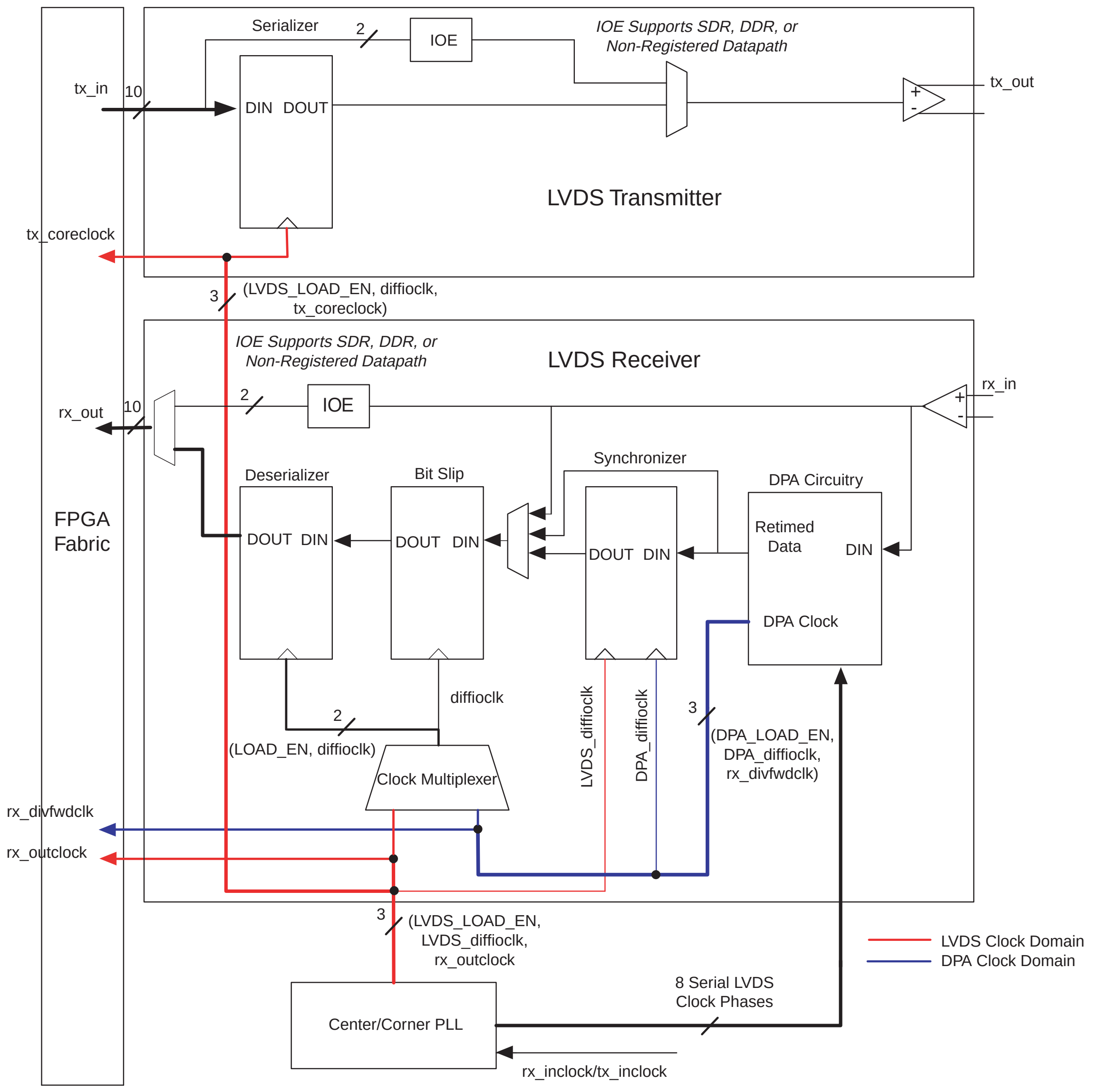
Notes to Table 8–3:

(1) Rx = dedicated LVDS input buffers with OCT R_D support.
(2) Tx = dedicated LVDS output buffers or LVDS input buffers without OCT R_D support.
(3) pTx = pseudo-LVDS output buffers, either LVDS_E_3R or LVDS_E_1R.
(4) The LVDS channel count does not include dedicated clock input pins and PLL clock output pins.

LVDS SERDES and DPA Block Diagram

The Arria II GX device family has dedicated SERDES and DPA circuitry for LVDS transmitters and receivers on the right side of the device. Figure 8–2 shows the LVDS SERDES and DPA block diagram. This diagram shows the interface signals for the transmitter and receiver data paths. For more information, refer to "Differential Transmitter" on page 8–6 and "Differential Receiver" on page 8–8.

Figure 8–2. LVDS SERDES and DPA Block Diagram *(Note 1), (2), (3)*



Notes to Figure 8–2:

(1) This diagram shows a shared PLL between the transmitter and receiver. If the transmitter and receiver are not sharing the same PLL, two PLLs on the right side of the device are required.

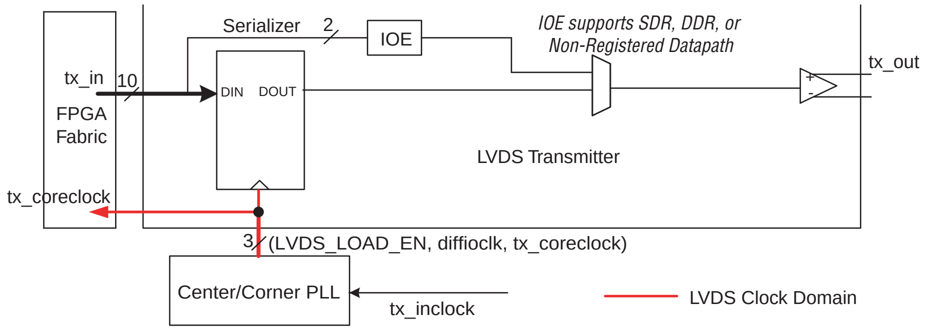
(2) In SDR and DDR mode, the data width is 1 and 2, respectively.

(3) The `tx_in` and `rx_out` ports have a maximum data width of 10.

Differential Transmitter

The serializer takes parallel data up to 10-bits wide from the FPGA fabric and converts the parallel data to serial data before sending the serial data to the differential output buffer. The differential output buffer supports programmable pre-emphasis and programmable V_{OD} controls, and can drive out mini-LVDS and RSDS signaling levels. Figure 8–3 is a block diagram of the LVDS transmitter.

Figure 8–3. Arria II GX LVDS Transmitter Block Diagram *(Note 1)*, *(2)*



Notes to Figure 8–3:

(1) In SDR and DDR mode, the data width is 1 and 2, respectively.

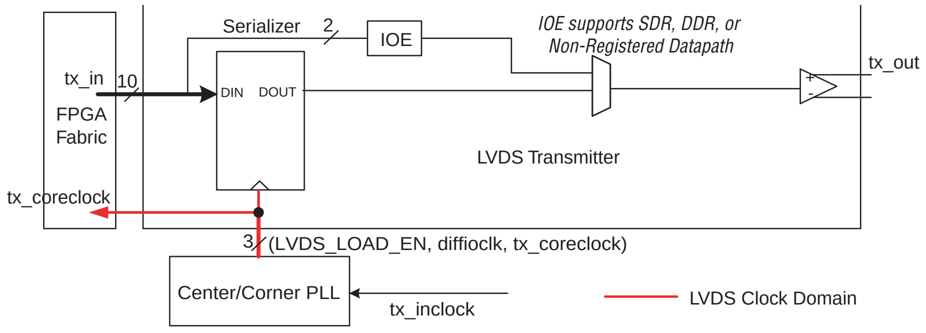
(2) The `tx_in` port has a maximum data width of 10.

Serializer

The serializer takes parallel data from the FPGA fabric, clocks it into the parallel load registers, and serializes it using the shift registers before sending the data to the differential output buffer. The MSB of the parallel data is transmitted first. The parallel load and shift registers are clocked by the high-speed clock running at the serial data rate (`diffioclk`) and controlled by the load enable signal (`LVDS_LOAD_EN`) generated from the PLL. You can statically set the serialization factor to ×4, ×6, ×7, ×8, or ×10 using the ALTLVDS megafunction. The load enable signal is derived from the serialization factor setting.

The serializer can be bypassed to support DDR (×2) and SDR (×1) operations to achieve a serialization factor of 2 and 1, respectively. The I/O element (IOE) contains two data output registers that can each operate in either DDR or SDR mode. Figure 8–4 shows the serializer bypass path.

Figure 8–4. Arria II GX Serializer Bypass *(Note 1), (2), (3)*



Notes to Figure 8–4:

(1) All disabled blocks and signals are grayed out.

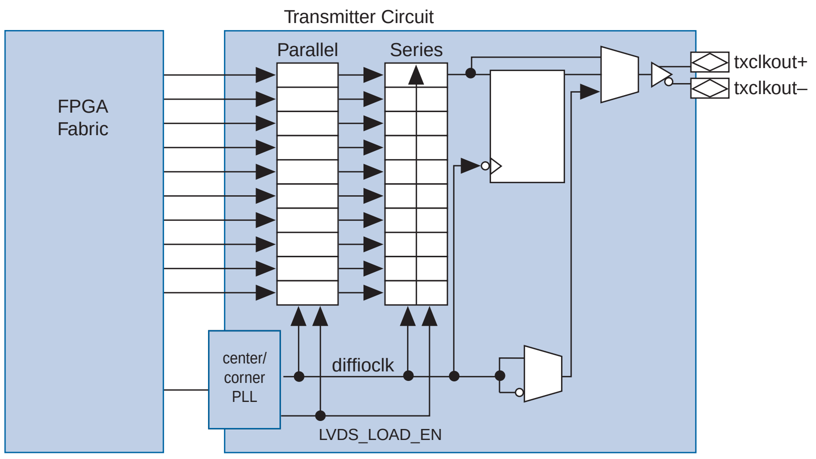
(2) In DDR mode, `tx_inclock` clocks the IOE register. In SDR mode, data is directly passed through the IOE.

(3) In SDR and DDR mode, the data width to the IOE is 1 and 2, respectively.

Differential applications often require specific clock-to-data alignments or specific data rate to clock rate factors. You can configure any Arria II GX LVDS transmitter to generate a source-synchronous transmitter clock output. This flexibility allows the placement of the output clock near the data outputs to simplify board layout and reduce clock-to-data skew. The output clock can also be divided by a factor of 1, 2, 4, 6, 8, or 10, depending on the serialization factor. The phase of the clock in relation to the data can be set at 0° or 180° (edge or center aligned). The center and corner PLLs provide additional support for other phase shifts in 45° increments.

Figure 8–5 shows the Arria II GX LVDS transmitter in clock output mode. In clock output mode, you can use an LVDS data channel as a clock output channel.

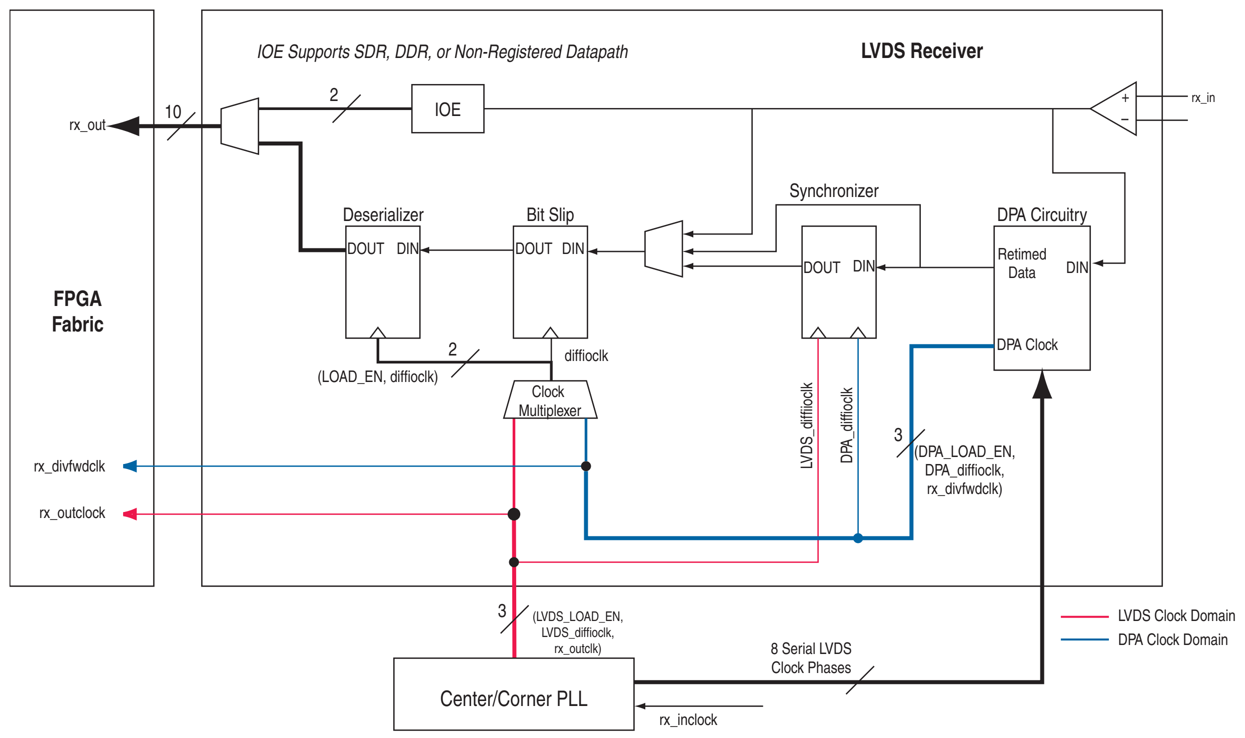
Figure 8–5. Arria II GX LVDS Transmitter in Clock Output Mode



Differential Receiver

Figure 8–6 shows a block diagram of an LVDS receiver in the right I/O bank.

Figure 8–6. LVDS Receiver Block Diagram *(Note 1)*, *(2)*



Notes to Table 8–6:

(1) In SDR and DDR mode, the data width from the IOE is 1 and 2, respectively.

(2) The `rx_out` port has a maximum data width of 10.

The center/corner PLL receives the external reference clock input (`rx_inclock`) and generates eight different phases of the same clock. The DPA block chooses one of the eight clock phases from the center/corner PLL and aligns to the incoming data to maximize receiver skew margin. The synchronizer circuit is a 1-bit wide by 6-bit deep FIFO buffer that compensates for any phase difference between the DPA block and the deserializer. If necessary, the user-controlled data realignment circuitry inserts a single bit of latency in the serial bit stream to align to the word boundary. The deserializer converts the serial data to parallel data, and sends the parallel data to the FPGA fabric.

The physical medium connecting the LVDS transmitter and the receiver channels may introduce skew between the serial data and the source synchronous clock. The instantaneous skew between each LVDS channel and the clock also varies with the jitter on the data and clock signals, as seen by the receiver.

The Arria II GX family supports the following receiver modes to overcome skew between the source-synchronous or reference clock and the received serial data:

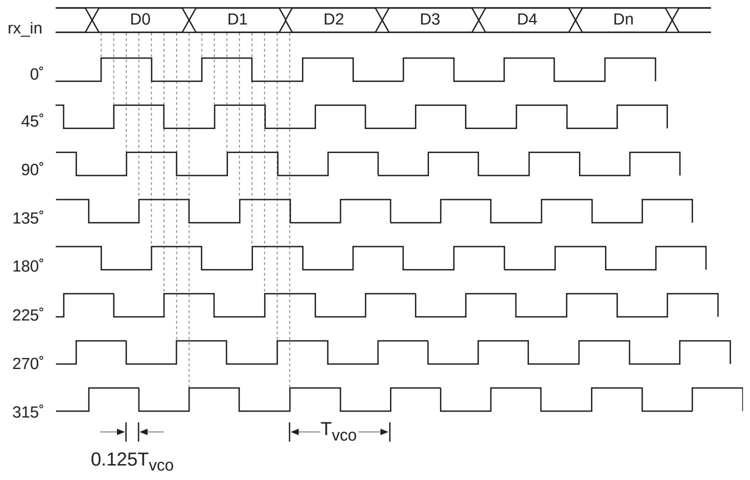
- Non-DPA mode
- DPA mode
- Soft-clock data recovery (CDR) mode

Dedicated SERDES and DPA circuitry only exist on the right side of the device. Top and bottom IO banks only support non-DPA mode, in which the serializer and deserializer are implemented in the core logic.

Dynamic Phase Alignment (DPA) Block

The DPA block takes in high-speed serial data from the differential input buffer and selects the optimal phase from one of the eight clock phases generated by the center/corner PLL to sample the data. The eight phases of the clock are equally divided, giving a 45° resolution. The maximum phase offset between the received data and the selected phase is 1/8 unit interval (UI), which is the maximum quantization error of the DPA. The optimal clock phase selected by the DPA block (`DPA_diffioclk`) is also used to write data into the FIFO buffer or to clock the SERDES for soft-CDR operation. Figure 8–7 shows the possible phase relationships between the DPA clocks and the incoming serial data.

Figure 8–7. DPA Clock Phase to Serial Data Timing Relationship *(Note 1)*



Note to Figure 8–7:

(1) T_{VCO} is defined as the PLL serial clock period.

The DPA block requires a training pattern and a training sequence of at least 256 repetitions. The training pattern is not fixed, so you can use any training pattern with at least one transition. An optional user controlled signal (`rx_dpll_hold`) freezes the DPA on its current phase when asserted. This is useful if you do not want the DPA to continuously adjust phase after initial phase selection.

The DPA loses lock when it switches phases to maintain an optimal sampling phase. Once locked, the DPA can lose lock under either of the following conditions:

- One phase change (adjacent to the current phase)
- Two phase changes in the same direction

An independent reset signal (`rx_reset`) is routed from the FPGA fabric to reset the DPA circuitry in user mode. The DPA circuitry must be retrained after reset.

Synchronizer

The synchronizer is a 1-bit wide and 6-bit deep FIFO buffer that compensates for the phase difference between the `DPA_diffioclk` and the high-speed clock (`LVDS_diffioclk`) produced by the center/corner PLL. Because every DPA channel might have a different phase selected to sample the data, the FIFO buffer is needed to synchronize the data to the high-speed LVDS clock domain. The synchronizer can only compensate for phase differences, not frequency differences between the data and the receiver's input reference clock, and is automatically reset when the DPA first locks to the incoming data.

An optional signal (`rx_fifo_reset`) is available to the FPGA fabric to reset the synchronizer. Altera® recommends using `rx_fifo_reset` to reset the synchronizer when the DPA signal is in a loss-of-lock condition and data checker indicates corrupted received data.

Data Realignment Block (Bit Slip)

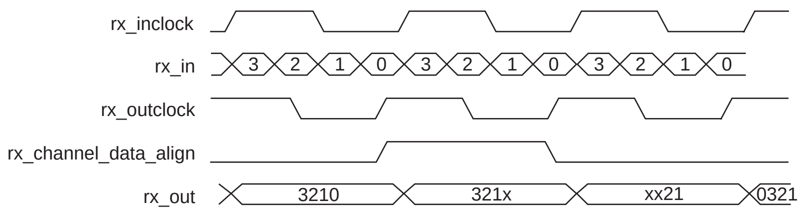
Skew in the transmitted data along with skew added by the link causes channel-to-channel skew on the received serial data streams. If DPA is enabled, the received data is captured with different clock phases on each channel. This might cause the received data to be misaligned from channel to channel. To compensate for this channel-to-channel skew and establish the correct received word boundary at each channel, each receiver channel has a dedicated data realignment circuit that realigns the data by inserting bit latencies into the serial stream.

An optional signal (`rx_channel_data_align`) controls the bit insertion of each receiver independently controlled from the internal logic. The data slips one bit on the rising edge of `rx_channel_data_align`. The following are requirements for the `rx_channel_data_align` signal:

- An edge-triggered signal
- The minimum pulse width is one period of the parallel clock in the logic array
- The minimum low time between pulses is one period of the parallel clock
- Holding `rx_channel_data_align` does not result in extra slips
- Valid data is available two parallel clock cycles after the rising edge of the `rx_channel_data_align` signal

Figure 8–8 shows receiver output after one bit-slip pulse with the deserialization factor set to 4.

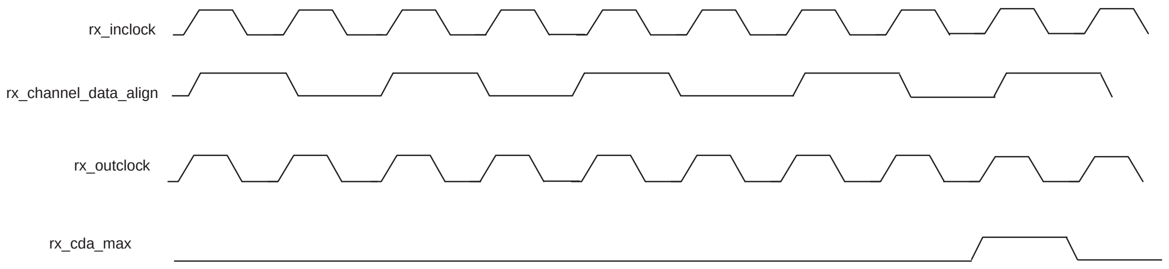
Figure 8–8. Data Realignment Timing



The data realignment circuit can have up to 11 bit-times of insertion before a rollover occurs. The programmable bit rollover point can be from 1 to 11 bit-times, independent of the deserialization factor. The programmable bit rollover point needs to be set to equal to or greater than the deserialization factor, allowing enough depth in the word alignment circuit to slip through a full word. You can set the value of the bit rollover point using the ALTLVDS megafunction. An optional status signal (`rx_cda_max`) is available to the FPGA fabric from each channel to indicate when the preset rollover point is reached.

Figure 8–9 shows a preset value of four-bit times before rollover occurs. The `rx_cda_max` signal pulses for one `rx_outclock` cycle to indicate that the rollover has occurred.

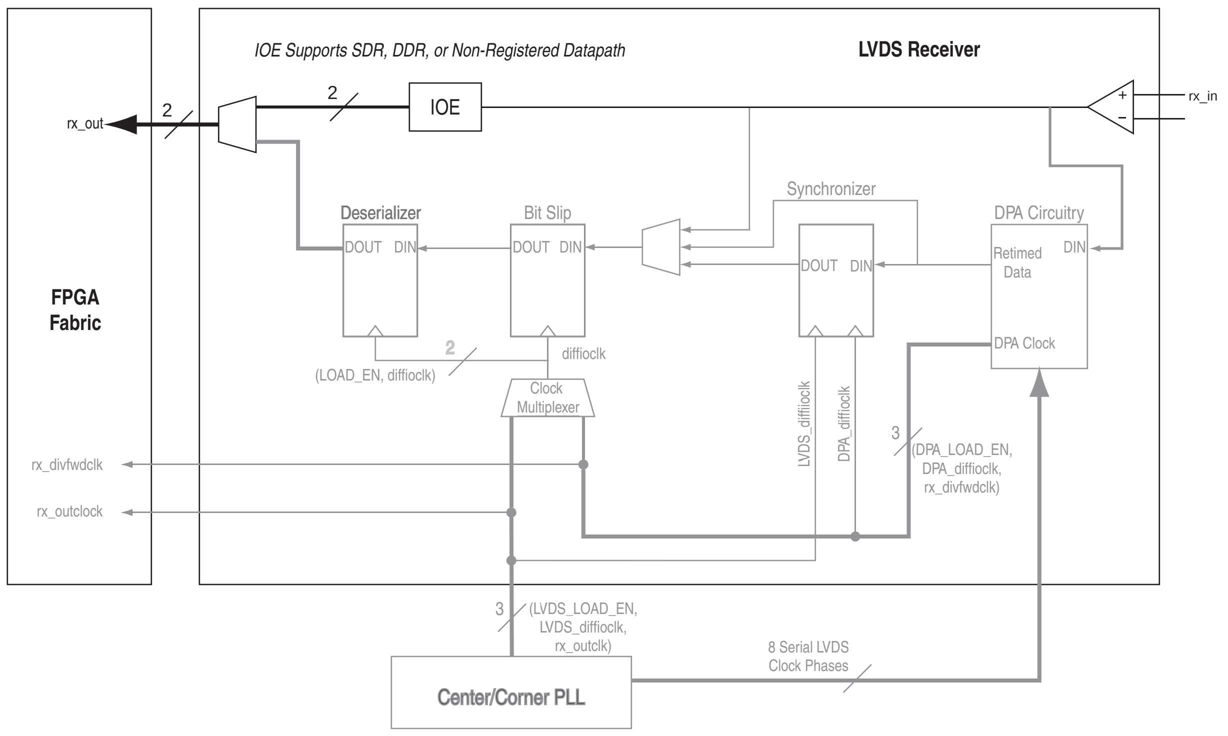
Figure 8–9. Receiver Data Re-Alignment Rollover



Deserializer

The deserializer, which includes shift registers and parallel load registers, converts the serial data from the bit slip to parallel data before sending the data to the FPGA fabric. The deserialization factor supported is 4, 6, 7, 8, or 10. The deserializer can be bypassed to support DDR (×2) and SDR (×1) operations, as shown in Figure 8–10. The DPA and data realignment circuit cannot be used when the deserializer is bypassed. The IOE contains two data input registers that can operate in DDR or SDR mode.

Figure 8–10. Arria II GX Deserializer Bypass *(Note 1)*, *(2)*, *(3)*



Notes to Figure 8–10:

(1) All disabled blocks and signals are grayed out.

(2) In DDR mode, `rx_inclock` clocks the IOE register. In SDR mode, data is directly passed through the IOE.

(3) In SDR and DDR mode, the data width from the IOE is 1 and 2, respectively.

Receiver Data Path Modes

The Arria II GX device family supports three receiver datapath modes:

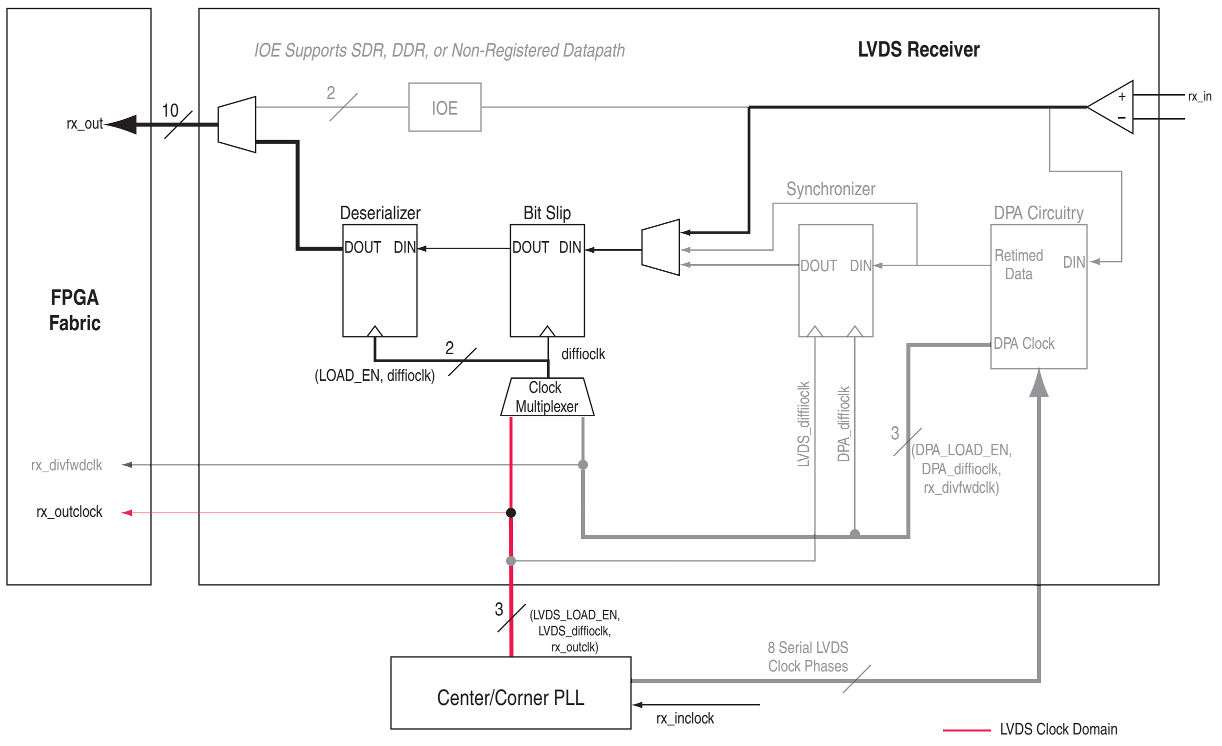
- Non-DPA mode
- DPA mode
- soft-CDR mode

Non-DPA Mode

Non-DPA mode allows you to statically select the optimal phase between the source-synchronous reference clock and the input serial data to compensate for any skew between the two signals. The reference clock must be a differential signal. Figure 8–11 shows the non-DPA datapath block diagram. Input serial data is registered at the rising or falling edge of the `LVDS_diffioclk` clock produced by the center/corner PLL. You can select the rising/falling edge option using the ALTLVDS megafunction. Both data realignment and deserializer blocks are clocked by the `LVDS_diffioclk` clock.

You should perform board trace skew compensation at data rate above 840Mbps in non-DPA mode.

Figure 8–11. Receiver Datapath in Non-DPA Mode *(Note 1)*, *(2)*, *(3)*



Notes to Figure 8–11:

(1) All disabled blocks and signals are grayed out.

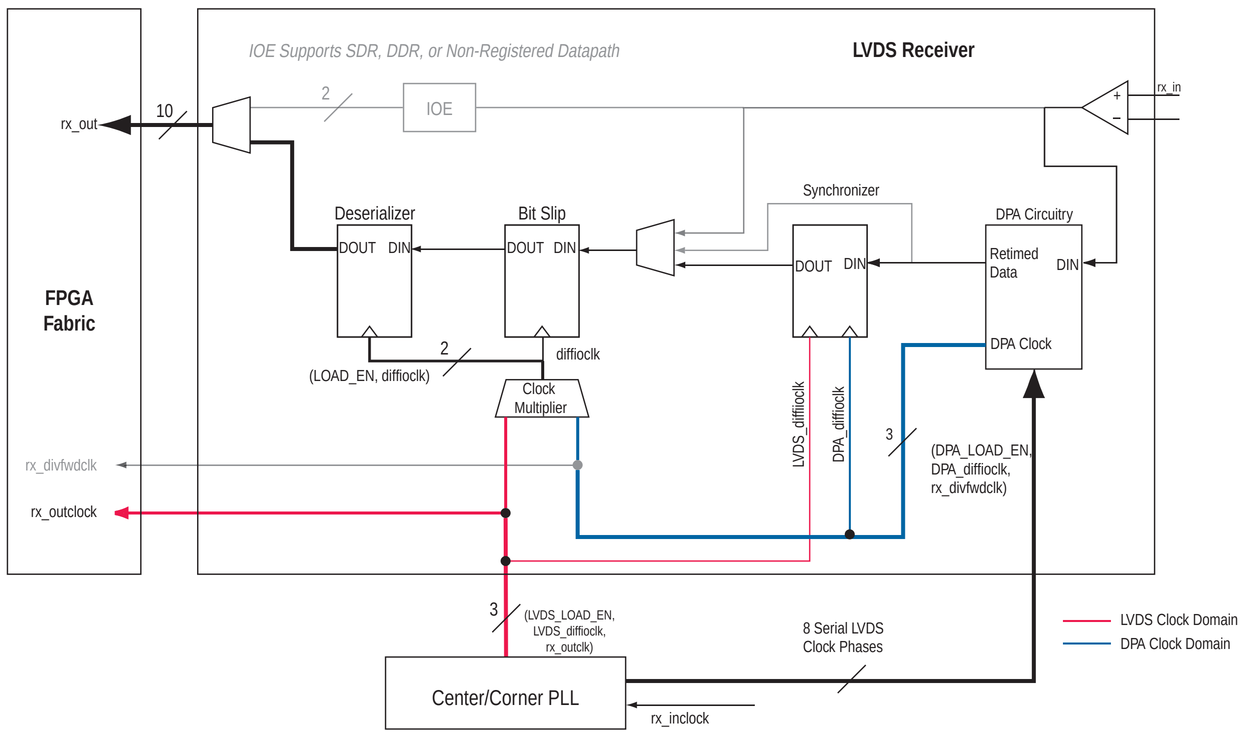
(2) In SDR and DDR mode, the data width from the IOE is 1 and 2, respectively.

(3) The `rx_out` port has a maximum data width of 10.

DPA Mode

In DPA mode, the DPA circuitry automatically chooses the optimal phase between the source synchronous reference clock and the input serial data to compensate for the skew between the two signals. The reference clock must be a differential signal. Figure 8–12 shows the DPA mode receiver datapath block diagram. The `DPA_diffioclk` clock is used to write serial data into the synchronizer. The `LVDS_diffioclk` clock is used to read the serial data from the synchronizer. The same `LVDS_diffioclk` clock is used in the data realignment and deserializer blocks.

Figure 8–12. Receiver Datapath in DPA Mode *(Note 1)*, *(2)*, *(3)*



Notes to Figure 8–12:

(1) All disabled blocks and signals are grayed out.

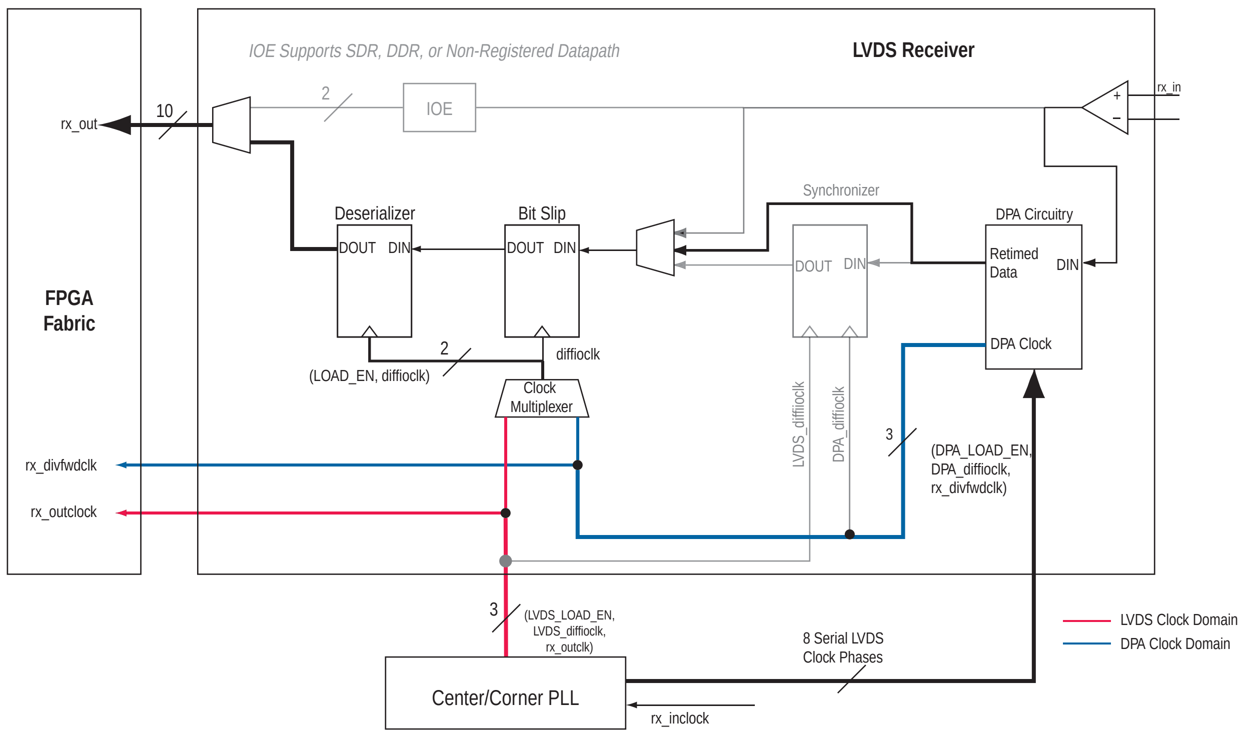
(2) In SDR and DDR mode, the data width from the IOE is 1 and 2, respectively.

(3) The `rx_out` port has a maximum data width of 10.

Soft-CDR Mode

Figure 8–13 shows the soft-CDR mode datapath block diagram. In soft-CDR mode, the PLL uses the local clock source as the reference clock. The reference clock must be a differential signal. The DPA continuously changes its phase to track the parts-per-million (PPM) difference between the upstream transmitter and the local receiver reference input clocks. The `DPA_diffioclk` clock is used for bit-slip operation and deserialization. The `DPA_diffioclk` clock is divided by the deserialization factor to produce the `rx_divfwdclk` clock, which is then forwarded to the FPGA fabric. The receiver output data (`rx_out`) to the FPGA fabric is synchronized to this clock. The parallel clock `rx_outclock`, generated by the center/corner PLL, is also forwarded to the FPGA fabric.

Figure 8–13. Receiver Datapath in Soft-CDR Mode *(Note 1)*, *(2)*, *(3)*



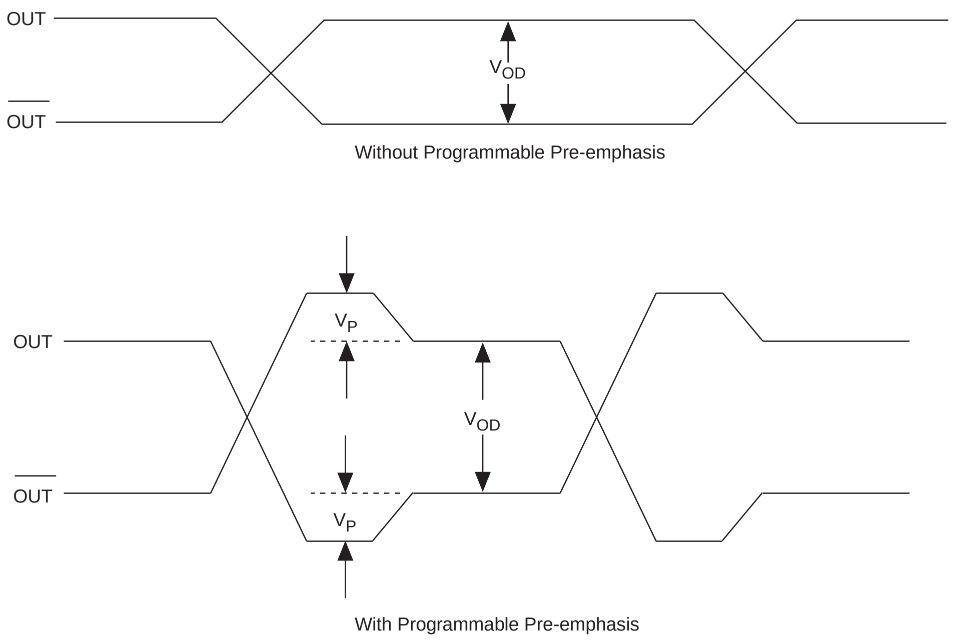
Notes to Figure 8–13:

(1) All disabled blocks and signals are grayed out.
(2) In SDR and DDR mode, the data width from the IOE is 1 and 2, respectively.
(3) The `rx_out` port has a maximum data width of 10.

Programmable Pre-Emphasis and Programmable V_{OD}

Pre-emphasis increases the amplitude of the high frequency component of the output signal and thus helps to compensate for the frequency-dependent attenuation along the transmission line. Figure 8–14 shows the LVDS output single-ended waveform with and without pre-emphasis. The definition of V_{OD} is also shown.

Figure 8–14. LVDS Output Single-Ended Waveform with and without Programmable Pre-Emphasis *(Note 1)*



Note to Figure 8–14:

(1) V_P— voltage boost from pre-emphasis.

Pre-emphasis is an important feature for high-speed transmission. Without pre-emphasis, the output current is limited by the V_{OD} setting and the output impedance of the driver. At high frequency, the slew rate may not be fast enough to reach the full V_{OD} before the next edge, producing a pattern-dependent jitter. With pre-emphasis, the output current is boosted momentarily during switching to increase the output slew rate. The overshoot introduced by the extra current happens only during switching and does not ring, unlike the overshoot caused by signal reflection. This overshoot should not be included in the V_{OD} voltage.

There are two pre-emphasis settings for each LVDS output buffer: no pre-emphasis and medium. The default setting is medium. Table 8–4 shows the assignment name and its possible values for programmable pre-emphasis in the Quartus II software Assignment Editor. The setting of no pre-emphasis is 1 and the setting of medium is 0.

Table 8–4. Programmable Pre-emphasis Settings in Quartus II Software Assignment Editor

Assignment name	Programmable Pre-emphasis
Allowed values	0, 1

There is one V_{OD} setting for each LVDS output buffer. Table 8–5 shows the assignment name and the value for programmable V_{OD} in the Quartus II software Assignment Editor.

Table 8–5. Programmable V_{OD} Settings in Quartus II Software Assignment Editor

Assignment name	Programmable Differential Output Voltage (V_{OD})
Allowed values	2

Differential I/O Termination

The Arria II GX device family provides 100-Ω differential OCT support for LVDS input buffers in top, right, and bottom I/O banks. OCT saves board space by not adding external resistors on the board. You can enable OCT in the Quartus II software Assignment Editor.

Figure 8–15 shows LVDS input OCT. Clock input pins (`CLK[4..15]`) do not support OCT.

Figure 8–15. LVDS Input Buffer On-Chip Differential I/O Termination

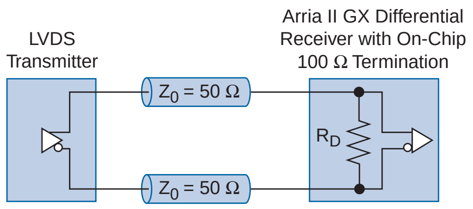

LVDS Transmitter
Arria II GX Differential Receiver with On-Chip 100 Ω Termination
Z0 = 50 Ω
Z0 = 50 Ω
RD

Table 8–6 shows the assignment name and its value for on-chip differential input termination in the Quartus II software Assignment Editor.

Table 8–6. On-Chip Differential Input Termination in Quartus II Software Assignment Editor

Assignment name	Input Termination (Accepts wildcards/groups)
Allowed values	Differential

PLLs

The Arria II GX device family contains up to six PLLs with up to four center and corner PLLs located on the right side of the device. The center/corner PLL on the right side of the device is used to generate parallel clocks (`rx_outclock` and `tx_outclock`) and high-speed clocks (`diffioclk`) for the SERDES and DPA circuitry. Figure 8–1 on page 8–3 shows the locations of the PLLs for Arria II GX devices. Clock switchover and dynamic reconfiguration are allowed using the center/corner PLLs in high-speed differential I/O support mode.

For more information about PLLs, refer to the *Clock Network and PLLs in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

LVDS and DPA Clock Networks

The Arria II GX device family only has LVDS and DPA clock networks on the right side of the device. The center/corner PLLs feed into the differential transmitter and receiver channels through the LVDS and DPA clock networks. Figure 8–16 and Figure 8–17 show the LVDS clock tree for family members without center PLLs and with center PLLs, respectively. The center PLLs can drive the LVDS clock tree above and below them. In Arria II GX devices with or without center PLLs, the corner PLLs can drive both top and bottom LVDS clock tree.

Figure 8–16. LVDS and DPA Clock Networks in the Arria II GX Family without Center PLLs

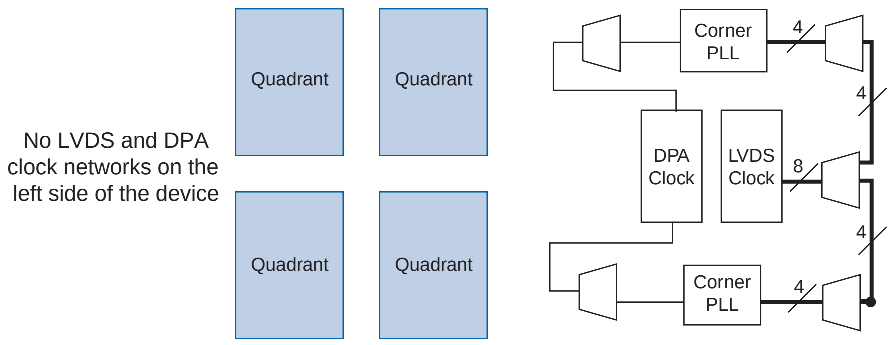
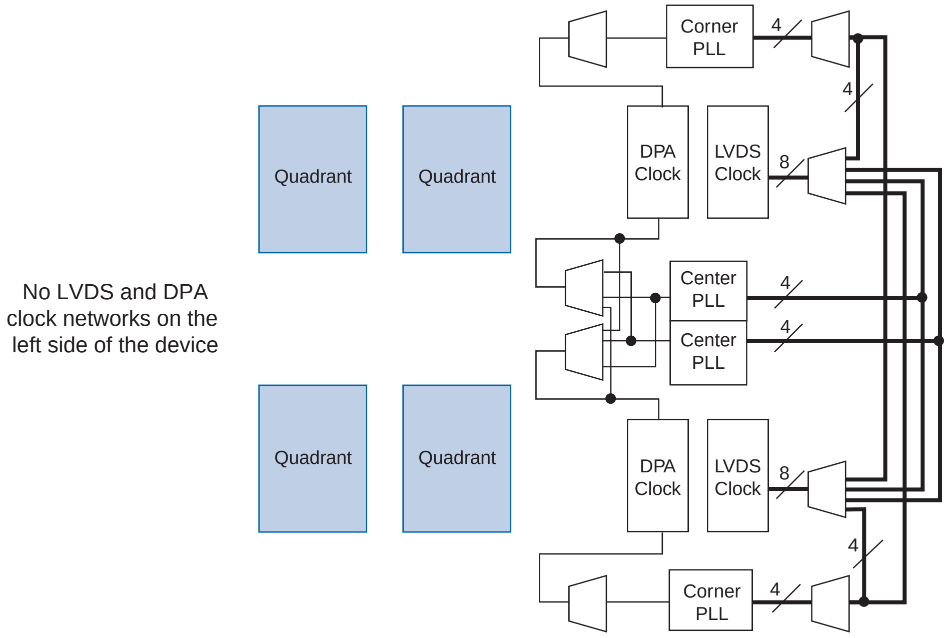


Figure 8–17. LVDS and DPA Clock Networks in the Arria II GX Family with Center PLLs



Source-Synchronous Timing Budget

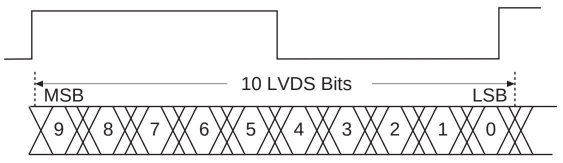
This section describes the timing budget, waveforms, and specifications for source-synchronous signaling in the Arria II GX device family. Timing analysis for the differential block is different from traditional synchronous timing analysis techniques. Therefore, it is important to understand how to analyze timing for high-speed differential signals. This section defines the source-synchronous differential data orientation timing parameters, timing budget definitions, and how to use these timing parameters to determine your design's maximum performance.

Differential Data Orientation

There is a set relationship between an external clock and the incoming data. For operation at 1 Gbps and a serialization factor of 10, the external clock is multiplied by 10. You can set the phase-alignment in the PLL to coincide with the sampling window of each data bit. The data is sampled on the falling edge of the multiplied clock.

Figure 8–18 shows the data bit orientation of the ×10 mode.

Figure 8–18. Bit Orientation



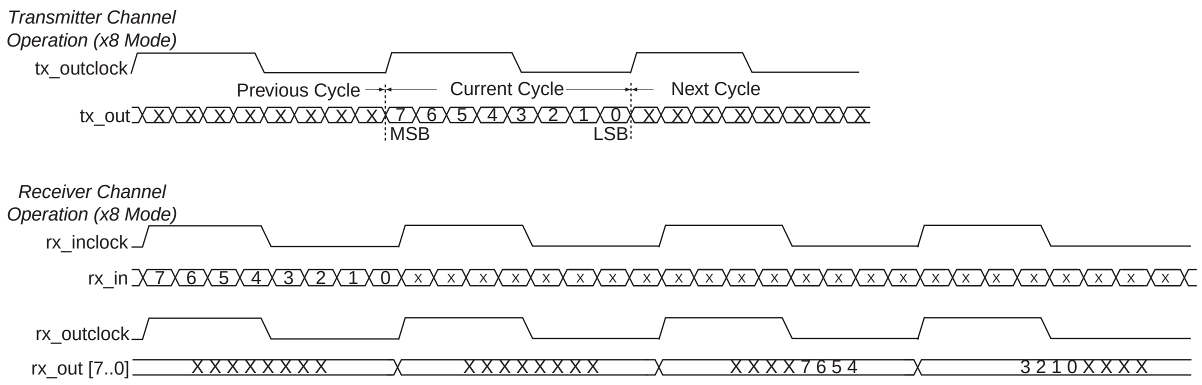
Differential I/O Bit Position

Data synchronization is necessary for successful data transmission at high frequencies. Figure 8–19 shows data bit orientation for a channel operation. These figures are based on the following:

- Serialization factor equals clock multiplication factor
- Edge alignment is selected for phase alignment
- Implemented in hard SERDES

For other serialization factors, use the Quartus II software tools and find the bit position in the word. The bit positions after deserialization are listed in Table 8–7.

Figure 8–19. Bit Order and Word Boundary for One Differential Channel *(Note 1)*


Transmitter Channel Operation (x8 Mode)
tx_outclock
Previous Cycle
Current Cycle
Next Cycle
tx_out
X X X X X X X X X X 7 6 5 4 3 2 1 0 X X X X X X X X X
MSB
LSB
Receiver Channel Operation (x8 Mode)
rx_inclock
rx_in
7 6 5 4 3 2 1 0 X X X X X X X X X X X X X X X X X X X X X X X X
rx_outclock
rx_out [7..0]
X X X X X X X X
X X X X X X X X
X X X X 7 6 5 4
3 2 1 0 X X X X

Note to Figure 8–19:

(1) These are only functional waveforms and are not intended to convey timing information.

Table 8–7 shows the conventions for differential bit naming for 18 differential channels. The MSB and LSB positions increase with the number of channels used in a system.

Table 8–7. Differential Bit Naming

Receiver Channel Data Number	Internal 8-Bit Parallel Data	
	MSB Position	**LSB Position**
1	7	0
2	15	8
3	23	16
4	31	24
5	39	32
6	47	40
7	55	48
8	63	56
9	71	64
10	79	72
11	87	80
12	95	88
13	103	96
14	111	104
15	119	112
16	127	120
17	135	128
18	143	136

Receiver Skew Margin for Non-DPA Mode

Changes in system environment, such as temperature, media (cable, connector, or PCB), and loading, effect the receiver's setup and hold times; internal skew affects the sampling ability of the receiver.

Different modes of LVDS receivers use different specifications, which can help in deciding the ability to sample the received serial data correctly. In DPA mode, use DPA jitter tolerance instead of receiver skew margin (RSKM).

In non-DPA mode, RSKM, transmitter channel-to-channel skew (TCCS), and sampling window (SW) specifications are used for high-speed source-synchronous differential signals in the receiver data path. The relationship between RSKM, TCCS, and SW can be expressed by the RSKM equation shown in Equation 8–1:

Equation 8–1.

$$\text{RSKM} = (\text{TUI} - \text{SW} - \text{TCCS})/2$$

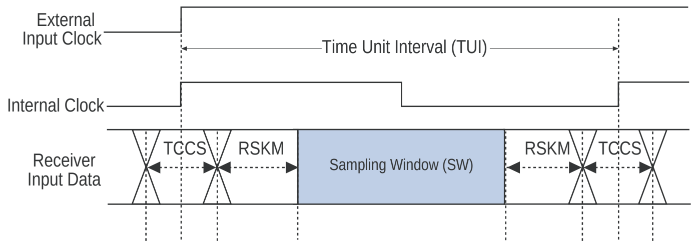
Where:

- TUI—the time period of the serial data.
- RSKM—the timing margin between the receiver's clock input and the data input SW.
- SW—the period of time that the input data must be stable to ensure that data is successfully sampled by the LVDS receiver. The sampling window is device property and varies with device speed grade.
- TCCS—the difference between the fastest and slowest data output transitions, including the t_{CO} variation and clock skew.

You must calculate the RSKM value to decide whether or not data can be sampled properly by the LVDS receiver with the given data rate and device. A positive RSKM value indicates the LVDS receiver can sample the data properly; a negative RSKM indicates the receiver cannot sample the data properly.

Figure 8–20 shows the relationship between the RSKM, TCCS, and SW.

Figure 8–20. Differential High-Speed Timing Diagram and Timing Budget for Non-DPA



For LVDS receivers, the Quartus II software provides the RSKM report showing SW, TUI, and RSKM values for non-DPA mode. You can generate the RSKM by executing the **report_RSKM** command in the TimeQuest Timing Analyzer. You can find the RSKM report in the Quartus II Compilation report under **TimeQuest Timing Analyzer** section.

To obtain the RSKM value, assign an appropriate input delay to the LVDS receiver through the TimeQuest Timing Analyzer constraints menu

Differential Pin Placement Guidelines

To ensure proper high-speed operation, differential pin placement guidelines are established. The Quartus II compiler automatically checks that these guidelines are followed and issues an error message if they are not adhered to. This section is divided into pin placement guidelines with and without DPA usage.

DPA-enabled differential channels refer to DPA mode or soft-CDR mode; DPA-disabled channels refer to non-DPA mode.

DPA-Enabled Channels and Single-Ended I/Os

When single-ended I/Os and LVDS I/Os share the same I/O bank, the placement of single-ended I/O pins with respect to LVDS I/O pins is restricted. The constraints on single-ended I/Os placement with respect to DPA-enabled or DPA-disabled LVDS I/Os are the same.

For more information about pin placement with respect to LVDS, mini-LVDS, and RSDS, refer to the *I/O Management* chapter in volume 2 of the *Quartus II Development Software Handbook*.

Guidelines for DPA-Enabled Differential Channels

When you use DPA-enabled channels, you must adhere to the guidelines listed in the following sections.

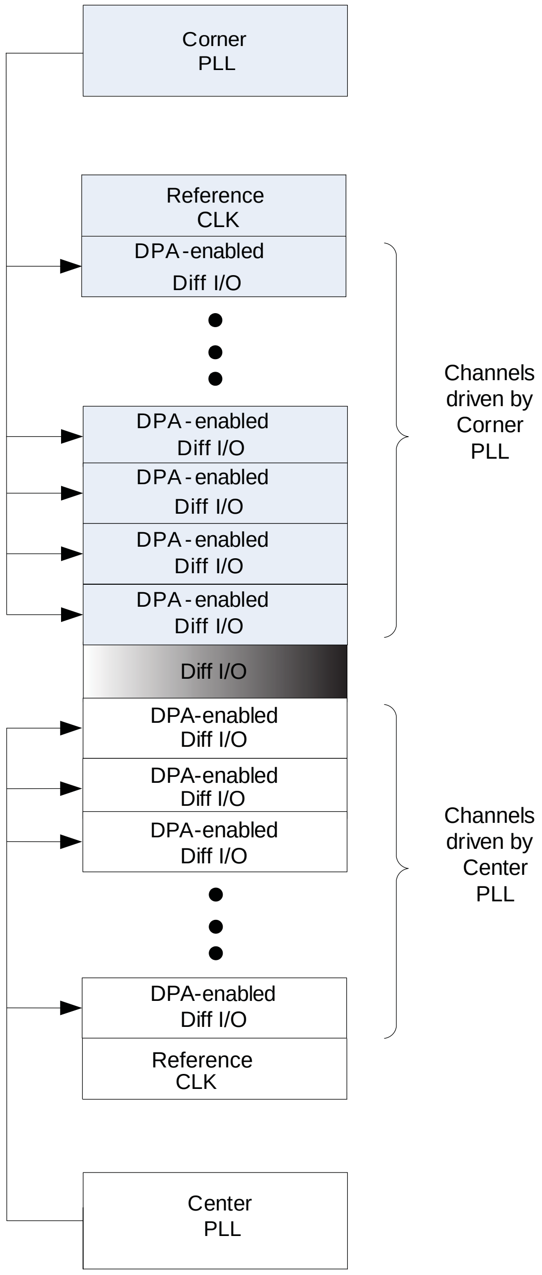
DPA-Enabled Channel Driving Distance

If the number of DPA-enabled channels driven by each center or corner PLL exceeds 25 logic array blocks (LAB) rows, Altera recommends implementing data realignment (bit slip) circuitry for all the DPA channels.

Using Center and Corner PLLs

- If the DPA-enabled channels in a bank are being driven by two PLLs, where the corner PLL is driving one group and the center PLL is driving another group, there must be at least one row of separation between the two groups of DPA-enabled channels, as shown in Figure 8–21. This is to prevent noise mixing because the two groups can operate at independent frequencies.
- No separation is necessary if a single PLL is driving both the DPA-enabled channels and DPA-disabled channels.

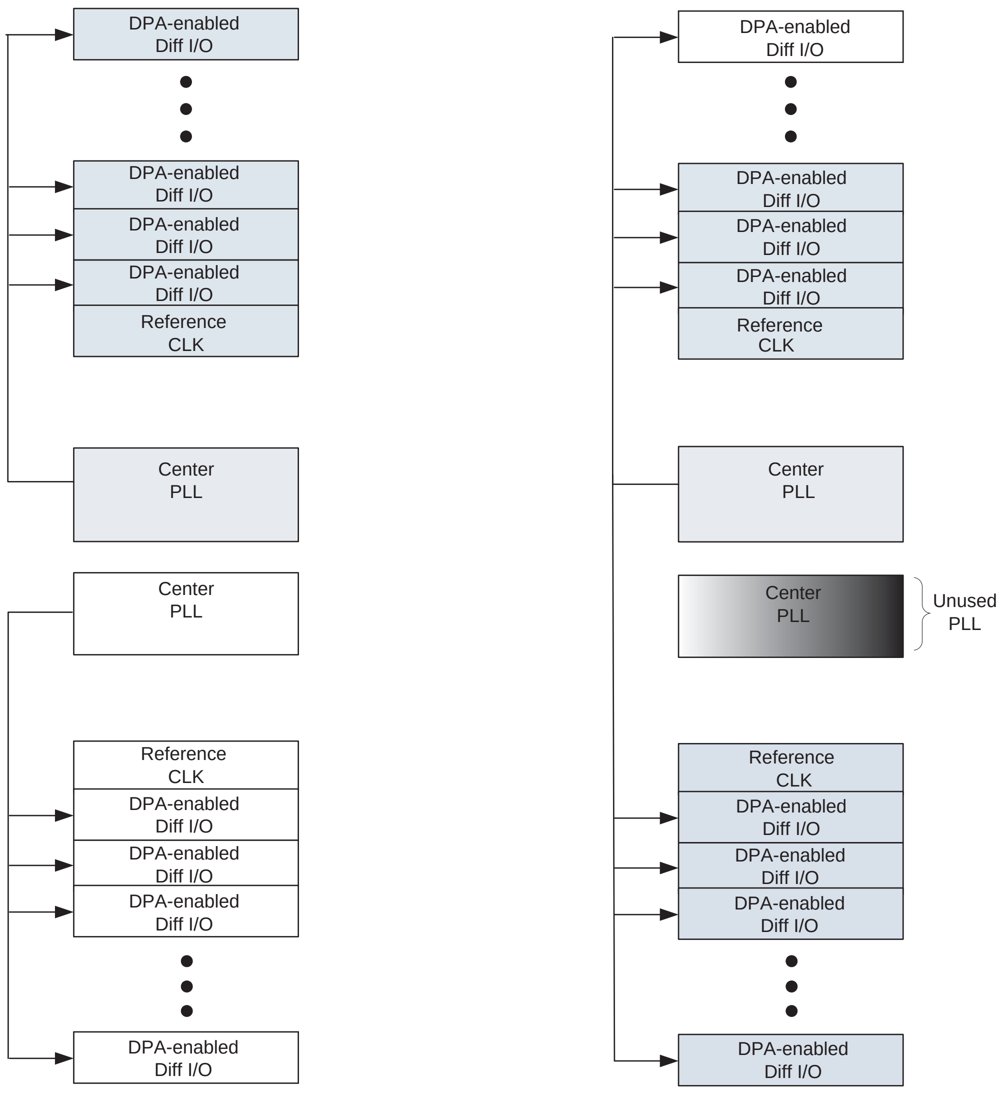
Figure 8–21. Center and Corner PLLs Driving DPA-Enabled Differential I/Os in the Same Bank



Using Both Center PLLs

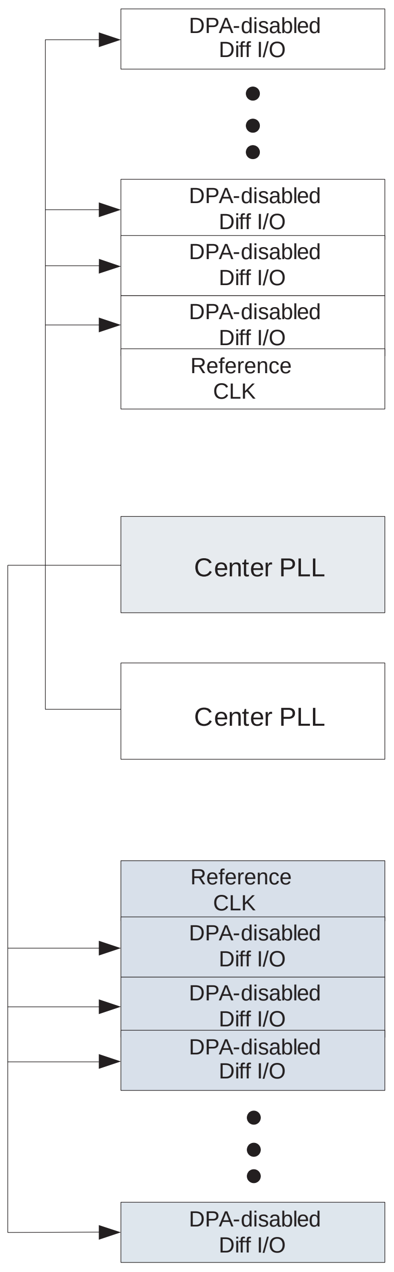
- You can use both center PLLs to drive DPA-enabled channels simultaneously, as long as they drive these channels in their adjacent banks only, as shown in Figure 8–21.
- If one of the center PLLs drives the DPA-enabled channels in the upper and lower I/O banks, the other center PLL cannot be used for DPA, as shown in Figure 8–22.

Figure 8–22. Center PLLs Driving DPA-Enabled Differential I/Os



- If the upper center PLL drives DPA-enabled channels in the lower I/O bank, the lower center PLL cannot drive DPA enabled channels in the upper I/O bank, and vice versa. In other words, the center PLLs cannot drive cross-banks simultaneously, as shown in Figure 8–23.

Figure 8–23. Invalid Placement of DPA-Disabled Differential I/Os Driven by Both Center PLLs

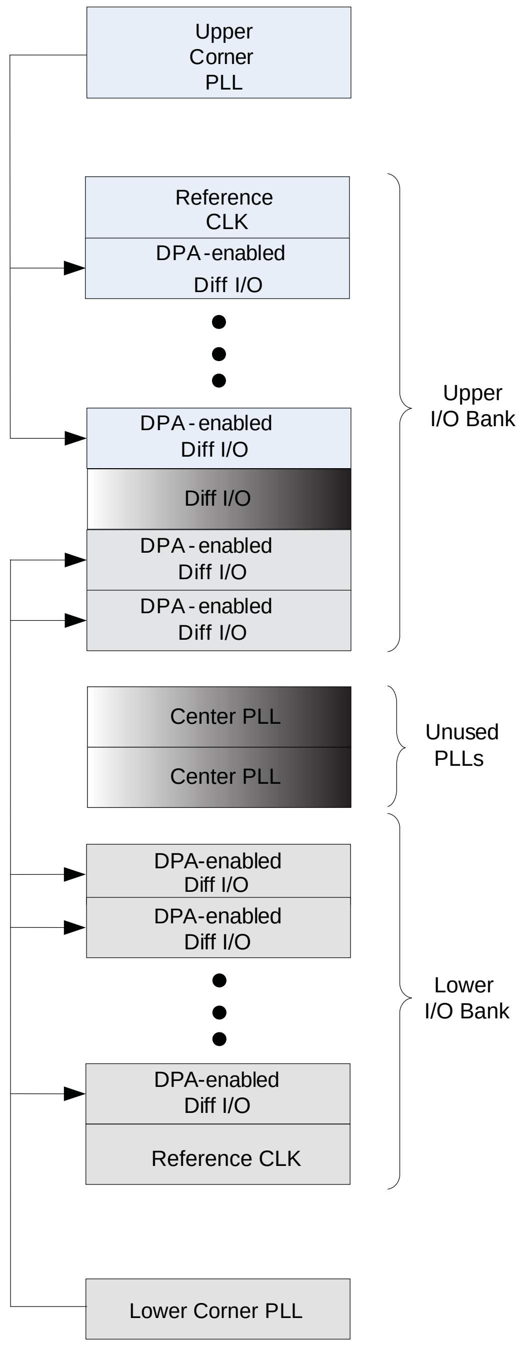


Using Both Corner PLLs

- You can use both corner PLLs to drive DPA-enabled channels simultaneously, as long as they drive the channels in their adjacent banks only. There must be at least one row of separation between the two groups of DPA-enabled channels.
- If one of the corner PLLs drives DPA-enabled channels in the upper and lower I/O banks, the center PLLs cannot be used. The other corner PLL can be used to drive DPA-enabled channels in their adjacent bank only. There must be at least one row of separation between the two groups of DPA-enabled channels. See Figure 8–24.

- If the upper corner PLL drives DPA-enabled channels in the lower I/O bank, the lower corner PLL cannot drive DPA-enabled channels in the upper I/O bank, and vice versa. In other words, the corner PLLs cannot drive cross-banks simultaneously, as shown in Figure 8–24.

Figure 8–24. Corner PLLs Driving DPA-Enabled Differential I/Os



Guidelines for DPA-Disabled Differential Channels

When you use DPA-disabled channels, you must adhere to the guidelines in the following sections.

DPA-Disabled Channel Driving Distance

Each PLL can drive all the DPA-disabled channels in the entire bank.

Using Corner and Center PLLs

- You can use a corner PLL to drive all transmitter channels and you can use a center PLL to drive all DPA-disabled receiver channels in the same I/O bank. In other words, you can drive a transmitter channel and a receiver channel in the same LAB row by two different PLLs, as shown in Figure 8–25.
- A corner PLL and a center PLL can drive duplex channels in the same I/O bank, as long as the channels driven by each PLL are not interleaved. No separation is necessary between the group of channels driven by the corner and center PLLs. See Figure 8–25 and Figure 8–26.

Figure 8–25. Corner and Center PLLs Driving DPA-Disabled Differential I/Os in the Same Bank

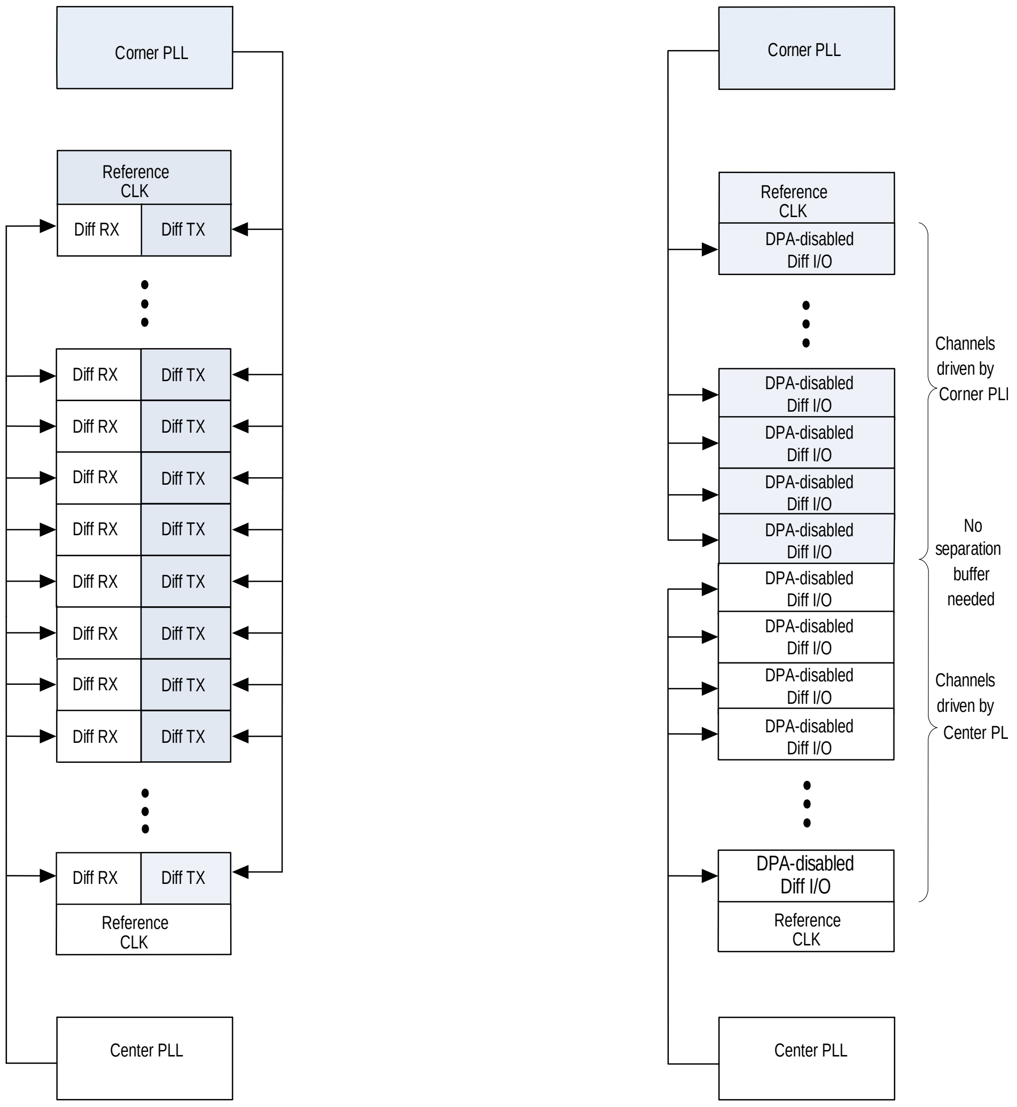
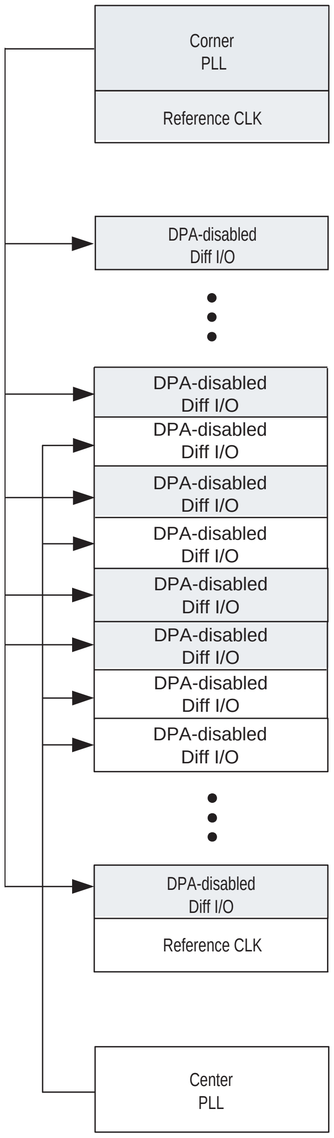


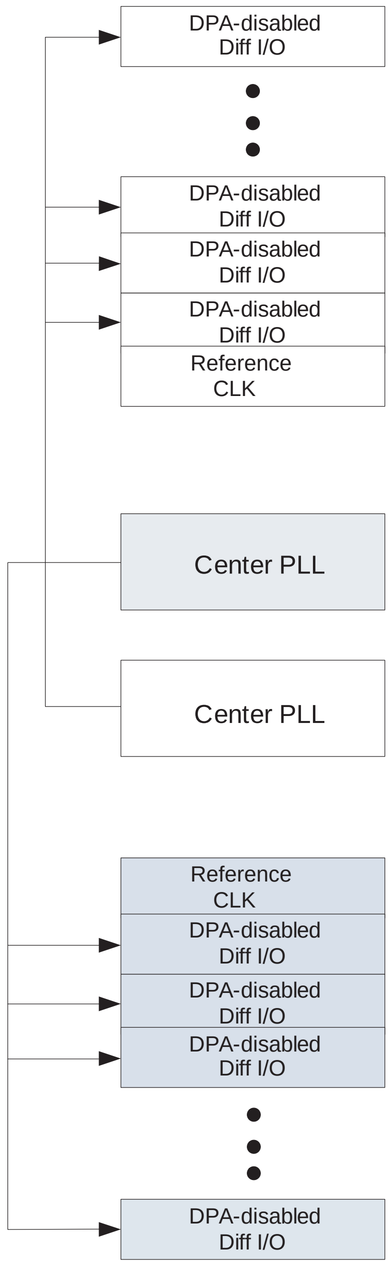
Figure 8–26. Invalid Placement of DPA-Disabled Differential I/Os Due to Interleaving of Channels Driven by the Corner and Center PLLs



Using Both Center PLLs

You can use both center PLLs simultaneously to drive DPA disabled channels on upper and lower I/O banks. Unlike DPA-enabled channels, the center PLLs can drive DPA-disabled channels cross-banks. For example, the upper center PLL can drive the lower I/O bank at the same time the lower center PLL is driving the upper I/O bank, and vice versa, as shown in Figure 8–27.

Figure 8–27. Both Center PLLs Driving Cross-Bank DPA-Disabled Channels Simultaneously



Using Both Corner PLLs

- You can use both corner PLLs to drive DPA disabled channels simultaneously. Both corner PLLs can drive cross-banks.
- You can use a corner PLL to drive all the transmitter channels and you can use the other corner PLL to drive all DPA-disabled receiver channels in the same I/O bank.
- Both corner PLLs can drive duplex channels in the same I/O bank, as long as the channels driven by each PLL are not interleaved. No separation is necessary between the group of channels driven by both corner PLLs.

Setting Up an LVDS Transmitter or Receiver Channel

Altera's ALTLVDS megafunction offers you the ease of setting up an LVDS transmitter or receiver channel. You can control the settings of SERDES and DPA circuitry in the ALTLVDS megafunction. When you instantiate an ALTLVDS megafunction, the PLL is instantiated automatically and you can set the parameters of the PLL. This simplifies the clocking setup for the LVDS transmitter or receiver channels. However, the drawback is reduced flexibility in PLL utilization.

The ALTLVDS megafunction provides an option for implementing the LVDS transmitter or receiver interfaces with external PLLs. With this option enabled, you can control the PLL settings, such as dynamically reconfiguring the PLLs to support different data rates, dynamic phase shift, and other settings. You also must instantiate an ALTPLL megafunction to generate the various clock and load enable signals.

For more information about how to control the PLL, SERDES, and DPA settings, and detail descriptions of the LVDS transmitter and receiver interface signals, refer to the *ALTLVDS Megafunction User Guide*.

For more information about the ALTPLL megafunction, refer to the *ALTPLL Megafunction User Guide*.

Document Revision History

Table 8–8 shows the revision history for this document.

Table 8–8. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial Release.	—

Section III. System Integration

This section provides information about Arria® II GX device configuration, design security, remote system upgrades, SEU mitigation, JTAG, and power requirements. This section includes the following chapters:

Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in this volume.

9. Configuration, Design Security, and Remote System Upgrades in Arria II GX Devices

AIIGX51009-1.0

Introduction

This chapter contains information about the Arria® II GX supported configuration schemes, instructions about how to execute the required configuration schemes, and all the necessary option pin settings.

This chapter includes the following sections:

Arria II GX devices use SRAM cells to store configuration data. As SRAM memory is volatile, you must download configuration data to the Arria II GX device each time the device powers up. You can configure Arria II GX devices using one of four configuration schemes:

- Fast passive parallel (FPP)
- Fast active serial (AS)
- Passive serial (PS)
- Joint Test Action Group (JTAG)

All configuration schemes use either an external controller (for example, a MAX® II device or microprocessor), a configuration device, or a download cable. For more information about the configuration features, refer to "Configuration Features" on page 9–2.

Configuration Devices

Altera® serial configuration devices support single-device and multi-device configuration solutions for Arria II GX devices and are used by the Fast AS configuration scheme. Serial configuration devices offer a low-cost, low pin-count configuration solution.

For information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook*.

All minimum timing information in this chapter covers the entire Arria II GX device family. Some devices may work at less than the minimum timing stated in this chapter due to process variations

Table 9–3 on page 9–7 shows the uncompressed raw binary file (**.rbf**) configuration file sizes for Arria II GX devices.

Configuration Features

Arria II GX devices offer decompression, design security, and remote system upgrade features. Arria II GX devices can receive a compressed configuration bitstream and decompress this data in real-time, reducing storage requirements and configuration time. Design security using configuration bitstream encryption is available in Arria II GX devices, which protects your designs. You can make real-time system upgrades of your Arria II GX designs from remote locations with the remote system upgrade feature.

Table 9–1 summarizes which configuration features you can use in each configuration scheme.

Table 9–1. Arria II GX Configuration Features

Configuration Scheme	Configuration Method	Decompression	Design Security	Remote System Upgrade
FPP	MAX II device or a microprocessor with flash memory	✓ *(1)*	✓ *(1)*	—
Fast AS	Serial configuration device	✓	✓	✓ *(2)*
PS	MAX II device or a microprocessor with flash memory	✓	✓	—
	Download cable	✓	✓	—
JTAG	MAX II device or a microprocessor with flash memory	—	—	—
	Download cable	—	—	—

Notes to Table 9–1:

(1) In these modes, the host system must send a DCLK that is ×4 the data rate.

(2) Remote system upgrade is only available in the Fast AS configuration scheme. Only remote update mode is supported when using the Fast AS configuration scheme. Local update mode is not supported.

You can also refer to the following:

- For more information about the configuration data decompression feature, refer to "Configuration Data Decompression" on page 9–40.
- For more information about the remote system upgrade feature, refer to "Remote System Upgrades" on page 9–42.
- For more information about the design security feature, refer to the "Design Security" on page 9–54.
- For more information about the parallel flash loader (PFL), refer to *AN 386: Using the MAX II Parallel Flash Loader with the Quartus II Software*.

If your system already contains a common flash interface (CFI) flash memory device, you can use it for the Arria II GX device configuration storage as well. The PFL feature in MAX II devices provides an efficient method to program CFI flash memory devices through the JTAG interface and logic to control configuration from the flash memory device to the Arria II GX device. Both PS and FPP configuration modes are supported using the PFL feature.

For more information about programming Altera serial configuration devices, refer to "Programming Serial Configuration Devices" on page 9–20.

Power-On Reset Circuit and Configuration Pins Power Supply

The following section describes the power-on reset (POR) circuit and the power supply for the configuration pins.

Power-On Reset Circuit

The POR circuit keeps the entire system in reset mode until the power supply voltage levels have stabilized on power-up. Upon power-up, the device does not release nSTATUS until V_{CCCB}, V_{CCA_PLL}, V_{CC}, V_{CCPD}, and V_{CCIO} for I/O banks 3C or 8C are above the device's POR trip point. On power down, brown-out occurs if V_{CC} ramps down below the POR trip point and any of the V_{CC}, V_{CCPD}, or V_{CCIO} for I/O banks 3C or 8C drops below the threshold level of the hot-socket circuitry.

In Arria II GX devices, you can select between a fast POR time or a standard POR time, depending on the MSEL pin settings. The fast POR time is typically 4 ms for fast configuration time. The standard POR time is typically 100 ms, which has a lower power-ramp rate.

V_{CCIO} Pins for I/O Banks 3C and 8C

In Arria II GX devices, all the dedicated configuration pins and some of the dual function pins are supplied by the V_{CCIO} for I/O banks 3C and 8C in which they reside. The supported configuration voltages are 1.8, 2.5, 3.0, and 3.3 V. Arria II GX devices do not support the 1.5-V configuration.

You must use V_{CCIO} for I/O banks 3C and 8C to power all dedicated configuration inputs, dedicated configuration outputs, dedicated configuration bidirectional pins, and some of the dual functional pins that you use for configuration. With V_{CCIO} for I/O banks 3C and 8C, configuration input buffers do not have to share power lines with the regular I/O buffer.

The operating voltage for the configuration input pin is independent of the I/O bank's power supply V_{CCIO} during configuration. Therefore, no configuration voltage constraints on V_{CCIO} are needed in Arria II GX devices.

V_{CCPD} Pins

Arria II GX devices have a dedicated programming power supply, V_{CCPD}, which must be connected to 3.3 V, 3.0 V, or 2.5 V to power the I/O pre-drivers, the JTAG output pin (`TDO`), and the `MSEL[3..0]` pins.

☞ V_{CCPD} and V_{CCIO} for I/O banks 3C and 8C must ramp up from 0 V to the desired voltage level in 100 ms when you select standard POR time or 4 ms when you select fast POR time. If these supplies are not ramped up in this specified time, your Arria II GX device will not configure successfully. If your system cannot ramp up the power supplies in 100 ms or 4 ms, you must hold `nCONFIG` low until all the power supplies are stable.

☞ You must connect V_{CCPD} according to the I/O standard used in the same bank:

- For 3.3-V I/O standards, connect V_{CCPD} to 3.3 V
- For 3.0-V I/O standards, connect V_{CCPD} to 3.0 V
- For 2.5-V and below I/O standards, connect V_{CCPD} to 2.5 V

For more information about configuration pins power supply, refer to "Device Configuration Pins" on page 9–34.

Configuration Process

The following sections describe the general configuration process for FPP, Fast AS, and PS schemes.

Power Up

To begin the configuration process, you must fully power V_{CC}, V_{CCCB}, V_{CCA_PLL}, V_{CCPD}, and V_{CCIO} (including I/O banks 3C and 8C where the configuration and JTAG pins reside) to the appropriate voltage levels.

☞ For FPP configuration, pins `DATA[7..1]` will be used and they are supplied by the V_{CCIO} for I/O bank 6A. This bank needs to be powered up when FPP configuration is used.

Reset

Upon power-up, the Arria II GX device goes through a POR. The POR delay is dependent on the MSEL pin settings. During POR, the device resets, holds `nSTATUS` low, clears configuration RAM bits, and tri-states all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. While `nCONFIG` is low, the device is in reset. When the device comes out of reset, `nCONFIG` must be at a logic-high level in order for the device to release the open-drain `nSTATUS` pin. After `nSTATUS` is released, it is pulled high by a pull-up resistor and the device is ready to receive configuration data.

Before and during configuration, all user I/O pins are tri-stated. If nIO_pullup is driven low during power up and configuration, the user I/O pins and dual-purpose I/O pins have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

Configuration

nCONFIG and nSTATUS must be at a logic-high level in order for the configuration stage to begin. The device receives configuration data on its DATA pins and (for synchronous configuration schemes) the clock source on the DCLK pin. Configuration data is latched into the FPGA on the rising edge of DCLK. After the FPGA has received all the configuration data successfully, it releases the CONF_DONE pin, which is pulled high by a pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

To ensure DCLK and DATA0 are not left floating at the end of configuration, they must be driven either high or low, whichever is convenient on your board. DATA[0] is a dedicated pin that is used for both passive and active configuration modes. It is not available as a user I/O pin after configuration.

For FPP and PS configuration schemes, the configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause the configuration by halting DCLK for an indefinite amount of time.

A reconfiguration is initiated by toggling the nCONFIG pin from high to low and then back to high with a minimum t_{CFG} low-pulse width either in the configuration, configuration error, initialization, or user mode stage. When nCONFIG is pulled low, nSTATUS and CONF_DONE are also pulled low and all I/O pins are tri-stated. Once nCONFIG and nSTATUS return to a logic-high level, configuration begins.

Configuration Error

If an error occurs during configuration, Arria II GX devices assert the nSTATUS signal low, indicating a data frame error; the CONF_DONE signal stays low. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box) is turned on, the Arria II GX device resets the configuration device and retries the configuration. If this option is turned off, the system must monitor nSTATUS for errors and then pulse nCONFIG low to restart the configuration.

Initialization

In Arria II GX devices, the initialization clock source is either the internal oscillator or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Arria II GX device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. You can turn on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. If you supply a clock on CLKUSR, it will not affect the configuration process. After all the configuration data is accepted and CONF_DONE goes high, CLKUSR is enabled after the time specified as t_{CD2CU}. After this time period elapses, Arria II GX devices require a minimum number of clock cycles to initialize properly and enter user mode as specified in the t_{CD2UMC} parameter.

User Mode

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. If you use the INIT_DONE pin, it is high due to an external 10-kΩ pull-up resistor 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 goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

Configuration Schemes

The following sections describe configuration schemes for Arria II GX devices.

MSEL Pin Settings

Select the configuration scheme by driving the Arria II GX device MSEL pins either high or low, as shown in Table 9–2. The MSEL input buffers are powered by the V_{CCPD} power supply. Altera recommends you hardwire the MSEL[] pins to V_{CCPD} or GND. The MSEL[3..0] pins have 5-kΩ internal pull-down resistors that are always active. During POR and during reconfiguration, the MSEL pins must be at LVTTL V_{IL} and V_{IH} levels to be considered logic low and logic high, respectively.

☞ To avoid problems with detecting an incorrect configuration scheme, hardwire the MSEL[] pins to V_{CCPD} or GND without pull-up or pull-down resistors. Do not drive the MSEL[] pins by a microprocessor or another device.

Table 9–2. Arria II GX Configuration Schemes (Part 1 of 2)

Configuration Scheme	MSEL3	MSEL2	MSEL2	MSEL0	POR Delay	Configuration Voltage Standard (V) *(1)*
Fast passive parallel (FPP)	0	0	0	0	Fast	3.3, 3.0/2.5
	0	1	1	1	Fast	1.8
FPP with design security feature and/or decompression enabled *(2)*	0	0	0	1	Fast	3.3, 3.0/2.5
	1	0	0	0	Fast	1.8

Table 9–2. Arria II GX Configuration Schemes (Part 2 of 2)

Configuration Scheme	MSEL3	MSEL2	MSEL2	MSEL0	POR Delay	Configuration Voltage Standard (V) *(1)*
Passive Serial (PS)	0	0	1	0	Fast	3.3, 3.0/2.5
	1	0	0	1	Fast	1.8
	1	0	1	0	Standard	3.3, 3.0/2.5
	1	0	1	1	Standard	1.8
Active Serial (AS) with or without remote system upgrade *(3)*	0	0	1	1	Fast	3.3
	1	1	0	1	Fast	3.0/2.5
	1	1	1	0	Standard	3.3
	1	1	1	1	Standard	3.0/2.5
JTAG-based configuration *(4)*	*(5)*	*(5)*	*(5)*	*(5)*	—	—

Notes to Table 9–2:

(1) Configuration voltage standard applied to the V_{CCIO} supply in which the configuration pins reside.

(2) These modes are only supported when using a MAX II device or a microprocessor with flash memory for configuration. In these modes, the host system must output a DCLK that is ×4 the data rate.

(3) EPCS16, EPCS64, and EPCS128 support up to a 40 MHz DCLK and are supported in Arria II GX devices. Existing batches of EPCS4 manufactured on 0.15 µm process geometry support up to a 40 MHz DCLK and are supported in Arria II GX devices. For information about product traceability and transition date to differentiate between the 0.15 µm process geometry and the 0.18 µm process geometry EPCS1 and EPCS4, refer to PCN 0514 Manufacturing Changes on EPCS Family process change notification on the Altera website at www.altera.com.

(4) JTAG-based configuration takes precedence over other configuration schemes, which means MSEL pin settings are ignored. JTAG-based configuration does not support the design security or decompression features.

(5) Do not leave MSEL pins floating. Connect them to V_{CCPD} or GND. These pins support the non-JTAG configuration scheme used in production. If you only use the JTAG configuration, Altera recommends that you connect the MSEL pins to GND.

Raw Binary File Size

Table 9–3 shows the uncompressed raw binary file (**.rbf**) configuration file sizes for Arria II GX devices.

Table 9–3. Arria II GX Uncompressed Raw Binary File (**.rbf**) Sizes *(Note 1)*

Device	Data Size (Mbits)	Data Size (MBytes)
EP2AGX20	—	—
EP2AGX30	—	—
EP2AGX45	—	—
EP2AGX65	—	—
EP2AGX95	—	—
EP2AGX125	—	—
EP2AGX190	—	—
EP2AGX260	—	—

Note to Table 9–3:

(1) These values are not available until the Quartus II software can generate the finalized raw binary file (**.rbf**).

Use the data in Table 9–3 to estimate the file size before design compilation. Different configuration file formats, such as a hexidecimal (**.hex**) or tabular text file (**.ttf**) format, have different file sizes. Refer to the Quartus II software for the different types of configuration file and file sizes. However, for any specific version of the Quartus II software, any design targeted for the same device will have the same uncompressed configuration file size. If you are using compression, the file size can vary after each compilation because the compression ratio is dependent on your design.

For more information about setting device configuration options or creating configuration files, refer to the *Device Configuration Options* and *Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

Fast Passive Parallel Configuration

Fast passive parallel (FPP) configuration in Arria II GX devices is designed to meet the continuously increasing demand for faster configuration times. Arria II GX devices are designed with the capability of receiving byte-wide configuration data per clock cycle.

You can perform FPP configuration of Arria II GX devices using an intelligent host such as a MAX II device or microprocessor.

FPP Configuration Using an External Host

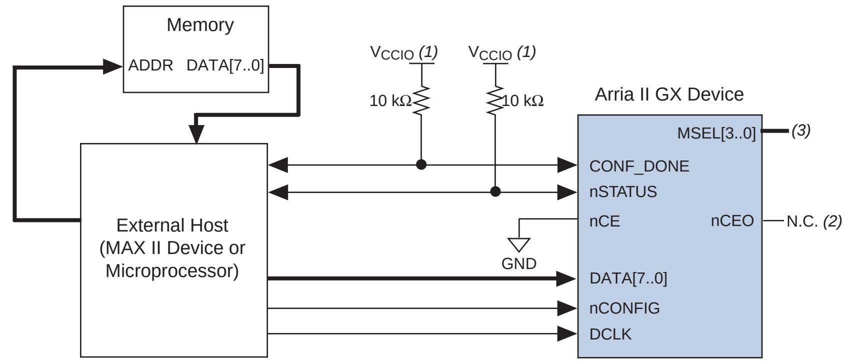
FPP configuration using compression and an external host provides the fastest method to configure Arria II GX devices. In this configuration scheme, you can use a MAX II device or microprocessor as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Arria II GX device. You can store configuration data in **.rbf**, **.hex**, or **.ttf** format. When using the MAX II device or microprocessor as an intelligent host, a design that controls the configuration process, such as fetching the data from flash memory and sending it to the device, must be stored in the MAX II device or microprocessor.

If you are using the Arria II GX decompression and/or design security features, the external host must be able to send a `DCLK` frequency that is ×4 the data rate.

The ×4 `DCLK` signal does not require an additional pin and is sent on the `DCLK` pin. The maximum `DCLK` frequency is 125 MHz, which results in a maximum data rate of 250 Mbps. If you are not using the Arria II GX decompression or design security features, the data rate is ×8 `DCLK` frequency.

Figure 9–1 shows the configuration interface connections between the Arria II GX device and a MAX II device for single device configuration.

Figure 9–1. Single Device FPP Configuration Using an External Host



Notes to Figure 9–1:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Arria II GX device. V_{CCIO} needs to be high enough to meet the V_{IH} specification of the I/O on both the device and external host. Altera recommends that you power up the configuration system's I/Os with V_{CCIO} for I/O bank 3C.

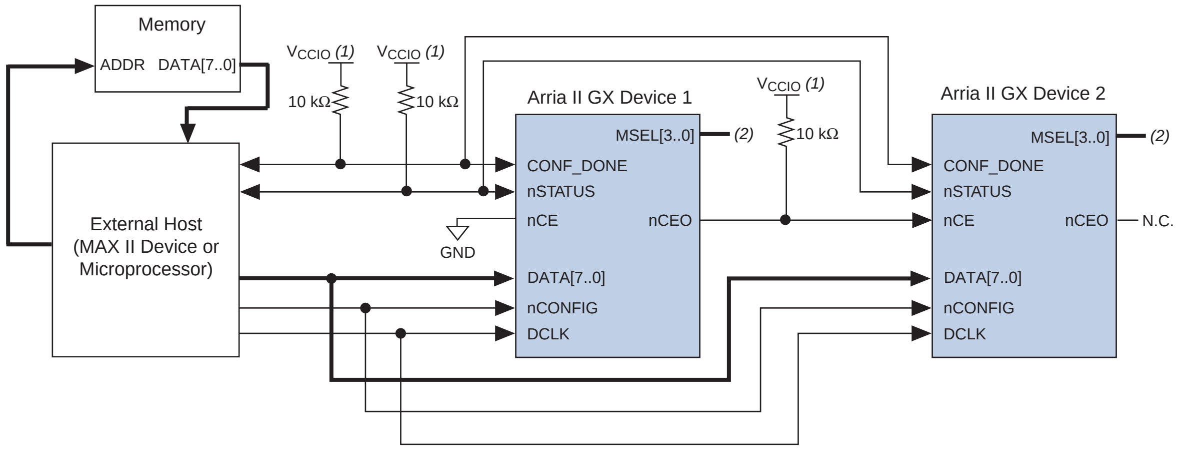
(2) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect `MSEL[3..0]`, refer to Table 9–2.

Arria II GX devices receive configuration data on the `DATA[7..0]` pins and the clock is received on the `DCLK` pin. Data is latched into the device on the rising edge of `DCLK`. If you are using the Arria II GX decompression and/or design security features, configuration data is latched on the rising edge of every fourth `DCLK` cycle. After the configuration data is latched in, it is processed during the following three `DCLK` cycles. Therefore, you can only stop `DCLK` after three clock cycles after the last data is latched into the Arria II GX Devices.

Figure 9–2 shows how to configure multiple devices using a MAX II device. This circuit is similar to the FPP configuration circuit for a single device, except the Arria II GX devices are cascaded for multi-device configuration.

Figure 9–2. Multi-Device FPP Configuration Using an External Host



Notes to Figure 9–2:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all Arria II GX devices in the chain. V_{CCIO} needs to be high enough to meet the V_{IH} specification of the I/O standard on the device and external host. Altera recommends you power up the configuration system's I/Os with V_{CCIO} for I/O bank 3C.

(2) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect `MSEL[3..0]`, refer to Table 9–2.

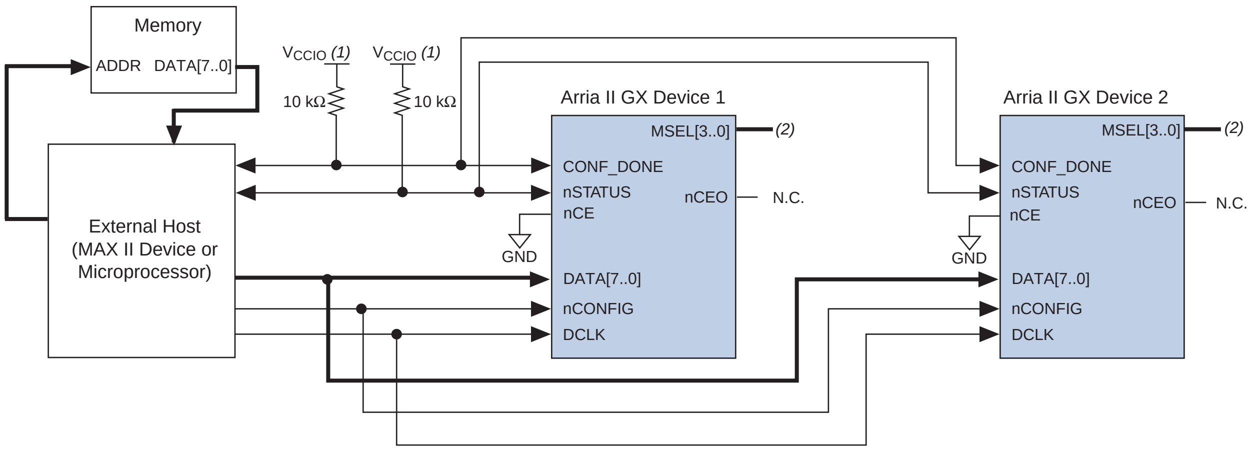
After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration in one clock cycle; therefore, the transfer of data destinations is transparent to the MAX II device or microprocessor. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

All nSTATUS and CONF_DONE pins are tied together and if any device detects an error, configuration stops for the entire chain and you must reconfigure the entire chain. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If a system has multiple devices that contain the same configuration data, tie all device nCE inputs to GND and leave the nCEO pins floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices start and complete configuration at the same time.

Figure 9–3 shows a multi-device FPP configuration when both Arria II GX devices are receiving the same configuration data.

Figure 9–3. Multiple-Device FPP Configuration Using an External Host When Both Devices Receive the Same Data



Notes to Figure 9–3:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all Arria II GX devices in the chain. V_{CCIO} needs to be high enough to meet the V_{IH} specification of the I/O standard on the device and external host. Altera recommends you power up all configuration system's I/Os with V_{CCIO} for I/O banks 3C and 8C.

(2) The MSEL pin settings vary for different configuration voltage standards and POR delay. To configure `MSEL[3..0]`, refer to Table 9–2.

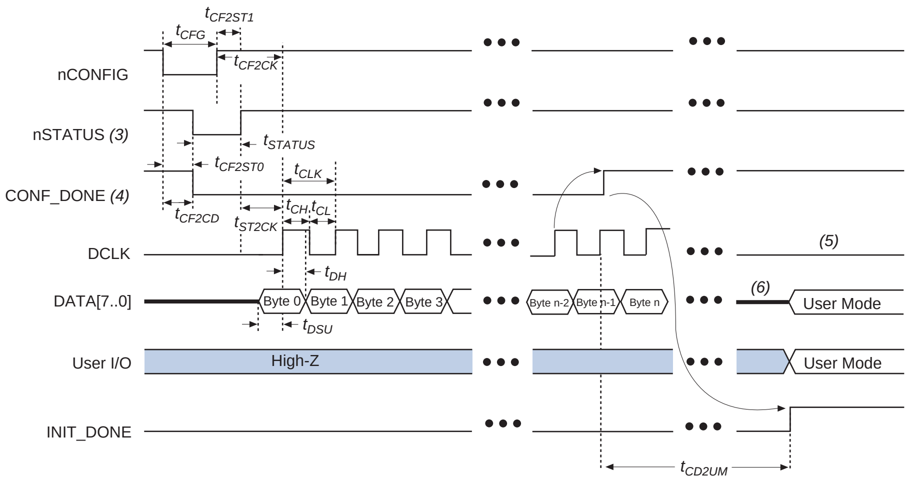
You can use a single configuration chain to configure Arria II GX devices with other Altera devices that support FPP configuration. To ensure that all devices in the chain complete configuration at the same time, or that an error flagged by one device initiates reconfiguration in all devices, tie all device `CONF_DONE` and `nSTATUS` pins together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

FPP Configuration Timing

Figure 9–4 shows the timing waveform for FPP configuration when using a MAX II device as an external host. This waveform shows timing when the decompression and design security features are not enabled.

Figure 9–4. FPP Configuration Timing Waveform with Decompression and Design Security not Enabled *(Note 1)*, *(2)*



Notes to Figure 9–4:

(1) Use this timing waveform when decompression and design security features are not used.

(2) 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.

(3) Upon power-up, the Arria II GX device holds `nSTATUS` low for the time of the POR delay.

(4) Upon power-up, before and during configuration, `CONF_DONE` is low.

(5) Do not leave `DCLK` floating after configuration. You can drive it high or low, whichever is more convenient.

(6) `DATA[7..1]` are available as user I/O pins after configuration. The state of these pins depends on the dual-purpose pin settings. `DATA[0]` is a dedicated pin that is used for both the passive and active configuration modes and is not available as a user I/O pin after configuration.

Table 9–4 defines the timing parameters for Arria II GX devices for FPP configuration when the decompression and design security features are not enabled.

Table 9–4. FPP Timing Parameters for Arria II GX Devices with Decompression and Design Security not Enabled *(Note 1)*, *(2)* (Part 1 of 2)

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	`nCONFIG` low to `CONF_DONE` low	—	800	ns
t_{CF2ST0}	`nCONFIG` low to `nSTATUS` low	—	800	ns
t_{CFG}	`nCONFIG` low pulse width	2	—	µs
t_{STATUS}	`nSTATUS` low pulse width	10	500 *(3)*	µs
t_{CF2ST1}	`nCONFIG` high to `nSTATUS` high	—	500 *(3)*	µs
t_{CF2CK}	`nCONFIG` high to first rising edge on `DCLK`	500	—	µs
t_{ST2CK}	`nSTATUS` high to first rising edge of `DCLK`	2	—	µs
t_{DSU}	Data setup time before rising edge on `DCLK`	4	—	ns
t_{DH}	Data hold time after rising edge on `DCLK`	0	—	ns
t_{CH}	`DCLK` high time	3.2	—	ns
t_{CL}	`DCLK` low time	3.2	—	ns

Table 9–4. FPP Timing Parameters for Arria II GX Devices with Decompression and Design Security not Enabled *(Note 1)*, *(2)* (Part 2 of 2)

Symbol	Parameter	Minimum	Maximum	Units
t_{CLK}	DCLK period	8	—	ns
f_{MAX}	DCLK frequency	—	125	MHz
t_R	Input rise time	—	40	ns
t	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(4)*	55	150	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (8532 × CLKUSR period)	—	—

Notes to Table 9–4:

(1) This information is preliminary.

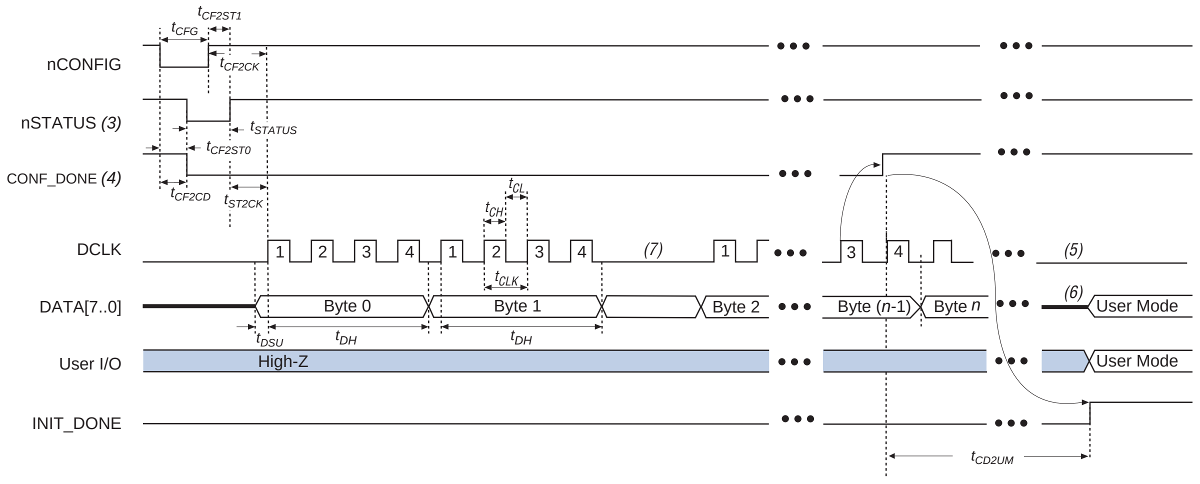
(2) Use these timing parameters when the decompression and design security features are not used.

(3) This value is obtainable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(4) The minimum and maximum numbers apply only if you chose the internal oscillator as the clock source for starting up the device.

Figure 9–5 shows the timing waveform for FPP configuration when using a MAX II device or microprocessor as an external host. This waveform shows timing when the decompression and/or design security features are enabled.

Figure 9–5. FPP Configuration Timing Waveform with Decompression or Design Security Enabled *(Note 1)*, *(2)*



Notes to Figure 9–5:

(1) Use this timing waveform when the decompression and/or design security features are used.

(2) 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.

(3) Upon power-up, the Arria II GX device holds `nSTATUS` low for the time of the POR delay.

(4) Upon power-up, before and during configuration, `CONF_DONE` is low.

(5) Do not leave `DCLK` floating after configuration. You can drive it high or low, whichever is more convenient.

(6) `DATA[7..1]` are available as user I/O pins after configuration. The state of these pins depends on the dual-purpose pin settings. `DATA[0]` is a dedicated pin that is used for both the passive and active configuration modes and is not available as a user I/O pin after configuration.

(7) If needed, you can pause `DCLK` by holding it low. When `DCLK` restarts, the external host must provide data on the `DATA[7..0]` pins prior to sending the first `DCLK` rising edge.

Table 9–5 defines the timing parameters for Arria II GX devices for FPP configuration when the decompression and/or the design security features are enabled.

Table 9–5. FPP Timing Parameters for Arria II GX Devices with the Decompression or Design Security Features Enabled *(Note 1)*, *(2)* (Part 1 of 2)

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	`nCONFIG` low to `CONF_DONE` low	—	800	ns
t_{CF2ST0}	`nCONFIG` low to `nSTATUS` low	—	800	ns
t_{CFG}	`nCONFIG` low pulse width	2	—	µs
t_{STATUS}	`nSTATUS` low pulse width	10	500 *(3)*	µs
t_{CF2ST1}	`nCONFIG` high to `nSTATUS` high	—	500 *(3)*	µs
t_{CF2CK}	`nCONFIG` high to first rising edge on `DCLK`	500	—	µs
t_{ST2CK}	`nSTATUS` high to first rising edge of `DCLK`	2	—	µs
t_{DSU}	Data setup time before rising edge on `DCLK`	4	—	ns
t_{DH}	Data hold time after rising edge on `DCLK`	24	—	ns
t_{CH}	`DCLK` high time	3.2	—	ns

Table 9–5. FPP Timing Parameters for Arria II GX Devices with the Decompression or Design Security Features Enabled *(Note 1)*, *(2)* (Part 2 of 2)

Symbol	Parameter	Minimum	Maximum	Units
t_{CL}	DCLK low time	3.2	—	ns
t_{CLK}	DCLK period	8	—	ns
f_{MAX}	DCLK frequency	—	125	MHz
t_{DATA}	Data rate	—	250	Mbps
t_R	Input rise time	—	40	ns
t	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(4)*	55	150	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (8532 × CLKUSR period)	—	—

Notes to Table 9–5:

(1) This information is preliminary.

(2) Use these timing parameters when the decompression and design security features are used.

(3) This value is obtainable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(4) The minimum and maximum numbers apply only if you choose the internal oscillator as the clock source for starting up the device.

For more information about setting device configuration options or creating configuration files, refer to the *Device Configuration Options* and *Configuration File Formats* chapters in volume 2 of the *Configuration Handbook.*

Active Serial Configuration (Serial Configuration Devices)

In the Fast AS configuration scheme, Arria II GX devices are configured using a serial configuration device. These configuration devices are low-cost devices with non-volatile memory that feature a simple four-pin interface and a small form factor. These features make serial configuration devices an ideal low-cost configuration solution.

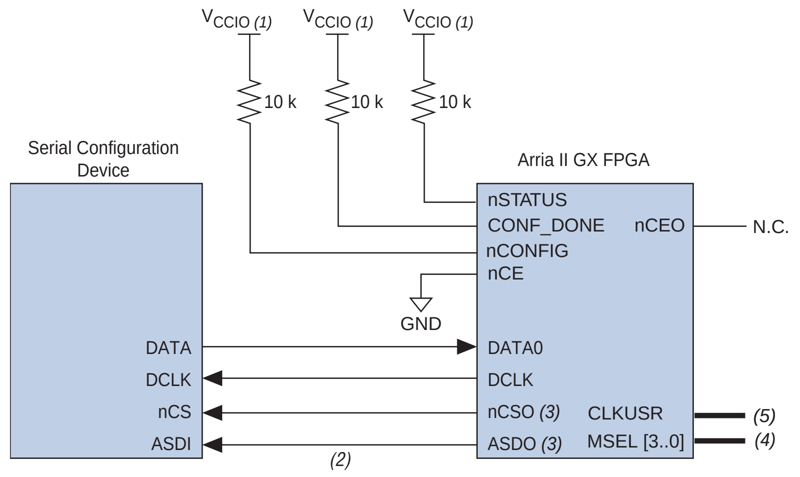
For more information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook.*

Serial configuration devices provide a serial interface to access configuration data. During device configuration, Arria II GX devices read configuration data using the serial interface, decompress data if necessary, and configure their SRAM cells. This scheme is referred to as the AS configuration scheme because the Arria II GX device controls the configuration interface. This scheme contrasts with the PS configuration scheme, where the configuration device controls the interface.

The Arria II GX decompression and design security features are fully available when configuring your Arria II GX device using AS mode.

Serial configuration devices have a four-pin interface: serial clock input (DCLK), serial data output (DATA), AS data input (ASDI), and active-low chip select (nCS). This four-pin interface connects to the Arria II GX device pins, as shown in Figure 9–6.

Figure 9–6. Single Device Fast AS Configuration



Notes to Figure 9–6:

(1) Connect the pull-up resistors to the V_{CCIO} supply of bank 3C.

(2) Arria II GX devices use the ASDO-to-ASDI path to control the configuration device.

(3) These are dual-purpose I/O pins. The FLASH_nCE pin functions as the nCSO pin in the AS configuration scheme. The DATA[1] pin functions as the ASDO pin in the AS configuration scheme.

(4) The MSEL pin settings vary for different configuration voltage standards and POR delay. To configure MSEL[3..0], refer to Table 9–2.

(5) Arria II GX devices have an option to select **CLKUSR** (40MHz maximum) as the external clock source for DCLK.

The serial clock (DCLK) generated by the Arria II GX device controls the entire configuration cycle and provides timing for the serial interface. Arria II GX devices use an internal oscillator or an external clock source to generate DCLK. You can select to use a slow clock (20 MHz maximum) or a fast clock (40 MHz maximum) from the internal oscillator. Arria II GX devices have an option to select **CLKUSR** (40 MHz maximum) as the external clock source for DCLK.

In AS configuration schemes, Arria II GX devices drive out control signals on the falling edge of DCLK. The serial configuration device responds to the instructions by driving out configuration data on the falling edge of DCLK. Then the data is latched into the Arria II GX device on the following falling edge of DCLK.

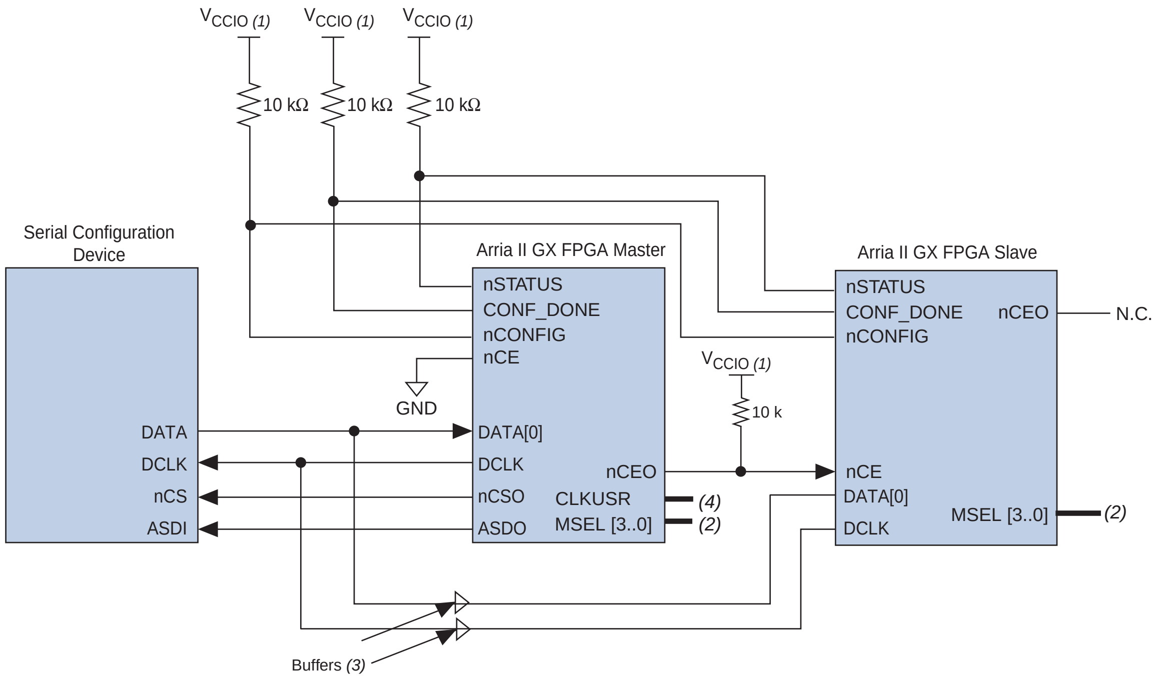
In configuration mode, Arria II GX devices enable the serial configuration device by driving the nCSO output pin low, which connects to the chip select (nCS) pin of the configuration device. The Arria II GX device uses the serial clock (DCLK) and serial data output (ASDO) pins to send operation commands and/or read address signals to the serial configuration device. The configuration device provides data on its serial data output (DATA) pin, which connects to the DATA0 input of the Arria II GX devices.

You can configure multiple Arria II GX devices using a single serial configuration device. You can cascade multiple Arria II GX devices using the chip-enable (nCE) and chip-enable-out (nCEO) pins. The first device in the chain must have its nCE pin connected to GND. You must connect its nCEO pin to the nCE pin of the next device in the chain. When the first device captures all its configuration data from the bitstream, it drives the nCEO pin low, enabling the next device in the chain. You must leave the nCEO pin of the last device unconnected. The nCONFIG, nSTATUS, CONF_DONE, DCLK, and DATA0 pins of each device in the chain are connected (refer to Figure 9–7).

The first Arria II GX device in the chain is the configuration master and controls configuration of the entire chain. You must connect its MSEL pins to select the AS configuration scheme. The remaining Arria II GX devices are configuration slaves. You must connect their MSEL pins to select the PS configuration scheme. Any other Altera device that supports PS configuration can also be part of the chain as a configuration slave.

Figure 9–7 shows the pin connections for the multi-device AS configuration.

Figure 9–7. Multi-Device AS Configuration



Notes to Figure 9–7:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the I/O bank 3C.

(2) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0], refer to Table 9–2.

(3) Connect the repeater buffers between the Arria II GX master and slave devices for DATA[0] and DCLK. This is to prevent any potential signal integrity and clock skew problems.

(4) Arria II GX devices have an option to select **CLKUSR** (40MHz maximum) as the external clock source for DCLK.

As shown in Figure 9–7, the nSTATUS and CONF_DONE pins on all target devices are connected together with external pull-up resistors. These pins are open-drain bidirectional pins on the devices. When the first device asserts nCEO (after receiving all its configuration data), it releases its CONF_DONE pin. But the subsequent devices in the chain keep this shared CONF_DONE line low until they have received their configuration data. When all target devices in the chain have received their configuration data and have released CONF_DONE, the pull-up resistor drives a high level on this line and all devices simultaneously enter initialization mode.

☞ While you can cascade Arria II GX devices, you cannot cascade or chain together serial configuration devices.

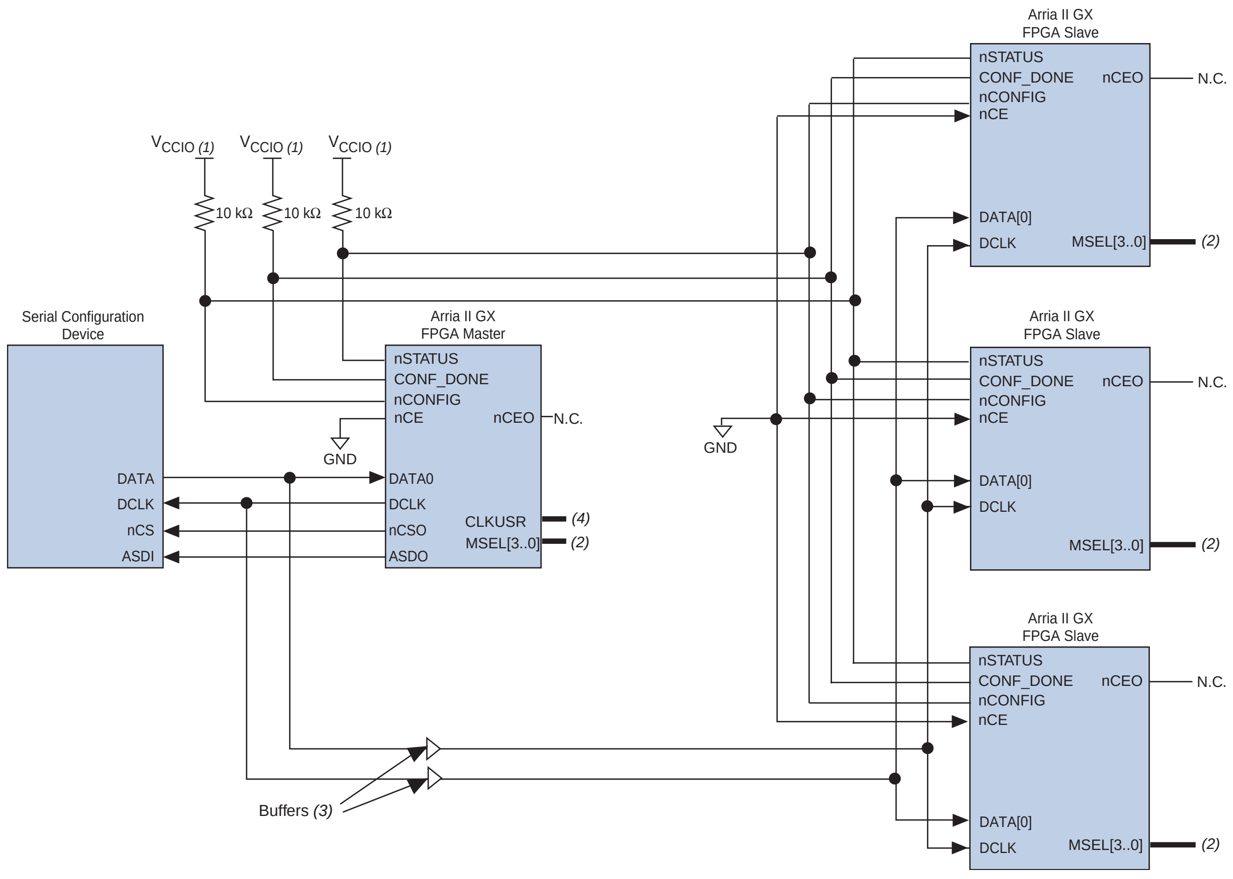
If the configuration bitstream size exceeds the capacity of a serial configuration device, you must select a larger configuration device and/or enable the compression feature. When configuring multiple devices, the size of the bitstream is the sum of the individual devices' configuration bitstreams.

A system may have multiple devices that contain the same configuration data. In active serial chains, you can implement this by storing one copy of the SRAM object file (**.sof**) in the serial configuration device. The same copy of the **.sof** file configures the master Arria II GX device and all remaining slave devices concurrently. All Arria II GX devices must be the same density and package.

To configure four identical Arria II GX devices with the same **.sof** file, you can set up the chain similar to the example shown in Figure 9–8. The first device is the master device and its MSEL pins need to be set to select AS configuration. The other three slave devices are set up for concurrent configuration and their MSEL pins need to be set to select PS configuration. The nCE input pins from the master and slave are connected to GND, and the DATA and DCLK pins connect in parallel to all four devices. During the configuration cycle, the master device reads its configuration data from the serial configuration device and transmits the configuration data to all three slave devices, configuring all of them simultaneously.

Figure 9–8 shows the multi-device AS configuration when the devices receive the same data using a single **.sof** file.

Figure 9–8. Multi-Device AS Configuration When the Devices Receive the Same Data Using a Single .sof File



Notes to Figure 9–8:

(1) Connect the pull-up resistors to the V_{CCIO} supply of I/O bank 3C.

(2) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect `MSEL[3..0]`, refer to Table 9–2.

(3) Connect the repeater buffers between the Arria II GX master and slave devices for `DATA[0]` and `DCLK`. This is to prevent any potential signal integrity and clock skew problems.

(4) Arria II GX devices have an option to select **CLKUSR** (40MHz maximum) as the external clock source for `DCLK`.

Estimating Active Serial Configuration Time

Active serial configuration time is dominated by the time it takes to transfer data from the serial configuration device to the Arria II GX device. This serial interface is clocked by the Arria II GX `DCLK` output (generated from an internal oscillator or an option to select **CLKUSR** as external clock source). Arria II GX devices support `DCLK` up to 40 MHz (25 ns).

Therefore, the minimum configuration time can be estimated as the following:

RBF Size × (minimum `DCLK` period / 1 bit per `DCLK` cycle) = estimated minimum configuration time

Enabling compression reduces the amount of configuration data that is transmitted to the Arria II GX device, which also reduces configuration time. On average, compression reduces configuration time, depending on your design.

Programming Serial Configuration Devices

Serial configuration devices are non-volatile, flash-memory-based devices. You can program these devices in-system using an USB-Blaster™, EthernetBlaster, or ByteBlaster™ II download cable. Alternatively, you can program them using the Altera programming unit (APU), supported third-party programmers, or a microprocessor with the SRunner software driver.

You can perform in-system programming of serial configuration devices using the conventional AS programming interface or JTAG interface solution.

Because serial configuration devices do not support the JTAG interface, the conventional method to program them is using the AS programming interface. The configuration data used to program serial configuration devices is downloaded using programming hardware.

During in-system programming, the download cable disables device access to the AS interface by driving the `nCE` pin high. Arria II GX devices are also held in reset mode by a low level on `nCONFIG`. After programming is complete, the download cable releases `nCE` and `nCONFIG`, allowing the pull-down and pull-up resistors to drive GND and V_{CCIO}, respectively.

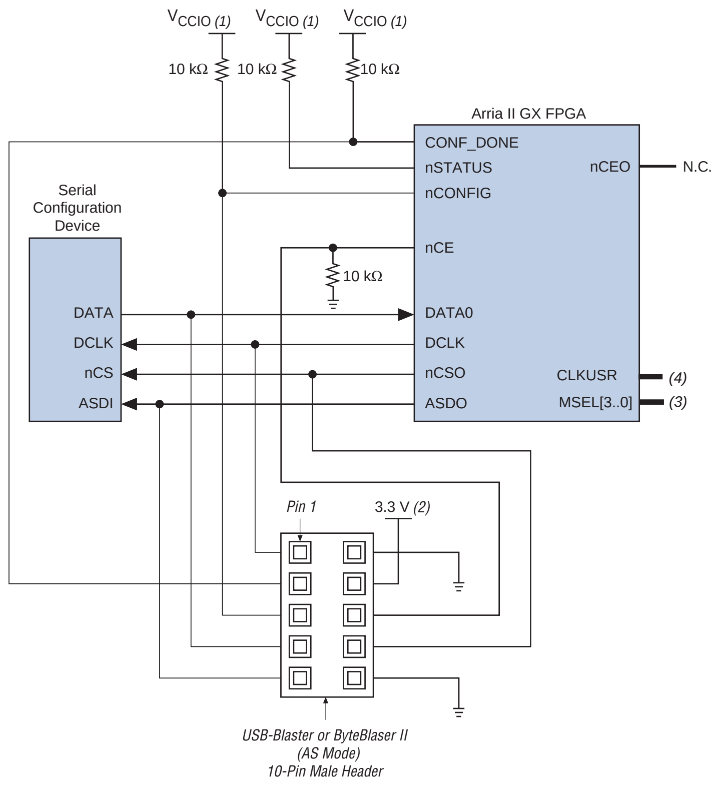
Altera has developed Serial FlashLoader (SFL); an in-system programming solution for serial configuration devices using the JTAG interface. This solution requires the Arria II GX device to be a bridge between the JTAG interface and the serial configuration device.

For more information about SFL, refer to *AN 370: Using the Serial FlashLoader with Quartus II Software*.

For more information about the USB-Blaster download cable, refer to the *USB-Blaster Download Cable User Guide*. For more information about the ByteBlaster II cable, refer to the *ByteBlaster II Download Cable User Guide*. For more information about the EthernetBlaster download cable, refer to the *EthernetBlaster Communications Cable User Guide*.

Figure 9–9 shows the download cable connections to the serial configuration device.

Figure 9–9. In-System Programming of Serial Configuration Devices



Notes to Figure 9–9:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the I/O bank 3C.

(2) Power up the USB-ByteBlaster, ByteBlaster II, or EthernetBlaster cable's V_{CC} (TRGT) with 3.3 V.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0], refer to Table 9–2.

(4) Arria II GX devices have an option to select **CLKUSR** (40 MHz maximum) as the external clock source for DCLK.

You can program serial configuration devices with the Quartus II software using the Altera programming hardware and the appropriate configuration device programming adapter.

In production environments, you can program serial configuration devices using multiple methods. You can use Altera programming hardware or other third-party programming hardware to program blank serial configuration devices before they are mounted onto PCBs. Alternatively, you can use an on-board microprocessor to program the serial configuration device in-system using C-based software drivers provided by Altera.

You can program a serial configuration device in-system by an external microprocessor using SRunner. SRunner is a software driver developed for embedded serial configuration device programming, which can be easily customized to fit in different embedded systems. SRunner is able to read a raw programming data (**.rpd**) file and write to serial configuration devices. The serial configuration device programming time using SRunner is comparable to the programming time with the Quartus II software.

For more information about SRunner, refer to *AN 418: SRunner: An Embedded Solution for EPCS Programming* and the source code on the Altera website at www.altera.com.

For more information about programming serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

Passive Serial Configuration

You can program PS configuration of Arria II GX devices using an intelligent host, such as a MAX II device or microprocessor with flash memory, or a download cable. In the PS scheme, an external host (a MAX II device, embedded processor, or host PC) controls configuration. Configuration data is clocked into the target Arria II GX device using the `DATA0` pin at each rising edge of `DCLK`.

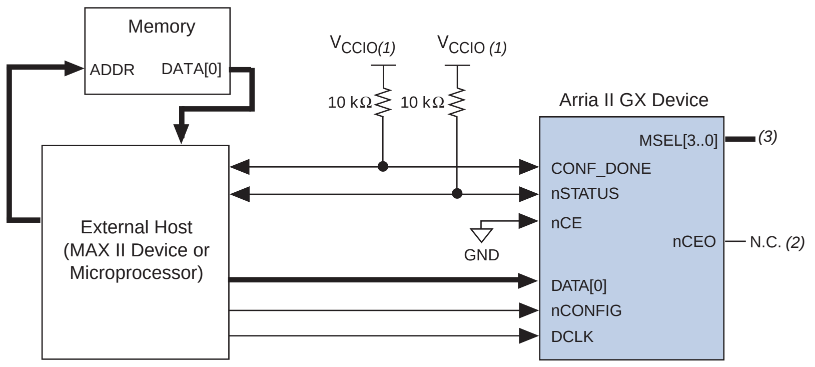
The Arria II GX decompression and design security features are fully available when configuring your Arria II GX device using PS mode.

PS Configuration Using an External Host

In this configuration scheme, you can use a MAX II device as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Arria II GX device. You can store configuration data in **.rbf**, **.hex**, or **.ttf** format.

Figure 9–10 shows the configuration interface connections between an Arria II GX device and a MAX II device for single device configuration.

Figure 9–10. Single Device PS Configuration Using an External Host



Note to Figure 9–10:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Arria II GX device. V_{CCIO} needs to be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends that you power up the configuration system's I/Os with V_{CCIO} for I/O bank 3C.

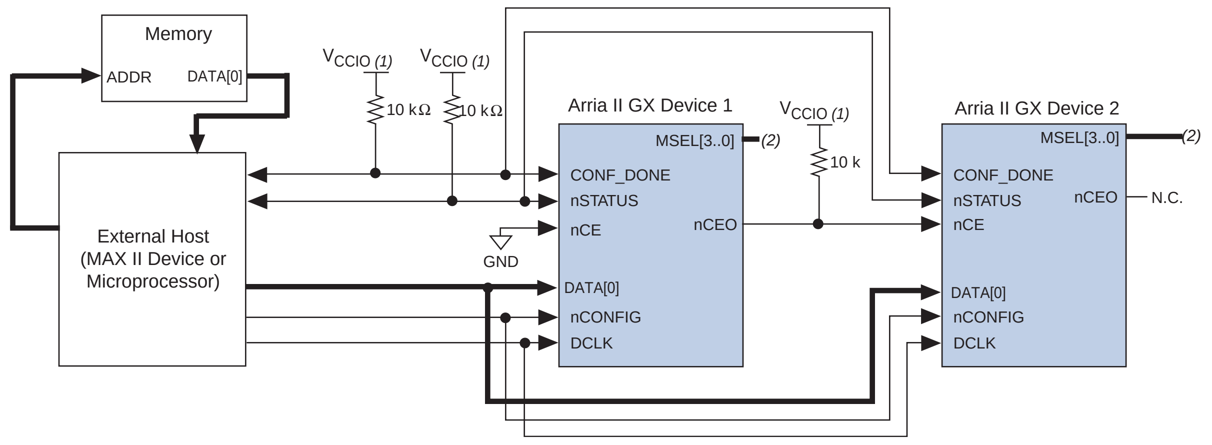
(2) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect `MSEL[3..0]`, refer to Table 9–2.

The Arria II GX device receives configuration data on the `DATA0` pin and the clock is received on the `DCLK` pin. Data is latched into the device on the rising edge of `DCLK`. If you are using configuration data in **.rbf**, **.hex**, or **.ttf** format, you must send the LSB of each data byte first. For example, if the **.rbf** file contains the byte sequence 02 1B EE 01 FA, the serial bitstream you should transmit to the device is `0100-0000 1101-1000 0111-0111 1000-0000 0101-1111`.

Figure 9–11 shows how to configure multiple devices using an external host. This circuit is similar to the PS configuration circuit for a single device, except Arria II GX devices are cascaded for multi-device configuration.

Figure 9–11. Multi-Device PS Configuration Using an External Host



Notes to Figure 9–11:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all Arria II GX devices in the chain. V_{CCIO} needs to be high enough to meet the V_{IH} specification of the I/O standard on the device and the external host. Altera recommends you power up the configuration system's I/Os with V_{CCIO} for I/O bank 3C.

(2) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect `MSEL[3..0]`, refer to Table 9–2.

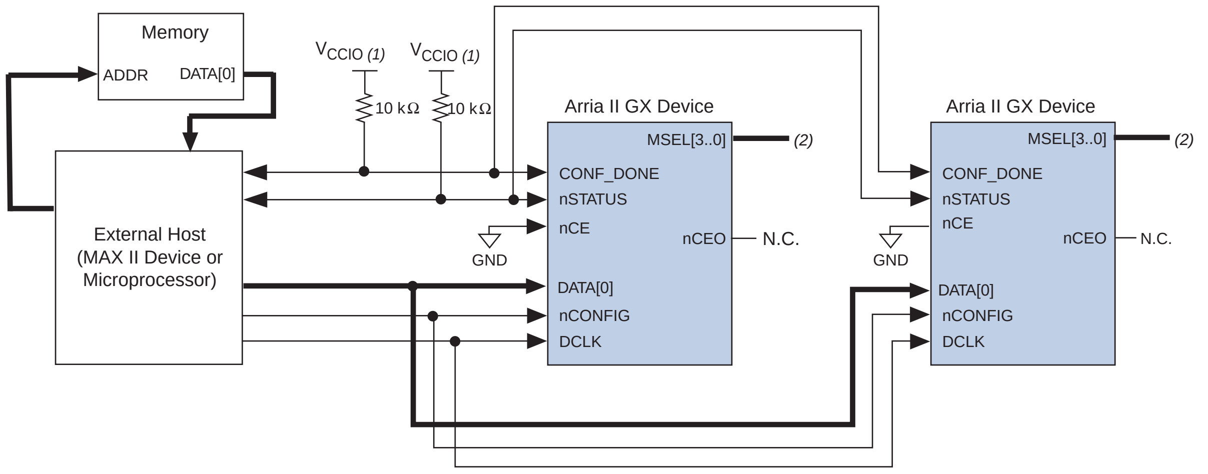
After the first device completes configuration in a multi-device configuration chain, its `nCEO` pin drives low to activate the second device's `nCE` pin, which prompts the second device to begin configuration. The second device in the chain begins configuration in one clock cycle. Therefore, the transfer of data destinations is transparent to the MAX II device or microprocessor. All other configuration pins (`nCONFIG`, `nSTATUS`, `DCLK`, `DATA0`, and `CONF_DONE`) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the `DCLK` and `DATA` lines are buffered for every fourth device. Because all device `CONF_DONE` pins are tied together, all devices initialize and enter user mode at the same time.

Because all `nSTATUS` and `CONF_DONE` pins are tied together, if any device detects an error, configuration stops for the entire chain and you must reconfigure the entire chain. For example, if the first device flags an error on `nSTATUS`, it resets the chain by pulling its `nSTATUS` pin low. This behavior is similar to a single device detecting an error.

In your system, you can have multiple devices that contain the same configuration data. To support this configuration scheme, all device `nCE` inputs are tied to GND, while `nCEO` pins are left floating. All other configuration pins (`nCONFIG`, `nSTATUS`, `DCLK`, `DATA0`, and `CONF_DONE`) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the `DCLK` and `DATA` lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time.

Figure 9–12 shows multi-device PS configuration when both Arria II GX devices are receiving the same configuration data.

Figure 9–12. Multiple-Device PS Configuration When Both Devices Receive the Same Data



Notes to Figure 9–12:

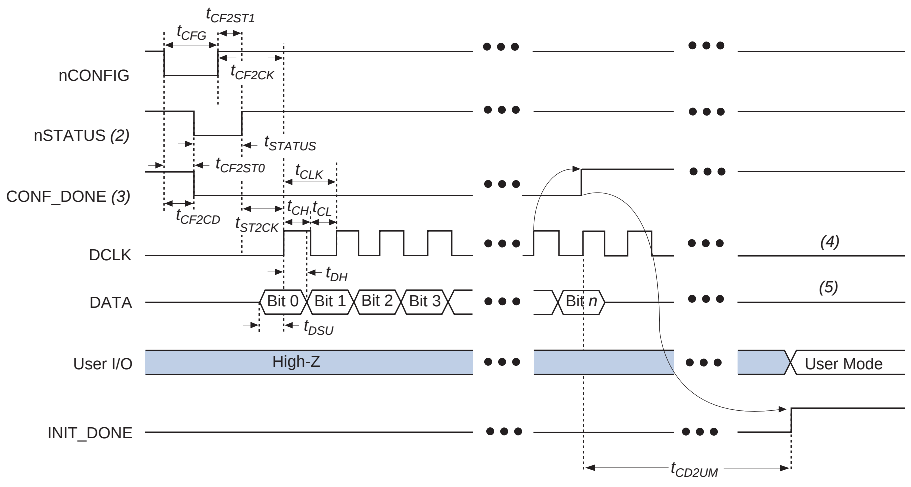
(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all Arria II GX devices in the chain. V_{CCIO} needs to be high enough to meet the V_{IH} specification of the I/O standard on the device and the external host. Altera recommends you power up all configuration systems I/Os with V_{CCIO} for I/O bank 3C and 8C.

(2) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect `MSEL[3..0]`, refer to Table 9–2.

PS Configuration Timing

Figure 9–13 shows the timing waveform for PS configuration when using a MAX II device or microprocessor as an external host.

Figure 9–13. PS Configuration Timing Waveform *(Note 1)*



Notes to Figure 9–13:

(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 Arria II GX device holds `nSTATUS` low for the time of the POR delay.
(3) Upon power-up, before and during configuration, `CONF_DONE` is low.
(4) Do not leave `DCLK` floating after configuration. You can drive it high or low, whichever is more convenient.
(5) `DATA[0]` is available as a user I/O pin after configuration. The state of this pin depends on the dual-purpose pin settings.

Table 9–6 defines the timing parameters for Arria II GX devices for PS configuration.

Table 9–6. PS Timing Parameters for Arria II GX Devices *(Note 1)* (Part 1 of 2)

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	`nCONFIG` low to `CONF_DONE` low	—	800	ns
t_{CF2ST0}	`nCONFIG` low to `nSTATUS` low	—	800 *(2)*	ns
t_{CFG}	`nCONFIG` low pulse width	2	—	µs
t_{STATUS}	`nSTATUS` low pulse width	10	500 *(2)*	µs
t_{CF2ST1}	`nCONFIG` high to `nSTATUS` high	—	500	µs
t_{CF2CK}	`nCONFIG` high to first rising edge on `DCLK`	500	—	µs
t_{ST2CK}	`nSTATUS` high to first rising edge of `DCLK`	2	—	µs
t_{DSU}	Data setup time before rising edge on `DCLK`	4	—	ns
t_{DH}	Data hold time after rising edge on `DCLK`	0	—	ns
t_{CH}	`DCLK` high time	3.2	—	ns
t_{CL}	`DCLK` low time	3.2	—	ns
t_{CLK}	`DCLK` period	8	—	ns
f_{MAX}	`DCLK` frequency	—	125	MHz
t_R	Input rise time	—	40	ns

Table 9–6. PS Timing Parameters for Arria II GX Devices *(Note 1)* (Part 2 of 2)

Symbol	Parameter	Minimum	Maximum	Units
t	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(3)*	55	150	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (8532 CLKUSR period)	—	—

Notes to Table 9–6:

(1) This information is preliminary.

(2) This value is applicable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(3) The minimum and maximum numbers apply only if you choose the internal oscillator as the clock source for starting the device.

For more information about device configuration options and how to create configuration files, refer to the *Device Configuration Options* and *Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

PS Configuration Using a Download Cable

In this section, the generic term "download cable" includes the Altera USB-Blaster universal serial bus (USB) port download cable, ByteBlaster II parallel port download cable, ByteBlasterMV™ parallel port download cable, and EthernetBlaster download cable.

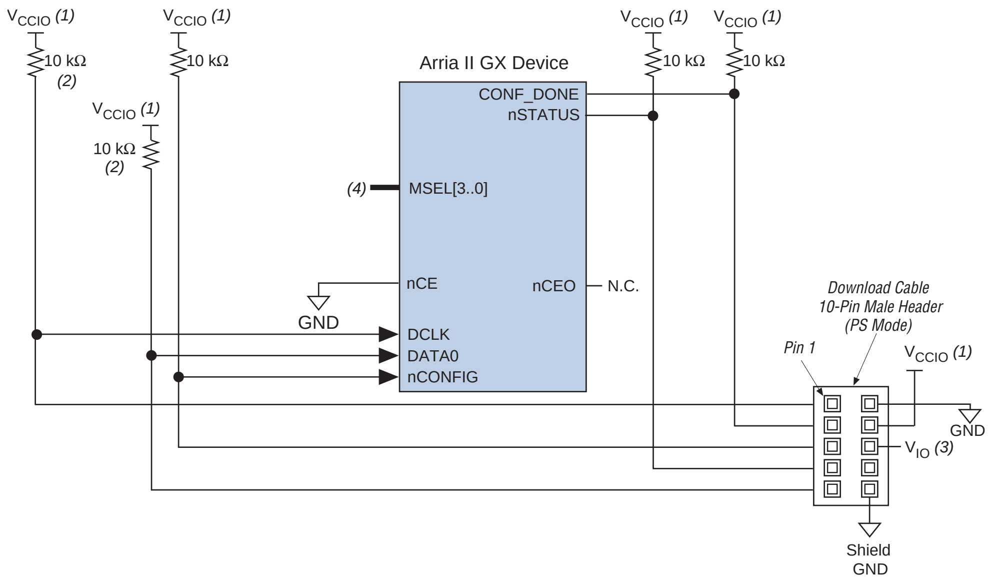
In a PS configuration with a download cable, an intelligent host (such as a PC) transfers data from a storage device to the Arria II GX device using the download cable.

During configuration, the programming hardware or download cable places the configuration data one bit at a time on the device's DATA0 pin. The configuration data is clocked into the target device until CONF_DONE goes high.

When using a download cable, setting the **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart the configuration in the Quartus II software when an error occurs. Additionally, the **Enable user-supplied start-up clock** (**CLKUSR**) option has no affect on the device initialization because this option is disabled in the **.sof** file when programming the device using the Quartus II programmer and 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 device with the Quartus II programmer and a download cable.

Figure 9–14 shows PS configuration for Arria II GX devices using a USB-Blaster, ByteBlaster II, ByteBlasterMV, or Ethernet Blaster cable.

Figure 9–14. PS Configuration Using a USB-Blaster, EthernetBlaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV Cable



Notes to Figure 9–14:

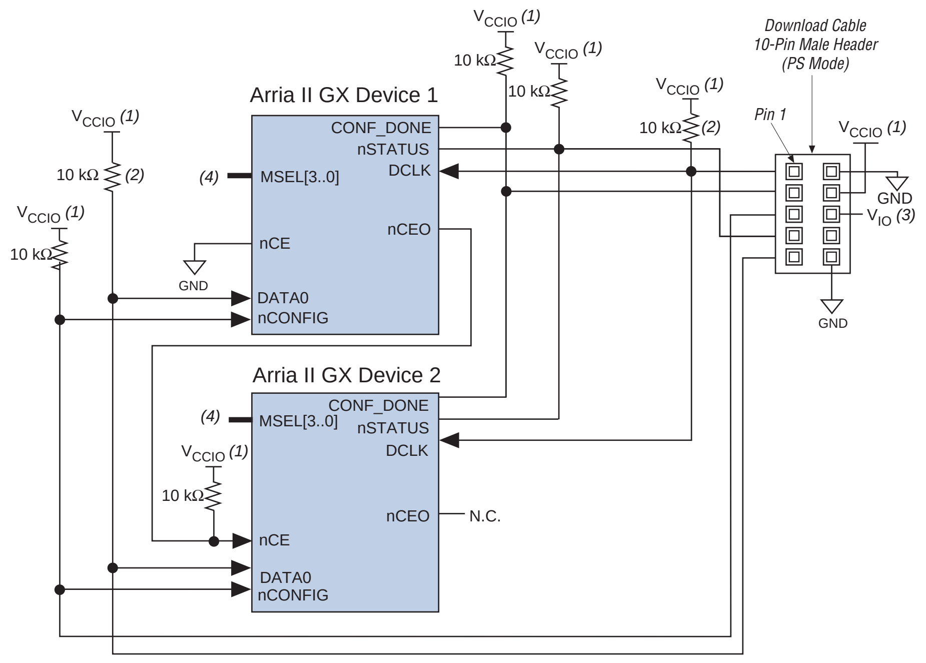
(1) Connect the pull-up resistor to the same supply voltage (V_{CCIO}) as the USB-Blaster, ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable.
(2) You only need the pull-up resistors on `DATA0` and `DCLK` if the download cable is the only configuration scheme used on your board. This ensures that `DATA0` and `DCLK` are not left floating after configuration. For example, if you are also using a configuration device, you do not need the pull-up resistors on `DATA0` and `DCLK`.
(3) In the ByteBlasterMV cable, pin 6 is a no connect. In the USB-Blaster and ByteBlaster II cables, this pin is connected to `nCE` when it is used for active serial programming; otherwise, it is a no connect.
(4) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect `MSEL[3..0]`, refer to Table 9–2.

You can use a download cable to configure multiple Arria II GX devices by connecting each device's `nCEO` pin to the subsequent device's `nCE` pin. The first device's `nCE` pin is connected to GND while its `nCEO` pin is connected to the `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. All other configuration pins (`nCONFIG`, `nSTATUS`, `DCLK`, `DATA0`, and `CONF_DONE`) are connected to every 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. The **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs.

Figure 9–15 shows how to configure multiple Arria II GX devices with a download cable.

Figure 9–15. Multi-Device PS Configuration Using a USB-Blaster, ByteBlaster II, ByteBlasterMV, or EthernetBlaster Cable



Notes to Figure 9–15:

(1) Connect the pull-up resistor to the same supply voltage (V_{CCIO}) as the USB-Blaster, ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable.
(2) You only need the pull-up resistors on `DATA0` and `DCLK` if the download cable is the only configuration scheme used on your board. This ensures that `DATA0` and `DCLK` are not left floating after configuration. For example, if you are also using a configuration device, you do not need the pull-up resistors on `DATA0` and `DCLK`.
(3) In the ByteBlasterMV cable, pin 6 is a no connect. In the USB-Blaster and ByteBlaster II cables, this pin is connected to `nCE` when it is used for active serial programming; otherwise, it is a no connect.
(4) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect `MSEL[3..0]`, refer to Table 9–2.

For more information about how to use the USB-Blaster, ByteBlaster II, ByteBlasterMV, or EthernetBlaster cables, refer to the following user guides:

- USB-Blaster Download Cable User Guide
- ByteBlaster II Download Cable User Guide
- ByteBlasterMV Download Cable User Guide
- Ethernet Blaster Communications Cable User Guide

JTAG 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 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 JTAG circuitry to shift configuration data into the device. The Quartus II software automatically generates **.sof** files that you can use for JTAG configuration with a download cable in the Quartus II software programmer.

For more information about JTAG boundary-scan testing and commands available using Arria II GX devices, refer to the following documents:

- *JTAG Boundary Scan Testing* chapter in volume 1 of the *Arria II GX Device Handbook*
- Programming Support for Jam STAPL Language

Arria II GX devices are designed such that JTAG instructions have precedence over any device configuration modes. Therefore, JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of Arria II GX devices during PS configuration, PS configuration is terminated and JTAG configuration begins.

You cannot use the Arria II GX decompression or design security features if you are configuring your Arria II GX device using JTAG-based configuration.

A device operating in JTAG mode uses four required pins, `TDI`, `TDO`, `TMS`, and `TCK`. The `TCK` pin has an internal weak pull-down resistor, while the `TDI and TMS` pins have weak internal pull-up resistors (typically 25 kΩ). All JTAG input pins are powered by the V_{CCIO} supply of I/O bank 8C. The `TDO` pin is powered up by the V_{CCIO} and V_{CCPD} power supply of I/O bank 8C. All the JTAG pins support only the LVTTL I/O standard.

All user I/O pins are tri-stated during JTAG configuration. Table 9–7 explains each JTAG pin's function.

For recommendations about how to connect a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary Scan Testing* chapter in volume 1 of the *Arria II GX Device Handbook.*

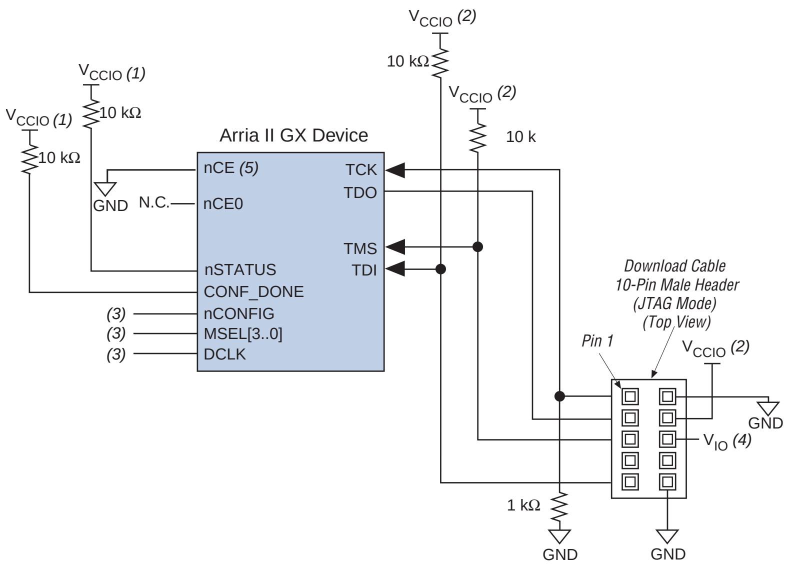
Table 9–7. Dedicated JTAG Pins

Pin Name	Pin Type	Description
`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`. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by connecting this pin to logic high.
`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. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by leaving this pin unconnected.
`TMS`	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. `TMS` is evaluated on the rising edge of `TCK`. Therefore, you must set up `TMS` before the rising edge of `TCK`. Transitions in the state machine occur on the falling edge of `TCK` after the signal is applied to `TMS`. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by connecting this pin to logic high.
`TCK`	Test clock input	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, you can disable the JTAG circuitry by connecting this pin to GND.

During JTAG configuration, you can download data to the device on the PCB through the USB-Blaster, ByteBlaster II, ByteBlasterMV, or EthernetBlaster download cable.

Figure 9–16 shows the JTAG configuration of a single Arria II GX device.

Figure 9–16. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 9–16:

(1) Connect the pull-up resistors to the V_{CCIO} supply of I/O bank 3C.

(2) Connect the pull-up resistor to the same supply voltage as the USB-Blaster, ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable. The voltage supply can be connected to the V_{CCIO} supply of I/O bank 8C of the device.

(3) Connect the `nCONFIG` and `MSEL[3..0]` pins to support a non-JTAG configuration scheme. If you only use the JTAG configuration, connect `nCONFIG` to V_{CCIO}, and `MSEL[3..0]` to GND. Pull `DCLK` either high or low, whichever is convenient on your board.

(4) In the USB-Blaster, ByteBlaster II, and ByteBlasterMV cables, this pin is a no connect.

(5) You must connect `nCE` to GND or drive it low for successful JTAG configuration.

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.

The Quartus II software verifies successful JTAG configuration upon completion. At the end of configuration, the software checks the state of `CONF_DONE` through the JTAG port. When the Quartus II software generates a Jam™ file (**.jam)** for a multi-device chain, it contains instructions so that all the devices in the chain are initialized at the same time. If `CONF_DONE` is not high, the Quartus II software indicates that configuration has failed. If `CONF_DONE` is high, the software indicates that configuration was successful. After the configuration bitstream is transmitted serially using the JTAG `TDI` port, the `TCK` port is clocked an additional 1,094 cycles to perform device initialization.

Arria II GX devices have dedicated JTAG pins that always function as JTAG pins. Not only can you perform JTAG testing on Arria II GX devices before and after, but also during configuration. While other device families do not support JTAG testing during configuration, Arria II GX devices support the bypass, ID code, and sample instructions during configuration without interrupting configuration. All other JTAG instructions may only be issued by first interrupting configuration and reprogramming I/O pins using the `CONFIG_IO` instruction.

The `CONFIG_IO` instruction allows I/O buffers to be configured using the JTAG port and when issued, interrupts configuration. This instruction allows you to perform board-level testing prior to configuring the Arria II GX device or waiting for a configuration device to complete configuration. Once configuration is interrupted and JTAG testing is complete, you must reconfigure the part using JTAG (`PULSE_CONFIG` instruction) or by pulsing `nCONFIG` low.

The chip-wide reset (`DEV_CLRn`) and chip-wide output enable (`DEV_OE`) pins on Arria II 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 Arria II GX devices, consider the dedicated configuration pins. Table 9–8 shows how these pins must be connected during JTAG configuration.

Table 9–8. Dedicated Configuration Pin Connections During JTAG Configuration

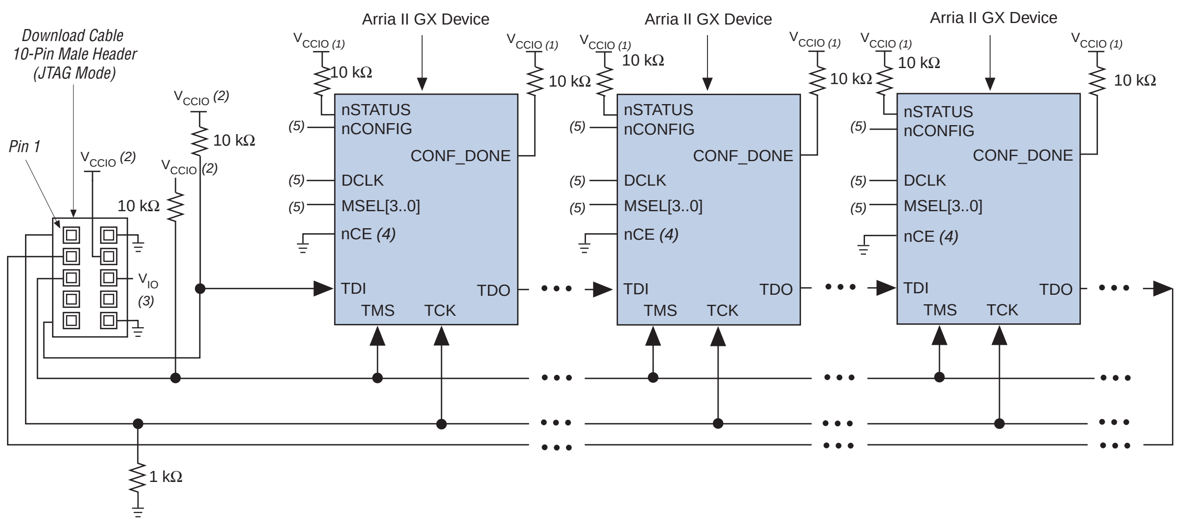
Signal	Description
`nCE`	On all Arria II GX devices in the chain, `nCE` must be driven low by connecting it to ground, pulling it low using a resistor, or driving it by some control circuitry. For devices that are also in multi-device FPP, AS, or PS configuration chains, the `nCE` pins must be connected to GND during JTAG configuration, or JTAG must be configured in the same order as the configuration chain.
`nCEO`	On all Arria II GX devices in the chain, you can leave `nCEO` floating or connected to `nCE` of the next device.
`MSEL`	These pins must not be left floating. These pins support whichever non-JTAG configuration is used in production. If you only use JTAG configuration, tie these pins to GND.
`nCONFIG`	Driven high by connecting to the V_{CCIO} supply of the bank in which the pin resides, pulling up using a resistor, or driven high by some control circuitry.
`nSTATUS`	Pull to the V_{CCIO} supply of the bank in which the pin resides using a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each `nSTATUS` pin must be pulled up to V_{CCIO} individually.
`CONF_DONE`	Pull to the V_{CCIO} supply of the bank in which the pin resides using a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each `CONF_DONE` pin must be pulled up to the V_{CCIO} supply of the bank in which the pin resides individually. `CONF_DONE` going high at the end of JTAG configuration indicates successful configuration.
`DCLK`	Do not leave floating. Drive low or high, whichever is more convenient on your board.

When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more 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 system contains multiple devices or when testing your system using JTAG BST circuitry.

Figure 9–17 shows a multi-device JTAG configuration.

Figure 9–17. JTAG Configuration of Multiple Devices Using a Download Cable



Notes to Figure 9–17:

(1) Connect the pull-up resistors to the V_{CCIO} supply of I/O bank 3C.

(2) You must connect the pull-up resistor to the same supply voltage as the USB-Blaster, ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable. You can connect the voltage supply to the V_{CCIO} supply of I/O bank 8C of the device.

(3) In the USB-Blaster, ByteBlaster II, and ByteBlasterMV cables, pin 6 is a no connect.

(4) You must connect `nCE` to GND or drive it low for successful JTAG configuration.

(5) You must connect the `nCONFIG` and `MSEL[3..0]` pins to support a non-JTAG configuration scheme. If you only use JTAG configuration, connect `nCONFIG` to the V_{CCIO} supply of the bank in which the pin resides and `MSEL[3..0]` to GND. Pull `DCLK` either high or low, whichever is convenient on your board.

You must connect the `nCE` pin to GND or drive it low during JTAG configuration. In multi-device FPP, AS, and PS 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. In addition, the `CONF_DONE` and `nSTATUS` signals are all shared in multi-device FPP, AS, or PS configuration chains so the devices can enter user mode at the same time after configuration is complete. When the `CONF_DONE` and `nSTATUS` signals are shared among all the devices, you must configure every device when JTAG configuration is performed.

If you only use JTAG configuration, Altera recommends that you connect the circuitry as shown in Figure 9–17, where each of the `CONF_DONE` and `nSTATUS` signals are isolated to enable each device to enter user mode individually.

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, 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 the `nCE` of the next device low when it has successfully been JTAG configured.

You can place other Altera devices that have JTAG support in the same JTAG chain for device programming and configuration.

JTAG configuration support is enhanced and allows more than 17 Arria II GX devices to be cascaded in a JTAG chain.

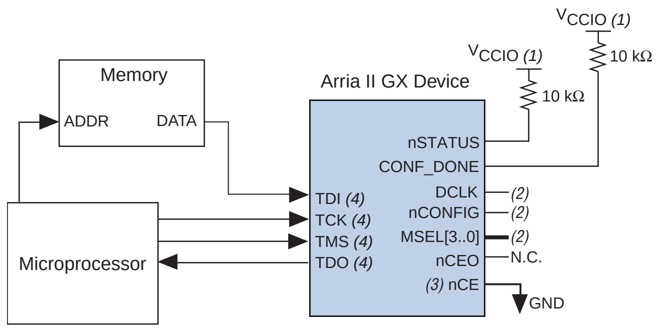
For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera Device Chains* chapter in volume 2 of the *Configuration Handbook*.

You can configure Arria II GX devices using multiple configuration schemes on the same board. Combining JTAG configuration with passive serial or active serial configuration on your board is useful in the prototyping environment because it allows multiple methods to configure your FPGA.

For more information about combining JTAG configuration with other configuration schemes, refer to the *Combining Different Configuration Schemes* chapter in volume 2 of the *Configuration Handbook*.

Figure 9–18 shows a JTAG configuration of an Arria II GX device using a microprocessor.

Figure 9–18. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 9–18:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all Arria II GX devices in the chain. The V_{CCIO} supply of the bank in which the pin resides needs to be high enough to meet the V_{IH} specification of the I/O on the device.

(2) Connect the `nCONFIG` and `MSEL[3..0]` pins to support a non-JTAG configuration scheme. If you use only the JTAG configuration, connect `nCONFIG` to the V_{CCIO} supply of the bank in which the pin resides and `MSEL[3..0]` to GND. Pull `DCLK` either high or low, whichever is convenient on your board.

(3) You must connect `nCE` to GND or drive it low for successful JTAG configuration.

(4) The microprocessor needs to use the same I/O standard as V_{CCIO} to drive the JTAG pins.

Jam STAPL

Jam 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. It is a freely licensed open standard.

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.

For more information about JTAG and Jam STAPL in embedded environments, refer to the *AN 425: Using Command-Line Jam STAPL Solution for Device Programming*. To download the jam player, visit the Altera website at www.altera.com.

Device Configuration Pins

The following tables describe the connections and functionality of all the configuration-related pins on the Arria II GX devices. Table 9–9 summarizes the Arria II GX configuration pins and their power supply.

Table 9–9. Arria II GX Configuration Pin Summary

Description	Input/Output	Dedicated	Powered By	Configuration Mode
TDI	Input	Yes	V_{CCIO}	JTAG
TMS	Input	Yes	V_{CCIO}	JTAG
TCK	Input	Yes	V_{CCIO}	JTAG
TDO	Output	Yes	V_{CCIO} and V_{CCPD} *(2)*	JTAG
CRC_ERROR	Output	—	Pull-up	Optional, all modes
DATA0	Input	—	V_{CCIO}	All modes except JTAG
DATA[7..1]	Input	—	V_{CCIO}	FPP
INIT_DONE	Output	—	Pull-up	Optional, all modes
CLKUSR	Input	—	V_{CCIO}	Optional
nSTATUS	Bidirectional	Yes	Pull-up	All modes
nCE	Input	Yes	V_{CCIO}	All modes
CONF_DONE	Bidirectional	Yes	Pull-up	All modes
nCONFIG	Input	Yes	V_{CCIO}	All modes
ASDO	Output	Yes	V_{CCIO}	AS
nCSO	Output	Yes	V_{CCIO}	AS
DCLK	Input	Yes	V_{CCIO}	PS, FPP
	Output	Yes	V_{CCIO}	AS
nIO_PULLUP	Input	Yes	V_{CC} *(1)*	All modes
nCEO	Output	Yes	Pull-up	All modes
MSEL[3..0]	Input	Yes	V_{CCPD}	All modes

Notes to Table 9–9:

(1) Although the nIO_PULLUP is powered up by V_{CC}, Altera recommends you connect these pins to V_{CCIO} or GND directly without using a pull-up or pull-down resistor.

(2) The TDO pin is powered up by the V_{CCIO} and V_{CCPD} power supply of I/O bank 8C.

Table 9–10 describes the dedicated configuration pins. You must connect these pins properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

Table 9–10. Dedicated Configuration Pins on the Arria II GX Device (Part 1 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
`VCCPD`	N/A	All	Power	Dedicated power pin. Use this pin to power the I/O pre-drivers, the JTAG output pin (`TDO`), and the `MSEL[3..0]`. You must connect V_{CCPD} according to the I/O standard used in the same bank: ■ For 3.3-V I/O standards, connect V_{CCPD} to 3.3 V ■ For 3.0-V I/O standards, connect V_{CCPD} to 3.0 V ■ For 2.5-V and below I/O standards, connect V_{CCPD} to 2.5 V V_{CCPD} must ramp-up from 0 V to 2.5 V, 3.0 V, or 3.3 V in 100 ms for standard POR or 4 ms for fast POR. If V_{CCPD} is not ramped up in this specified time, your Arria II GX device will not configure successfully. If your system does not allow for a V_{CCPD} to ramp-up time in 100 ms or 4 ms, you must hold `nCONFIG` low until all power supplies are stable.
`nIO_PULLUP`	N/A	All	Input	Dedicated input that chooses whether the internal pull-up resistors on the user I/O pins and dual-purpose I/O pins (`nCSO`, `nASDO`, `DATA[7..0]`, `CLKUSR`, `INIT_DONE`) are on or off before and during configuration. A logic high turns off the weak internal pull-up resistors, while a logic low turns them on. The `nIO-PULLUP` input buffer is powered by V_{CC} and has an internal 5-kΩ pull-down resistor that is always active. You can tie the `nIO-PULLUP` directly to the V_{CCIO} supply of the bank in which the pin resides or GND.
`MSEL[3..0]`	N/A	All	Input	Four-bit configuration input that sets the Arria II GX device configuration scheme. Refer to Table 9–2 on page 9–6 for the appropriate connections. You must hardwire these pins to the V_{CCPD} or GND. The `MSEL[3..0]` pins have internal 5-kΩ pull-down resistors that are always active.
`nCONFIG`	N/A	All	Input	Configuration control input. Pulling this pin low during user-mode causes the device to lose its configuration data, enter a reset state, and tri-state all I/O pins. Returning this pin to a logic-high level starts a reconfiguration. Configuration is possible only if this pin is high, except in JTAG programming mode, when `nCONFIG` is ignored.

Table 9–10. Dedicated Configuration Pins on the Arria II GX Device (Part 2 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS	N/A	All	Bi-directional open-drain	The device drives nSTATUS low immediately after power-up and releases it after the POR time. During user mode and regular configuration, this pin is pulled high by an external 10-kΩ resistor. This pin, when driven low by the Arria II GX device, indicates that the device has encountered an error during configuration. 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 you use a configuration device, driving nSTATUS low causes the configuration device to attempt to configure the device, but because the device ignores transitions on nSTATUS in user mode, the device does not reconfigure. To begin a reconfiguration, nCONFIG must be pulled low. If the V_{CCIO} supply of the bank in which the nSTATUS pin resides is not fully powered up, the following could occur: ■ V_{CCIO} is powered high enough for the nSTATUS buffer to function properly and nSTATUS is driven low. When V_{CCIO} is ramped up, POR trips and nSTATUS is released after POR expires. ■ V_{CCIO} is not powered high enough for the nSTATUS buffer to function properly. In this situation, nSTATUS might appear logic high, triggering a configuration attempt that would fail because POR did not yet trip. When V_{CCPD} is powered up, nSTATUS is pulled low because POR did not yet trip. When POR trips after V_{CCIO} is powered up, nSTATUS is released and pulled high. At that point, reconfiguration is triggered and the device is configured.
CONF_DONE	N/A	All	Bi-directional open-drain	Status output. The target device 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 for the device to initialize. Driving CONF_DONE low after configuration and initialization does not affect the configured device.

Table 9–10. Dedicated Configuration Pins on the Arria II GX Device (Part 3 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
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 must 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 device.
nCEO	I/O	All	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 nCEO pin is powered by the V_{CCIO} supply of the bank in which the nCEO pin resides. After configuration, nCEO is available as user I/O pins. The state of the nCEO pin depends on the **Dual-Purpose Pin** settings.
ASDO	N/A	AS	Output	Control signal from the Arria II GX device to the serial configuration device in AS mode used to read out configuration data. In AS mode, ASDO has an internal pull-up resistor that is always active.
nCSO	N/A	AS	Output	Output control signal from the Arria II GX device to the serial configuration device in AS mode that enables the configuration device. In AS mode, nCSO has an internal pull-up resistor that is always active.
DCLK	N/A	Synchronous configuration schemes (PS, FPP, AS)	Input (PS, FPP) Output (AS)	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 device on the rising edge of DCLK. In AS mode, DCLK is an output from the Arria II GX device that provides timing for the configuration interface. In AS mode, DCLK has an internal pull-up resistor (typically 25 kΩ) that is always active. 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 must be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device.

Table 9–10. Dedicated Configuration Pins on the Arria II GX Device (Part 4 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
`DATA0`	N/A	PS, FPP, AS	Input	Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the `DATA0` pin. In AS mode, `DATA0` has an internal pull-up resistor that is always active. The `DATA[0]` is a dedicated pin that is used for both passive and active configuration modes and it is not available as a user I/O pin after configuration.
`DATA[7..1]`	I/O	Parallel configuration schemes (FPP)	Inputs	Data inputs. Byte-wide configuration data is presented to the target device on `DATA[7..0]`. In serial configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated. After FPP configuration, `DATA[7..1]` are available as user I/O pins and the state of these pin depends on the **Dual-Purpose Pin** settings.

Table 9–11 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-up resistors.

Table 9–11. Optional Configuration Pins (Part 1 of 2)

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. Enable this pin 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	Use as `Status` pin 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 resistor. 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 device 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.

Table 9–11. Optional Configuration Pins (Part 2 of 2)

Pin Name	User Mode	Pin Type	Description
`DEV_OE`	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated. When this pin is driven high, all I/O pins behave as programmed. Enable this pin 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 9–12 describes the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. `TDI` and `TMS` have weak internal pull-up resistors while `TCK` has a weak internal pull-down resistor (typically 25 kΩ). If you plan to use the SignalTap® embedded logic array analyzer, you must connect the JTAG pins of the Arria II GX device to a JTAG header on your board.

Table 9–12. 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 on the rising edge of `TCK`. The `TDI` pin is powered by the V_{CCIO} supply of I/O bank 8C. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by connecting this pin to logic high.
`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. The `TDO` pin is powered up by the V_{CCIO} and V_{CCPD} power supply of I/O bank 8C. For recommendations about connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary Scan Testing* chapter in volume 1 of the *Arria II GX Device Handbook*. If the JTAG interface is not required on the board, you can disable the JTAG circuitry 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. `TMS` is evaluated on the rising edge of `TCK`. Therefore, you must set up `TMS` before the rising edge of `TCK`. Transitions in the state machine occur on the falling edge of `TCK` after the signal is applied to `TMS`. The `TMS` pin is powered by the V_{CCIO} supply of I/O bank 8C. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by connecting this pin to logic high.
`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. The `TCK` pin is powered by the V_{CCIO} supply of I/O bank 8C. It is expected that the clock input waveform have a nominal 50% duty cycle. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by connecting `TCK` to GND.

Configuration Data Decompression

Arria II GX devices support configuration data decompression, which saves configuration memory space and time. This feature allows you to store compressed configuration data in configuration or other memory devices and transmit this compressed bitstream to Arria II GX devices. During configuration, the Arria II GX device decompresses the bitstream in real time and programs its SRAM cells.

Preliminary data indicates that compression typically reduces the configuration bitstream size by 35 to 55% based on the designs used.

Arria II GX devices support decompression in the FPP (when using a MAX II device or microprocessor + flash), AS, and PS configuration schemes. The Arria II GX decompression feature is not available in the JTAG configuration scheme.

When using FPP mode, the intelligent host must provide a `DCLK` that is ×4 the data rate. Therefore, the configuration data must be valid for four `DCLK` cycles.

In PS mode, use the Arria II GX decompression feature, because sending compressed configuration data reduces configuration time.

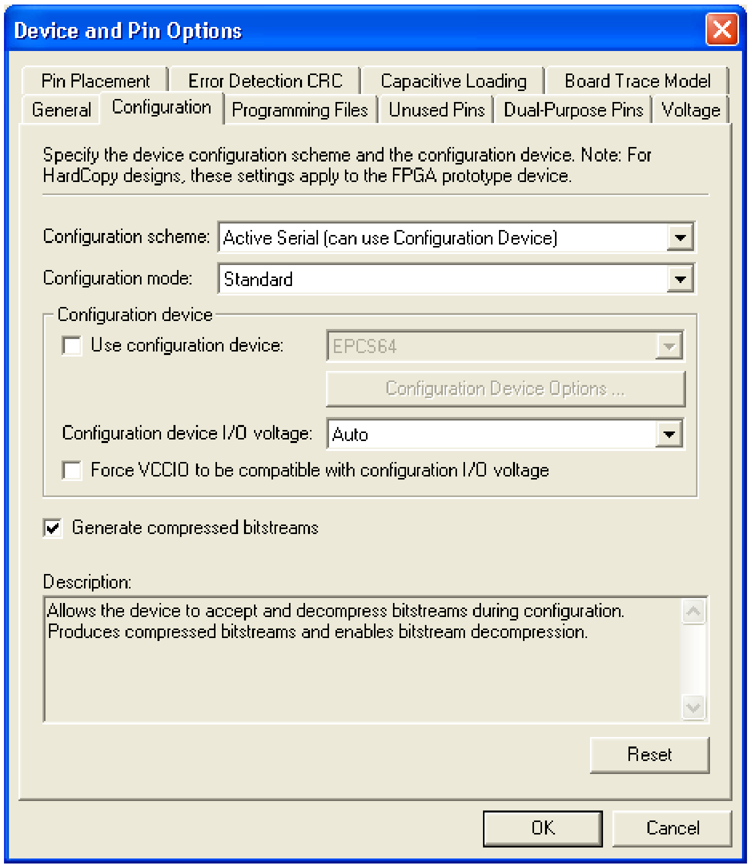
When you enable compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash memory, and decreases the time needed to transmit the bitstream to the Arria II GX device. The time required by a Arria II GX device to decompress a configuration file is less than the time needed to transmit the configuration data to the device.

There are two ways to enable compression for Arria II GX bitstreams: before design compilation (in the Compiler Settings menu) and after design compilation (in the **Convert Programming Files** window).

To enable compression in the project's Compiler Settings menu, perform the following steps:

1. On the Assignments menu, click **Device** to bring up the **Settings** dialog box.
2. After selecting your Arria II GX device, open the **Device and Pin Options** window.
3. In the **Configuration settings** tab, enable the check box for **Generate compressed bitstreams** (as shown in Figure 9–19).

Figure 9–19. Enabling Compression for Arria II GX Bitstreams in Compiler Settings



Specify the device configuration scheme and the configuration device. Note: For HardCopy designs, these settings apply to the FPGA prototype device.

Configuration scheme: Active Serial (can use Configuration Device)

Configuration mode: Standard

Configuration device

☐ Use configuration device: EPCS64

Configuration Device Options ...

Configuration device I/O voltage: Auto

☐ Force VCCIO to be compatible with configuration I/O voltage

☑ Generate compressed bitstreams

Description:

Allows the device to accept and decompress bitstreams during configuration. Produces compressed bitstreams and enables bitstream decompression.

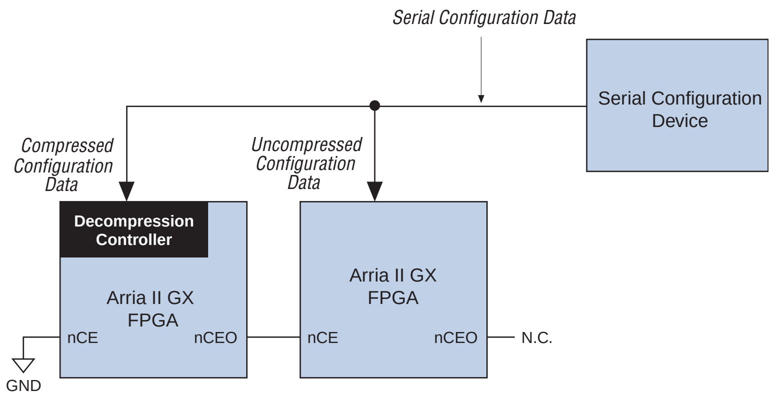
You can also enable compression when creating programming files from the **Convert Programming Files** window. To do this, perform the following steps:

1. On the File menu, click **Convert Programming Files**.
2. Select the programming file type (**.pof**, **.sram**, **.hex**, **.rbf**, or **.ttf**).
3. For POF output files, select a configuration device.
4. In the **Input files to convert** box, select **SOF Data**.
5. Select **Add File** and add a Arria II GX device **.sof** files.
6. Select the name of the file you added to the **SOF Data** area and click **Properties**.
7. Check the **Compression** check box.

When multiple Arria II GX devices are cascaded, you can selectively enable the compression feature for each device in the chain if you are using a serial configuration scheme. Figure 9–20 depicts a chain of two Arria II GX devices. The first Arria II GX device has compression enabled and therefore receives a compressed bitstream from the configuration device. The second Arria II GX device has the compression feature disabled and receives uncompressed data.

In a multi-device FPP configuration chain (with a MAX II device or microprocessor + flash), all Arria II GX devices in the chain must either enable or disable the decompression feature. You cannot selectively enable the compression feature for each device in the chain because of the DATA and DCLK relationship.

Figure 9–20. Compressed and Uncompressed Serial Configuration Data in the Same Configuration File


Serial Configuration Data
Serial Configuration Device
Compressed Configuration Data
Uncompressed Configuration Data
Decompression Controller
Arria II GX FPGA
nCE
nCEO
Arria II GX FPGA
nCE
nCEO
N.C.
GND

You can generate programming files for this setup by clicking **Convert Programming Files** on the File menu in the Quartus II software.

Remote System Upgrades

This section describes the functionality and implementation of the dedicated remote system upgrade circuitry. It also defines several concepts related to remote system upgrades, including factory configuration, application configuration, remote update mode, and user watchdog timer. Additionally, this section provides design guidelines for implementing remote system upgrades with the supported configuration schemes.

System designers sometimes face challenges such as shortened design cycles, evolving standards, and system deployments in remote locations. Arria II GX devices help overcome these challenges with their inherent reprogrammability and dedicated circuitry to perform remote system upgrades. Remote system upgrades help deliver feature enhancements and bug fixes without costly recalls, reduce time-to-market, extend product life, and help to avoid system downtime.

Arria II GX devices feature dedicated remote system upgrade circuitry. Soft logic (either the Nios® II embedded processor or user logic) implemented in an Arria II GX device can download a new configuration image from a remote location, store it in configuration memory, and direct the dedicated remote system upgrade circuitry to start a reconfiguration cycle. The dedicated circuitry performs error detection during and after the configuration process, recovers from any error condition by reverting back to a safe configuration image, and provides error status information.

Remote system upgrades are supported in fast active serial (AS) Arria II GX configuration schemes. You can also implement remote system upgrades in conjunction with advanced Arria II GX features such as real-time decompression of configuration data and design security using the advanced encryption standard (AES) for secure and efficient field upgrades. The largest serial configuration device currently supports 128 MBits of configuration bitstream.

Functional Description

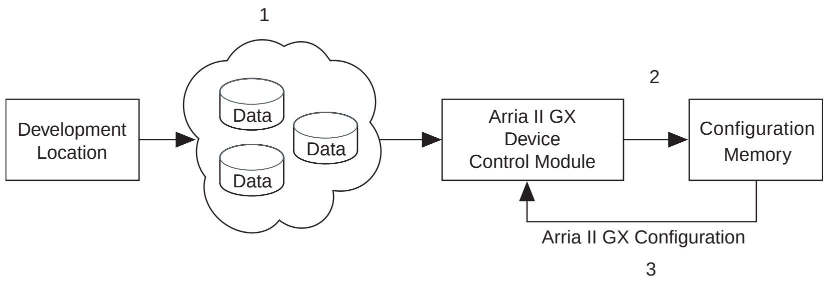
The dedicated remote system upgrade circuitry in Arria II GX devices manages remote configuration and provides error detection, recovery, and status information. User logic or a Nios II processor implemented in the Arria II GX device logic array provides access to the remote configuration data source and an interface to the system's configuration memory.

Arria II GX devices have remote system upgrade processes that involve the following steps:

1. A Nios II processor (or user logic) implemented in the Arria II GX device logic array receives new configuration data from a remote location. The connection to the remote source uses a communication protocol such as transmission control protocol/Internet protocol (TCP/IP), peripheral component interconnect (PCI), user datagram protocol (UDP), universal asynchronous receiver/transmitter (UART), or a proprietary interface.
2. The Nios II processor (or user logic) stores this new configuration data in non-volatile configuration memory.
3. The Nios II processor (or user logic) starts a reconfiguration cycle with the new or updated configuration data.
4. The dedicated remote system upgrade circuitry detects and recovers from any errors that might occur during or after the reconfiguration cycle and provides error status information to the user design.

Figure 9–21 shows the steps required for performing remote configuration updates. (The numbers in Figure 9–21 coincide with the steps just mentioned.)

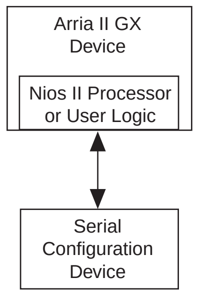
Figure 9–21. Functional Diagram of Arria II GX Remote System Upgrade



☞ Arria II GX devices only support remote system upgrade in the single device AS configuration scheme.

Figure 9–22 shows a block diagram for implementing a remote system upgrade with the Arria II GX AS configuration scheme.

Figure 9–22. Remote System Upgrade Block Diagram for Arria II GX AS Configuration Scheme



You must set the mode select pins (`MSEL[3..0]`) to **AS mode** to use the remote system upgrade in your system.

☞ The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect `MSEL[3..0]`, refer to Table 9–2 on page 9–6.

☞ When using fast AS mode, you must select **remote update mode** in the Quartus II software and insert the ALTREMOTE_UPDATE megafunction to access the circuitry. For more information, refer to "ALTREMOTE_UPDATE Megafunction" on page 9–54.

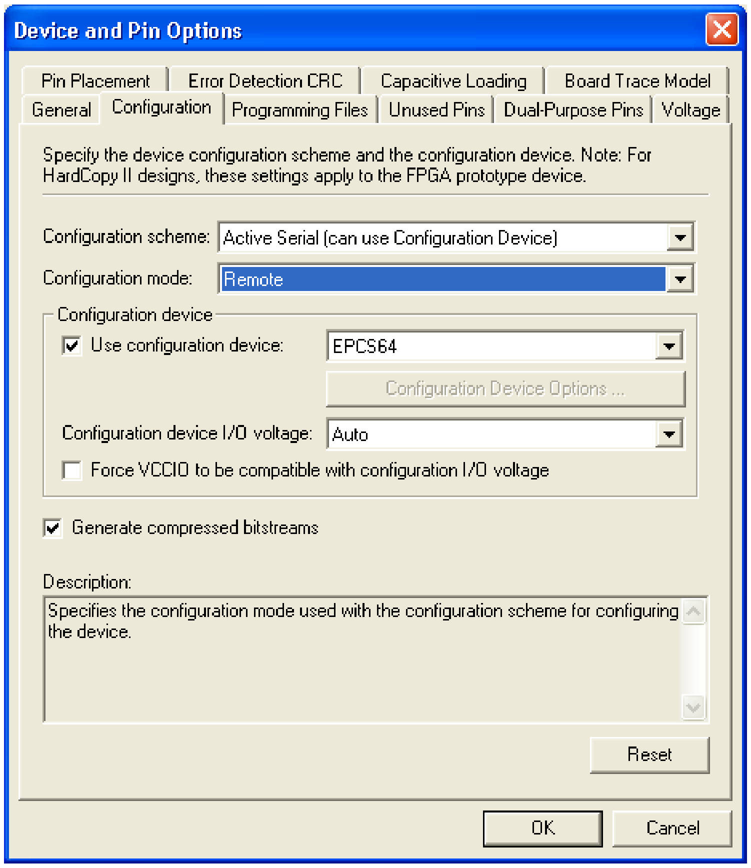
Enabling Remote Update

You can enable remote update for Arria II GX devices in the Quartus II software before design compilation (in the Compiler Settings menu). In remote update mode, the **auto-restart configuration after error** option is always enabled. To enable remote update in the project's compiler settings, perform the following steps in the Quartus II software:

1. On the Assignment menu, click **Device**. The **Settings** dialog box appears.

2. Click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
3. Click the **Configuration** tab.
4. From the Configuration scheme list, select **Active Serial** (you can also use **Configuration Device**) (Figure 9–23).
5. From the Configuration Mode list, select **Remote** (Figure 9–23).
6. Click **OK**.
7. In the **Settings** dialog box, click **OK**.

Figure 9–23. Enabling Remote Update for Arria II GX Devices in the Compiler Settings Menu



Specify the device configuration scheme and the configuration device. Note: For HardCopy II designs, these settings apply to the FPGA prototype device.

Configuration scheme: Active Serial (can use Configuration Device)

Configuration mode: Remote

Configuration device

[✓] Use configuration device: EPCS64

Configuration device I/O voltage: Auto

[ ] Force VCCIO to be compatible with configuration I/O voltage

[✓] Generate compressed bitstreams

Description:

Specifies the configuration mode used with the configuration scheme for configuring the device.

Configuration Image Types

When performing a remote system upgrade, Arria II GX device configuration bitstreams are classified as factory configuration images or application configuration images. An image, also referred to as a configuration, is a design loaded into the Arria II GX device that performs certain user-defined functions.

Each Arria II GX device in your system requires one factory image or the addition of one or more application images. The factory image is a user-defined fall-back, or safe configuration, and is responsible for administering remote updates in conjunction with the dedicated circuitry. Application images implement user-defined functionality in the target Arria II GX device. You may include the default application image functionality in the factory image.

A remote system upgrade involves storing a new application configuration image or updating an existing one using the remote communication interface. After an application configuration image is stored or updated remotely, the user design in the Arria II GX device starts a reconfiguration cycle with the new image. Any errors during or after this cycle are detected by the dedicated remote system upgrade circuitry and cause the device to automatically revert to the factory image. The factory image then performs error processing and recovery. The factory configuration is written to the serial configuration device only once by the system manufacturer and should not be remotely updated. On the other hand, application configurations may be remotely updated in the system. Both images can begin system reconfiguration.

Remote System Upgrade Mode

Remote system upgrade has only one mode of operation: remote update mode. Remote update mode allows you to determine the functionality of your system upon power-up and offers different features.

Overview

In remote update mode, Arria II GX devices load the factory configuration image upon power up. The user-defined factory configuration determines which application configuration is to be loaded and triggers a reconfiguration cycle. The factory configuration may also contain application logic.

When used with serial configuration devices, remote update mode allows an application configuration to start at any flash sector boundary. For example, this translates to a maximum of 256 pages in the EPCS128 device and 64 pages in the EPCS32 device, where the minimum size of each page is 512 KBits. Altera does not recommend using the same page in the serial configuration device for two images. Additionally, remote update mode features a user watchdog timer that determines the validity of an application configuration.

Remote Update Mode

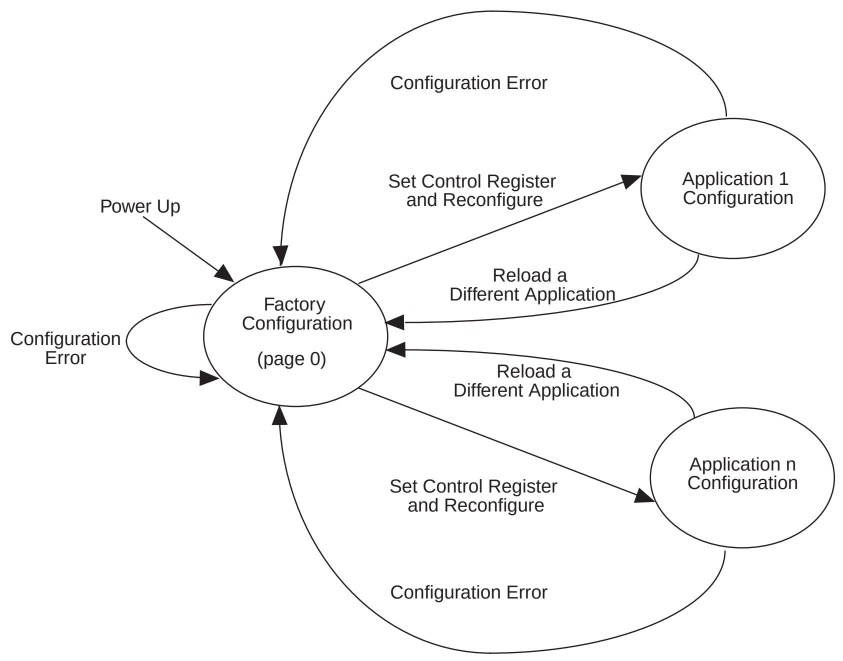
When an Arria II GX device is first powered up in remote update mode, it loads the factory configuration located at page zero (page registers `PGM[23:0] = 24'b0`). Always store the factory configuration image for your system at page address zero. This corresponds to the start address location 0×000000 in the serial configuration device.

The factory image is user-designed and contains soft logic to:

- Process any errors based on status information from the dedicated remote system upgrade circuitry
- Communicate with the remote host and receive new application configurations and store this new configuration data in the local non-volatile memory device
- Determine which application configuration is to be loaded into the Arria II GX device
- Enable or disable the user watchdog timer and load its time-out value (optional)
- Instruct the dedicated remote system upgrade circuitry to start a reconfiguration cycle

Figure 9–24 shows the transitions between the factory and application configurations in remote update mode.

Figure 9–24. Transitions between Configurations in Remote Update Mode


Configuration Error
Set Control Register and Reconfigure
Application 1 Configuration
Power Up
Reload a Different Application
Factory Configuration
(page 0)
Configuration Error
Reload a Different Application
Application n Configuration
Set Control Register and Reconfigure
Configuration Error

After power up or a configuration error, the factory configuration logic is loaded automatically. The factory configuration also needs to specify whether to enable the user watchdog timer for the application configuration and if enabled, to include the timer setting information as well.

The user watchdog timer ensures that the application configuration is valid and functional. The timer must be continually reset in a specific amount of time during user mode operation of an application configuration. Only valid application configurations contain the logic to reset the timer in user mode. This timer reset logic needs to be part of a user-designed hardware and/or software health monitoring signal that indicates error-free system operation. If the timer is not reset in a specific amount of time; for example, the user application configuration detects a functional problem or if the system hangs, the dedicated circuitry updates the remote system upgrade status register, triggering the loading of the factory configuration.

The user watchdog timer is automatically disabled for factory configurations. For more information about the user watchdog timer, refer to "User Watchdog Timer" on page 9–52.

If there is an error while loading the application configuration, the cause of the reconfiguration is written by the dedicated circuitry to the remote system upgrade status register. Actions that cause the remote system upgrade status register to be written are:

- nSTATUS driven low externally
- Internal CRC error

- User watchdog timer time-out
- A configuration reset (logic array `nCONFIG` signal or external `nCONFIG` pin assertion to low)

Arria II GX devices automatically load the factory configuration located at page address zero. This user-designed factory configuration can read the remote system upgrade status register to determine the reason for the reconfiguration. The factory configuration then takes appropriate error recovery steps and writes to the remote system upgrade control register to determine the next application configuration to be loaded.

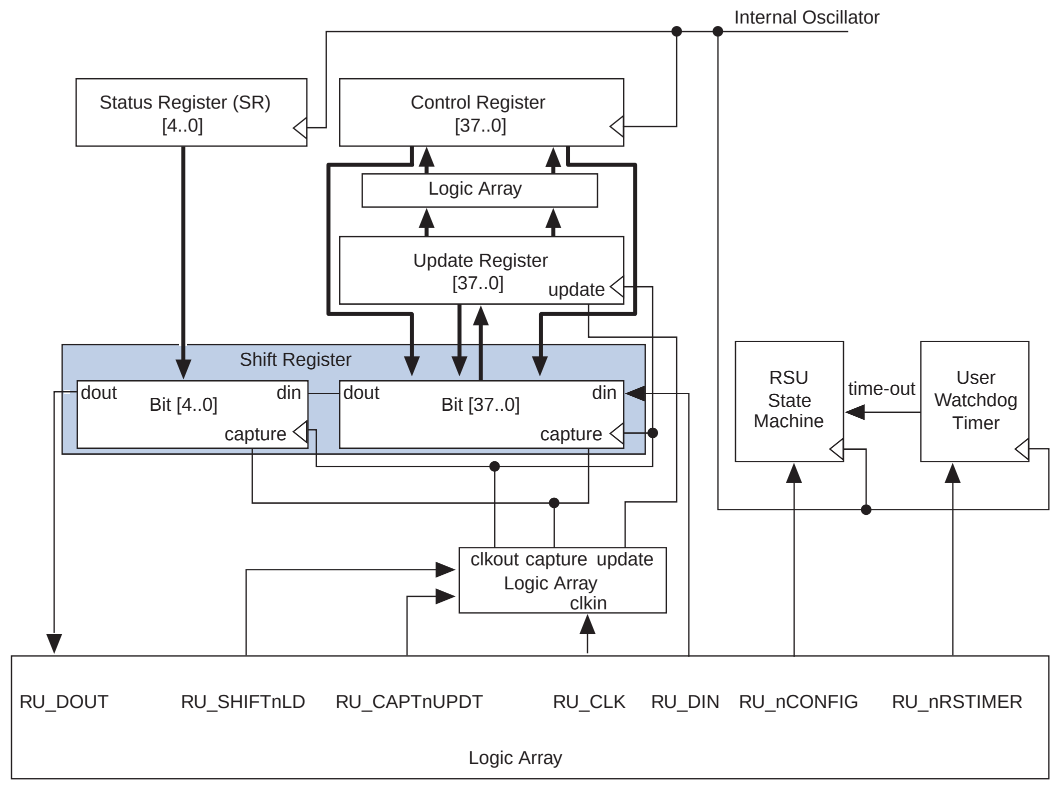
When Arria II GX devices successfully load the application configuration, they enter into user mode. In user mode, the soft logic (Nios II processor or state machine and the remote communication interface) assists the Arria II GX device in determining when a remote system update is arriving. When a remote system update arrives, the soft logic receives the incoming data, writes it to the configuration memory device, and triggers the device to load the factory configuration. The factory configuration reads the remote system upgrade status register and control register, determines the valid application configuration to load, writes the remote system upgrade control register accordingly, and initiates system reconfiguration.

Dedicated Remote System Upgrade Circuitry

This section describes the implementation of the Arria II GX dedicated remote system upgrade circuitry. The remote system upgrade circuitry is implemented in hard logic. This dedicated circuitry interfaces with the user-defined factory and application configurations implemented in the Arria II GX device logic array to provide the complete remote configuration solution. The remote system upgrade circuitry contains the remote system upgrade registers, a watchdog timer, and a state machine that controls those components.

Figure 9–25 shows the remote system upgrade block's data path.

Figure 9–25. Remote System Upgrade Circuit Data Path *(Note 1)*



Note to Figure 9–25:

(1) The `RU_DOUT`, `RU_SHIFTnLD`, `RU_CAPTnUPDT`, `RU_CLK`, `RU_DIN`, `RU_nCONFIG`, and `RU_nRSTIMER` signals are internally controlled by the ALTREMOTE_UPDATE megafunction.

Remote System Upgrade Registers

The remote system upgrade block contains a series of registers that store the page addresses, watchdog timer settings, and status information. These registers are detailed in Table 9–13.

Table 9–13. Remote System Upgrade Registers (Part 1 of 2)

Register	Description
Shift register	This register is accessible by the logic array and allows the update, status, and control registers to be written and sampled by user logic.
Control register	This register contains the current page address, user watchdog timer settings, and one bit specifying whether the current configuration is a factory configuration or an application configuration. During a read operation in an application configuration, this register is read into the shift register. When a reconfiguration cycle is initiated, the contents of the update register are written into the control register.

Table 9–13. Remote System Upgrade Registers (Part 2 of 2)

Register	Description
Update register	This register contains data similar to that in the control register. However, it can only be updated by the factory configuration by shifting data into the shift register and issuing an update operation. When a reconfiguration cycle is triggered by the factory configuration, the control register is updated with the contents of the update register. During a capture in a factory configuration, this register is read into the shift register.
Status register	This register is written to by the remote system upgrade circuitry on every reconfiguration to record the cause of the reconfiguration. This information is used by the factory configuration to determine the appropriate action following a reconfiguration. During a capture cycle, this register is read into the shift register.

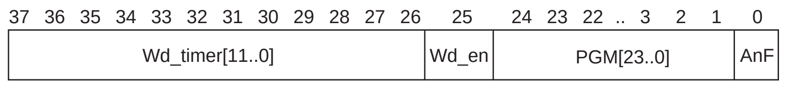
The remote system upgrade control and status registers are clocked by the 10-MHz internal oscillator (the same oscillator that controls the user watchdog timer). However, the remote system upgrade shift and update registers are clocked by the user clock input (`RU_CLK`).

Remote System Upgrade Control Register

The remote system upgrade control register stores the application configuration page address and user watchdog timer settings. The control register functionality depends on the remote system upgrade mode selection. In remote update mode, the control register page address bits are set to all zeros (`24'b0 = 0×000000`) at power up to load the factory configuration. A factory configuration in remote update mode has write access to this register.

The control register bit positions are shown in Figure 9–26 and defined in Table 9–14. In the figure, the numbers show the bit position of a setting in a register. For example, bit number 25 is the enable bit for the watchdog timer.

Figure 9–26. Remote System Upgrade Control Register



The application-not-factory (`AnF`) bit indicates whether the current configuration loaded in the Arria II GX device is the factory configuration or an application configuration. This bit is set low by the remote system upgrade circuitry when an error condition causes a fall-back to the factory configuration. When the `AnF` bit is high, the control register access is limited to read operations. When the `AnF` bit is low, the register allows write operations and disables the watchdog timer.

In remote update mode, the factory configuration design sets this bit high (1'b1) when updating the contents of the update register with the application page address and watchdog timer settings.

Table 9–14 shows the remote system upgrade control register contents.

Table 9–14. Remote System Upgrade Control Register Contents

Control Register Bit	Remote System Upgrade Mode	Value *(2)*	Definition
`AnF` *(1)*	Remote update	1'b0	Application not factory
`PGM[23..0]`	Remote update	24'b0×000000	AS configuration start address (`StAdd[23..0]`)
`Wd_en`	Remote update	1'b0	User watchdog timer enable bit
`Wd_timer[11..0]`	Remote update	12'b000000000000	User watchdog time-out value (most significant 12 bits of 29-bit count value: `{Wd_timer[11..0], 17'b0}`)

Notes to Table 9–14:

(1) In remote update mode, the remote configuration block does not update the `AnF` bit automatically (you can update it manually).

(2) This is the default value of the control register bit.

Remote System Upgrade Status Register

The remote system upgrade status register specifies the reconfiguration trigger condition. The various trigger and error conditions include:

- Cyclical redundancy check (CRC) error during application configuration
- `nSTATUS` assertion by an external device due to an error
- Arria II GX device logic array triggered a reconfiguration cycle, possibly after downloading a new application configuration image
- External configuration reset (`nCONFIG`) assertion
- User watchdog timer time-out

Figure 9–27 specifies the contents of the status register. The numbers in the figure show the bit positions in a 5-bit register.

Figure 9–27. Remote System Upgrade Status Register

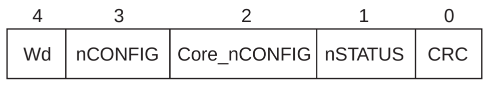


Table 9–15 lists the status register contents for remote system upgrade.

Table 9–15. Remote System Upgrade Status Register Contents (Part 1 of 2)

Status Register Bit	Definition	POR Reset Value
`CRC` (from configuration)	`CRC` error caused reconfiguration	1 bit '0'
`nSTATUS`	`nSTATUS` caused reconfiguration	1 bit '0'
`CORE_nCONFIG` *(1)*	Device logic array caused reconfiguration	1 bit '0'
`nCONFIG`	`nCONFIG` caused reconfiguration	1 bit '0'

Table 9–15. Remote System Upgrade Status Register Contents (Part 2 of 2)

Status Register Bit	Definition	POR Reset Value
`Wd`	Watchdog timer caused reconfiguration	1 bit '0'

Note to Table 9–15:

(1) Logic array reconfiguration forces the system to load the application configuration data into the Arria II GX device. This occurs after the factory configuration specifies the appropriate application configuration page address by updating the update register.

Remote System Upgrade State Machine

The remote system upgrade control and update registers have identical bit definitions, but serve different roles (refer to Figure 9–26 on page 9–50). While both registers can only be updated when the device is loaded with a factory configuration image, the update register writes are controlled by the user logic; the control register writes are controlled by the remote system upgrade state machine.

In factory configurations, the user logic sends the `AnF` bit (set high), page address, and watchdog timer settings for the next application configuration bit to the update register. When the logic array configuration reset (`RU_nCONFIG`) goes low, the remote system upgrade state machine updates the control register with the contents of the update register and starts system reconfiguration from the new application page.

If there is an error or reconfiguration trigger condition, the remote system upgrade state machine directs the system to load a factory or application configuration (page zero or page one, based on the mode and error condition) by setting the control register accordingly. Table 9–16 lists the contents of the control register after such an event occurs for all possible error or trigger conditions.

The remote system upgrade status register is updated by the dedicated error monitoring circuitry after an error condition but before the factory configuration is loaded.

Table 9–16. Control Register Contents after an Error or Reconfiguration Trigger Condition

Reconfiguration Error/Trigger	Control Register Setting Remote Update
`nCONFIG` reset	All bits are 0
`nSTATUS` error	All bits are 0
`CORE` triggered reconfiguration	Update register
`CRC` error	All bits are 0
`Wd` time out	All bits are 0

Capture operations during factory configuration access the contents of the update register. This feature is used by the user logic to verify that the page address and watchdog timer settings were written correctly. Read operations in application configurations access the contents of the control register. This information is used by the user logic in the application configuration.

User Watchdog Timer

The user watchdog timer prevents a faulty application configuration from stalling the device indefinitely. The system uses the timer to detect functional errors after an application configuration is successfully loaded into the Arria II GX device.

The user watchdog timer is a counter that counts down from the initial value loaded into the remote system upgrade control register by the factory configuration. The counter is 29-bits wide and has a maximum count value of 2^{29}. When specifying the user watchdog timer value, specify only the most significant 12 bits. The granularity of the timer setting is 2^{15} cycles. The cycle time is based on the frequency of the 10-MHz internal oscillator. Table 9–17 specifies the operating range of the 10-MHz internal oscillator.

Table 9–17. 10-MHz Internal Oscillator Specifications *(Note 1)*

Minimum	Typical	Maximum	Units
4.3	5.3	10	MHz

Note to Table 9–17:

(1) These values are preliminary.

The user watchdog timer begins counting once the application configuration enters device user mode. This timer must be periodically reloaded or reset by the application configuration before the timer expires by asserting `RU_nRSTIMER`. If the application configuration does not reload the user watchdog timer before the count expires, a time-out signal is generated by the remote system upgrade dedicated circuitry. The time-out signal tells the remote system upgrade circuitry to set the user watchdog timer status bit (`Wd`) in the remote system upgrade status register and reconfigures the device by loading the factory configuration.

The user watchdog timer is not enabled during the configuration cycle of the device. Errors during configuration are detected by the CRC engine. Also, the timer is disabled for factory configurations. Functional errors should not exist in the factory configuration because it is stored and validated during production and is never updated remotely.

The user watchdog timer is disabled in factory configurations and during the configuration cycle of the application configuration. It is enabled after the application configuration enters user mode.

Quartus II Software Support

The Quartus II software provides the flexibility to include the remote system upgrade interface between the Arria II GX device logic array and the dedicated circuitry, generates configuration files for production, and allows remote programming of the system configuration memory.

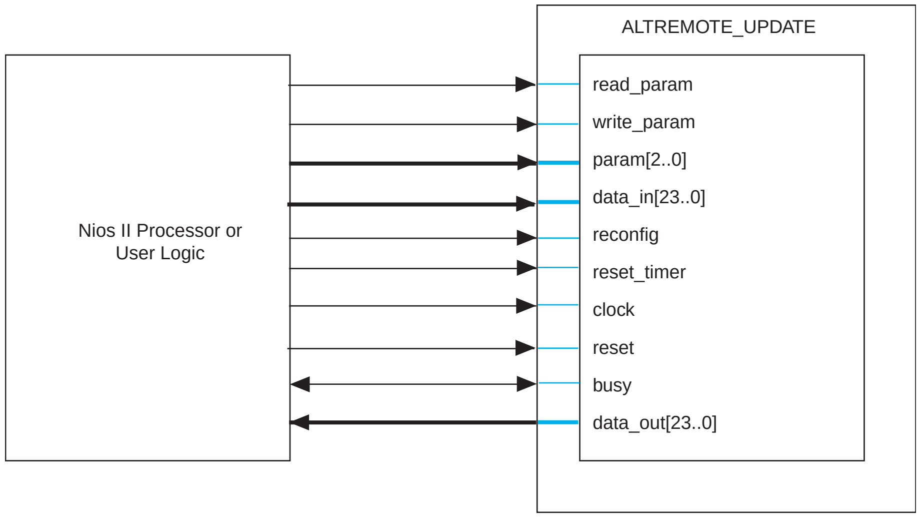
The ALTREMOTE_UPDATE megafunction is the implementation option in the Quartus II software that is used for the interface between the remote system upgrade circuitry and the device logic array interface. Using the megafunction block instead of creating your own logic saves design time and offers more efficient logic synthesis and device implementation.

ALTREMOTE_UPDATE Megafunction

The ALTREMOTE_UPDATE megafunction provides a memory-like interface to the remote system upgrade circuitry and handles the shift register read and write protocol in the Arria II GX device logic. This implementation is suitable for designs that implement the factory configuration functions using a Nios II processor or user logic in the device.

Figure 9–28 shows the interface signals between the ALTREMOTE_UPDATE megafunction and Nios II processor or user logic.

Figure 9–28. Interface Signals between the ALTREMOTE_UPDATE Megafunction and the Nios II Processor



For more information about the ALTREMOTE_UPDATE megafunction and the description of ports listed in Figure 9–28, refer to the *ALTREMOTE_UPDATE Megafunction User Guide*.

Design Security

This section provides an overview of the design security features and their implementation on Arria II GX devices using advanced encryption standard (AES). It also describes the security modes available in Arria II GX devices that allow you to use these new features in your designs.

As Arria II GX devices start to play roles in larger and more critical designs in competitive commercial and military environments, it is increasingly important to protect your designs from copying, reverse engineering, and tampering.

Arria II GX devices address these concerns with both volatile and non-volatile security feature support. Arria II GX devices have the ability to decrypt configuration bitstreams using the AES algorithm, an industry-standard encryption algorithm that is FIPS-197 certified. Arria II GX devices have a design security feature which uses a 256-bit security key.

Arria II GX devices store configuration data in SRAM configuration cells during device operation. Because SRAM memory is volatile, the SRAM cells must be loaded with configuration data each time the device powers up. It is possible to intercept configuration data when it is being transmitted from the memory source (flash memory or a configuration device) to the device. The intercepted configuration data could then be used to configure another device.

When using the Arria II GX design security feature, the security key is stored in the Arria II GX device. Depending on the security mode, you can configure the Arria II GX device using a configuration file that is encrypted with the same key, or for board testing, configured with a normal configuration file.

The design security feature is available when configuring Arria II GX devices using the fast passive parallel configuration mode with an external host (such as a MAX II device or microprocessor), or when using active serial (AS) or passive serial (PS) configuration schemes. The design security feature is also available in remote update mode with AS configuration. The design security feature is not available when you are configuring your Arria II GX device using Joint Test Action Group (JTAG)-based configuration. For more details, refer to "Supported Configuration Schemes" on page 9–59.

☞ When using a serial configuration scheme such as PS or fast AS, configuration time is the same whether or not the design security feature is enabled. If the FPP scheme is used with the design security or decompression feature, a ×4 `DCLK` is required. This results in a slower configuration time when compared to the configuration time of an Arria II GX device that has neither the design security nor the decompression feature enabled.

Arria II GX Security Protection

Arria II GX device designs are protected from copying, reverse engineering, and tampering using configuration bitstream encryption.

Security Against Copying

The security key is securely stored in the Arria II GX device and cannot be read out through any interface. In addition, as configuration file read-back is not supported in Arria II GX devices, your design information cannot be copied.

Security Against Reverse Engineering

Reverse engineering from an encrypted configuration file is very difficult and time consuming because the Arria II GX configuration file formats are proprietary and the file contains millions of bits which require specific decryption. Reverse engineering the Arria II GX device is just as difficult because the device is manufactured on the most advanced 40-nm process technology.

Security Against Tampering

Once the Tamper Protection bit is set in the key programming file generated by the Quartus II software, the Arria II GX device can only be configured with configuration files encrypted with the same key. Tampering is prevented using both volatile and non-volatile keys.

AES Decryption Block

The main purpose of the AES decryption block is to decrypt the configuration bitstream prior to entering data decompression or configuration.

Prior to receiving encrypted data, you must enter and store the 256-bit security key in the device. You can choose between a non-volatile security key and a volatile security key with battery backup.

The security key is scrambled prior to storing it in the key storage to make it more difficult for anyone to retrieve the stored key using de-capsulation of the device.

Flexible Security Key Storage

Arria II GX devices support two types of security key programming: volatile and non-volatile. Table 9–18 shows the differences between volatile keys and non-volatile keys.

Table 9–18. Security Key Options

Options	Volatile Key	Non-Volatile Key
Key programmability	Reprogrammable and erasable	One-time programmable
External battery	Required	Not required
Key programming method *(1)*	On-board	On and off board
Design protection	Secure against copying and reverse engineering. Tamper resistant if volatile tamper protection bit is set.	Secure against copying and reverse engineering. Tamper resistant if tamper protection bit is set.

Note to Table 9–18:

(1) Key programming is carried out using the JTAG interface.

You can program the non-volatile key to the Arria II GX device without an external battery. Also, there are no additional requirements of any of the Arria II GX power supply inputs.

V_{CCBAT} is a dedicated power supply for volatile key storage and not shared with other on-chip power supplies, such as V_{CCIO} or V_{CC}. V_{CCBAT} continuously supplies power to the volatile register regardless of the on-chip supply condition.

After power-up, wait 100 ms (standard POR delay) or 4 ms (fast POR delay) before beginning the key programming to ensure that V_{CCBAT} is at its full rail.

As an example, the following are lithium coin-cell type batteries used for volatile key storage purposes: BR1220 (-30° to +80°C) and BR2477A (-40°C to +125°C).

For more information about battery specifications, refer to the *Device Data Sheet* chapter in volume 3 of the *Arria II GX Device Handbook*.

For more information about the VCCBAT pin connection recommendations, refer to *Arria II GX Device Family Pin Connection Guidelines*.

Arria II GX Design Security Solution

Arria II GX devices are SRAM-based devices. To provide design security, Arria II GX devices require a 256-bit security key for configuration bitstream encryption.

You can carry out secure configuration in the following three steps, as shown in Figure 9–29:

1. Program the security key into the Arria II GX device.

   Program the user-defined 256-bit AES keys to the Arria II GX device through the JTAG interface.

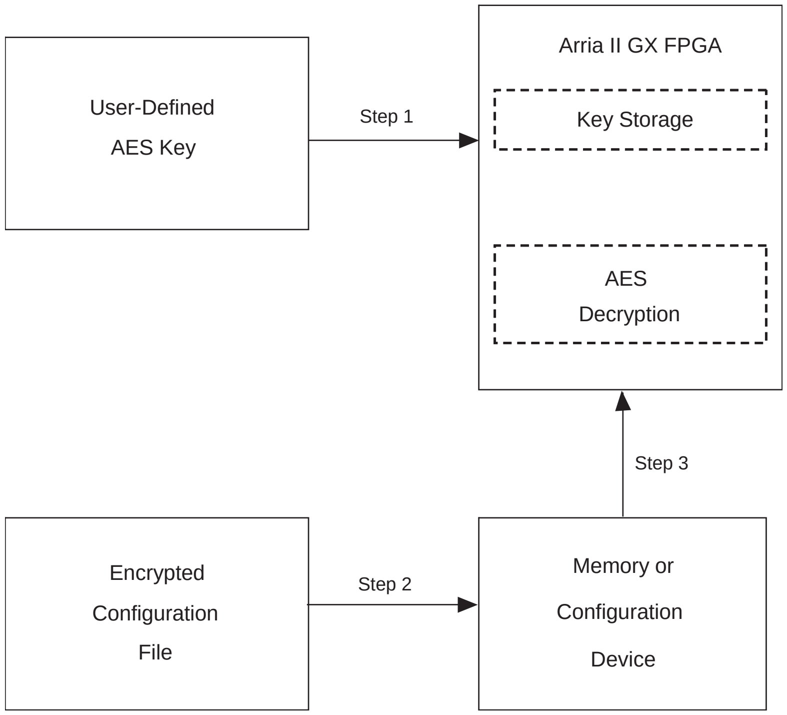
2. Encrypt the configuration file and store it in the external memory.

   Encrypt the configuration file with the same 256-bit keys used to program the Arria II GX device. Encryption of the configuration file is done using the Quartus II software. The encrypted configuration file is then loaded into the external memory, such as a configuration or flash device.

3. Configure the Arria II GX device.

At system power-up, the external memory device sends the encrypted configuration file to the Arria II GX device.

Figure 9–29. Design Security *(Note 1)*



Note to Figure 9–29:

(1) Step 1, Step 2, and Step 3 correspond to the procedure detailed in "Design Security" on page 9–54.

Security Modes Available

The following security modes are available on Arria II GX devices:

Volatile Key

Secure operation with volatile key programmed and required external battery—this mode accepts both encrypted and unencrypted configuration bitstreams. Use the unencrypted configuration bitstream support for board-level testing only.

Non-Volatile Key

Secure operation with one-time-programmable (OTP) security key programmed—this mode accepts both encrypted and unencrypted configuration bitstreams. Use the unencrypted configuration bitstream support for board level testing only.

Volatile Key with Tamper Protection Bit Set

Secure operation in tamper resistant mode with volatile security key programmed—only encrypted configuration bitstreams are allowed to configure the device. Tamper protection disables JTAG configuration with unencrypted configuration bitstream.

☞ Enabling the Tamper Protection bit disables the test mode in Arria II GX devices. This process is irreversible and prevents Altera from carry-out failure analysis. Contact Altera Technical Support to enable the tamper protection bit.

Non-Volatile Key with Tamper Protection Bit Set

Secure operation in tamper resistant mode with OTP security key programmed—only encrypted configuration bitstreams are allowed to configure the device. Tamper protection disables JTAG configuration with unencrypted configuration bitstream.

☞ Enabling the Tamper Protection bit disables the test mode in Arria II GX devices. This process is irreversible and prevents Altera from carry out failure analysis. Contact Altera Technical Support to enable the tamper protection bit.

Volatile or Non-Volatile Key with JTAG Anti-Tamper Protection Bit Set

All the JTAG instructions except `PULSE_NCONFIG`, `BYPASS`, `KEY_VERIFY`, `KEY_CLR_VREG`, and `KEY_PROG_VOL` are disabled.

☞ Enabling the JTAG Anti-Tamper protection bit disables the boundary-scan test (BST) features in Arria II GX devices.

No Key Operation

Only unencrypted configuration bitstreams are allowed to configure the device.

Table 9–19 summarizes the different security modes and configuration bitstream supported for each mode.

Table 9–19. Security Modes Supported

Mode *(1)*	Function	Configuration File
Volatile key	Secure	Encrypted
	Board-level testing	Unencrypted
Non-volatile key	Secure	Encrypted
	Board-level testing	Unencrypted
Volatile key with tamper protection bit set	Secure (tamper resistant) *(2)*	Encrypted
Non-volatile key with tamper protection bit set	Secure (tamper resistant) *(2)*	Encrypted
Volatile or non-volatile key with JTAG anti-tamper protection bit set	JTAG tamper resistant	*(3)*

Notes to Table 9–19:

(1) In No key operation, only the unencrypted configuration file is supported.

(2) The tamper protection bit setting does not prevent the device from being reconfigured.

(3) It does not control the decryption data path.

Supported Configuration Schemes

The Arria II GX device supports only selected configuration schemes, depending on the security mode you select when you encrypt the Arria II GX device.

Figure 9–30 shows the restrictions of each security mode when encrypting Arria II GX devices.

Figure 9–30. Arria II GX Security Modes—Sequence and Restrictions

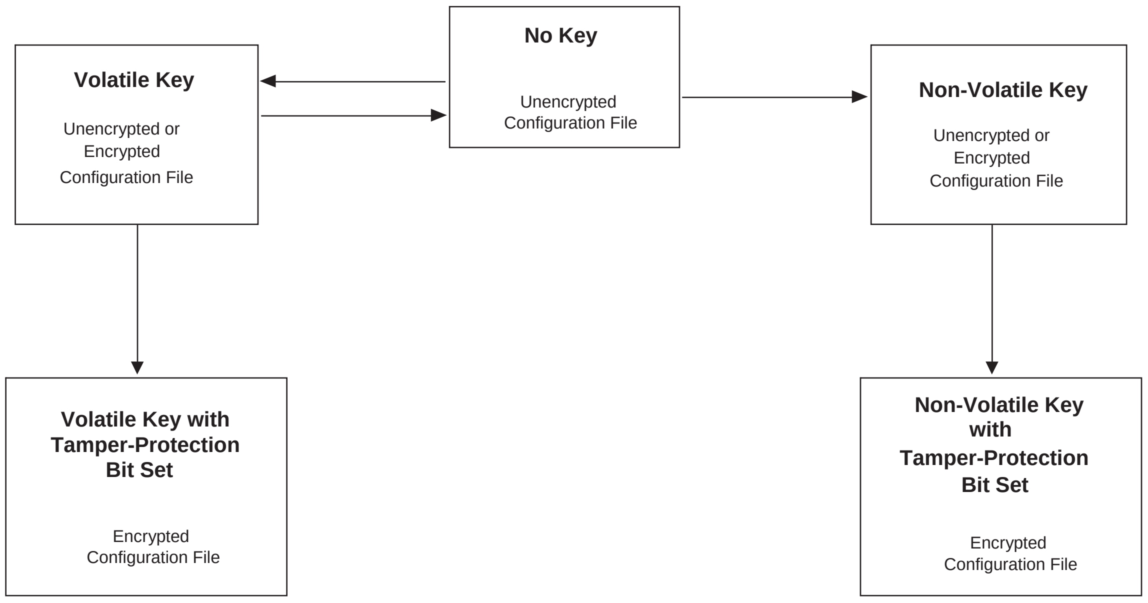

No Key
Unencrypted Configuration File
Volatile Key
Unencrypted or Encrypted Configuration File
Non-Volatile Key
Unencrypted or Encrypted Configuration File
Volatile Key with Tamper-Protection Bit Set
Encrypted Configuration File
Non-Volatile Key with Tamper-Protection Bit Set
Encrypted Configuration File

Table 9–20 shows the configuration modes allowed in each of the security modes.

Table 9–20. Allowed Configuration Modes for Various Security Modes *(Note 1)*

Security Mode	Configuration File	Allowed Configuration Modes
No key	Unencrypted	All configuration modes that do not engage the design security feature.
Secure with volatile key	Encrypted	■ Passive serial with AES (and/or with decompression) ■ Fast passive parallel with AES (and/or with decompression) ■ Remote update fast AS with AES (and/or with decompression) ■ AS (and/or with decompression)
Board-level testing with volatile key	Unencrypted	All configuration modes that do not engage the design security feature.
Secure with non-volatile key	Encrypted	■ Passive serial with AES (and/or with decompression) ■ Fast passive parallel with AES (and/or with decompression) ■ Remote update fast AS with AES (and/or with decompression) ■ AS (and/or with decompression)
Board-level testing with non-volatile key	Unencrypted	All configuration modes that do not engage the design security feature.
Secure in tamper resistant mode using volatile or non-volatile key with tamper protection set	Encrypted	■ Passive serial with AES (and/or with decompression) ■ Fast passive parallel with AES (and/or with decompression) ■ Remote update fast AS with AES (and/or with decompression) ■ AS (and/or with decompression)

Note to Table 9–20:

(1) There is no impact to the configuration time required compared to unencrypted configuration modes except when using FPP with AES (and/or decompression), which requires DCLK that is 4× the data rate.

The design security feature is available in all configuration methods except JTAG. Therefore, you can use the design security feature in FPP mode (when using an external controller, such as a MAX II device or a microprocessor and flash memory), or in fast AS and PS configuration schemes.

Table 9–21 summarizes the configuration schemes that support the design security feature both for volatile key and non-volatile key programming.

Table 9–21. Design Security Configuration Schemes Availability (Part 1 of 2)

Configuration Scheme	Configuration Method	Design Security
FPP	MAX II device or microprocessor and flash memory	✓ *(1)*
	Enhanced configuration device	—
Fast AS	Serial configuration device	✓
PS	MAX II device or microprocessor and flash memory	✓
	Download cable	✓

Table 9–21. Design Security Configuration Schemes Availability (Part 2 of 2)

Configuration Scheme	Configuration Method	Design Security
JTAG *(2)*	MAX II device or microprocessor and flash memory	—
	Download cable	—

Notes to Table 9–21:

(1) In this mode, the host system must send a DCLK that is 4× the data rate.

(2) JTAG configuration supports only unencrypted configuration file.

You can use the design security feature with other configuration features, such as compression and remote system upgrade features. When you use compression with the design security feature, the configuration file is first compressed and then encrypted using the Quartus II software. During configuration, the Arria II GX device first decrypts and then decompresses the configuration file.

Document Revision History

Table 9–22 shows the revision history for this document.

Table 9–22. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial release.	—

10. SEU Mitigation in Arria II GX Devices

AIIGX51010-1.0

Introduction

In critical applications such as avionics, telecommunications, system control, and military applications, it is important to be able to do the following:

- Confirm that the configuration data stored in an Arria® II GX device is correct.
- Alert the system to the occurrence of a configuration error.

The error detection feature is enhanced in the Arria II GX device family. This section describes how to activate and use the error detection cyclical redundancy check (CRC) feature when your Arria II GX device is in user mode and describes how to recover from configuration errors caused by CRC errors.

For Arria II GX devices, the error detection CRC feature is provided in the Quartus® II software starting with version 9.0.

Using the error detection CRC feature for the Arria II GX device family has no impact on fitting or performance.

This chapter contains the following sections:

- "Error Detection Fundamentals" on page 10–1
- "Configuration Error Detection" on page 10–2
- "User Mode Error Detection" on page 10–2
- "Error Detection Pin Description" on page 10–5
- "Error Detection Block" on page 10–5
- "Error Detection Timing" on page 10–8
- "Recovering From CRC Errors" on page 10–11

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

Error Detection Fundamentals

Error detection determines if the data received through a medium is corrupted during transmission. To accomplish this, the transmitter uses a function to calculate a checksum value for the data and appends the checksum to the original data frame. The receiver uses the same calculation methodology to generate a checksum for the received data frame and compares the received checksum to the transmitted checksum. If the two checksum values are equal, the received data frame is correct and no data corruption occurred during transmission or storage.

The error detection CRC feature uses the same concept. When Arria II GX devices are successfully configured and are in user mode, the error detection CRC feature ensures the integrity of the configuration data.

Configuration Error Detection

In configuration mode, a frame-based CRC is stored in the configuration data and contains the CRC value for each data frame.

During configuration, the Arria II GX device calculates the CRC value based on the frame of data that is received and compares it against the frame CRC value in the data stream. Configuration continues until either the device detects an error or configuration is complete.

In Arria II GX devices, the CRC value is calculated during the configuration stage. A parallel CRC engine generates 16 CRC check bits per frame and then stores them into the configuration random access memory (CRAM). The CRAM chain used for storing CRC check bits is 16-bits wide; its length is equal to the number of frames in the device.

User Mode Error Detection

Arria II GX devices have built-in error detection circuitry to detect data corruption by soft errors in the CRAM cells. This feature allows all CRAM contents to be read and verified to match a configuration-computed CRC value. Soft errors are changes in a CRAM's bit state due to an ionizing particle.

The error detection capability continuously computes the CRC of the configured CRAM bits and compares it with the pre-calculated CRC. If the CRCs match, there is no error in the current configuration CRAM bits. The process of error detection continues until the device is reset (by setting `nCONFIG` low).

When the device transitions into user mode, the error detection process is enabled if you enabled the **CRC error detection** option in the Quartus II software.

A single 16-bit CRC calculation is done on a per-frame basis. Once it has finished the CRC calculation for a frame, the resulting 16-bit signature is `hex 0000` if there are no detected CRAM bit errors in a frame by the error detection circuitry and the output signal `CRC_ERROR` is 0. If a CRAM bit error is detected by the circuitry in a frame in the device, the resulting signature is non-zero. This causes the CRC engine to start searching the error bit location.

The error detection logic in Arria II GX devices calculates CRC check bits for each frame and pulls the `CRC_ERROR` pin high when it detects bit errors in the chip. Within a frame, it can detect all single-bit, double-bit, and three-bit errors. The probability of more than three CRAM bits being flipped by an single event upset (SEU) is very low. In general, the probability of detection for all error patterns is 99.998%.

The CRC engine reports the bit location and determines the type of error for all single-bit errors and over 99.641% of double-adjacent errors. The probability of other error patterns is very low and report of location of bit flips is not guaranteed by the CRC engine.

You can also read-out the error bit location through the Joint Test Action Group (JTAG) and the core interface. You must shift these bits out from the error message register through either the JTAG instruction, `SHIFT_EDERROR_REG`, or the core interface, before the CRC detects the next error in another frame. The CRC circuit continues to run, and if an error is detected, you need to decide whether to complete a reconfiguration or to ignore the CRC error.

For more information about the timing requirement to shift out error information from the error message register, refer to "Error Detection Timing" on page 10–8.

The error detection logic continues to calculate the `CRC_ERROR` and 16-bit signatures for the next frame of data regardless of whether an error has occurred in the current frame or not. You need to monitor these signals and take the appropriate actions if a soft error occurs.

The error detection circuitry in Arria II GX devices uses a 16-bit CRC-ANSI standard (16-bit polynomial) as the CRC generator. The computed 16-bit CRC signature for each frame is stored in the CRAM. The total storage size is 16 (number of bits per frame) × the number of frames.

The Arria II GX device error detection CRC feature does not check memory blocks and I/O buffers. Thus, the `CRC_ERROR` signal may stay solid high or low, depending on the error status of the previously checked CRAM frame. The I/O buffers are not verified during error detection because these bits use flipflops as storage elements that are more resistant to soft errors compared to CRAM cells. MLAB and M9K memory blocks support parity bits that are used to check the contents of memory blocks for any error.

For more information about error detection in Arria II GX memory blocks, refer to the *Embedded Memory Blocks in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook.*

To provide testing capability of the error detection block, a JTAG instruction, `EDERROR_INJECT`, is provided. This instruction is able to change the content of the 21-bit JTAG fault injection register, used for error injection in Arria II GX devices, thereby enabling testing of the error detection block.

You can only execute the `EDERROR_INJECT` JTAG instruction when the device is in user mode.

Table 10–1 describes the `EDERROR_INJECT` JTAG instruction.

Table 10–1. EDERROR_INJECT JTAG Instruction

JTAG Instruction	Instruction Code	Description
`EDERROR_INJECT`	00 0001 0101	This instruction controls the 21-bit JTAG fault injection register, which is used for error injection.

You can create a Jam™ file (**.jam**) to automate the testing and verification process. This allows you to verify the CRC functionality in-system and on-the-fly, without having to reconfigure the device. You can then switch to the CRC circuit to check for real errors induced by an SEU.

You can introduce a single error or double errors adjacent to each other to the configuration memory. This provides an extra way to facilitate design verification and system fault tolerance characterization. Use the JTAG fault injection register with `EDERROR_INJECT` instruction to flip the readback bits. The Arria II GX device is then forced into error test mode. Altera recommends you reconfigure the device after the test completes.

The content of the JTAG fault injection register is not loaded into the fault injection register during the processing of the last frame and the first frame. It is only loaded at the end of this period.

You can only introduce error injection in the first data frame, but you can monitor the error information at any time. For more information about the JTAG fault injection register and fault injection register, refer to "Error Detection Registers" on page 10–7.

Table 10–2 shows how the fault injection register is implemented and describes error injection.

Table 10–2. Fault Injection Register

Bit	**Bit[20..19]**			**Bit[18..8]**	**Bit[7..0]**
Description	**Error Type**			**Byte Location of the Injected Error**	**Error Byte Value**
Content	Error Type (1)		Error injection type	Depicts the location of the injected error in the first data frame.	Depicts the location of the bit error and corresponds to the error injection type selection.
	Bit[20]	Bit[19]			
	0	1	Single byte error injection		
	1	0	Double-adjacent byte error injection		
	0	0	No error injection		

Note to Table 10–2:

(1) Bit[20] and Bit[19] cannot both be set to 1, as this is not a valid selection. The error detection circuitry decodes this as no error injection.

Automated Single Event Upset Detection

Arria II GX devices offer on-chip circuitry for automated checking of single event upset detection. Some applications that require the device to operate error-free in high-neutron flux environments require periodic checks to ensure continued data integrity. The error detection CRC feature ensures data reliability and is one of the best options for mitigating SEU.

You can implement the error detection CRC feature with existing circuitry in Arria II GX devices, eliminating the need for external logic. The `CRC_ERROR` pin reports a soft error when configuration CRAM data is corrupted; you must decide whether to reconfigure the device or to ignore the error.

Error Detection Pin Description

The following sections describe the CRC_ERROR pin.

CRC_ERROR Pin

Table 10–3 describes the CRC_ERROR pin.

Table 10–3. CRC_ERROR Pin Description

Pin Name	Pin Type	Description
CRC_ERROR	I/O, output open-drain	Active high signal that indicates that the error detection circuit has detected errors in the configuration CRAM bits. This is an optional pin and is used when the error detection CRC circuit is enabled. When the error detection CRC circuit is disabled, it is a user I/O pin. The CRC error output, when using the WYSIWYG function, is a dedicated path to the CRC_ERROR pin. To use the CRC_ERROR pin, tie this pin to VCCPGM through a 10-kΩ resistor. Alternatively, depending on the input voltage specification of the system receiving the signal, tie this pin to a different pull-up voltage.

☞ WYSIWYG is an optimization technique that performs optimization on the Verilog Quartus Mapping (VQM) netlist within the Quartus II software.

Error Detection Block

The error detection block contains the logic necessary to calculate the 16-bit CRC signature for the configuration CRAM bits in the Arria II GX device.

The CRC circuit continues running even if an error occurs. When a soft error occurs, the device sets the CRC_ERROR pin high. The two types of CRC detection that check the configuration bits are shown in Table 10–4.

Table 10–4. Two Types of CRC Detection

First Type of CRC Detection	Second Type of CRC Detection
■ This is the CRAM error checking ability (16-bit CRC) during user mode for use by the CRC_ERROR pin. ■ For each frame of data, the pre-calculated 16-bit CRC enters the CRC circuit at the end of the frame data and determines whether there is an error or not. ■ If an error occurs, the search engine finds the location of the error. ■ The error messages can be shifted out through the JTAG instruction or core interface logics while the error detection block continues running. ■ The JTAG interface reads out the 16-bit CRC result for the first frame and also shifts the 16-bit CRC bits to the 16-bit CRC storage registers for test purposes. ■ Single error, double errors, or double errors adjacent to each other can be deliberately introduced to configuration memory for testing and design verification.	■ This is the 16-bit CRC that is embedded in every configuration data frame. ■ During configuration, after a frame of data is loaded into the Arria II GX device, the pre-computed CRC is shifted into the CRC circuitry. ■ At the same time, the CRC value for the data frame shifted-in is calculated. If the pre-computed CRC and calculated CRC values do not match, nSTATUS is set low. Every data frame has a 16-bit CRC; therefore, there are many 16-bit CRC values for the whole configuration bitstream. Every device has different lengths of the configuration data frame.

☞ The "Error Detection Block" section focuses on the first type, the 16-bit CRC only, when the device is in user mode.

Error Detection Registers

There is one set of 16-bit registers in the error detection circuitry that stores the computed CRC signature. A non-zero value on the syndrome register causes the CRC_ERROR pin to be set high.

Figure 10–1 shows the block diagram of the error detection circuitry, syndrome registers, and error injection block.

Figure 10–1. Error Detection Circuitry, Syndrome Registers, and Error Injection Block

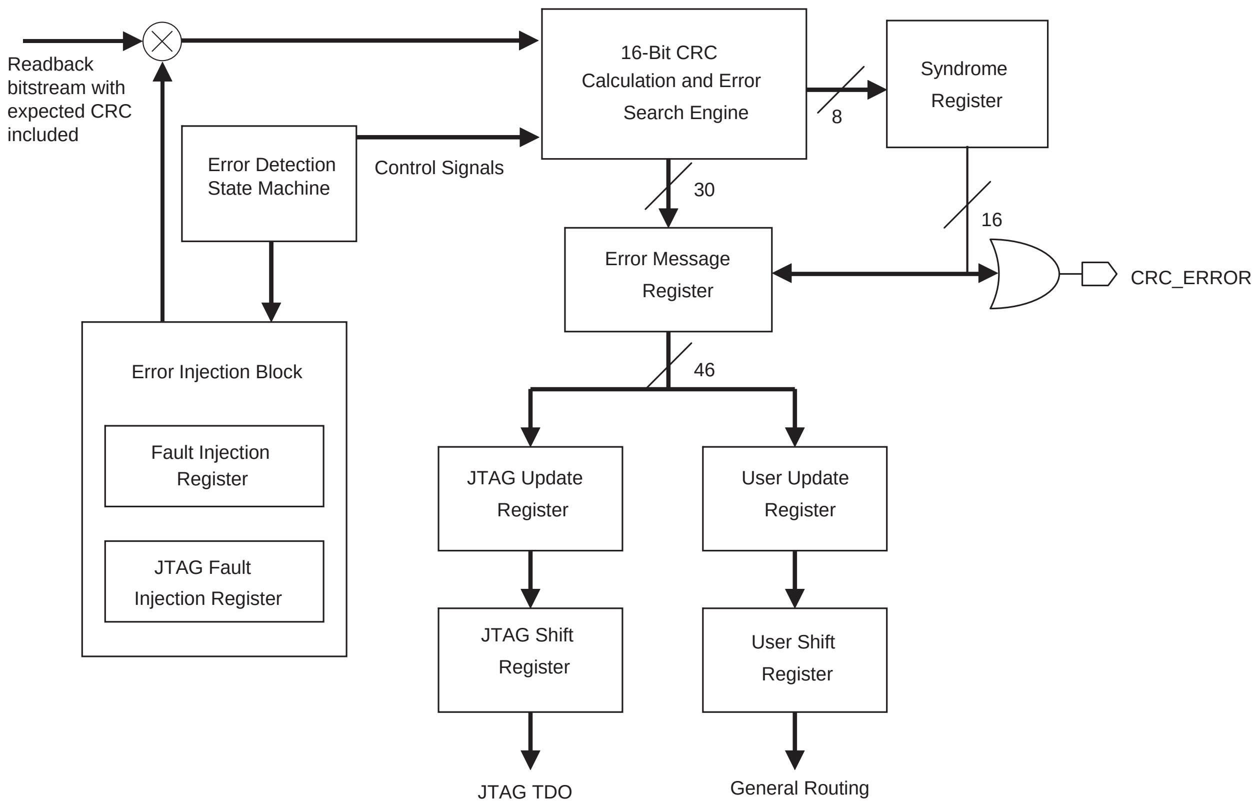


Table 10–5 defines the registers shown in Figure 10–1.

Table 10–5. Error Detection Registers (Part 1 of 2)

Register	Description
Syndrome Register	This register contains the CRC signature of the current frame through the error detection verification cycle. The CRC_ERROR signal is derived from the contents of this register.
Error Message Register	This 46-bit register contains information on the error type, location of the error, and the actual syndrome. The types of errors and location reported are single- and double-adjacent bit errors. The location bits for other types of errors are not identified by the error message register. The content of the register is shifted out through the JTAG SHIFT_EDERROR_REG instruction or to the core through the core interface.

Table 10–5. Error Detection Registers (Part 2 of 2)

Register	Description
JTAG Update Register	This register is automatically updated with the contents of the error message register one cycle after the 46-bit register content is validated. It includes a clock enable, which must be asserted prior to being sampled into the JTAG shift register. This requirement ensures that the JTAG Update Register is not being written into by the contents of the Error Message Register at exactly the same time that the JTAG shift register is reading its contents.
User Update Register	This register is automatically updated with the contents of the error message register one cycle after the 46-bit register content is validated. It includes a clock enable, which must be asserted prior to being sampled into the user shift register. This requirement ensures that the user update register is not being written into by the contents of the error message register at exactly the same time that the user shift register is reading its contents.
JTAG Shift Register	This register is accessible by the JTAG interface and allows the contents of the JTAG update register to be sampled and read out by JTAG instruction `SHIFT_EDERROR_REG`.
User Shift Register	This register is accessible by the core logic and allows the contents of the user update register to be sampled and read by user logic.
JTAG Fault Injection Register	This 21-bit register is fully controlled by the JTAG instruction `EDERROR_INJECT`. This register holds the information of the error injection that you want in the bitstream.
Fault Injection Register	The content of the JTAG fault injection register is loaded into this 21-bit register when it is being updated.

Error Detection Timing

When the error detection CRC feature is enabled through the Quartus II software, the device automatically activates the CRC process upon entering user mode, after configuration, and after initialization is complete.

If an error is detected within a frame, `CRC_ERROR` is driven high at the end of the error location search after the error message register is updated. At the end of this cycle, the `CRC_ERROR` pin is pulled low for a minimum of 32 clock cycles. If the next frame contains an error, `CRC_ERROR` is driven high again after the error message register is overwritten by the new value. You can start to unload the error message on each rising edge of the `CRC_ERROR` pin. Error detection runs until the device is reset.

The error detection circuitry runs off an internal configuration oscillator with a divisor that sets the maximum frequency. Table 10–6 shows the minimum and maximum error detection frequencies.

Table 10–6. Minimum and Maximum Error Detection Frequencies

Device Type	Error Detection Frequency	Maximum Error Detection Frequency	Minimum Error Detection Frequency	Valid Divisors (n)
Arria II GX	100 MHz / 2^n	50 MHz	390 kHz	1, 2, 3, 4, 5, 6, 7, 8

You can set a lower clock frequency by specifying a division factor in the Quartus II software (refer to "Software Support" on page 10–10). The divisor is a power of two (2), where *n* is between 1 and 8. The divisor ranges from 2 through 256. See the following equation:

Equation 10–1.

$$\text{error detection frequency} = \frac{100\text{MHz}}{2^n}$$

The error detection frequency reflects the frequency of the error detection process for a frame because the CRC calculation in Arria II GX devices is done on a per-frame basis.

The error message register is updated whenever an error or errors occur. If the error location and message are not shifted out before the next error location is found, the previous error location and message is overwritten by the new information. To avoid this, you must shift these bits out within one frame CRC verification. The minimum interval time between each update for the error message register depends on the device and the error detection clock frequency. However, slowing down the error detection clock frequency slows down the error recovery time for the SEU event.

Table 10–7 shows the estimated minimum interval time between each update for the error message register for Arria II GX devices.

Table 10–7. Minimum Update Interval for Error Message Register *(Note 1)*

Device	Timing Interval (μs)
EP2AGX20	8.12
EP2AGX30	8.12
EP2AGX45	12.24
EP2AGX65	12.24
EP2AGX95	16.08
EP2AGX125	16.08
EP2AGX190	20.84
EP2AGX260	20.84

Note to Table 10–7:

(1) These timing numbers are preliminary.

CRC calculation time for the error detection circuitry to check from the first until the last frame depends on the device and the error detection clock frequency.

Table 10–8 shows the estimated time for each CRC calculation with minimum and maximum clock frequencies for Arria II GX devices. The minimum CRC calculation time is calculated with the maximum error detection frequency with divisor factor 1 while the maximum CRC calculation time is calculated with the minimum error detection frequency with divisor factor 8.

Table 10–8. CRC Calculation Time

Device	Minimum Time (ms)	Maximum Time (s)
EP2AGX20	18.06	5.15
EP2AGX30	18.06	5.15
EP2AGX45	44.00	12.20
EP2AGX65	44.00	12.20
EP2AGX95	82.40	22.82
EP2AGX125	82.40	22.82
EP2AGX190	160.00	44.36
EP2AGX260	160.00	44.36

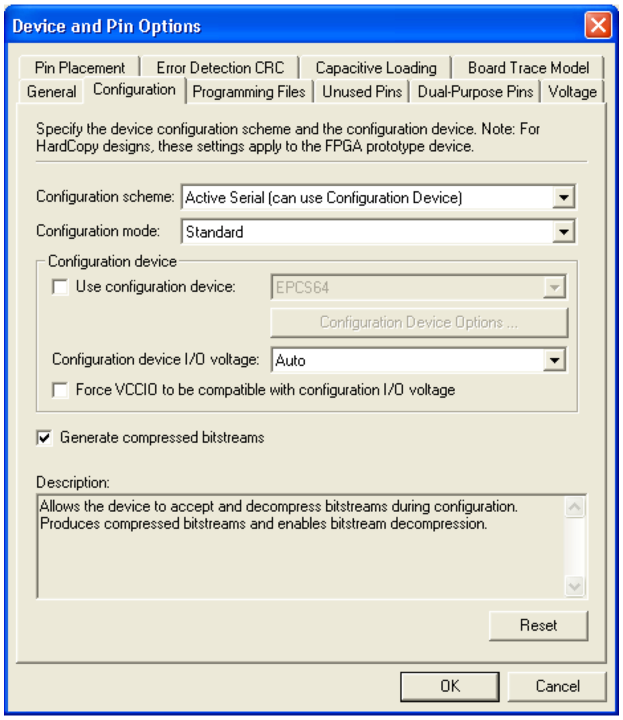
Software Support

The Quartus II software, starting with version 8.1, supports the error detection CRC feature for Arria II GX devices. Enabling this feature in the **Device and Pin Options** dialog box generates the `CRC_ERROR` output to the optional dual purpose `CRC_ERROR` pin.

Enable the error detection feature using CRC by performing the following steps:

1. Open the Quartus II software and load a project using an Arria II GX device.
2. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
3. In the **Category** list, select **Device**. The **Device** screen appears.
4. Click **Device and Pin Options**. The **Device and Pin Options** dialog box appears (see Figure 10–2).
5. In the **Device and Pin Options** dialog box, click the **Error Detection CRC** tab.
6. Turn on **Enable error detection CRC** (Figure 10–2).

Figure 10–2. Enabling the Error Detection CRC Feature in the Quartus II Software



Specify the device configuration scheme and the configuration device. Note: For HardCopy designs, these settings apply to the FPGA prototype device.

Configuration scheme: Active Serial (can use Configuration Device)

Configuration mode: Standard

Configuration device

☐ Use configuration device: EPCS64

Configuration device I/O voltage: Auto

☐ Force VCCIO to be compatible with configuration I/O voltage

☑ Generate compressed bitstreams

Description:

Allows the device to accept and decompress bitstreams during configuration. Produces compressed bitstreams and enables bitstream decompression.

7. In the **Divide error check frequency by** box, enter a valid divisor as documented in Table 10–6 on page 10–8.
8. Click **OK**.

Recovering From CRC Errors

The system that the Arria II GX device resides in must control device reconfiguration. After detecting an error on the `CRC_ERROR` pin, strobing the `nCONFIG` signal low directs the system to perform the reconfiguration at a time when it is safe.

When the data bit is rewritten with the correct value by reconfiguring the device, the device functions correctly.

While soft errors are uncommon in Altera devices, certain high-reliability applications may require a design to account for these errors.

Document Revision History

Table 10–9 shows the revision history for this chapter.

Table 10–9. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial release.	—

11. JTAG Boundary-Scan Testing

AIIGX51011-1.0

Introduction

This chapter describes the boundary-scan test (BST) features that are supported in Arria® II GX devices. The features are similar to Arria GX devices, unless stated in this document.

Arria II GX devices support IEEE Std. 1149.1 and IEEE Std. 1149.6. The IEEE Std. 1149.6 is only supported on the high-speed serial interface (HSSI) transceivers in Arria II GX devices. The purpose of IEEE Std. 1149.6 is to enable board-level connectivity checking between transmitters and receivers that are AC coupled (connected with a capacitor in series between the source and destination).

This chapter includes the following sections:

- "IEEE Std. 1149.6 Boundary-Scan Register" on page 11–1
- "BST Operation Control" on page 11–3
- "I/O Voltage Support in a JTAG Chain" on page 11–5
- "Boundary-Scan Description Language Support" on page 11–6

For information about the following topics, refer to the *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*:

- IEEE Std. 1149.1 BST architecture and circuitry
- TAP controller state-machine
- IEEE Std. 1149.1 JTAG instructions
- JTAG instructions code with descriptions
- IEEE Std. 1149.1 BST guidelines

IEEE Std. 1149.6 Boundary-Scan Register

The boundary-scan cell (BSC) for HSSI transmitters (`GXB_TX[p,n]`) and receivers/input clock buffer (`GXB_RX[p,n]`)/(`REFCLK[0..7]`) in Arria II GX devices are different from the BSCs for I/O pins.

Figure 11–1 shows the Arria II GX HSSI transmitter boundary-scan cell.

Figure 11–1. Arria II GX HSSI Transmitter BSC with IEEE Std. 1149.6 BST Circuitry

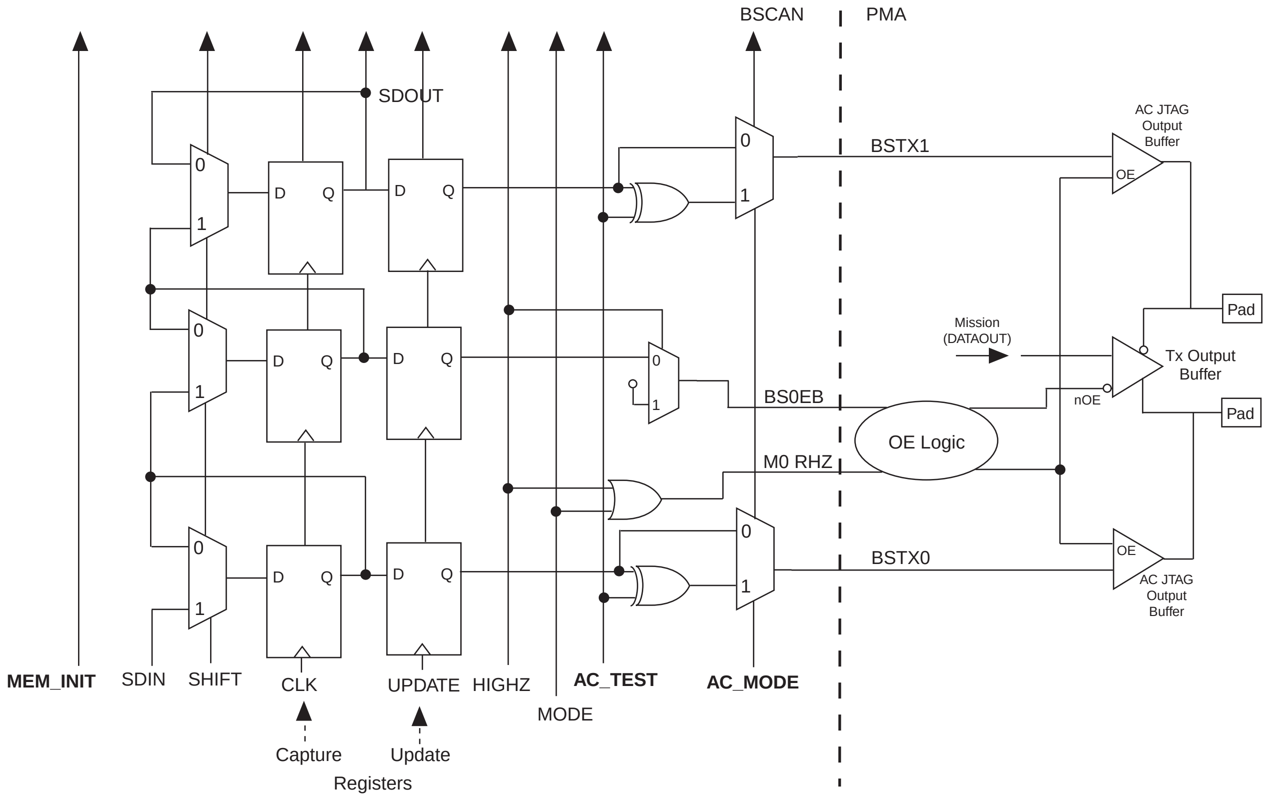
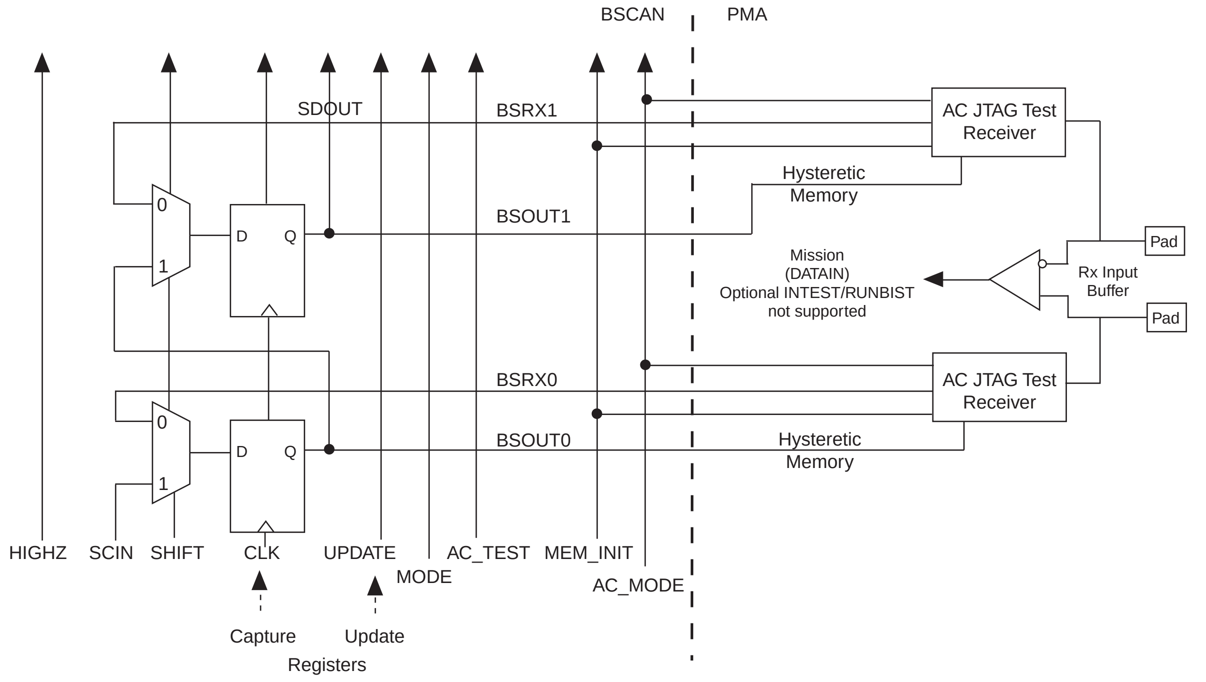


Figure 11–2 shows the Arria II GX HSSI receiver/input clock buffer BSC.

Figure 11–2. Arria II GX HSSI Receiver/Input Clock Buffer BSC with IEEE Std. 1149.6 BST Circuitry



For information about Arria II GX user I/O boundary-scan cells, refer to the *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

BST Operation Control

Table 11–1 shows the boundary-scan register length for Arria II GX devices.

Table 11–1. Arria II GX Boundary-Scan Register Length

Device	Boundary-Scan Register Length
EP2AGX20	819
EP2AGX30	819
EP2AGX45	1227
EP2AGX65	1227
EP2AGX95	1467
EP2AGX125	1467
EP2AGX190	1971
EP2AGX260	1971

Table 11–2 shows the IDCODE information for Arria II GX devices.

Table 11–2. 32-Bit Arria II GX Device IDCODE

Device	IDCODE (32 Bits) (1)			
	Version (4 Bits)	Part Number (16 Bits)	Manufacturer Identity (11 Bits)	LSB (1 Bit) (2)
EP2AGX20	0000	0010010100010001	000 0110 1110	1
EP2AGX30	0000	0010010100000001	000 0110 1110	1
EP2AGX45	0000	0010010100010010	000 0110 1110	1
EP2AGX65	0000	0010010100000010	000 0110 1110	1
EP2AGX95	0000	0010010100010011	000 0110 1110	1
EP2AGX125	0000	0010010100000011	000 0110 1110	1
EP2AGX190	0000	0010010100010100	000 0110 1110	1
EP2AGX260	0000	0010010100000100	000 0110 1110	1

Notes to Table 11–2:

(1) The MSB is on the left.

(2) The IDCODE LSB is always 1.

☞ To read IDCODE correctly, issue the IDCODE instruction after initialization, which is signaled by nSTATUS going high.

IEEE Std.1149.6 mandates the addition of two new instructions: EXTEST_PULSE and EXTEST_TRAIN. These two instructions enable edge-detecting behavior on the signal path containing the AC pins.

☞ When JTAG Anti Tamper mode is enabled, boundary scan test is disabled.

EXTEST_PULSE

The instruction code for EXTEST_PULSE is 0010001111. The EXTEST_PULSE instruction generates three output transitions:

- Driver drives data on the falling edge of TCK in UPDATE_IR/DR.
- Driver drives inverted data on the falling edge of TCK after entering the RUN_TEST/IDLE state.
- Driver drives data on the falling edge of TCK after leaving the RUN_TEST/IDLE state.

EXTEST_TRAIN

The instruction code for EXTEST_TRAIN is 0001001111. The EXTEST_TRAIN instruction behaves the same as the EXTEST_PULSE instruction with one exception. The output continues to toggle on the TCK falling edge as long as the TAP controller is in the RUN_TEST/IDLE state.

I/O Voltage Support in a JTAG Chain

An Arria II GX device operating in BST mode uses four required pins: `TDI`, `TDO`, `TMS`, and `TCK`. All JTAG input pins are powered by the V_{CCIO} power supply. The `TDO` pin is powered up by the V_{CCIO} and V_{CCPD} power supply of I/O Bank 8C. You must connect V_{CCPD} according to the I/O standard used in the same bank:

- For 3.3-V I/O standards, connect V_{CCPD} to 3.3 V
- For 3.0-V I/O standards, connect V_{CCPD} to 3.0 V
- For 2.5-V and below I/O standards, connect V_{CCPD} to 2.5 V

The JTAG chain supports several devices. However, use caution if the chain contains devices that have different V_{CCIO} levels.

Table 11–3 shows board design recommendations to ensure proper JTAG chain operation.

Table 11–3. Supported TDO/TDI Voltage Combinations

Device	**TDI Input Buffer Power**	**Arria II GX TDO V_{CCPD} and V_{CCIO} Voltage Level in I/O Bank 8C**				
		V_{CCIO} = 3.3 V	**V_{CCIO} = 3.0 V**	**V_{CCIO} = 2.5 V**	**V_{CCIO} = 1.8 V**	**V_{CCIO} = 1.5 V**
Arria II GX	V_{CCIO} = 3.3 V	✓ (1)	✓ (1)	✓ (2)	✓ (3)	Level shifter required
	V_{CCIO} = 3.0 V	✓ (1)	✓ (1)	✓ (2)	✓ (3)	Level shifter required
	V_{CCIO} = 2.5 V	✓ (1)	✓ (1)	✓ (2)	✓ (3)	Level shifter required
	V_{CCIO} = 1.8 V	✓ (1)	✓ (1)	✓ (2)	✓ (3)	Level shifter required
	V_{CCIO} = 1.5 V	✓ (1)	✓ (1)	✓ (2)	✓ (3)	✓
Non-Arria II GX	V_{CC} = 3.3 V	✓ (1)	✓ (1)	✓ (2)	✓ (3)	Level shifter required
	V_{CC} = 2.5 V	✓ (1), (4)	✓ (1), (4)	✓ (2)	✓ (3)	Level shifter required
	V_{CC} = 1.8 V	✓ (1), (4)	✓ (1), (4)	✓ (2), (5)	✓	Level shifter required
	V_{CC} = 1.5 V	✓ (1), (4)	✓ (1), (4)	✓ (2), (5)	✓ (6)	✓

Notes to Table 11–3:

(1) The TDO output buffer meets V_{OH} (Min.) = 2.4 V.
(2) The TDO output buffer meets V_{OH} (Min.) = 2.0 V.
(3) An external 250-Ω pull-up resistor is not required; however, they are recommended if signal levels on the board are not optimal.
(4) The input buffer must be 3.0-V tolerant.
(5) The input buffer must be 2.5-V tolerant.
(6) The input buffer must be 1.8-V tolerant.

For more information about I/O voltage support in the JTAG chain, refer to the *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

Boundary-Scan Description Language Support

The boundary-scan description language (BSDL), a subset of VHDL, provides a syntax that allows you to describe the features of an IEEE Std. 1149.6 BST-capable device that can be tested. You can test software development systems then use the BSDL files for test generation, analysis, and failure diagnostics.

For more information about BSDL files for IEEE Std. 1149.6-compliant Arria II GX devices, visit the Altera® website at www.altera.com.

You can also generate BSDL files (pre-configuration and post-configuration) for IEEE std. 1149.6-compliant Arria II GX devices with the Quartus® II software version 9.1 and later. For the procedure for generating BSDL files using the Quartus II software, visit the Altera website at www.altera.com.

Revision History

Table 11–4 shows the revision history for this document.

Table 11–4. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v 1.0	Initial release.	—

12. Power Requirements for Arria II GX Devices

AIIGX51012-1.0

Introduction

The total power of an Altera® Arria® II GX FPGA includes static power and dynamic power. Static power is the power consumed by the FPGA when it is configured, but no clocks are operating. Dynamic power is composed of switching power when the device is configured and running.

The Quartus® II software optimizes all designs with Arria II GX power technology to ensure performance is met at the lowest power consumption. This automatic process allows you to concentrate on the functionality of your design instead of the power consumption of your design.

This chapter includes the following sections:

For more information about using the PowerPlay Power Analyzer in the Quartus II software, refer to the *Power Estimation and Power Analysis* section in volume 3 of the *Quartus II Handbook*.

External Power Supply Requirements

This section describes the different external power supplies needed to power Arria II GX devices. Table 12–1 lists the external power supply pins for Arria II GX devices. You can supply some of the power supply pins with the same external power supply, provided they need the same voltage level.

For each Altera recommended power supply's operating conditions, refer to the *Device Data Sheet* chapter in volume 3 of the *Arria II GX Device Handbook*.

For power supply pin connection guidelines and power regulator sharing, refer to the *Arria II GX Device Family Pin Connection Guidelines*.

Table 12–1. Arria II GX Power Supply Requirements (Part 1 of 2)

Power Supply Pin	Nominal Voltage Level (V)	Description
VCC	0.9 V	Supplies power to the core, periphery, I/O registers, PCIe HIP block, and transceiver PCS
VCCD_PLL	0.9 V	Supplies power to the digital portions of the PLL
VCCA_PLL *(1)*	2.5 V	Supplies power to the analog portions of the PLL and device-wide power management circuitry
VCCCB	1.5 V	Supplies power to the configuration RAM bits
VCCPD	2.5 V, 3.0 V, 3.3 V	Supplies power to the I/O pre-drivers, differential input buffers, and MSEL circuitry

Table 12–1. Arria II GX Power Supply Requirements (Part 2 of 2)

Power Supply Pin	Nominal Voltage Level (V)	Description
VCCIO	1.2 V, 1.5 V, 1.8 V, 2.5 V, 3.0 V, 3.3 V	Supplies power to the I/O banks
VREF	0.6 V, 0.75 V, 0.9 V, 1.25 V	Reference voltage for the voltage-referenced I/O standards
VCCBAT	1.2 V–3.3 V	Battery back-up power supply for the design security volatile key register
VCCA	2.5 V	Supplies power to the transceiver PMA regulator
VCCH_GXB	1.5 V	Supplies power to the transceiver PMA output (TX) buffer
VCCL_GXB	1.1 V	Supplies power to the transceiver PMA TX, PMA RX, and clocking
GND	0 V	Ground

Note to Table 12–1:

(1) `VCCA_PLL` must be powered up even if the PLL is not used.

Power-On Reset Circuitry

The Arria II GX power-on reset (POR) circuitry generates a POR signal to keep the device in reset state until the power supplies voltage levels have stabilized during power-up. The POR circuitry monitors V_{CC}, V_{CCA_PLL}, V_{CCCB}, V_{CCPD}, and V_{CCIO} for I/O banks 3C and 8C, where the configuration pins are located. The POR circuitry tri-states all user I/O pins until the power supplies reach the recommended operating levels. These power supplies are required to monotonically reach their full-rail values without plateaus and within the maximum power supply ramp time, t_{RAMP}. The POR circuitry de-asserts the POR signal after the power supplies reach their full-rail values to release the device from reset state.

POR circuitry is important to ensure that all the circuits in the Arria II GX device are at certain known states during power up. You can select the POR signal pulse width between fast POR time or standard POR time using the MSEL pin settings. For fast POR time, the POR signal pulse width is set to 4 ms for the power supplies to ramp up to full rail. For standard POR time, the POR signal pulse width is set to 100 ms for the power supplies to ramp up to full rail. In both cases, you can extend the POR time with an external component to assert the `nSTATUS` pin low.

For more information about the POR specification, refer to the *Device Data Sheet* chapter in volume 3 of the *Arria II GX Device Handbook*.

For more information about MSEL pin settings, refer to the *Configuration, Design Security, and Remote System Upgrades in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

Hot Socketing

Arria II GX device I/O pins are hot-socketing compliant without the need for external components or special design requirements. Hot-socketing support in Arria II GX devices has the following advantages:

- You can drive the device before power up without damaging it.

- I/O pins remain tri-stated during power up. The device does not drive out before or during power up, thereby not affecting other buses in operation.
- You can insert or remove an Arria II GX device from a powered-up system board without damaging or interfering with normal system and board operation.

Devices Can Be Driven before Power Up

You can drive signals into regular and transceiver Arria II GX I/O pins before or during power up or power down without damaging the device. Arria II GX devices also support power up or power down of the power supplies in any sequence to simplify system-level design.

I/O Pins Remain Tri-Stated During Power Up

A device that does not support hot socketing may interrupt system operation or cause contention by driving out before or during power up. In a hot-socketing situation, the Arria II GX device output buffers are turned off during system power up or power down. Also, the Arria II GX device does not drive out until the device is configured and working within recommended operating conditions.

Insertion or Removal of an Arria II GX Device from a Powered-Up System

Devices that do not support hot socketing can short power supplies when powered up through the device signal pins. This irregular power up can damage both the driving and driven devices and can disrupt card power up.

An Arria II GX device may be inserted into (or removed from) a powered up system board without damaging or interfering with system-board operation.

You can power up or power down the V_{CCIO}, V_{CC}, and V_{CCPD} supplies in any sequence and at any time between them. The power supply ramp rates can range from 50 µs to 100 ms.

For more information about the hot-socketing specification, refer to the *Device Data Sheet* chapter in volume 3 of the *Arria II GX Device Handbook* and the *Hot-Socketing and Power-Sequencing Feature and Testing for Altera Devices* White Paper.

Hot Socketing Feature Implementation

Arria II GX devices are immune to latch-up when hot socketing. The hot-socketing feature turns off the output buffer during power up and power down of the V_{CC}, V_{CCIO}, or V_{CCPD} power supplies. Hot-socketing circuitry generates an internal `HOTSCKT` signal when the V_{CC}, V_{CCIO}, or V_{CCPD} power supplies are below the threshold voltage. Hot-socketing circuitry is designed to prevent excess I/O leakage during power up. When the voltage ramps up very slowly, it is still relatively low, even after the POR signal is released and the configuration is completed. The `CONF_DONE`, `nCEO`, and `nSTATUS` pins fail to respond, as the output buffer cannot flip from the state set by the hot-socketing circuit at this low voltage. Therefore, the hot-socketing circuit is removed on these configuration pins to ensure that they are able to operate during configuration. Thus, it is expected behavior for these pins to drive out during power-up and power-down sequences.

Revision History

Table 12–2 shows the revision history for this document.

Table 12–2. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial release.	—

Additional Information

About this Handbook

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

How to Contact Altera

For the most up-to-date information about Altera products, see the following table.

Contact *(Note 1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Altera literature services	Email	literature@altera.com
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions that this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicates command names and dialog box titles. For example, **Save As** dialog box.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, dialog box options, software utility names, and other GUI labels. For example, **\qdesigns** directory, **d:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicates document titles. For example, *AN 519: Stratix IV Design Guidelines.*
Italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, <*file name*> and <*project name*>.**pof** file.
Initial Capital Letters	Indicates keyboard keys and menu names. For example, Delete key and the Options menu.
"Subheading Title"	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, "Typographic Conventions."

Visual Cue	Meaning
`Courier type`	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. Active-low signals are denoted by suffix `n`. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
1., 2., 3., and a., b., c., and so on.	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■	Bullets indicate a list of items when the sequence of the items is not important.
☞	The hand points to information that requires special attention.
⚠ CAUTION	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
⚠ WARNING	A warning calls attention to a condition or possible situation that can cause you injury.
↵	The angled arrow instructs you to press **Enter**.
👣	The feet direct you to more information about a particular topic.

Arria II GX Device Handbook

Volume 2

101 Innovation Drive
San Jose, CA 95134
www.altera.com

AIIGX5V2-1.0

I.S. EN ISO 9001

Contents

Chapter Revision Dates

The chapters in this book, *Arria II GX Device Handbook Volume 2*, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Section I. Transceiver Architecture

This section provides information about Arria® II GX transceiver architecture and clocking. It also describes configuring multiple protocols, data rates, and reset control and power down in the Arria II GX device family. This section includes the following chapters:

Revision History

Refer to each chapter for its own specific revision history. For information about when each chapter was updated, refer to the Chapter Revision Dates section, which appears in this volume.

1. Arria II GX Transceiver Architecture

AIIGX52001-1.0

Introduction

This chapter describes the Arria® II GX transceiver architecture. Arria II GX FPGAs deliver a breakthrough level of system bandwidth for mainstream applications. Arria II GX devices provide up to 16 full-duplex CDR-based transceivers with physical coding sublayer (PCS) and physical medium attachment (PMA), at serial data rates between 600 Mbps and 3.75 Gbps.

This chapter includes the following sections:

Transceiver channels are designed to support the following serial protocols:

- PCI Express (PIPE)
  - Gen1 (2.5 Gbps)
- Serial RapidIO (1.25 Gbps, 2.5 Gbps, 3.125 Gbps)
- Serial ATA (SATA)/Serial attached SCSI (SAS)
  - SATA I (1.5 Gbps)
  - SATA II (3.0 Gbps)
  - SAS (1.5 Gbps, 3.0 Gbps)
- Serial Digital Interface (SDI)
  - HD-SDI (1.485 Gbps, 1.4835 Gbps)
  - 3G-SDI (2.97 Gbps, 2.967 Gbps)
- ASI (270 Mbps)
- CPRI (614.4 Mbps, 1228.8 Mbps, 2457.6 Mbps, 3072 Mbps)
- OBSAI (768 Mbps, 1536 Mbps, 3072 Mbps)

- Gigabit Ethernet (GIGE) (1.25 Gbps)
- XAUI (3.125 Gbps to 3.75 Gbps for HiGig/HiGig+ support)
- SONET/SDH
  - OC-3 (155 Mbps)
  - OC-12 (622 Mbps)
  - OC-48 (2.488 Gbps)
- GPON (1.244 uplink, 2.488 downlink)
- SerialLite II (0.6 to 3.75 Gbps)

Transceiver channels also support the following highly flexible functional mode to implement proprietary protocols:

- Basic single-width (600 Mbps to 3.75 Gbps)

Transceiver Channel Locations

Arria II GX transceivers are structured into full-duplex (transmitter and receiver) four-channel groups called transceiver blocks. The total number of transceiver channels and the location of transceiver blocks varies among different device members of the Arria II GX family.

Table 1–1 summarizes the total number of transceiver channels and transceiver block locations in each Arria II GX device.

Table 1–1. Number of Transceiver Channels and Transceiver Block Locations in Arria II GX Devices (Part 1 of 2)

Device Member	Total Number of Transceiver Channels	Transceiver Channel Arrangement *(1)*
EP2AGX20CU17 EP2AGX30CU17 EP2AGX45CU17 EP2AGX65CU17 EP2AGX20CF25 EP2AGX30CF25	4	Four transceiver channels located in one transceiver block; GXBL0 on the left side of the device.
EP2AGX45DF25 EP2AGX65DF25 EP2AGX95DF25 EP2AGX125DF25 EP2AGX45DF29 EP2AGX65DF29	8	Eight transceiver channels located in two transceiver blocks; GXBL0 and GXBL1, on the left side of the device, with GXBL0 at the bottom of the device progressing up to GXBL1.

Table 1–1. Number of Transceiver Channels and Transceiver Block Locations in Arria II GX Devices (Part 2 of 2)

Device Member	Total Number of Transceiver Channels	Transceiver Channel Arrangement *(1)*
EP2AGX95EF29 EP2AGX125EF29 EP2AGX190EF29 EP2AGX260EF29 EP2AGX95EF35 EP2AGX125EF35	12	Twelve transceiver channels located in three transceiver blocks; GXBL0, GXBL1, and GXBL2, on the left side of the device, with GXBL0 at the bottom of the device progressing up to GXBL2.
EP2AGX190FF35 EP2SGX260FF35	16	Sixteen transceiver channels located in four transceiver blocks; GXBL0, GXBL1, GXBL2, and GXBL3, on the left side of the device, with GXBL0 at the bottom of the device progressing up to GXBL3.

Note to Table 1–1:

(1) Arrangement of the transceivers is shown in Figure 1–1 through Figure 1–4.

Figure 1–1 shows a die-top view of the four transceiver channel locations in Arria II GX devices.

Figure 1–1. Arria II GX Devices with Four Transceiver Channels

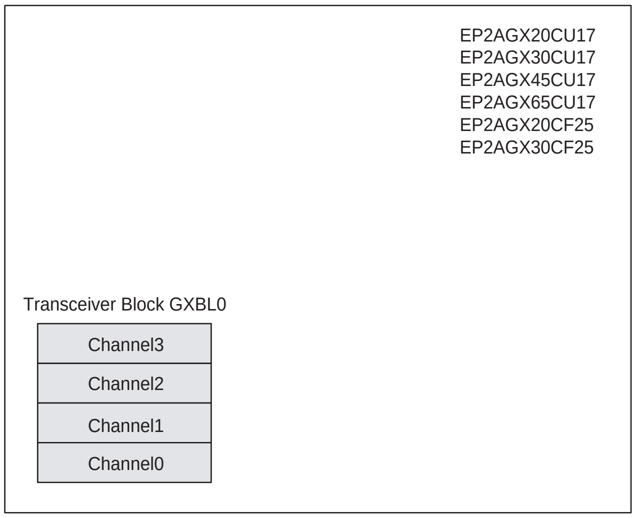


Figure 1–2 shows a die-top view of the eight transceiver channel locations in Arria II GX devices.

Figure 1–2. Arria II GX Devices with Eight Transceiver Channels

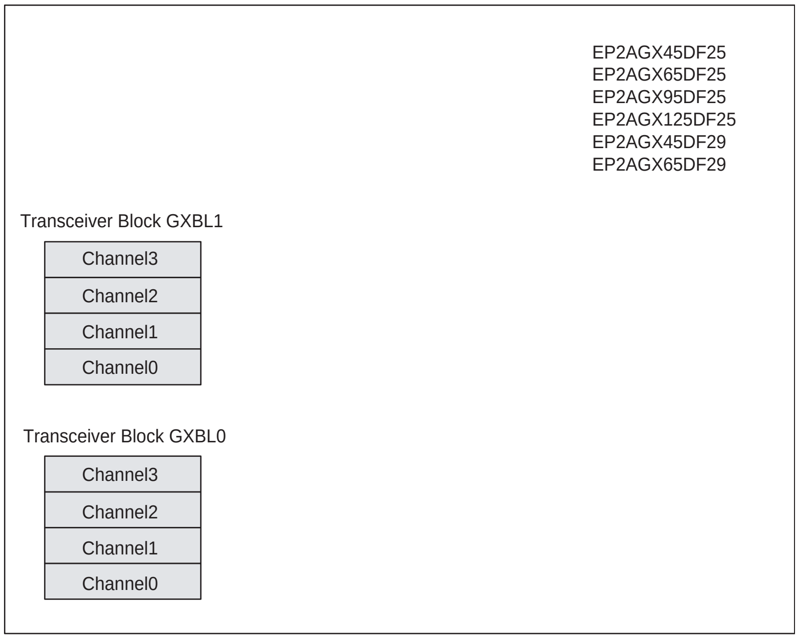


Figure 1–3 shows a die-top view of the twelve transceiver channel locations in Arria II GX devices.

Figure 1–3. Arria II GX Devices with Twelve Transceiver Channels

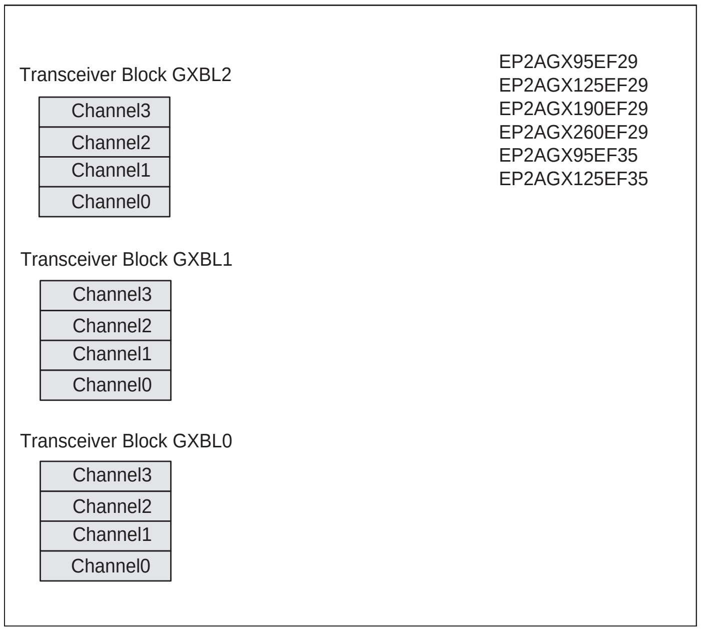
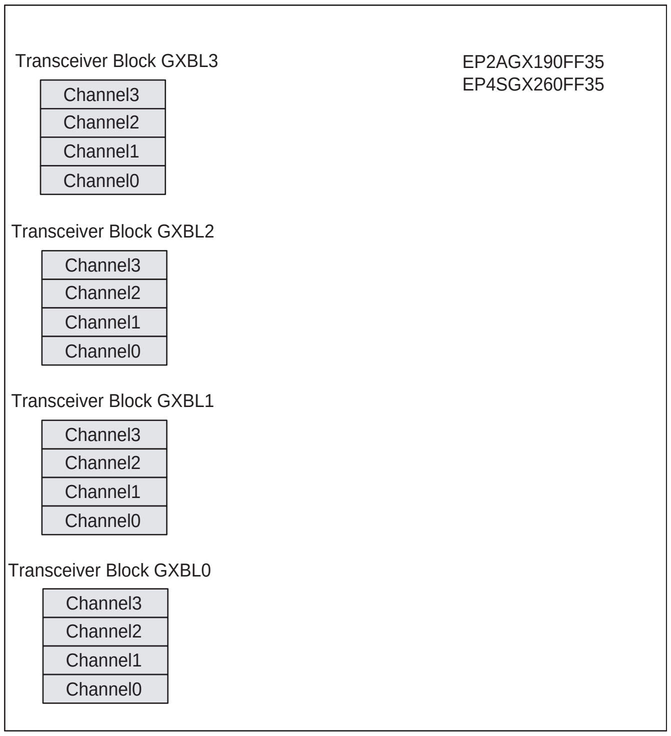


Figure 1–4 shows a die-top view of the sixteen transceiver channel locations in Arria II GX devices.

Figure 1–4. Arria II GX Device with Sixteen Transceiver Channels



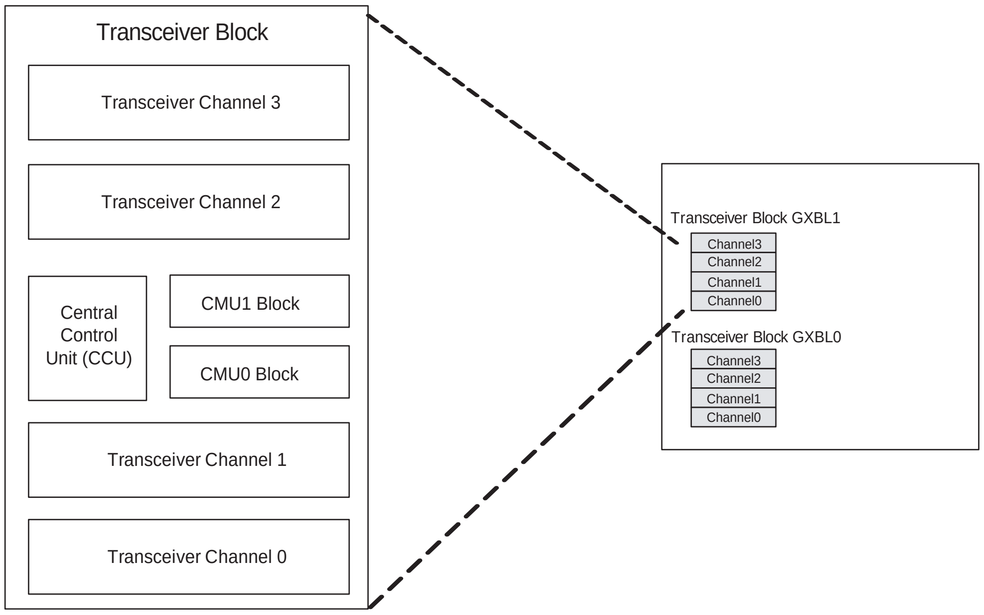
Transceiver Block Architecture Overview

Each transceiver block has:

- Two clock multiplier units (CMU)—CMU0 and CMU1 provide the high-speed serial and low-speed parallel clocks to the transceiver channels
- Four full-duplex (transmitter and receiver) transceiver channels that support serial data rates from 600 Mbps to 3.75 Gbps
- Central control unit (CCU) that implements a XAUI state machine for XGMII-to-PCS code group conversion, XAUI deskew state machine, shared control signal generation block, and reset control logic
  - The shared control signal generation block provides control signals to the transceiver channels in bonded functional modes such as XAUI, PCI Express (PIPE), and Basic ×4

Figure 1–5 shows a block diagram of transceiver block architecture.

Figure 1–5. Top-Level View of a Transceiver Block



For architecture details of CMU blocks and transceiver channels, refer to "CMU Blocks" on page 1–21 and "Transceiver Channel Architecture" on page 1–27.

Dynamic Reconfiguration

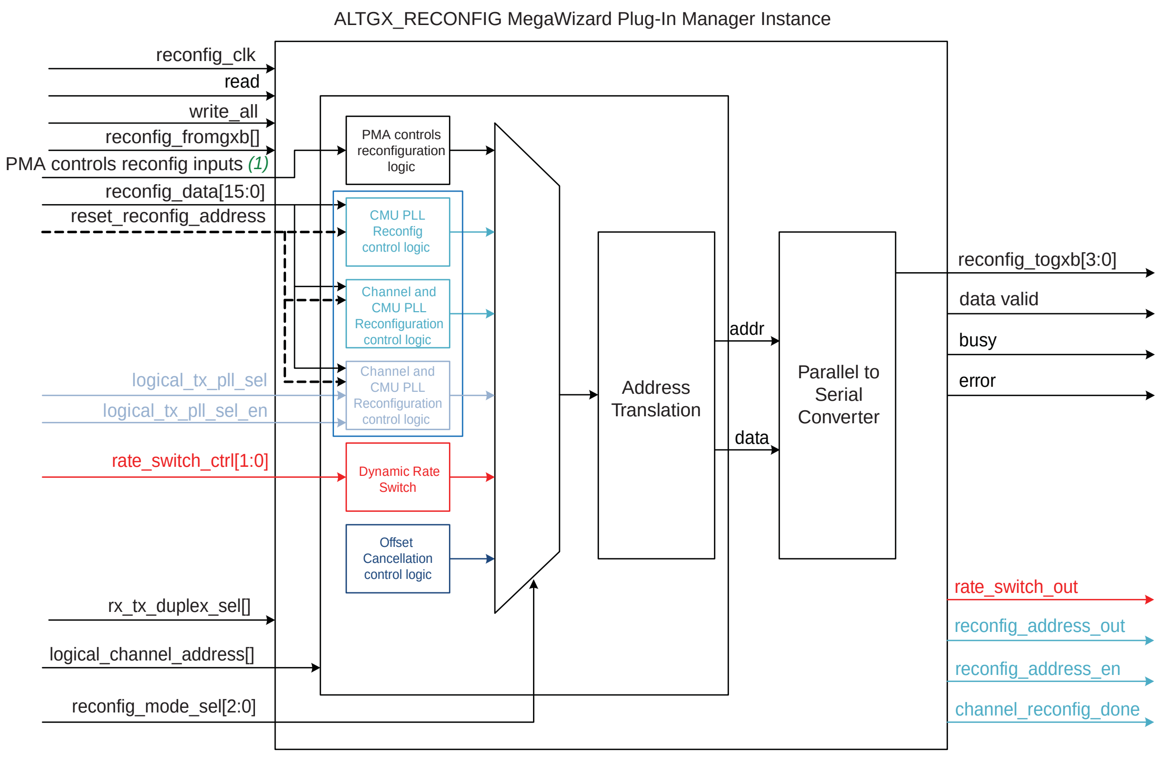
Dynamic reconfiguration is a unique feature that enables your end system's high-speed I/O frequency to be reconfigured in your system without reconfiguring the FPGA. For hot pluggable or open standard systems, this feature allows you to support multiple data rates or standards without reconfiguring the system. For all systems, it allows you to make changes to the bit error rate to compensate in-system for the effects of process and temperature. Each transceiver channel has multiple physical medium attachment controls that you can program to achieve the desired bit error ratio (BER) for your system. When you enable the dynamic reconfiguration feature, you can reconfigure dynamically without powering down other transceiver channels or the FPGA fabric of the device:

- Transmit and receive analog settings
- Transmit data rate in multiples of 1, 2, and 4
- One channel at a time
- Channel and clock multiplier unit PLL
- CMU PLL only

Dynamic Reconfiguration Controller Architecture

The dynamic reconfiguration controller is a soft IP which uses FPGA-fabric resources. You can use only one controller per transceiver block. You cannot use the dynamic reconfiguration controller to control multiple Arria II GX devices or any off-chip interfaces. Figure 1–6 shows the conceptual view of the dynamic reconfiguration controller architecture.

Figure 1–6. Block Diagram of the Dynamic Reconfiguration Controller



Note to Figure 1–6:

(1) The PMA control ports consist of the V_{OD} controls, pre-emphasis controls, DC gain controls, and manual equalization controls.

The dynamic reconfiguration controller is comprised of the following control logic modules:

- PMA controls reconfiguration control logic
- Data rate division control logic to the TX local divider
- Offset cancellation control logic for receiver channels
- Channel reconfiguration with TX PLL select/reconfig control logic
- CMU PLL reconfiguration control logic
- Channel and CMU PLL reconfiguration control logic
- Channel reconfiguration with TX PLL select control logic

The dynamic reconfiguration controller reads from its control inputs or from a Memory Initialization File (**.mif**) file. For PMA controls reconfiguration and data rate division using control logic, the input to the controller is translated to address and data bus within. The address and data bus are then converted into serial data and forwarded to the selected transceiver channel. When a **.mif** file is used for reconfiguration, the dynamic reconfiguration controller receives 16-bit words from the **.mif** that you generate and sends this information to the transceiver channel selected.

For more information, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II GX Devices*. This application note will be available in March, 2009. To review this document when it is available, go to the Literature page of the Altera website (www.altera.com).

Dynamic Reconfiguration Modes

The different modes of dynamic reconfiguration are:

- PMA controls reconfiguration
- Receiver offset cancellation
- Transceiver channel reconfiguration

PMA Controls Reconfiguration

You can dynamically reconfigure the following PMA controls:

- Pre-emphasis settings
- Equalization settings
- DC gain settings
- Voltage output differential (V_{OD}) settings

Offset Cancellation

Variations in process create offsets in analog circuit voltages, pushing them outside the expected range. The Arria II GX device provides an offset cancellation circuit per receiver channel to counter the offset variations due to process. Calibration of the offset cancellation circuit is done at power-up. The receiver buffer and receiver clock data recovery (CDR) require offset calibration.

Offset cancellation is automatically executed once every time the device is powered on. The control logic for offset cancellation is integrated into the dynamic reconfiguration controller.

The offset cancellation for receiver channels option is automatically enabled in both the ALTGX and ALTGX_RECONFIG MegaWizard™ Plug-In Managers for **Receiver and Transmitter** and **Receiver only** configurations. It is not available for **Transmitter only** configurations. For **Receiver and Transmitter** and **Receiver only** configurations, you must connect the ALTGX_RECONFIG instance to the ALTGX instances with receiver channels in your design. You must connect the `reconfig_fromgxb`, `reconfig_togxb`, and necessary clock signals to both the ALTGX_RECONFIG and ALTGX (with receiver channels) instances.

For proper device operation, you must always connect the ALTGX_RECONFIG and ALTGX (with receiver channels) instances.

Transceiver Channel Reconfiguration

For transceiver channels, dynamic reconfiguration involves the reconfiguration of the following:

- Data Rate Reconfiguration. This can be achieved by:
  - Switching between two TX PLLs set to different data rates and reconfiguring the RX PLLS
  - Reconfiguring the local dividers in the transmit and receive sides
- Functional Mode Reconfiguration

For more information, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II GX Devices*. This application note will be available in March, 2009. To review this document when it is available, go to the Literature page of the Altera website (www.altera.com).

Transceiver Port List

You instantiate the Arria II GX transceivers using the ALTGX megafunction instance in the Quartus® II MegaWizard Plug-In Manager. The ALTGX megafunction instance allows you to configure the transceivers for your intended protocol and select optional control and status ports to and from the instantiated transceiver channels.

Table 1–2 provides a description of all the ALTGX megafunction ports.

Table 1–2. Arria II GX ALTGX Megafunction Ports (Part 1 of 13)

Port Name	Input/Output	Description	Scope
Clock Multiplier Unit (CMU)			
`pll_inclk`	Input	Input reference clock for the CMU phase-locked loop (PLL).	Transceiver block
`pll_locked`	Output	CMU PLL lock indicator. A high level indicates that the CMU PLL is locked to the input reference clock; a low level indicates that the CMU PLL is not locked to the input reference clock. Asynchronous signal.	Transceiver block
`pll_powerdown`	Input	CMU PLL power down. When asserted high, the CMU PLL is powered down. When de-asserted low, the CMU PLL is active and locks to the input reference clock. Note: Assertion of the `pll_powerdown` signal does not power down the `refclk` buffers. Asynchronous signal. The minimum pulse-width is 1 µs (pending characterization).	Transceiver block

Table 1–2. Arria II GX ALTGX Megafunction Ports (Part 2 of 13)

Port Name	Input/Output	Description	Scope
`coreclkout`	Output	FPGA fabric-transceiver interface clock. Generated by the CMU0 clock divider in the transceiver block in ×4 bonded channel configurations. Generated by the CMU0 clock divider in the master transceiver block in ×8 bonded channel configurations. Not available in non-bonded channel configurations. This clock is used to clock the write port of the transmitter phase compensation FIFOs in all bonded channels. Use this clock signal to clock parallel data `tx_datain` from the FPGA fabric into the transmitter phase compensation FIFO of all bonded channels. This clock is used to clock the read port of the receiver phase compensation FIFOs in all bonded channels with rate match FIFO enabled. Use this signal to clock parallel data `rx_dataout` from the receiver phase compensation FIFOs of all bonded channels (with rate match FIFO enabled) into the FPGA fabric.	Transceiver block
Receiver Physical Coding Sublayer (PCS) Ports			
Word Aligner			
`rx_enapatternalign`	Input	Manual word alignment enable control. Enables word aligner configured in manual alignment mode to align to the word alignment pattern. In single-width modes (except SONET/SDH OC12 and OC48), this signal is level-sensitive. When high, the word aligner re-aligns if the word alignment pattern appears in a new word boundary. Asynchronous signal. The minimum pulse-width is 2 recovered clock cycles.	Channel
`rx_patterndetect`	Output	Word alignment pattern detect indicator. A high level indicates that the word alignment pattern is found on the current word boundary. The width of this signal depends on the channel width shown below: Channel Width `rx_patterndetect` 8/10 1 16/20 2	Channel

Table 1–2. Arria II GX ALTGX Megafunction Ports (Part 3 of 13)

Port Name	Input/Output	Description	Scope
`rx_syncstatus`	Output	Word alignment synchronization status indicator. For word aligner in automatic synchronization state machine mode. This signal is driven high if the conditions required to remain in synchronization are met. It is driven low if the conditions required to lose synchronization are met. For word aligner in manual alignment mode, the behavior of this signal depends on whether the transceiver is configured in single-width mode. For more information, refer to "Word Aligner in Single-Width Mode" on page 1–59. This signal is not available for word aligner in bit-slip mode. The width of this signal depends on the channel width shown below: Channel Width `rx_syncstatus` 8/10 1 16/20 2	Channel
`rx_bitslip`	Input	Bit-slip control for word aligner configured in bit-slip mode. At every rising edge of this signal, word aligner slips one bit into the received data stream, effectively shifting the word boundary by 1 bit. Asynchronous signal. The minimum pulse-width is 2 recovered clock cycles.	Channel
`rx_ bitslipboundary selectout`	Output	Indicates number of bits slipped in the word aligner when word aligner is configured in manual mode. Asynchronous signal.	Channel
`rx_ala2size`	Input	Available only in SONET OC-12 and OC-48 modes to select between one of the following two word alignment options: ■ 0 - 16-bit A1A2 ■ 1 - 32-bit A1A1A2A2	Channel
`rx_rlv`	Output	Run-length violation indicator. A high pulse is driven when the number of consecutive 1s or 0s in the received data stream exceeds the programmed run length violation threshold. Asynchronous signal. Driven for a minimum of 2 recovered clock cycles in configurations without byte serializer and a minimum of 3 recovered clock cycles in configurations with byte serializer.	Channel

Table 1–2. Arria II GX ALTGX Megafunction Ports (Part 4 of 13)

Port Name	Input/Output	Description	Scope
`rx_invpolarity`	Input	Generic receiver polarity inversion control. Useful feature for correcting situations where the positive and negative signals of the differential serial link are accidentally swapped during board layout. When asserted high the polarity of every bit of the 8-bit or 10-bit input data word to the word aligner gets inverted. Asynchronous signal.	Channel
`rx_revbitorderwa`	Input	Receiver bit reversal control. Available only in Basic single-width mode with word aligner configured in bit-slip mode. Useful feature where the link transmission order is MSB to LSB. When asserted high in Basic single-width mode, the 8-bit or 10-bit data `D[7:0]` or `D[9:0]` at the output of the word aligner gets rewired to `D[0:7]` or `D[0:9]`, respectively. Asynchronous signal.	Channel
Deskew FIFO			
`rx_channelaligned`	Output	10-gigabit attachment unit interface (XAUI) deskew FIFO channel aligned indicator. Available only in XAUI mode. A high level indicates that the XAUI deskew state machine is either in ALIGN_ACQUIRED_1, ALIGN_ACQUIRED_2, ALIGN_ACQUIRED_3, or ALIGN_ACQUIRED_4 state, as specified in the PCS deskew state diagram in IEEE P802.3ae specification. A low level indicates that the XAUI deskew state machine is either in LOSS_OF_ALIGNMENT, ALIGN_DETECT_1, ALIGN_DETECT_2, or ALIGN_DETECT_3 state, as specified in the PCS deskew state diagram in IEEE P802.3ae specification.	Transceiver block
Rate Match (Clock Rate Compensation) FIFO			
`rx_rmfifodata inserted`	Output	Rate match FIFO insertion status indicator. A high level indicates that the rate match pattern byte was inserted to compensate for the parts-per-million (PPM) difference in reference clock frequencies between the upstream transmitter and the local receiver.	Channel
`rx_rmfifodata deleted`	Output	Rate match FIFO deletion status indicator. A high level indicates that the rate match pattern byte was deleted to compensate for the PPM difference in reference clock frequencies between the upstream transmitter and the local receiver.	Channel

Table 1–2. Arria II GX ALTGX Megafunction Ports (Part 5 of 13)

Port Name	Input/Output	Description	Scope
`rx_rmfifofull`	Output	Rate match FIFO full status indicator. A high level indicates that the rate match FIFO is full. Asynchronous signal. Driven for a minimum of 2 recovered clock cycles in configurations without byte serializer and a minimum of 3 recovered clock cycles in configurations with byte serializer.	Channel
`rx_rmfifoempty`	Output	Rate match FIFO empty status indicator. A high level indicates that the rate match FIFO is empty. Asynchronous signal. Driven for a minimum of 2 recovered clock cycles in configurations without byte serializer and a minimum of 3 recovered clock cycles in configurations with byte serializer.	Channel
8B/10B Decoder			
`rx_ctrldetect`	Output	Receiver control code indicator. Available in configurations with 8B/10B decoder. A high level indicates that the associated received code group is a control (/Kx.y/) code group. A low level indicates that the associated received code group is a data (/Dx.y/) code group. The width of this signal depends on the channel width shown below: Channel Width `rx_ctrldetect` 8 1 16 2	Channel

Table 1–2. Arria II GX ALTGX Megafunction Ports (Part 6 of 13)

Port Name	Input/Output	Description	Scope
`rx_errdetect`	Output	8B/10B code group violation or disparity error indicator. Available in configurations with 8B/10B decoder. A high level indicates that a code group violation or disparity error was detected on the associated received code group. Use with the `rx_disperr` signal to differentiate between a code group violation and/or a disparity error as follows: `[rx_errdetect: rx_disperr]` 2'b00 - no error 2'b10 - code group violation 2'b11 - disparity error or both The width of this signal depends on the channel width shown below: Channel Width `rx_errdetect` 8 1 16 2	Channel
`rx_disperr`	Output	8B/10B disparity error indicator port. Available in configurations with 8B/10B decoder. A high level indicates that a disparity error was detected on the associated received code group. The width of this signal depends on the channel width shown below: Channel Width `rx_disperr` 8 1 16 2	Channel
`rx_runningdisp`	Output	8B/10B running disparity indicator. Available in configurations with the 8B/10B decoder. A high level indicates that data on the `rx_dataout` port was received with a negative running disparity. A low level indicates that data on the `rx_dataout` port was received with a positive running disparity. The width of this signal depends on the channel width shown below: Channel Width `rx_runningdisp` 8 1 16 2	Channel

Table 1–2. Arria II GX ALTGX Megafunction Ports (Part 7 of 13)

Port Name	Input/Output	Description	Scope
Byte Ordering Block			
`rx_enabyteord`	Input	Enable byte ordering control. Available in configurations with byte ordering block enabled. The byte ordering block is rising-edge sensitive to this signal. A low-to-high transition triggers the byte ordering block to restart the byte ordering operation. Asynchronous signal.	Channel
`rx_ byteorderalign status`	Output	Byte ordering status indicator. Available in configurations with byte ordering block enabled. A high level indicates that the byte ordering block has detected the programmed byte ordering pattern in the LSByte of the received data from the byte deserializer.	Channel
Receiver Phase Compensation FIFO			
`rx_dataout`	Output	Parallel data output from the receiver to the FPGA fabric. The bus width depends on the channel width multiplied by the number of channels per instance.	
`rx_clkout`	Output	Recovered clock from the receiver channel. Available only when the rate match FIFO is not used in the receiver datapath.	Channel
`rx_coreclk`	Input	Optional read clock port for the receiver phase compensation FIFO. If not selected, the Quartus II software automatically selects `rx_clkout/tx_clkout/coreclkout` as the read clock for the receiver phase compensation FIFO. If selected, you must drive this port with a clock that has 0 PPM difference with respect to `rx_clkout/tx_clkout/ coreclkout`.	Channel
`rx_phase_comp_fifo_ error`	Output	Receiver phase compensation FIFO full or empty indicator. A high level indicates that the receiver phase compensation FIFO is either full or empty.	Channel
Receiver Physical Media Attachment (PMA)			
`rx_datain`	Input	Receiver serial data input port.	Channel
`rx_cruclk`	Input	Input reference clock for the receiver clock and data recovery (CDR).	Channel
`rx_pll_locked`	Output	Receiver CDR lock-to-reference (LTR) indicator. A high level indicates that the receiver CDR is locked to the input reference clock. A low level indicates that the receiver CDR is not locked to the input reference clock. Asynchronous signal.	Channel

Table 1–2. Arria II GX ALTGX Megafunction Ports (Part 8 of 13)

Port Name	Input/Output	Description	Scope
`rx_freqlocked`	Output	Receiver CDR lock mode indicator. A high level indicates that the receiver CDR is in lock-to-data (LTD) mode. A low level indicates that the receiver CDR is in lock-to-reference mode. Asynchronous signal.	Channel
`rx_locktodata`	Input	Receiver CDR lock-to-data mode control signal. When asserted high, the receiver CDR is forced to lock-to-data mode. When de-asserted low, the receiver CDR lock mode depends on the `rx_locktorefclk` signal level.	Channel

`rx_locktorefclk`	Input	Receiver CDR lock-to-reference mode control signal.
		The `rx_locktorefclk` signal along with `rx_locktodata` signal controls whether the receiver CDR is in lock-to-reference or lock-to-data mode, as follows:
		
		`rx_locktodata/`
		`rx_locktorefclk`
		0/0 - receiver CDR is in automatic mode
		0/1 - receiver CDR is in LTR mode
		1/x - receiver CDR is in LTD mode
		
		Asynchronous Signal.
			Channel

`rx_signaldetect`	Output	Signal threshold detect indicator. Available only in PCI Express (PIPE) mode. A high level indicates that the signal present at the receiver input buffer is above the programmed signal detection threshold value. If the electrical idle inference block is disabled in PCI Express (PIPE) mode, the `rx_signaldetect` signal is inverted and driven on the `pipeelecidle` port. Asynchronous signal.	Channel
`rx_seriallpbken`	Input	Serial loopback control port. 0 - normal data path, no serial loopback 1 - serial loopback	Channel

Table 1–2. Arria II GX ALTGX Megafunction Ports (Part 9 of 13)

Port Name	Input/Output	Description	Scope
Transmitter Physical Coding Sublayer Ports			
Transmitter Phase Compensation FIFO			
`tx_datain`	Input	Parallel data input from the FPGA fabric to the transmitter. The bus width depends on the channel width multiplied by the number of channels per instance.	Channel
`tx_clkout`	Output	FPGA fabric-transceiver interface clock. Each channel has a `tx_clkout` signal in non-bonded channel configurations. Use this clock signal to clock the parallel data `tx_datain` from the FPGA fabric into the transmitter. This signal is not available in bonded channel configurations.	Channel
`tx_coreclk`	Input	Optional write clock port for the transmitter phase compensation FIFO. If not selected, the Quartus II software automatically selects `tx_clkout/coreclkout` as the write clock for the transmitter phase compensation FIFO. If selected, you must drive this port with a clock that is frequency locked to `tx_clkout/coreclkout`.	Channel
`tx_phase_comp_fifo_error`	Output	Transmitter phase compensation FIFO full or empty indicator. A high level indicates that the transmitter phase compensation FIFO is either full or empty.	Channel
8B/10B Encoder			
`tx_ctrlenable`	Input	8B/10B encoder /Kx.y/ or /Dx.y/ control. When asserted high, the 8B/10B encoder encodes the data on the `tx_datain` port as a /Kx.y/ control code group. When de-asserted low, it encodes the data on the `tx_datain` port as a /Dx.y/ data code group. The width of this signal depends on the channel width shown below: Channel Width — `tx_ctrlenable` 8 — 1 16 — 2	Channel
`tx_forcedisp`	Input	8B/10B encoder force disparity control. When asserted high, it forces the 8B/10B encoder to encode the data on the `tx_datain` port with a positive or negative disparity, depending on the `tx_dispval` signal level. When de-asserted low, the 8B/10B encoder encodes the data on the `tx_datain` port according to the 8B/10B running disparity rules. The width of this signal depends on the channel width shown below: Channel Width — `tx_forcedisp` 8 — 1 16 — 2	Channel

Table 1–2. Arria II GX ALTGX Megafunction Ports (Part 10 of 13)

Port Name	Input/Output	Description	Scope
`tx_dispval`	Input	8B/10B encoder force disparity value. A high level on the `tx_dispval` signal when the `tx_forcedisp` signal is asserted high forces the 8B/10B encoder to encode the data on the `tx_datain` port with a negative starting running disparity. A low level on the `tx_dispval` signal when the `tx_forcedisp` signal is asserted high forces the 8B/10B encoder to encode the data on the `tx_datain` port with a positive starting running disparity. The width of this signal depends on the channel width shown below: Channel Width `tx_dispval` 8 1 16 2	Channel
`tx_invpolarity`	Input	Transmitter polarity inversion control. Useful feature for correcting situations where the positive and negative signals of the differential serial link are accidentally swapped during board layout. When asserted high the polarity of every bit of the 8-bit or 10-bit input data to the serializer gets inverted. Asynchronous signal.	Channel
`tx_ bitslipboundary select`	Input	Indicates the number of bits to slip at the transmitter for word alignment at the receiver.	Channel
Transmitter Physical Media Attachment			
`tx_dataout`	Output	Transmitter serial data output port.	Channel
`fixedclk`	Input	125-MHz clock for receiver detect functionality in PCI Express (PIPE) mode.	
PIPE Interface (Available only in PCI Express [PIPE] functional mode)			
`powerdn`	Input	PCI Express (PIPE) power state control. Functionally equivalent to the `PowerDown[1:0]` signal defined in PIPE specification revision 2.00. The width of this signal is 2 bits and is encoded as follows: ■ 2'b00: P0 - Normal Operation ■ 2'b01: P0s - Low Recovery Time Latency, Low Power State ■ 2'b10: P1 - Longer Recovery Time Latency, Lower Power State ■ 2'b11: P2 - Lowest Power State	Channel
`tx_ forcedispcompliance`	Input	Force 8B/10B encoder to encode with a negative running disparity. Functionally equivalent to the `TxCompliance` signal defined in PIPE specification revision 2.00. Must be asserted high only when transmitting the first byte of the PCI Express Compliance Pattern to force the 8B/10B encode with a negative running disparity, as required by the PCI Express (PIPE) protocol.	Channel

Table 1–2. Arria II GX ALTGX Megafunction Ports (Part 11 of 13)

Port Name	Input/Output	Description	Scope
`tx_forceelecidle`	Input	Force transmitter buffer to PCI Express (PIPE) electrical idle signal levels. Functionally equivalent to the `TxElecIdle` signal defined in PIPE specification revision 2.00.	Channel
`pipe8b10binv polarity`	Input	PCI Express (PIPE) polarity inversion control. Functionally equivalent to the `RxPolarity` signal defined in PIPE specification revision 2.00. Available only in PCI Express (PIPE) mode. When asserted high, the polarity of every bit of the 10-bit input data to the 8B/10B decoder gets inverted.	Channel
`tx_detectrxloopback`	Input	Receiver detect or PCI Express (PIPE) loopback control. Functionally equivalent to the `TxDetectRx/Loopback` signal defined in PIPE specification revision 2.00. When asserted high in P1 power state with the `tx_forceelecidle` signal asserted, the transmitter buffer begins the receiver detection operation. Once the receiver detect completion is indicated on the `pipephydonestatus` port, this signal must be de-asserted. When asserted high in P0 power state with the `tx_forceelecidle` signal de-asserted, the transceiver datapath gets dynamically configured to support parallel loopback as described in "PCI Express (PIPE) Reverse Parallel Loopback" on page 1–136.	Channel
`pipestatus`	Output	PCI Express (PIPE) receiver status port. Functionally equivalent to the `RxStatus[2:0]` signal defined in PIPE specification revision 2.00. The width of this signal is 3 bits per channel. The encoding of receiver status on the `pipestatus` port is as follows: ■ 000 - Received data OK ■ 001 - 1 skip added ■ 010 - 1 skip removed ■ 011 - Receiver detected ■ 100 - 8B/10B decoder error ■ 101 - Elastic buffer overflow ■ 110 - Elastic buffer underflow ■ 111 - Received disparity error.	Channel
`pipephydonestatus`	Output	PHY function completion indicator. Functionally equivalent to the `PhyStatus` signal defined in PIPE specification revision 2.00. Asserted high for one parallel clock cycle to communicate completion of several PHY functions, such as power state transition and receiver detection.	Channel

Table 1–2. Arria II GX ALTGX Megafunction Ports (Part 12 of 13)

Port Name	Input/Output	Description	Scope
`rx_pipedatavalid`	Output	Valid data and control on the `rx_dataout` and `rx_ctrldetect` ports indicator. Functionally equivalent to the `RxValid` signal defined in PIPE specification revision 2.00.	Channel
`pipeelecidle`	Output	Electrical idle detected or inferred at the receiver indicator. Functionally equivalent to the `RxElecIdle` signal defined in PIPE specification revision 2.00. If the electrical idle inference block is enabled, it drives this signal high when it infers an electrical idle condition, as described in "Electrical Idle Inference" on page 1–99. Otherwise, it drives this signal low. If the electrical idle inference block is disabled, the `rx_signaldetect` signal from the signal detect circuitry in the receiver buffer is inverted and driven on this port. Asynchronous Signal.	Channel
Reset and Power Down			
`gxb_powerdown`	Input	Transceiver block power down. When asserted high, all digital and analog circuitry in the PCS, PMA, CMU channels, and the CCU of the transceiver block gets powered down. Note that asserting the `gxb_powerdown` signal does not power down the `refclk` buffers. Asynchronous signal. The minimum pulse width is 1 us (pending characterization).	Transceiver block
`rx_digitalreset`	Input	Receiver PCS reset. When asserted high, the receiver PCS blocks get reset. The minimum pulse width is 2 parallel clock cycles. Refer to the *Reset Control and Power Down* chapter in volume 2 of the *Arria II GX Device Handbook* for more details.	Channel
`rx_analogreset`	Input	Receiver PMA reset. When asserted high, analog circuitry in the receiver PMA gets reset. The minimum pulse width is 2 parallel clock cycles. Refer to the *Reset Control and Power Down* chapter in volume 2 of the *Arria II GX Device Handbook* for more details.	Channel
`tx_digitalreset`	Input	Transmitter PCS reset. When asserted high, the transmitter PCS blocks get reset. The minimum pulse width is 2 parallel clock cycles. Refer to the *Reset Control and Power Down* chapter in volume 2 of the *Arria II GX Device Handbook* for more details.	Channel

Table 1–2. Arria II GX ALTGX Megafunction Ports (Part 13 of 13)

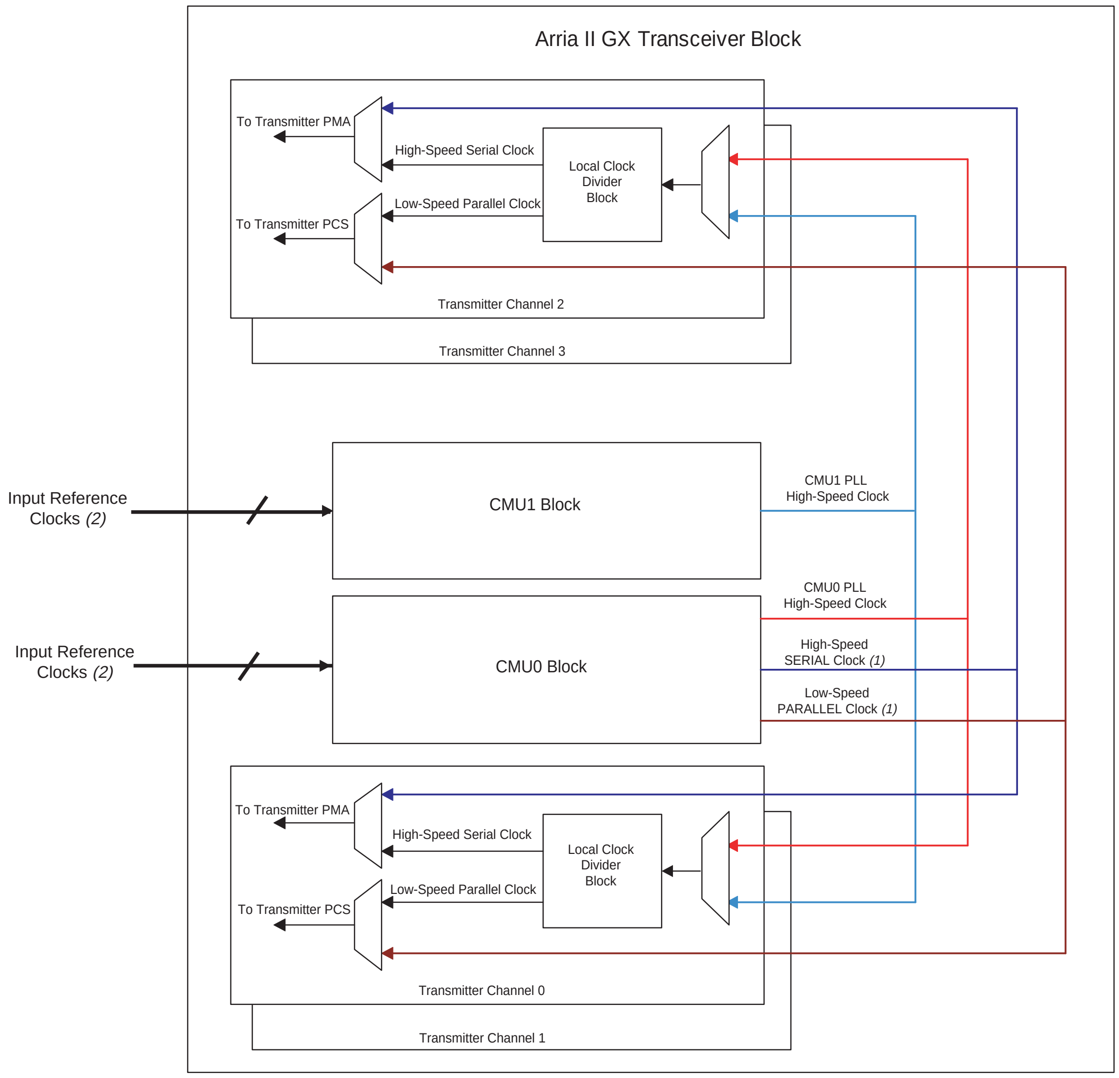
Port Name	Input/Output	Description	Scope
Calibration Block			
`cal_blk_clk`	Input	Clock for transceiver calibration blocks.	Device
`cal_blk_powerdown`	Input	Calibration block power down control.	Device
Reconfiguration Block			
`reconfig_fromgxb`	Input	The width of this signal is determined by the value you set in the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen. For more information, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II GX Devices*. This application note will be available in March, 2009. To review this document when it is available, go to the Literature page of the Altera website (www.altera.com).	Device
`reconfig_togxb[3:0]`	Output	The width of this signal is fixed to four bits. It is independent of the value you set in the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen. For more information, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II GX Devices*. This application note will be available in March, 2009. To review this document when it is available, go to the Literature page of the Altera website (www.altera.com).	Device

CMU Blocks

Each transceiver block contains two CMU blocks—CMU0 and CMU1. Each CMU block contains a CMU PLL that provides clocks to all the transmitter channels in the same transceiver block. The CMU0 block has additional capabilities to support bonded protocol functional modes such as Basic ×4, XAUI, and PCI Express (PIPE). You can select these functional modes from the ALTGX MegaWizard Plug-In Manager. You can enable Basic ×4 functional mode in the ALTGX MegaWizard Plug-In Manager by selecting the ×4 option in Basic mode.

Figure 1–7 shows a top-level block diagram of the CMU channels in a transceiver block.

Figure 1–7. Top-Level Diagram of CMU Channels in a Transceiver Block



Notes to Figure 1–7:

(1) Clocks provided to support bonded channel functional mode.

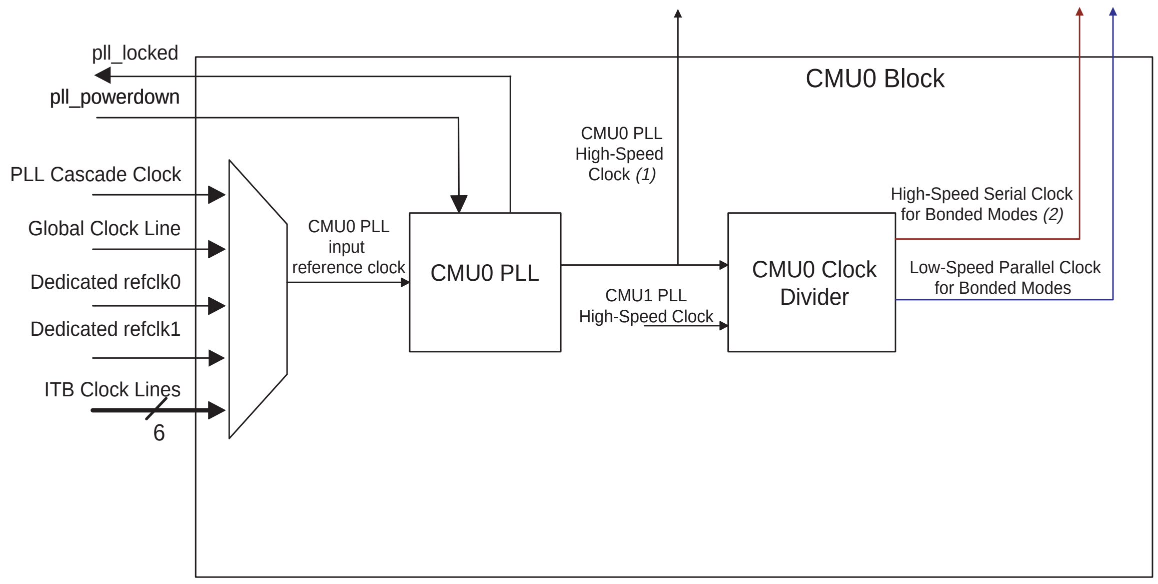
(2) For more information, refer to the *Arria II GX Transceiver Clocking* chapter in volume 2 of the *Arria II GX Device Handbook*.

CMU0 Block

The CMU0 block, shown in Figure 1–8, contains the following blocks:

- CMU0 PLL
- CMU0 clock divider

Figure 1–8. Diagram of CMU0 Block


pll_locked
pll_powerdown
PLL Cascade Clock
Global Clock Line
Dedicated refclk0
Dedicated refclk1
ITB Clock Lines
6
CMU0 Block
CMU0 PLL input reference clock
CMU0 PLL
CMU0 PLL High-Speed Clock (1)
CMU1 PLL High-Speed Clock
CMU0 Clock Divider
High-Speed Serial Clock for Bonded Modes (2)
Low-Speed Parallel Clock for Bonded Modes

Notes to Figure 1–8:

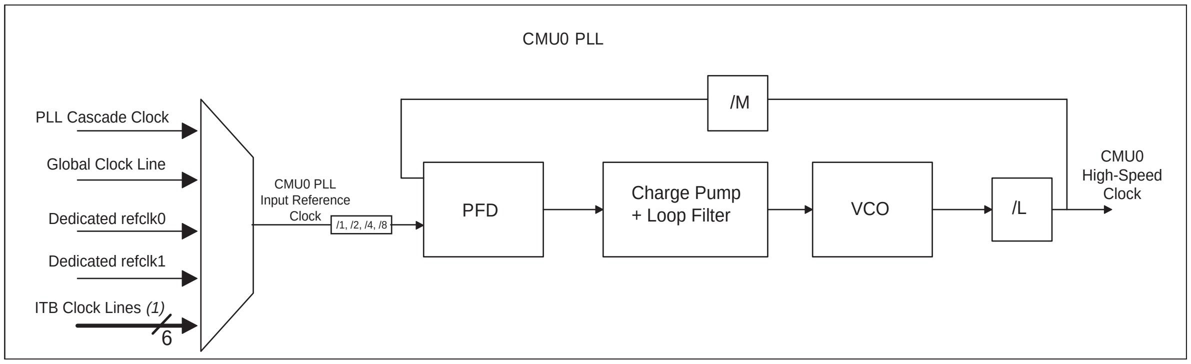
(1) In non-bonded functional modes (for example, GIGE functional mode), the transmitter channel uses the transmitter local clock divider to divide this high-speed clock output to provide clocks for its PMA and PCS blocks.

(2) Used in XAUI, Basic ×4, and PCI Express (PIPE) ×4 functional modes. In PCI Express (PIPE) ×8 functional mode, only the CMU0 channel of the master transceiver block provides clock output to all eight transceiver channels configured in PCI Express (PIPE) functional mode.

CMU0 PLL

Figure 1–9 shows the block diagram of the CMU0 PLL.

Figure 1–9. Diagram of the CMU0 PLL



Note to Figure 1–9:

(1) The ITB clock lines shown are the maximum value. The actual number of ITB lines in your device depends on the number of transceiver blocks on one side of the device.

For more information about input reference clocks, refer to the “CMU PLL and Receiver CDR Input Reference Clocks” section of the *Arria II GX Transceiver Clocking* chapter in volume 2 of the *Arria II GX Device Handbook*.

You can select the input reference clock to the CMU0 PLL from multiple clock sources. The various clock sources are:

- PLL cascade clock—the PLL cascade clock is the output from the general purpose PLLs in the FPGA fabric
- Global clock line—the input reference clock from the dedicated CLK pins are connected to the global clock line
- `refclk0`—dedicated `refclk` in the transceiver block
- `refclk1`—dedicated `refclk` in the transceiver block
- Inter transceiver block (ITB) lines—the ITB lines connect the `refclk0` and `refclk1` of all other transceiver blocks on the same side of the device.

The CMU0 PLL generates the high-speed clock from the input reference clock. The phase frequency detector (PFD) tracks the voltage-controlled oscillator (VCO) output with the input reference clock. The input frequency range of the PFD is 50 to 325 MHz.

For more information about transceiver input reference clocks, refer to the *Arria II GX Transceiver Clocking* chapter in volume 2 of the *Arria II GX Device Handbook.*

The VCO in the CMU0 PLL is half rate and runs at half the serial data rate. The CMU0 PLL uses two multiplier blocks (/M and /L) in the feedback path (shown in Figure 1–9) to generate the high-speed clock needed to support a native data rate range of 600 Mbps to 3.75 Gbps. Table 1–3 lists the available /M and /L settings.

The Quartus II software automatically selects the /M and /L settings based on the input reference clock frequency and serial data rate.

Each CMU PLL (CMU0 PLL and CMU1 PLL) has a dedicated `pll_locked` signal that gets asserted to indicate that the CMU PLL is locked to the input reference clock. You can use the `pll_locked` signal in your transceiver reset sequence as described in the *Reset Control and Power Down* chapter in volume 2 of the *Arria II GX Device Handbook*.

Table 1–3. Multiplier Block Settings in the CMU0 PLL

Multiplier Block	Available Values
/M	1, 4, 5, 8, 10,16, 20, 25
/L	1, 2, 4

PLL Bandwidth Setting

You can program the PLL bandwidth setting using the ALTGX MegaWizard Plug-In Manager. The bandwidth of a PLL is the measure of its ability to track input clock and jitter. It is determined by the –3 dB frequency of the closed-loop gain of the PLL. There are three bandwidth settings: high, medium, and low.

- The high bandwidth setting filters out internal noise from the VCO because it tracks the input clock above the frequency of the internal VCO noise.

- With the low bandwidth setting, if the noise on the input reference clock is greater than the internal noise of the VCO, the PLL filters out the noise above the –3 dB frequency of the closed-loop gain of the PLL.
- The medium bandwidth setting is a compromise between the high and low settings.

The –3 dB frequencies for these settings can vary because of the non-linear nature and frequency dependencies of the circuit.

Power Down CMU0 PLL

You can power down the CMU0 PLL by asserting the `pll_powerdown` signal.

For more information about recommended reset sequences, refer to the *Reset Control and Power Down* chapter in volume 2 of the *Arria II GX Device Handbook*.

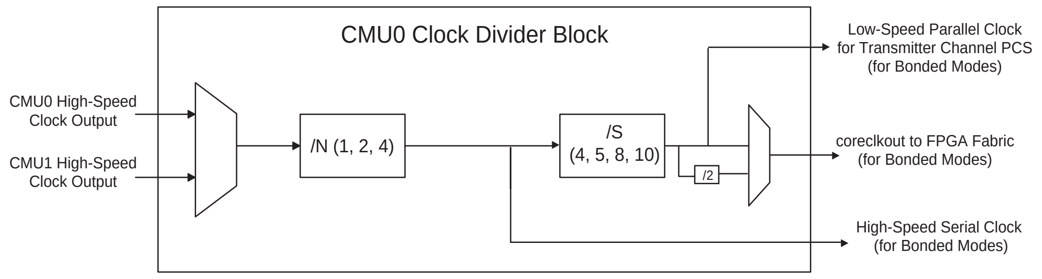
CMU0 Clock Divider Block

The high-speed clock output from the CMU0 PLL is forwarded to two clock divider blocks: the CMU0 clock divider block and the transmitter channel local clock divider block. The CMU0 clock divider block is used only in bonded channel functional modes. In all non-bonded functional modes (example GIGE functional mode), the local clock divider block divides the high-speed clock to provide clocks for its PCS and PMA blocks. This section only discusses the CMU0 clock divider block.

For more information about the local clock divider block, refer to the "Transceiver Channel Datapath Clocking" section in the *Arria II GX Transceiver Clocking* chapter in volume 2 of the *Arria II GX Device Handbook*.

You can configure the CMU0 clock divider block, shown in Figure 1–10, to select the high-speed clock output from the CMU0 PLL or CMU1 PLL.

Figure 1–10. Diagram of the CMU0 Clock Divider Block


CMU0 Clock Divider Block
CMU0 High-Speed Clock Output
CMU1 High-Speed Clock Output
/N (1, 2, 4)
/S (4, 5, 8, 10)
/2
Low-Speed Parallel Clock for Transmitter Channel PCS (for Bonded Modes)
coreclkout to FPGA Fabric (for Bonded Modes)
High-Speed Serial Clock (for Bonded Modes)

High-Speed Serial Clock Generation

The /N divider receives the high-speed clock output from one of the CMU PLLs and produces a high-speed serial clock. This high-speed serial clock is used for bonded functional modes such as Basic ×4, XAUI, and PCI Express (PIPE) ×4 configurations. In XAUI, PCI Express (PIPE) ×4 mode, and Basic ×4 modes, the high-speed serial clock is provided to all the transmitter channels in the transceiver block.

In PCI Express (PIPE) ×8 mode, only the CMU0 clock divider of the master transceiver block provides the high-speed serial clock to all eight channels.

In PCI Express (PIPE) ×1 mode, the CMU0 clock divider does NOT provide high-speed serial clock. Instead, the local clock divider block in the transmitter channel receives the CMU0 PLL or CMU1 PLL high-speed clock output and generates the high-speed serial clock to its serializer.

Low-Speed Parallel Clock Generation

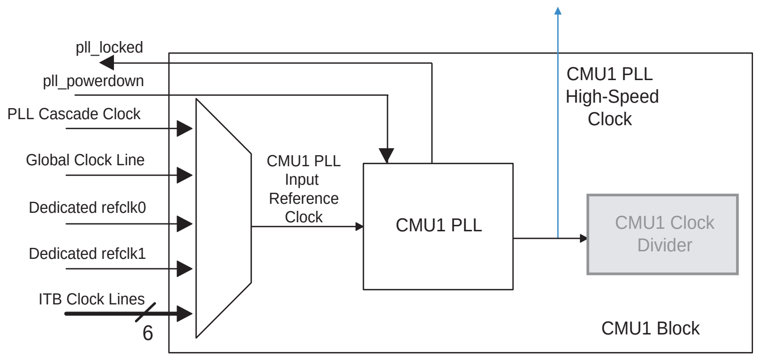
The /S divider receives the clock output from the /N divider and generates the low-speed parallel clock for the PCS block of all transmitter channels and `coreclkout` for the FPGA fabric. If the byte serializer block is enabled in the bonded channel modes, the /S divider output is divided by the /2 divider and sent out as `coreclkout` to the FPGA fabric. The Quartus II software automatically selects the /S values based on the deserialization width setting (single-width mode) that you select in the ALTGX MegaWizard Plug-In Manager.

The Quartus II software automatically selects all the divider settings based on the input clock frequency, data rate, deserialization width, and channel width settings. The deserialization default width setting for Arria II GX devices is single-width mode and cannot be changed.

CMU1 Block

The CMU1 block shown in Figure 1–11 contains the CMU1 PLL that provides the high-speed clock to the transmitter channels in the transceiver block. The CMU1 PLL is similar to the CMU0 PLL. The functionality of the CMU0 PLL is discussed in "CMU0 PLL" on page 1–23.

Figure 1–11. CMU1 Block (Grayed Area Shows the Inactive Block)



The CMU1 PLL generates the high-speed clock that is only used in non-bonded functional modes. In non-bonded functional modes, the transmitter channels in the transceiver block can receive high-speed clock from either of the two CMU PLLs and use local dividers to provide clocks to its PCS and PMA blocks.

For more information about using two CMU PLLs to configure transmitter channels, refer to the *Configuring Multiple Protocols and Data Rates* chapter in volume 2 of *Arria II GX Device Handbook*.

Power Down CMU1 PLL

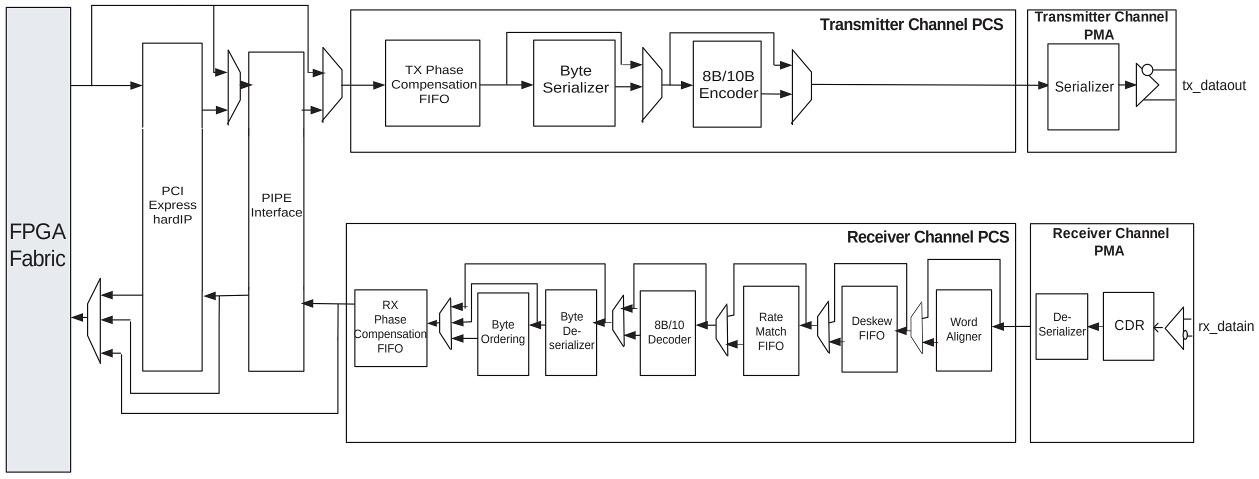
You can power down the CMU1 PLL by asserting the `pll_powerdown` signal.

For more information about using the `pll_powerdown` signal, refer to the *Reset Control and Power Down* chapter in volume 2 of the *Arria II GX Device Handbook*.

Transceiver Channel Architecture

Figure 1–12 shows the Arria II GX transceiver channel datapath.

Figure 1–12. Arria II GX Transceiver Datapath



Each transceiver channel consists of:

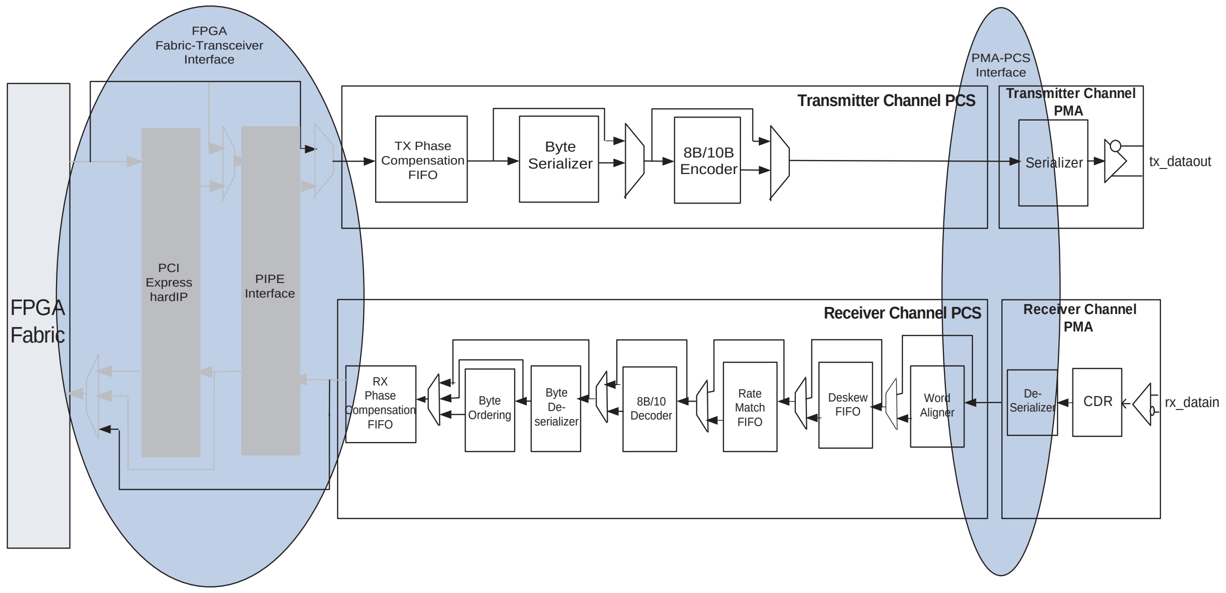
- Transmitter channel, further divided into
  - Transmitter channel PCS
  - Transmitter channel PMA
- Receiver channel, further divided into
  - Receiver channel PCS
  - Receiver channel PMA

Each transceiver channel interfaces to either the PCI Express hard IP block (PCI Express hard IP—transceiver interface) or directly to the FPGA fabric (FPGA fabric—transceiver interface). The transceiver channel interfaces to the PCI Express hard IP block if the hard IP block is used to implement the PCI Express PHY MAC, Data Link Layer, and Transaction Layer. Otherwise, the transceiver channel interfaces directly to the FPGA fabric.

The PCI Express hard IP—transceiver interface is out of the scope of this chapter. This chapter focuses on the FPGA fabric—transceiver interface.

Figure 1–13 shows the FPGA fabric—transceiver interface and the transceiver PMA-PCS interface.

Figure 1–13. FPGA Fabric—Transceiver Interface and the Transceiver PMA-PCS Interface



The transceiver channel datapath is described by the single-width mode based on the FPGA fabric, transceiver interface width (channel width), and transceiver channel PMA-PCS width (serialization factor).

Table 1–4 shows the FPGA fabric—transceiver interface widths (channel width) and transceiver PMA-PCS widths (serialization factor) allowed in single-width mode.

Table 1–4. FPGA Fabric—Transceiver Interface Width and Transceiver PMA-PCS Widths *(Note 1)*

Name	Single-Width
PMA-PCS interface widths	8/10 bit
FPGA fabric—transceiver interface width	8/10 bit 16/20 bit
Supported functional modes	■ PCI Express (PIPE) Gen1 ■ XAUI ■ GIGE ■ Serial RapidIO ■ SONET/SDH OC12 and OC48 ■ SDI ■ Basic single width
Data rate range in Basic functional mode	0.6 Gbps - 3.75 Gbps

Note to Table 1–4:

(1) In low-latency PCS mode in Basic single-width configuration, the data rate range is 0.6 Gbps to 3.75 Gbps.

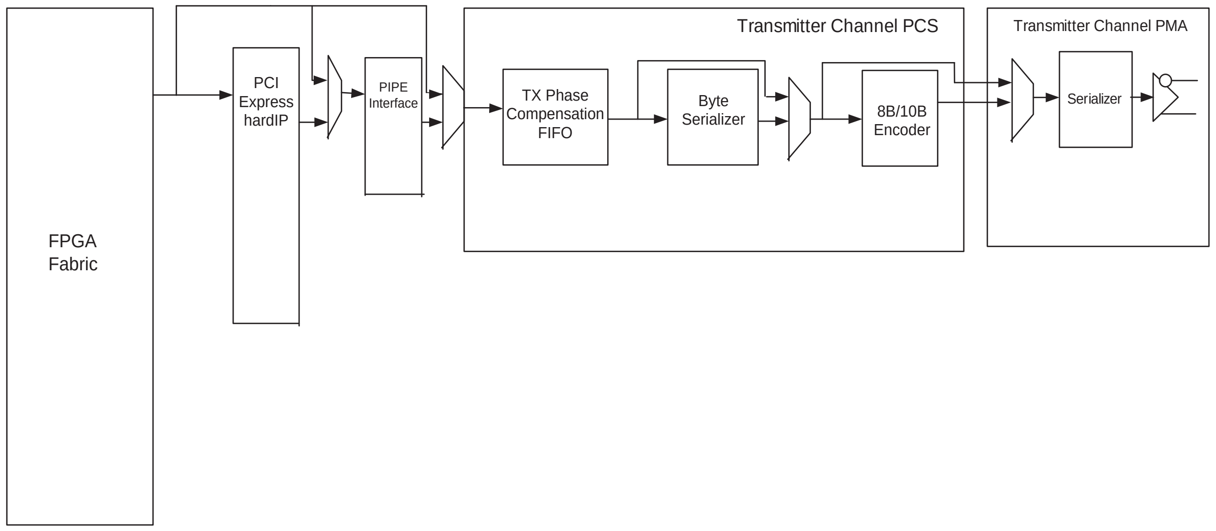
Transmitter Channel Datapath

The transmitter channel datapath, shown in Figure 1–14, consists of the following blocks:

- TX phase compensation FIFO
- Byte serializer
- 8B/10B encoder
- Transmitter output buffer

The Arria II GX transceiver provides the **enable low latency PCS mode** option in the ALTGX MegaWizard Plug-In Manager. If you select this option, the 8B/10B encoder in the data path is disabled.

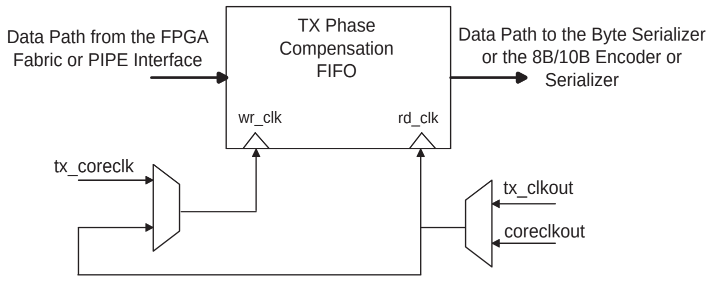
Figure 1–14. Transmitter Channel Datapath



TX Phase Compensation FIFO

The TX phase compensation FIFO interfaces the transmitter channel PCS and the FPGA fabric PIPE interface. It compensates for the phase difference between the low-speed parallel clock and the FPGA fabric interface clock. The two modes in which the TX phase compensation FIFO operates are low-latency and high-latency mode. Figure 1–15 shows the datapath and clocking of the TX phase compensation FIFO.

Figure 1–15. TX Phase Compensation FIFO


Data Path from the FPGA Fabric or PIPE Interface
TX Phase Compensation FIFO
Data Path to the Byte Serializer or the 8B/10B Encoder or Serializer
wr_clk
rd_clk
tx_coreclk
tx_clkout
coreclkout

TX phase compensation FIFO:

- In low-latency mode, the FIFO is four words deep. The latency through the FIFO is two to three FPGA fabric parallel clock cycles (pending characterization). Low-latency mode is chosen automatically in every mode except PCI Express (PIPE) mode.
- In high-latency mode, the FIFO is eight words deep. The latency through the FIFO is approximately four to five FPGA parallel cycles (pending characterization). High-latency mode is automatically used for PCI Express (PIPE) mode.

In non-bonded functional modes such as GIGE, the read port of the phase compensation FIFO is clocked by the low-speed parallel clock. The write clock is fed by the `tx_clkout` port of the associated channel. In bonded functional modes such as XAUI, the write clock of the FIFO is clocked by `coreclkout` provided by the CMU0 clock divider block. You can clock the write side by using `tx_coreclk` provided from the FPGA fabric by enabling the `tx_coreclk` port in the ALTGX MegaWizard Plug-In Manager. If you use this port, ensure that there is 0 PPM difference in frequency between the write and read side. The Quartus II software requires that you provide a 0 PPM assignment in the assignment editor.

For more information about the TX phase compensation FIFO, refer to the "Limitation of the Quartus II Software Selected Transmitter Phase Compensation FIFO Clocks" section in the *Arria II GX Transceiver Clocking* chapter in volume 2 of the *Arria II GX Device Handbook*.

Input Data

In PCI Express (PIPE) functional mode, the input data comes from the PIPE interface. In all other functional modes, the input data comes directly from the FPGA fabric.

Output Data Destination Block

The output from the TX phase compensation FIFO is used by the byte serializer block, 8B/10B encoder, or serializer block. Table 1–5 lists the conditions under which the TX phase compensation FIFO outputs are provided to these blocks.

Table 1–5. Output Data Destination Block for TX Phase Compensation FIFO Output Data

Byte Serializer	8B/10B Encoder	Serializer
If you select: single-width mode and channel width = 16 or 20	If you select: single-width mode and channel width = 8 and 8B/10B encoder enabled	If you select: low latency PCS bypass mode enabled or single-width mode and channel width = 8 or 10

TX Phase Compensation FIFO Status Signal

An optional `tx_phase_comp_fifo_error` port is available in all functional modes to indicate a receiver phase compensation FIFO overflow or under run condition. The `tx_phase_comp_fifo_error` signal is asserted high when the TX phase compensation FIFO either overflows or under runs due to any frequency PPM difference between the FIFO read and write clocks. If the `tx_phase_comp_fifo_error` flag gets asserted, verify the FPGA fabric—transceiver interface clocking to ensure that there is 0 PPM difference between the TX phase compensation FIFO read and write clocks.

Byte Serializer

The byte serializer divides the input datapath by two. This allows you to run the transceiver channel at higher data rates while keeping the FPGA fabric interface frequency within the maximum limit of 200 MHz. In single-width mode, it converts the two-byte wide datapath to a one-byte wide datapath.

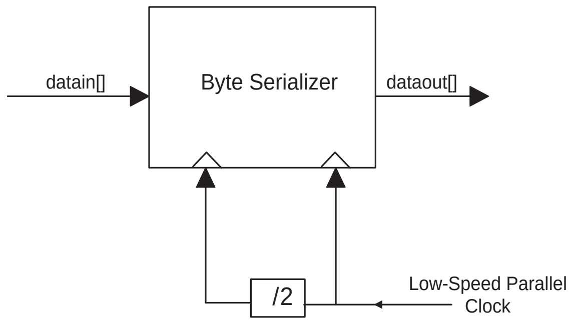
For example, if you would like to run the transceiver channel at 3.125 Gbps, without the byte serializer, in single-width mode, the FPGA fabric interface clock frequency must be 312.5 MHz (3.125 G/10). This violates the FPGA fabric interface frequency limit. When you use the byte serializer, the FPGA fabric interface frequency is 156.25 MHz (3.125 G/20).

☞ The byte deserializer is required in configurations that exceed the FPGA fabric—transceiver interface clock upper frequency limit. It is optional in configurations that does not exceed the FPGA fabric—transceiver interface clock upper frequency limit.

Single-Width Mode

Figure 1–16 shows the byte serializer datapath in single-width mode.

Figure 1–16. Byte Serializer Datapath in Single-Width Mode *(Note 1)*, *(2)*



Notes to Figure 1–16:

(1) Refer to Table 1–6 for the `datain[]` and `dataout[]` port width.

(2) The `datain` signal is the input from the FPGA fabric that has already passed through the TX phase compensation FIFO.

The byte serializer forwards the LSB first, followed by the MSB. The input data width to the byte serializer depends on the channel width option that you selected in the ALTGX MegaWizard Plug-In Manager. For example, in single-width mode, assuming a channel width of 20, the byte serializer sends out the least significant word `datain[9:0]` of the parallel data from the FPGA fabric, followed by `datain[19:10]`. Table 1–6 shows the input and output data widths of the byte serializer in single-width mode.

Table 1–6. Input and Output Data Width of the Byte Serializer in Single-Width Mode

Deserialization Width	Input Data Width to the Byte Serializer	Output Data Width from the Byte Serializer
Single-width mode	16	8
	20 *(1)*	10 *(1)*

Note to Table 1–6:

(1) The 8B/10B encoder is automatically bypassed by the Quartus II software for channel width of 10 or 20 bits.

Asserting the `tx_digitalreset` signal resets the byte serializer block.

For channel widths of 8 or 16 bits, if you select the **8B/10B Encoder** option in the ALTGX MegaWizard Plug-In Manager, the 8B/10B encoder uses the output from the byte serializer. Otherwise, the byte serializer output is forwarded to the serializer.

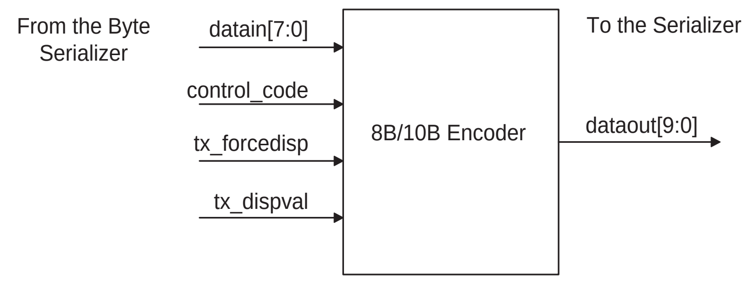
8B/10B Encoder

The 8B/10B encoder generates 10-bit code groups from the 8-bit data and 1-bit control identifier. In single-width mode, the 8B/10B encoder generates a 10-bit code group from the 8-bit data and 1-bit control identifier.

Single-Width Mode

Figure 1–17 shows the inputs and outputs of the 8B/10B encoder.

Figure 1–17. 8B/10B Encoder in Single-Width Mode

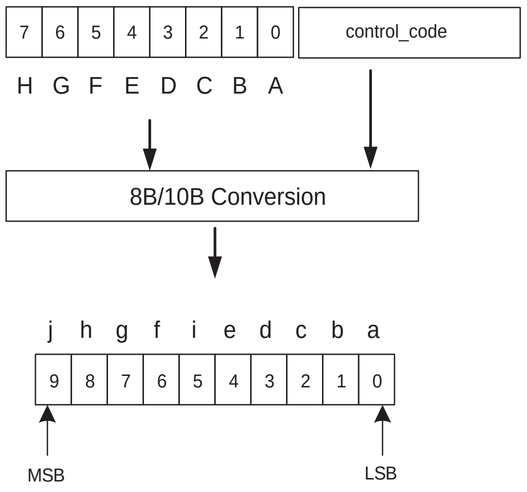


In single-width mode, the 8B/10B encoder translates the 8-bit data to a 10-bit code group (control word or data word) with proper disparity. If the `control_code` input is high, the 8B/10B encoder translates the input `data[7:0]` to a 10-bit control word. If the `control_code` input is low, the 8B/10B encoder translates the input `data[7:0]` to a 10-bit data word.

You can use the `tx_forcedisp` and `tx_dispval` ports to control the running disparity of the generated output data. For more information, see "Controlling Running Disparity" on page 1–34.

Figure 1–18 shows the conversion format. The LSB is transmitted first.

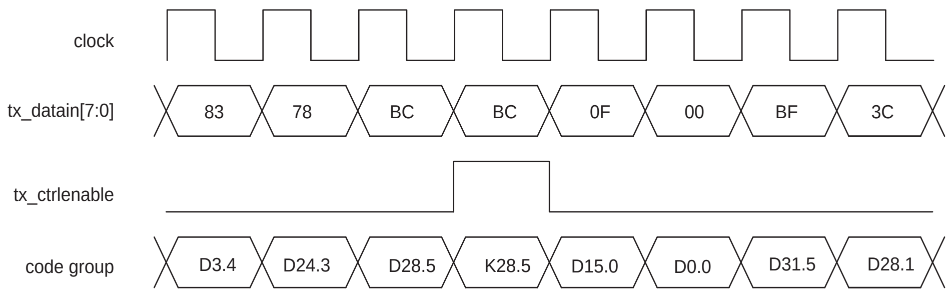
Figure 1–18. 8B/10B Conversion Format



Control Code Encoding

The ALTGX MegaWizard Plug-In Manager provides the `tx_ctrlenable` port to indicate whether the 8-bit data at the `tx_datain` port should be encoded as a control word (Kx.y). When `tx_ctrlenable` is low, the 8B/10B encoder block encodes the byte at the `tx_datain` port (the user-input port) as data (Dx.y). When `tx_ctrlenable` is high, the 8B/10B encoder encodes the input data as a Kx.y code group. The waveform in Figure 1–19 shows the second 0 × BC encoded as a control word (K28.5). The rest of the `tx_datain` bytes are encoded as a data word (Dx.y).

Figure 1–19. Control Word and Data Word Transmission



The IEEE 802.3 8B/10B encoder specification identifies only a set of 8-bit characters for which `tx_ctrlenable` should be asserted. If you assert `tx_ctrlenable` for any other set of bytes, the 8B/10B encoder might encode the output 10-bit code as an invalid code (it does not map to a valid Dx.y or Kx.y code), or an unintended valid Dx.y code, depending on the value entered. It is possible for a downstream 8B/10B decoder to decode an invalid control word into a valid Dx.y code without asserting any code error flags.

For example, depending on the current running disparity, the invalid code K24.1 (`tx_datain` = 8'h38 + `tx_ctrl` = 1'b1) can be encoded to 10'b0110001100 (0 × 18C), which is equivalent to a D24.6+ (8'hD8 from the RD+ column). Altera recommends that you do not assert `tx_ctrlenable` for unsupported 8-bit characters.

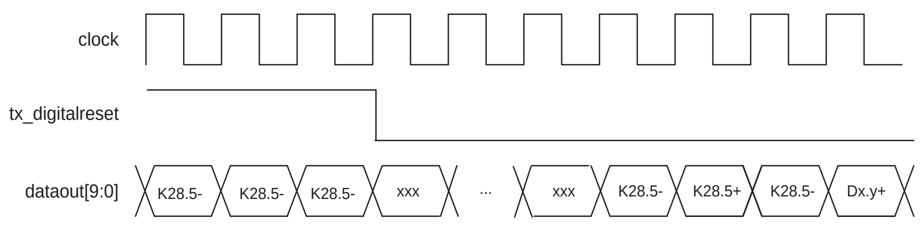
Reset Condition

The `tx_digitalreset` signal resets the 8B/10B encoder. During reset, the running disparity and data registers are cleared. Also, the 8B/10B encoder outputs a K28.5 pattern from the RD- column continuously until `tx_digitalreset` is de-asserted. The input data and control code from the FPGA fabric is ignored during the reset state. Once out of reset, the 8B/10B encoder starts with a negative disparity (RD-) and transmits three K28.5 code groups for synchronization before it starts encoding and transmitting data on its output.

While `tx_digitalreset` is asserted, the downstream 8B/10B decoder that receives the data might observe synchronization or disparity errors.

Figure 1–20 shows the reset behavior of the 8B/10B encoder. When in reset (`tx_digitalreset` is high), a K28.5- (K28.5 10-bit code group from the RD- column) is sent continuously until `tx_digitalreset` is low. Due to some pipelining of the transmitter channel PCS, some “don’t cares” (10'hxxx) are sent before the three synchronizing K28.5 code groups. User data follows the third K28.5 code group.

Figure 1–20. 8B/10B Encoder Output during tx_digitalreset Assertion



Controlling Running Disparity

Upon power on or reset, the 8B/10B encoder has a negative disparity and chooses the 10-bit code from the RD- column (refer to the 8B/10B encoder specification for the RD+ and RD- column values). The ALTGX MegaWizard Plug-In Manager provides the `tx_forcedisp` and `tx_dispval` ports to control the running disparity of the output from the 8B/10B encoder. These ports are available only in Basic single-width mode.

A high value on the `tx_forcedisp` port is the control signal to the disparity value of the output data. The disparity value (RD+ or RD-) is indicated by the value on the `tx_dispval` port. If the `tx_forcedisp` port is low, `tx_dispval` is ignored and the current running disparity is not altered. Forcing disparity can either maintain the current running disparity calculations if the forced disparity value (on the `tx_dispval` bit) happens to match the current running disparity, or flip the current running disparity calculations if it does not. If the forced disparity flips the current running disparity, the downstream 8B/10B decoder might detect a disparity error.

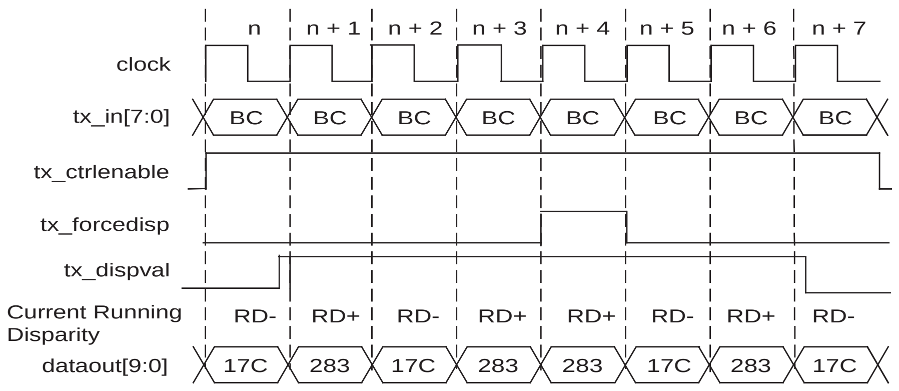
Table 1–7 shows the `tx_forcedisp` and `tx_dispval` port values.

Table 1–7. tx_forcedisp and tx_dispval Port Values

tx_forcedisp	tx_dispval	Disparity Value
0	X	Current running disparity has no change
1	0	Encoded data has positive disparity
1	1	Encoded data has negative disparity

Figure 1–21 shows the current running disparity being altered in Basic single-width mode by forcing a positive disparity K28.5 when it was supposed to be a negative disparity K28.5. In this example, a series of K28.5 code groups are continuously being sent. The stream alternates between a positive running disparity (RD+) K28.5 and a negative running disparity (RD-) K28.5 to maintain a neutral overall disparity. The current running disparity at time n + 3 indicates that the K28.5 in time n + 4 should be encoded with a negative disparity. Because `tx_forcedisp` is high at time n + 4, and `tx_dispval` is also high, the K28.5 at time n + 4 is encoded as a positive disparity code group.

Figure 1–21. 8B/10B Encoder Force Running Disparity Operation in Single-Width Mode

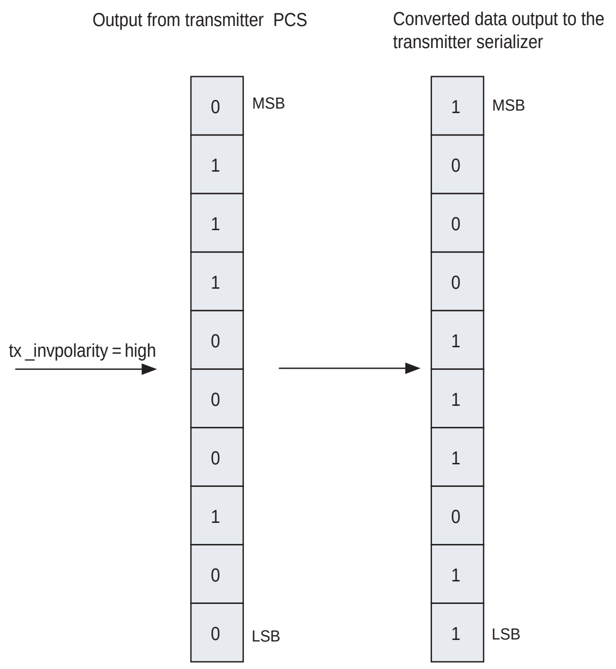


Transmitter Polarity Inversion

The positive and negative signals of a serial differential link might accidentally be swapped during board layout. Solutions like a board re-spin or major updates to the logic in the FPGA fabric can be expensive. The transmitter polarity inversion feature is provided to correct this situation. An optional `tx_invpolarity` port is available in all functional modes to dynamically enable the transmitter polarity inversion feature.

In single-width mode, a high value on the `tx_invpolarity` port inverts the polarity of every bit of the 8-bit or 10-bit input data word to the serializer in the transmitter datapath. Because inverting the polarity of each bit has the same effect as swapping the positive and negative signals of the differential link, correct data is seen by the receiver. The `tx_invpolarity` signal is dynamic and might cause initial disparity errors at the receiver of an 8B/10B encoded link. The downstream system must be able to tolerate these disparity errors. Figure 1–22 shows the transmitter polarity inversion feature in a single-width 10-bit wide datapath configuration.

Figure 1–22. Transmitter Polarity Inversion in Single-Width Mode


Output from transmitter PCS
Converted data output to the transmitter serializer
tx _invpolarity = high
0 MSB
1
1
1
0
0
0
1
0
0 LSB
1 MSB
0
0
0
1
1
1
0
1
1 LSB

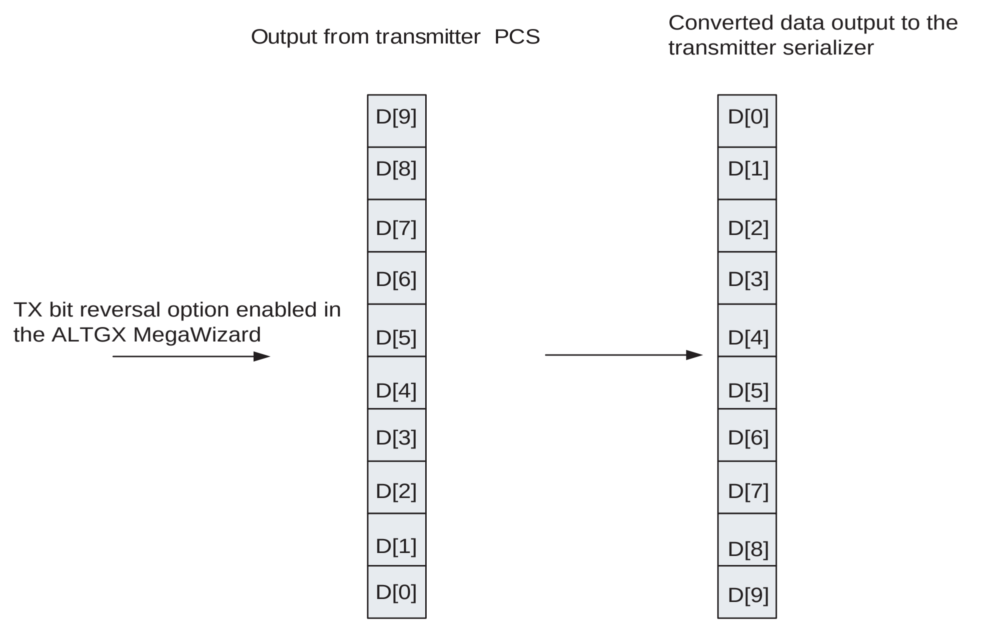
Transmitter Bit Reversal

By default, the Arria II GX transmit bit order is LSBit to MSBit. In single-width mode, the LSB of the 8- or 10-bit data word is transmitted first, followed by the MSB. The transmitter bit reversal feature allows reversing the transmit bit order as MSB to LSB before it is forwarded to the serializer.

If you enable the transmitter bit reversal feature in Basic single-width mode, the 8-bit `D[7:0]` or 10-bit `D[9:0]` data at the input of the serializer gets rewired to `D[0:7]` or `D[0:9]`, respectively.

Figure 1–23 shows the transmitter bit reversal feature in Basic single-width for a 10-bit wide datapath configuration.

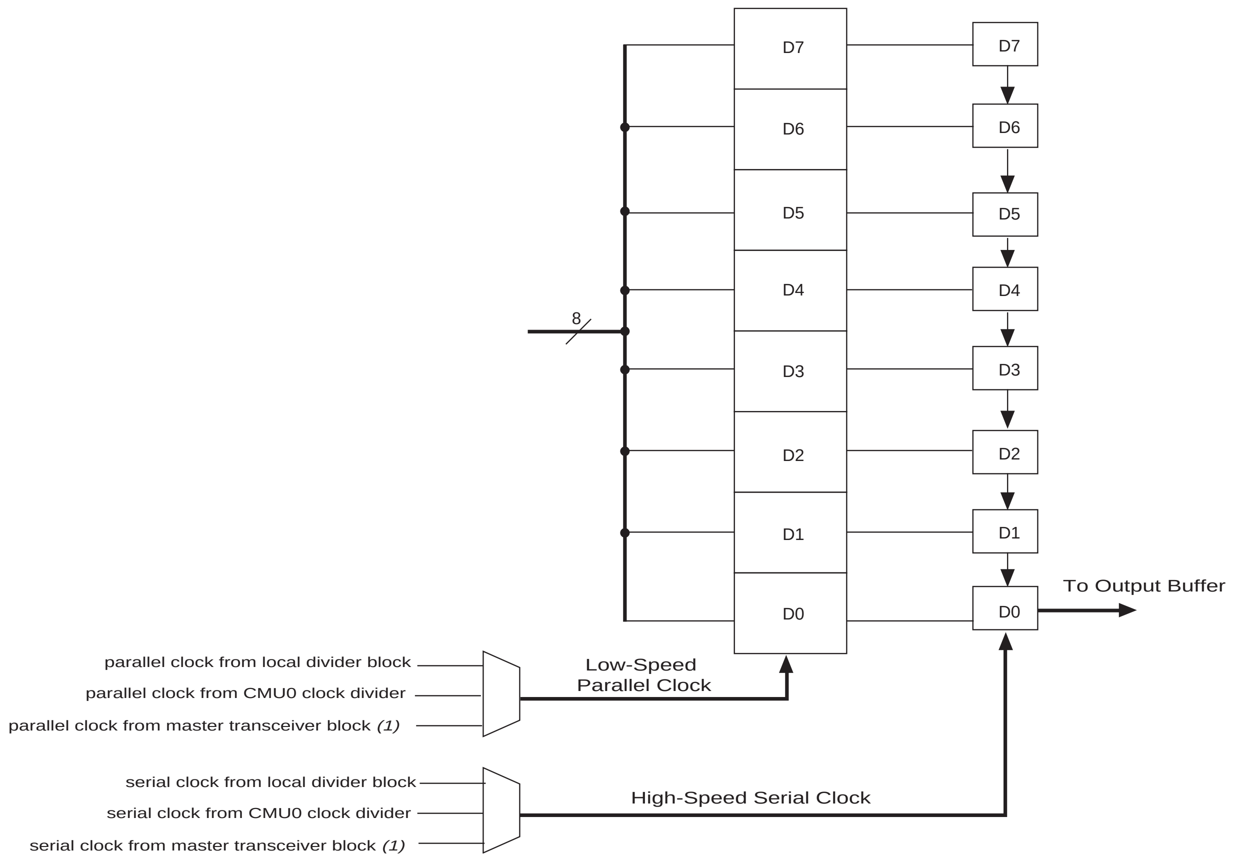
Figure 1–23. Transmitter Bit Reversal Operation in Basic Single-Width Mode


Output from transmitter PCS
Converted data output to the transmitter serializer
TX bit reversal option enabled in the ALTGX MegaWizard
D[9]
D[8]
D[7]
D[6]
D[5]
D[4]
D[3]
D[2]
D[1]
D[0]
D[0]
D[1]
D[2]
D[3]
D[4]
D[5]
D[6]
D[7]
D[8]
D[9]

Serializer

The serializer converts the incoming low-speed parallel signal from the transceiver PCS to high-speed serial data and sends it to the transmitter buffer. The serializer supports an 8-bit or 10-bit serialization factor in single-width mode. The serializer block drives the serial data to the output buffer, as shown in Figure 1–24. The serializer block sends out the LSB of the input data. Figure 1–25 shows the serial bit order of the serializer block output. In this example, a constant 8'h6A (01101010) value is serialized and the serial data is transmitted from LSBit to MSBit.

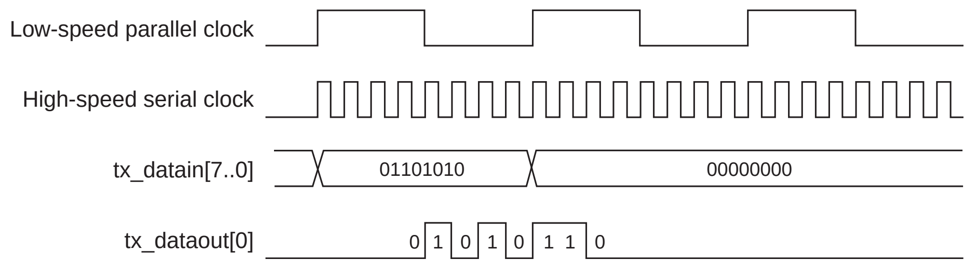
Figure 1–24. Serializer Block in an 8-Bit PCS-PMA Interface


D7
D6
D5
D4
D3
D2
D1
D0
8
To Output Buffer
parallel clock from local divider block
parallel clock from CMU0 clock divider
parallel clock from master transceiver block (1)
Low-Speed Parallel Clock
serial clock from local divider block
serial clock from CMU0 clock divider
serial clock from master transceiver block (1)
High-Speed Serial Clock

Note to Figure 1–24:

(1) The CMU0 clock divider of the master transceiver block provides the clocks. It is used only in PCI Express (PIPE) ×8 mode.

Figure 1–25. Serializer Bit Order *(Note 1)*



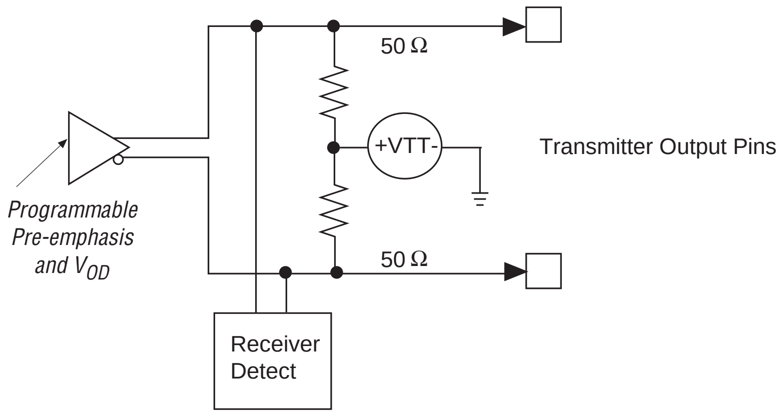
Note to Figure 1–25:

(1) This figure assumes that the input data to the serializer is 8 bits (channel width = 8 bits with the 8B/10B encoder disabled).

Transmitter Output Buffer

The Arria II GX transmitter buffers support 1.5-V pseudo current mode logic (PCML) and can drive 40 inches of FR4 trace across two connectors. The transmitter buffer voltage level is (V_{CCH}) set to 1.5 V by the ALTGX MegaWizard Plug-In Manager. With the 1.5 V (available in Quartus II software version 8.1 or later) setting, you can run the transmitter channel from 600 Mbps to 3.75 Gbps. The transmitter buffer power supply only provides voltage to the transmitter output buffers in the transceiver channels. The transmitter output buffer, as shown in Figure 1–26, has additional circuitry to improve signal integrity, such as V_{OD}, programmable pre-emphasis circuit, internal termination circuitry, and receiver detect capability to support PCI Express (PIPE) functional mode.

Figure 1–26. Transmitter Output Buffer



Transmitter Termination

The Arria II GX transmitter buffer includes on-chip differential termination of 100 Ω. The resistance is adjusted by the on-chip calibration circuit in the calibration block (for more information, refer to "Calibration Blocks" on page 1–137), which compensates for temperature, voltage, and process changes. The Arria II GX transmitter buffers in the transceiver are current-mode drivers. Therefore, the resultant V_{OD} is a function of the transmitter termination value. For more information about resultant V_{OD} values, refer to "Programmable Output Differential Voltage" on page 1–40.

You can disable on-chip termination (OCT) and use external termination. If you select external termination, the transmitter common mode is tri-stated. You can set transmitter termination in the ALTGX MegaWizard Plug-In Manager.

You can also set OCT through the assignment editor. Set the assignment shown in Table 1–8 to the transmitter serial output pin.

Table 1–8. Arria II GX OCT Assignment Settings

Assign To	Transmitter Serial Output Data Pin
Assignment Name:	Output Termination
Available Values:	OCT 100 Ω

Programmable Output Differential Voltage

The Arria II GX device allows you to customize the differential output voltage to handle different trace lengths, various backplanes, and various receiver requirements, as shown in Figure 1–27.

Figure 1–27. V_{OD} (Differential) Signal Level

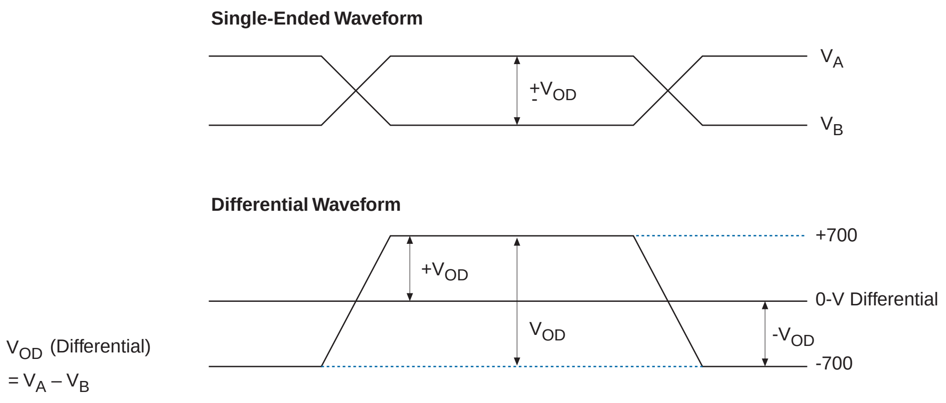


Table 1–9 shows the V_{OD} values for 100 Ω termination resistor setting.

Table 1–9. Programmable V_{OD} Differential Peak-to-Peak *(Note 1)*

V_{OD} for 100 Ω Termination Setting	Unit
400	mV
600	mV
800	mV
900	mV
1000	mV
1200	mV

Note to Table 1–9:

(1) These values are preliminary.

You can set the above V_{OD} values in the ALTGX MegaWizard Plug-In Manager.

Programmable Pre-Emphasis

The programmable pre-emphasis module in each transmit buffer boosts high frequencies in the transmit data signal, which might be attenuated in the transmission media. Using pre-emphasis can maximize the data eye opening at the far-end receiver. The transmission line's transfer function can be represented in the frequency domain as a low-pass filter. Any frequency components below the –3dB frequency can pass through with minimal loss. Frequency components greater than –3dB frequency are attenuated. This variation in frequency response yields data-dependent jitter and other intersymbol interference (ISI) effects. By applying pre-emphasis, the high-frequency components are boosted; that is, pre-emphasized. Pre-emphasis equalizes the frequency response at the receiver so the differences between the low-frequency and high-frequency components are reduced, which minimizes the ISI effects from the transmission medium. Pre-emphasis requirements increase as data rates through legacy backplanes increase. Set the pre-emphasis settings in the ALTGX MegaWizard Plug-In Manager.

Transmitter Output Buffer Power (V_{CCH})

Arria II GX transmitter buffer power is set to 1.5 V by the ALTGX MegaWizard Plug-In Manager.

Common Mode Voltage (V_{CM}) Settings

The Arria II GX devices provide a V_{CM} of 650 mV.

PCI Express (PIPE) Receiver Detect

The Arria II GX transmitter buffer has a built-in receiver detection circuit for use in the PCI Express (PIPE) mode for Gen1 data rates. This circuit detects if there is a receiver downstream by sending out a pulse on the common mode of the transmitter and monitoring the reflection. This mode requires the transmitter buffer to be tri-stated (in Electrical Idle mode), OCT utilization, and the 125 MHz `fixedclk` signal. You can enable this feature in PCI Express (PIPE) mode by setting the `tx_forceelecidle` and `tx_detectrxloopback` ports to **1'b1**. Receiver detect circuitry is active only in the P1 power state (refer to the PIPE 2.00 specification for more information about power states).

In the P1 power state, the transmitter output buffer is tri-stated because the transmitter output buffer is in electrical idle. A high on the `tx_detectrxloopback` port triggers the receiver detect circuitry to alter the transmitter output buffer common mode voltage. The sudden change in common mode voltage effectively appears as a step voltage at the tri-stated transmitter buffer output. If a receiver (that complies with PCI Express [PIPE] input impedance requirements) is present at the far end, the time constant of the step voltage is higher. If a receiver is not present or is powered down, the time constant of the step voltage is lower. The receiver detect circuitry snoops the transmitter buffer output for the time constant of the step voltage to detect the presence of the receiver at the far end. A high pulse is driven on the `pipephydonestatus` port and 3'b011 is driven on the `pipestatus` port to indicate that a receiver has been detected. There is some latency after asserting the `tx_detectrxloopback` signal, before the receiver detection is indicated on the `pipephydonestatus` port. For the signal timing to perform the receiver detect operation, refer to Figure 1–74 on page 1–94.

☞ The `tx_forceelecidle` port must be asserted at least 10 parallel clock cycles prior to the `tx_detectrxloopback` port to ensure that the transmitter buffer is tri-stated.

PCI Express (PIPE) Electrical Idle

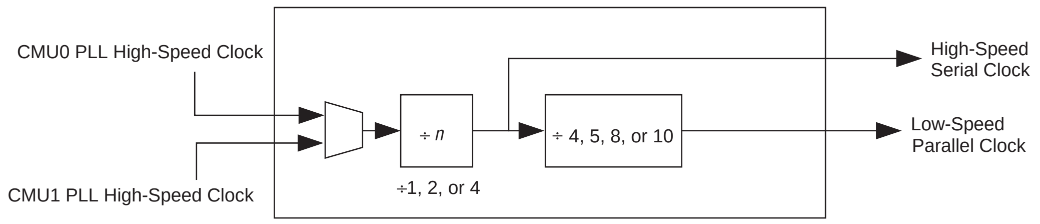
The Arria II GX transmitter output buffer supports transmission of PCI Express Electrical Idle (or individual transmitter tri-state). This feature is only active in PCI Express (PIPE) mode. The `tx_forceelecidle` port puts the transmitter buffer in Electrical Idle mode. This port has a specific functionality in each power state. Refer to PIPE specification 2.00 for use of the `tx_forceelecidle` signal under different power states. For the signal timing to perform the electrical idle transmission in PCI Express (PIPE) mode, refer to Figure 1–73 on page 1–93.

Transmitter Local Clock Divider Block

Each transmitter channel contains a local clock divider block. It receives the high-speed clock from CMU0 PLL or CMU1 PLL and generates the high-speed serial clock for the serializer and the low-speed parallel clock for the transmitter PCS data path. The low-speed parallel clock is also forwarded to the FPGA fabric (`tx_clkout`). The local clock divider block allows each transmitter channel to run at /1, /2, or /4 of the CMU PLL data rate. Note that the local clock divider block is used only in non-bonded functional modes (for example, GIGE, SONET/SDH, and SDI mode). The Quartus II software automatically selects the local clock divider block in non-bonded functional modes.

Figure 1–28 shows the transmitter local clock divider block.

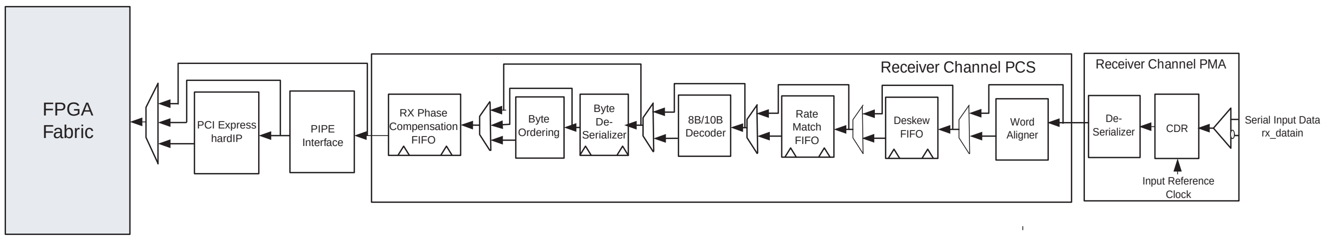
Figure 1–28. Transmitter Local Clock Divider Block



Receiver Channel Datapath

Figure 1–29 shows the receiver channel datapath in Arria II GX devices.

Figure 1–29. Receiver Channel Datapath



The receiver channel PMA datapath consists of the following blocks:

- Receiver input buffer
- Clock and data recovery (CDR) unit
- Deserializer

The receiver channel PCS datapath consists of the following blocks:

- Word aligner
- Deskew FIFO
- Rate match (clock rate compensation) FIFO
- 8B/10B decoder
- Byte deserializer
- Byte ordering
- Receiver phase compensation FIFO
- PIPE interface

The receiver datapath is very flexible and allows multiple configurations, depending on the selected functional mode. You can configure the receiver datapath using the ALTGX MegaWizard Plug-In Manager.

This section discusses Arria II GX receiver channel datapath architecture. The sub-blocks in the receiver datapath are described in order from the serial receiver input buffer to the receiver phase compensation FIFO buffer at the FPGA fabric—transceiver interface.

Receiver Input Buffer

The receiver input buffer receives serial data from the `rx_datain` port and feeds it to the CDR unit. In the reverse serial loopback (pre-CDR) configuration, it also feeds the received serial data to the transmitter output buffer. Figure 1–30 shows the receiver input buffer.

Figure 1–30. Receiver Input Buffer

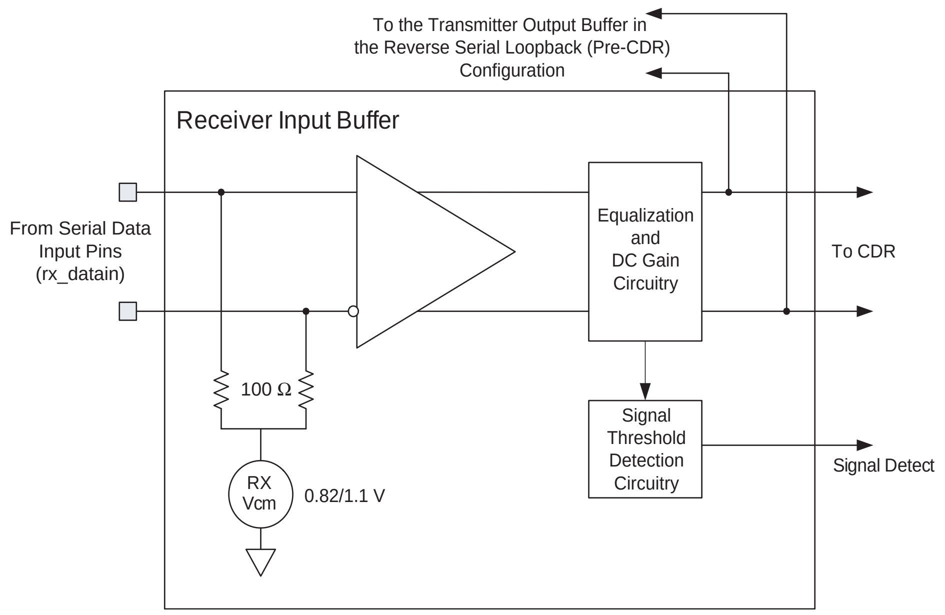

To the Transmitter Output Buffer in the Reverse Serial Loopback (Pre-CDR) Configuration
Receiver Input Buffer
From Serial Data Input Pins (rx_datain)
Equalization and DC Gain Circuitry
To CDR
100 Ω
RX Vcm
0.82/1.1 V
Signal Threshold Detection Circuitry
Signal Detect

Table 1–10 shows the electrical features supported by the receiver input buffer.

Table 1–10. Electrical Features Supported by the Receiver Input Buffer

Data Rate (Gbps)	I/O Standard	Differential On-Chip Termination with Calibration (Ω)	Common Mode Voltage (V)	Coupling
0.6 – 3.75	1.4 V PCML	100	0.82	AC, DC
	1.5 V PCML	100	0.82	AC, DC
	2.5 V PCML	100	0.82	AC
	LVPECL	100	0.82	AC
	LVDS	100	1.1	AC, DC

The Arria II GX receiver buffer supports the following features:

- Programmable differential OCT
- Programmable common mode voltage
- AC and DC coupling
- Programmable equalization and DC gain
- Signal threshold detection circuitry

Differential On-Chip Termination

The Arria II GX receiver buffers support optional differential on-chip termination of 100 Ω. On-chip termination can be set using the Quartus II software Assignment Editor, as shown in Table 1–11.

Table 1–11. Arria II GX Receiver On-Chip Termination Assignment Settings

Assign To	rx_datain (Receiver Input Data Pins)
Assignment Name:	Input Termination
Available Values:	OCT 100 Ω, Off

☞ The Arria II GX receiver OCT resistors have calibration support to compensate for process, voltage, and temperature variations. For more information about OCT calibration support, refer to "Calibration Blocks" on page 1–137.

Programmable Common Mode Voltage

The Arria II GX receiver buffers have on-chip biasing circuitry to establish the required common mode voltage at the receiver input. It supports two common mode voltage settings of 0.82 V and 1.1 V that you can select in the ALTGX MegaWizard Plug-In Manager.

You must select **0.82 V** as the receiver buffer common mode voltage for the following receiver input buffer I/O standards:

- 1.4-V PCML
- 1.5-V PCML
- 2.5-V PCML
- LVPECL

You must select **1.1 V** as the receiver buffer common mode voltage for the following receiver input buffer I/O standard:

- LVDS

☞ On-chip biasing circuitry is effective only if you select on-chip receiver termination. If you select external termination, you must implement off-chip biasing circuitry to establish the common mode voltage at the receiver input buffer.

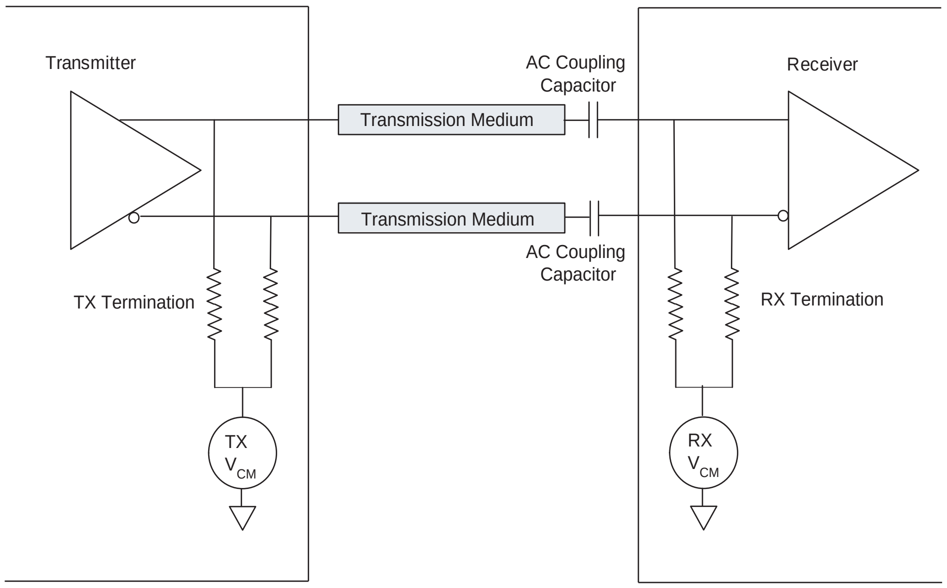
Link Coupling

A high-speed serial link can either be AC-coupled or DC-coupled, depending on the serial protocol being implemented. While most of the serial protocols require links to be AC-coupled, protocols similar to SONET optionally allow DC coupling.

AC-Coupled Links

In an AC-coupled link, the AC-coupling capacitor blocks the transmitter DC common mode voltage. The on-chip receiver termination and biasing circuitry automatically restores the selected common mode voltage. Figure 1–31 shows an AC-coupled link.

Figure 1–31. AC-Coupled Link



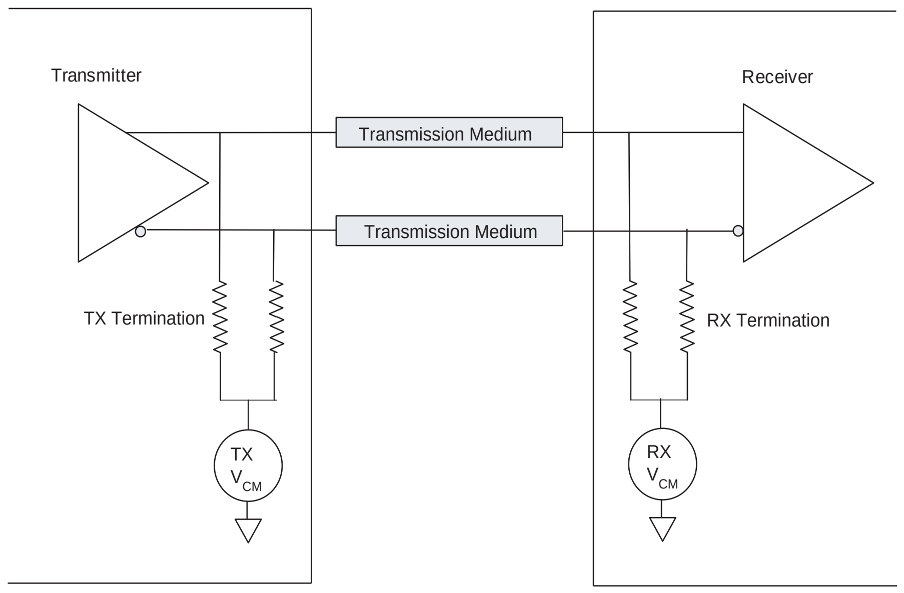
The following protocols supported by Arria II GX devices mandate AC-coupled links:

- PCI Express (PIPE)
- Gigabit Ethernet
- Serial RapidIO
- XAUI
- SDI

DC-Coupled Links

In a DC-coupled link, the transmitter DC common mode voltage is seen unblocked at the receiver buffer. The link common mode voltage depends on the transmitter common mode voltage and the receiver common mode voltage. The on-chip or off-chip receiver termination and biasing circuitry must ensure compatibility between the transmitter and the receiver common mode voltage. Figure 1–32 shows a DC-coupled link.

Figure 1–32. DC-Coupled Link



You might choose to use the DC-coupled high-speed link for Basic single-width mode only.

The following sections discuss DC-coupling requirements for a high-speed link with an Arria II GX device used as the transmitter, receiver, or both. Specifically, the following link configurations are discussed:

- Arria II GX Transmitter (PCML) to Arria II GX Receiver (PCML)
- Stratix II GX Transmitter (PCML) to Arria II GX Receiver (PCML)
- Arria II GX Transmitter (PCML) to Stratix II GX Receiver (PCML)
- LVDS Transmitter to Arria II GX Receiver (PCML)

Figure 1–33 shows a typical Arria II GX transmitter (PCML) to Arria II GX Receiver (PCML) DC-coupled link.

Figure 1–33. Arria II GX Transmitter (PCML) to Arria II GX Receiver (PCML) DC-Coupled Link

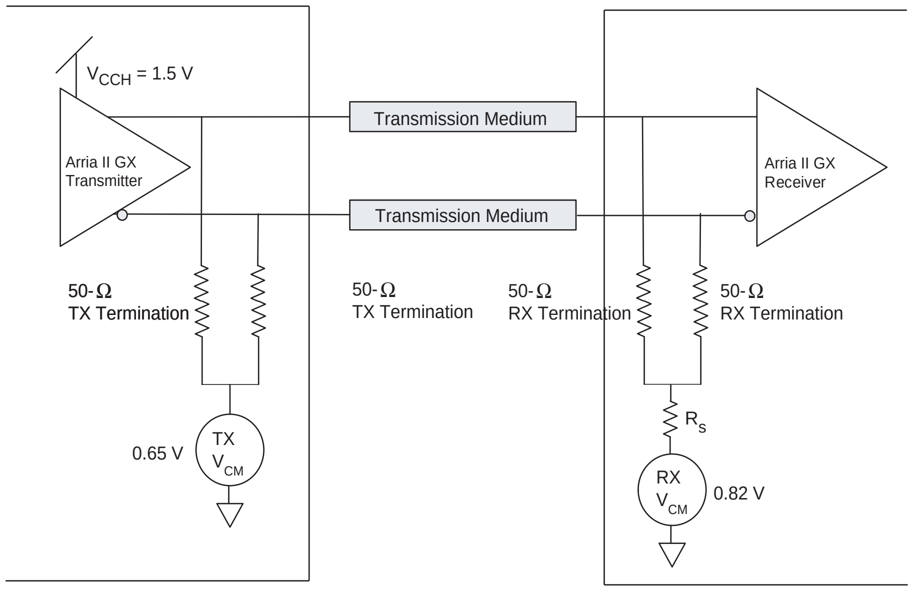


Table 1–12 shows the allowed transmitter and receiver settings in an Arria II GX transmitter (PCML) to Arria II GX receiver (PCML) DC coupled link.

Table 1–12. Settings for an Arria II GX Transmitter (PCML) to Arria II GX Receiver (PCML) DC-Coupled Link

Transmitter (Arria II GX) Settings				Receiver (Arria II GX) Settings		
Data Rate	VCCH (1)	TX VCM	Differential Termination	Data Rate	RX VCM	Differential Termination
600-3750 Mbps	1.5 V	0.65 V	100 Ω	600-3750 Mbps	0.82 V	100-Ω

Note to Table 1–12:

(1) V_{CCH} = 1.5 V can support data rates from 600 Mbps to 3750 Mbps.

Figure 1–34 shows the Stratix II GX transmitter (PCML) to Arria II GX receiver (PCML) DC-coupled link.

Figure 1–34. Stratix II GX Transmitter (PCML) to Arria II GX Receiver (PCML) DC-Coupled Link

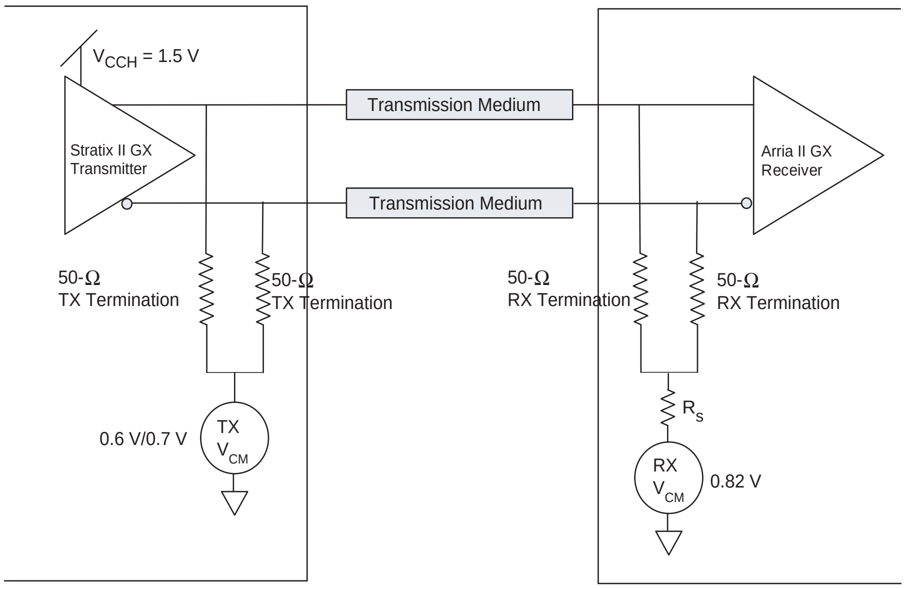


Table 1–13 shows the allowed transmitter and receiver settings in a Stratix II GX to Arria II GX coupled link.

Table 1–13. Settings for a Stratix II GX to Arria II GX DC-Coupled Link

Transmitter (Stratix II GX) Settings				Receiver (Arria II GX) Settings		
Data Rate	**VCCH (1)**	**TX VCM (1)**	**Differential Termination**	**Data Rate**	**RX VCM**	**Differential Termination**
600-3750 Mbps	1.5 V (1.5 V PCML)	0.6 V/0.7 V	100 Ω	600-3750 Mbps	0.82 V	100-Ω

Note to Table 1–13:

(1) V_{CCH} = 1.5 V with TX Vcm = 0.7 V can support data rates from 600 Mbps to 3125 Mbps. V_{CCH} = 1.5 V with TX Vcm = 0.6 V can support data rates from 600 Mbps to 6375 Mbps.

Figure 1–35 shows the Arria II GX transmitter (PCML) to Stratix II GX receiver (PCML) DC-coupled link.

Figure 1–35. Arria II GX Transmitter (PCML) to Stratix II GX Receiver (PCML) DC-Coupled Link

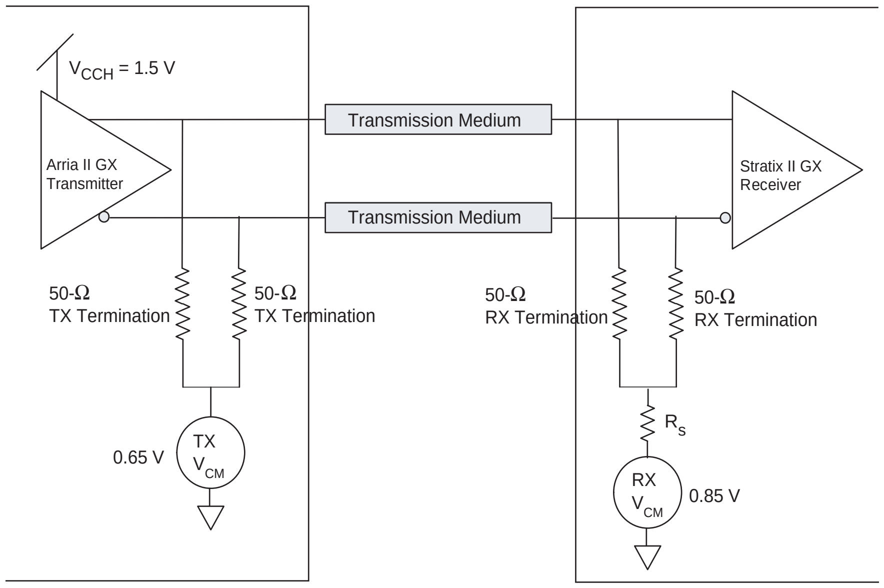


Table 1–14 shows the allowed transmitter and receiver settings in an Arria II GX transmitter (PCML) to Stratix II GX receiver (PCML) DC-coupled link.

Table 1–14. Settings for an Arria II GX to Stratix II GX DC-Coupled Link

Transmitter (Arria II GX) Settings				Receiver (Stratix II GX) Settings			
Data Rate	**VCCH (1)**	**TX VCM**	**Differential Termination**	**Data Rate**	**I/O Standard**	**RX VCM**	**Differential Termination**
600-3750 Mbps	1.5 V	0.65 V	100 Ω	600-3750 Mbps	1.5 V PCML	0.85 V	100-Ω

Note to Table 1–14:

(1) V_{CCH} = 1.5 V can support data rates from 600 Mbps to 3750 Mbps.

Figure 1–36 shows the LVDS transmitter to Arria II GX receiver (PCML) DC-coupled link.

Figure 1–36. LVDS Transmitter to Arria II GX Receiver (PCML) DC-Coupled Link

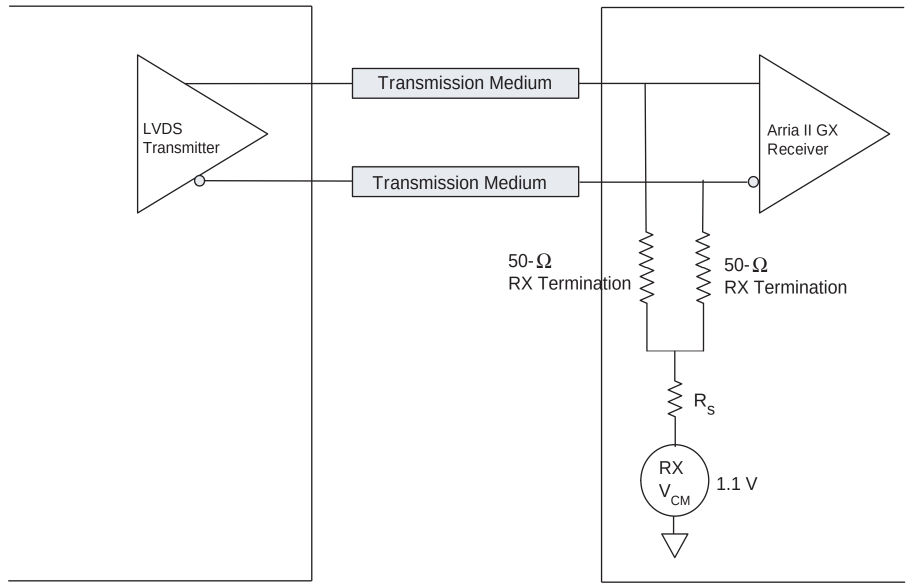


Table 1–15 shows the allowed transmitter and receiver settings in a LVDS transmitter to Arria II GX receiver DC-coupled link.

Table 1–15. Settings for a LVDS transmitter to Arria II GX Receiver DC-Coupled Link *(Note 1)*

Receiver (Arria II GX) Settings		
RX VCM	**Differential Termination**	**RS**
1.1 V	100 Ω	*(2)*

Notes to Table 1–15:

(1) When DC coupling an LVDS transmitter to the Arria II GX receiver, use RX Vcm = 1.1 V and series resistance value RS to verify compliance to the LVDS specification.

(2) Pending characterization.

Programmable Equalization and DC Gain

The transfer function of the physical medium can be represented as a low-pass filter in the frequency domain. Frequency components below –3 dB frequency pass through with minimal loss. Frequency components greater than –3 dB frequency get attenuated as a function of frequency due to skin-effect and dielectric losses. This variation in frequency response yields data-dependant jitter and other ISI effects, which can cause incorrect sampling of the input data.

Each Arria II GX receiver buffer has independently programmable equalization circuitry that boosts the high-frequency gain of the incoming signal, thereby compensating for the low-pass filter effects of the physical medium. The amount of high-frequency gain required depends on the loss characteristics of the physical medium. Arria II GX equalization circuitry supports equalization settings that provide up to 7 dB of high-frequency boost. You can select the appropriate equalization setting in the ALTGX MegaWizard Plug-In Manager.

The Arria II GX receiver buffer also supports programmable DC gain circuitry. Unlike equalization circuitry, DC gain circuitry provides equal boost to the incoming signal across the frequency spectrum. The receiver buffer supports DC gain settings of 0 dB, 3 dB, and 6 dB. You can select the appropriate DC gain setting in the ALTGX MegaWizard Plug-In Manager.

Signal Threshold Detection Circuitry

In PCI Express (PIPE) mode, you can enable the optional signal threshold detection circuitry by NOT selecting the **Force signal detection** option in the ALTGX MegaWizard Plug-In Manager. If enabled, this option senses whether the signal level present at the receiver input buffer is above the signal detect threshold voltage that you specified in the **What is the signal detect and signal loss threshold?** option in the ALTGX MegaWizard Plug-In Manager.

☞ The appropriate signal detect threshold level that complies with the PCI Express (PIPE) compliance parameter VRX-IDLE-DETDIFFp-p is pending characterization.

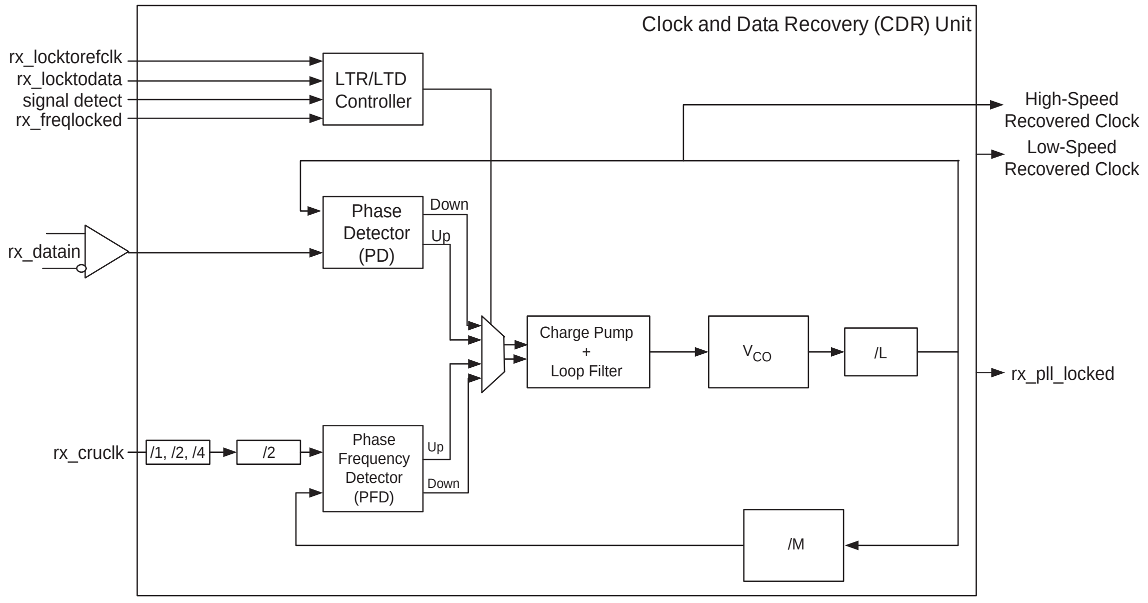
Signal threshold detection circuitry has a hysteresis response that filters out any high-frequency ringing caused by inter-symbol interference or high-frequency losses in the transmission medium. If the signal threshold detection circuitry senses the signal level present at the receiver input buffer to be higher than the signal detect threshold, it asserts the `rx_signaldetect` signal high. Otherwise, the signal threshold detection circuitry de-asserts the `rx_signaldetect` signal low. If you select the **Force signal detection** option in the ALTGX MegaWizard Plug-In Manager, `rx_signaldetect` is always asserted high, irrespective of the signal level on the receiver input buffer.

The `rx_signaldetect` signal is also used by the lock-to-reference/lock-to-data (LTR/LTD) controller in the receiver CDR to switch between LTR and LTD lock modes. When the signal threshold detection circuitry de-asserts the `rx_signaldetect` signal, the LTR/LTD controller switches the receiver CDR from LTD to LTR lock mode. For more information, refer to "LTR/LTD Controller" on page 1–55.

Clock and Data Recovery Unit

Each Arria II GX receiver channel has an independent CDR unit to recover the clock from the incoming serial data stream. The high-speed and low-speed recovered clocks are used to clock the receiver PMA and PCS blocks. Figure 1–37 shows the CDR block diagram.

Figure 1–37. CDR Unit



The CDR operates either in LTR mode or LTD mode. In LTR mode, the CDR tracks the input reference clock. In LTD mode, the CDR tracks the incoming serial data.

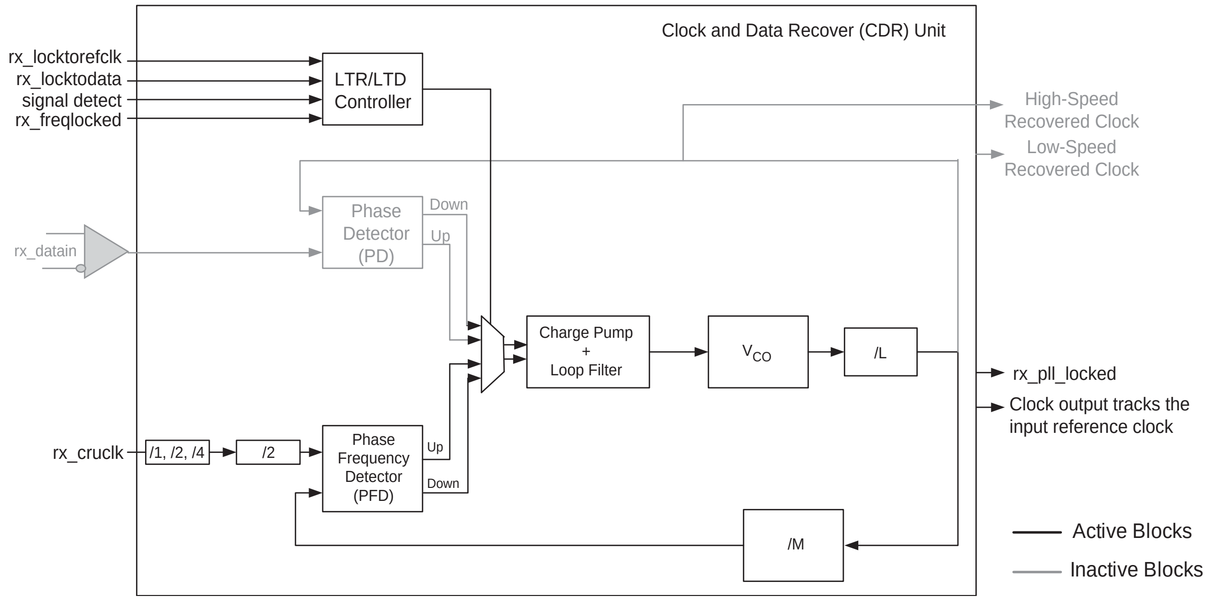
After the receiver power-up and reset cycle, the CDR must be kept in LTR mode until it locks to the input reference clock. Once locked to the input reference clock, the CDR output clock is trained to the configured data rate. The CDR can now switch to LTD mode to recover the clock from incoming data. The LTR/LTD controller controls the switch between LTR and LTD modes.

Lock-to-Reference Mode

In LTR mode, the phase frequency detector in the CDR tracks the receiver input reference clock, `rx_cruclk`. The PFD controls the charge pump that tunes the VCO in the CDR. Depending on the data rate and the selected input reference clock frequency, the Quartus II software automatically selects the appropriate /M and /L divider values such that the CDR output clock frequency is half the data rate. An active high, the `rx_pll_locked` status signal is asserted to indicate that the CDR has locked to phase and frequency of the receiver input reference clock. Figure 1–38 shows active blocks when CDR is in LTR mode.

The phase detector (PD) is inactive in LTR mode.

Figure 1–38. CDR in Lock-To-Reference Mode


Clock and Data Recover (CDR) Unit
rx_locktorefclk
rx_locktodata
signal detect
rx_freqlocked
LTR/LTD Controller
High-Speed Recovered Clock
Low-Speed Recovered Clock
Phase Detector (PD)
Down
Up
rx_datain
Charge Pump + Loop Filter
VCO
/L
rx_pll_locked
Clock output tracks the input reference clock
rx_cruclk
/1, /2, /4
/2
Phase Frequency Detector (PFD)
Up
Down
/M
Active Blocks
Inactive Blocks

You can drive the receiver input reference clock with the following clock sources:

- Dedicated `refclk` pins (`refclk0` and `refclk1`) of the associated transceiver block
- Inter-transceiver block clock lines from other transceiver blocks on the same side of the device (up to six ITB clock lines, two from each transceiver block)
- Global PLD clock driven by a dedicated clock input pin
- Clock output from the left PLLs in the FPGA fabric

Table 1–16 shows CDR specifications in LTR mode.

Table 1–16. CDR Specification in Lock-To-Reference Mode

Parameter	Value
Input Reference Clock Frequency	50 MHz – 637.5 MHz
PFD Input Frequency	50 MHz – 325 MHz
/M Divider	4, 5, 8, 10, 16, 20, 25
/L Divider	1, 2, 4, 8

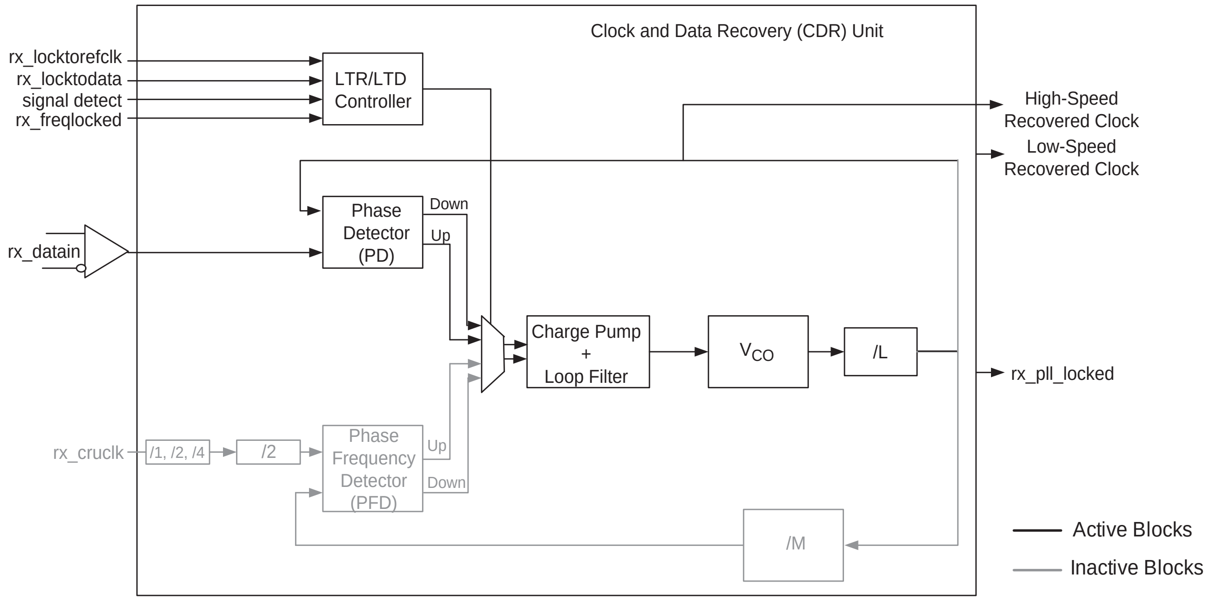
For input reference clock frequencies greater than 325 MHz, the Quartus II software automatically selects the appropriate /1, /2, or /4 pre-divider to meet the PFD input frequency limitation of 325 MHz.

Lock-to-Data Mode

The CDR must be in LTD mode to recover clock from the incoming serial data during normal operation. In LTD mode, the phase detector in the CDR tracks the incoming serial data at the receiver buffer. Depending on the phase difference between the incoming data and the CDR output clock, the PD controls the CDR charge pump that tunes the VCO. Figure 1–39 shows active blocks when the CDR is in LTD mode.

The PFD is inactive in LTD mode. The `rx_pll_locked` signal toggles randomly and has no significance in LTD mode.

Figure 1–39. CDR in Lock-To-Data Mode



After switching to LTD mode, it can take a maximum of 1 ms for the CDR to get locked to the incoming data and produce a stable recovered clock. The actual lock time depends on the transition density of the incoming data and the PPM difference between the receiver input reference clock and the upstream transmitter reference clock. The receiver PCS logic must be held in reset until the CDR produces a stable recovered clock.

For more information about receiver reset recommendations, refer to the *Reset Control and Power Down* chapter in volume 2 of the *Arria II GX Device Handbook*.

LTR/LTD Controller

The LTR/LTD controller controls whether the CDR is in LTR or LTD mode. Two optional input ports (`rx_locktorefclk` and `rx_locktodata`) allow you to configure the LTR/LTD controller in either automatic lock mode or manual lock mode. Table 1–17 shows the relationship between these optional input ports and the LTR/LTD controller lock mode.

Table 1–17. Optional Input Ports and LTR/LTD Controller Lock Mode

rx_locktorefclk	rx_locktodata	LTR/LTD Controller Lock Mode
1	0	Manual – LTR Mode
x	1	Manual – LTD Mode
0	0	Automatic Lock Mode

If you do not instantiate the optional rx_locktorefclk and rx_locktodata signals, the Quartus II software automatically configures the LTR/LTD controller in automatic lock mode.

Automatic Lock Mode

In automatic lock mode, the LTR/LTD controller initially sets the CDR to lock to the input reference clock (LTR mode). After the CDR locks to the input reference clock, the LTR/LTD controller automatically sets it to lock to the incoming serial data (LTD mode) when the following three conditions are met:

- Signal threshold detection circuitry indicates the presence of valid signal levels at the receiver input buffer
- CDR output clock is within the configured PPM frequency threshold setting with respect to the input reference clock (frequency locked)
- CDR output clock and input reference clock are phase matched within approximately 0.08 UI (phase locked)

The switch from LTR to LTD mode is indicated by the assertion of the rx_freqlocked signal.

In LTD mode, the CDR uses a phase detector to keep the recovered clock phase-matched to the data. If the CDR does not stay locked-to-data due to frequency drift or severe amplitude attenuation, the LTR/LTD controller switches the CDR back to LTR mode to lock to the input reference clock. In automatic lock mode, the LTR/LTD controller switches the CDR from LTD to LTR mode when the following conditions are met:

- Signal threshold detection circuitry indicates the absence of valid signal levels at the receiver input buffer
- CDR output clock is not in the configured PPM frequency threshold setting with respect to the input reference clock

The switch from LTD to LTR mode is indicated by the de-assertion of the rx_freqlocked signal.

Manual Lock Mode

In automatic lock mode, the LTR/LTD controller relies on the PPM detector and the phase relationship detector to set the CDR in LTR or LTD mode. The PPM detector and phase relationship detector reaction times can be too long for some applications that require faster CDR lock time. You can manually control the CDR to reduce its lock time by using the rx_locktorefclk and rx_locktodata ports. In manual lock mode, the LTR/LTD controller sets the CDR in LTR or LTD mode, depending on the logic level on the rx_locktorefclk and rx_locktodata signals.

When the rx_locktorefclk signal is asserted high, the LTR/LTD controller forces the CDR to lock to the reference clock. When the rx_locktodata signal is asserted high, it forces the CDR to lock to data. When both signals are asserted, the rx_locktodata signal takes precedence over the rx_locktorefclk signal, forcing the CDR to lock to data.

When the `rx_locktorefclk` signal is asserted high, the `rx_freqlocked` signal does not have any significance and is always driven low, indicating that the CDR is in LTR mode. When the `rx_locktodata` signal is asserted high, the `rx_freqlocked` signal is always driven high, indicating that the CDR is in LTD mode. If both signals are de-asserted, the CDR is in automatic lock mode.

The Altera-recommended transceiver reset sequence varies depending on the CDR lock mode.

For more information about reset sequence recommendations, refer to the *Reset Control and Power Down* chapter in volume 2 of the *Arria II GX Device Handbook.*

Deserializer

The deserializer block clocks in serial input data from the receiver buffer using the high-speed serial recovered clock and deserializes it using the low-speed parallel recovered clock. It forwards the deserialized data to the receiver PCS channel.

In single-width mode, the deserializer supports 8-bit and 10-bit deserialization factors. Figure 1–40 shows the deserializer operation in single-width mode with a 10-bit deserialization factor.

Figure 1–40. 10-Bit Deserializer Operation in Single-Width Mode

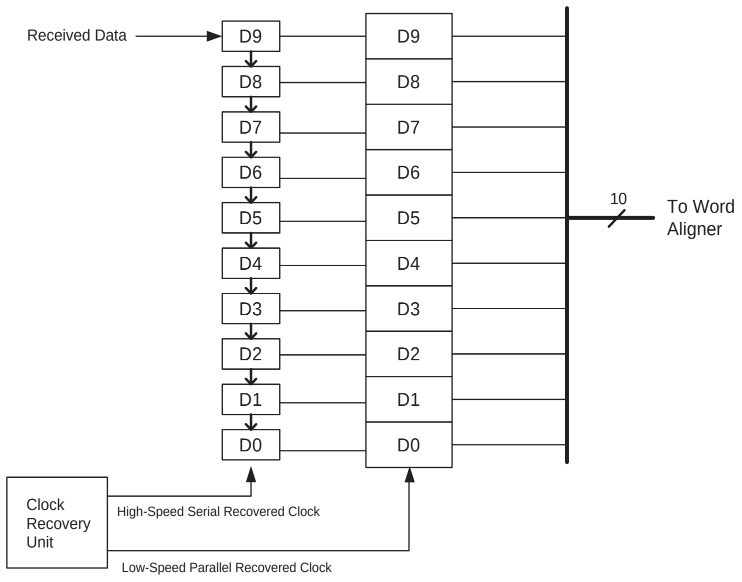
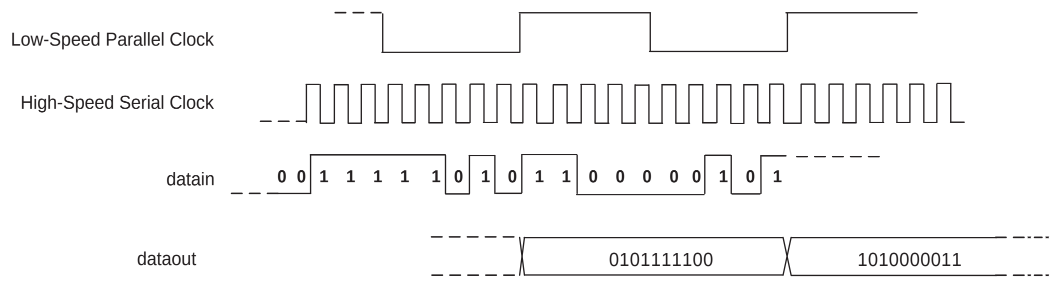


Figure 1–41 shows the serial bit order of the deserializer block input and the parallel data output of the deserializer block in single-width mode with 10-bit deserialization factor. The serial stream (0101111100) is deserialized to a value 10'h17C. The serial data is assumed to be received LSB to MSB.

Figure 1–41. 10-Bit Deserializer Bit Order in Single-Width Mode



Word Aligner

Because the data is serialized before transmission and then deserialized at the receiver, it loses the word boundary of the upstream transmitter upon deserialization. The word aligner receives parallel data from the deserializer and restores the word boundary based on a pre-defined alignment pattern that must be received during link synchronization.

Serial protocols like PCI Express (PIPE), XAUI, Gigabit Ethernet, Serial RapidIO, and SONET/SDH, specify a standard word alignment pattern. For proprietary protocols, the Arria II GX transceiver architecture allows you to select a custom word alignment pattern specific to your implementation.

In addition to restoring the word boundary, the word aligner also implements the following features:

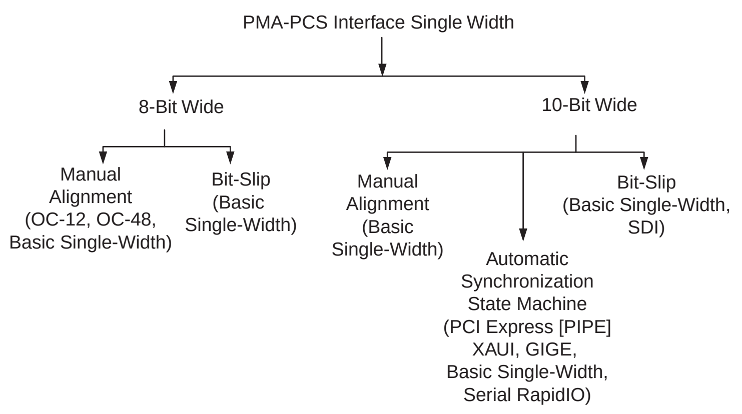
- Synchronization state machine in functional modes like PCI Express (PIPE), XAUI, GIGE, Serial RapidIO, and Basic single-width
- Programmable run length violation detection in all functional modes
- Receiver polarity inversion in all functional modes except PCI Express (PIPE)
- Receiver bit reversal in Basic single-width mode

Depending on the configured functional mode, the word aligner operates in one of the following three modes:

- Manual alignment mode
- Automatic synchronization state machine mode
- Bit-slip mode

Figure 1–42 shows the word aligner operation in all supported configurations.

Figure 1–42. Word Aligner in All Supported Configurations


PMA-PCS Interface Single Width
8-Bit Wide
10-Bit Wide
Manual Alignment (OC-12, OC-48, Basic Single-Width)
Bit-Slip (Basic Single-Width)
Manual Alignment (Basic Single-Width)
Automatic Synchronization State Machine (PCI Express [PIPE] XAUI, GIGE, Basic Single-Width, Serial RapidIO)
Bit-Slip (Basic Single-Width, SDI)

Word Aligner in Single-Width Mode

In single-width mode, the PMA-PCS interface is either 8-bit or 10-bit wide. In 8-bit wide PMA-PCS interface modes, the word aligner receives 8-bit wide data from the deserializer. In 10-bit wide PMA-PCS interface modes, the word aligner receives 10-bit wide data from the deserializer. Depending on the configured functional mode, you can configure the word aligner in manual alignment mode, automatic synchronization state machine mode, or bit-slip mode.

Word Aligner in Single-Width Mode with 8-Bit PMA-PCS Interface Modes

The following functional modes support the 8-bit PMA-PCS interface:

- SONET/SDH OC-12
- SONET/SDH OC-48
- Basic single-width

Table 1–18 shows the word aligner configurations allowed in functional modes with an 8-bit wide PMA-PCS interface.

Table 1–18. Word Aligner Configurations with an 8-Bit Wide PMA-PCS Interface

Functional Mode	Allowed Word Configurations	Allowed Word Alignment Pattern Length
SONET/SDH OC-12	Manual Alignment	16 bits
SONET/SDH OC-48	Manual Alignment	16 bits
Basic single-width	Manual Alignment, Bit-Slip	16 bits

Manual Alignment Mode Word Aligner in 8-Bit PMA-PCS Interface Modes

In manual alignment mode, word aligner operation is controlled by the input signal `rx_enapatternalign`. Word aligner operation is edge-sensitive to the `rx_enapatternalign` signal. After de-assertion of `rx_digitalreset`, a rising edge on the `rx_enapatternalign` signal triggers the word aligner to look for the word alignment pattern in the received data stream. In SONET/SDH OC-12 and

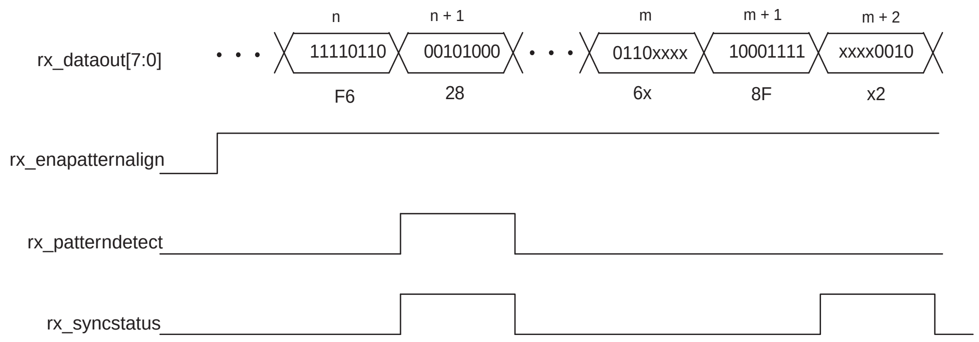
OC-48 modes, the word aligner looks for 16'hF628 (A1A2) or 32'hF6F62828 (A1A1A2A2), depending on whether the input signal `rx_a1a2size` is driven low or high, respectively. In Basic single-width mode, the word aligner looks for the 16-bit word alignment pattern programmed in the ALTGX MegaWizard Plug-In Manager. The word aligner aligns the 8-bit word boundary to the first word alignment pattern received after the rising edge on the `rx_enapatternalign` signal.

Two status signals, `rx_syncstatus` and `rx_patterndetect`, with the same latency as the datapath, are forwarded to the FPGA fabric to indicate word aligner status. On receiving the first word alignment pattern after the rising edge on the `rx_enapatternalign` signal, both the `rx_syncstatus` and `rx_patterndetect` signals are driven high for one parallel clock cycle synchronous to the MSByte of the word alignment pattern. Any word alignment pattern received thereafter in the same word boundary causes only the `rx_patterndetect` signal to go high for one clock cycle. Any word alignment pattern received thereafter in a different word boundary causes only the `rx_syncstatus` signal to go high for one clock cycle.

☞ In order for the word aligner to re-synchronize to a new word boundary, you must de-assert `rx_enapatternalign` and re-assert it again to create a rising edge.

Figure 1–43 shows word aligner behavior in SONET/SDH OC-12 functional mode. The LSByte (8'hF6) and the MSByte (8'h28) of the 16-bit word alignment pattern are received in parallel clock cycles n and n + 1, respectively. The `rx_syncstatus` and `rx_patterndetect` signals are both driven high for one parallel clock cycle synchronous to the MSByte (8'h28) of the word alignment pattern. After initial word alignment, the 16-bit word alignment pattern is again received across the word boundary in clock cycles m, m + 1, and m + 2. Because the word alignment pattern is received in a different word boundary, only the `rx_syncstatus` signal is driven high for one clock cycle. If you create a rising edge on the `rx_enapatternalign` signal before the word alignment pattern received across clock cycles m, m + 1, and m + 2, the word aligner re-aligns to the new word boundary.

Figure 1–43. Bit-Slip Mode in 8-Bit PMA-PCS Interface Mode



Bit-Slip Mode Word Aligner in 8-Bit PMA-PCS Interface Modes

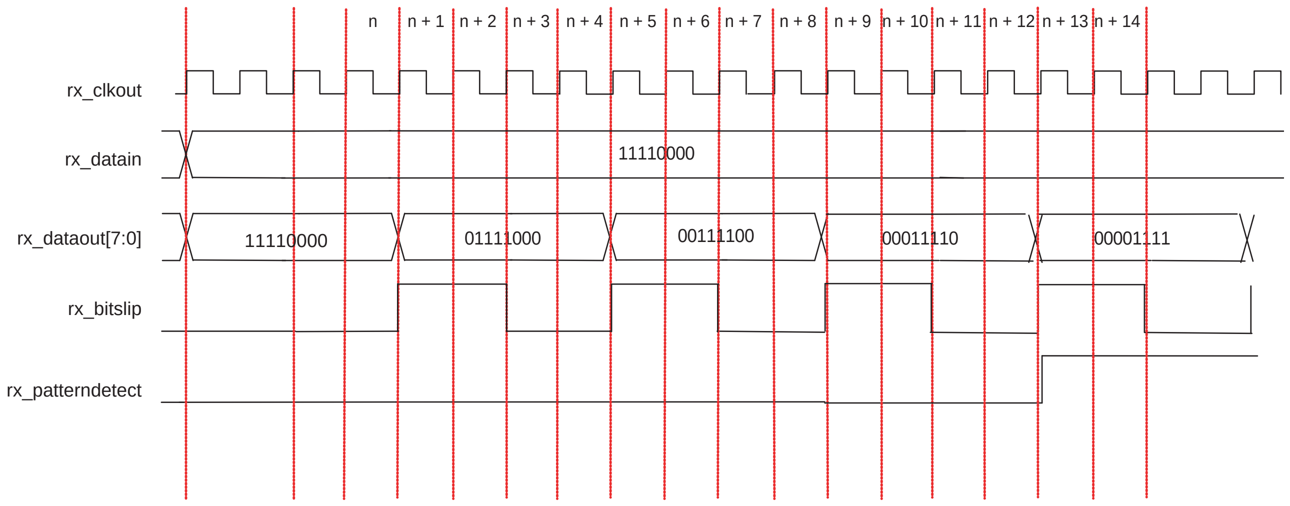
Basic single-width mode with 8-bit PMA-PCS interface width allows the word aligner to be configured in bit-slip mode. Word aligner operation is controlled by the input signal `rx_bitslip` in bit-slip mode. At every rising edge of the `rx_bitslip` signal, the bit-slip circuitry slips one bit into the received data stream, effectively shifting the word boundary by one bit. In bit-slip mode, the word aligner status signal `rx_patterndetect` is driven high for one parallel clock cycle when the received data after bit-slipping matches the 16-bit word alignment pattern programmed in the ALTGX MegaWizard Plug-In Manager.

You can implement a bit-slip controller in the FPGA fabric that monitors either the `rx_dataout` signal and/or the `rx_patterndetect` signal and controls the `rx_bitslip` signal to achieve word alignment.

Figure 1–44 shows an example of word aligner configured in bit-slip mode. For this example, consider that:

- 8'b11110000 is received back-to-back
- 16'b0000111100011110 is specified as the word alignment pattern
- A rising edge on the `rx_bitslip` signal at time n + 3 slips a single bit 0 at the MSB position, forcing the `rx_dataout` to 8'b01111000.
- Another rising edge on the `rx_bitslip` signal at time n + 5 forces `rx_dataout` to 8'b00111100.
- Another rising edge on the `rx_bitslip` signal at time n + 9 forces `rx_dataout` to 8'b00011110.
- Another rising edge on the `rx_bitslip` signal at time n + 13 forces the `rx_dataout` to 8'b00001111. At this instance, `rx_dataout` in cycles n + 12 and n + 13 are 8'b00011110 and 8'b00001111, respectively, which matches the specified 16-bit alignment pattern 16'b0000111100011110. This results in the assertion of the `rx_patterndetect` signal.

Figure 1–44. Example of Word Aligner Configured in Bit-Slip Mode



Word Aligner in Single-Width Mode with 10-Bit PMA-PCS Interface Modes

The following functional modes support the 10-bit PMA-PCS interface:

- PCI Express (PIPE) Gen1
- Serial RapidIO
- XAUI
- GIGE
- SDI
- Basic single-width mode

This section covers the following word aligner 10-bit PMA-PCS interface modes:

- Automatic synchronization state machine mode in 10-bit PMA-PCS interface mode
- Manual alignment mode in 10-bit PMA-PCS interface mode
- Bit-slip mode in 10-bit PMA-PCS interface mode

Table 1–19 shows the word aligner configurations allowed in functional modes with a 10-bit wide PMA-PCS interface.

Table 1–19. Word Aligner Configurations in a 10-Bit Wide PMA-PCS Interface

Functional Mode	Allowed Word Aligner Configurations	Allowed Word Alignment Pattern Length
PCI express (PIPE)	Automatic Synchronization State Machine	10 bits
Serial RapidIO	Automatic Synchronization State Machine	10 bits
XAUI	Automatic Synchronization State Machine	10 bits
GIGE	Automatic Synchronization State Machine	10 bits
SDI	Bit-Slip	N/A
Basic single-width mode	Manual Alignment, Automatic Synchronization State Machine, Bit-Slip	7 bits, 10 bits

Automatic Synchronization State Machine Mode Word Aligner in 10-Bit PMA-PCS Interface Mode

Protocols like PCI Express (PIPE), XAUI, Gigabit Ethernet, and Serial RapidIO require the receiver PCS logic to implement a synchronization state machine to provide hysteresis during link synchronization. Each of these protocols defines a specific number of synchronization code groups that the link must receive in order to acquire synchronization and a specific number of erroneous code groups that it must receive to fall out of synchronization.

In PCI Express (PIPE), XAUI, Gigabit Ethernet, and Serial RapidIO functional modes, the Quartus II software configures the word aligner in automatic synchronization state machine mode. It automatically selects the word alignment pattern length and pattern as specified by each protocol. In each of these functional modes, the protocol-compliant synchronization state machine is implemented in the word aligner.

In Basic single-width functional mode with a 10-bit PMA-PCS interface, you can configure the word aligner in automatic synchronization state machine mode by selecting the **Use the built-in synchronization state machine** option in the ALTGX MegaWizard Plug-In Manager. It also allows you to program a custom 7-bit or 10-bit word alignment pattern that the word aligner uses for synchronization.

☞ The 10-bit input data to the word aligner configured in automatic synchronization state machine mode must be 8B/10B encoded.

Table 1–20 shows the synchronization state machine parameters that the Quartus II software allows in supported functional modes. The synchronization state machine parameters are fixed for PCI Express (PIPE), XAUI, GIGE, and Serial RapidIO modes as specified by the respective protocol. For Basic single-width mode, you can program these parameters as suited to your proprietary protocol implementation.

Table 1–20. Synchronization State Machine Functional Modes

Functional Mode	PCI Express (PIPE)	XAUI	GIGE	Serial RapidIO	Basic Single-Width Mode
Number of valid synchronization code groups or ordered sets received to achieve synchronization	4	4	3	127	1—256
Number of erroneous code groups received to lose synchronization	17	4	4	3	1—64
Number of continuous good code groups received to reduce the error count by one	16	4	4	255	1—256

After de-assertion of the `rx_digitalreset` signal in automatic synchronization state machine mode, the word aligner starts looking for the word alignment pattern or synchronization code groups in the received data stream. When the programmed number of valid synchronization code groups or ordered sets is received, the `rx_syncstatus` signal is driven high to indicate that synchronization is acquired. The `rx_syncstatus` signal is constantly driven high until the programmed number of erroneous code groups is received without receiving intermediate good groups; after which `rx_syncstatus` is driven low. The word aligner indicates loss of synchronization (`rx_syncstatus` remains low) until the programmed number of valid synchronization code groups are received again.

Manual Alignment Mode Word Aligner in 10-Bit PMA-PCS Interface Mode

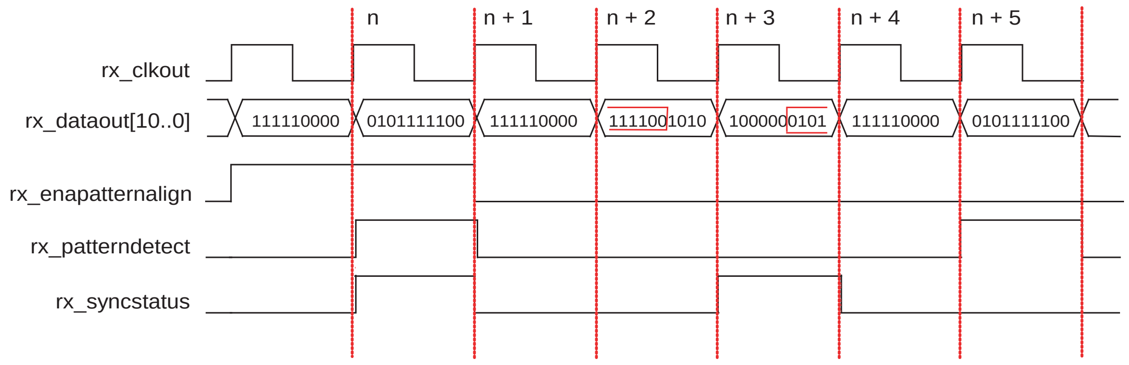
In Basic single-width mode with a 10-bit PMA-PCS interface, you can configure the word aligner in manual alignment mode by selecting the **Use manual word alignment mode** option in the ALTGX MegaWizard Plug-In Manager.

In manual alignment mode, word aligner operation is controlled by the input signal `rx_enapatternalign`. Word aligner operation is level-sensitive to the `rx_enapatternalign` signal. If the `rx_enapatternalign` signal is held high, the word aligner looks for the programmed 7-bit or 10-bit word alignment pattern in the received data stream. It updates the word boundary if it finds the word alignment pattern in a new word boundary. If the `rx_enapatternalign` signal is de-asserted low, the word aligner maintains the current word boundary even when it sees the word alignment pattern in a new word boundary.

Two status signals, `rx_syncstatus` and `rx_patterndetect`, both with the same latency as the datapath, are forwarded to the FPGA fabric to indicate word aligner status. On receiving the first word alignment pattern after the `rx_enapatternalign` signal is asserted high, both the `rx_syncstatus` and `rx_patterndetect` signals are driven high for one parallel clock cycle. Any word alignment pattern received thereafter in the same word boundary causes only the `rx_patterndetect` signal to go high for one clock cycle. Any word alignment pattern received thereafter in a different word boundary causes the word aligner to update the word boundary and the `rx_syncstatus` signal to go high for one clock cycle.

Figure 1–45 shows the manual alignment mode word aligner operation in 10-bit PMA-PCS interface mode. In this example, /K28.5/ (10'b0101111100) is specified as the word alignment pattern. The word aligner aligns to the /K28.5/ alignment pattern in cycle n because the `rx_enapatternalign` signal is asserted high. The `rx_syncstatus` signal goes high for one clock cycle, indicating alignment to a new word boundary. The `rx_patterndetect` signal also goes high for one clock cycle to indicate initial word alignment. At time n + 1, the `rx_enapatternalign` signal is de-asserted to instruct the word aligner to lock the current word boundary. The alignment pattern is detected again in a new word boundary across cycles n + 2 and n + 3. The `rx_patterndetect` signal remains low and `rx_syncstatus` goes high for one clock cycle in cycle n + 3 to indicate that the word alignment pattern is found in a new word boundary. The word aligner does not align to this new word boundary because the `rx_enapatternalign` signal is held low.

Figure 1–45. Word Aligner in 10-Bit PMA-PCS Manual Alignment Mode



If the word alignment pattern is known to be unique and does not appear between word boundaries, you can constantly hold the `rx_enapatternalign` signal high because there is no possibility of false word alignment. If there is a possibility of the word alignment pattern occurring across word boundaries, you must control the `rx_enapatternalign` signal to lock the word boundary after the desired word alignment is achieved to avoid re-alignment to an incorrect word boundary.

Bit-Slip Mode Word Aligner in 10-Bit PMA-PCS Interface Mode

In some Basic single-width configurations with 10-bit PMA-PCS interface mode, you can configure the word aligner in bit-slip mode by selecting the **Use manual bit slipping mode** option in the ALTGX MegaWizard Plug-In Manager.

The word aligner operation for Basic single-width with 10-bit PMA-PCS interface mode is similar to the word aligner operation in Basic single-width mode with 8-bit PMA-PCS interface mode. For word aligner operation in bit-slip mode, refer to "Manual Alignment Mode Word Aligner in 8-Bit PMA-PCS Interface Modes" on page 1–59. The only difference is that the bit-slip word aligner in 10-bit PMA-PCS interface mode allows 7-bit and 10-bit word alignment patterns, whereas the bit-slip word aligner in 8-bit PMA-PCS interface mode allows only 16-bit word alignment pattern.

Programmable Run Length Violation Detection

The programmable run length violation circuit resides in the word aligner block and detects consecutive 1s or 0s in the data. If the data stream exceeds the preset maximum number of consecutive 1s or 0s, the violation is signified by the assertion of the `rx_rlv` signal.

The run length violation status signal on the `rx_rlv` port has lower latency compared to the parallel data on the `rx_dataout` port. The `rx_rlv` signal in each channel is clocked by its parallel recovered clock. The FPGA fabric clock might have phase difference, PPM difference (in asynchronous systems), or both, with respect to the recovered clock. To ensure that the FPGA fabric clock can latch the `rx_rlv` signal reliably, the run length violation circuitry asserts the `rx_rlv` signal for a minimum of two recovered clock cycles in single-width mode. The `rx_rlv` signal can be asserted longer, depending on the run length of the received data.

In single-width mode, the run length violation circuit detects up to a run length of 128 (for an 8-bit deserialization factor) or 160 (for a 10-bit deserialization factor). The settings are in increments of 4 or 5 for the 8-bit or 10-bit deserialization factors, respectively.

Table 1–21 summarizes the detection capabilities of the run length violation circuit.

Table 1–21. Detection Capabilities of the Run Length Violation Circuit

Mode	PMA-PCS Interface Width	Run Length Violation Detector Range	
		Minimum	Maximum
Single-width mode	8-bit	4	128
	10-bit	5	160

Receiver Polarity Inversion

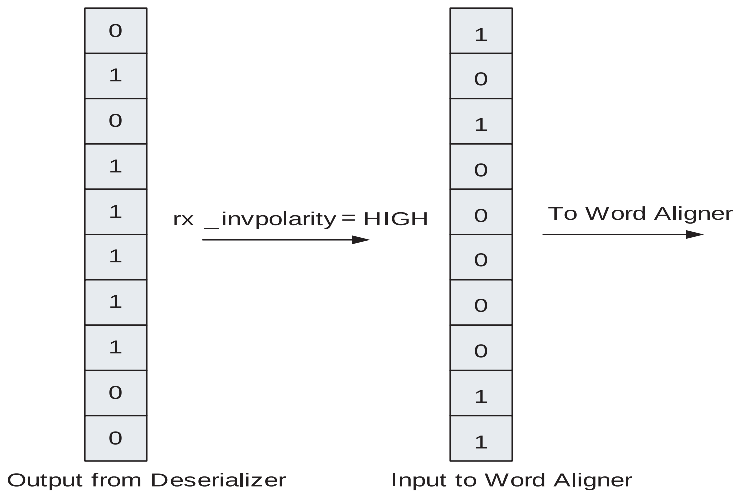
The positive and negative signals of a serial differential link are often erroneously swapped during board layout. Solutions like board re-spin or major updates to the PLD logic can be expensive. The receiver polarity inversion feature is provided to correct this situation.

An optional `rx_invpolarity` port is available in all single-width modes except PCI Express (PIPE) to dynamically enable the receiver polarity inversion feature. In single-width modes, a high value on the `rx_invpolarity` port inverts the polarity of every bit of the 8-bit or 10-bit input data word to the word aligner in the receiver data path. Because inverting the polarity of each bit has the same effect as swapping the positive and negative signals of the differential link, correct data is seen by the receiver. The `rx_invpolarity` signal is dynamic and can cause initial disparity errors in an 8B/10B encoded link. The downstream system must be able to tolerate these disparity errors.

The generic receiver polarity inversion feature is different from the PCI Express (PIPE) 8B/10B polarity inversion feature. The generic receiver polarity inversion feature inverts the polarity of the data bits at the input of the word aligner and is not available in PCI Express (PIPE) mode. The PCI Express (PIPE) 8B/10B polarity inversion feature inverts the polarity of the data bits at the input of the 8B/10B decoder and is available only in PCI Express (PIPE) mode.

Figure 1–46 shows the receiver polarity inversion feature in single-width 10-bit wide data path configurations.

Figure 1–46. 10-Bit Receiver Polarity Inversion in Single-Width Mode



Receiver Bit Reversal

By default, the Arria II GX receiver assumes a LSBit-to-MSBit transmission. If the transmission order is MSBit-to-LSBit, the receiver forwards the bit-flipped version of the parallel data to the FPGA fabric on the `rx_dataout` port. The receiver bit reversal feature is available to correct this situation.

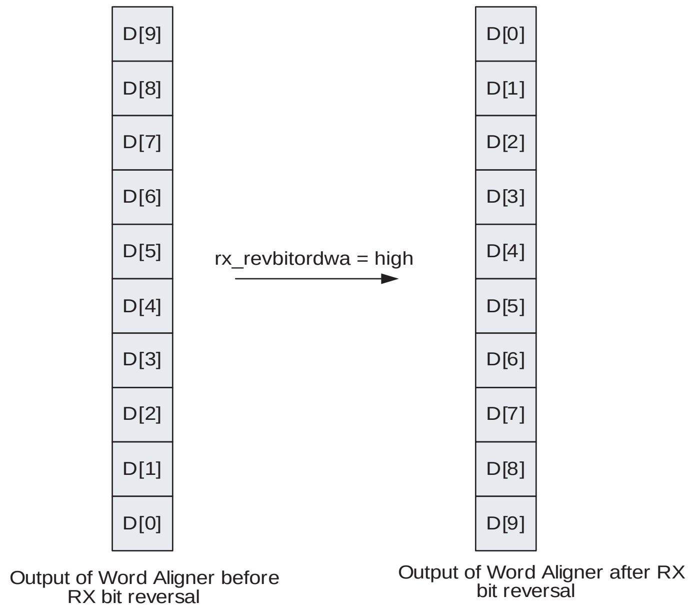
The receiver bit reversal feature is available through the `rx_revbitordwa` port only in Basic single-width mode with the word aligner configured in bit-slip mode.

- When the `rx_revbitordwa` signal is driven high in Basic single-width mode, the 8-bit or 10-bit data `D[7:0]` or `D[9:0]` at the output of the word aligner gets rewired to `D[0:7]` or `D[0:9]`, respectively.

Flipping the parallel data using this feature allows the receiver to forward the correct bit-ordered data to the FPGA fabric on the `rx_dataout` port in the case of MSBit-to-LSBit transmission.

Figure 1–47 shows the receiver bit reversal feature in Basic single-width 10-bit wide data path configurations.

Figure 1–47. 10-Bit Receiver Bit Reversal in Single-Width Mode



Deskew FIFO

Code groups received across four lanes in a XAUI link can be misaligned with respect to one another because of skew in the physical medium or differences between the independent clock recoveries per lane. The XAUI protocol allows a maximum skew of 40 UI (12.8 ns) as seen at the receiver of the four lanes.

The XAUI protocol requires the physical layer device to implement deskew circuitry to align all four channels. To enable the deskew circuitry at the receiver to align the four channels, the transmitter sends a /A/ (/K28.3/) code group simultaneously on all four channels during inter-packet gap (IPG). The skew introduced in the physical medium and the receiver channels can cause the /A/ code groups to be received misaligned with respect to each other.

Deskew operation is performed by the deskew circuitry in XAUI functional mode. Deskew circuitry consists of:

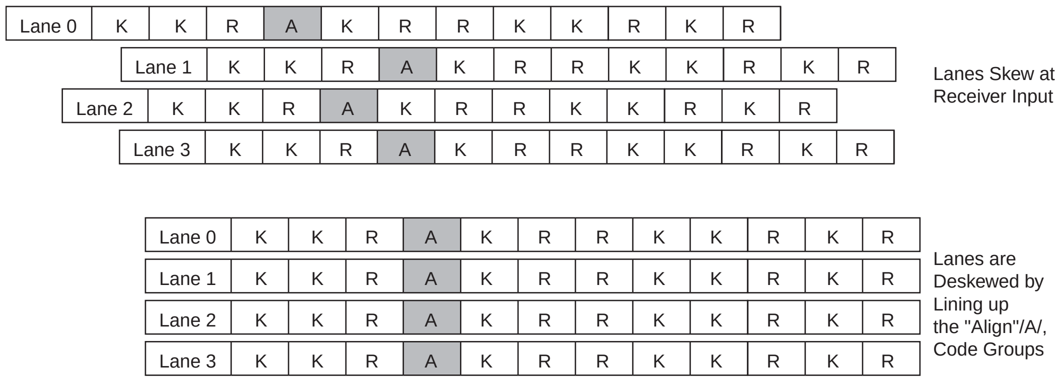
- A 16-word deep deskew FIFO in each of the four channels
- Control logic in the CMU0 channel of the transceiver block that controls the deskew FIFO write and read operations in each channel

☞ Deskew circuitry is only available in XAUI mode.

The deskew FIFO in each channel receives data from its word aligner. The deskew operation begins only after link synchronization is achieved on all four channels as indicated by a high level on the `rx_syncstatus` signal from the word aligner in each channel. Until the first /A/ code group is received, the deskew FIFO read and write pointers in each channel are not incremented. After the first /A/ code group is received, the write pointer starts incrementing for each word received but the read pointer is frozen. If the /A/ code group is received on each of the four channels in 10 recovered clock cycles of each other, the read pointer of all four deskew FIFOs is released simultaneously, aligning all four channels.

Figure 1–48 shows lane skew at the receiver input and how the deskew FIFO uses the /A/ code group to align the channels.

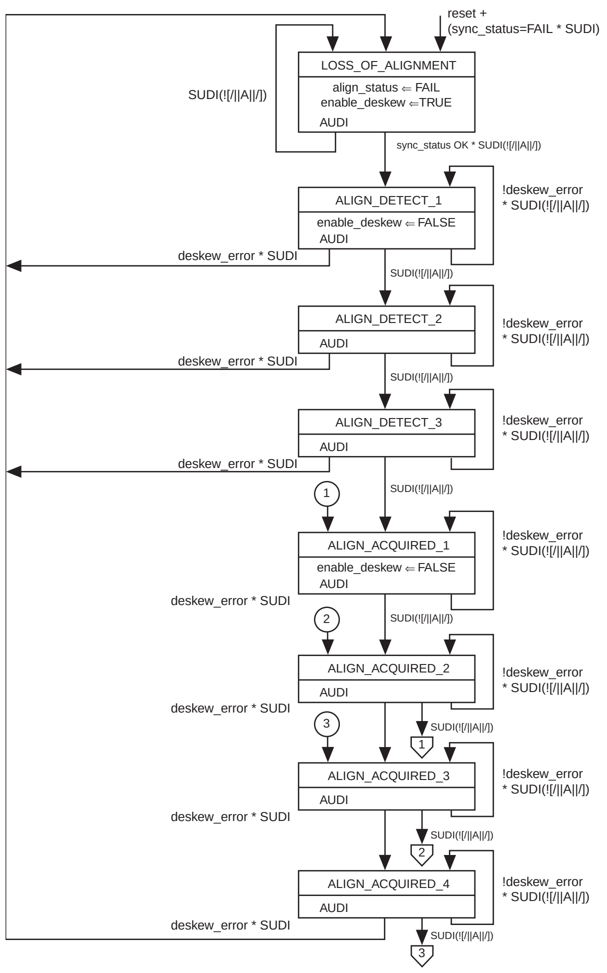
Figure 1–48. Deskew FIFO—Lane Skew at the Receiver Input


Lane 0 K K R A K R R K K R K R
Lane 1 K K R A K R R K K R K R
Lane 2 K K R A K R R K K R K R
Lane 3 K K R A K R R K K R K R
Lanes Skew at Receiver Input
Lane 0 K K R A K R R K K R K R
Lane 1 K K R A K R R K K R K R
Lane 2 K K R A K R R K K R K R
Lane 3 K K R A K R R K K R K R
Lanes are Deskewed by Lining up the "Align"/A/, Code Groups

After alignment of the first ||A|| column, if three additional aligned ||A|| columns are observed at the output of the deskew FIFOs of the four channels, the `rx_channelaligned` signal is asserted high, indicating channel alignment is acquired. After acquiring channel alignment, if four misaligned ||A|| columns are seen at the output of the deskew FIFOs in all four channels with no aligned ||A|| columns in between, the `rx_channelaligned` signal is de-asserted low, indicating loss of channel alignment.

Deskew operation in XAUI functional mode is compliant to the PCS deskew state machine diagram specified in clause 48 of the IEEE P802.3ae, as shown in Figure 1–49.

Figure 1–49. Deskew FIFO Operation in XAUI Functional Mode *(Note 1)*



Note to Figure 1–49:

(1) Source: IEEE P802.3ae™-2002, IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.

Rate Match (Clock Rate Compensation) FIFO

In asynchronous systems, the upstream transmitter and local receiver can be clocked with independent reference clocks. Frequency differences in the order of a few hundred PPM can corrupt the data when latching from the recovered clock domain (the same clock domain as the upstream transmitter reference clock) to the local receiver reference clock domain.

The rate match FIFO compensates for small clock frequency differences between the upstream transmitter and the local receiver clocks by inserting or removing SKP symbols or ordered-sets from the IPG or idle streams. It deletes SKP symbols or ordered-sets when the upstream transmitter reference clock frequency is higher than the local receiver reference clock frequency. It inserts SKP symbols or ordered-sets when the local receiver reference clock frequency is higher than the upstream transmitter reference clock frequency.

Rate match FIFO consists of a 20-word deep FIFO and necessary logic that controls insertion and deletion of a SKP character or ordered-set, depending on the PPM difference.

Rate match FIFO is mandatory and cannot be bypassed in the following functional modes:

- PCI Express (PIPE)
- XAUI
- GIGE

Rate match FIFO is optional in the following functional modes:

- Basic single-width

Rate match FIFO receives data from the word aligner (non-XAUI functional modes) or deskew FIFO (XAUI functional mode) in the receiver datapath. It provides the following status signals forwarded to the FPGA fabric:

- `rx_rmfifodatainserted`—indicates insertion of a skip character or ordered-set
- `rx_rmfifodatadeleted`—indicates deletion of a skip character or ordered-set
- `rx_rmfifofull`—indicates rate match FIFO full condition
- `rx_rmfifoempty`—indicates rate match FIFO empty condition

Rate match FIFO status signals are not available in PCI Express (PIPE) mode. These signals are encoded on the `pipestatus[2:0]` signal in PCI Express (PIPE) mode, as specified in the PIPE specification.

Rate Match FIFO in PCI Express (PIPE) Mode

In PCI Express (PIPE) mode, the rate match FIFO is capable of compensating up to ± 300 PPM (total 600 PPM) difference between the upstream transmitter and the local receiver. The PCI Express (PIPE) protocol requires the transmitter to send SKP ordered sets during IPGs, adhering to rules listed in the base specification. The SKP ordered set is defined as a /K28.5/ COM symbol followed by three consecutive /K28.0/ SKP symbol groups. The PCI Express (PIPE) protocol requires the receiver to recognize a SKP ordered set as a /K28.5/ COM symbol followed by one-to-five consecutive /K28.0/ SKP symbols.

Rate match FIFO operation is compliant to PCI Express Base Specification 1.1. The rate match operation begins after the synchronization state machine in the word aligner indicates synchronization is acquired by driving the `rx_syncstatus` signal high. The rate match FIFO looks for the SKP ordered set and deletes or inserts SKP symbols as necessary to prevent the rate match FIFO from overflowing or under running.

The rate match FIFO inserts or deletes only one SKP symbol per SKP ordered set received. Rate match FIFO insertion and deletion events are communicated to the FPGA fabric on the `pipestatus[2:0]` port from each channel. The `pipestatus[2:0]` signal is driven to 3'b001 for one clock cycle synchronous to the /K28.5/ COM symbol of the SKP ordered set in which the /K28.0/ SKP symbol is inserted. The `pipestatus[2:0]` signal is driven to 3'b010 for one clock cycle synchronous to the /K28.5/ COM symbol of the SKP ordered set from which the /K28.0/ SKP symbol is deleted.

Figure 1–50 shows an example of rate match deletion in the case where two /K28.0/ SKP symbols are required to be deleted. Only one /K28.0/ SKP symbol is deleted per SKP ordered set received.

Figure 1–50. Example of Rate Match Deletion in PCI Express (PIPE) Mode

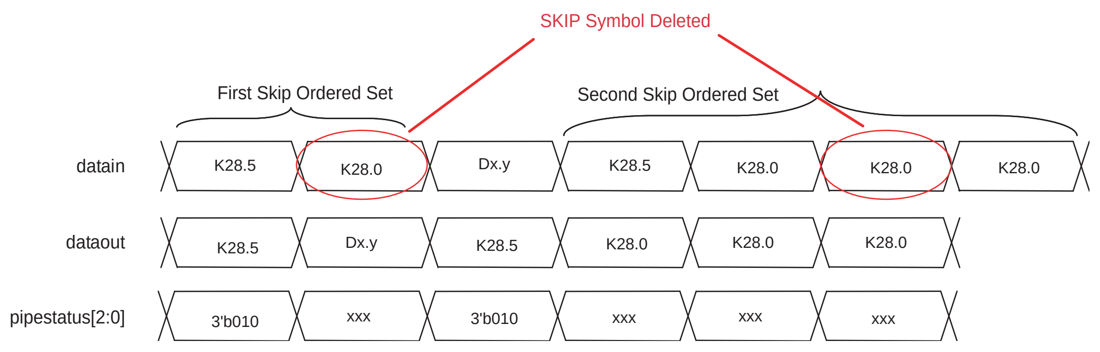
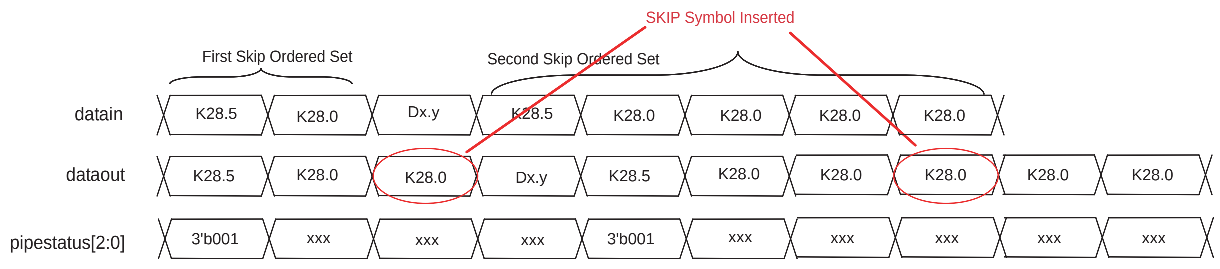


Figure 1–51 shows an example of rate match insertion in the case where two SKP symbols are required to be inserted. Only one /K28.0/ SKP symbol is inserted per SKP ordered set received.

Figure 1–51. Example of Rate Match Insertion in PCI Express (PIPE) Mode

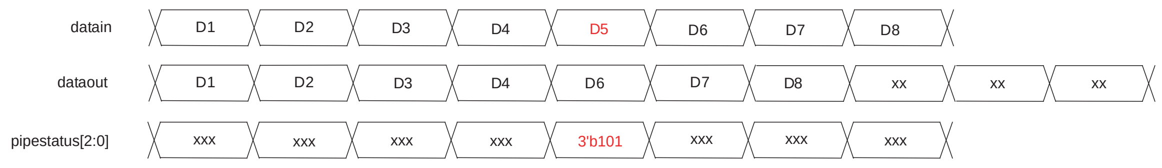


Rate match FIFO full and empty conditions are communicated to the FPGA fabric on the `pipestatus[2:0]` port from each channel.

Rate match FIFO in PCI Express (PIPE) mode automatically deletes the data byte that causes the FIFO to go full and drives `pipestatus[2:0] = 3'b101` synchronous to the subsequent data byte.

Figure 1–52 shows the rate match FIFO full condition in PCI Express (PIPE) mode. The rate match FIFO becomes full after receiving data byte D4.

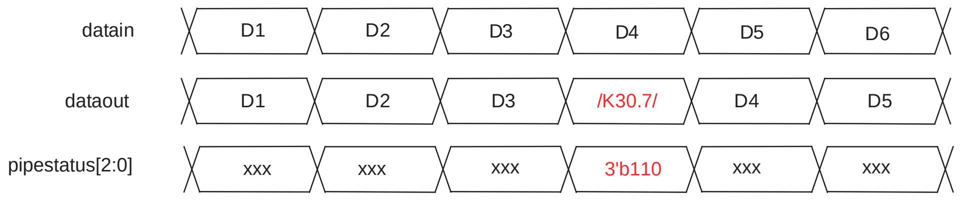
Figure 1–52. Example of Rate Match FIFO Full Condition in PCI Express (PIPE) Mode



The rate match FIFO automatically inserts /K30.7/ (9'h1FE) after the data byte that causes the FIFO to go empty and drives the `pipestatus[2:0] = 3’b110` flag synchronous to the inserted /K30.7/ (9'h1FE).

Figure 1–53 shows rate match FIFO empty condition in PCI Express (PIPE) mode. The rate match FIFO becomes empty after reading out data byte D3.

Figure 1–53. Example of Rate Match FIFO Empty Condition in PCI Express (PIPE) Mode



You can configure the rate match FIFO in low-latency mode by turning off the **Enable Rate Match FIFO** option in the ALTGX MegaWizard Plug-In Manager.

Rate Match FIFO in XAUI Mode

In XAUI mode, the rate match FIFO is capable of compensating up to ±100 PPM (total 200 PPM total) difference between the upstream transmitter and the local receiver reference clock. The XAUI protocol requires the transmitter to send /R/ (/K28.0/) code groups simultaneously on all four lanes (denoted as ||R|| column) during inter-packet gaps, adhering to rules listed in the IEEE P802.3ae specification. Rate match FIFO operation in XAUI mode is compliant to the IEEE P802.3ae specification.

Rate match operation begins after:

- The synchronization state machine in the word aligner of all four channels indicates synchronization was acquired by driving its `rx_syncstatus` signal high
- The deskew FIFO block indicates alignment was acquired by driving the `rx_channelaligned` signal high

Rate match FIFO looks for the ||R|| column (simultaneous /R/ code group on all four channels) and deletes or inserts ||R|| column to prevent the rate match FIFO from overflowing or under running. It can insert or delete as many ||R|| columns as necessary to perform the rate match operation.

Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, that indicate rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric. If an ||R|| column is deleted, the `rx_rmfifodeleted` flag from each of the four channels goes high for one clock cycle per deleted ||R|| column. If an ||R|| column is inserted, the `rx_rmfifoinserted` flag from each of the four channels goes high for one clock cycle per inserted ||R|| column.

Figure 1–54 shows an example of rate match deletion in the case where three ||R|| columns are required to be deleted.

Figure 1–54. Example of Rate Match Deletion in XAUI Mode

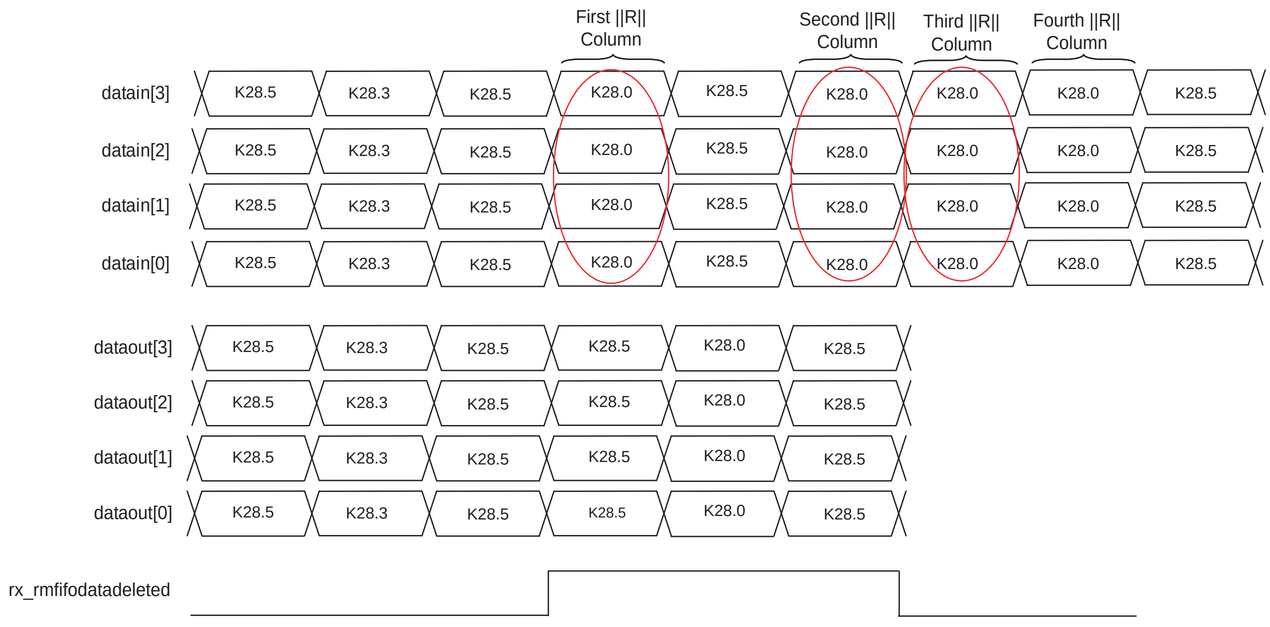
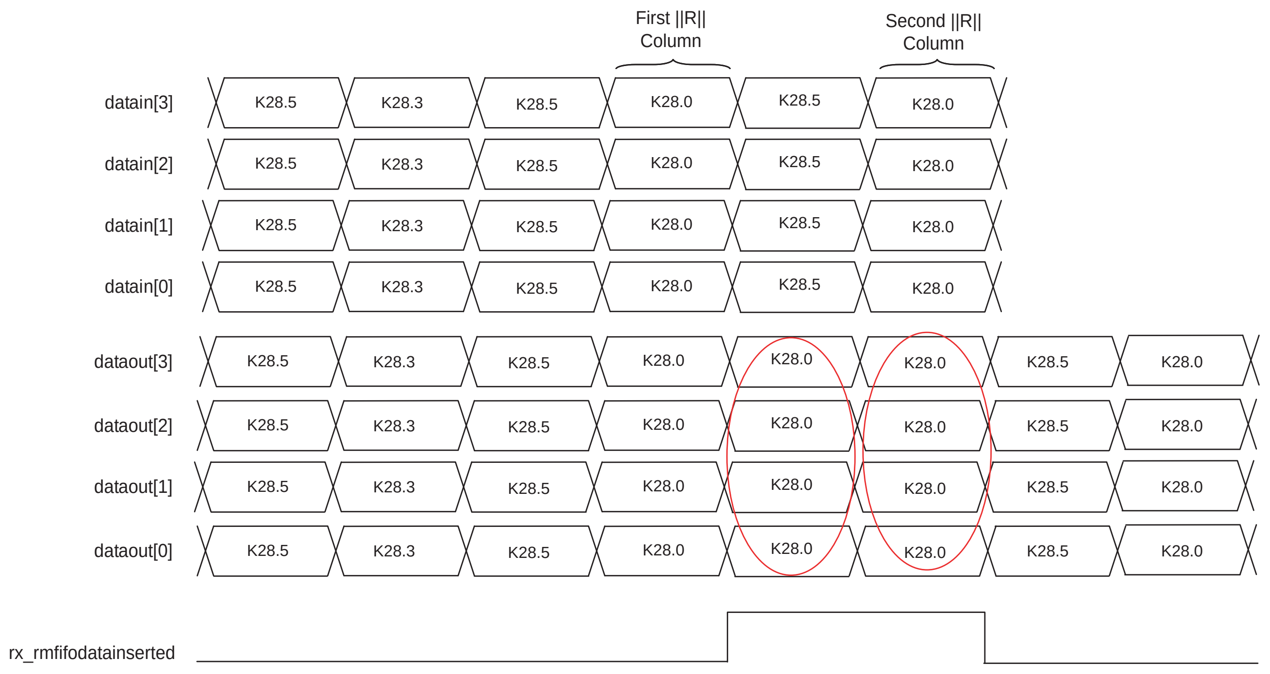


Figure 1–55 shows an example of rate match insertion in the case where two ||R|| columns are required to be inserted.

Figure 1–55. Example of Rate Match Insertion in XAUI Mode



Two flags, `rx_rmfifofull` and `rx_rmfifoempty`, are forwarded to the FPGA fabric to indicate rate match FIFO full and empty conditions.

In XAUI mode, the rate match FIFO does not automatically insert or delete code groups to overcome FIFO empty and full conditions, respectively. It asserts the `rx_rmfifofull` and `rx_rmfifoempty` flags for at least three recovered clock cycles to indicate rate match FIFO full and empty conditions, respectively.

In case of rate match FIFO full and empty conditions, you must assert the `rx_digitalreset` signal to reset the receiver PCS blocks.

Rate Match FIFO in GIGE Mode

In GIGE mode, the rate match FIFO is capable of compensating up to ±100 PPM (total 200 PPM total) difference between the upstream transmitter and the local receiver reference clock. The GIGE protocol requires the transmitter to send idle ordered sets /I1/ (/K28.5/D5.6/) and /I2/ (/K28.5/D16.2/) during inter-packet gaps, adhering to rules listed in the IEEE 802.3 specification.

Rate match operation begins after the synchronization state machine in the word aligner indicates synchronization is acquired by driving the `rx_syncstatus` signal high. The rate match FIFO is capable of deleting or inserting the /I2/ (/K28.5/D16.2/) ordered set to prevent the rate match FIFO from overflowing or under running during normal packet transmission. The rate match FIFO is also capable of deleting or inserting the first two bytes of the /C2/ ordered set (/K28.5/D2.2/Dx.y/Dx.y/) to prevent the rate match FIFO from overflowing or under running during the auto negotiation phase.

Rate match FIFO can insert or delete as many /I2/ or /C2/ (first two bytes) as necessary to perform the rate match operation.

Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, that indicate rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric. Both the `rx_rmfifodatadeleted` and `rx_rmfifodatainserted` flags are asserted for two clock cycles for each deleted and inserted /I2/ ordered-set, respectively.

Figure 1–56 shows an example of rate match FIFO deletion in the case where three symbols are required to be deleted. Because the rate match FIFO can only delete /I2/ ordered-set, it deletes two /I2/ ordered-sets (four symbols deleted).

Figure 1–56. Example of Rate Match Deletion in GIGE Mode

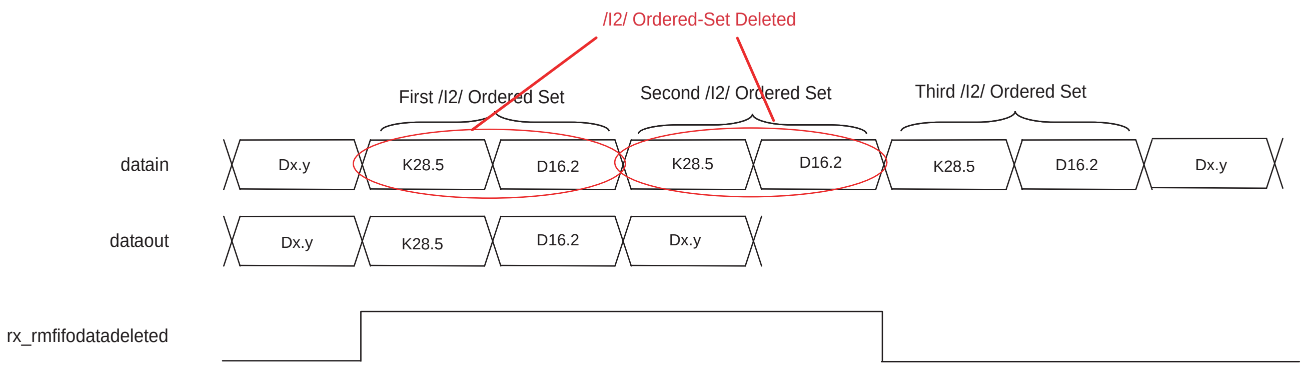
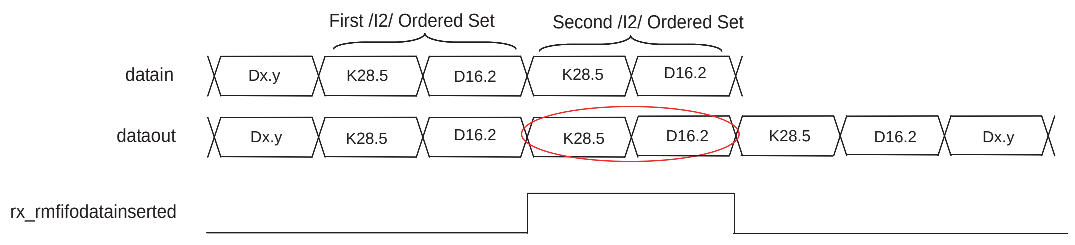


Figure 1–57 shows an example of rate match FIFO insertion in the case where one symbol is required to be inserted. Because the rate match FIFO can only delete /I2/ ordered-set, it inserts one /I2/ ordered-sets (two symbols inserted).

Figure 1–57. Example of Rate Match Insertion in GIGE Mode



Two flags, `rx_rmfifofull` and `rx_rmfifoempty`, are forwarded to the FPGA fabric to indicate rate match FIFO full and empty conditions.

In GIGE mode, the rate match FIFO does not insert or delete code groups automatically to overcome FIFO empty and full conditions, respectively. It asserts the `rx_rmfifofull` and `rx_rmfifoempty` flags for at least two recovered clock cycles to indicate rate match FIFO full and empty conditions, respectively.

In case of rate match FIFO full and empty conditions, you must assert the `rx_digitalreset` signal to reset the receiver PCS blocks.

Rate Match FIFO in Basic Single-Width Mode

In Basic single-width mode, the rate match FIFO is capable of compensating up to ±300 PPM (total 600 PPM total) difference between the upstream transmitter and the local receiver reference clock.

To enable the rate match FIFO in Basic single-width mode, the transceiver channel must have both the transmitter and receiver channels instantiated. You must select the **Receiver and Transmitter** option in the **What is the operation mode?** field in the ALTGX MegaWizard Plug-In Manager. You must also enable the 8B/10B encoder/decoder in Basic single-width mode with rate match FIFO enabled.

Depending on your proprietary protocol implementation, you can select two 20-bit rate match patterns in the ALTGX MegaWizard Plug-In Manager under the **What is the rate match pattern1** and **What is the rate match pattern2** fields. Each of the two programmed 20-bit rate match patterns consists of a 10-bit skip pattern and a 10-bit control pattern. You must choose 10-bit code groups that have a neutral disparity as the skip patterns. The rate match FIFO operation begins after the word aligner synchronization status `rx_syncstatus` goes high. When the rate matcher receives either of the two 10-bit control patterns followed by the respective 10-bit skip pattern, it inserts or deletes the 10-bit skip pattern as necessary to avoid the rate match FIFO from overflowing or under running.

Rate match FIFO can delete a maximum of four skip patterns from a cluster, provided there is one skip pattern left in the cluster after deletion. The rate match FIFO can insert a maximum of four skip patterns in a cluster, provided there are no more than five skip patterns in the cluster after insertion. Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, indicating rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric.

Figure 1–58 shows an example of rate match FIFO deletion in the case where three skip patterns are required to be deleted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern followed by two /K28.0/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by four /K28.0/ skip patterns. The rate match FIFO deletes only one /K28.0/ skip pattern from the first skip cluster to maintain at least one skip pattern in the cluster after deletion. Two /K28.0/ skip patterns are deleted from the second cluster for a total of three skip patterns deleted.

Figure 1–58. Example of Rate Match Deletion in Basic Single-Width Mode

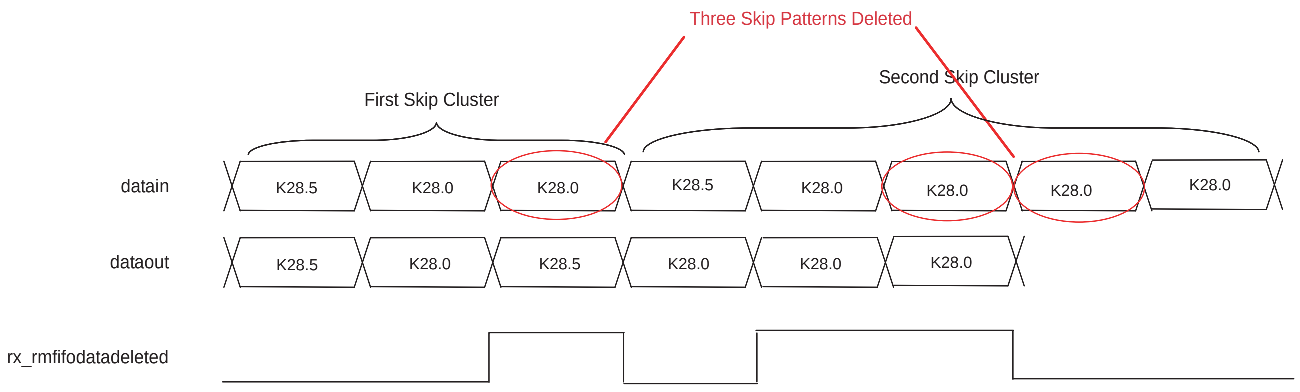
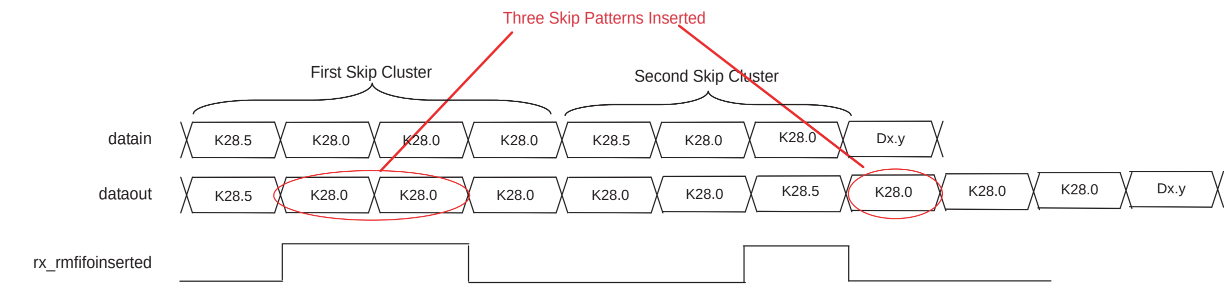


Figure 1–59 shows an example of rate match FIFO insertion in the case where three skip patterns are required to be inserted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern followed by three /K28.0/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by one /K28.0/ skip pattern. The rate match FIFO inserts only two /K28.0/ skip patterns into the first skip cluster to maintain a maximum of five skip patterns in the cluster after insertion. One /K28.0/ skip pattern is inserted into the second cluster for a total of three skip patterns inserted.

Figure 1–59. Example of Rate Match Insertion in Basic Single-Width Mode

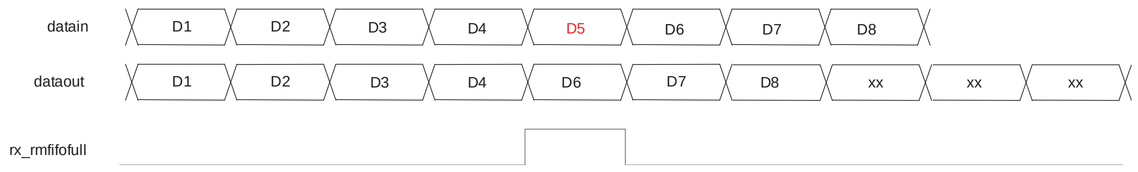


Two flags, `rx_rmfifofull` and `rx_rmfifoempty`, are forwarded to the FPGA fabric to indicate rate match FIFO full and empty conditions.

The rate match FIFO in Basic single-width mode automatically deletes the data byte that causes the FIFO to go full and asserts the `rx_rmfifofull` flag synchronous to the subsequent data byte.

Figure 1–60 shows the rate match FIFO full condition in Basic single-width mode. The rate match FIFO becomes full after receiving data byte D4.

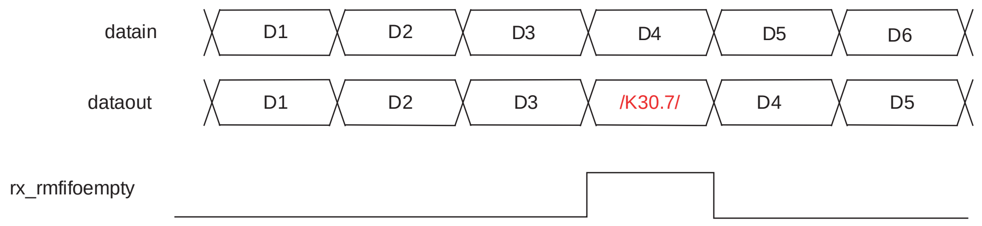
Figure 1–60. Rate Match FIFO Full Condition in Basic Single-Width Mode



The rate match FIFO automatically inserts /K30.7/ (9'h1FE) after the data byte that causes the FIFO to go empty and asserts the `rx_fifoempty` flag synchronous to the inserted /K30.7/ (9'h1FE).

Figure 1–61 shows rate match FIFO empty condition in Basic single-width mode. The rate match FIFO becomes empty after reading out data byte D3.

Figure 1–61. Rate Match FIFO Empty Condition in Basic Single-Width Mode



8B/10B Decoder

Protocols like PCI Express (PIPE), XAUI, GIGE, and Serial RapidIO require the serial data sent over the link to be 8B/10B encoded to maintain the DC balance in the serial data transmitted. These protocols require the receiver PCS logic to implement an 8B/10B decoder to decode the data before forwarding it to the upper layers for packet processing.

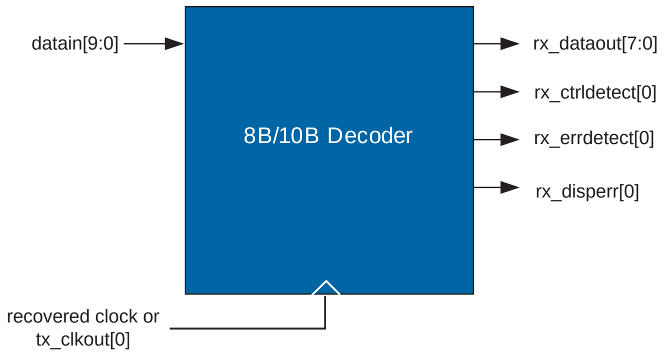
The Arria II GX receiver channel PCS datapath implements the 8B/10B decoder after the rate matcher. In functional modes with rate matcher enabled, the 8B/10B decoder receives data from the rate matcher. In functional modes with rate matcher disabled, the 8B/10B decoder receives data from the word aligner.

The Arria II GX 8B/10B decoder operates in single-width mode only.

8B/10B Decoder in Single-Width Mode

Figure 1–62 shows the block diagram of the 8B/10B decoder in single-width mode.

Figure 1–62. 8B/10B Decoder in Single-Width Mode



In single-width mode, the 8B/10B decoder receives 10-bit data from the rate matcher or word aligner (when rate matcher is disabled) and decodes it into an 8-bit data + 1-bit control identifier. The decoded data is fed to the byte deserializer or the receiver phase compensation FIFO (if byte deserializer is disabled).

☞ The 8B/10B decoder is compliant to Clause 36 in the IEEE802.3 specification.

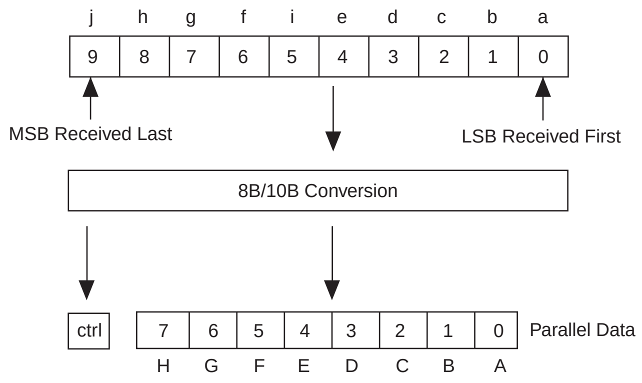
The 8B/10B decoder operates in single-width mode in the following functional modes:

- PCI Express (PIPE)
- XAUI
- GIGE
- Serial RapidIO
- Basic single-width

For PCI Express (PIPE), XAUI, GIGE, and Serial RapidIO functional modes, the ALTGX MegaWizard Plug-In Manager forces selection of the 8B/10B decoder in the receiver datapath. In Basic single-width mode, it allows you to enable or disable the 8B/10B decoder depending on your proprietary protocol implementation.

Figure 1–63 shows a 10-bit code group decoded into an 8-bit data and a 1-bit control identifier by the 8B/10B decoder in single-width mode.

Figure 1–63. 8B/10B Decoder in Single-Width Mode

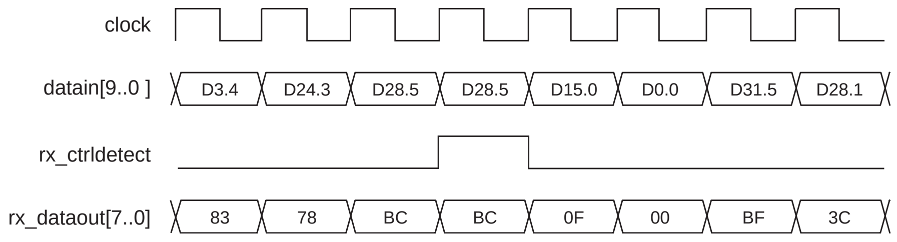


Control Code Group Detection

The 8B/10B decoder indicates whether the decoded 8-bit code group is a data or control code group on the `rx_ctrldetect` port. If the received 10-bit code group is one of the 12 control code groups (/Kx.y/) specified in the IEEE802.3 specification, the `rx_ctrldetect` signal is driven high. If the received 10-bit code group is a data code group (/Dx.y/), the `rx_ctrldetect` signal is driven low.

Figure 1–64 illustrates the 8B/10B decoder decoding the received 10-bit /K28.5/ control code group into an 8-bit data code group (8'hBC) driven on the `rx_dataout` port. `rx_ctrldetect` is asserted high synchronous with 8'hBC on the `rx_dataout` port, indicating that it is a control code group. The rest of the codes received are data code groups /Dx.y/.

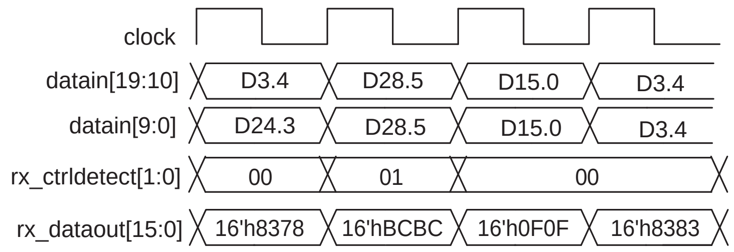
Figure 1–64. 8B/10B Decoder in Control Code Group Detection



Control Code Group Detection

Figure 1–65 shows 8B/10B decoding of the received 10-bit /K28.5/ control code group into 8-bit data code group (8'hBC) driven on the `rx_dataout` port. The `rx_ctrldetect` is asserted high synchronous with 8'hBC on the `rx_dataout` port, indicating that it is a control code group. The rest of the codes received are data code groups /Dx.y/.

Figure 1–65. 8B/10B Decoder 10-Bit Control Code Group



Byte Deserializer

The FPGA fabric-transceiver interface frequency has an upper limit of 200 MHz. In functional modes that have a receiver PCS frequency greater than 200 MHz, the parallel received data and status signals cannot be forwarded directly to the FPGA fabric because it violates the upper limit of the 200 MHz FPGA fabric—transceiver interface frequency. In such configurations, the byte deserializer is required to reduce the FPGA fabric—transceiver interface frequency to half while doubling the parallel data width. For example, at 3.2 Gbps data rate with a deserialization factor of 10, the receiver PCS datapath runs at 320 MHz. The 10-bit parallel received data and status signals at 320 MHz cannot be forwarded to the FPGA fabric because it violates the upper limit of 200 MHz. The byte serializer converts the 10-bit parallel received data at 320 MHz into 20-bit parallel data at 160 MHz before forwarding to the FPGA fabric.

The byte deserializer is required in configurations that exceed the FPGA fabric—transceiver interface clock upper frequency limit. It is optional in configurations that do not exceed the FPGA fabric—transceiver interface clock upper frequency limit.

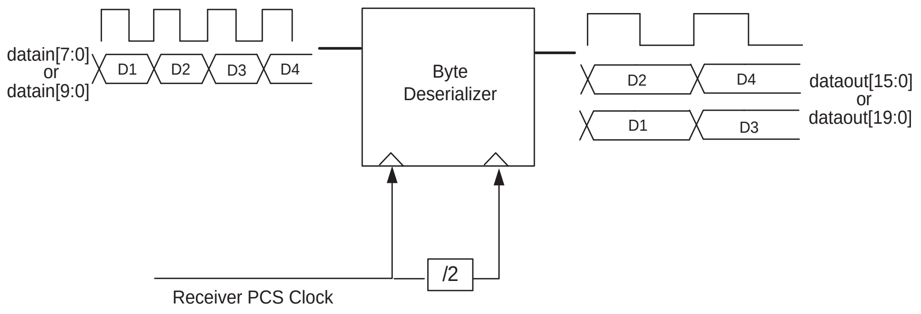
The Arria II GX byte deserializer operates in single-width mode only.

Byte Deserializer in Single-Width Mode

In single-width mode, the byte deserializer receives 8-bit wide data from the 8B/10B decoder or 10-bit wide data from the word aligner (if the 8B/10B decoder is disabled) and deserializes it into 16-bit or 20-bit wide data at half the speed.

Figure 1–66 shows the block diagram of the byte deserializer in single-width mode.

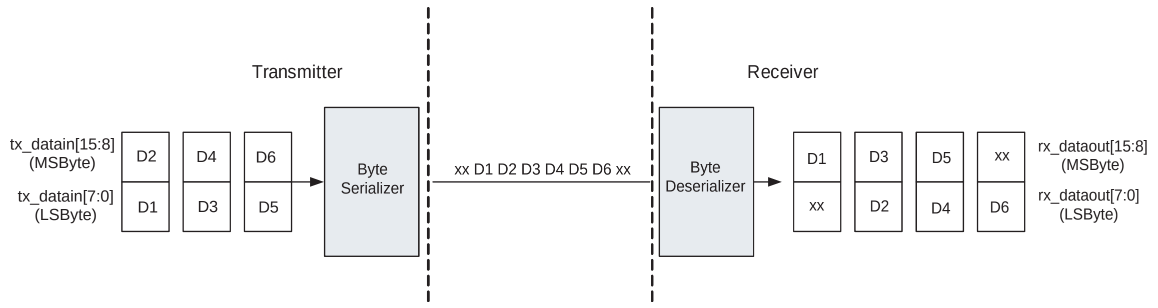
Figure 1–66. Byte Deserializer in Single-Width Mode



Byte Ordering Block

In single-width modes with the 16-bit or 20-bit FPGA fabric—transceiver interface, the byte deserializer receives one data byte (8- or 10-bit) and deserializes it into two data bytes (16- or 20-bit). Depending on when the receiver PCS logic comes out of reset, the byte ordering at the output of the byte deserializer may or may not match the original byte ordering of the transmitted data. The byte misalignment resulting from byte deserialization is unpredictable because it depends on which byte is being received by the byte deserializer when it comes out of reset. Figure 1–67 shows a scenario in which the MSByte and LSByte of the two-byte transmitter data appears straddled across two word boundaries after getting byte deserialized at the receiver.

Figure 1–67. MSByte and LSByte of the Two-Byte Transmitter Data Straddled Across Two Word Boundaries



Arria II GX transceivers have an optional byte ordering block in the receiver data path that you can use to restore proper byte ordering before forwarding the data to the FPGA fabric. The byte ordering block looks for the user-programmed byte ordering pattern in the byte-deserialized data. You must select a byte ordering pattern that you know appears at the LSByte/LSBytes position of the parallel transmitter data. If the byte ordering block finds the programmed byte ordering pattern in the MSByte/MSBytes position of the byte-deserialized data, it inserts the appropriate number of user-programmed PAD bytes to push the byte ordering pattern to the LSByte/LSBytes position, thereby restoring proper byte ordering.

Byte Ordering Block in Single-Width Modes

The byte ordering block is available in the following single-width functional modes:

- SONET/SDH OC-48
- Basic single-width mode with:
  - 16-bit FPGA fabric—transceiver interface
  - No 8B/10B decoder (8-bit PMA-PCS interface)
  - Word aligner in manual alignment mode
- Basic single-width mode with:
  - 16-bit FPGA fabric—transceiver interface
  - 8B/10B decoder
  - Word aligner in automatic synchronization state machine mode

For more information about configurations that allow the byte ordering block in the receiver datapath, refer to "Basic Single-Width Mode Configurations" on page 1–87.

The Quartus II software automatically configures the byte ordering pattern and byte ordering PAD pattern for SONET/SDH OC-48 functional mode. For more information, refer to "OC-48 Byte Ordering" on page 1–126.

In Basic single-width mode, you can program a custom byte ordering pattern and byte ordering PAD pattern in the ALTGX MegaWizard Plug-In Manager. Table 1–22 shows the byte ordering pattern length allowed in Basic single-width mode.

Table 1–22. Byte Ordering Pattern Length in Basic Single-Width Mode

Functional Mode	Byte Ordering Pattern Length	Byte Ordering PAD Pattern Length
Basic single-width mode with: ■ 16-bit FPGA fabric-transceiver interface ■ no 8B/10B decoder ■ Word aligner in manual alignment mode	8-bit	8-bit
Basic single-width mode with: ■ 16-bit FPGA fabric-transceiver interface ■ 8B/10B decoder ■ Word aligner in automatic synchronization state machine mode	9-bit *(1)*	9-bit

Note to Table 1–22:

(1) If a /Kx.y/ control code group is selected as the byte ordering pattern, the MSB of the 9-bit byte ordering pattern must be 1'b1. If a /Dx.y/ data code group is selected as the byte ordering pattern, the MSB of the 9-bit byte ordering pattern must be 1'b0. The least significant 8 bits must be the 8B/10B decoded version of the code group used for byte ordering.

The byte ordering block modes of operation in single-width mode:

- Word-alignment-based byte ordering
- User-controlled byte ordering

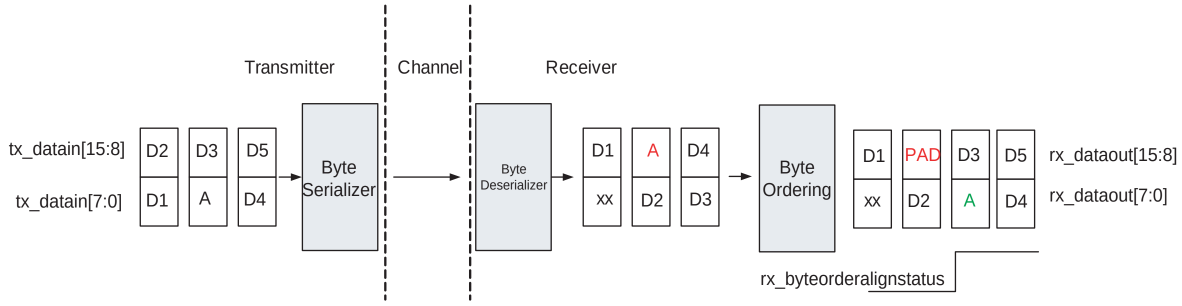
Word-Alignment-Based Byte Ordering

In word-alignment-based byte ordering, the byte ordering block starts looking for the byte ordering pattern in the byte-deserialized data every time it sees a rising edge on the `rx_syncstatus` signal. After a rising edge on the `rx_syncstatus` signal, if the byte ordering block finds the first data byte that matches the programmed byte ordering pattern in the MSByte position of the byte-deserialized data, it inserts one programmed PAD pattern to push the byte ordering pattern into the LSByte position. If the byte ordering blocks finds the first data byte that matches the programmed byte ordering pattern in the LSByte position of the byte-deserialized data, it considers the data to be byte ordered and does not insert any PAD pattern. In either case, the byte ordering block asserts the `rx_byteorderalignstatus` signal.

You can choose word-alignment-based byte ordering by selecting the sync status signal from the **word aligner** option in the **What do you want the byte ordering to be based on?** field in the ALTGX MegaWizard Plug-In Manager.

Figure 1–68 shows an example of the byte ordering operation in single-width modes. In this example, A is the programmed byte ordering pattern and PAD is the programmed PAD pattern. The byte deserialized data places the byte ordering pattern A in the MSByte position resulting in incorrect byte ordering. Assuming that a rising edge on the `rx_syncstatus` signal had occurred before the byte ordering block sees the byte ordering pattern A in the MSByte position, the byte ordering block inserts a PAD byte and pushes the byte ordering pattern A into the LSByte position. The data at the output of the byte ordering block has correct byte ordering as reflected on the `rx_byteorderalignstatus` signal.

Figure 1–68. Example of Byte Ordering in Single-Width Modes



If the byte ordering block sees another rising edge on the `rx_syncstatus` signal from the word aligner, it de-asserts the `rx_byteorderalignstatus` signal and repeats the byte ordering operation as previously discussed.

User-Controlled Byte Ordering

Unlike word-alignment-based byte ordering, user-controlled byte ordering provides control to the user logic to restore correct byte ordering at the receiver. When enabled, an `rx_enabyteord` port is available that you can use to trigger the byte ordering operation. A rising edge on the `rx_enabyteord` port triggers the byte ordering block. After a rising edge on the `rx_enabyteord` signal, if the byte ordering block finds the first data byte that matches the programmed byte ordering pattern in the MSByte position of the byte-deserialized data, it inserts one programmed PAD pattern to push the byte ordering pattern into the LSByte position. If the byte ordering blocks find the first data byte that matches the programmed byte ordering pattern in the LSByte position of the byte-deserialized data, it considers the data to be byte ordered and does not insert any PAD byte. In either case, the byte ordering block asserts the `rx_byteorderalignstatus` signal.

Receiver Phase Compensation FIFO

The receiver phase compensation FIFO in each channel ensures reliable transfer of data and status signals between the receiver channel and the FPGA fabric. The receiver phase compensation FIFO compensates for the phase difference between the parallel receiver PCS clock (FIFO write clock) and the FPGA fabric clock (FIFO read clock).

The receiver phase compensation FIFO operates in one of the following two modes:

- Low latency mode—The Quartus II software automatically configures the receiver phase compensation FIFO in low latency mode in all functional modes except PCI Express (PIPE). In this mode, the FIFO is four words deep and the latency through the FIFO is 2-to-3 parallel clock cycles (pending characterization).
- High latency mode—The Quartus II software automatically configures the receiver phase compensation FIFO in high latency mode in PCI Express (PIPE) functional mode. In this mode, the FIFO is eight words deep and the latency through the FIFO is 4-to-5 parallel clock cycles (pending characterization).
- Registered mode —The Quartus II software configures the receiver phase compensation FIFO in registered mode if the selection is made in the ALTGX MegaWizard Plug-In Manager. In this mode, latency uncertainty through the receiver phase compensation FIFO is 0 cycles. This mode is available for CPRI data rates in Basic functional mode. For more information, refer to "Basic Functional Mode" on page 1–85.

The receiver phase compensation FIFO write clock source varies with the receiver channel configuration. Table 1–23 shows the receiver phase compensation FIFO write clock source in different configurations.

Table 1–23. Receiver Phase Compensation FIFO Write Clock Source

Configuration	Receiver Phase Compensation FIFO Write Clock	
	Without Byte Serializer	**With Byte Serializer**
Non-bonded channel configuration with rate matcher	Parallel transmitter PCS clock from the local clock divider in the associated channel (`tx_clkout`)	Divide-by-two version of the parallel transmitter PCS clock from the local clock divider in the associated channel (`tx_clkout`)
Non-bonded channel configuration without rate matcher	Parallel recovered clock from the receiver PMA in the associated channel (`rx_clkout`)	Divide-by-two version of the parallel recovered clock from the receiver PMA in the associated channel (`rx_clkout`)
×4 bonded channel configuration	Parallel transmitter PCS clock from the central clock divider in the CMU0 of the associated transceiver block (`coreclkout`)	Divide-by-two version of the parallel transmitter PCS clock from the central clock divider in CMU0 of the associated transceiver block (`coreclkout`)
×8 bonded channel configuration	Parallel transmitter PCS clock from the central clock divider in CMU0 of the master transceiver block (`coreclkout` from master transceiver block)	Divide-by-two version of the parallel transmitter PCS clock from the central clock divider in CMU0 of the master transceiver block (`coreclkout` from master transceiver block)

The receiver phase compensation FIFO read clock source varies depending on whether or not you instantiate the `rx_coreclk` port in the ALTGX MegaWizard Plug-In Manager. Table 1–24 shows the receiver phase compensation FIFO read clock source in different configurations.

Table 1–24. Receiver Phase Compensation FIFO Read Clock Source

Configuration	Receiver Phase Compensation FIFO Read Clock	
	rx_coreclk Port Not Instantiated	**rx_coreclk Port Instantiated** *(1)*
Non-bonded channel configuration with rate matcher	FPGA fabric clock driven by the clock signal on the `tx_clkout` port	FPGA fabric clock driven by the clock signal on the `rx_coreclk` port
Non-bonded channel configuration without rate matcher	FPGA fabric clock driven by the clock signal on the `rx_clkout` port	FPGA fabric clock driven by the clock signal on the `rx_coreclk` port
×4 bonded channel configuration	FPGA fabric clock driven by the clock signal on the `coreclkout` port	FPGA fabric clock driven by the clock signal on the `rx_coreclk` port
×8 bonded channel configuration	FPGA fabric clock driven by the clock signal on the `coreclkout` port	FPGA fabric clock driven by the clock signal on the `rx_coreclk` port

Note to Table 1–24:

(1) The clock signal driven on the `rx_coreclk` port must have 0 PPM frequency difference with respect to the receiver phase compensation FIFO write clock.

Receiver Phase Compensation FIFO Error Flag

An optional `rx_phase_comp_fifo_error` port is available in all functional modes to indicate a receiver phase compensation FIFO overrun or underflow condition. The `rx_phase_comp_fifo_error` signal is asserted high when the phase compensation FIFO gets either full or empty. This feature is useful to verify a phase compensation FIFO overrun or underflow condition as a probable cause of link errors.

Functional Modes

You can configure Arria II GX transceivers in one of the following functional modes using the ALTGX MegaWizard Plug-In Manager:

- Basic single-width at 600 Mbps to 3.75 Gbps
- PCI Express (PIPE) (Gen1 at 2.5 Gbps)
- XAUI (3.125 Gbps up to HiGig/HiGig+ at 3.75 Gbps)
- GIGE (1.25 Gbps)
- Serial RapidIO (1.25 Gbps, 2.5 Gbps, 3.125 Gbps)
- SONET/SDH (OC-12, OC-48)
- SDI (HD at 1.485/1.4835 Gbps, 3G at 2.97/2.967 Gbps)

Basic Functional Mode

The Arria II GX transceiver datapath is extremely flexible in Basic functional mode. To configure the transceiver in Basic functional mode, you must select **Basic** in the **Which protocol will you be using?** option of the ALTGX MegaWizard Plug-In Manager.

The Basic functional mode runs in Basic single-width mode.

Table 1–25 shows the PCS-PMA interface widths and data rates supported in Basic single-width mode.

Table 1–25. PCS-PMA Interface Widths and Data Rates in Basic Single-Width Mode

Basic Functional Mode	Supported Data Rate Range *(1)*	PMA-PCS Interface Width
Basic single-width mode	600 Mbps to 3.75 Gbps	8-bit
		10-bit

Note to Table 1–25:

(1) The data rate range supported in Basic single-width mode varies depending on whether or not you use the byte serializer/deserializer. For more information, refer to "Basic Single-Width Mode Configurations" on page 1–87.

Low Latency PCS Datapath

The ALTGX MegaWizard Plug-In Manager provides an **Enable low latency PCS mode** option when configured in Basic single-width mode. If you select this option, the following transmitter and receiver channel PCS blocks are truly bypassed to yield a low latency PCS datapath:

- 8B/10B encoder and decoder
- Word aligner
- Deskew FIFO
- Rate match (clock rate compensation) FIFO
- Byte ordering

In low latency PCS modes, the transmitter and receiver phase compensation FIFOs are always enabled. Depending on the targeted data rate, the byte serializer and deserializer blocks can be optionally bypassed. For more information, refer to "Basic Single-Width Mode Configurations" on page 1–87.

The PCS latency in Basic single-width mode with and without the low latency PCS mode option is pending characterization.

Basic Single-Width Mode Configurations

Figure 1–69 shows Arria II GX transceiver configurations allowed in Basic single-width functional mode with an 8-bit wide PMA-PCS interface.

Figure 1–69. Transceiver Configurations in Basic Single-Width Mode with an 8-Bit Wide PMA-PCS Interface

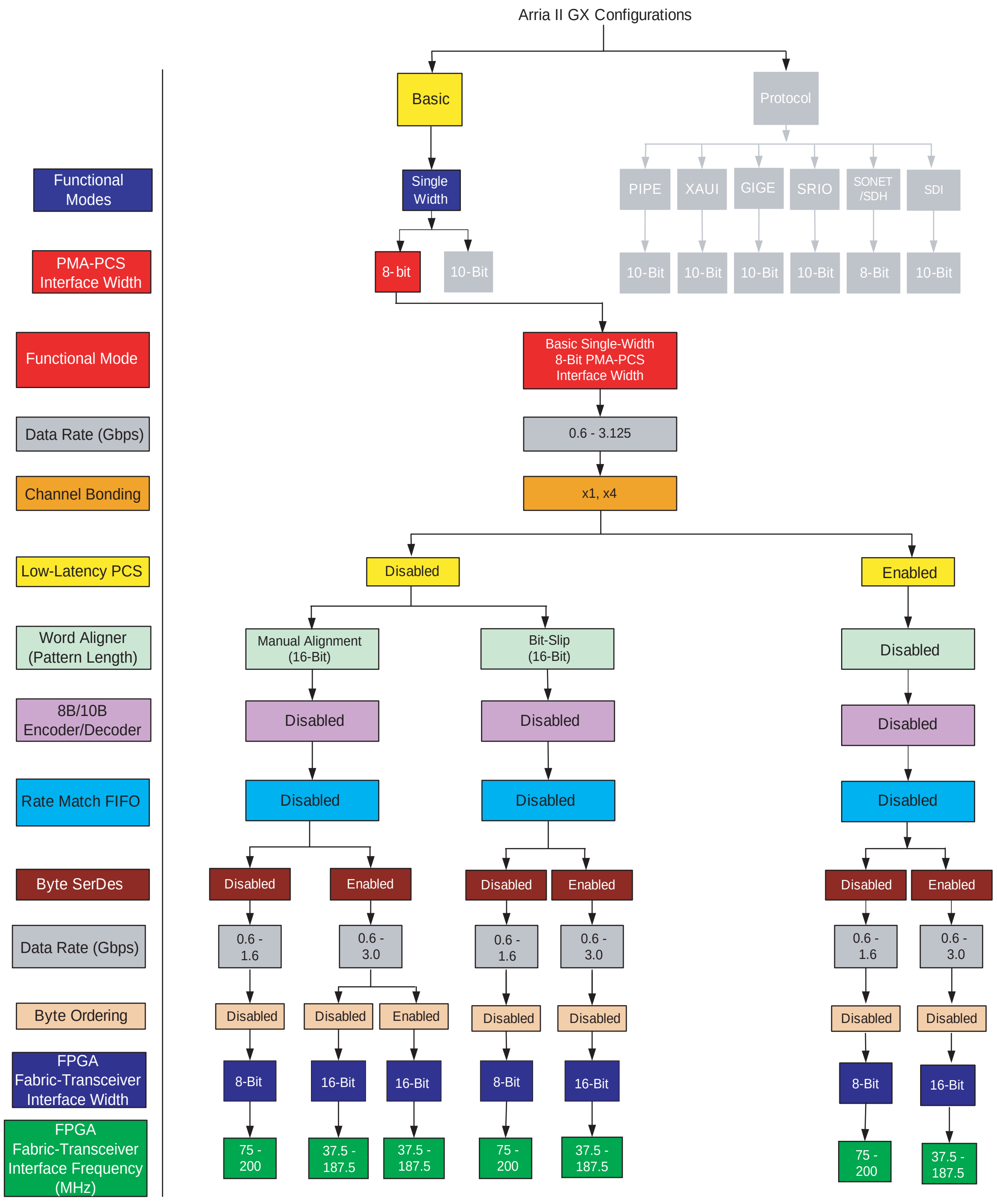
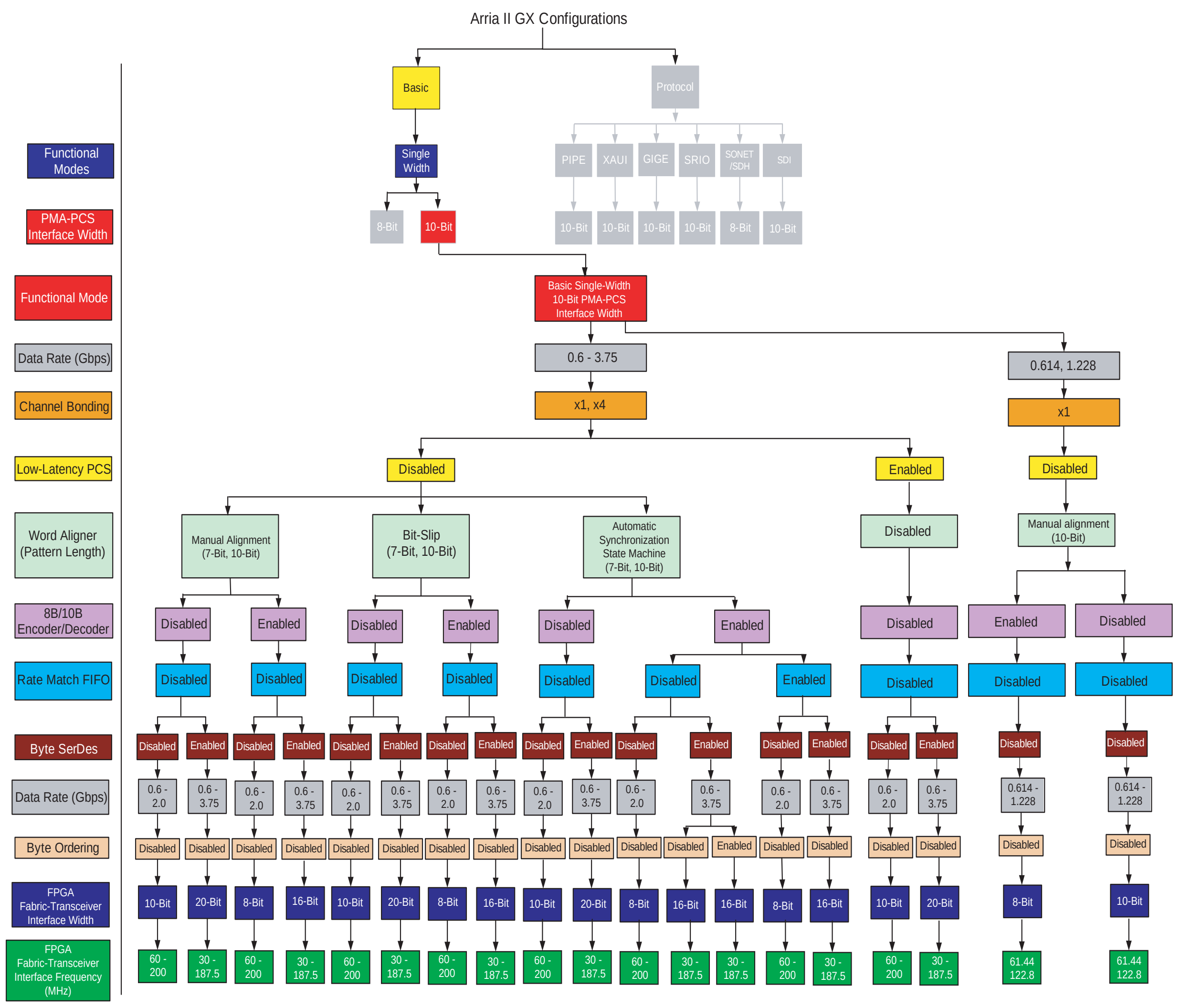


Figure 1–70 shows Arria II GX transceiver configurations allowed in Basic single-width functional mode with a 10-bit wide PMA-PCS interface.

Figure 1–70. Transceiver Configurations in Basic Single-Width Mode with a 10-Bit Wide PMA-PCS Interface



The receiver phase compensation FIFO can be configured in registered mode in single-width Basic mode to reduce latency uncertainty through the PCS for CPRI data rates of 614.4 Mbps and 1228.8 Mbps shown in the two right-most branches of Figure 1–70.

PCI Express (PIPE) Mode

Intel Corporation has developed a PHY interface for the PCI Express Architecture (PIPE) specification to enable implementation of a PCI Express-compliant physical layer device. The PIPE specification also defines a standard interface between the physical layer device and the media access control layer (MAC). Version 2.00 of the PIPE specification provides implementation details for a PCI Express-compliant physical layer device at Gen1 (2.5 GT/s) signaling rate.

To implement a Version 2.00 PIPE-compliant PHY, you must configure the Arria II GX transceivers in PCI Express (PIPE) functional mode. Arria II GX devices have built-in PCI Express hard IP blocks that you can use to implement the PHY-MAC layer, Data Link layer, and Transaction layer of the PCI Express protocol stack. You can also bypass the PCI Express hard IP blocks and implement the PHY-MAC layer, Data Link layer, and Transaction layer in the FGPA fabric using a Soft IP. If you enable the PCI Express hard IP blocks, the Arria II GX transceivers interface with these hard IP blocks. Otherwise, the Arria II GX transceivers interface with the FPGA fabric.

You can configure Arria II GX transceivers in PCI Express (PIPE) functional mode using one of the following two methods:

- ALTGX MegaWizard Plug-In Manager if you do not use the PCI Express hard IP block
- PCI Express Compiler if you use the PCI Express hard IP block

☞ Descriptions of PCI Express hard IP architecture and PCI Express (PIPE) mode configurations allowed when using PCI Express hard IP block are out of the scope of this chapter. For more information about the PCI Express hard IP block, refer to the *PCI Express Compiler User Guide.*

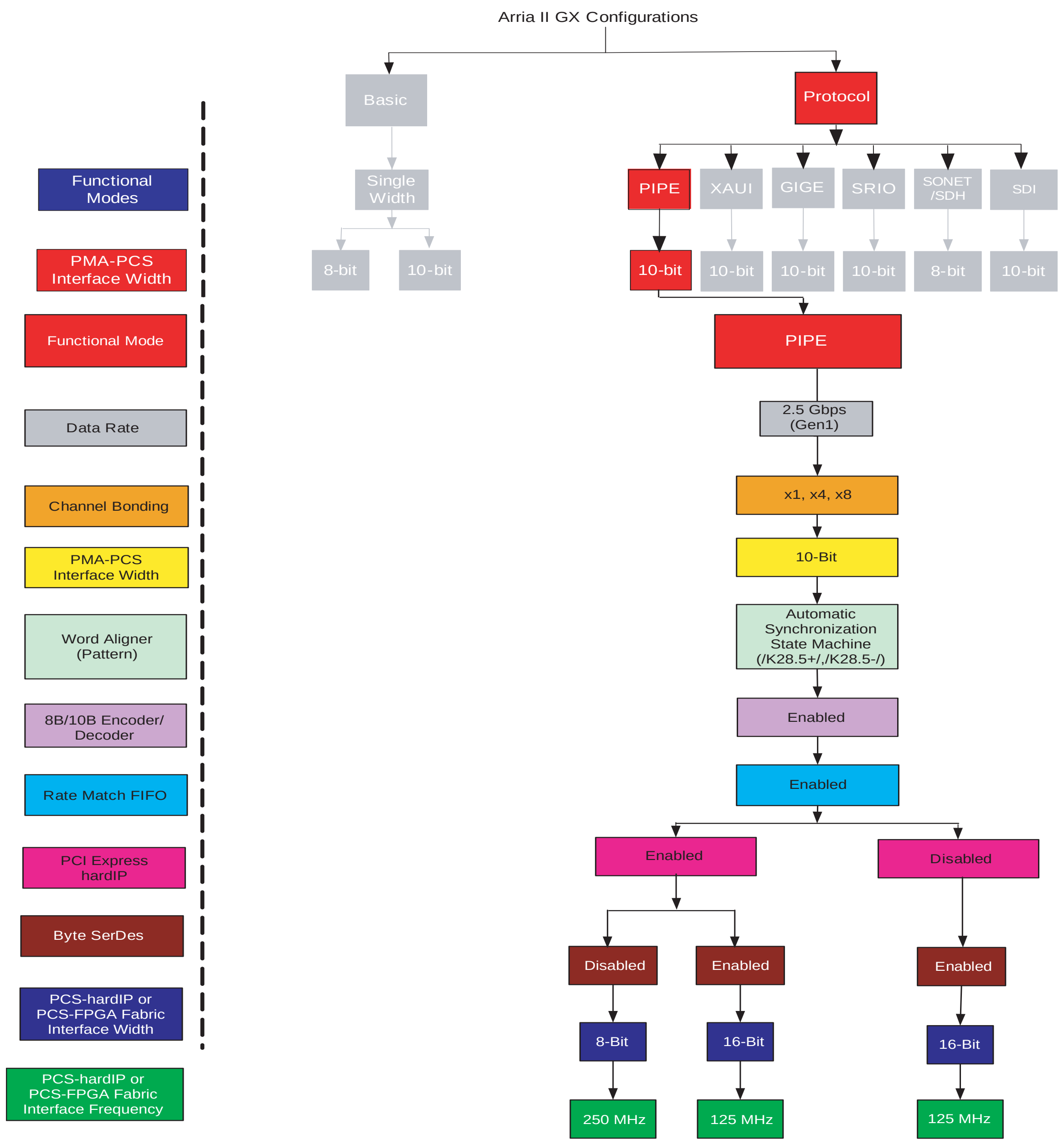
PCI Express (PIPE) Mode Configurations

Arria II GX transceivers support Gen1 (2.5 Gbps) data rates in PCI Express (PIPE) functional mode.

Arria II GX transceivers support ×1, ×4, and ×8 lane configurations in PIPE functional mode at 2.5 Gbps data rates. In PCI Express (PIPE) ×1 configuration, the PCS and PMA blocks of each channel are clocked and reset independently. PCI Express (PIPE) ×4 and ×8 configurations support channel bonding for four-lane and eight-lane PCI Express (PIPE) links. In these bonded channel configurations, the PCS and PMA blocks of all bonded channels share common clock and reset signals.

Figure 1–71 shows the Arria II GX transceiver configurations allowed in PCI Express (PIPE) functional mode.

Figure 1–71. Arria II GX Transceivers in PCI Express (PIPE) Functional Mode



PCI Express (PIPE) Mode Datapath

Figure 1–72 shows the Arria II GX transceiver datapath when configured in PCI Express (PIPE) functional mode.

Figure 1–72. Arria II GX Transceiver Datapath in PCI Express (PIPE) Mode

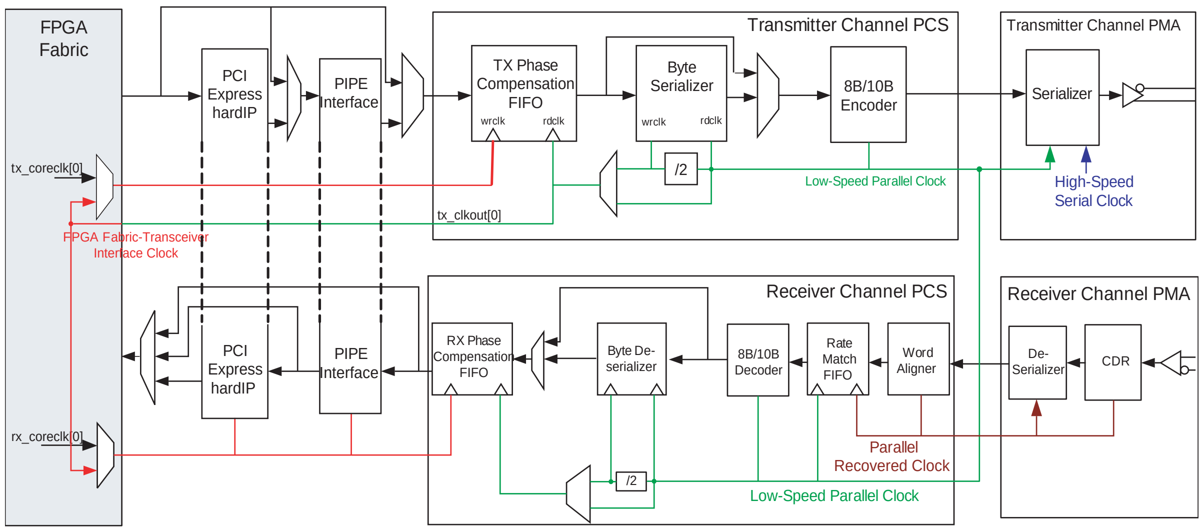


Table 1–26 shows the transceiver datapath clock frequencies in PCI Express (PIPE) functional mode configured using the ALTGX MegaWizard Plug-In Manager.

Table 1–26. Arria II GX Transceiver Datapath Clock Frequencies in PCI Express (PIPE) Mode

Functional Mode	Data Rate	High-Speed Serial Clock Frequency	Parallel Recovered Clock and Low-Speed Parallel Clock Frequency	FPGA Fabric-Transceiver Interface Clock Frequency With Byte Serializer/ Deserializer (16-Bit Wide)
PCI Express (PIPE) ×1, ×4, ×8 (Gen1)	2.5 Gbps	1.25 GHz	250 MHz	125 MHz

The transceiver datapath clocking varies between non-bonded (×1) and bonded (×4 and ×8) configurations in PCI Express (PIPE) mode.

For more information about transceiver datapath clocking in different PCI Express (PIPE) configurations, refer to the *Arria II GX Transceiver Clocking* chapter in volume 2 of the *Arria II GX Device Handbook*.

The transmitter datapath in PCI Express (PIPE) mode consists of:

- PIPE interface
- Transmitter phase compensation FIFO
- Optional byte serializer (enabled for 16-bit and disabled for 8-bit FPGA fabric-transceiver interface)
- 8B/10B encoder

- 10:1 serializer
- Transmitter buffer with receiver detect circuitry

The receiver datapath in PCI Express (PIPE) mode consists of:

- Receiver buffer with signal detect circuitry
- 1:10 deserializer
- Word aligner that implements PCI Express-compliant synchronization state machine
- Optional rate match FIFO (clock rate compensation) that can tolerate up to 600 PPM frequency difference
- 8B/10B decoder
- Optional byte deserializer (enabled for 16-bit and disabled for 8-bit FPGA fabric-transceiver interface)
- Receiver phase compensation FIFO
- PIPE interface

Table 1–27 shows features supported in PCI Express (PIPE) functional mode for 2.5 Gbps data rate configurations.

Table 1–27. Supported Features in PCI Express (PIPE) Mode

Feature	2.5 Gbps (Gen1)
×1, ×4, ×8 link configurations	✓
PCI Express-compliant synchronization state machine	✓
±300 PPM (total 600 PPM) clock rate compensation	✓
8-bit FPGA fabric-Transceiver interface	✓
16-bit FPGA fabric-Transceiver interface	✓
Transmitter buffer electrical idle	✓
Receiver Detection	✓
8B/10B encoder disparity control when transmitting compliance pattern	✓
Power state management	✓
Receiver status encoding	✓

PCI Express (PIPE) Interface

In PCI Express (PIPE) mode, each channel has a PIPE interface block that transfers data, control, and status signals between the PHY-MAC layer and the transceiver channel PCS and PMA blocks. The PIPE interface block is compliant to version 2.00 of the PCI Express (PIPE) specification. If you use the PCI Express hard IP block, the PHY-MAC layer is implemented in the hard IP block. Otherwise, the PHY-MAC layer may be implemented using Soft IP in the FPGA fabric.

☞ The PIPE interface block is only used in PCI Express (PIPE) mode and cannot be bypassed.

Besides transferring data, control, and status signals between the PHY-MAC layer and the transceiver, the PIPE interface block implements the following functions required in a PIPE-compliant physical layer device:

- Force the transmitter buffer in electrical idle state
- Initiate receiver detect sequence
- 8B/10B encoder disparity control when transmitting compliance pattern
- Manage PIPE power states
- Indicate completion of various PHY functions like receiver detection and power state transitions on the `pipephydonestatus` signal
- Encode receiver status and error conditions on the `pipestatus[2:0]` signal as specified in the PIPE specification

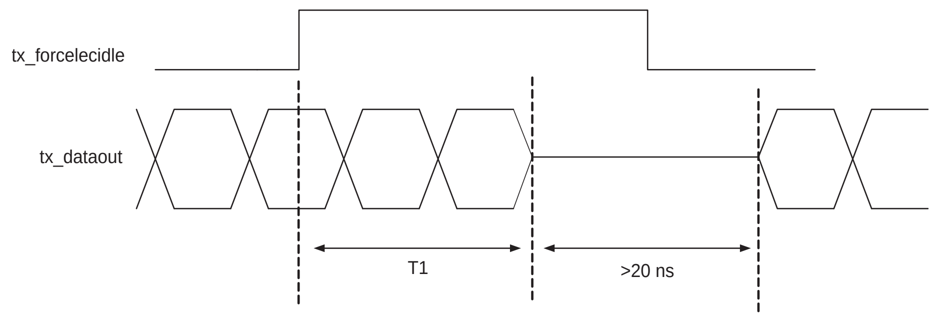
Transmitter Buffer Electrical Idle

When the input signal `tx_forceelecidle` is asserted high, the PIPE interface block puts the transmitter buffer in that channel in electrical idle state. During electrical idle, the transmitter buffer differential and common mode output voltage levels are compliant to the PCI Express Base Specification 1.1 for PCI Express Gen1 data rate.

Figure 1–73 shows the relationship between assertion of the `tx_forceelecidle` signal and the transmitter buffer output on the `tx_dataout` port. Time T1 taken from the assertion of the `tx_forceelecidle` signal to the transmitter buffer reaching electrical idle voltage levels is pending characterization. Once in electrical idle state, the PCI Express (PIPE) protocol requires the transmitter buffer to stay in electrical idle for a minimum of 20 ns for Gen1 data rate.

☞ The minimum period of time for which the `tx_forceelecidle` signal must be asserted high such that the transmitter buffer stays in electrical idle state for at least 20 ns is pending characterization.

Figure 1–73. Transmitter Buffer Electrical Idle State



The PCI Express (PIPE) specification requires the transmitter buffer to be in electrical idle in certain power states. Refer to Table 1–26 on page 1–91 for more information on the `tx_forceelecidle` signal levels required in different PCI Express (PIPE) power states.

Receiver Detection

During the detect substate of the long-term sample storage module (LTSSM) state machine, the PCI Express (PIPE) protocol requires the transmitter channel to perform a receiver detect sequence to detect if a receiver is present at the far end of each lane. The PCI Express (PIPE) specification requires that a receiver detect operation be performed during the P1 power state.

The PIPE interface block in Arria II GX transceivers provides an input signal `tx_detectrxloopback` for the receiver detect operation. When the input signal `tx_detectrxloopback` is asserted high in the P1 power state, the PIPE interface block sends a command signal to the transmitter buffer in that channel to initiate a receiver detect sequence. In the P1 power state, the transmitter buffer must always be in electrical idle state. On receiving this command signal, the receiver detect circuitry creates a step voltage at the output of the transmitter buffer. If an active receiver (that complies to the PCI Express [PIPE] input impedance requirements) is present at the far end, the time constant of the step voltage on the trace is higher compared to when the receiver is not present. The receiver detect circuitry monitors the time constant of the step signal seen on the trace to decide if a receiver was detected or not. The receiver detect circuitry monitor needs a 125-MHz clock for operation that you must drive on the `fixedclk` port.

For the receiver detect circuitry to function reliably, the AC-coupling capacitor on the serial link and the receiver termination values used in your system must be compliant to the PCI Express (PIPE) Base Specification 1.1.

Receiver detect circuitry communicates the status of the receiver detect operation to the PIPE interface block. If a far-end receiver is successfully detected, the PIPE interface block asserts `pipephydonestatus` for one clock cycle and synchronously drives the `pipestatus[2:0]` signal to 3'b011. If a far-end receiver is not detected, the PIPE interface block asserts `pipephydonestatus` for one clock cycle and synchronously drives the `pipestatus[2:0]` signal to 3'b000.

Figure 1–74 and Figure 1–75 show the receiver detect operation where a receiver was successfully detected and where a receiver was not detected, respectively.

Figure 1–74. Receiver Detect, Successfully Detected

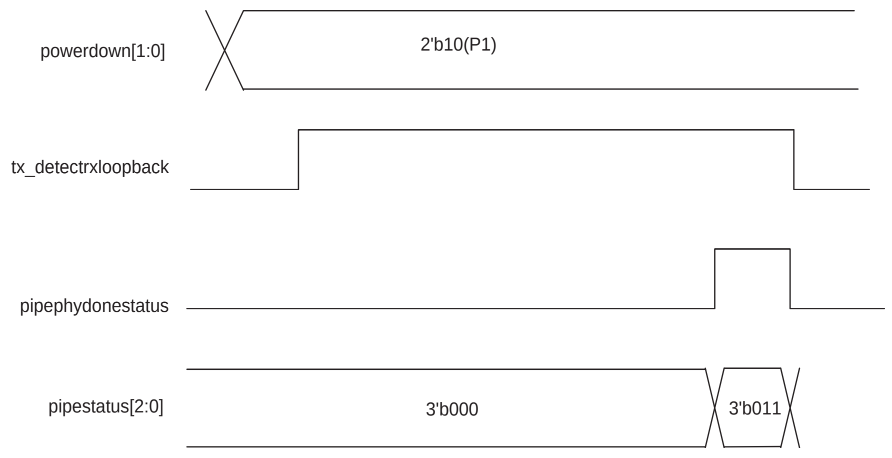
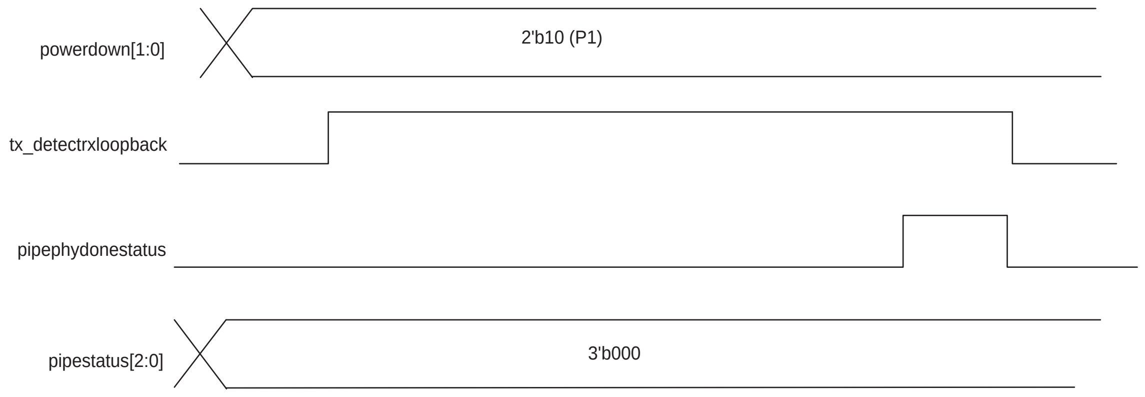


Figure 1–75. Receiver Detect, Unsuccessful



Compliance Pattern Transmission Support

The LTSSM state machine can enter the “polling.compliance” substate where the transmitter is required to transmit a compliance pattern as specified in the PCI Express (PIPE) Base Specification 1.1. The “polling.compliance” substate is intended to assess if the transmitter is electrically compliant with the PCI Express (PIPE) voltage and timing specifications.

The compliance pattern is a repeating sequence of the following four code groups:

- /K28.5/
- /D21.5/
- /K28.5/
- /D10.2/

PCI Express (PIPE) protocol requires the first /K28.5/ code group of the compliance pattern to be encoded with negative current disparity. To satisfy this requirement, the PIPE interface block provides an input signal, `tx_forcedispcompliance`. A high level on `tx_forcedispcompliance` forces the associated parallel transmitter data on the `tx_datain` port to transmit with negative current running disparity.

- For 8-bit transceiver channel width configurations, you must drive `tx_forcedispcompliance` high in the same parallel clock cycle as the first /K28.5/ of the compliance pattern on the `tx_datain` port.
- For 16-bit transceiver channel width configurations, you must drive only the LSB of `tx_forcedispcompliance[1:0]` high in the same parallel clock cycle as /K28.5/D21.5/ of the compliance pattern on the `tx_datain` port.

Figure 1–76 and Figure 1–77 show the required level on the `tx_forcedispcompliance` signal while transmitting the compliance pattern in 8-bit and 16-bit channel width configurations, respectively.

Figure 1–76. Compliance Pattern Transmission Support, 8-Bit Wide Channel Configurations

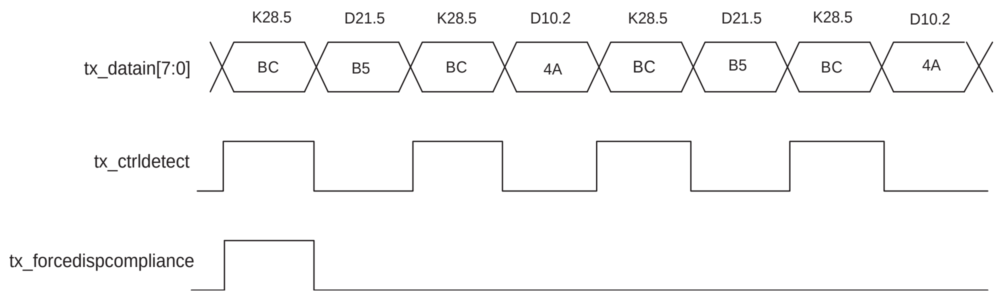
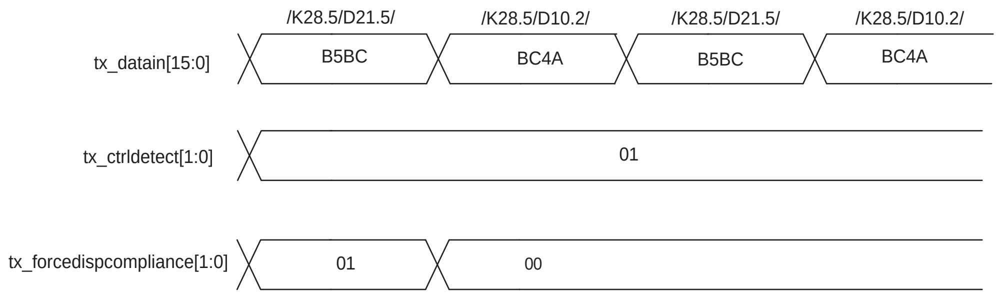


Figure 1–77. Compliance Pattern Transmission Support, 16-Bit Wide Channel Configurations



Power State Management

The PCI Express (PIPE) specification defines four power states, namely P0, P0s, P1, and P2, that the physical layer device must support to minimize power consumption.

- P0 is the normal operation state during which packet data is transferred on the PCI Express (PIPE) link.
- P0s, P1, and P2 are low-power states into which the physical layer must transition as directed by the PHY-MAC layer to minimize power consumption.

The PCI Express (PIPE) specification provides the mapping of these power states to the LTSSM states specified in the PCI Express (PIPE) Base Specification 1.1. The PHY-MAC layer is responsible for implementing the mapping logic between the LTSSM states and the four power states in the PCI Express (PIPE)-compliant PHY.

The PIPE interface in Arria II GX transceivers provides an input port, `powerdn[1:0]`, for each transceiver channel configured in PCI Express (PIPE) mode.

Table 1–30 shows mapping between the logic levels driven on the `powerdn[1:0]` port and the resulting power state that the PIPE interface block puts the transceiver channel into.

Table 1–28. powerdn[1:0] Port Power State Functions and Descriptions (Part 1 of 2)

Power State	powerdn	Function	Description
P0	2'b00	Transmits normal data, transmits Electrical Idle, or enters into loopback mode	Normal operation mode
P0s	2'b01	Only transmits Electrical Idle	Low recovery time saving state

Table 1–28. powerdn[1:0] Port Power State Functions and Descriptions (Part 2 of 2)

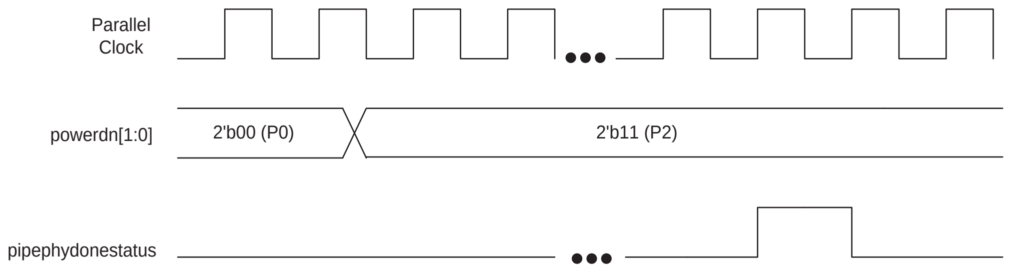
Power State	powerdn	Function	Description
P1	2'b10	Transmitter buffer is powered down and can do a receiver detect while in this state	High recovery time power saving state
P2	2'b11	Transmits Electrical Idle or a beacon to wake up the downstream receiver	Lowest power saving state

When transitioning from the P0 power state to lower power states (P0s, P1, and P2), the PCI Express (PIPE) specification requires the physical layer device to implement power saving measures. Arria II GX transceivers do not implement these power saving measures except when putting the transmitter buffer in Electrical Idle in the lower power states.

The PIPE interface block indicates successful power state transition by asserting the `pipephydonestatus` signal for one parallel clock cycle as specified in the PCI Express (PIPE) specification. The PHY-MAC layer must not request any further power state transition until the `pipephydonestatus` signal has indicated the completion of the current power state transition request.

Figure 1–78 shows an example waveform for a transition from P0 to P2 power state.

Figure 1–78. Example of Power State Transition from P0 to P2



The PCI Express (PIPE) specification allows the PIPE interface to perform protocol functions like receiver detect, loopback, and beacon transmission in specified power states only. This requires the PHY-MAC layer to drive the `tx_detectrxloopback` and `tx_forceelecidle` signals appropriately in each power state to perform these functions. Table 1–29 summarizes the logic levels that the PHY-MAC layer must drive on the `tx_detectrxloopback` and `tx_forceelecidle` signals in each power state.

Table 1–29. Logic Levels for the PHY-MAC Layer (Part 1 of 2)

Power State	tx_detectrxloopback	tx_forceelecidle
P0	0: normal mode 1: data path in loopback mode	0: Must be deasserted 1: Illegal mode
P0s	Don't care	0: Illegal mode 1: Must be asserted in this state

Table 1–29. Logic Levels for the PHY-MAC Layer (Part 2 of 2)

Power State	tx_detectrxloopback	tx_forceelecidle
P1	0: Electrical Idle 1: receiver detect	0: Illegal mode 1: Must be asserted in this state
P2	Don't care	Deasserted in this state for sending beacon. Otherwise asserted.

Receiver Status

The PCI Express (PIPE) specification requires the PHY to encode the receiver status on a 3-bit `RxStatus[2:0]` signal. This status signal is used by the PHY-MAC layer for its operation.

The PIPE interface block receives status signals from the transceiver channel PCS and PMA blocks and encodes the status on the 3-bit output signal `pipestatus[2:0]` to the FPGA fabric. The encoding of the status signals on `pipestatus[2:0]` is compliant with the PCI Express (PIPE) specification and listed in Table 1–30.

Table 1–30. Encoding of the Status Signals on pipestatus[2:0]

pipestatus[2:0]	Description	Error Condition Priority
3'b000	Received data OK	N/A
3'b001	One SKP symbol added	5
3'b010	One SKP symbol deleted	6
3'b011	Receiver detected	N/A
3'b100	8B/10B decode error	1
3'b101	Elastic buffer (rate match FIFO) overflow	2
3'b110	Elastic buffer (rate match FIFO) underflow	3
3'b111	Received disparity error	4

Two or more of the error conditions, for example, 8B/10B decode error (code group violation), rate match FIFO overflow or underflow, or receiver disparity error, can occur simultaneously. The PIPE interface follows the priority listed in Table 1–30 while encoding the receiver status on the `pipestatus[2:0]` port. For example, if the PIPE interface receives an 8B/10B decode error and disparity error for the same symbol, it drives 3'b100 on the `pipestatus[2:0]` signal.

Fast Recovery Mode

The PCI Express Base specification fast training sequences (FTS) are used for bit and byte synchronization to transition from L0s to L0 (PIPE P0s to P0) power states. The PCI Express Base Specification requires the physical layer device to acquire bit and byte synchronization after transitioning from L0s to L0 state within 16 ns to 4 µs.

If the Arria II GX receiver CDR is configured in Automatic Lock mode, the receiver cannot meet the PCI Express specification of acquiring bit and byte synchronization within 4 µs due to the signal detect and PPM detector time. To meet this specification, each Arria II GX transceiver has a built-in Fast Recovery circuitry that you can optionally enable.

To enable the Fast Recovery circuitry, select the **Enable fast recovery mode** option in the ALTGX MegaWizard Plug-In Manager.

If enabled, the Fast Recovery circuitry controls the receiver CDR `rx_locktorefclk` and `rx_locktodata` signals to force the receiver CDR in LTR or LTD modes. It relies on the Electrical Idle Ordered Sets (EIOS), NFTS sequences received in L0 power state, and the signal detect signal from the receiver input buffer to control the receiver CDR lock mode.

The Fast Recovery circuitry is self-operational and does not require control inputs from you. When enabled, the `rx_locktorefclk` and `rx_locktodata` ports are not available in the ALTGX MegaWizard Plug-In Manager.

Electrical Idle Inference

The PCI Express (PIPE) protocol allows inferring the electrical idle condition at the receiver instead of detecting the electrical idle condition using analog circuitry. Clause 4.2.4.3 in PCI Express (PIPE) Base Specification 1.1 specifies conditions to infer electrical idle at the receiver in various sub-states of the LTSSM state machine.

In all PCI Express (PIPE) modes (×1, ×4, and ×8), each receiver channel PCS has an optional Electrical Idle Inference module designed to implement the electrical idle inference conditions specified in the PCI Express (PIPE) Base Specification 1.1. You can enable the Electrical Idle Inference module by selecting the **Enable electrical idle inference functionality** option in the ALTGX MegaWizard Plug-In manager.

If enabled, this module infers electrical idle depending on the logic level driven on the `rx_elecidleinfersel[2:0]` input signal. The Electrical Idle Inference module in each receiver channel indicates whether the electrical idle condition is inferred or not on the `pipeelecidle` signal of that channel. The Electrical Idle Interface module drives the `pipeelecidle` signal high if it infers an electrical idle condition; otherwise, it drives it low.

Table 1–31 shows electrical idle inference conditions specified in the PCI Express (PIPE) Base Specification 1.1 and implemented in the Electrical Idle Inference module to infer electrical idle in various substates of the LTSSM state machine. For the Electrical Idle Inference Module to correctly infer an electrical idle condition in each LTSSM substate, you must drive the `rx_elecidleinfersel[2:0]` signal appropriately, as shown in Table 1–31.

Table 1–31. Electrical Idle Inference Conditions (Part 1 of 2)

LTSSM State	Gen1 (2.5 Gbps)	rx_elecidleinfersel[2:0]
L0	Absence of update FC or alternatively skip ordered set in 128 µs window	3'b100
Recovery.RcvrCfg	Absence of TS1 or TS2 ordered set in 1280 UI interval	3'b101
Recovery.Speed when successful speed negotiation = 1'b1	Absence of TS1 or TS2 ordered set in 1280 UI interval	3'b101

Table 1–31. Electrical Idle Inference Conditions (Part 2 of 2)

LTSSM State	Gen1 (2.5 Gbps)	rx_elecidleinfersel[2:0]
Recovery.Speed when successful speed negotiation = 1'b0	Absence of an exit from Electrical Idle in 2000 UI interval	3'b110
Loopback.Active (as slave)	Absence of an exit from Electrical Idle in 128 µs window	3'b111

In the "Recovery.Speed" substate of the LTSSM state machine with unsuccessful speed negotiation (`rx_elecidleinfersel[2:0]` = *3'b110*), the PCI Express (PIPE) Base Specification requires the receiver to infer an electrical idle condition (`pipeelecidle` = high) if absence of an exit from Electrical Idle is detected in a 2000 UI interval for Gen1 data rate. The electrical idle inference module detects an absence of exit from Electrical Idle if four /K28.5/ COM code groups are not received in the specified interval.

In other words, when configured for Gen1 data rate and `rx_elecidleinfersel[2:0]` = 3'b110, the Electrical Idle inference module asserts `pipeelecidle` high if it does not receive four /K28.5/ COM code groups in a 2000 UI interval. When configured for Gen1 data rate and `rx_elecidleinfersel[2:0]` = 3'b111 in Loopback.Active substate of the LTSSM state machine, the Electrical Idle inference module asserts `pipeelecidle` high if it does not receive four /K28.5/ COM code groups in a 128 µs interval.

☞ The Electrical Idle inference module does not have the capability to detect electrical idle exit condition based on reception of the electrical idle exit ordered set (EIEOS), as specified in the PCI Express (PIPE) Base Specification.

If you select the **Enable Electrical Idle Inference Functionality** option in the ALTGX MegaWizard Plug-In Manager and drive `rx_elecidleinfersel[2:0] = 3'b0xx`, the Electrical Idle Inference Block uses EIOS detection from the Fast Recovery circuitry to drive the `pipeelecidle` signal

If you do not select the **Enable electrical idle inference functionality** option in the ALTGX MegaWizard Plug-In manager, the Electrical Idle inference module is disabled. In this case, the `rx_signaldetect` signal from the signal detect circuitry in the receiver buffer is inverted and driven as the `pipeelecidle` signal.

PCI Express Cold Reset Requirements

The PCI Express Base Specification 1.1 defines the following three types of conventional resets to the PCI Express system components:

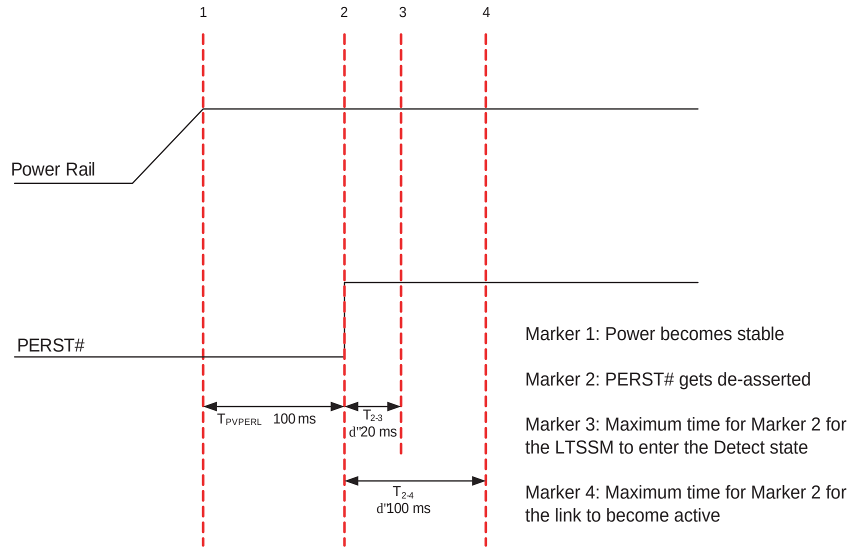
- Cold Reset—fundamental reset after power up
- Warm Reset—fundamental reset without removal and re-application of power
- Hot Reset—In-band conventional reset initiated by higher layer by setting the Hot Reset bit in the TS1 or TS2 training sequences

Fundamental reset is provided by the system to the component or adapter card using the auxiliary signal PERST#. The PCI Express Base Specification 1.1 requires that PERST# must be kept asserted for a minimum of 100 ms (TPVPERL) after the system power becomes stable in a cold reset situation. Additionally, all system components must enter the LTSSM Detect state within 20 ms and the link must become active within 100 ms after de-assertion of the PERST# signal. This implies that each PCI Express system component must become active within 200 ms after the power becomes stable.

☞ The link being active is interpreted as the physical layer device coming out of Electrical Idle in L0 state of the LTSSM state machine.

Figure 1–79 shows the PCI Express cold reset timing requirements.

Figure 1–79. PCI Express Cold Reset Requirements


1
2
3
4
Power Rail
PERST#
TPVPERL 100 ms
T2-3 d"20 ms
T2-4 d"100 ms
Marker 1: Power becomes stable
Marker 2: PERST# gets de-asserted
Marker 3: Maximum time for Marker 2 for the LTSSM to enter the Detect state
Marker 4: Maximum time for Marker 2 for the link to become active

Time taken by a PCI Express port implemented using the Arria II GX device to go from power up to link active state is stated below:

- Power On Reset—begins after power rails become stable. Typically takes 12 ms
- FPGA Configuration/Programming—begins after power on reset. Configuration time depends on the FPGA density
- Time taken from de-assertion of PERST# to link active—typically takes 40 ms (pending characterization and verification of the PCI Express soft IP and hard IP)

To meet the PCI Express specification of 200 ms from power on to link active, the Arria II GX device configuration time must be less than 148 ms (200 ms - 12 ms for power on reset - 40 ms for link to become active after PERST# de-assertion).

Table 1–32 shows typical configuration times for Arria II GX devices when updated using the Fast Passive Parallel (FPP) configuration scheme at 125 MHz.

Table 1–32. Typical Configuration Times for Arria II GX Devices Configured with Fast Passive Parallel

Device	Time (ms) *(1)*
EP2AGX20	24
EP2AGX30	24
EP2AGX45	24
EP2AGX65	35
EP2AGX95	35
EP2AGX125	48
EP2AGX190	48
EP2AGX260	79

Note to Table 1–32:

(1) Pending characterization.

For more information about FPP configuration scheme, refer to the *Configuration, Design Security, Remote System Upgrades with Arria II GX Devices* chapter in volume 1 in the *Arria II GX Device Handbook.*

Most flash memories available in the market can run up to 100 MHz. To configure the Arria II GX devices at 125 MHz, Altera recommends using a MAX II device to convert the 16-bit flash memory output at 62.5 MHz to 8-bit configuration data input to the Arria II GX devices at 125 MHz.

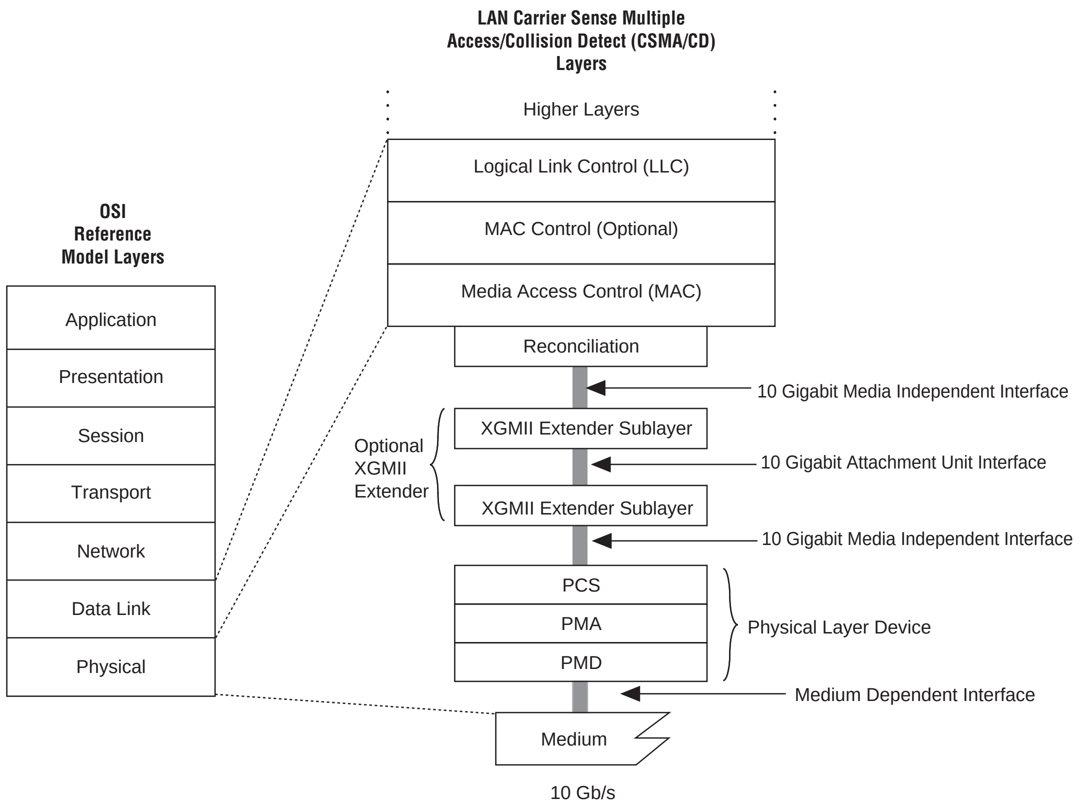
XAUI Mode

The XAUI is an optional, self-managed interface that you can insert between the reconciliation sublayer and the PHY layer to transparently extend the physical reach of the 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 a low-voltage differential signaling method, the electrical distance 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 1–80 shows the relationships between the XGMII and XAUI layers.

Figure 1–80. XAUI and XGMII Layers


LAN Carrier Sense Multiple Access/Collision Detect (CSMA/CD) Layers
Higher Layers
Logical Link Control (LLC)
MAC Control (Optional)
Media Access Control (MAC)
Reconciliation
OSI Reference Model Layers
Application
Presentation
Session
Transport
Network
Data Link
Physical
10 Gigabit Media Independent Interface
Optional XGMII Extender
XGMII Extender Sublayer
10 Gigabit Attachment Unit Interface
XGMII Extender Sublayer
10 Gigabit Media Independent Interface
PCS
PMA
PMD
Physical Layer Device
Medium Dependent Interface
Medium
10 Gb/s

The XGMII interface consists of four lanes of 8 bits. At the transmit side of the XAUI interface, the data and control characters are converted within the XGMII extender sublayer (XGXS) into an 8B/10B encoded data stream. Each data stream is then transmitted across a single differential pair running at 3.125 Gbps (3.75 Gbps for HiGig/HiGig+). At the XAUI receiver, the incoming data is decoded and mapped back to the 32-bit XGMII format. This 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 is handled with no upper layer support requirements. This functionality is based on the PCS code groups that are used during the IPG time and idle periods. PCS code groups are mapped by the XGXS to XGMII characters specified in Table 1–33.

Table 1–33. XGMII Character to PCS Code-Group Mapping (Part 1 of 2)

XGMII TXC	XGMII TXD *(1)*	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

Table 1–33. XGMII Character to PCS Code-Group Mapping (Part 2 of 2)

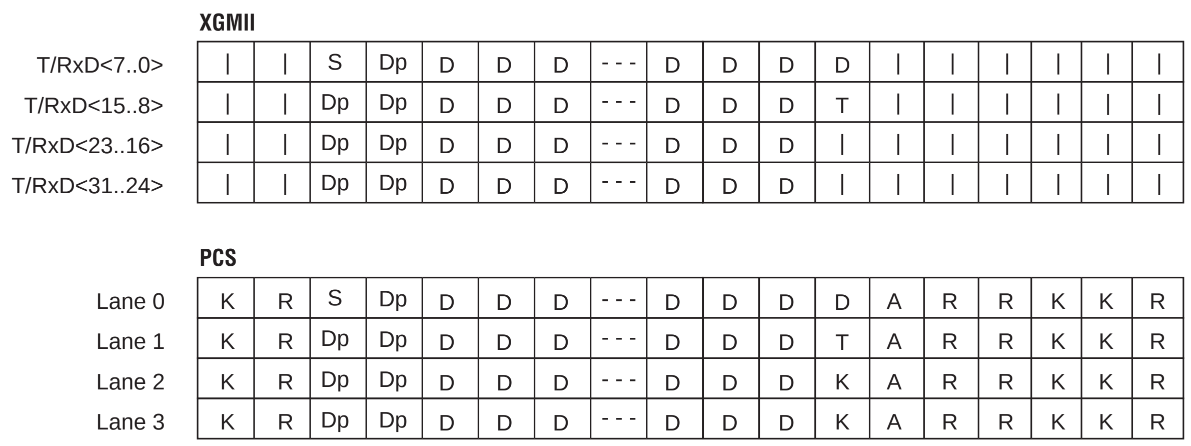
XGMII TXC	XGMII TXD *(1)*	PCD Code Group	Description
1	FB	K27.7	Start
1	FD	K29.7	Terminate
1	FE	K30.7	Error
1	Any other value	K30.7	Invalid XGMII character

Note to Table 1–33:

(1) The values in the XGMII TXD column are in hexadecimal.

Figure 1–81 shows an example of mapping between XGMII characters and the PCS code groups that are used in XAUI. The idle characters are mapped to a pseudo-random sequence of /A/, /R/, and /K/ code groups.

Figure 1–81. Example of Mapping XGMII Characters to PCS Code Groups



XGMII

T/RxD<7..0>	|	|	S	Dp	D	D	D	- - -	D	D	D	D	|	|	|	|	|	|
T/RxD<15..8>	|	|	Dp	Dp	D	D	D	- - -	D	D	D	T	|	|	|	|	|	|
T/RxD<23..16>	|	|	Dp	Dp	D	D	D	- - -	D	D	D	|	|	|	|	|	|	|
T/RxD<31..24>	|	|	Dp	Dp	D	D	D	- - -	D	D	D	|	|	|	|	|	|	|

PCS

Lane 0	K	R	S	Dp	D	D	D	- - -	D	D	D	D	A	R	R	K	K	R
Lane 1	K	R	Dp	Dp	D	D	D	- - -	D	D	D	T	A	R	R	K	K	R
Lane 2	K	R	Dp	Dp	D	D	D	- - -	D	D	D	K	A	R	R	K	K	R
Lane 3	K	R	Dp	Dp	D	D	D	- - -	D	D	D	K	A	R	R	K	K	R

The PCS code groups are sent using 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 1–34 lists the defined idle ordered sets (||I||) that are used for the self-managed properties of XAUI.

Table 1–34. Defined Idle Ordered Set

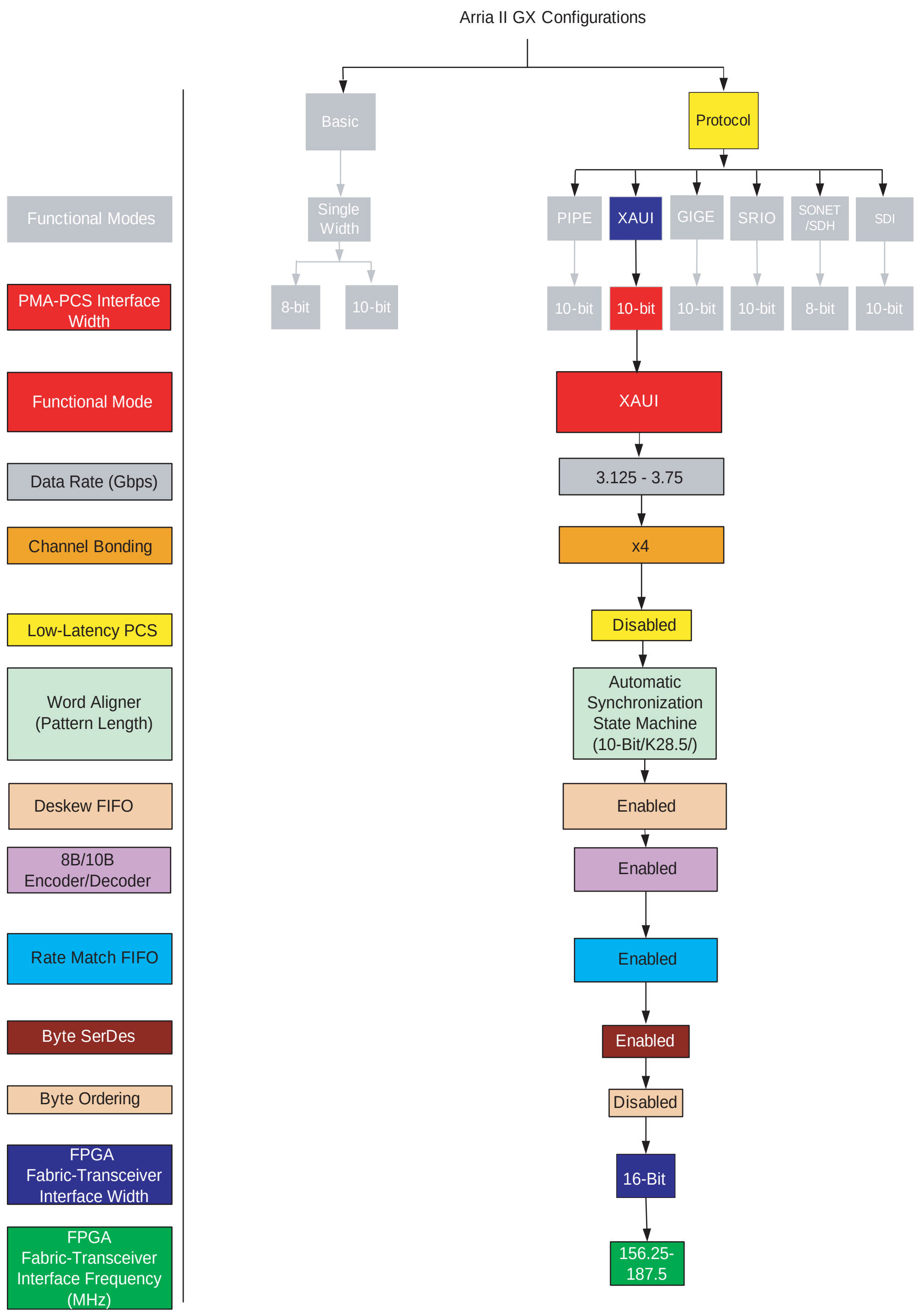
Code	Ordered Set	Number of Code Groups	Encoding
||I||	Idle		Substitute for XGMII Idle
||K||	Synchronization column	4	/K28.5/K28.5/K28.5/K28.5/
||R||	Skip column	4	/K28.0/K28.0/K28.0/K28.0/
||A||	Align column	4	/K28.3/K28.3/K28.3/K28.3/

Arria II GX transceivers configured in XAUI mode provide the following protocol features:

- XGMII-to-PCS code conversion at the transmitter
- PCS-to-XGMII code conversion at the receiver
- 8B/10B encoding and decoding
- IEEE P802.3ae-compliant synchronization state machine
- ±100 PPM clock rate compensation
- Channel deskew of four lanes of the XAUI link

Figure 1–82 shows the XAUI mode configuration supported in Arria II GX devices.

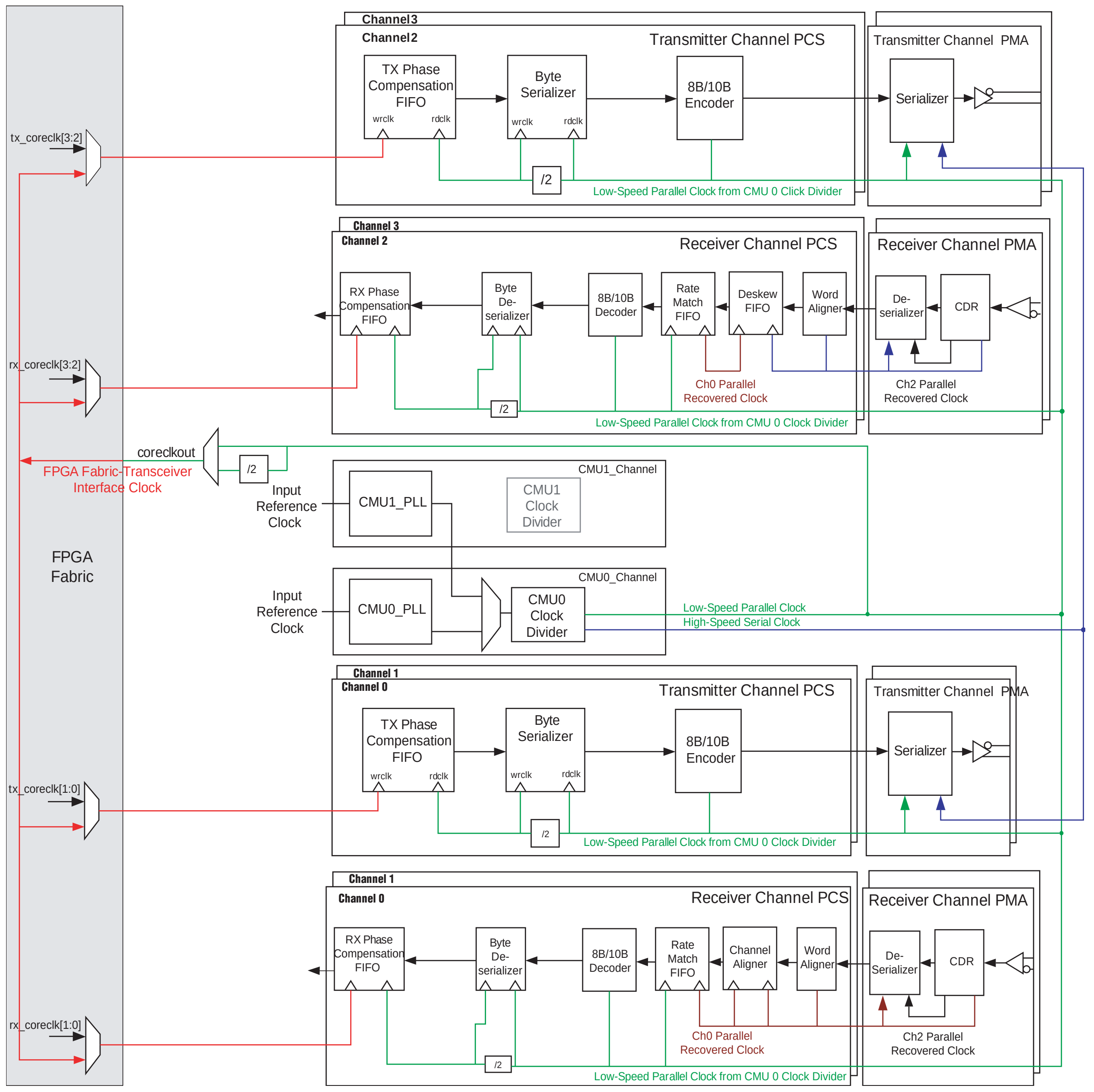
Figure 1–82. Arria II GX XAUI Mode Configuration



XAUI Mode Datapath

Figure 1–83 shows the ALTGX megafunction transceiver datapath when configured in XAUI mode.

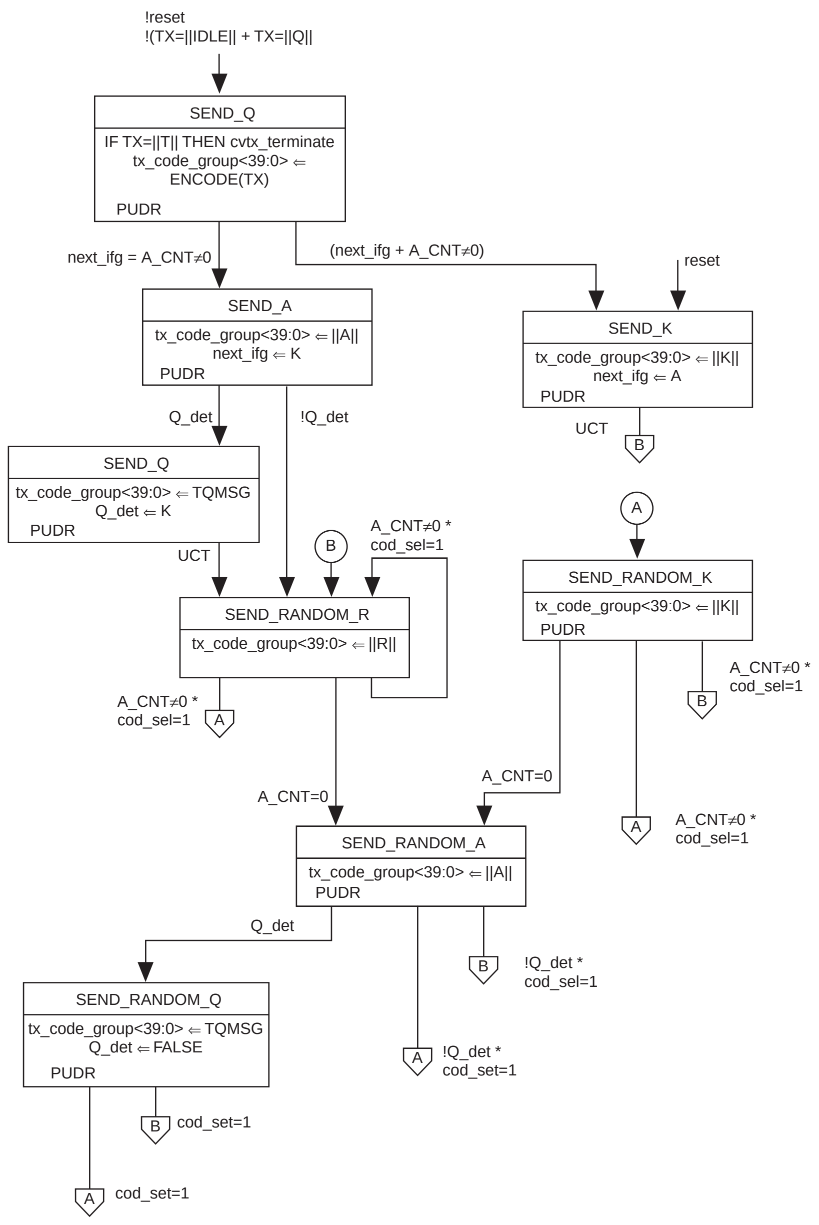
Figure 1–83. Transceiver Datapath in XAUI Mode



XGMII-To-PCS Code Conversion at the Transmitter

In XAUI mode, the 8b/10b encoder in the Arria II GX transmitter datapath is controlled by a transmitter state machine that maps various 8-bit XGMII codes to 10-bit PCS code groups. This state machine complies with the IEEE P802.3ae PCS transmit source state diagram shown in Figure 1–84.

Figure 1–84. XGMII-To-PCS Code Conversion in XAUI Mode *(Note 1)*



Note to Figure 1–84:

(1) Source: IEEE P802.3ae™-2002, IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.

Table 1–35 lists the XGMII-to-PCS code group conversion in XAUI functional mode. The XGMII TXC control signal is equivalent to the `tx_ctrlenable` signal; the XGMII TXD control signal is equivalent to the `tx_datain[7:0]` signal.

Table 1–35. XGMII Character to PCS Code-Group Mapping

XGMII TXC	XGMII TXD *(1)*	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 1–35:

(1) The values in the XGMII TXD column are in hexadecimal.

PCS-To-XGMII Code Conversion at the Receiver

In XAUI mode, the 8b/10b decoder in the Arria II GX receiver datapath is controlled by a XAUI receiver state machine that converts received PCS code groups into specific 8-bit XGMII codes.

Table 1–36 lists the PCS-to-XGMII code group conversion in XAUI functional mode. The XGMII RXC control signal is equivalent to the `rx_ctrldetect` signal; the XGMII RXD control signal is equivalent to the `rx_dataout[7:0]` signal.

Table 1–36. PCS Code Group to XGMII Character Mapping

XGMII RXC	XGMII RXD *(1)*	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	FE	Invalid code group	Received code group

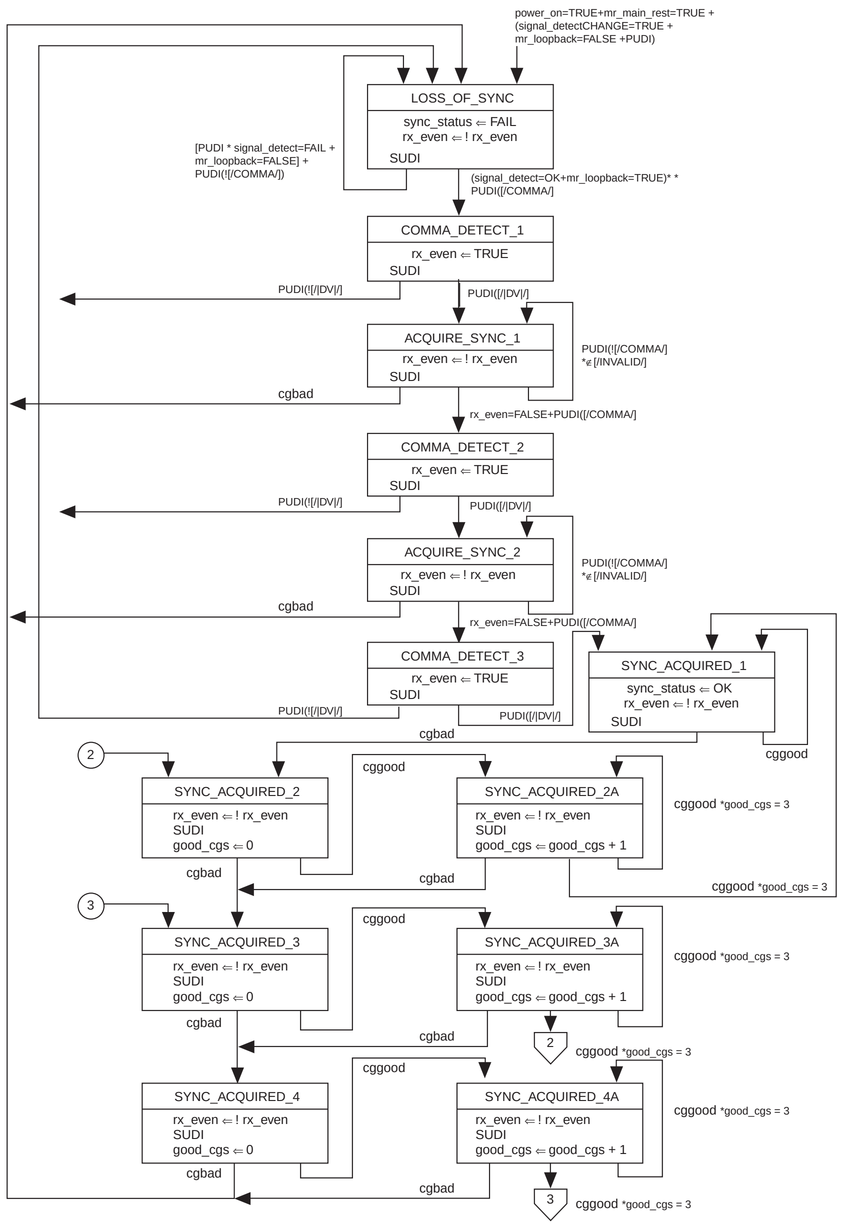
Note to Table 1–36:

(1) The values in the XGMII RXD column are in hexadecimal.

Word Aligner

The word aligner in XAUI functional mode is configured in automatic synchronization state machine mode. The Quartus II software automatically configures the synchronization state machine to indicate synchronization when the receiver acquires four /K28.5/ comma code groups without intermediate invalid code groups. The synchronization state machine implemented in XAUI mode is compliant to the PCS synchronization state diagram specified in Clause 48 of the IEEE P802.3ae specification and is shown in Figure 1–85.

Figure 1–85. IEEE 802.3ae PCS Synchronization State Diagram *(Note 1)*



Note to Figure 1–85:

(1) Source: IEEE P802.3ae™-2002, IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.

Receiver synchronization is indicated on the `rx_syncstatus` port of each channel. A high on the `rx_syncstatus` port indicates that the lane is synchronized; a low on the `rx_syncstatus` port indicates that it has fallen out of synchronization. The receiver loses synchronization when it detects four invalid code groups separated by less than four valid code groups, or when it is reset.

Deskew FIFO

Code groups received across four lanes in a XAUI link can be misaligned with respect to one another because of skew in the physical medium or differences between the independent clock recoveries per lane. The XAUI protocol allows a maximum skew of 40 UI (12.8 ns) as seen at the receiver of the four lanes.

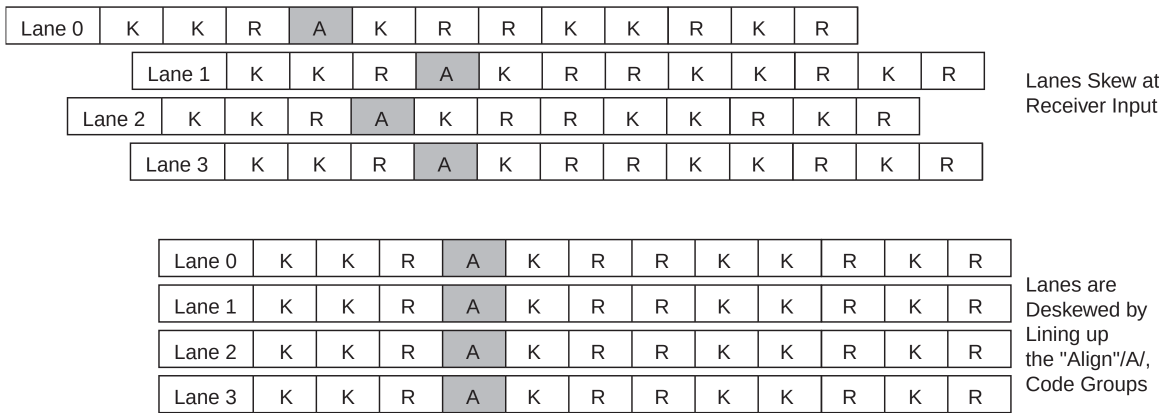
The XAUI protocol requires the physical layer device to implement a deskew circuitry to align all four channels. To enable the deskew circuitry at the receiver to align the four channels, the transmitter sends a /A/ (/K28.3/) code group simultaneously on all four channels during inter-packet gap. The skew introduced in the physical medium and the receiver channels can be /A/ code groups to be received misaligned with respect to each other.

The deskew operation is performed by the deskew FIFO in XAUI functional mode.

The deskew FIFO in each channel receives data from its word aligner. The deskew operation begins only after link synchronization is achieved on all four channels, as indicated by a high on the `rx_syncstatus` signal from the word aligner in each channel. Until the first /A/ code group is received, the deskew FIFO read and write pointers in each channel are not incremented. After the first /A/ code group is received, the write pointer starts incrementing for each word received but the read pointer is frozen. If the /A/ code group is received on each of the four channels within 10 recovered clock cycles of each other, the read pointer of all four deskew FIFOs is released simultaneously, aligning all four channels.

Figure 1–86 shows lane skew at the receiver input and how the deskew FIFO uses the /A/ code group to align the channels.

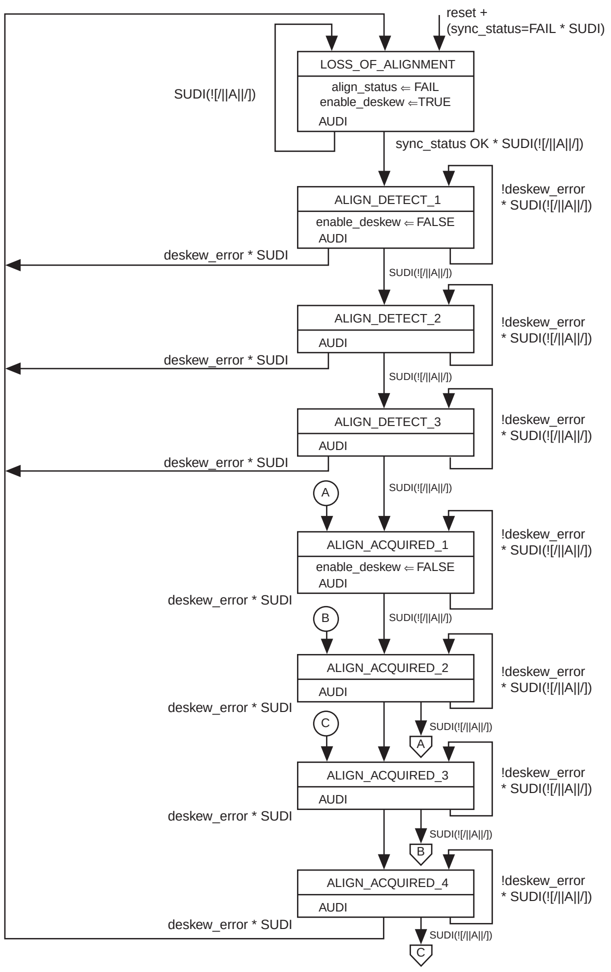
Figure 1–86. Receiver Input Lane Skew in XAUI Mode


Lane 0 K K R A K R R K K R K R
Lane 1 K K R A K R R K K R K R
Lane 2 K K R A K R R K K R K R
Lane 3 K K R A K R R K K R K R
Lanes Skew at Receiver Input
Lane 0 K K R A K R R K K R K R
Lane 1 K K R A K R R K K R K R
Lane 2 K K R A K R R K K R K R
Lane 3 K K R A K R R K K R K R
Lanes are Deskewed by Lining up the "Align"/A/, Code Groups

After alignment of the first ||A|| column, if three additional aligned ||A|| columns are observed at the output of the deskew FIFOs of the four channels, the `rx_channelaligned` signal is asserted high, indicating channel alignment is acquired. After acquiring channel alignment, if four misaligned ||A|| columns are seen at the output of the deskew FIFOs in all four channels with no aligned ||A|| columns in between, the `rx_channelaligned` signal is deasserted low, indicating loss of channel alignment.

The deskew FIFO operation in XAUI functional mode is compliant to the PCS deskew state machine diagram specified in clause 48 of the IEEE P802.3ae, as shown in Figure 1–87.

Figure 1–87. Deskew FIFO in XAUI Mode *(Note 1)*



Note to Figure 1–87:

(1) Source: IEEE P802.3ae™-2002, IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.

Rate Match FIFO

In XAUI mode, the rate match FIFO is capable of compensating up to ±100 PPM (total 200 PPM total) difference between the upstream transmitter and the local receiver reference clock. The XAUI protocol requires the transmitter to send /R/ (/K28.0/) code groups simultaneously on all four lanes (denoted as ||R|| column) during inter-packet gaps, adhering to rules listed in the IEEE P802.3ae specification. The rate match FIFO operation in XAUI mode is compliant to the IEEE P802.3ae specification.

The rate match operation begins after:

- The synchronization state machine in the word aligner of all four channels indicates synchronization acquired by driving its `rx_syncstatus` signal high
- The deskew FIFO indicates alignment acquired by driving the `rx_channelaligned` signal high

The rate match FIFO looks for the ||R|| column (simultaneous /R/ code groups on all four channels) and deletes or inserts ||R|| columns to prevent the rate match FIFO from overflowing or under running. The rate match FIFO can insert or delete as many ||R|| columns as necessary to perform the rate match operation.

Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, that indicate rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric. If an ||R|| column is deleted, the `rx_rmfifodeleted` flag from each of the four channels goes high for one clock cycle per deleted ||R|| column. If an ||R|| column is inserted, the `rx_rmfifoinserted` flag from each of the four channels goes high for one clock cycle per inserted ||R|| column.

Figure 1–88 shows an example of rate match deletion in the case where three ||R|| columns are required to be deleted.

Figure 1–88. Example of Rate Match Deletion in XAUI Mode

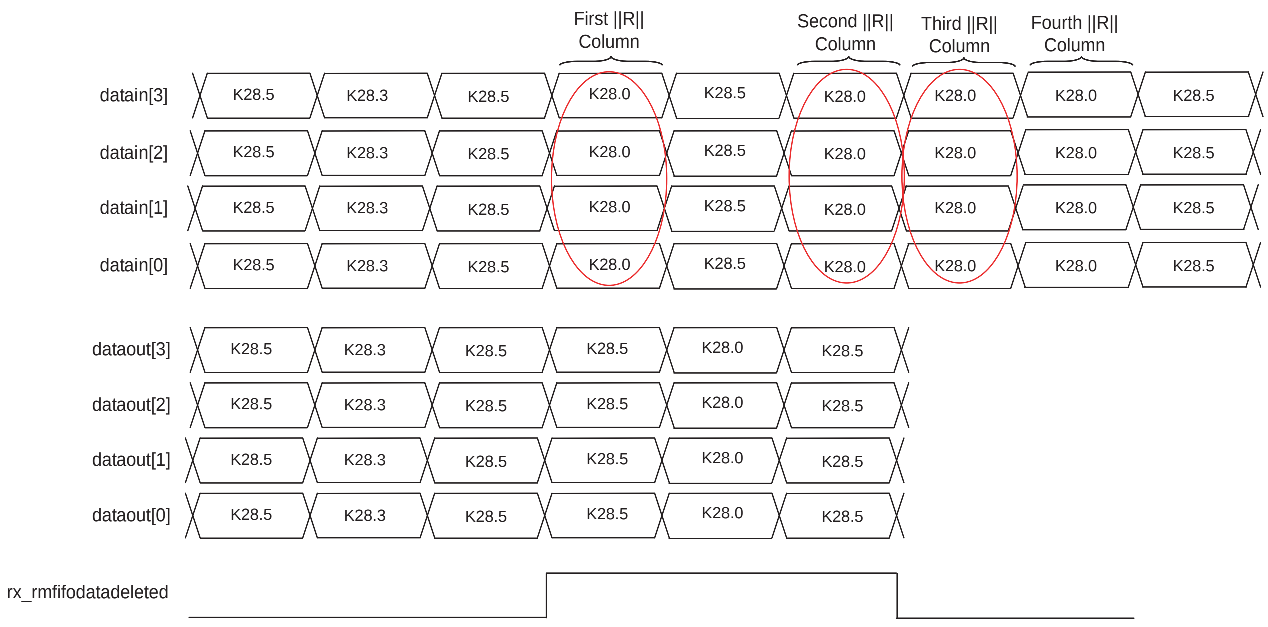
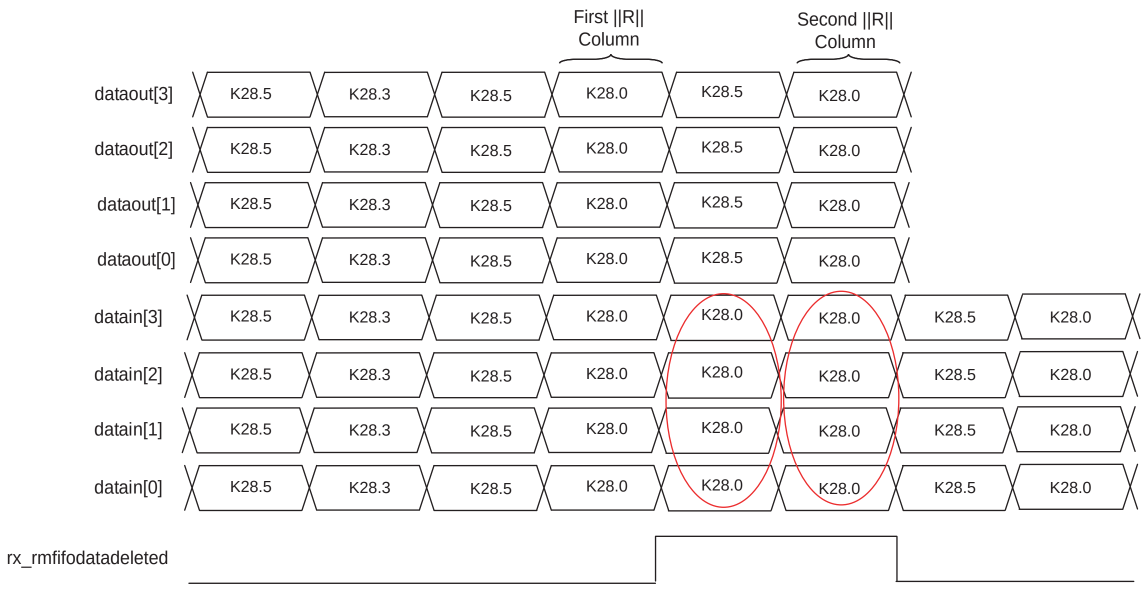


Figure 1–89 shows an example of rate match insertion in the case where two ||R|| columns are required to be inserted.

Figure 1–89. Example of Rate Match Insertion in XAUI Mode



GIGE Mode

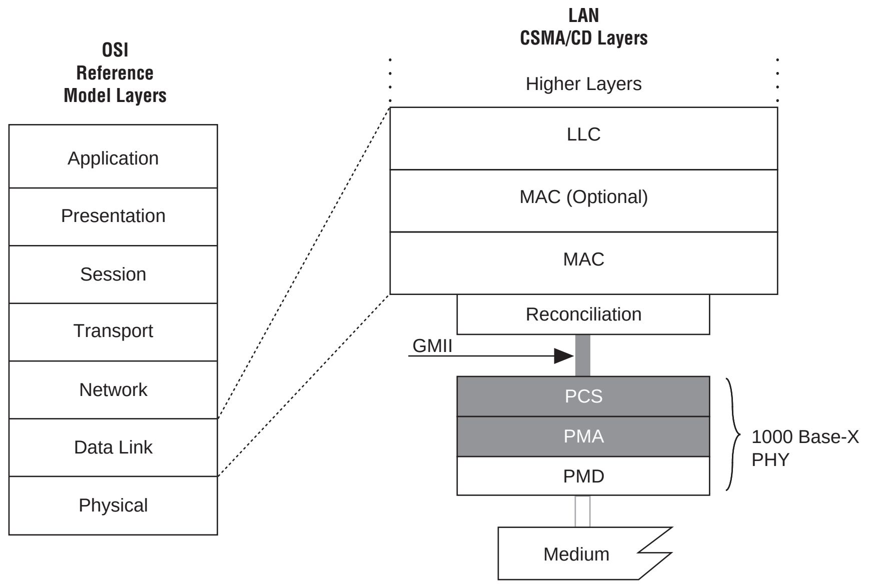
IEEE 802.3 defines the 1000 Base-X PHY as an intermediate, or transition, layer that interfaces various physical media with the media access control (MAC) in a gigabit ethernet system. It shields the MAC layer from the specific nature of the underlying medium. The 1000 Base-X PHY is divided into three sublayers:

- the physical coding sublayer
- the physical media attachment
- the physical medium dependent (PMD)

The PCS sublayer interfaces with the MAC through the gigabit medium independent interface (GMII). The 1000 Base-X PHY defines a physical interface data rate of 1.25 Gbps.

Figure 1–90 shows the 1000 Base-X PHY position in a Gigabit Ethernet OSI reference model.

Figure 1–90. 1000 Base-X PHY in a Gigabit Ethernet OSI Reference Model



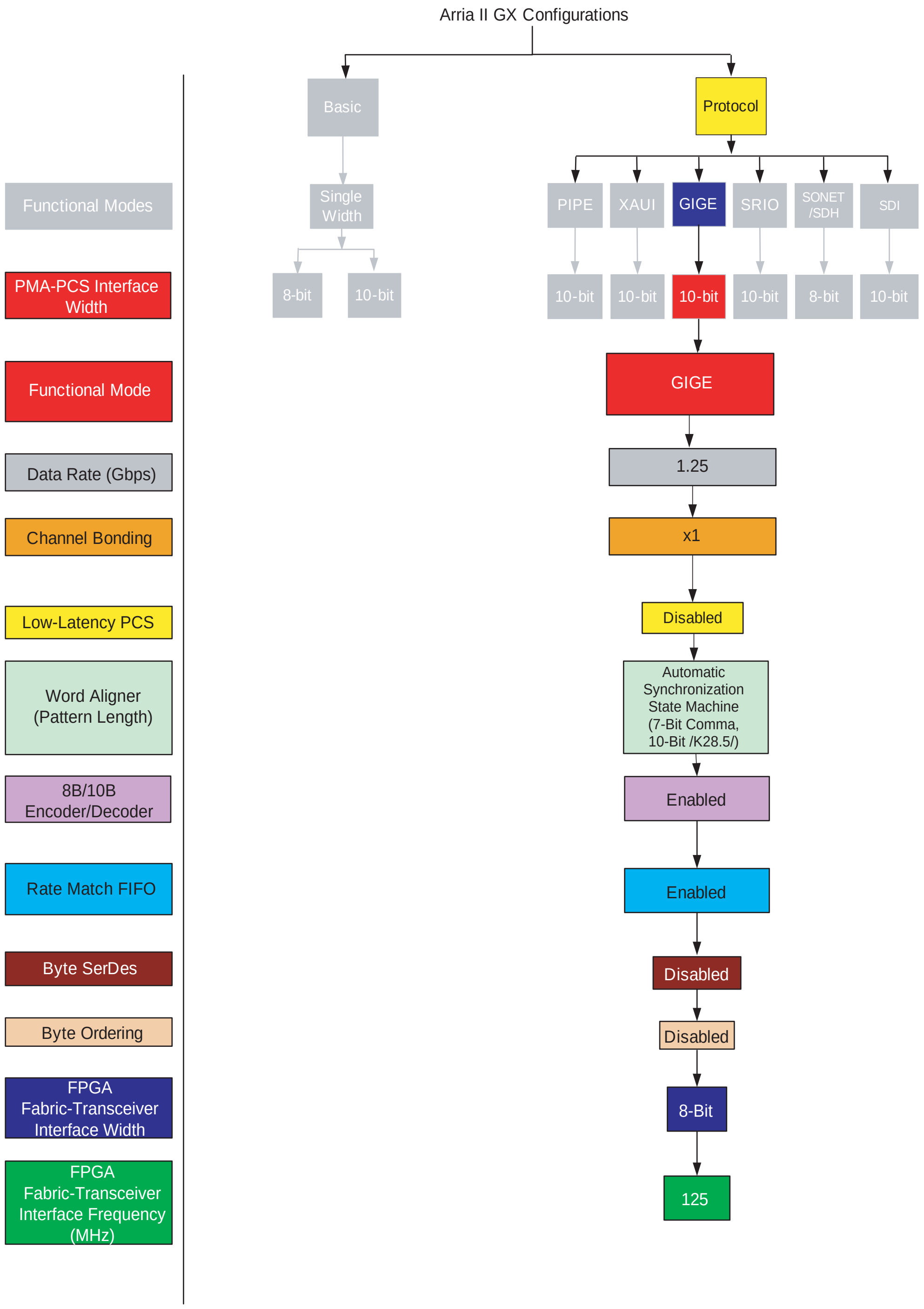
Arria II GX transceivers, when configured in GIGE functional mode, have built-in circuitry to support the following PCS and PMA functions defined in the IEEE 802.3 specification:

- 8B/10B encoding and decoding
- Synchronization
- Upstream transmitter and local receiver clock frequency compensation (rate matching)
- Clock recovery from the encoded data forwarded by the receiver PMD
- Serialization and deserialization

☞ Arria II GX transceivers do not have built-in support for other PCS functions; for example, auto-negotiation state machine, collision-detect, and carrier-sense. If required, you must implement these functions in a PLD logic array or external circuits.

Figure 1–91 shows the GIGE mode configuration supported in Arria II GX devices.

Figure 1–91. GIGE Mode



GIGE Mode Datapath

Figure 1–92 shows the transceiver datapath when configured in GIGE functional mode.

Figure 1–92. GIGE Mode Datapath

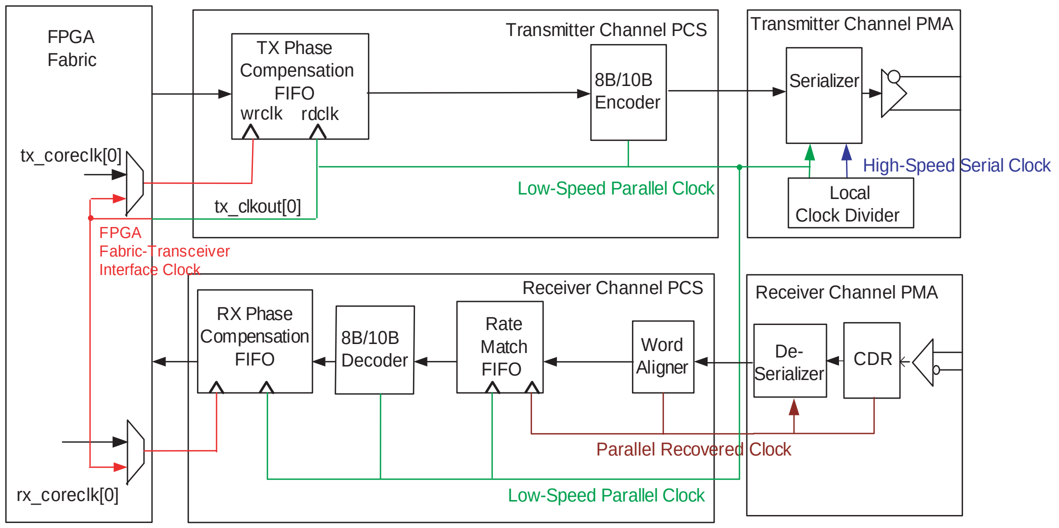


Table 1–37 shows the transceiver datapath clock frequencies in GIGE functional mode.

Table 1–37. Transceiver Datapath Clock Frequencies in GIGE Mode

Functional Mode	Data Rate	High-Speed Serial Clock Frequency	Parallel Recovered Clock and Low-Speed Parallel Clock Frequency	FPGA Fabric-Transceiver Interface Clock Frequency
GIGE	1.25 Gbps	625 MHz	125 MHz	125 MHz

8B/10B Encoder

In GIGE mode, the 8B/10B encoder clocks in 8-bit data and a 1-bit control identifier from the transmitter phase compensation FIFO and generates 10-bit encoded data. The 10-bit encoded data is fed to the serializer. Refer to "8B/10B Encoder" on page 1–32 for more information about 8B/10B encoder functionality.

GIGE Protocol—Ordered Sets and Special Code Groups

Table 1–38 lists ordered sets and special code groups specified in the IEEE802.3 specification.

Table 1–38. GIGE Ordered Sets (Part 1 of 2)

Code	Ordered Set	Number of Code Groups	Encoding
/C/	Configuration	—	Alternating /C1/ and /C2/
/C1/	Configuration 1	4	/K28.5/D21.5/`Config_Reg` *(1)*
/C2/	Configuration 2	4	/K28.5/D2.2/`Config_Reg` *(1)*

Table 1–38. GIGE Ordered Sets (Part 2 of 2)

Code	Ordered Set	Number of Code Groups	Encoding
/I/	IDLE	—	Correcting /I1/, Preserving /I2/
/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 1–38:

(1) Two data code groups represent the `Config_Reg` value.

Idle Ordered-Set Generation

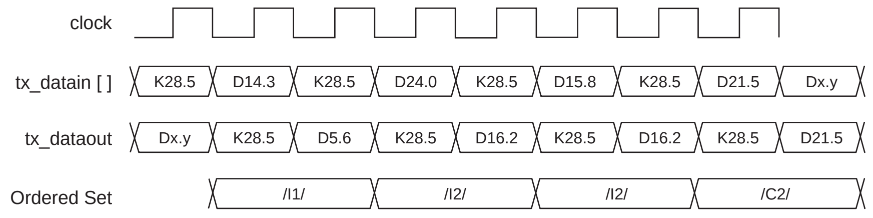
The IEEE 802.3 specification requires the GIGE PHY to transmit idle ordered sets (/I/) continuously and repetitively whenever the GMII is idle. This ensures that the receiver maintains bit and word synchronization whenever there is no active data to be transmitted.

In GIGE functional mode, any /Dx.y/ following a /K28.5/ comma is replaced by the transmitter with either a /D5.6/ (/I1/ ordered set) or a /D16.2/ (/I2/ ordered set), depending on the current running disparity. The exception is when the data following the /K28.5/ is /D21.5/ (/C1/ ordered set) or /D2.2/ (/C2/) ordered set. If the running disparity before the /K28.5/ is positive, an /I1/ ordered set is generated. If the running disparity is negative, a /I2/ ordered set is generated. The disparity at the end of a /I1/ is the opposite of that at the beginning of the /I1/. The disparity at the end of a /I2/ is the same as the beginning running disparity (right before the idle code). This ensures a negative running disparity at the end of an idle ordered set. A /Kx.y/ following a /K28.5/ is not replaced.

Note that /D14.3/, /D24.0/, and /D15.8/ are replaced by /D5.6/ or /D16.2/ (for /I1/, /I2/ ordered sets). /D21.5/ (part of the /C1/ order set) is not replaced.

Figure 1–93 shows the automatic idle ordered set generation.

Figure 1–93. Example of Automatic Ordered Set Generation



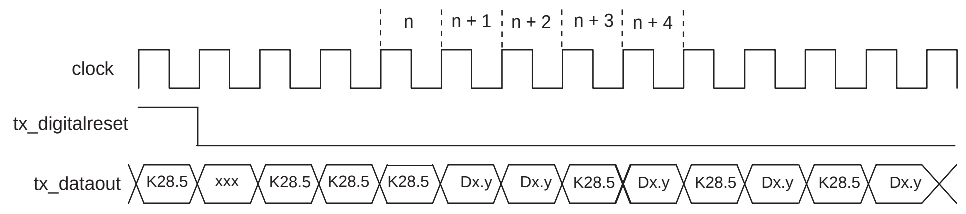
Reset Condition

After de-assertion of `tx_digitalreset`, the GIGE transmitter automatically transmits three /K28.5/ comma code groups before transmitting user data on the `tx_datain` port. This could affect the synchronization state machine behavior at the receiver.

Depending on when you start transmitting the synchronization sequence, there could be an even or odd number of /Dx.y/ code groups transmitted between the last of the three automatically sent /K28.5/ code groups and the first /K28.5/ code group of the synchronization sequence. If there is an even number of /Dx.y/ code groups received between these two /K28.5/ code groups, the first /K28.5/ code group of the synchronization sequence begins at an odd code group boundary (`rx_even` = FALSE). An IEEE802.3-compliant GIGE synchronization state machine treats this as an error condition and goes into Loss of Sync state.

Figure 1–94 shows an example of even numbers of /Dx.y/ between the last automatically sent /K28.5/ and the first user-sent /K28.5/. The first user-sent /K28.5/ code group received at an odd code group boundary in cycle n + 3 takes the receiver synchronization state machine in Loss of Sync state. The first synchronization ordered-set /K28.5/Dx.y/ in cycles n + 3 and n + 4 is discounted and three additional ordered sets are required for successful synchronization.

Figure 1–94. Example of Reset Condition in GIGE Mode



Word Aligner

The word aligner in GIGE functional mode is configured in automatic synchronization state machine mode. The Quartus II software automatically configures the synchronization state machine to indicate synchronization when the receiver acquires three consecutive synchronization ordered sets. A synchronization ordered set is a /K28.5/ code group followed by an odd number of valid /Dx.y/ code groups. The fastest way for the receiver to achieve synchronization is to receive three continuous {/K28.5/, /Dx.y/} ordered set.

Receiver synchronization is indicated on the `rx_syncstatus` port of each channel. A high on the `rx_syncstatus` port indicates that the lane is synchronized; a low on the `rx_syncstatus` port indicates that the lane has fallen out of synchronization. The receiver loses synchronization when it detects four invalid code groups separated by less than three valid code groups, or when it is reset.

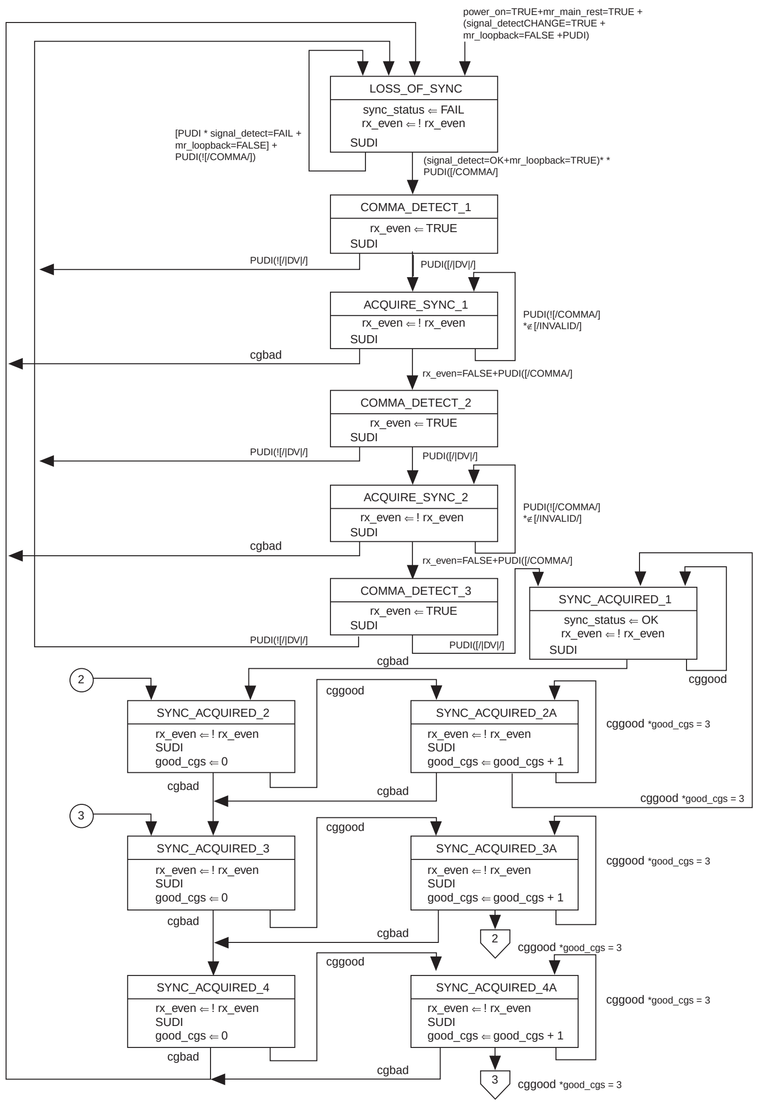
Table 1–39 lists the synchronization state machine parameters when configured in GIGE mode.

Table 1–39. Synchronization State Machine Parameters in GIGE Functional Mode

Synchronization State Machine Parameters	Setting
Number of valid {/K28.5/, /Dx,y/} ordered-sets received to achieve synchronization	3
Number of errors received to lose synchronization	4
Number of continuous good code groups received to reduce the error count by 1	4

Figure 1–95 shows the synchronization state machine implemented in GIGE mode.

Figure 1–95. Synchronization State Machine in GIGE Mode *(Note 1)*



Note to Figure 1–95:

(1) Source: IEEE P802.3ae™-2002, IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.

Rate Match FIFO

In GIGE mode, the rate match FIFO is capable of compensating up to ±100 PPM (total 200 PPM total) difference between the upstream transmitter and the local receiver reference clock. The GIGE protocol requires the transmitter to send idle ordered sets /I1/ (/K28.5/D5.6/) and /I2/ (/K28.5/D16.2/) during inter-packet gaps, adhering to the rules listed in the IEEE 802.3 specification.

The rate match operation begins after the synchronization state machine in the word aligner indicates synchronization is acquired by driving the `rx_syncstatus` signal high. The rate matcher deletes or inserts both symbols (/K28.5/ and /D16.2/) of the /I2/ ordered sets, even if it requires deleting only one symbol to prevent the rate match FIFO from overflowing or under running. It can insert or delete as many /I2/ ordered sets as necessary to perform the rate match operation.

Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, indicating rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric. Both the `rx_rmfifodatadeleted` and `rx_rmfifodatainserted` flags are asserted for two clock cycles for each deleted and inserted /I2/ ordered-set, respectively.

Figure 1–96 shows an example of rate match FIFO deletion in the case where three symbols are required to be deleted. Because the rate match FIFO can only delete /I2/ ordered set, it deletes two /I2/ ordered sets (four symbols deleted).

Figure 1–96. Example of Rate Match Deletion in GIGE Mode

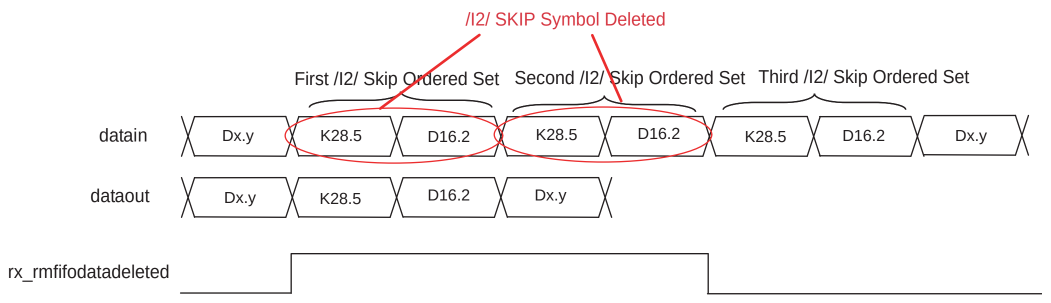
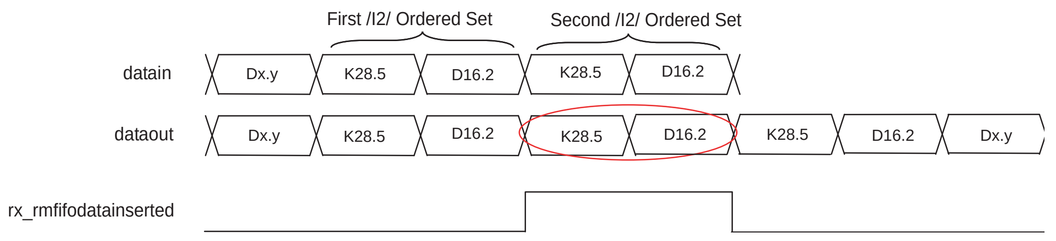


Figure 1–97 shows an example of rate match FIFO insertion in the case where one symbol is required to be inserted. Because the rate match FIFO can only delete /I2/ ordered set, it inserts one /I2/ ordered sets (two symbols inserted).

Figure 1–97. Example of Rate Match Insertion in GIGE Mode



SONET/SDH Mode

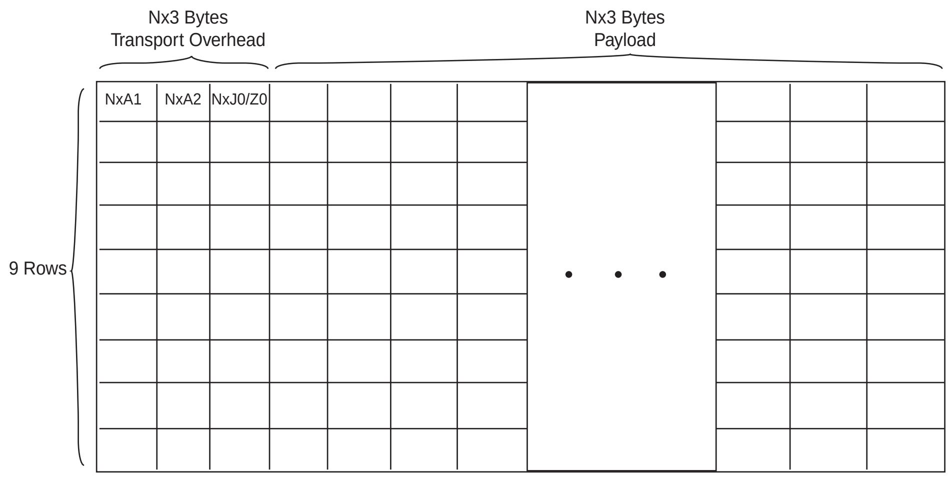
SONET/SDH is one of the most common serial-interconnect protocols used in backplanes deployed in communications and telecom applications. SONET/SDH defines various optical carrier (OC) subprotocols for carrying signals of different capacities through a synchronous optical hierarchy.

SONET/SDH Frame Structure

Base OC-1 frames are byte-interleaved to form SONET/SDH frames. For example, 12 OC-1 frames are byte-interleaved to form 1 OC-12 frame; 48 OC-1 frames are byte-interleaved to form 1 OC-48 frame and so on. SONET/SDH frame sizes are constant, with a frame transfer rate of 125 µs.

Figure 1–98 shows the SONET/SDH frame structure.

Figure 1–98. SONET/SDH Mode



Transport overhead bytes A1 and A2 are used for restoring frame boundary from the serial data stream. Frame sizes are fixed, so the A1 and A2 bytes appear within the serial data stream every 125 µs . In an OC-12 system, 12 A1 bytes are followed by 12 A2 bytes. Similarly, in an OC-48 system, 48 A1 bytes are followed by 48 A2 bytes.

In SONET/SDH systems, byte values of A1 and A2 are fixed as follows:

- A1 = 11110110 or 8'hF6
- A2 = 00101000 or 8'h28

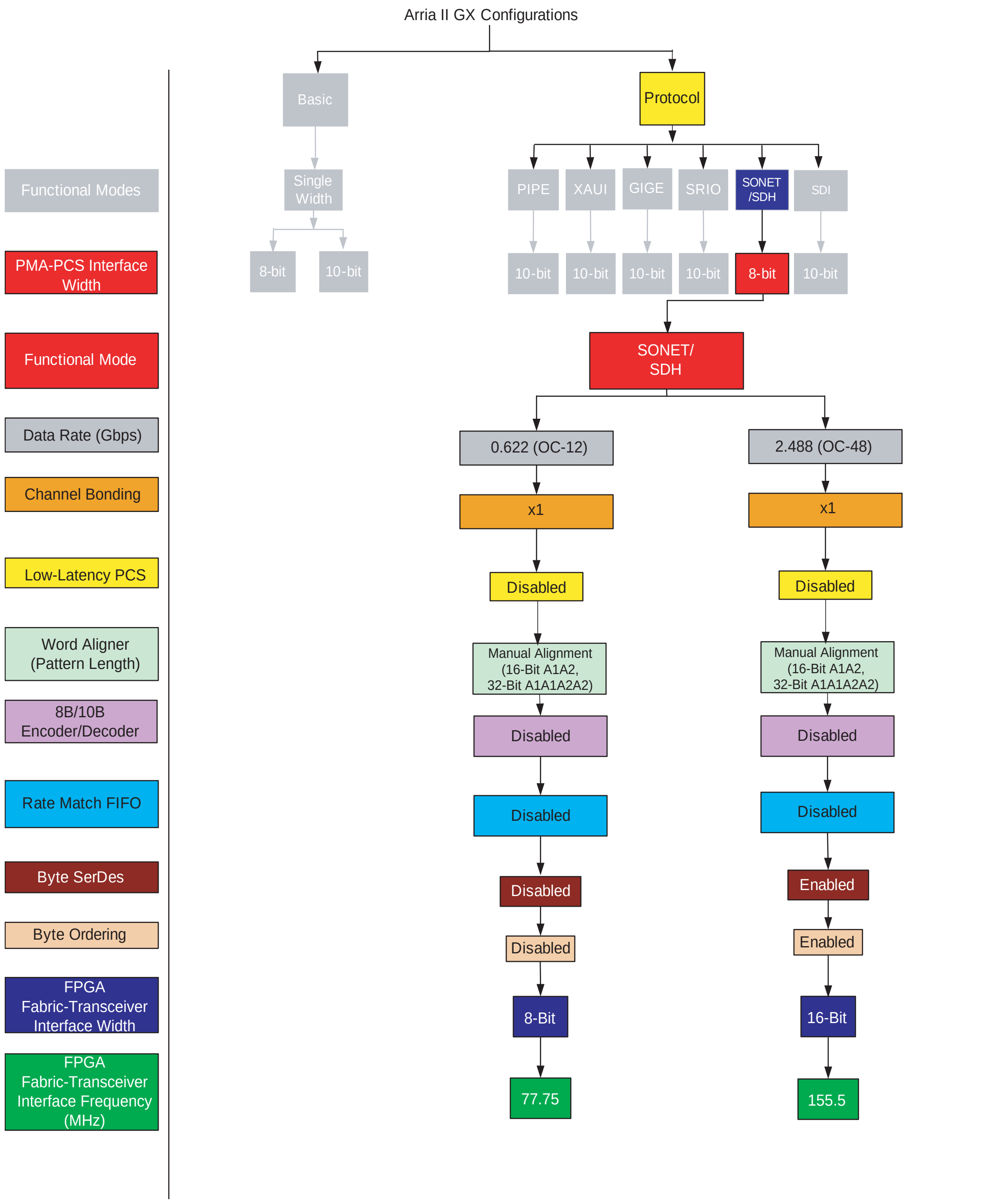
You can employ Arria II GX transceivers as physical layer devices in a SONET/SDH system. These transceivers provide support for SONET/SDH protocol-specific functions and electrical features; for example, alignment to A1A2 or A1A1A2A2 pattern.

Arria II GX transceivers are designed to support the following three SONET/SDH subprotocols:

- OC-12 at 622 Mbps with 8-bit channel width
- OC-48 at 2488.32 Mbps with 16-bit channel width

Figure 1–99 shows SONET/SDH mode configurations supported in Arria II GX devices.

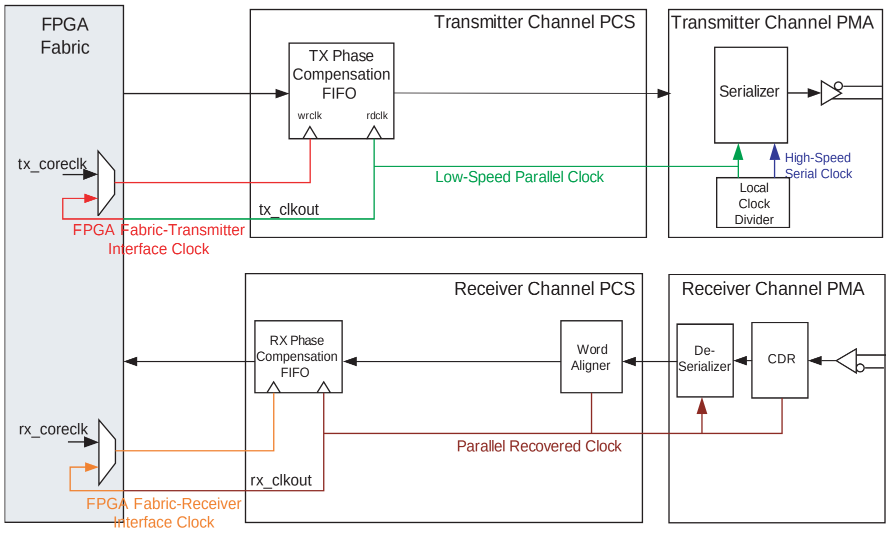
Figure 1–99. SONET/SDH Mode Configurations in Arria II GX Devices



SONET/SDH OC-12 Datapath

Figure 1–100 shows the transceiver datapath when configured in SONET/SDH OC-12 mode.

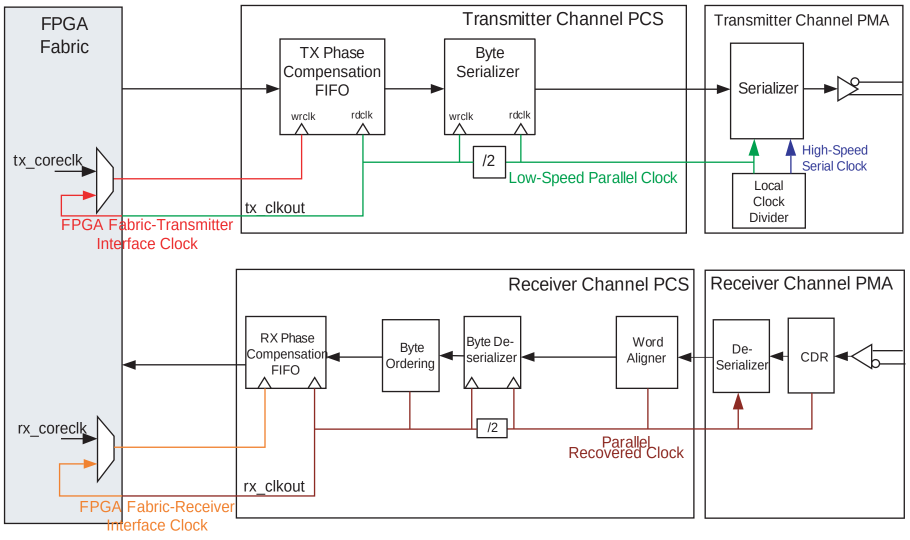
Figure 1–100. SONET/SDH OC-12 Datapath



SONET/SDH OC-48 Datapath

Figure 1–101 shows the transceiver datapath when configured in SONET/SDH OC-48 mode.

Figure 1–101. SONET/SDH OC-48 Datapath



SONET/SDH Transmission Bit Order

Unlike Ethernet, where the LSB of the parallel data byte is transferred first, SONET/SDH requires the MSB to be transferred first and the LSB to be transferred last. To facilitate the MSB-to-LSB transfer, you must enable the following options in the ALTGX MegaWizard Plug-In Manager:

- **Flip transmitter input data bits**
- **Flip receiver output data bits**

Depending on whether data bytes are transferred MSB-to-LSB or LSB-to-MSB, you must select the appropriate word aligner settings in the ALTGX MegaWizard Plug-In Manager. Table 1–40 lists the correct word aligner settings for each bit transmission order.

Word Alignment

The word aligner in SONET/SDH OC-12 and OC-48 modes is configured in manual alignment mode as described in "Word Aligner in Single-Width Mode with 8-Bit PMA-PCS Interface Modes" on page 1–59.

In OC-12 and OC-48 configurations, you can configure the word aligner to either align to a 16-bit A1A2 pattern or a 32-bit A1A1A2A2 pattern. This is controlled by the `rx_a1a2size` input port to the transceiver. A low level on the `rx_a1a2size` port configures the word aligner to align to a 16-bit A1A2 pattern; a high level on the `rx_a1a2size` port configures the word aligner to align to a 32-bit A1A1A2A2 pattern.

You can configure the word aligner to flip the alignment pattern bits programmed in the wizard and compare them with the incoming data for alignment. This feature offers flexibility to the SONET backplane system for either a MSBit-to-LSBit or LSBit-to-MSBit data transfer. Table 1–40 lists word alignment patterns that you must program in the ALTGX MegaWizard Plug-In Manager based on the bit-transmission order and the word aligner bit-flip option.

Table 1–40. Word Aligner Settings

Serial Bit Transmission Order	Word Alignment Bit Flip	Word Alignment Pattern
MSBit-to-LSBit	On	1111011000101000 (16'hF628)
MSBit-to-LSBit	Off	0001010001101111 (16'h146F)
LSBit-to-MSBit	Off	0010100011110110 (16'h28F6)

The behavior of the SONET/SDH word aligner control and status signals along with an operational timing diagram are explained in "Word Aligner in Single-Width Mode with 8-Bit PMA-PCS Interface Modes" on page 1–59.

OC-48 Byte Serializer and Deserializer

The OC-48 transceiver datapath includes the byte serializer and deserializer to allow the PLD interface to run at a lower speed. The OC-12 configuration does not use the byte serializer and deserializer blocks.

The byte serializer and deserializer blocks are explained in "Byte Serializer" on page 1–31 and "Byte Deserializer" on page 1–80, respectively.

The OC-48 byte serializer converts 16-bit data words from the FPGA fabric and translates the 16-bit data words into two 8-bit data bytes at twice the rate. The OC-48 byte deserializer takes in two consecutive 8-bit data bytes and translates them into a 16-bit data word to the FPGA fabric at half the rate.

OC-48 Byte Ordering

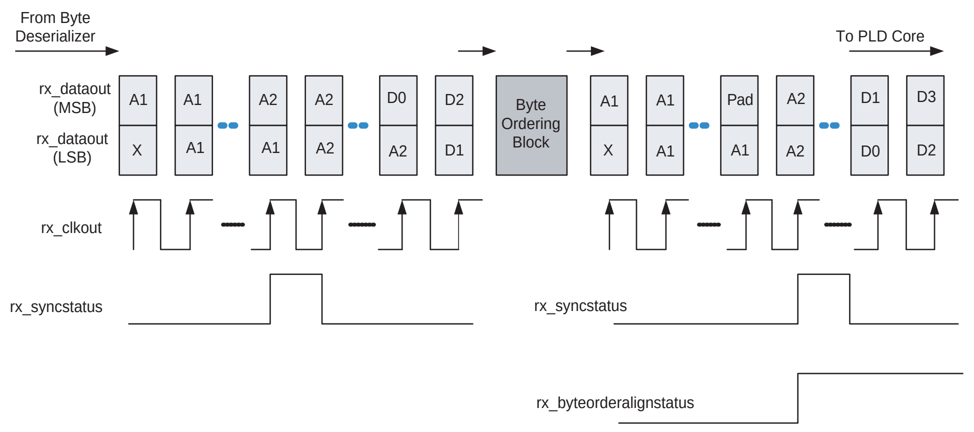
Because of byte deserialization, the most significant byte of a word might appear at the `rx_dataout` port along with the least significant byte of the next word.

In an OC-48 configuration, the byte ordering block is built into the datapath and can be used to perform byte ordering. Byte ordering in an OC-48 configuration is automatic, as explained in "Word-Alignment-Based Byte Ordering" on page 1–82.

In automatic mode, the byte ordering block is triggered by the rising edge of the `rx_syncstatus` signal. As soon as the byte ordering block sees the rising edge of the `rx_syncstatus` signal, it compares the least significant byte coming out of the byte deserializer with the A2 byte of the A1A2 alignment pattern. If the least significant byte coming out of the byte deserializer does not match the A2 byte set in the ALTGX MegaWizard Plug-In Manager, the byte ordering block inserts a PAD character, as seen in Figure 1–102. Insertion of this PAD character enables the byte ordering block to restore the correct byte order.

The PAD character is defaulted to the A1 byte of the A1A2 alignment pattern.

Figure 1–102. OC-48 Byte Ordering in Automatic Mode



SDI Mode

The Society of Motion Picture and Television Engineers (SMPTE) defines various SDI standards for transmission of uncompressed video.

The following three SMPTE standards are popular in video broadcasting applications:

- SMPTE 259M standard—more popularly known as the standard-definition (SD) SDI, is defined to carry video data at 270 Mbps.
- SMPTE 292M standard—more popularly known as the high-definition (HD) SDI, is defined to carry video data at either 1485 Mbps or 1483.5 Mbps.

- SMPTE 424M standard—more popularly known as the third-generation (3G) SDI, is defined to carry video data at either 2970 Mbps or 2967 Mbps.

You can configure Arria II GX transceivers in HD-SDI or 3G-SDI configuration using the ALTGX MegaWizard Plug-In Manager.

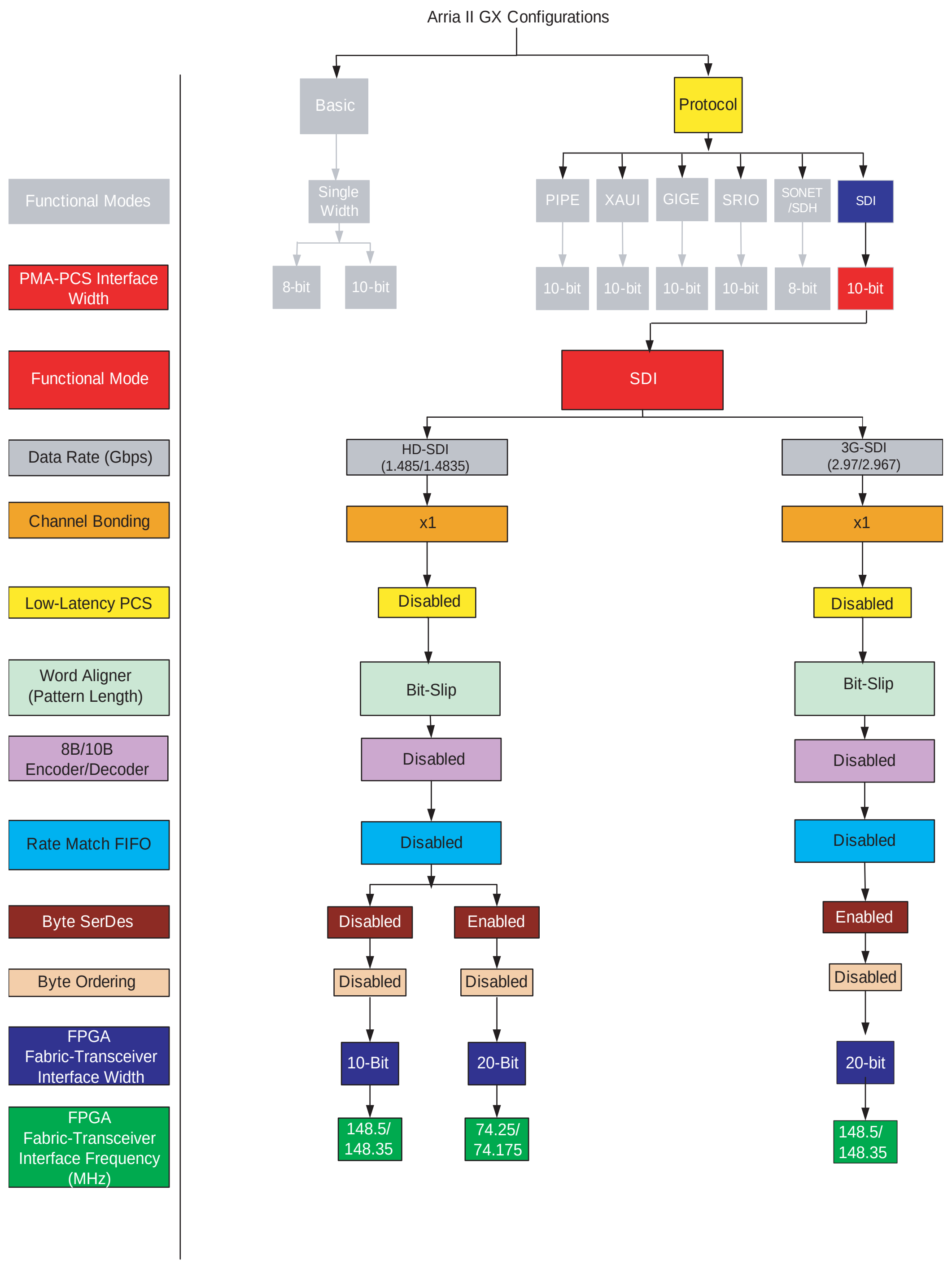
Table 1–41 shows ALTGX configurations supported by the Arria II GX transceivers in SDI mode.

Table 1–41. ALTGX Configurations in SDI Mode

Configuration	Data Rate (Mbps)	refclk Frequencies (MHz)	FPGA Fabric-Transceiver Interface Width
HD	1485	74.25, 148.5	10-bit, 20-bit
	1483.5	74.175, 148.35	10-bit, 20-bit
3G	2970	148.5, 297	Only 20-bit interface allowed in 3G
	2967	148.35, 296.7	Only 20-bit interface allowed in 3G

Figure 1–103 shows SDI mode configurations supported in Arria II GX devices.

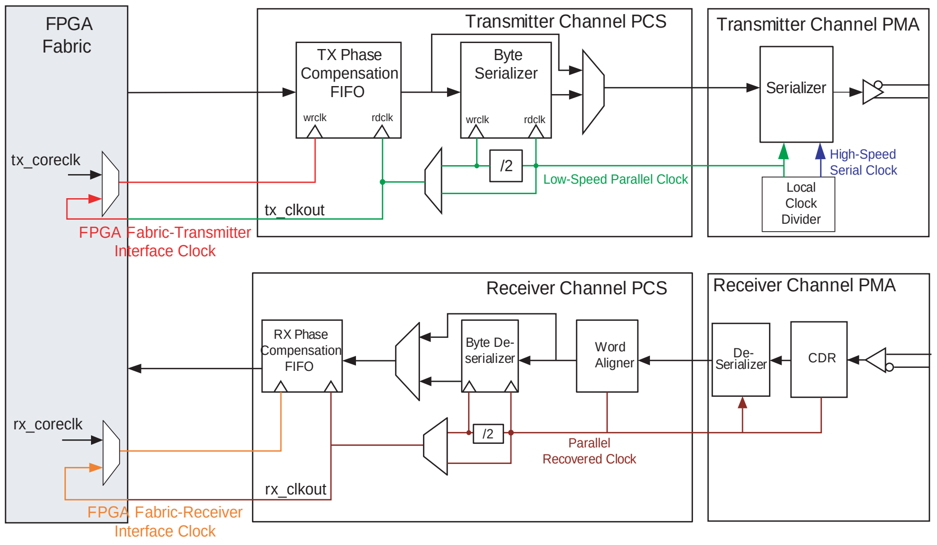
Figure 1–103. SDI Mode



SDI Mode Datapath

Figure 1–104 shows the transceiver datapath when configured in SDI mode.

Figure 1–104. SDI Mode Datapath



Transmitter Datapath

The transmitter datapath, in HD-SDI configuration with 10-bit wide FPGA fabric-transceiver interface, consists of the transmitter phase compensation FIFO and the 10:1 serializer. The transmitter datapath, in HD-SDI and 3G-SDI configurations with 20-bit wide FPGA fabric-transceiver interface, also includes the byte serializer.

In SDI mode, the transmitter is purely a parallel-to-serial converter. SDI transmitter functions, such as scrambling and cyclical redundancy check (CRC) code generation, must be implemented in the FPGA logic array.

Receiver Datapath

In the 10-bit channel width SDI configuration, the receiver datapath consists of the clock recovery unit (CRU), 1:10 deserializer, word aligner in bit-slip mode, and receiver phase compensation FIFO. In the 20-bit channel width SDI configuration, the receiver datapath also includes the byte deserializer.

SDI receiver functions, such as de-scrambling, framing, and CRC checker, must be implemented in the FPGA logic array.

Receiver Word Alignment and Framing

In SDI systems, the word aligner in the receiver datapath is not useful because word alignment and framing happens after de-scrambling. Altera recommends driving the ALTGXB megafunction `rx_bitslip` signal low to avoid the word aligner from inserting bits in the received data stream.

Serial RapidIO Mode

The RapidIO Trade Association defines a high-performance, packet-switched interconnect standard to pass data and control information between microprocessors, digital signal, communications, and network processors, system memories, and peripheral devices.

Serial RapidIO physical layer specification defines three line rates:

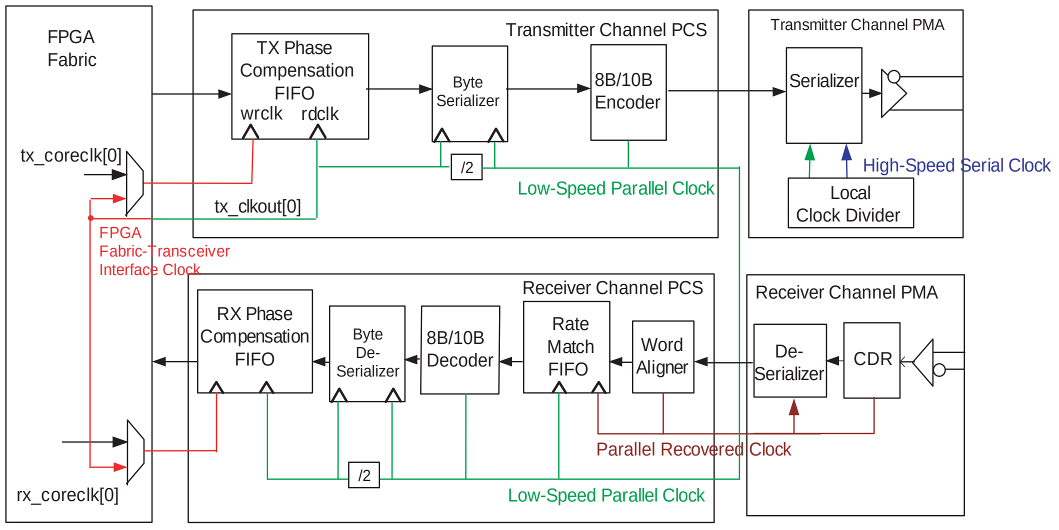
- 1.25 Gbps
- 2.5 Gbps
- 3.125 Gbps

It also defines two link widths—single-lane (1×) and bonded four-lane (4×) at each line rate.

Arria II GX transceivers support only single-lane (1×) configuration at all three line rates. Four 1× channels configured in Serial RapidIO mode can be instantiated to achieve a 4× Serial RapidIO link. The four transmitter channels in this 4× Serial RapidIO link are not bonded. The four receiver channels in this 4× Serial RapidIO link do not have lane alignment or deskew capability.

Figure 1–105 shows the ALTGX transceiver data path when configured in Serial RapidIO mode.

Figure 1–105. Serial RapidIO Mode Datapath

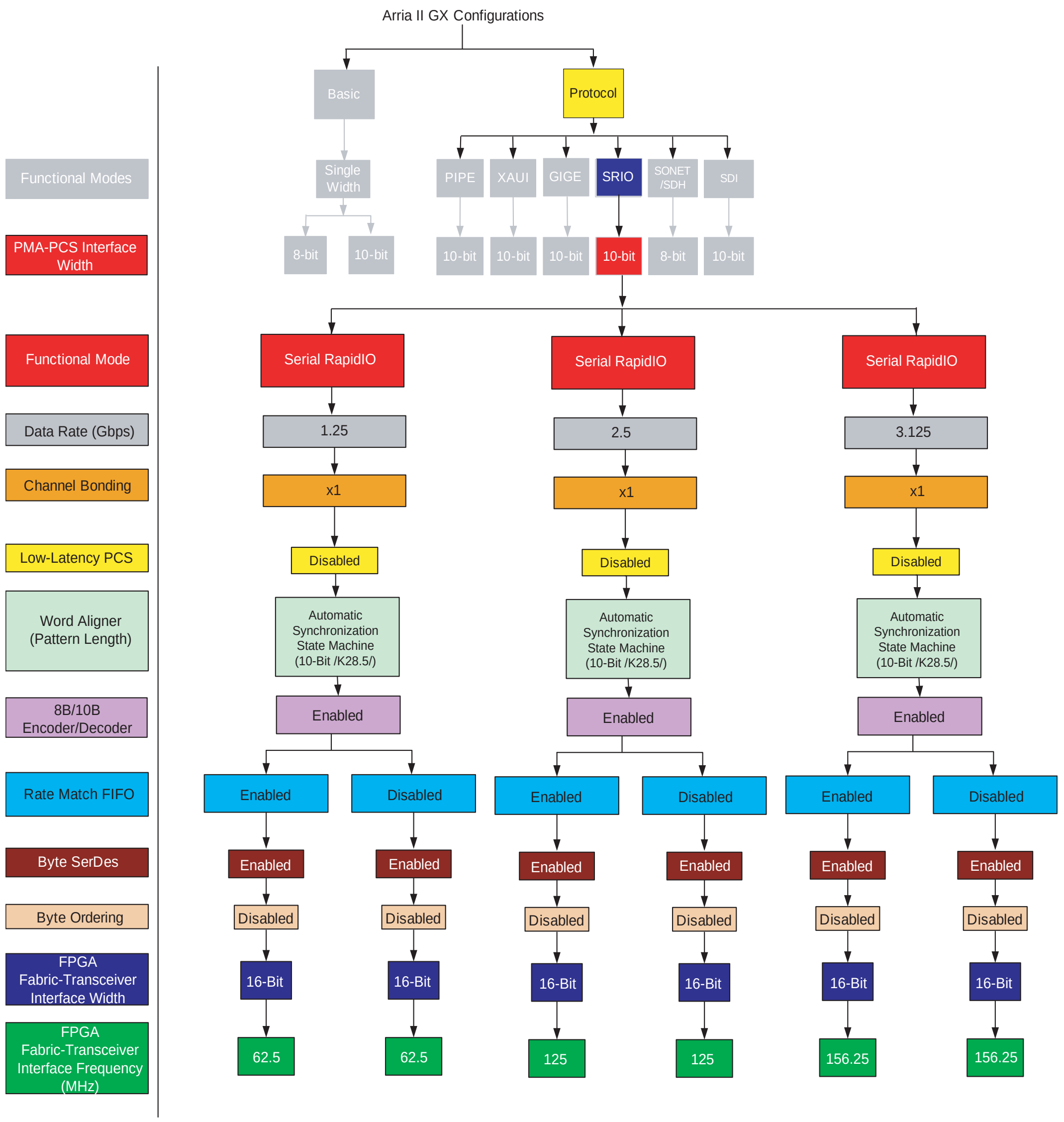


Arria II GX transceivers, when configured in Serial RapidIO functional mode, provide the following PCS and PMA functions:

- 8B/10B encoding/decoding
- Word alignment
- Lane Synchronization State Machine
- Clock recovery from the encoded data
- Serialization/deserialization

Figure 1–106 shows the Serial RapidIO mode configuration supported in Arria II GX devices.

Figure 1–106. Serial RapidIO Mode



Arria II GX transceivers do not have built-in support for other PCS functions; for example, pseudo-random idle sequence generation and lane alignment in 4× mode. Depending on your system requirements, you must implement these functions in the logic array or external circuits.

Synchronization State Machine

In Serial RapidIO mode, the ALTGX MegaWizard Plug-In Manager defaults the word alignment pattern to K28.5. The word aligner has a synchronization state machine that handles the receiver lane synchronization.

The ALTGX MegaWizard Plug-In Manager automatically defaults the synchronization state machine to indicate synchronization when the receiver acquires 127 K28.5 (10'b0101111100 or 10'b1010000011) synchronization code groups without receiving an intermediate invalid code group. Once synchronized, the state machine indicates loss of synchronization when it detects three invalid code groups separated by less than 255 valid code groups, or when it is reset.

Receiver synchronization is indicated on the `rx_syncstatus` port of each channel. A high on the `rx_syncstatus` port indicates that the lane is synchronized and a low indicates that it has fallen out of synchronization.

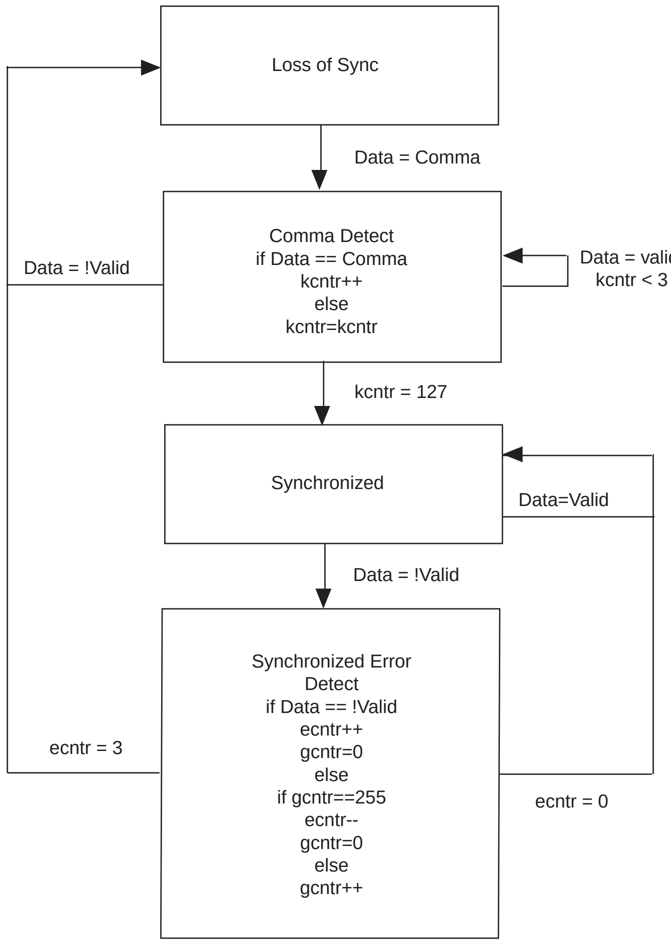
Table 1–42 lists the ALTGX megafunction synchronization state machine parameters when configured in Serial RapidIO mode.

Table 1–42. Synchronization State Machine Parameters in Serial RapidIO Mode

Parameters	Number
Number of valid K28.5 code groups received to achieve synchronization.	127
Number of errors received to lose synchronization.	3
Number of continuous good code groups received to reduce the error count by one.	255

Figure 1–107 gives a conceptual view of the synchronization state machine implemented in Serial RapidIO functional mode.

Figure 1–107. Synchronization State Machine in Serial RapidIO Mode



Loopback Modes

Arria II GX devices provide various loopback options that allow you to verify the working of different functional blocks in the transceiver channel. The available loopback options are:

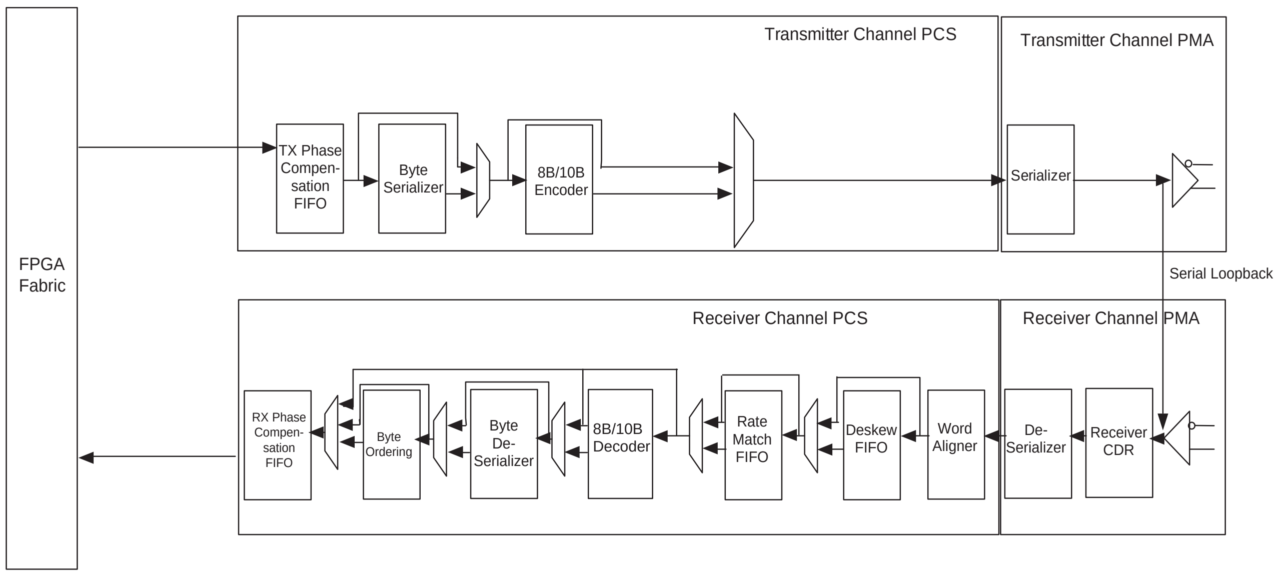
- Serial loopback—available in all functional modes except PCI Express (PIPE) mode
- Reverse serial loopback—available in Basic mode only
- Reverse serial pre-CDR loopback—available in Basic mode only
- PCI Express (PIPE) reverse parallel loopback—supported in PCI Express (PIPE) protocol only

Serial Loopback

The serial loopback option is available for all functional modes except PCI Express (PIPE) mode. Figure 1–108 shows the datapath for serial loopback. The data from the FPGA fabric passes through the transmitter channel and gets looped back to the receiver channel, bypassing the receiver buffer. The received data is available to the FPGA logic for verification. Using this option, you can check the working for all enabled PCS and PMA functional blocks in the transmitter and receiver channels. When you enable the serial loopback option, the ALTGX MegaWizard Plug-In Manager provides the `rx_seriallpbken` port to dynamically enable serial loopback on a channel-by-channel basis. Set the `rx_seriallpbken` signal to logic high to enable serial loopback.

When serial loopback is enabled, the transmitter channel sends the data to both the `tx_dataout` output port and the receiver channel. The differential output voltage on the `tx_dataout` ports is based on the selected V_{OD} settings. The looped back data is received by the receiver CDR and is retimed through different clock domains. You must provide an alignment pattern for the word aligner to enable the receiver channel to retrieve the byte boundary.

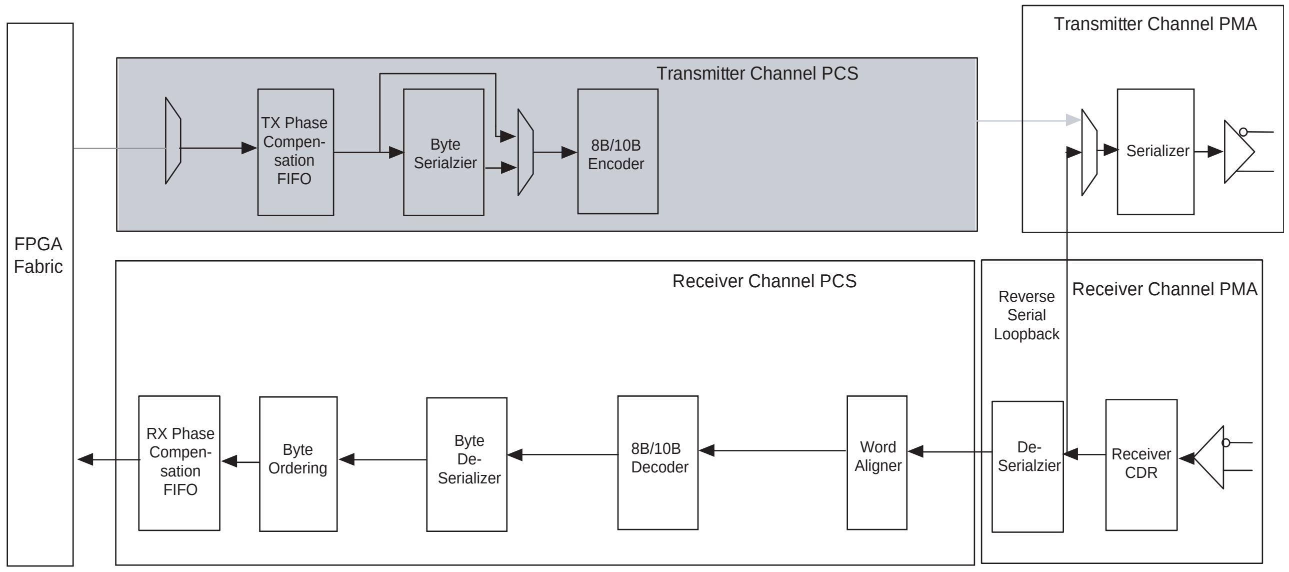
Figure 1–108. Serial Loopback Datapath



Reverse Serial Loopback

Reverse serial loopback is available as a subprotocol under Basic functional mode. In reverse serial loopback mode, the data is received through the `rx_datain` port, retimed through the receiver CDR, and sent out to the `tx_dataout` port. The received data is also available to the FPGA logic. Figure 1–109 shows the transceiver channel datapath for reverse serial loopback mode. Note that the active block of the transmitter channel is only the transmitter buffer. You can change the output differential voltage on the transmitter buffer through the ALTGX MegaWizard Plug-In Manager. You cannot alter the pre-emphasis settings for the transmitter buffer. Reverse serial loopback is often implemented when using a bit error rate tester (BERT) on the upstream transmitter.

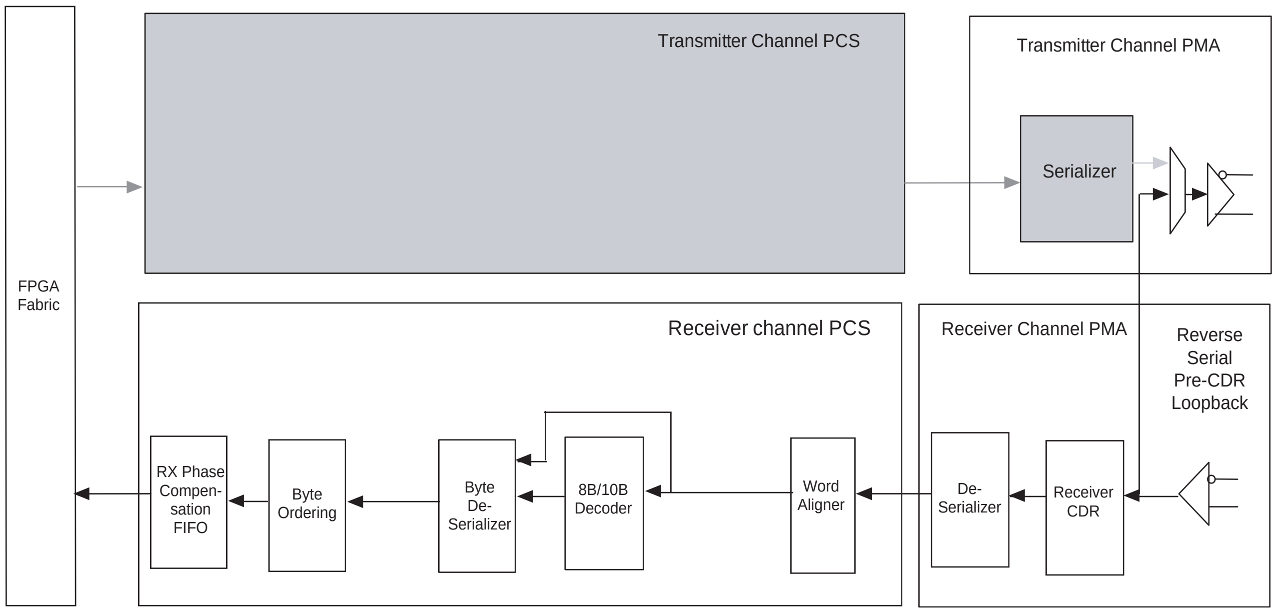
Figure 1–109. Reverse Serial Loopback Datapath (Grayed-Out Blocks are Not Active in this Mode)



Reverse Serial Pre-CDR Loopback

The reverse serial pre-CDR loopback is available as a subprotocol under Basic functional mode. In reverse serial pre-CDR loopback, the data received through the `rx_datain` port is looped back to the `tx_dataout` port **BEFORE** the receiver CDR. The received data is also available to the FPGA logic. Figure 1–110 shows the transceiver channel datapath for reverse serial pre-CDR loopback mode. Note that the active block of the transmitter channel is only the transmitter buffer. You can change the output differential voltage on the transmitter buffer through the ALTGX MegaWizard Plug-In Manager. You cannot change the pre-emphasis settings for the transmitter buffer.

Figure 1–110. Reverse Serial Pre-CDR Loopback Datapath



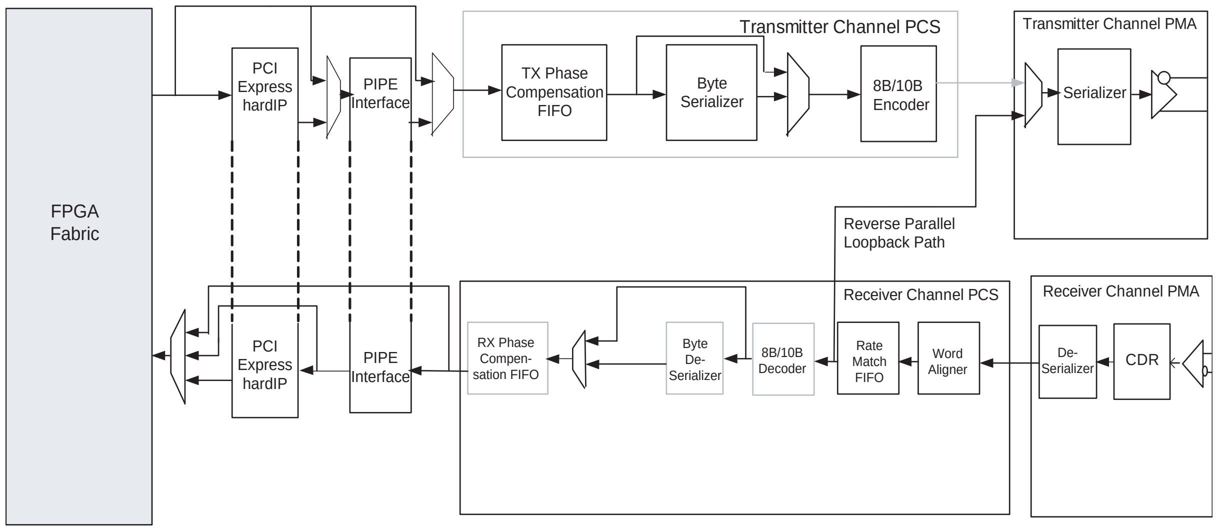
PCI Express (PIPE) Reverse Parallel Loopback

PCI Express (PIPE) reverse parallel loopback is only available in PCI Express (PIPE) functional mode for Gen1 data rate. As shown in Figure 1–111, the received serial data passes through the receiver CDR, deserializer, word aligner, and rate matching FIFO buffer. It is then looped back to the transmitter serializer and transmitted out through the `tx_dataout` port. The received data is also available to the FPGA fabric through the `rx_dataout` port. This loopback mode is compliant with the PCI Express (PIPE) specification 2.0. To enable this loopback mode, assert the `tx_detectrxloopback` port.

☞ This is the only loopback option supported in PCI Express (PIPE) functional mode.

In Figure 1–111, the grayed areas show the inactive paths when the PCI Express (PIPE) reverse parallel loop back mode is enabled.

Figure 1–111. PCI Express (PIPE) Reverse Parallel Loopback Mode Datapath



Calibration Blocks

Arria II GX devices contain calibration circuits that calibrate the OCT resistors and the analog portions of the transceiver blocks to ensure that the functionality is independent of process, voltage, or temperature variations.

Calibration Block Location

Figure 1–112 shows the location and number of calibration blocks available for different transceiver block device families.

In Figure 1–112 through Figure 1–114, the calibration block L0 refers to the calibration blocks on the left side.

Figure 1–112. Calibration Blocks

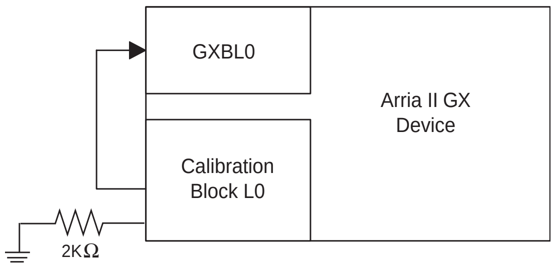


Figure 1–113 shows Arria II GX device families that have three transceiver blocks on the left side.

Figure 1–113. Calibration Three-Transceiver Blocks

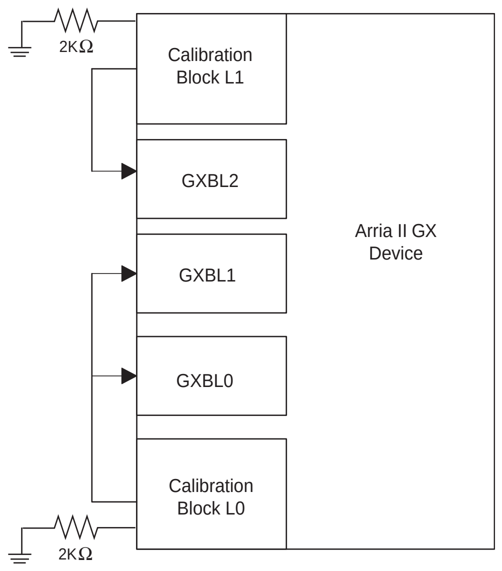
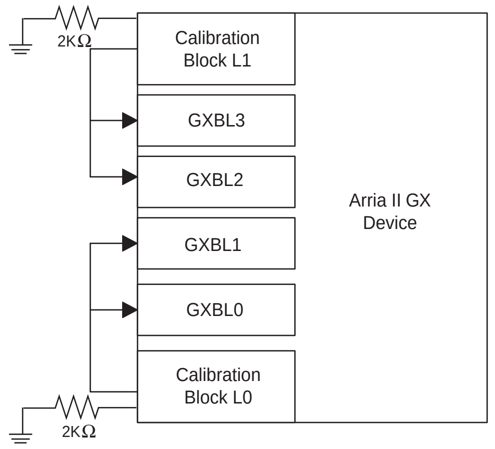


Figure 1–114 shows Arria II GX device families that have four transceiver blocks on the left side.

Figure 1–114. Calibration Four-Transceiver Blocks



The Quartus II software automatically selects the appropriate calibration block based on the assignment of the transceiver `tx_dataout` and `rx_datain` pins.

Calibration

The calibration block internally generates a constant internal reference voltage, independent of process, voltage, or temperature variations. It uses the internal reference voltage and external reference resistor (you must connect the resistor to the RREF pin) to generate constant reference currents. These reference currents are used by the analog block calibration circuit to calibrate the transceiver blocks.

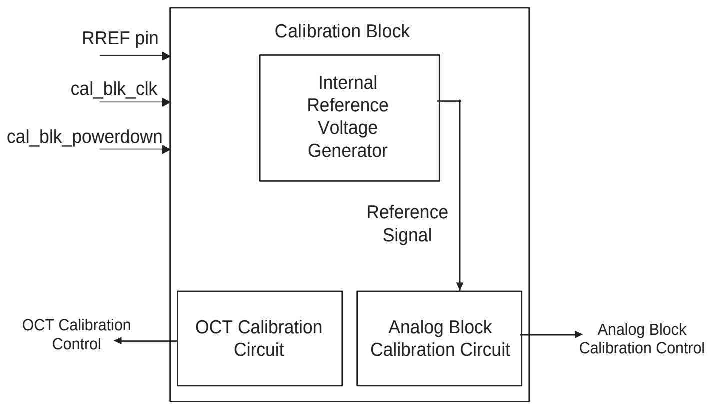
The OCT calibration circuit calibrates the OCT resistors present in the transceiver channels. You can enable the OCT resistors in the transceiver channels through the ALTGX MegaWizard Plug-In Manager.

You must connect a separate 2 kΩ (tolerance max ± 1%) external resistor on each RREF pin in the Arria II GX device to ground. To ensure proper operation of the calibration block, the RREF resistor connection in the board must be free from any external noise.

Input Signals to the Calibration Block

Figure 1–115 shows the required inputs to the calibration block. The ALTGX MegaWizard Plug-In Manager provides the `cal_blk_clk` and `cal_blk_powerdown` ports to control the calibration block.

Figure 1–115. Input Signals to the Calibration Blocks *(Note 1)*



Note to Figure 1–115:

(1) The RREF pin must be connected through a 2K resistor to ground. The resistor value is preliminary. Final values will be available upon characterization.

- `cal_blk_clk`—you must use the `cal_blk_clk` port to provide input clock to the calibration clock. The frequency of `cal_blk_clk` must be in the range of 10 MHz to 125 MHz (this range is preliminary. Final values will be available upon characterization). You can use dedicated clock routes such as the global or regional clock. If you do not have a suitable input reference clock or dedicated clock routing resources available, use divide-down logic from the FPGA fabric to generate a slow clock and use local clocking routing. Drive the `cal_blk_clk` port of all ALTGX instances that are associated with the same calibration block from the same input pin or logic.

- `cal_blk_powerdown`—you can perform calibration multiple times with the `cal_blk_powerdown` port available through the ALTGX MegaWizard Plug-In Manager. Assert this signal for approximately 500 ns (this is preliminary, final values will be available upon characterization). Following de-assertion of `cal_blk_powerdown`, the calibration block restarts the calibration process. Drive the `cal_blk_powerdown` port of all ALTGX instances that are associated with the same calibration block from the same input pin or logic.

Document Revision History

Table 1–43 shows the revision history for this chapter.

Table 1–43. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial release.	—

2. Arria II GX Transceiver Clocking

AIIGX52002-1.0

Introduction

This chapter provides detailed information about Arria® II GX transceiver clocking architecture; namely, input reference clocking, transceiver channel datapath clocking, FPGA fabric-transceiver interface clocking, and FPGA fabric PLLs-transceiver PLLs cascading.

This chapter includes the following sections:

- "CMU PLL and Receiver CDR Input Reference Clocking" on page 2–1
- "Transceiver Channel Datapath Clocking" on page 2–6
- "FPGA Fabric-Transceiver Interface Clocking" on page 2–27
- "FPGA Fabric PLLs-Transceiver PLLs Cascading" on page 2–56

CMU PLL and Receiver CDR Input Reference Clocking

Each transceiver block has:

- Two clock multiplier unit (CMU) phase-locked loops (PLLs) (`CMU0 PLL` and `CMU1 PLL`), one in each clock multiplier unit channel (CMU channel)
- Four clock data recovery (CDR) units, one in each receiver channel

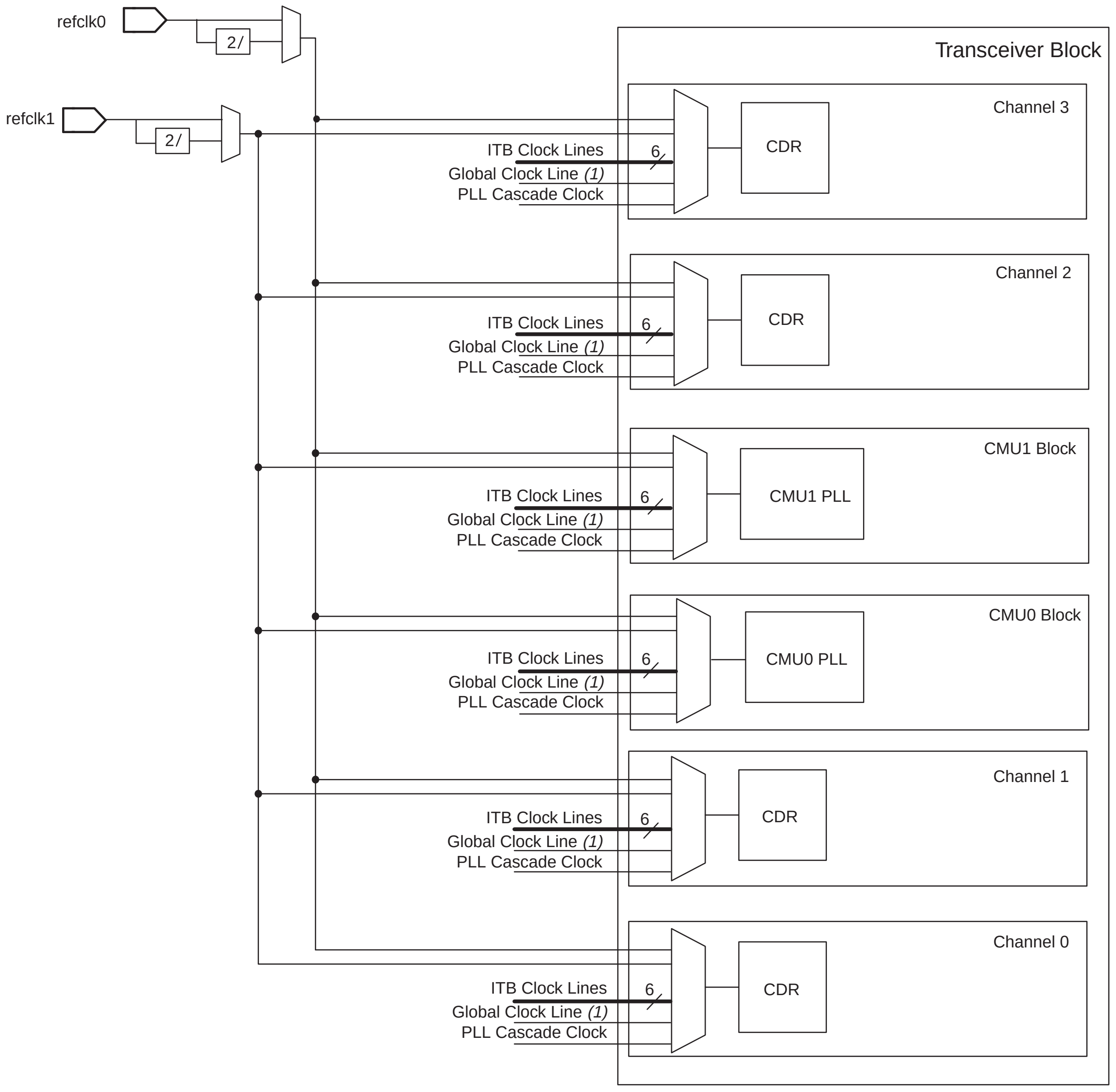
The CMU PLLs and receiver CDRs require an input reference clock for their operation. The CMU PLL synthesizes the input reference clock to generate the high-speed serial clock used in the transmitter PMA. The receiver CDR uses the input reference clock as a training clock when it is in lock-to-reference (LTR) mode.

The CMU PLLs and receiver CDRs in each transceiver block can derive input reference from one of the following sources:

- `refclk0` and `refclk1` pins of the same transceiver block
- `refclk0` and `refclk1` pins of other transceiver blocks on the same side of the device using the inter-transceiver block (ITB) clock network
- Dedicated CLK input pins on the FPGA global clock network
- Clock output pins from the left PLLs in the FPGA fabric

Figure 2–1 shows the input reference clock sources for CMU PLLs and receiver CDRs within a transceiver block.

Figure 2–1. Input Reference Clock Sources in a Transceiver Block

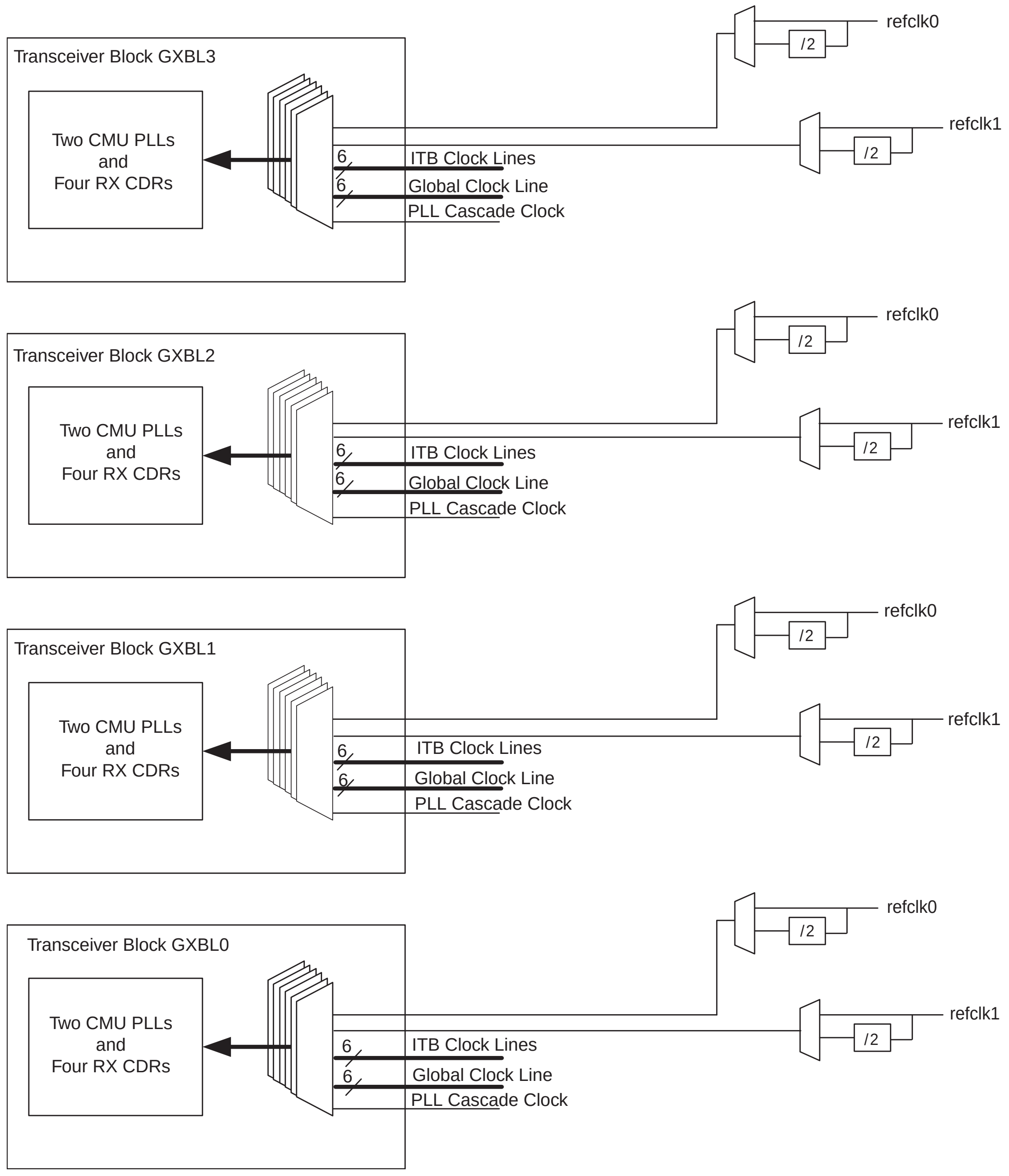


Note to Figure 2–1:

(1) One global clock line is available for each CMU PLL and receiver CDR in a transceiver block. This configuration allows each CMU PLL and receiver CDR to derive its input reference clock from a separate FPGA CLK input pin.

Figure 2–2 shows the input reference clock sources for CMU PLLs and receiver CDRs in four transceiver blocks on the left side of the EP2AGX260FF35 device.

Figure 2–2. Input Reference Clock Sources across Transceiver Blocks



refclk0 and refclk1 Pins

Each transceiver block has two dedicated `refclk` pins that you can use to drive the CMU PLL, receiver CDR, input reference clocks, or all three. Each of the two CMU PLLs and four receiver CDRs within a transceiver block can derive its input reference clock from either the `refclk0` or `refclk1` pin.

The `refclk` pins provide the cleanest input reference clock path to the CMU PLLs. Altera recommends using the `refclk` pins to drive the CMU PLL input reference clock for improved transmitter output jitter performance.

Table 2–17 shows the electrical specifications for the input reference clock signal driven on the `refclk` pins.

Table 2–1. Electrical Specifications for the Input Reference Clock

Protocol	I/O Standard	Coupling	Termination
■ GIGE ■ XAUI ■ Serial RapidIO ■ SONET/SDH ■ SDI ■ Basic	■ 1.2-V PCML ■ 1.5-V PCML ■ 2.5-V PCML ■ Differential LVPECL ■ LVDS	AC	On-chip
PCI Express (PIPE) *(1)*, *(2)*	■ 1.2-V PCML ■ 1.5-V PCML ■ 2.5-V PCML ■ DIfferential LVPECL ■ LVDS	AC	On-chip
	HCSL	DC	Off-chip

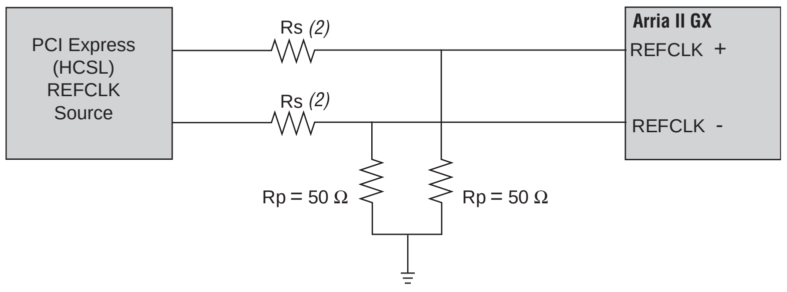
Notes to Table 2–17:

(1) In PCI Express (PIPE) mode, you have the option of selecting the HCSL standard for the reference clock if compliance to the PCI Express (PIPE) protocol is required. The Quartus® II software automatically selects DC coupling with external termination for the `refclk` pins signal if configured as HCSL.

(2) Refer to Figure 2–3 for an example termination scheme.

Figure 2–3 shows an example termination scheme for a reference clock signal when configured as HCSL.

Figure 2–3. Termination Scheme for a Reference Clock Signal When Configured as HCSL *(Note 1)*, *(2)*



Notes to Figure 2–3:

(1) No biasing is required if the reference clock signals are generated from a clock source that conforms to the PCI Express (PIPE) specification.

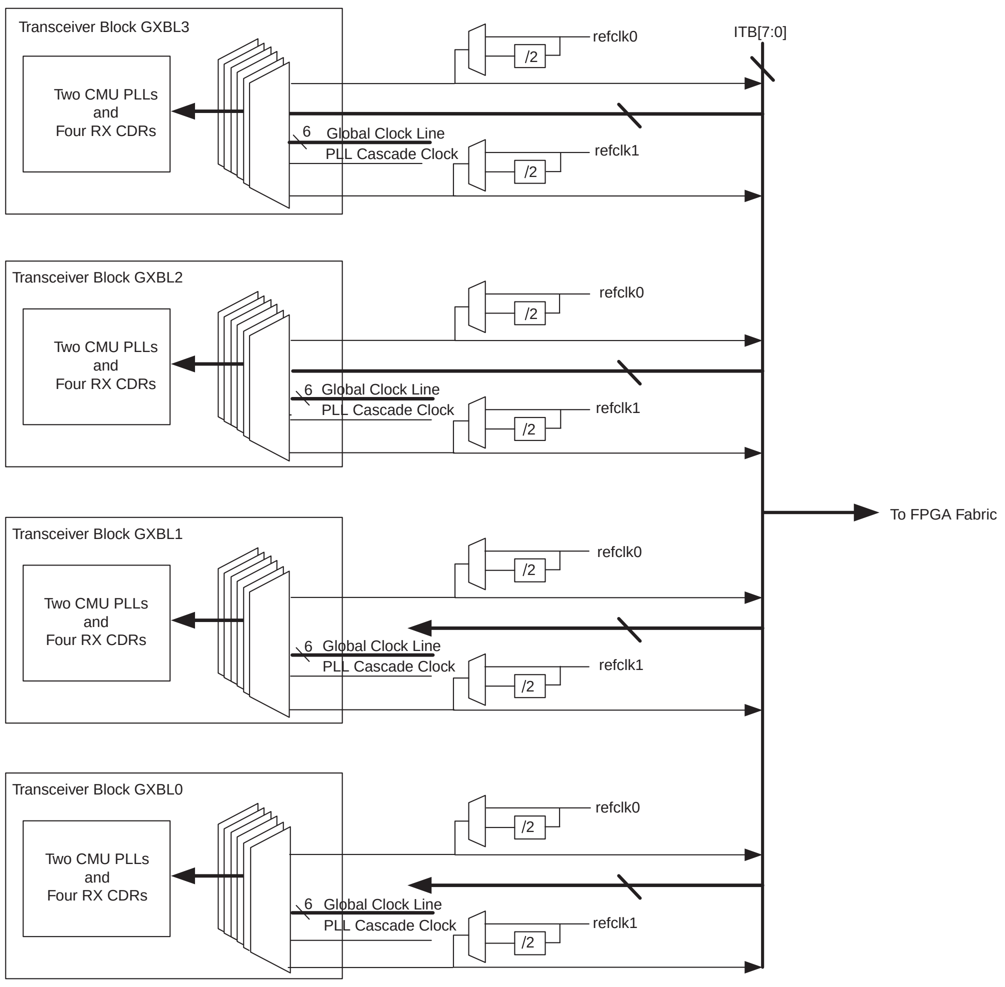
(2) Select resistor values as recommended by the PCI Express (PIPE) clock source vendor.

Inter-Transceiver Block (ITB) Clock Lines

The ITB clock lines provide an input reference clock path from the `refclk` pins of one transceiver block to the CMU PLLs and receiver CDRs of other transceiver blocks. In designs that have channels located in different transceiver blocks, the ITB clock lines eliminate the need to connect the on-board reference clock crystal oscillator to the `refclk` pin of each transceiver block. The ITB clock lines also drive the clock signal on the `refclk` pins to clock logic in the FPGA fabric.

Each `refclk` pin drives one ITB clock line for a total of up to eight ITB clock lines on the left side of the device, as shown in Figure 2–4.

Figure 2–4. Inter-Transceiver Block Clock Lines *(Note 1)*



Note to Figure 2–4:

(1) This figure shows the ITB clock lines on the left side of the EP2AGX60FF35 device. The number of ITB clock lines available in any Arria II GX device is equal to the number of `refclk` pins available in that device.

Dedicated CLK Input Pins on the FPGA Global Clock Network

Arria II GX devices provide six differential `CLK[5:0]` input pins located in non-transceiver I/O banks that you can use to provide the input reference clock to the transceiver blocks. The Quartus II software automatically chooses the global clock (GCLK) network to route the input reference clock signal from the CLK pins to the transceiver blocks.

For more information, refer to the "Dedicated Clock Input Pins" section in the *Clock Networks and PLLs in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

One global clock resource is available for each CMU PLL and receiver CDR within a transceiver block. This configuration allows each CMU PLL and receiver CDR to derive its input reference clock from a separate FPGA CLK input pin.

Clock Output from Left PLLs in the FPGA Fabric

Use the synthesized clock output from one of the left PLLs in the FPGA fabric to provide the input reference clock to the CMU PLLs and receiver CDRs. These devices provide a dedicated clock path from the left PLLs (PLL_1 and PLL_4) in the FPGA fabric to the PLL cascade network in the transceiver blocks located on the left side of the device. The additional clock multiplication factors available in the left PLLs allow more options for on-board crystal oscillator frequencies. For more information, refer to "FPGA Fabric PLLs-Transceiver PLLs Cascading" on page 2–56.

Transceiver Channel Datapath Clocking

The following sections describe transmitter and receiver channel datapath clocking in various configurations. Datapath clocking varies with physical coding sublayer (PCS) configurations in different functional modes as well as channel bonding options.

Transmitter Channel Datapath Clocking

This section describes transmitter channel PMA and PCS datapath clocking in non-bonded and bonded channel configurations. Transmitter datapath clocking in bonded channel configurations is set up to provide low channel-to-channel skew when compared to non-bonded channel configurations.

The following factors contribute to transmitter channel-to-channel skew:

- High-speed serial clock and low-speed parallel clock skew between channels
- Unequal latency in the transmitter phase compensation FIFO

In non-bonded channel configurations, the high-speed serial clock and low-speed parallel clock in each channel are generated independently by its local clock divider, as shown in Figure 2–5 on page 2–8. This results in higher channel-to-channel clock skew. The transmitter phase compensation FIFO in each non-bonded channel has its own pointers and control logic that can result in unequal latency in the transmitter phase compensation FIFO of each channel. The higher transceiver clock skew and unequal latency in the transmitter phase compensation FIFO in each channel can result in higher channel-to-channel skew in non-bonded channel configurations.

In bonded channel configurations, the high-speed serial clock and low-speed parallel clock for all bonded channels are generated by the same `CMU0` clock divider block (see Figure 2–6 on page 2–11), resulting in lower channel-to-channel clock skew. The transmitter phase compensation FIFO in all bonded channels share common pointers and control logic generated in the `CMU0` channel, resulting in equal latency in the transmitter phase compensation FIFO of all bonded channels. The lower transceiver clock skew and equal latency in the transmitter phase compensation FIFOs in all channels provides lower channel-to-channel skew in bonded channel configurations.

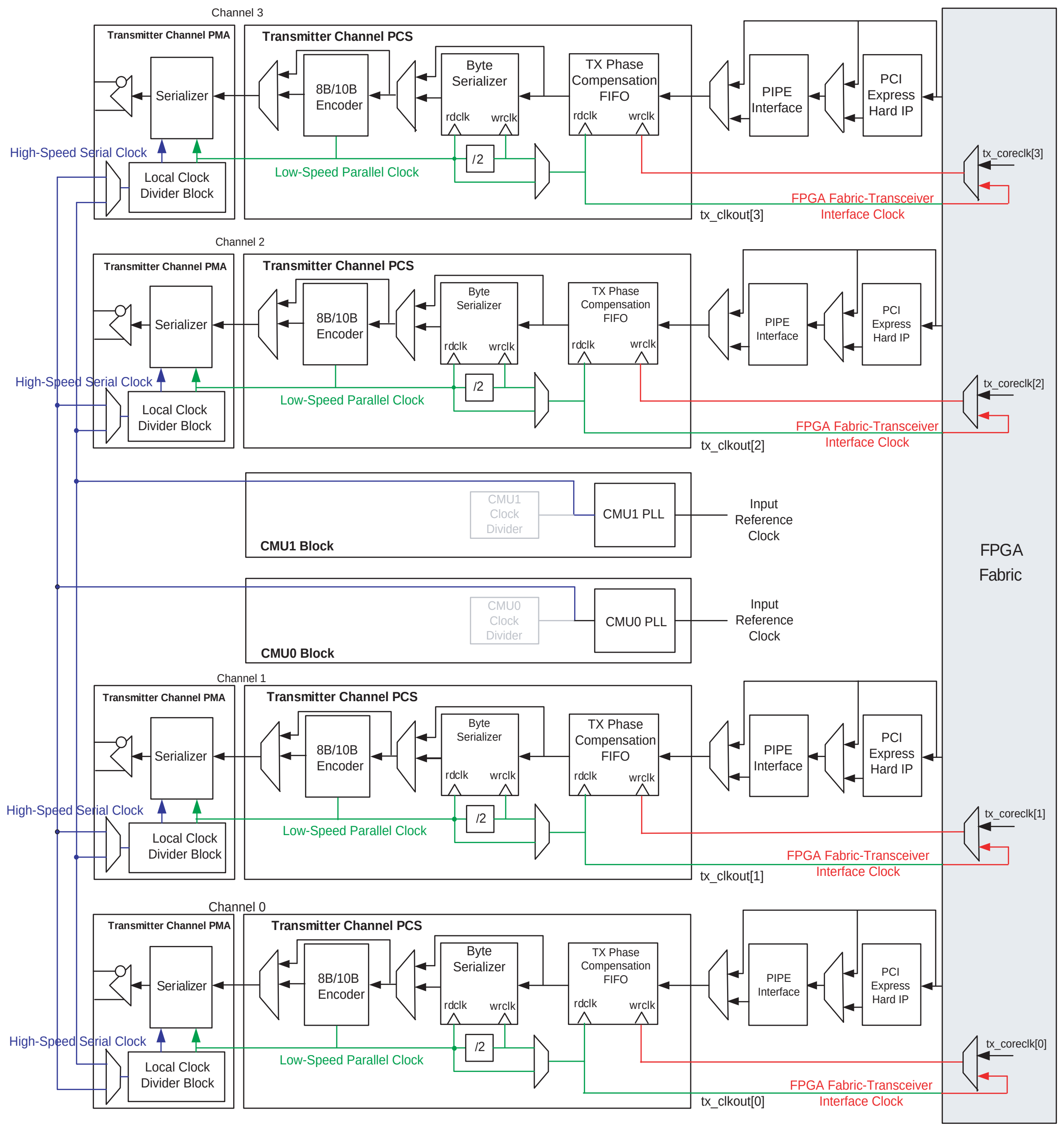
Non-Bonded Channel Configurations

The following functional modes support non-bonded transmitter channel configuration:

- PCI Express (PIPE) ×1
- Gigabit Ethernet (GIGE)
- Serial RapidIO (SRIO)
- SONET/SDH
- SDI
- Basic (except Basic ×4 mode)

Figure 2–5 shows the transmitter channel datapath clocking in a non-bonded configuration.

Figure 2–5. Transmitter Datapath Clocking in a Non-Bonded Configuration



In non-bonded channel configurations, each channel can derive its clock independently from either `CMU0 PLL` or `CMU1 PLL` within the same transceiver block. The CMU PLL synthesizes the input reference clock to generate a clock that runs at a frequency of half the configured data rate. This half rate clock from the CMU PLL is fed to the local clock divider block in each channel. Depending on the configured functional mode, the local clock divider block in each channel generates the low-speed parallel clock and high-speed serial clock. The serializer in the transmitter channel PMA uses both the low-speed parallel clock and high-speed serial clock for its parallel-in, serial-out operation. The low-speed parallel clock clocks both the 8B/10B encoder (if enabled) and the read port of the byte serializer (if enabled) in the transmitter channel PCS.

If the configured functional mode does not use the byte serializer, the low-speed parallel clock provides clock to the read port of the transmitter phase compensation FIFO. The low-speed parallel clock is also driven directly on the `tx_clkout` port as the FPGA fabric-transceiver interface clock. You can use the `tx_clkout` port to clock transmitter data and control logic in the FPGA fabric.

If the configured functional mode uses a byte serializer to reduce the FPGA fabric-transceiver interface speed, the low-speed parallel clock is divided by two. This divide-by-two version of the low-speed parallel clock provides clock to the write port of the byte serializer and the read port of the transmitter phase compensation FIFO. It is also driven on the `tx_clkout` port as the FPGA fabric-transceiver interface clock. You can use `tx_clkout` to regulate transmitter data and control logic in the FPGA fabric.

Table 2–2 shows the transmitter channel datapath clock frequencies in non-bonded functional modes that have a fixed data rate.

Table 2–2. Transmitter Channel Datapath Clock Frequencies in Non-Bonded Functional Modes

Functional Mode	Data Rate	High-Speed Serial Clock Frequency	Low-Speed Parallel Clock Frequency	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer	With Byte Serializer
PCI Express (PIPE) ×1 (Gen 1)	2.5 Gbps	1.25 GHz	250 MHz	N/A *(1)*	125 MHz
GIGE	1.25 Gbps	625 MHz	125 MHz	125 MHz	N/A
Serial RapidIO	1.25 Gbps	625 MHz	125 MHz	N/A	62.5 MHz
	2.5 Gbps	1.25 GHz	250 MHz	N/A	125 MHz
	3.125 Gbps	1.5625 GHz	312.5 MHz	N/A	156.25 MHz
SONET/SDH OC12	622 Mbps	311 MHz	77.75 MHz	77.75 MHz	N/A
SONET/SDH OC48	2.488 Gbps	1.244 GHz	311 MHz	N/A	155.5 MHz
HD-SDI	1.485 Gbps	742.5 MHz	148.5 MHz	148.5 MHz	74.25 MHz
	1.4835 Gbps	741.75 MHz	148.35 MHz	148.35 MHz	74.175 MHz
3G-SDI	2.97 Gbps	1.485 GHz	297 MHz	N/A	148.5 MHz
	2.967 Gbps	1.4835 GHz	296.7 MHz	N/A	148.35 MHz

Note to Table 2–2:

(1) 250 MHz when PCI Express hard IP is enabled.

Bonded Channel Configurations

Arria II GX devices support bonded channel configurations.

Arria II GX devices support ×4 PCS and PMA channel bonding that allows bonding of four channels within the same transceiver block. These devices also support ×8 channel bonding in PCI Express (PIPE) mode that allows bonding of eight PCS and PMA channels across two transceiver blocks on the same side of the device. Arria II GX devices with at least two transceiver blocks support ×8 bonding.

×4 Bonded Channel Configuration

The following functional modes support ×4 bonded transmitter channel configuration:

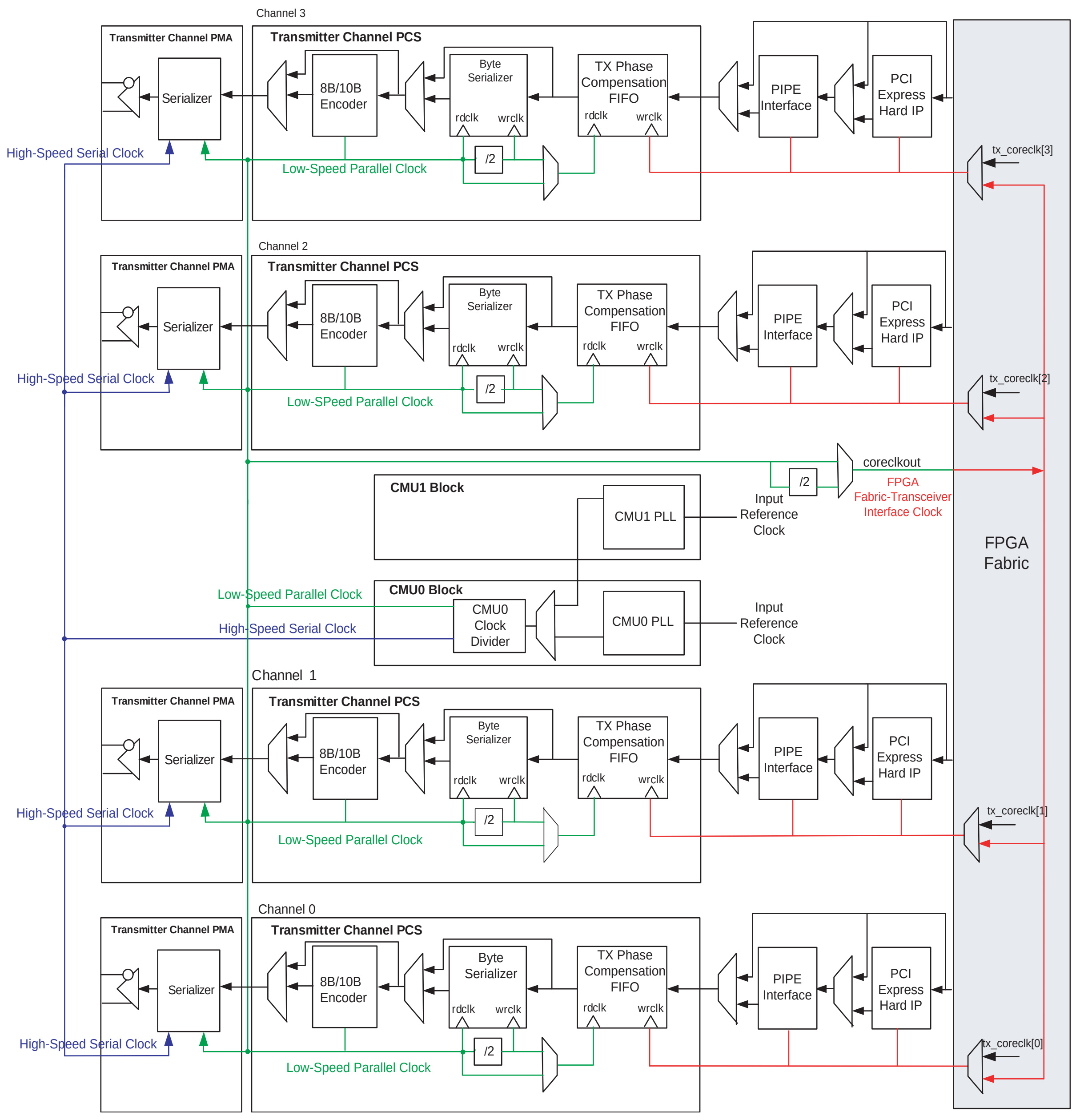
- PCI Express (PIPE) ×4
- XAUI
- Basic ×4

Figure 2–6 shows the transmitter channel datapath clocking in ×4 channel bonding configurations.

☞ The Quartus II compilation generates an error if you do not assign:

- `tx_dataout[0]` of the ×4 bonded link (XAUI or PCI Express [PIPE] ×4) to physical channel 0 of the transceiver block
- `tx_dataout[1]` to physical channel 1 of the transceiver block
- `tx_dataout[2]` to physical channel 2 of the transceiver block
- `tx_dataout[3]` to physical channel 3 of the transceiver block

Figure 2–6. Transmitter Datapath Clocking in ×4 Bonded Configurations



In ×4 bonded channel configurations, `CMU0 PLL` or `CMU1 PLL` synthesizes the input reference clock to generate a clock that runs at a frequency of half the configured data rate. The half-rate clock from either of the CMU PLLs is fed to the `CMU0` clock divider in the `CMU0 Channel`. Depending on the configured functional mode, the `CMU0` clock divider block generates the high-speed serial clock and the low-speed parallel clock. The serializer in the transmitter channel PMA of the four bonded channels uses the same low-speed parallel clock and high-speed serial clock from the `CMU0` block for their parallel-in, serial-out operation. The low-speed parallel clock provides clock to the 8B/10B encoder and the read port of the byte serializer (if enabled) in the transmitter channel PCS.

If the configured functional mode does not use the byte serializer, the low-speed parallel clock from the `CMU0` clock divider block clocks the read port of the transmitter phase compensation FIFO in all four bonded channels. This low-speed parallel clock is also driven directly on the `coreclkout` port as the FPGA fabric-transceiver interface clock. You can use the `coreclkout` signal to clock transmitter data and control logic in the FPGA fabric for all four bonded channels.

If the configured functional mode uses the byte serializer, the low-speed parallel clock from the `CMU0` clock divider is divided by two. This divide-by-two version of the low-speed parallel clock provides clock to the write port of the byte serializer and the read port of the transmitter phase compensation FIFO in all four bonded channels. It is also driven on the `coreclkout` port as the FPGA fabric-transceiver interface clock. You can use the `coreclkout` signal to clock transmitter data and control logic in the FPGA fabric for all four bonded channels.

In ×4 bonded channel configurations, the transmitter phase compensation FIFOs in all four bonded channels share common read and write pointers and enable signals generated in the `CMU0` block channel of the transceiver block. This ensures equal transmitter phase compensation FIFO latency across all four bonded channels, resulting in low transmitter channel-to-channel skew.

Table 2–3 shows the transmitter datapath clock frequencies in ×4 bonded functional modes that have a fixed data rate.

Table 2–3. Transmitter Datapath Clock Frequencies in ×4 Bonded Functional Modes

Functional Mode	Data Rate	High-Speed Serial Clock Frequency	Low-Speed Parallel Clock Frequency	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer	With Byte Serializer
PCI Express (PIPE) ×4 (Gen 1)	2.5 Gbps	1.25 GHz	250 MHz	N/A *(1)*	125 MHz
XAUI	3.125 Gbps	1.5625 GHz	312.5 MHz	N/A	156.25 MHz

Note to Table 2–3:

(1) 250 MHz when PCI Express hard IP is enabled.

×8 Bonded Channel Configuration

The PCI Express (PIPE) ×8 functional mode supports ×8 bonded channel configuration in Arria II GX devices with at least two transceiver blocks. The eight bonded channels are located in two transceiver blocks, referred to as the master transceiver block and slave transceiver block, with four channels each. The `CMU0` clock divider in the `CMU0` block of the master transceiver block provides the serial PMA clock and parallel PCS clock to all eight bonded channels. The serializer in the transmitter channel PMA of the eight bonded channels uses the same low-speed parallel clock and high-speed serial clock from the `CMU0 Channel` of the master transceiver block for their parallel-in, serial-out operation. The low-speed parallel clock from the `CMU0 Channel` of the master transceiver block clocks the 8B/10B encoder and read port of the byte serializer (if enabled) in the transmitter channel PCS of all eight channels.

For an 8-bit FPGA fabric-transceiver channel interface that does not use the byte serializer, the low-speed parallel clock from the `CMU0` clock divider block in the master transceiver block clocks the read port of the transmitter phase compensation FIFO in all eight bonded channels. This low-speed parallel clock is also driven directly on the `coreclkout` port as the FPGA fabric-transceiver interface clock. You can use the `coreclkout` signal to clock the transmitter data and control logic in the FPGA fabric for all eight bonded channels.

For a 16-bit FPGA fabric-transceiver channel interface that uses the byte serializer, the low-speed parallel clock from the `CMU0` clock divider block in the master transceiver block is divided by two. This divide-by-two version of the low-speed parallel clock provides clock to the write port of the byte serializer and the read port of the transmitter phase compensation FIFO in all eight bonded channels. It is also driven on the `coreclkout` port as the FPGA fabric-transceiver interface clock. You can use the `coreclkout` signal to clock the transmitter data and control logic in the FPGA fabric for all eight bonded channels.

In the PCI Express (PIPE) ×8 bonded channel configuration, the transmitter phase compensation FIFOs in all eight bonded channels share common read and write pointers and enable signals generated in the `CMU0` block of the master transceiver block. This ensures equal transmitter phase compensation FIFO latency across all eight bonded channels, resulting in low transmitter channel-to-channel skew.

Figure 2–7 shows transmitter datapath clocking in PCI Express (PIPE) ×8 channel bonding configurations.

Figure 2–7. Transmitter Datapath Clocking in a ×8 Bonded Configuration

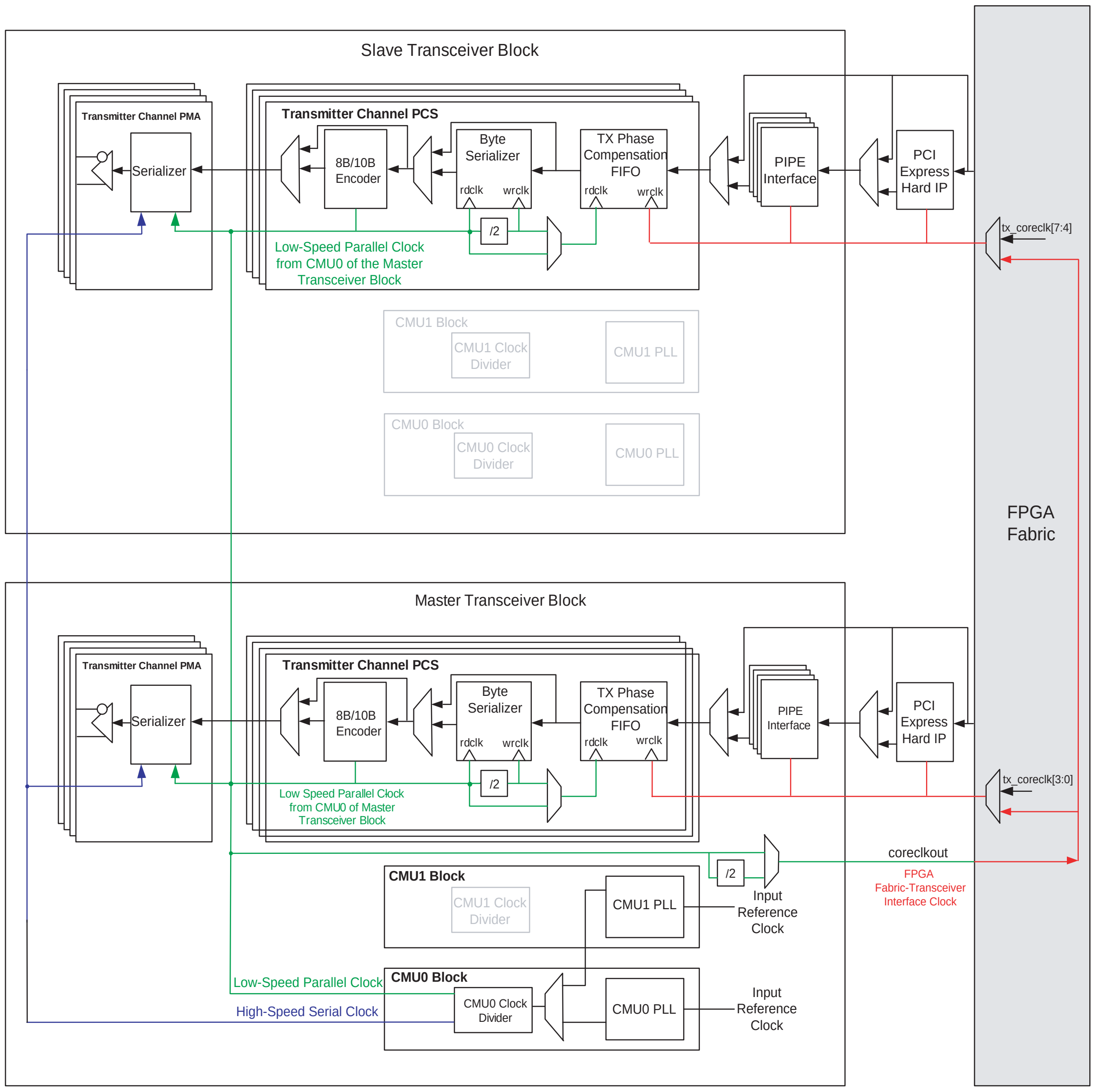


Figure 2–8 through Figure 2–10 show allowed master and slave transceiver block locations and PCI Express (PIPE) logical lane-to-physical transceiver channel mapping in all Arria II GX devices.

☞ The Quartus II compilation generates an error if you do not map the PCI Express (PIPE) logical lanes to the physical transceiver channels, as shown in Figure 2–8 through Figure 2–10.

Figure 2–8 shows the PCI Express (PIPE) ×8 link in two transceiver block devices.

Figure 2–8. One PCI Express (PIPE) ×8 Link in Two Transceiver Block Devices

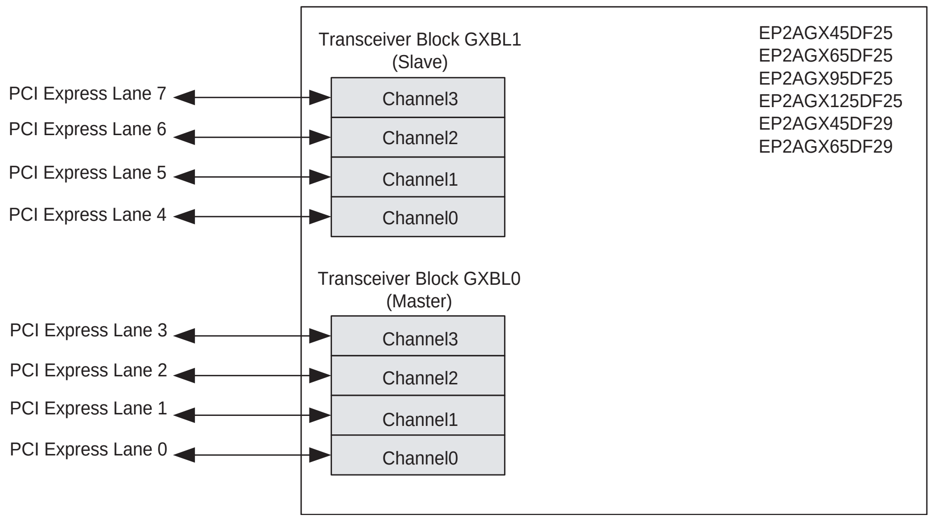
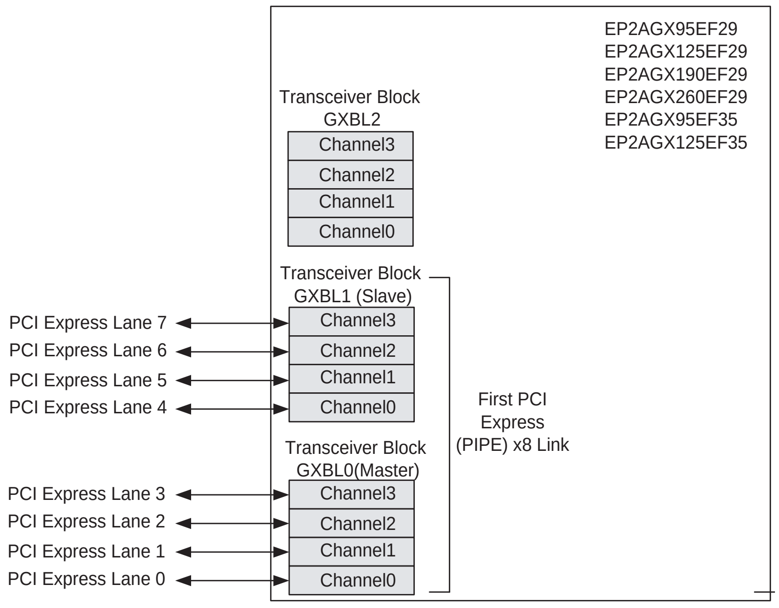


Figure 2–9 shows the PCI Express (PIPE) ×8 link in three transceiver block devices.

Figure 2–9. Two PCI Express (PIPE) ×8 Link in Three Transceiver Block Devices *(Note 1)*

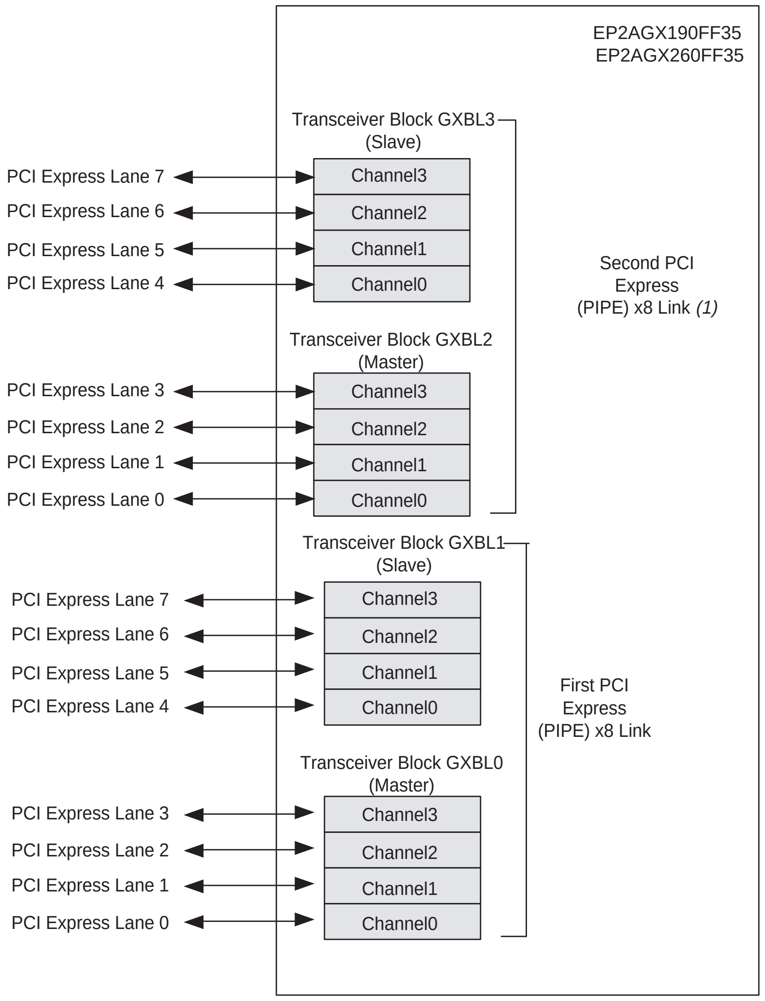


Note to Figure 2–9:

(1) Arria II GX devices with three transceiver blocks allow a maximum of one PCI Express (PIPE) ×8 link occupying two transceiver blocks. You can configure the other transceiver block to implement other functional modes.

Figure 2–10 shows the PCI Express (PIPE) ×8 link in four transceiver block devices.

Figure 2–10. Two PCI Express (PIPE) ×8 Link in Four Transceiver Block Devices



Note to Figure 2–10:

(1) The second ×8 link does not have PCI Express hard IP support. Use soft IP support for the second ×8 link.

Receiver Channel Datapath Clocking

This section describes receiver PMA and PCS datapath clocking in supported configurations. The receiver datapath clocking varies between non-bonded and bonded channel configurations. It also varies with the use of PCS blocks; for example, deskew FIFO and rate matcher.

Non-Bonded Channel Configurations

In non-bonded channel configurations, receiver PCS blocks of each channel are clocked independently. Each non-bonded channel also has separate `rx_analogreset` and `rx_digitalreset` signals that allow independent reset of the receiver PCS logic in each channel.

For more information about transceiver reset and power down signals, refer to the *Reset Control and Power Down* chapter in volume 2 of the *Arria II GX Device Handbook*.

In addition, in non-bonded channel configurations, receiver PCS clocking varies, depending on whether the configured functional mode uses the rate matcher block or not.

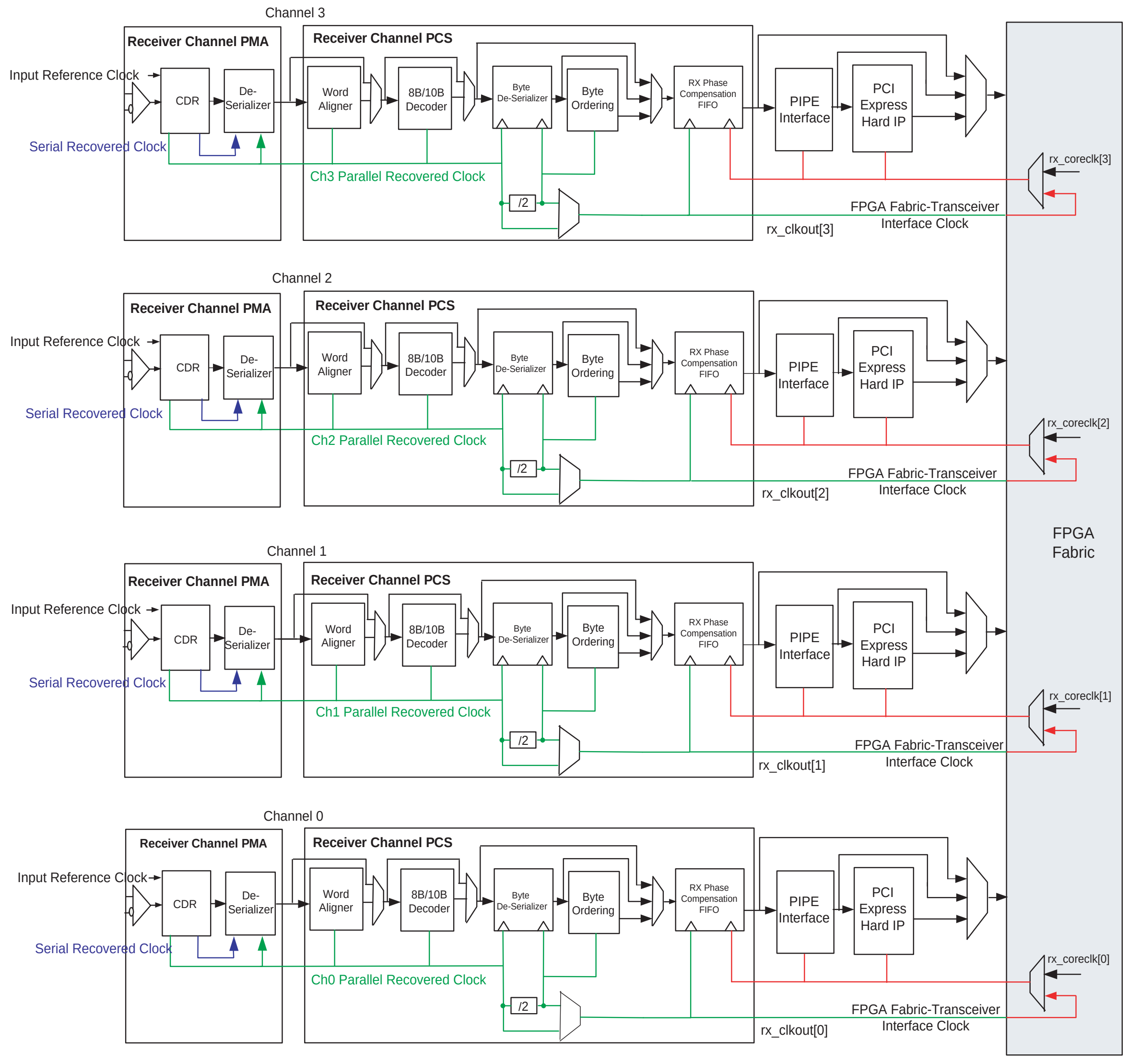
Non-Bonded Receiver Clocking without Rate Matcher

The following functional modes have non-bonded receiver channel configuration without rate-matcher:

- Serial RapidIO
- SONET/SDH
- SDI
- Basic without rate matcher

Figure 2–11 shows receiver datapath clocking in non-bonded channel configurations without rate matcher.

Figure 2–11. Receiver Datapath Clocking in Non-Bonded Configurations without Rate Matcher



In non-bonded configurations without rate matcher, the CDR in each receiver channel recovers the serial clock from the received data. Also, the serial recovered clock frequency is half the configured data rate due to the half-rate CDR architecture. The serial recovered clock is divided within the receiver PMA to generate the parallel recovered clock. The deserializer uses the serial recovered clock in the receiver PMA. The parallel recovered clock and deserialized data is forwarded to the receiver PCS. The parallel recovered clock in each channel clocks the word aligner and 8B/10B decoder (if enabled).

If the configured functional mode does not use the byte deserializer, the parallel recovered clock also clocks the write side of the receiver phase compensation FIFO. It is also driven on the `rx_clkout` port as the FPGA fabric-transceiver interface clock. You can use the `rx_clkout` signal to latch the receiver data and status signals in the FPGA fabric.

If the configured functional mode uses the byte deserializer, the parallel recovered clock is divided by two. This divide-by-two version of the parallel recovered clock clocks the read side of the byte deserializer, the byte ordering block (if enabled), and the write side of the receiver phase compensation FIFO. It is also driven on the `rx_clkout` port as the FPGA fabric-transceiver interface clock. You can use the `rx_clkout` signal to latch the receiver data and status signals in the FPGA fabric.

Table 2–4 shows receiver datapath clock frequencies in non-bonded functional modes without rate matcher.

Table 2–4. Receiver Datapath Clock Frequencies in Non-Bonded Functional Modes without Rate Matcher

Functional Mode	Data Rate	High-Speed Serial Clock Frequency	Low-Speed Parallel Clock Frequency	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer	With Byte Serializer
Serial RapidIO	1.25 Gbps	625 MHz	125 MHz	N/A	62.5 MHz
	2.5 Gbps	1.25 GHz	250 MHz	N/A	125 MHz
	3.125 Gbps	1.5625 GHz	312.5 MHz	N/A	156.25 MHz
SONET/SDH OC12	622 Mbps	311 MHz	77.75 MHz	77.75 MHz	N/A
SONET/SDH OC48	2.488 Gbps	1.244 GHz	311 MHz	N/A	155.5 MHz
HD SDI	1.485 Gbps	742.5 MHz	148.5 MHz	148.5 MHz	74.25 MHz
	1.4835 Gbps	741.75 MHz	148.35 MHz	148.35 MHz	74.175 MHz
3G-SDI	2.97 Gbps	1.485 GHz	297 MHz	N/A	148.5 MHz
	2.967 Gbps	1.4835 Ghz	296.7 MHz	N/A	148.35 MHz

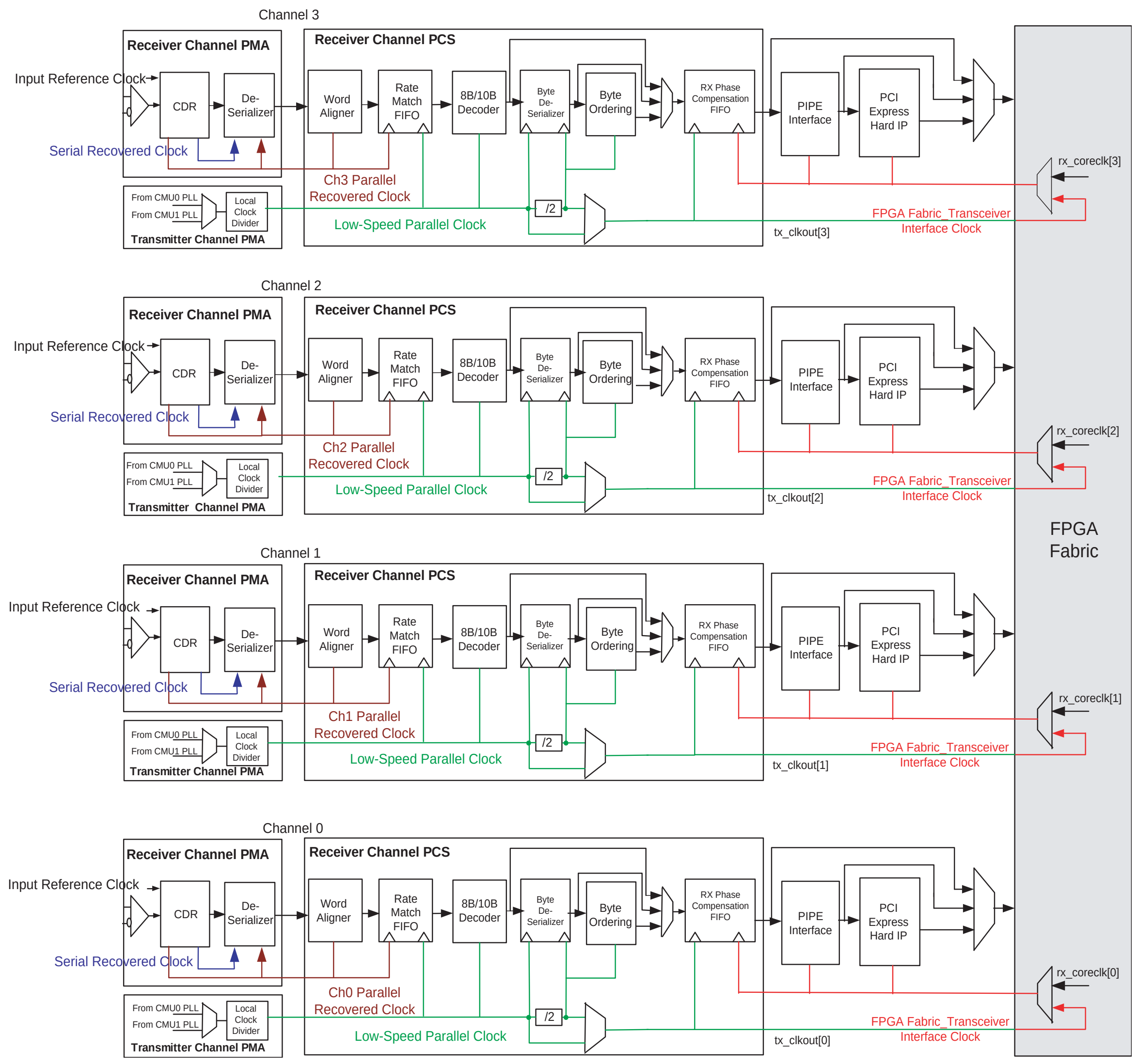
Non-Bonded Receiver Clocking with Rate Matcher

The following functional modes have non-bonded receiver channel configurations with rate-matcher:

- PCI Express (PIPE) ×1
- GIGE
- Serial RapidIO
- Basic with rate matcher

Figure 2–12 shows receiver datapath clocking in non-bonded channel configurations with rate matcher.

Figure 2–12. Receiver Datapath Clocking in Non-Bonded Configurations with Rate Matcher



In non-bonded configurations with rate matcher, the CDR in each receiver channel recovers the serial clock from the received data. Also, the serial recovered clock frequency is half the configured data rate due to the half rate CDR architecture. The serial recovered clock is divided within the receiver PMA to generate the parallel recovered clock. The deserializer uses the serial recovered clock in the receiver PMA. The parallel recovered clock and deserialized data is forwarded to the receiver PCS.

The parallel recovered clock from the receiver PMA in each channel clocks the word aligner and the write port of the rate match FIFO. The low-speed parallel clock from the transmitter local clock divider block in each channel clocks the read port of the rate match FIFO, 8B/10B decoder, and the write port of the byte deserializer (if enabled). The parallel transmitter PCS clock or its divide-by-two version (if byte deserializer is enabled) clocks the write port of the receiver phase compensation FIFO. It is also driven on the `tx_clkout` port as the FPGA fabric-transceiver interface clock. You can use the `tx_clkout` signal to latch the receiver data and status signals in the FPGA fabric.

Table 2–5 shows the receiver datapath clock frequencies in non-bonded functional modes with rate matcher.

Table 2–5. Receiver Datapath Clock Frequencies in Non-Bonded Functional Modes with Rate Matcher

Functional Mode	**Data Rate**	**Serial Recovered Clock Frequency**	**Parallel Recovered Clock Frequency**	**FPGA Fabric-Transceiver Interface Clock Frequency**	
				Without Byte Serializer	**With Byte Serializer**
PCI Express (PIPE) ×1 (Gen 1)	2.5 Gbps	1.25 GHz	250 MHz	N/A *(1)*	125 MHz
GIGE	1.25	625 MHz	125 MHz	125 MHz	N/A
Serial RapidIO	1.25 Gbps	625 MHz	125 MHz	N/A	62.5 MHz
	2.5 Gbps	1.25 GHz	250 MHz	N/A	125 MHz
	3.125 Gbps	1.5625 GHz	312.5 MHz	N/A	156.25 MHz

Note to Table 2–5:

(1) 250 MHz when PCI Express hard IP is enabled.

Bonded Channel Configurations

The Arria II GX device supports ×4 channel bonding that allows bonding of four channels within the same transceiver block. It also supports ×8 channel bonding that allows bonding of eight channels across two transceiver blocks in PCI Express (PIPE) mode.

×4 Bonded Channel Configuration

The following functional modes support ×4 receiver channel bonded configuration:

- PCI Express (PIPE) ×4
- XAUI

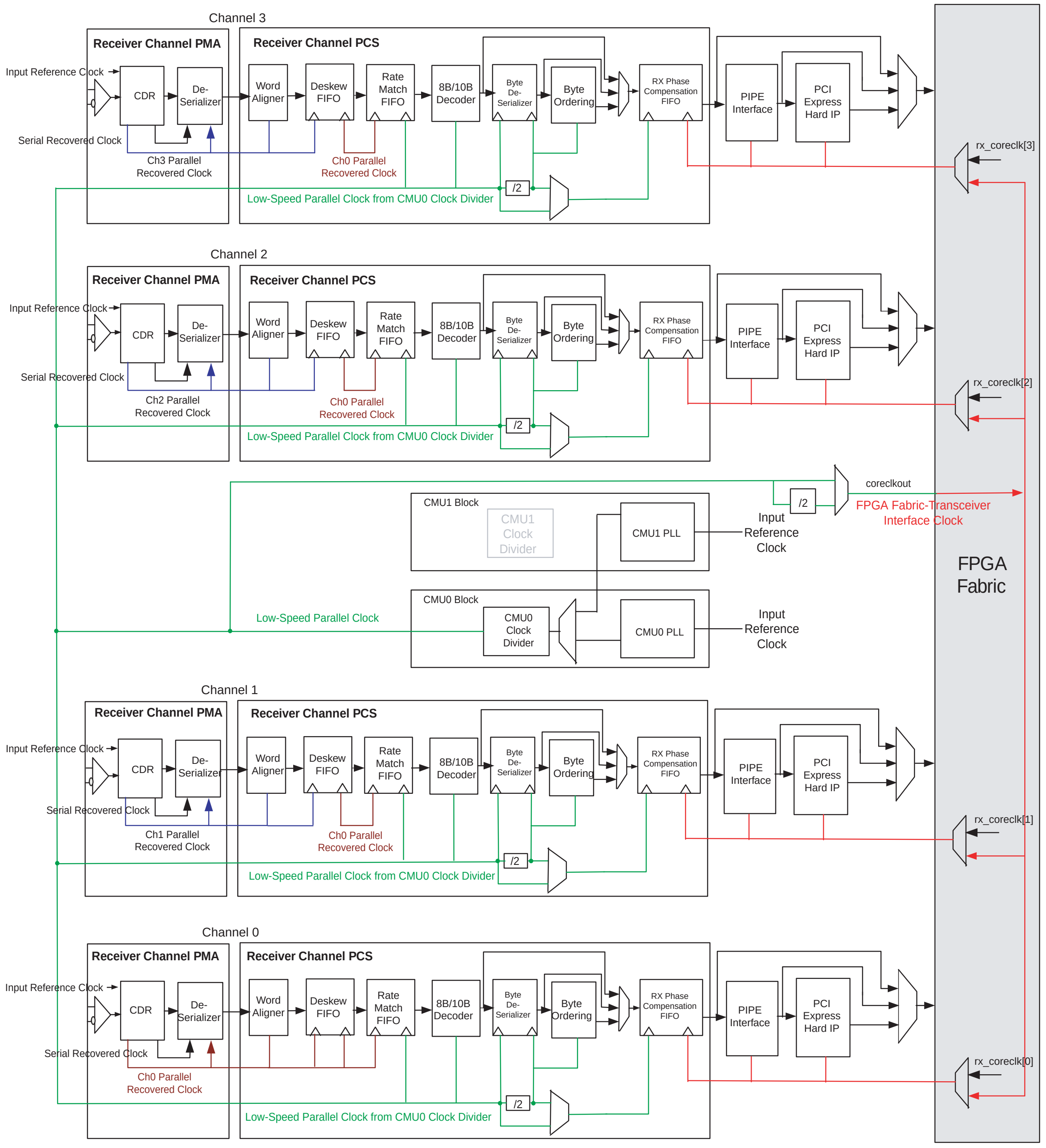
In ×4 bonded channel configurations, the receiver datapath clocking varies, depending on whether the configured functional mode uses the deskew FIFO or not.

×4 Bonded Channel Configuration with Deskew FIFO

XAUI functional mode has ×4 bonded channel configuration with deskew FIFO.

Figure 2–13 shows receiver datapath clocking in ×4 channel bonding configurations with deskew FIFO.

Figure 2–13. Receiver Datapath Clocking in ×4 Bonded Channel Configuration with Deskew FIFO



In ×4 bonded channel configurations with deskew FIFO, the CDR in each receiver channel recovers the serial clock from the received data. Also, the serial recovered clock frequency is half the configured data rate due to the half-rate CDR architecture. The serial recovered clock is divided within each channel's receiver PMA to generate the parallel recovered clock. The deserializer uses the serial recovered clock in the receiver PMA. The parallel recovered clock and deserialized data is forwarded to the receiver PCS in each channel.

The parallel recovered clock from the receiver PMA in each channel clocks the word aligner in that channel. The parallel recovered clock from `Channel 0` clocks the deskew FIFO and the write port of the rate match FIFO in all four bonded channels. The low-speed parallel clock from the `CMU0` clock divider block clocks the read port of the rate match FIFO, 8B/10B decoder, and the write port of the byte deserializer (if enabled) in all four bonded channels. The low-speed parallel clock or its divide-by-two version (if byte deserializer is enabled) clocks the write port of the receiver phase compensation FIFO. It is also driven on the `coreclkout` port as the FPGA fabric-transceiver interface clock. You can use the `coreclkout` signal to latch the receiver data and status signals in the FPGA fabric for all four bonded channels.

In ×4 bonded channel configurations, the receiver phase compensation FIFOs in all four bonded channels share common read and write pointers and enable signals generated in the `CMU0` block of the transceiver block.

Table 2–6 shows receiver datapath clock frequencies in ×4 bonded functional modes with deskew FIFO.

Table 2–6. Receiver Datapath Clock Frequencies in ×4 Bonded Functional Modes with Deskew FIFO

Functional Mode	Data Rate	Serial Recovered Clock Frequency	Parallel Recovered Clock and Parallel Transmitter PCS Clock Frequency	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer	With Byte Serializer
PCI Express (PIPE) ×4 (Gen 1)	2.5 Gbps	1.25 GHz	250 MHz	N/A (1)	125 MHz
XAUI	3.125 Gbps	1.5625 GHz	312.5 MHz	N/A	156.25 MHz

Note to Table 2–6:

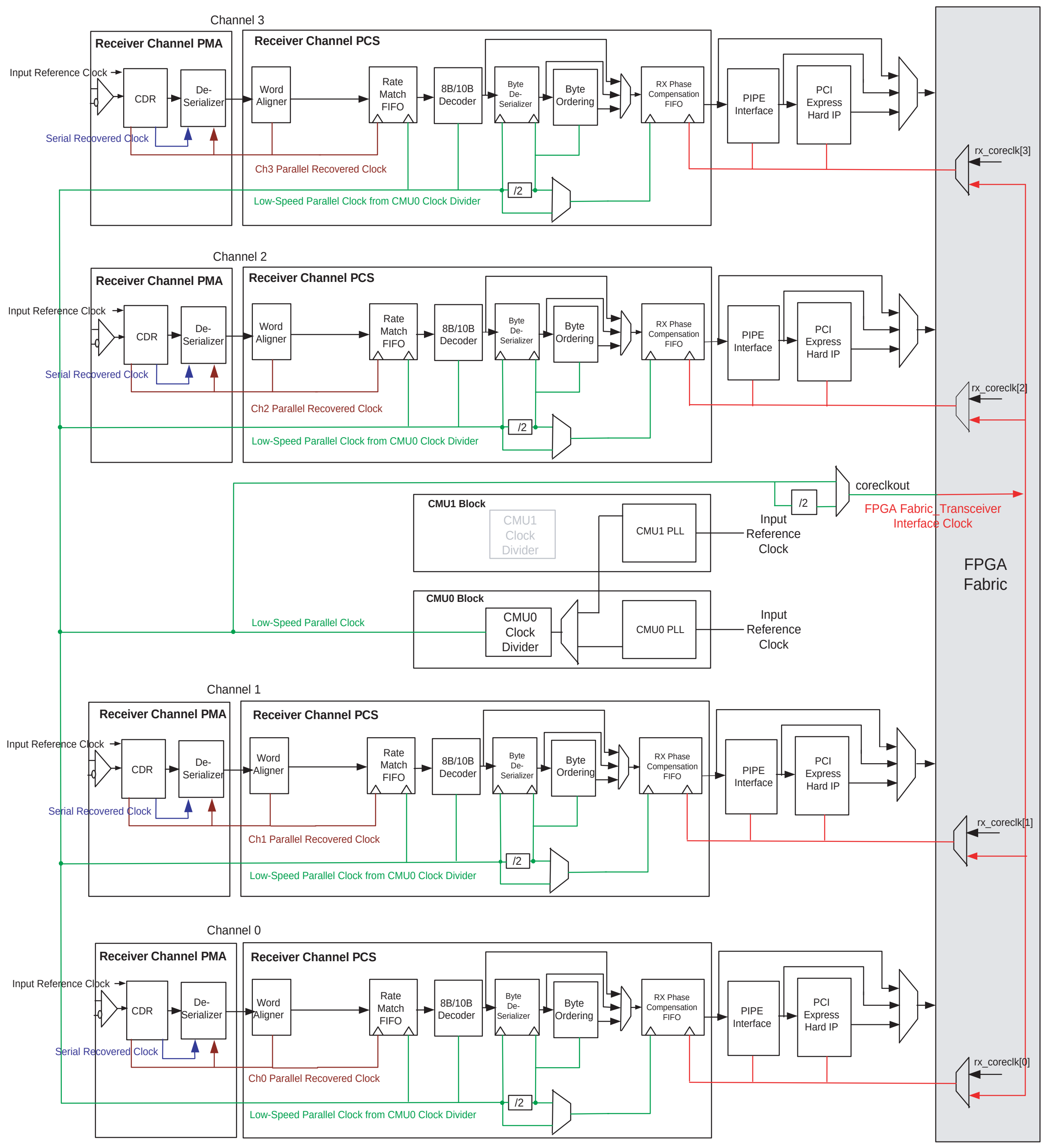
(1) 250 MHz when PCI Express hard IP is enabled.

x4 Bonded Channel Configurations without Deskew FIFO

PCI Express (PIPE) ×4 functional modes have ×4 bonded channel configurations without deskew FIFO.

Figure 2–14 shows receiver datapath clocking in ×4 channel bonding configurations without deskew FIFO.

Figure 2–14. Receiver Datapath Clocking in ×4 Bonded Channel Configurations without Deskew FIFO



In ×4 bonded channel configurations without deskew FIFO, the CDR in each receiver channel recovers the serial clock from the received data. The serial recovered clock frequency is half the configured data rate due to the half-rate CDR architecture. The serial recovered clock is divided within each channel's receiver PMA to generate the parallel recovered clock. The deserializer uses the serial recovered clock in the receiver PMA. The parallel recovered clock and deserialized data is forwarded to the receiver PCS in each channel.

The parallel recovered clock from the receiver PMA in each channel clocks the word aligner and write side of the rate match FIFO in that channel. The low-speed parallel clock from the `CMU0` clock divider block in `CMU0 Channel` clocks the read port of the rate match FIFO, 8B/10B decoder, and the write port of the byte deserializer (if enabled). The low-speed parallel clock or its divide-by-two version (if byte deserializer is enabled) clocks the receiver phase compensation FIFO. It is also driven on the `coreclkout` port as the FPGA fabric-transceiver interface clock. You can use the `coreclkout` signal to latch the receiver data and status signals in the FPGA fabric for all four bonded channels.

In ×4 bonded channel configurations, the receiver phase compensation FIFOs in all four bonded channels share common read and write pointers and enable signals generated in the `CMU0` channel of the transceiver block.

Table 2–7 shows receiver datapath clock frequencies in ×4 bonded functional modes without deskew FIFO.

Table 2–7. Receiver Datapath Clock Frequencies in ×4 Bonded Functional Modes without Deskew FIFO

Functional Mode	Data Rate	Serial Recovered Clock Frequency	Parallel Recovered Clock and Parallel Transmitter PCS Clock Frequency	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer	With Byte Serializer
PCI Express (PIPE) ×4 (Gen 1)	2.5 Gbps	1.25 GHz	250 MHz	N/A (1)	125 MHz

Note to Table 2–7:

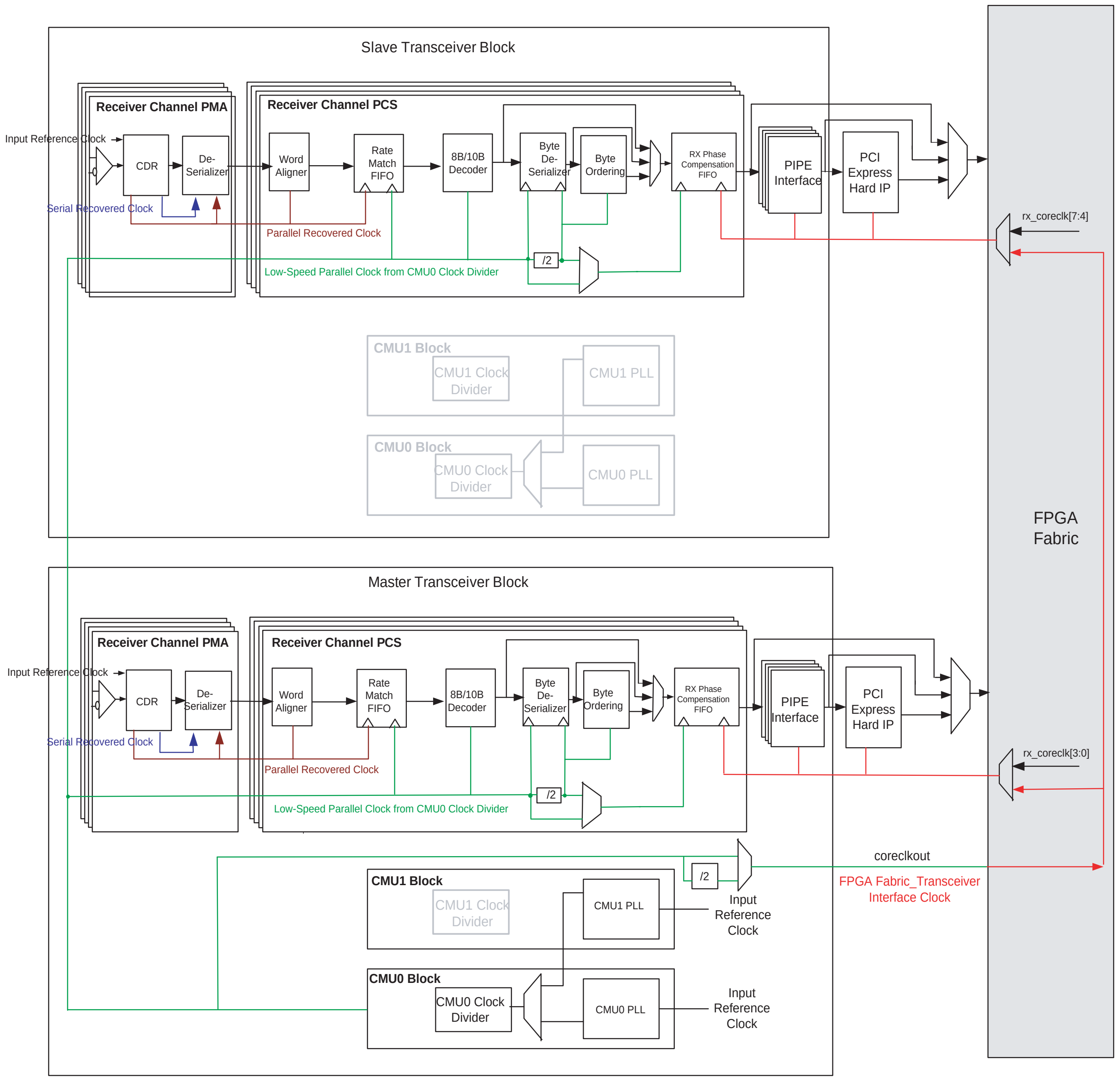
(1) 250 MHz when PCI Express hard IP is enabled.

×8 Bonded Channel Configuration

PCI Express (PIPE) ×8 functional mode supports ×8 receiver channel bonding configuration. The eight bonded channels are located in two transceiver blocks, referred to as the master transceiver block and slave transceiver block, with four channels each.

Figure 2–15 shows receiver datapath clocking in PCI Express (PIPE) ×8 bonded channel configuration.

Figure 2–15. Receiver Datapath Clocking in ×8 Bonded Channel Configuration



The CDR in each of the eight receiver channels recovers the serial clock from the received data on that channel. The serial recovered clock frequency is half the configured data rate. The serial recovered clock is divided within each channel's receiver PMA to generate the parallel recovered clock. The deserializer uses the serial recovered clock in the receiver PMA. The parallel recovered clock and deserialized data from the receiver PMA in each channel is forwarded to the receiver PCS in that channel.

The parallel recovered clock from the receiver PMA in each channel clocks the word aligner and write side of the rate match FIFO in that channel. The low-speed parallel clock from the `CMU0` clock divider of the master transceiver block clocks the read port of the rate match FIFO, 8B/10B decoder, and the write port of the byte deserializer (if enabled) in all eight channels. The low-speed parallel clock or its divide-by-two version (if byte-deserializer is enabled) clocks the write port of the receiver phase compensation FIFO in all eight channels. It is also driven on the `coreclkout` port as the FPGA fabric-transceiver interface clock. You can use the `coreclkout` signal to latch the receiver data and status signals in the FPGA fabric for all eight bonded channels.

Both the receiver phase compensation FIFO pointers and the control circuitry from `Channel 0` in the master transceiver block are shared by the receiver phase compensation FIFOs across all eight channels in PCI Express (PIPE) ×8 mode.

Table 2–8 shows the receiver datapath clock frequencies in PCI Express (PIPE) ×8 functional mode.

Table 2–8. Receiver Datapath Clock Frequencies in PCI Express (PIPE) ×8 Functional Modes

Functional Mode	Data Rate	Serial Recovered Clock Frequency	Parallel Recovered clock and Parallel Transmitter PCS Clock Frequency	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer	With Byte Serializer
PCI Express (PIPE) ×8 (Gen 1)	2.5 Gbps	1.25 GHz	250 MHz	N/A (1)	125 MHz

Note to Table 2–8:

(1) 250 MHz when PCI Express hard IP is enabled.

FPGA Fabric-Transceiver Interface Clocking

The FPGA fabric-transceiver interface clocks consist of clock signals from the FPGA fabric to the transceiver blocks and clock signals from the transceiver blocks to the FPGA fabric.

The FPGA fabric-transceiver interface clocks are subdivided into the following three categories:

- "Input Reference Clocks" on page 2–27
- "Phase Compensation FIFO Clocks" on page 2–28
- "Other Transceiver Clocks" on page 2–28

Input Reference Clocks

The CMU PLLs and receiver CDRs in each transceiver block derive the input reference from one of the following sources:

- `refclk0` and `refclk1` pins of the same transceiver block
- `refclk0` and `refclk1` pins of other transceiver blocks on the same side of the device using the inter-transceiver block (ITB) clock network

- CLK input pins on the FPGA global clock (GCLK) network
- Clock output pins from the left PLLs in the FPGA fabric

The input reference clock follows these guidelines:

- If the input reference clock to the CMU PLL or receiver CDR is provided through the `FPGA CLK` input pins or the clock output from the left PLLs in the FPGA fabric, the input reference clock becomes a part of the FPGA fabric-transceiver interface clocks.
- If the input reference clock is provided through the `FPGA CLK` input pins, the Quartus II software automatically routes the input reference clock on the FPGA fabric global clock (GCLK) network.
- If the input reference clock is provided through the output clock from a left PLL, the Quartus II software routes the input reference clock on a dedicated clock path from the left PLL to the CMU PLL or receiver CDR.

Phase Compensation FIFO Clocks

The transmitter and receiver phase compensation FIFOs in each channel ensure reliable transfer of data, control, and status signals between the FPGA fabric and the transceiver channels. The transceiver channel forwards the `tx_clkout` signal (in non-bonded modes) or the `coreclkout` signal (in bonded channel modes) to the FPGA fabric to clock the data and control signals into the transmitter phase compensation FIFO. The transceiver channel also forwards the recovered clock `rx_clkout` (in configurations without rate matcher) or `tx_clkout/coreclkout` (in configurations with rate matcher) to the FPGA fabric to clock the data and status signals from the receiver phase compensation FIFO into the FPGA fabric.

The phase compensation FIFO clocks form a part of the FPGA fabric-transceiver interface clocks and are routed on either a global clock resource (GCLK), regional clock resource (RCLK), or periphery clock (PCLK) resource in the FPGA fabric.

Other Transceiver Clocks

The following transceiver clocks form a part of the FPGA fabric-transceiver interface clocks:

- `cal_blk_clk`—calibration block clock
- `fixed_clk`—125 MHz fixed-rate clock used in PCI Express (PIPE) receiver detect circuitry

The Quartus II software automatically routes `fixed_clk` on the FPGA fabric global clock (GCLK) or regional clock (RCLK) network.

Table 2–9 summarizes FPGA fabric-transceiver interface clocks.

Table 2–9. FPGA Fabric-Transceiver Interface Clocks

Clock Name	Clock Description	Interface Direction	FPGA Fabric Clock Resource Utilization *(1)*
`pll_inclk`	CMU PLL input reference clock when driven from an FPGA CLK input pin	FPGA fabric-to-transceiver	GCLK
`rx_cruclk`	Receiver CDR input reference clock when driven from an FPGA CLK input pin	FPGA fabric-to-transceiver	GCLK
`tx_clkout`	Phase compensation FIFO clock	Transceiver-to-FPGA fabric	GCLK, RCLK, PCLK
`coreclkout`	Phase compensation FIFO clock	Transceiver-to-FPGA fabric	GCLK, RCLK, PCLK
`rx_clkout`	Phase compensation FIFO clock	Transceiver-to-FPGA fabric	GCLK, RCLK, PCLK
`fixed_clk`	PCI Express receiver detect clock	FPGA fabric-to-transceiver	GCLK, RCLK

Note to Table 2–9:

(1) For more information about GCLK, RCLK, and PCLK resources available in each device, refer to the *Clock Networks and PLLs in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

"FPGA Fabric-Transmitter Interface Clocking" (below) and "FPGA Fabric-Receiver Interface Clocking" on page 2–42 describe the criteria and methodology to share transmitter and receiver phase compensation FIFO clocks in order to reduce the GCLK, RCLK, and PCLK resource utilization in your design.

FPGA Fabric-Transmitter Interface Clocking

The transmitter phase compensation FIFO compensates for the phase difference between the FPGA fabric clock (phase compensation FIFO write clock) and the parallel transmitter PCS clock (phase compensation FIFO read clock). The transmitter phase compensation FIFO write clock forms the FPGA fabric-transmitter interface clock. The phase compensation FIFO write and read clocks must have exactly the same frequency, in other words, 0 PPM frequency difference.

Arria II GX transceivers provide the following two options for selecting the transmitter phase compensation FIFO write clock:

- Quartus II software-selected transmitter phase compensation FIFO write clock
- User-selected transmitter phase compensation FIFO write clock

Quartus II Software-Selected Transmitter Phase Compensation FIFO Write Clock

If you do not select the `tx_coreclk` port in the ALTGX MegaWizard™ Plug-In Manager, the Quartus II software automatically selects the transmitter phase compensation FIFO write clock for each channel in that ALTGX instance. The Quartus II software selects the FIFO write clock depending on the channel configuration.

Non-Bonded Channel Configuration

In the non-bonded channel configuration, the transmitter channels may or may not be identical. Identical transmitter channels are defined as channels that have exactly the same CMU PLL input reference clock source, have exactly the same CMU PLL configuration, and have exactly the same transmitter PMA and PCS configuration.

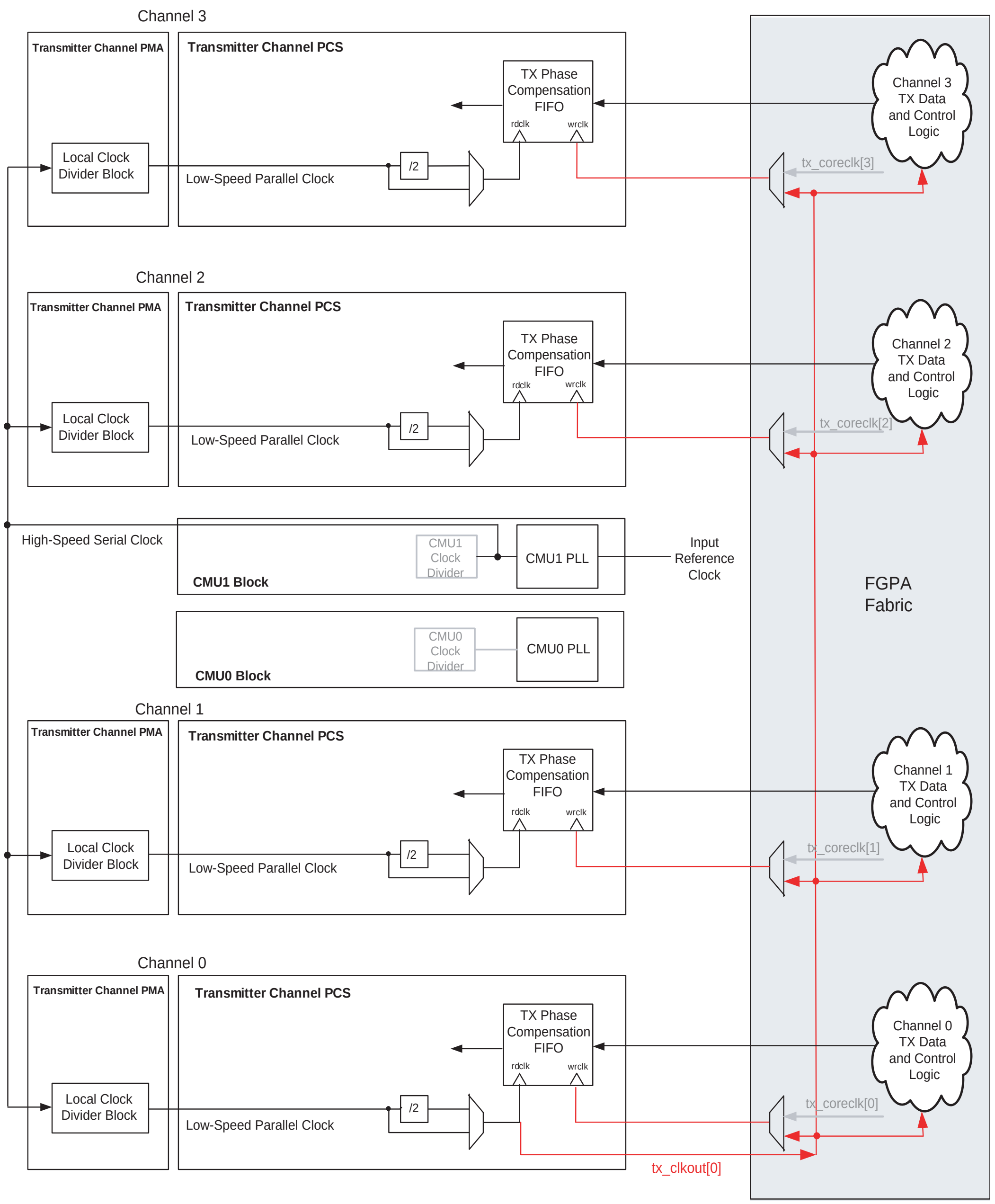
☞ Identical transmitter channels may have different transmitter voltage output differential (VOD) or pre-emphasis settings.

Example 1: Four Identical Channels in a Transceiver Block

If all four channels within a transceiver block are identical, the Quartus II software automatically drives the write port of the transmitter phase compensation FIFO in all four channels with `tx_clkout[0]`, as shown in Figure 2–16. Use the `tx_clkout[0]` signal to clock the transmitter data and control logic for all four channels in the FPGA fabric.

☞ This configuration uses only one FPGA GCLK, RCLK, or PCLK resource for `tx_clkout[0]`.

Figure 2–16. Four Identical Channels in a Transceiver Block for Example 1



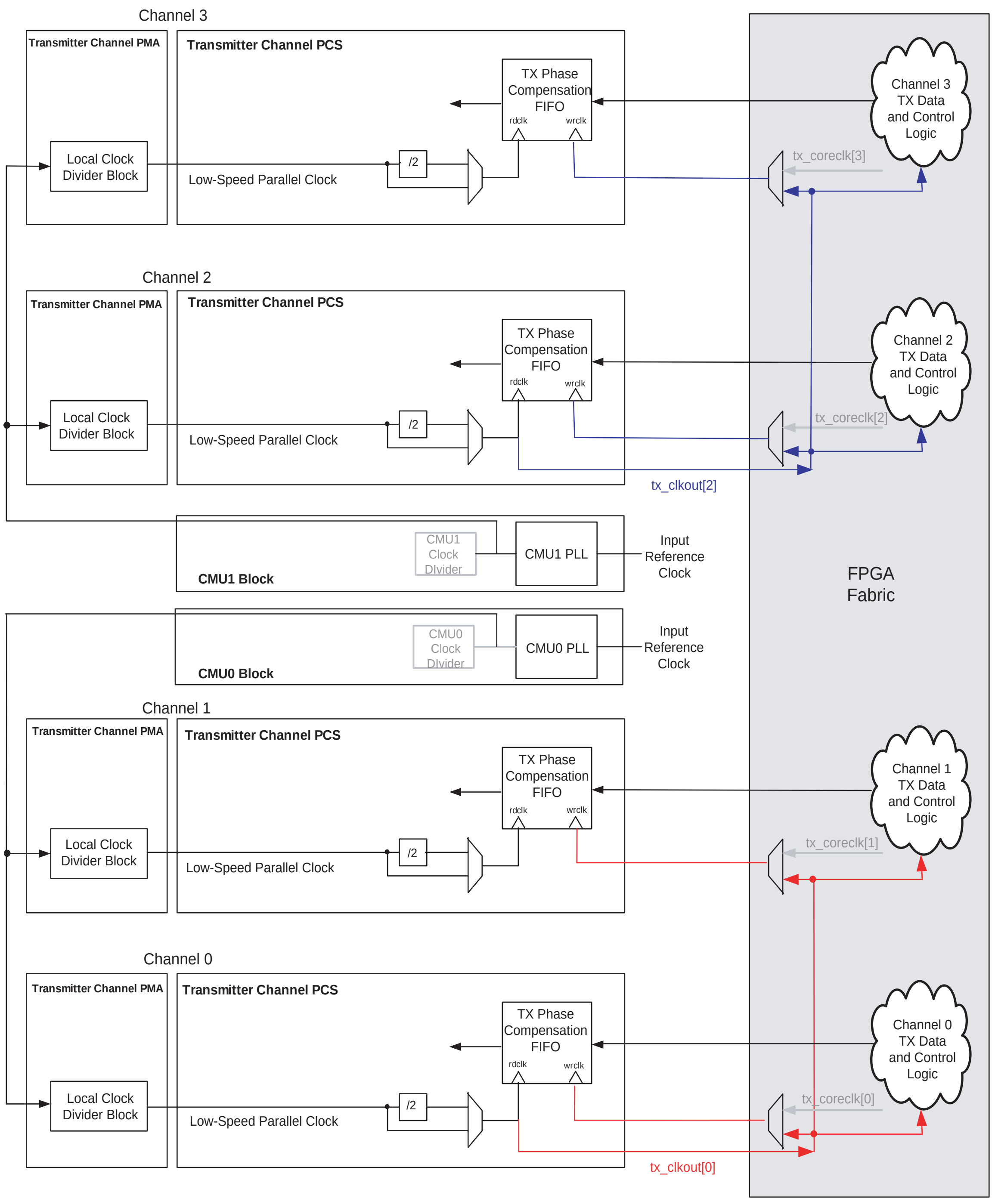
Example 2: Two Groups of Two Identical Channels in a Transceiver Block

Example 2 assumes channels 0 and 1, driven by `CMU0 PLL` in a transceiver block, are identical. Also, channels 2 and 3, driven by `CMU1 PLL` in the same transceiver block, are identical. In this case, the Quartus II software automatically drives the write port of the transmitter phase compensation FIFO in channels 0 and 1 with the `tx_clkout[0]` signal. It also drives the write port of the transmitter phase compensation FIFO in channels 2 and 3 with the `tx_clkout[2]` signal. Use the `tx_clkout[0]` signal to clock the transmitter data and control logic for channels 0 and 1 in the FPGA fabric. Use the `tx_clkout[2]` signal to clock the transmitter data and control logic for channels 2 and 3 in the FPGA fabric.

This configuration uses two FPGA clock resources (global, regional, or both), one for the `tx_clkout[0]` signal and one for the `tx_clkout[2]` signal.

Figure 2–17 shows FPGA fabric-transmitter interface clocking for Example 2.

Figure 2–17. FPGA Fabric-Transmitter Interface Clocking for Example 2



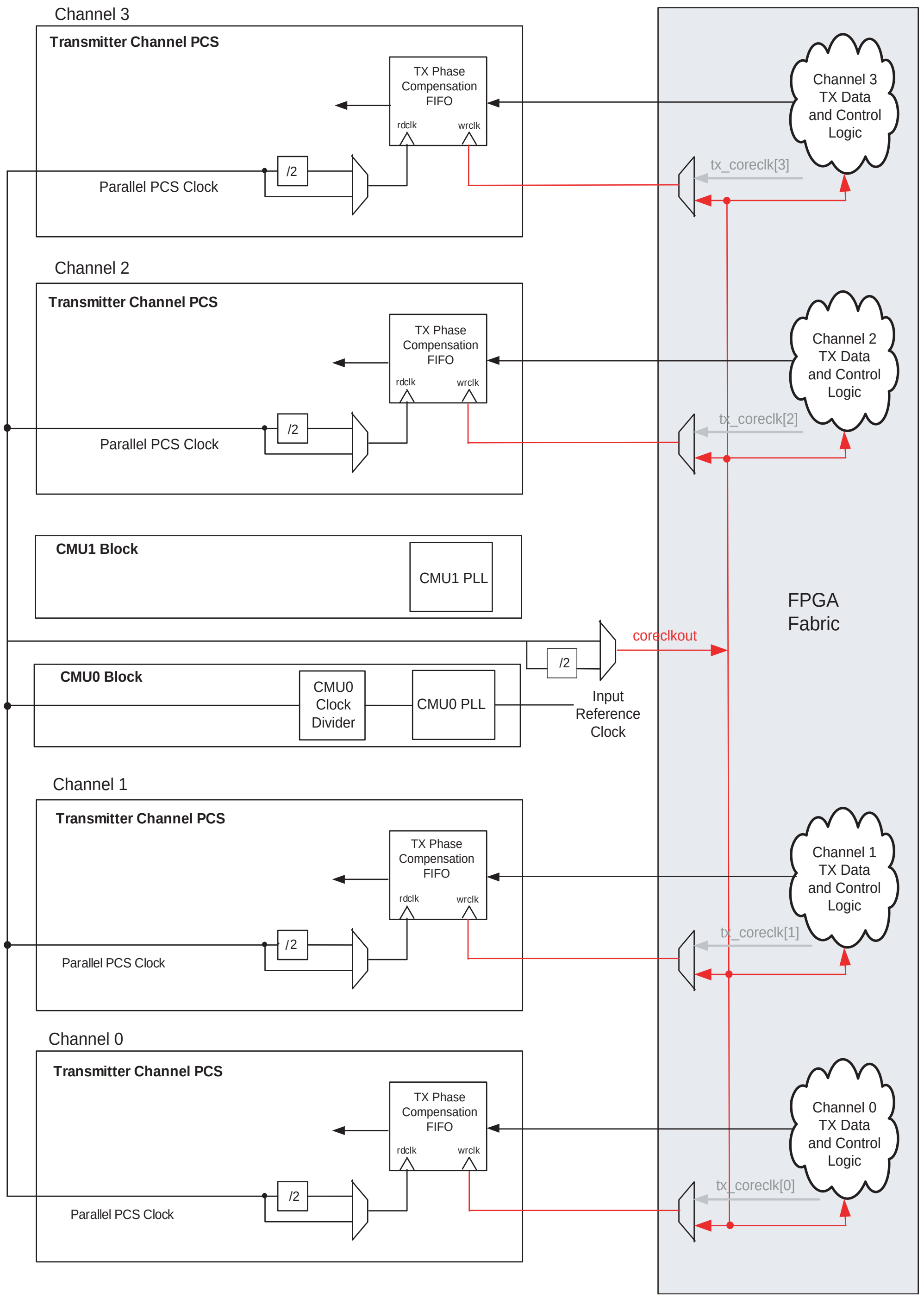
Bonded Channel Configuration

In the ×4 bonded channel configuration, all four channels within the transceiver block are identical. The Quartus II software automatically drives the write port of the transmitter phase compensation FIFO in all four channels with the `coreclkout` signal. Use the `coreclkout` signal to clock the transmitter data and control logic for all four channels in the FPGA fabric.

In the ×8 bonded channel configuration, all eight channels across two transceiver blocks are identical. The Quartus II software automatically drives the write port of the transmitter phase compensation FIFO in all eight channels with the `coreclkout` signal from the master transceiver block. Use the `coreclkout` signal to clock the transmitter data and control logic for all eight channels in the FPGA fabric.

Figure 2–18 shows FPGA fabric-transmitter interface clocking in an ×4 bonded channel configuration.

Figure 2–18. FPGA Fabric-Transmitter Interface Clocking in an ×4 Bonded Channel Configuration



Limitations of the Quartus II Software-Selected Transmitter Phase Compensation FIFO Write Clock

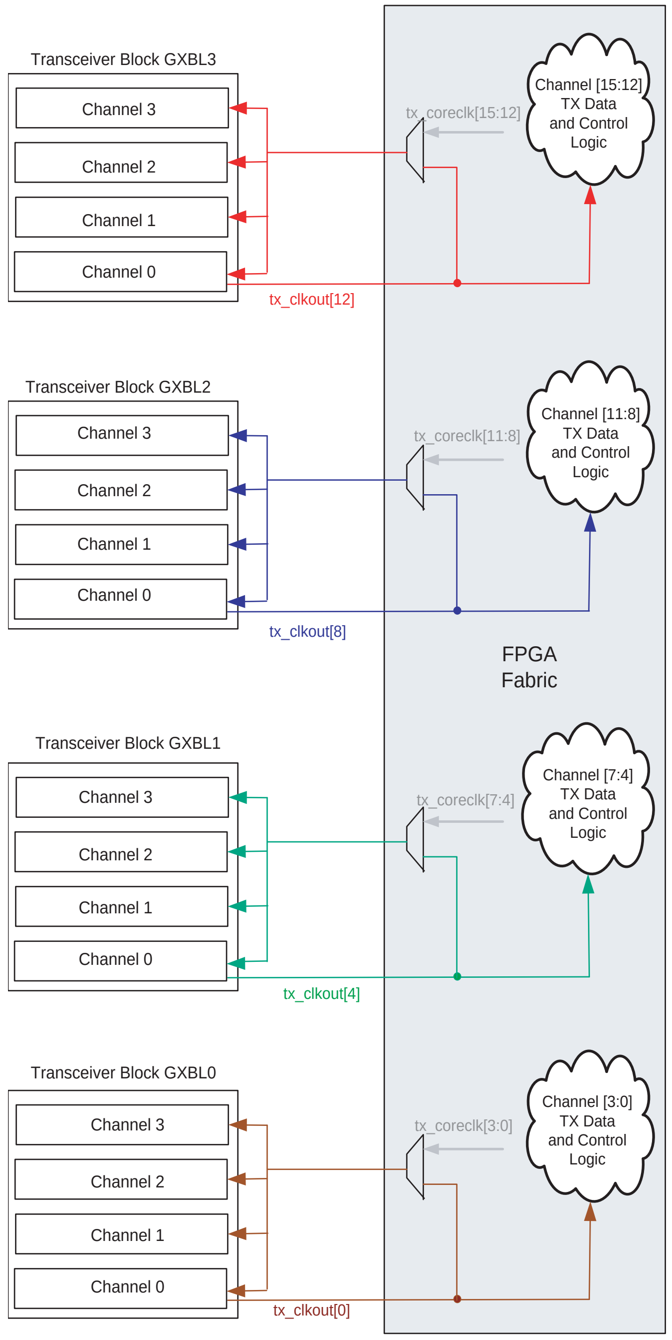
The Quartus II software uses a single `tx_clkout` signal to clock the transmitter phase compensation FIFO write port of all identical channels within a transceiver block. This results in one global or regional clock resource being used for each group of identical channels within a transceiver block.

For identical channels located across transceiver blocks, the Quartus II software does not use a single `tx_clkout` signal to clock the write port of the transmitter phase compensation FIFOs for all channels. Instead, it uses one `tx_clkout` signal for each group of identical channels per transceiver block. This results in higher clock resource utilization.

Example 3: Sixteen Identical Channels Across Four Transceiver Blocks

Consider 16 identical transmitter channels located across four transceiver blocks, as shown in Figure 2–19. The Quartus II software uses `tx_clkout` from Channel 0 in each transceiver block to clock the write port of the transmitter phase compensation FIFO in all four channels of that transceiver block. This results in four clocks resources (global, regional, or both) being used, one for each transceiver block.

Figure 2–19. Sixteen Identical Channels across Four Transceiver Blocks for Example 3



Because all 16 channels are identical, using a single `tx_clkout` to clock the transmitter phase compensation FIFO in all 16 channels results in only one global or regional clock resource being used instead of four. To achieve this, you must choose the transmitter phase compensation FIFO write clocks instead of the Quartus II software automatic selection, as described in "User-Selected Transmitter Phase Compensation FIFO Write Clock" on page 2–37.

User-Selected Transmitter Phase Compensation FIFO Write Clock

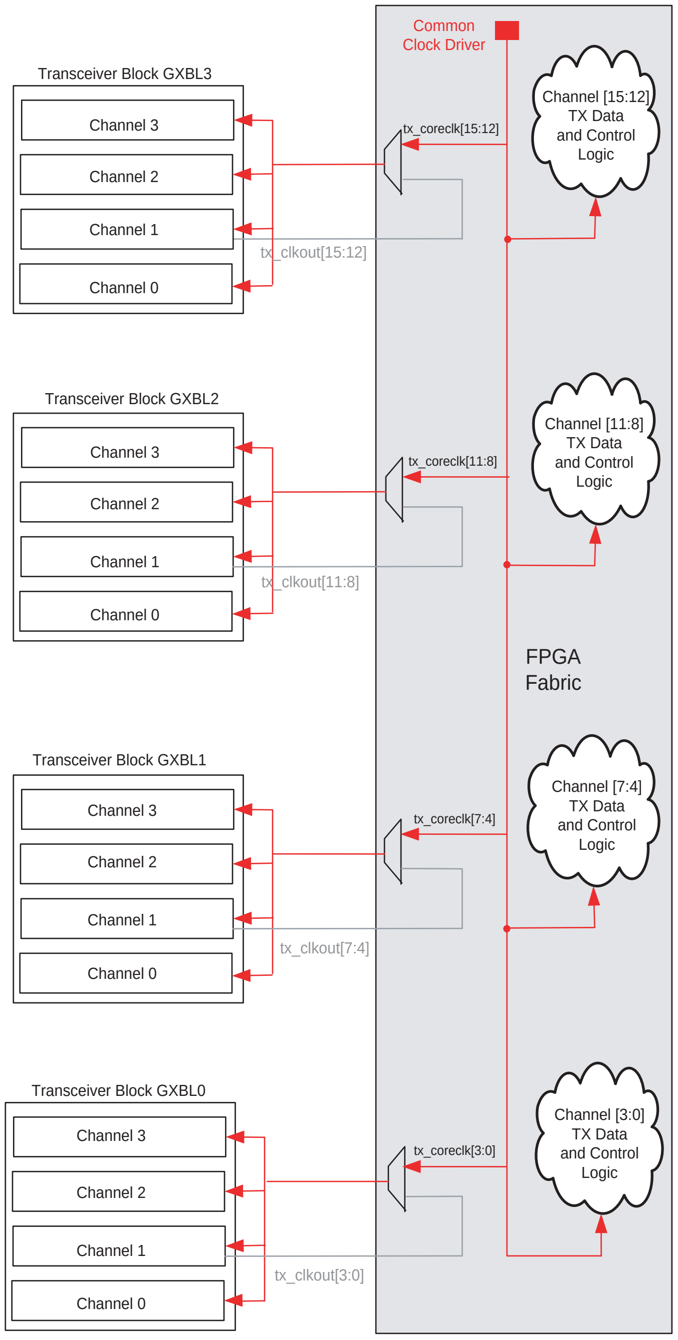
The ALTGX MegaWizard Plug-In Manager provides an optional port named `tx_coreclk` for each instantiated transmitter channel. If you enable this port, the Quartus II software does not automatically select the transmitter phase compensation FIFO write clock source. Instead, the signal that you drive on the `tx_coreclk` port of the channel clocks the write side of its transmitter phase compensation FIFO.

Use the flexibility of selecting the transmitter phase compensation FIFO write clock to reduce clock resource utilization (global, regional, or both). You can connect the `tx_coreclk` ports of all identical channels in your design and drive them using a common clock driver that has 0 PPM frequency difference with respect to the FIFO read clocks of all your channels. Use the common clock driver to clock the transmitter data and control logic in the FPGA fabric for all identical channels. This FPGA fabric-transceiver interface clocking scheme utilizes only one global or regional clock resource for all identical channels in your design.

Example 4: Sixteen Identical Channels Across Four Transceiver Blocks

Figure 2–20 shows 16 identical transmitter channels located across four transceiver blocks. The `tx_coreclk` ports of all 16 transmitter channels are connected together and driven by a common clock driver. This common clock driver also drives the transmitter data and control logic of all 16 transmitter channels in the FPGA fabric. Only one global or regional clock resource is used with this clocking scheme, compared to four clock resources (global, regional, or both) needed without the `tx_coreclk` ports (the Quartus II software-selected transmitter phase compensation FIFO write clock).

Figure 2–20. Sixteen Identical Channels across Four Transceiver Blocks for Example 4



Common Clock Driver Selection Rules

The common clock driver driving the `tx_coreclk` ports of all identical channels must have 0 PPM frequency difference with respect to the transmitter phase compensation FIFO read clocks of these channels. If there is any frequency difference between the FIFO write clock (`tx_coreclk`) and the FIFO read clock, the FIFO overflows or under runs, resulting in corrupted data transfer between the FPGA fabric and the transmitter.

Table 2–10 shows the transmitter phase compensation FIFO read clocks that the Quartus II software selects in various configurations.

Table 2–10. Transmitter Phase Compensation FIFO Read Clocks

Configuration	Transmitter Phase Compensation FIFO Read Clock	
	Without Byte Serializer	**With Byte Serializer**
Non-Bonded Channel Configuration	Parallel transmitter PCS clock from the local clock divider in the associated channel (`tx_clkout`)	Divide-by-two version of the parallel transmitter PCS clock from the local clock divider in the associated channel (`tx_clkout`)
×4 Bonded Channel Configuration	Low-speed parallel clock from the `CMU0` clock divider of the associated transceiver block (`coreclkout`)	Divide-by-two version of the low-speed parallel clock from the `CMU0` clock divider of the associated transceiver block (`coreclkout`)
×8 Bonded Channel Configuration	Low-speed parallel clock from the `CMU0` clock divider of the master transceiver block (`coreclkout` from master transceiver block)	Divide-by-two version of the low-speed parallel clock from the `CMU0` clock divider of the master transceiver block (`coreclkout` from master transceiver block)

To ensure that you understand the 0 PPM clock driver rule, the Quartus II software expects the following set of user assignments whenever the `tx_coreclk` port is used to drive the transmitter phase compensation FIFO write clock:

- GXB 0 PPM Core Clock Setting

Failure to make this assignment when using the `tx_coreclk` port results in a Quartus II compilation error.

GXB 0 PPM Core Clock Setting

The GXB 0 PPM core clock setting is intended for advanced users who know the clocking configuration of the entire system and want to reduce the FPGA fabric global and regional clock resource utilization. The GXB 0 PPM core clock setting allows the following clock drivers to drive the `tx_coreclk` ports:

- `tx_clkout` in non-bonded channel configurations
- `coreclkout` in bonded channel configurations
- FPGA CLK input pins
- Transceiver `REFCLK` pins
- Clock output from the left corner PLLs (PLL_1 and PLL_4)

The Quartus II software does not allow gated clocks or clocks generated in FPGA logic to drive the `tx_coreclk` ports.

Because the GXB 0 PPM core clock setting allows FPGA CLK input pins and transceiver `REFCLK` pins as the clock driver, the Quartus II compiler cannot determine if there is a 0 PPM difference between the FIFO write clock and read clock for each channel.

You must ensure that the clock driver for all connected `tx_coreclk` ports has a 0 PPM difference with respect to the FIFO read clock in those channels.

Figure 2–20 shows the Quartus II assignments that you must make in the assignment editor.

Table 2–11. Quartus II Assignments

Assignment	**Description**
From:	Full design hierarchy name of one of the following clock drivers that you choose to drive the `tx_coreclk` ports of all identical channels *(1)*: ■ `tx_clkout` ■ `coreclkout` ■ FPGA CLK input pins ■ Transceiver `REFCLK` pins ■ Clock output from left corner PLLs
To:	`tx_dataout` pins of all identical channels whose `tx_coreclk` ports are connected together and driven by the 0 PPM clock driver.
Assignment Name:	GXB 0 PPM Core Clock Setting
Value:	ON

Note to Table 2–11:

(1) You can find the full hierarchy name of the 0 PPM clock driver using the **Node Finder** feature in the Quartus II Assignment Editor.

Example 5: Sixteen Identical Channels Across Four Transceiver Blocks

Figure 2–21 shows 16 identical transmitter channels located across four transceiver blocks. The `tx_coreclk` ports of all 16 transmitter channels are connected together and driven by the `tx_clkout[4]` signal from channel 0 in transceiver block `GXBL1`. The `tx_clkout[4]` signal also drives the transmitter data and control logic of all 16 transmitter channels in the FPGA fabric. Only one global clock resource is used by the `tx_clkout[4]` signal with this clocking scheme.

Figure 2–21. Sixteen Identical Channels across Four Transceiver Blocks for Example 5

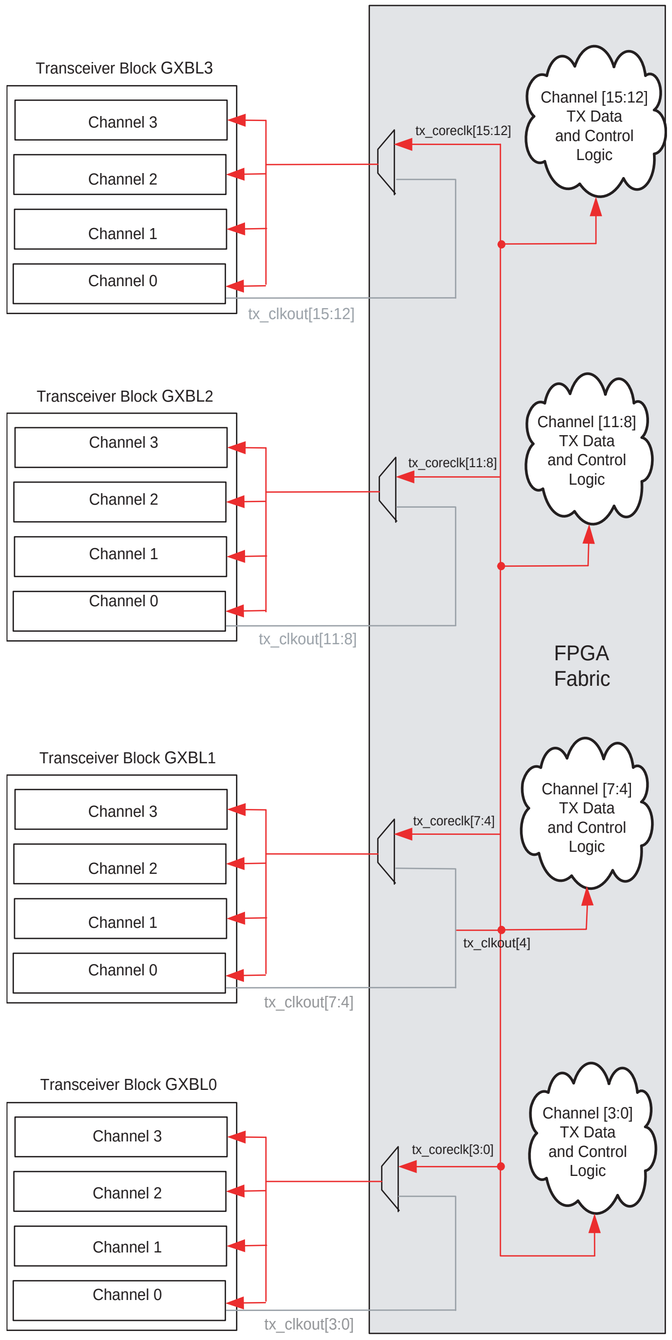


Table 2–11 shows the Quartus II assignments that you must make for the clocking scheme shown in Figure 2–21.

Table 2–12. Quartus II Assignments

Assignment	Description
From:	`top_level/top_xcvr_instance1/altgx_component/tx_clkout[4]` *(1)*
To:	`tx_dataout[15..0]`
Assignment Name:	GXB 0 PPM Core Clock Setting
Value:	ON

Note to Table 2–12:

(1) This is an example design hierarchy path for the `tx_clkout[4]` signal.

FPGA Fabric-Receiver Interface Clocking

The receiver phase compensation FIFO compensates for the phase difference between the parallel receiver PCS clock (FIFO write clock) and the FPGA fabric clock (FIFO read clock). The receiver phase compensation FIFO read clock forms the FPGA fabric-receiver interface clock. The FIFO write and read clocks must have exactly the same frequency, in other words, 0 PPM frequency difference.

Arria II GX transceivers provide the following two options for selecting the receiver phase compensation FIFO read clock:

- Quartus II software-selected receiver phase compensation FIFO read clock
- User-selected receiver phase compensation FIFO read clock

Quartus II Software-Selected Receiver Phase Compensation FIFO Read Clock

If you do not select the `rx_coreclk` port in the ALTGX MegaWizard Plug-In Manager, the Quartus II software automatically selects the receiver phase compensation FIFO read clock for each channel in that ALTGX instance. The Quartus II software selects the FIFO read clock depending on the channel configuration.

Non-Bonded Channel Configuration with Rate Matcher

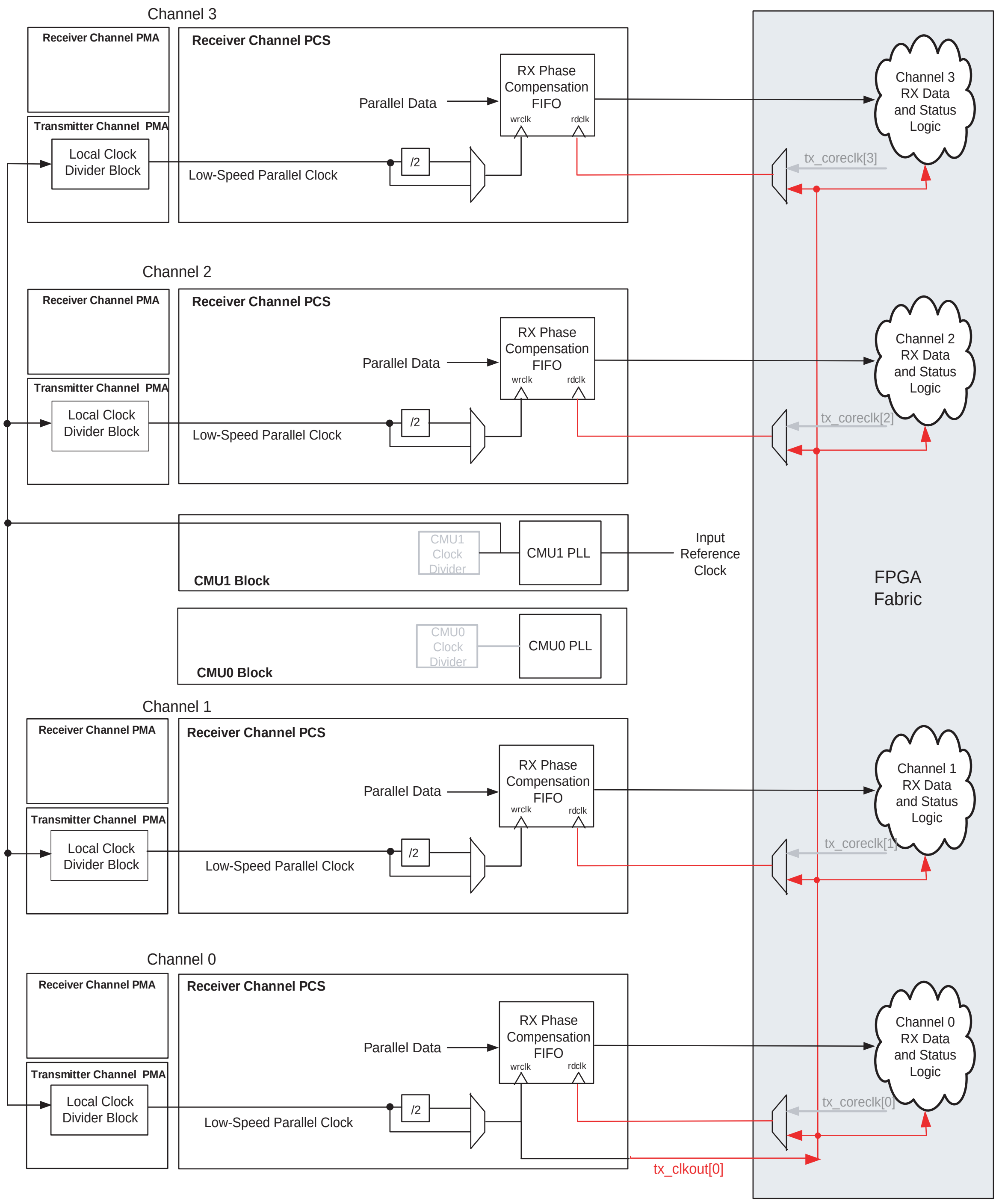
In the non-bonded channel configuration, the transceiver channels may or may not be identical. Identical transceiver channels are defined as channels that have the same CMU PLL and receiver CDR input reference clock source, have exactly the same CMU PLL and receiver CDR configuration, and have exactly the same PMA and PCS configuration.

Example 6: Four Identical Channels in a Transceiver Block

If all four channels within a transceiver block are identical, the Quartus II software automatically drives the read port of the receiver phase compensation FIFO in all four channels with `tx_clkout[0]`, as shown in Figure 2–22. Use the `tx_clkout[0]` signal to latch the receiver data and status signals from all four channels in the FPGA fabric.

This configuration uses only one FPGA global or regional clock resource for `tx_clkout[0]`.

Figure 2–22. Four Identical Channels in a Transceiver Block for Example 6



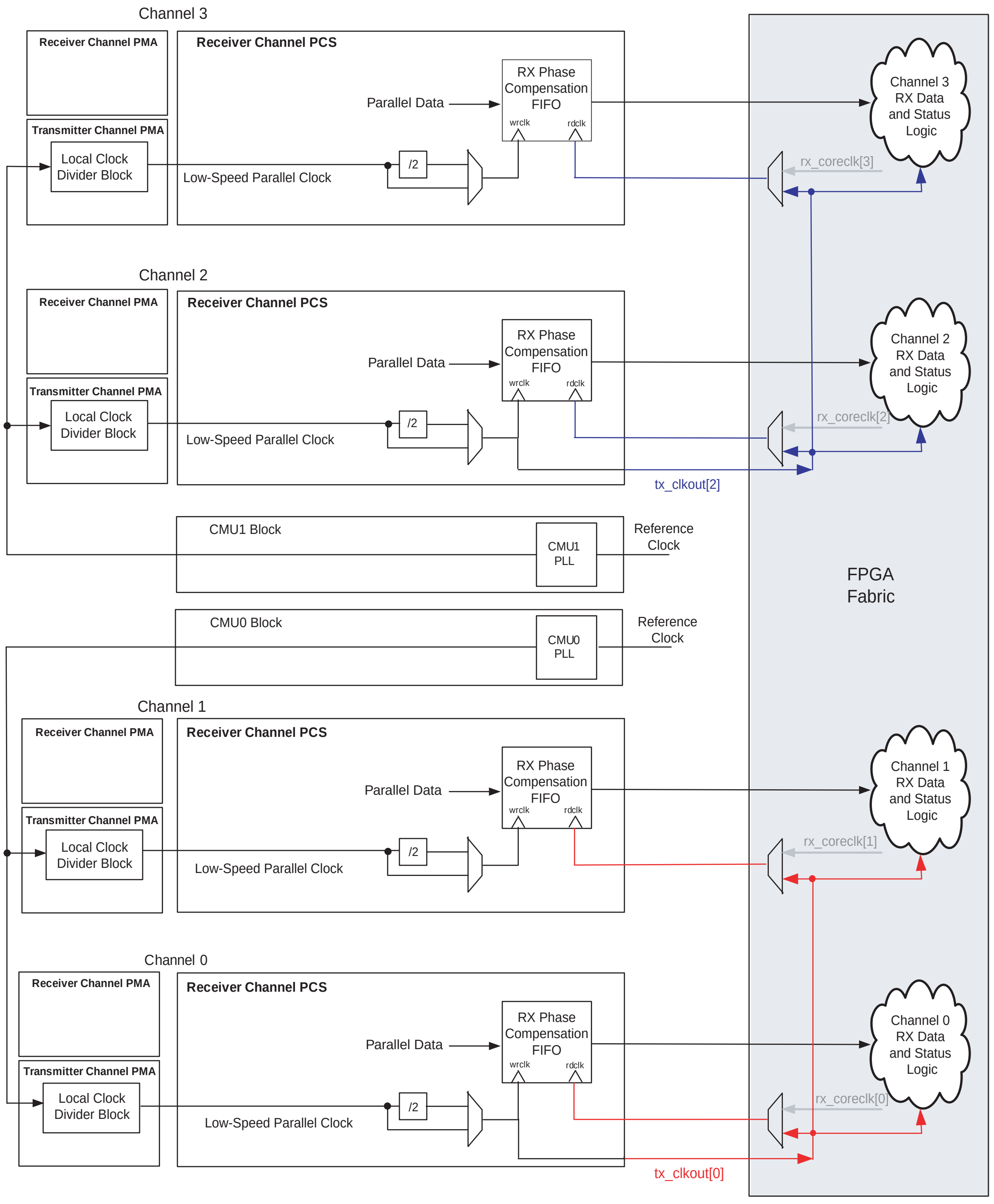
Example 7: Two Groups of Two Identical Channels in a Transceiver Block

Example 7 assumes channels 0 and 1, driven by `CMU0 PLL` in a transceiver block, are identical. Also, channels 2 and 3, driven by `CMU1 PLL` in the same transceiver block, are identical. In this case, the Quartus II software automatically drives the read port of the receiver phase compensation FIFO in channels 0 and 1 with the `tx_clkout[0]` signal. It also drives the read port of the receiver phase compensation FIFO in channels 2 and 3 with the `tx_clkout[2]` signal. Use the `tx_clkout[0]` signal to latch the receiver data and status signals from channels 0 and 1 in the FPGA fabric. Use the `tx_clkout[2]` signal to latch the receiver data and status signals from channels 2 and 3 in the FPGA fabric.

☞ This configuration uses two FPGA clock resources (global, regional, or both), one for the `tx_clkout[0]` signal and one for the `tx_clkout[2]` signal.

Figure 2–23 shows FPGA fabric-receiver interface clocking for Example 7.

Figure 2–23. FPGA Fabric-Receiver Interface Clocking for Example 7



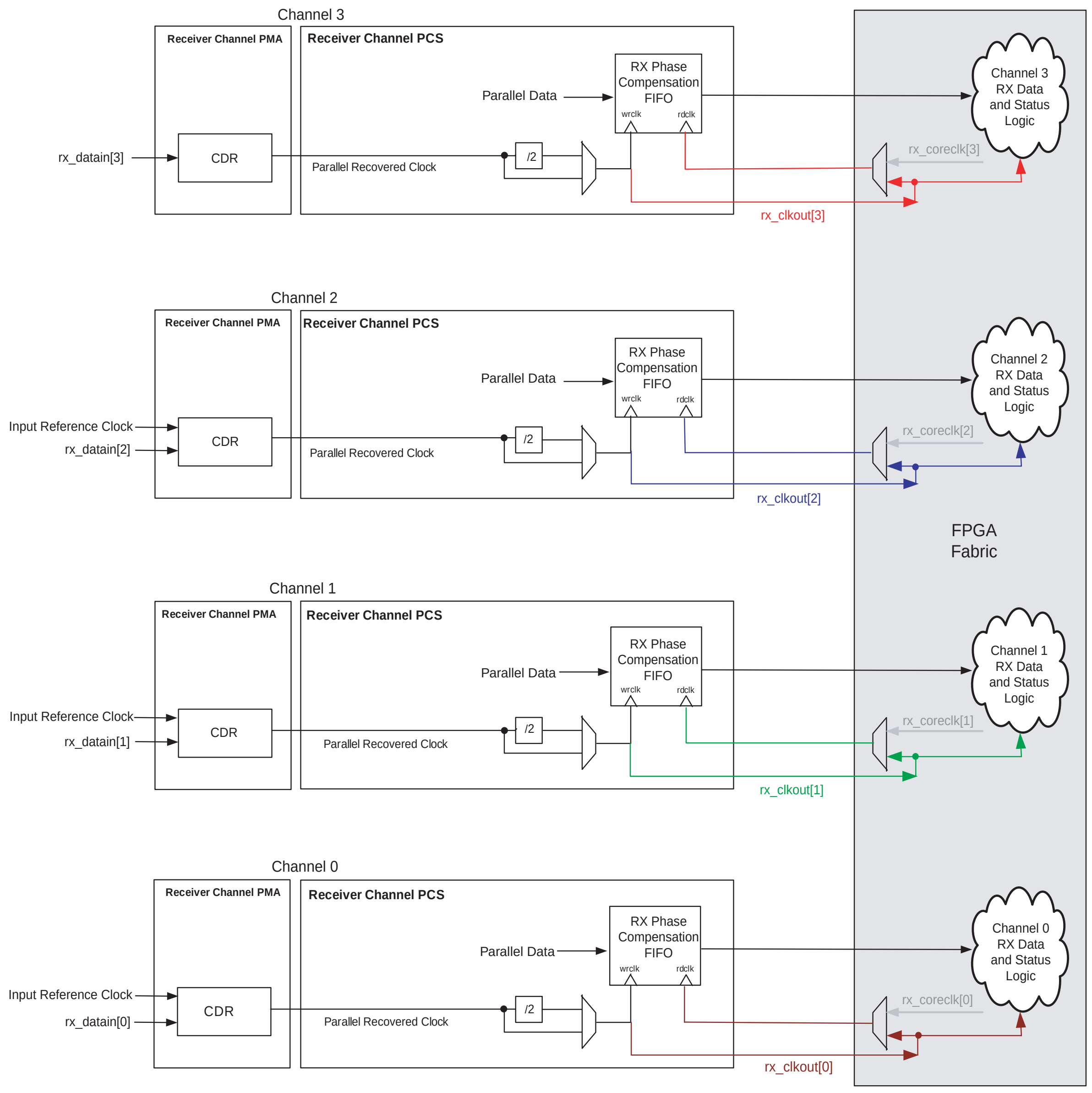
Non-Bonded Channel Configuration without Rate Matcher

In non-bonded channel configuration without rate matcher, the Quartus II software cannot determine if the incoming serial data in all channels have a 0 PPM frequency difference. The Quartus II software automatically drives the read port of the receiver phase compensation FIFO in each channel with the recovered clock driven on the `rx_clkout` port of that channel. Use the `rx_clkout` signal from each channel to latch its receiver data and status signals in the FPGA fabric.

This configuration uses one FPGA clock resource (global, regional, or both) per channel for the `rx_clkout` signal.

Figure 2–24 shows the FPGA fabric-receiver interface clocking for non-bonded channel configurations without rate matcher.

Figure 2–24. FPGA Fabric-Receiver Interface Clocking for Non-Bonded Channel Configurations without Rate Matcher



Bonded Channel Configuration

All bonded transceiver channel configurations have a rate matcher in the receiver data path.

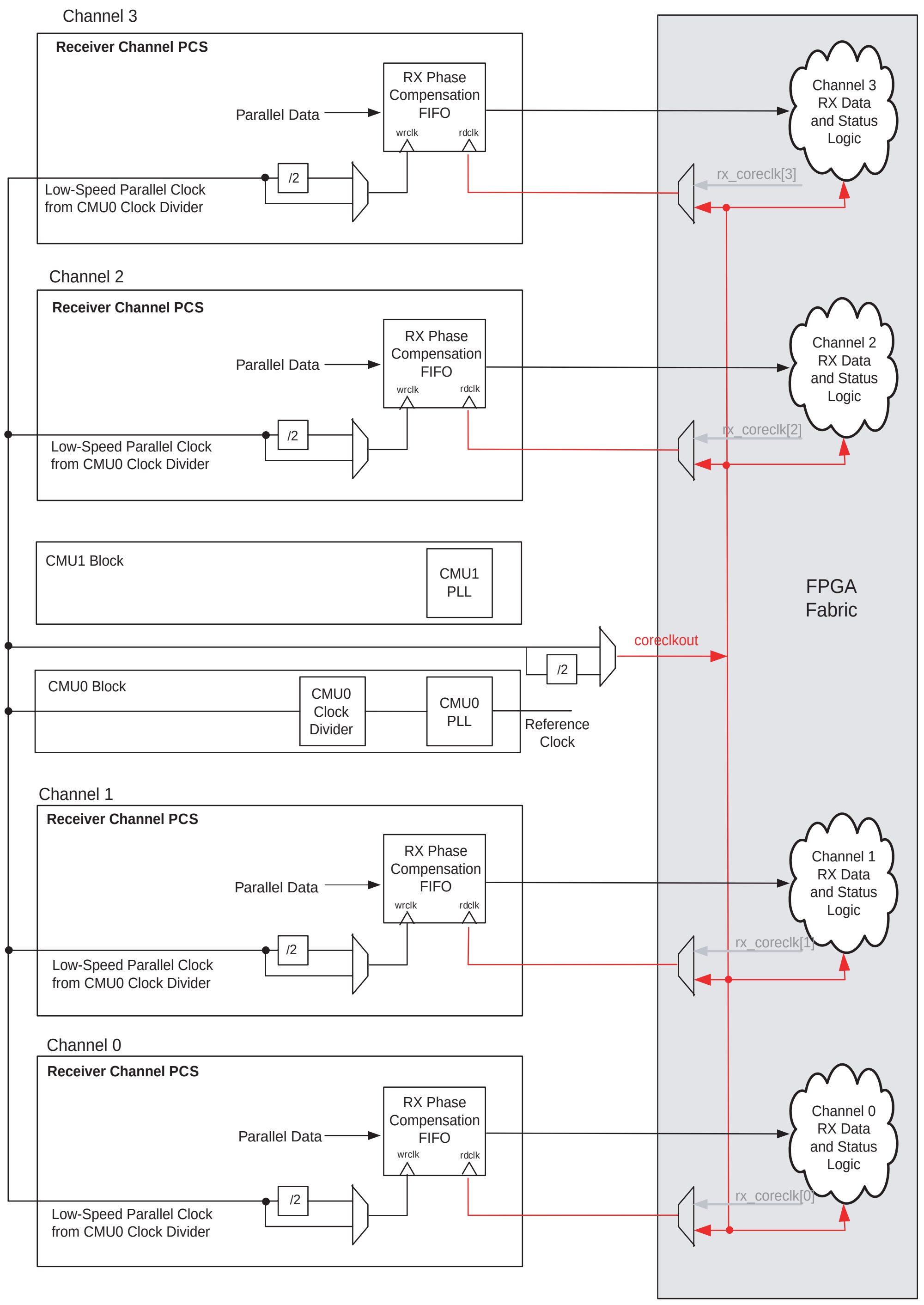
In the ×4 bonded channel configurations, the Quartus II software automatically drives the read port of the receiver phase compensation FIFO in all four channels with the `coreclkout` signal. Use the `coreclkout` signal to latch the receiver data and status signals from all four channels in the FPGA fabric.

In ×8 bonded channel configurations, the Quartus II software automatically drives the read port of the receiver phase compensation FIFO in all eight channels with the `coreclkout` signal from the master transceiver block. Use the `coreclkout` signal to latch the receiver data and status signals from all eight channels in the FPGA fabric.

☞ This configuration uses one FPGA global or regional clock resource per bonded link for the `coreclkout` signal.

Figure 2–25 shows FPGA fabric-receiver interface clocking in an ×4 bonded channel configuration.

Figure 2–25. FGPA Fabric-Receiver Interface Clocking in an ×4 Bonded Channel Configuration


Channel 3
Receiver Channel PCS
RX Phase Compensation FIFO
wrclk
rdclk
Parallel Data
/2
Low-Speed Parallel Clock from CMU0 Clock Divider
Channel 3 RX Data and Status Logic
rx_coreclk[3]
Channel 2
Receiver Channel PCS
RX Phase Compensation FIFO
wrclk
rdclk
Parallel Data
/2
Low-Speed Parallel Clock from CMU0 Clock Divider
Channel 2 RX Data and Status Logic
rx_coreclk[2]
CMU1 Block
CMU1 PLL
FPGA Fabric
coreclkout
/2
CMU0 Block
CMU0 Clock Divider
CMU0 PLL
Reference Clock
Channel 1
Receiver Channel PCS
RX Phase Compensation FIFO
wrclk
rdclk
Parallel Data
/2
Low-Speed Parallel Clock from CMU0 Clock Divider
Channel 1 RX Data and Status Logic
rx_coreclk[1]
Channel 0
Receiver Channel PCS
RX Phase Compensation FIFO
wrclk
rdclk
Parallel Data
/2
Low-Speed Parallel Clock from CMU0 Clock Divider
Channel 0 RX Data and Status Logic
rx_coreclk[0]

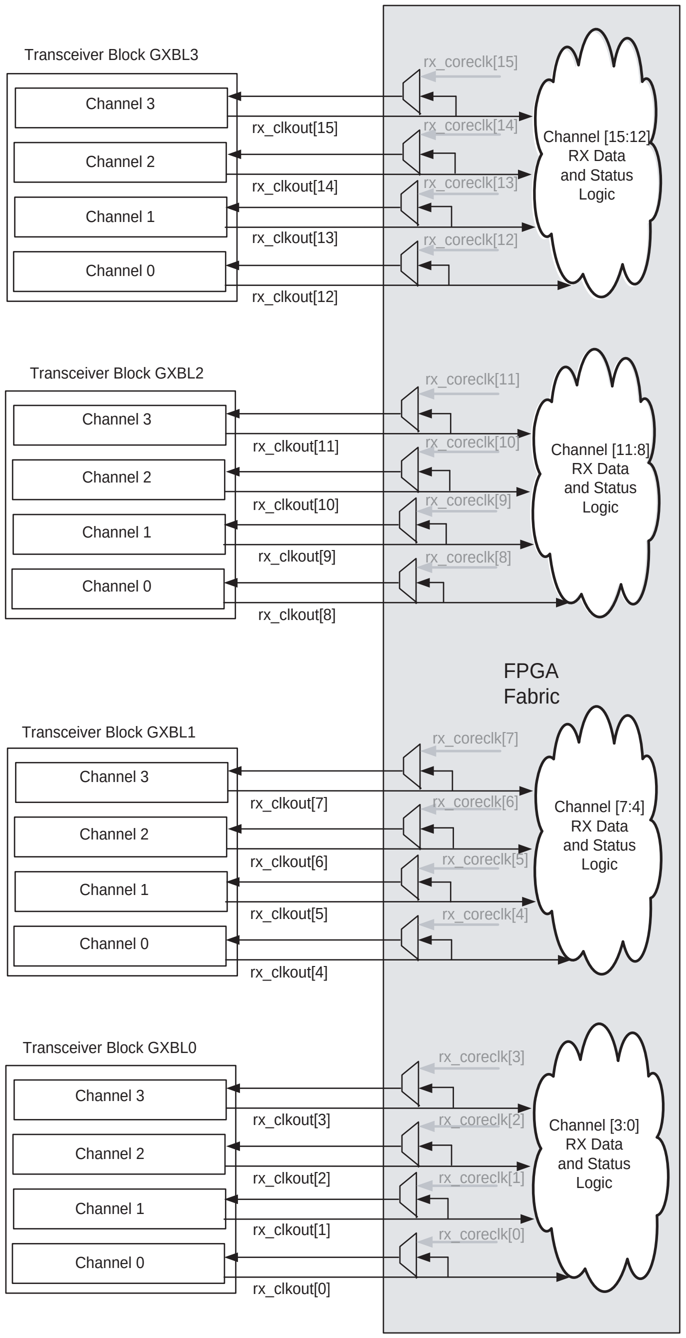
Limitations of Quartus II Software-Selected Receiver Phase Compensation FIFO Read Clock

In non-bonded channel configurations without rate matcher, the Quartus II software cannot determine if the incoming serial data in all channels has a 0 PPM frequency difference. The Quartus II software uses the recovered clock `rx_clkout` signal from each channel to clock the read port of its receiver phase compensation FIFO. This results in one clock resource (global, regional, or both) being used per channel for the `rx_clkout` signal.

Example 8: Sixteen Channels Across Four Transceiver Blocks

Consider 16 non-bonded receiver channels without rate matcher located across four transceiver blocks, as shown in Figure 2–26. The incoming serial data for all 16 channels has a 0 PPM frequency difference with respect to each other. The Quartus II software uses `rx_clkout` from each channel to clock the read port of its receiver phase compensation FIFO. This results in 16 clocks resources (global, regional, or both) being used, one for each channel.

Figure 2–26. Sixteen Non-Bonded Receiver Channels without Rate Matcher for Example 8



Since the recovered clock `rx_clkout` signals from all 16 channels have a 0 PPM frequency difference, you can use a single `rx_clkout` to clock the receiver phase compensation FIFO in all 16 channels. This results in only one global or regional clock resource being used instead of 16. To achieve this, you must select the receiver phase compensation FIFO read clocks instead of the Quartus II software automatic selection, as described in "User-Selected Receiver Phase Compensation FIFO Read Clock" on page 2–51.

User-Selected Receiver Phase Compensation FIFO Read Clock

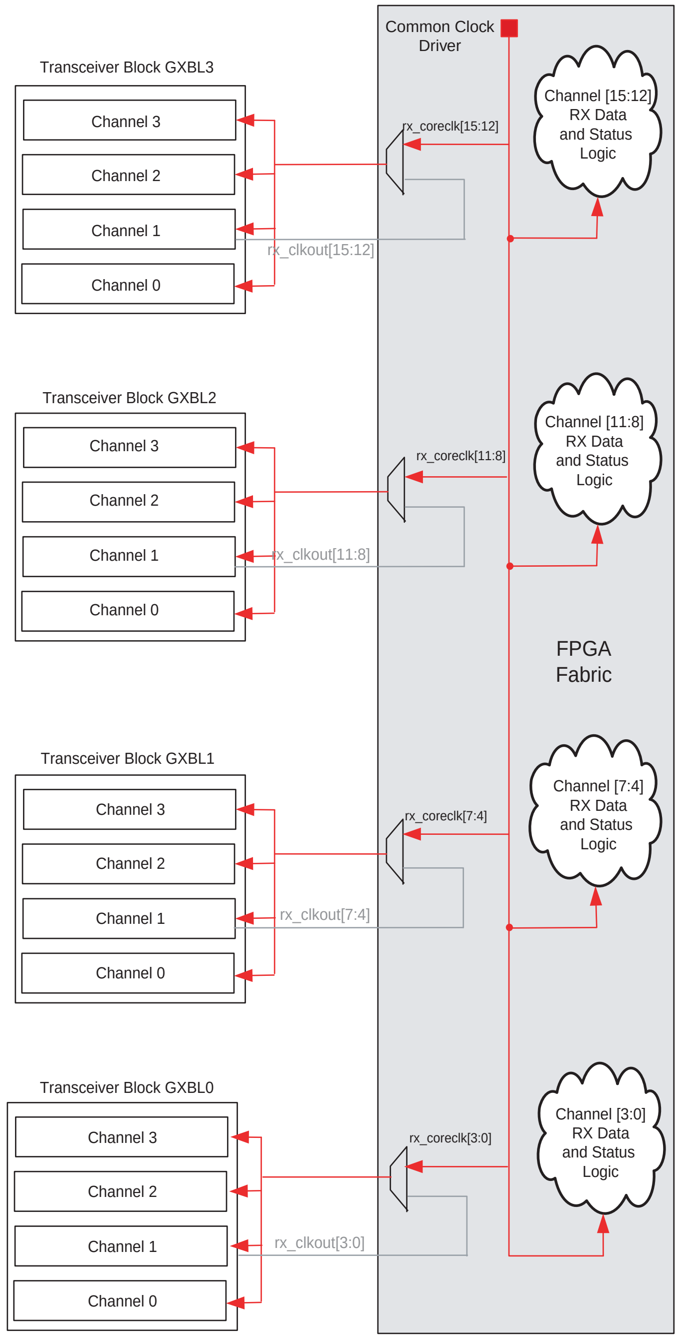
The ALTGX MegaWizard Plug-In Manager provides an optional port named `rx_coreclk` for each instantiated receiver channel. If you enable this port, the Quartus II software does not automatically select the receiver phase compensation FIFO read clock source. Instead, the signal that you drive on the `rx_coreclk` port of the channel clocks the read side of its receiver phase compensation FIFO.

You can use the flexibility of selecting the receiver phase compensation FIFO read clock to reduce clock resource utilization (global, regional, or both). You can connect the `rx_coreclk` ports of all receiver channels in your design and drive them using a common clock driver that has a 0 PPM frequency difference with respect to the FIFO write clocks of these channels. Use this common clock driver to latch the receiver data and status signals in the FPGA fabric for these channels. This FPGA fabric transceiver interface clocking scheme utilizes only one global or regional clock resource for all channels.

Example 9: Sixteen Identical Channels Across Four Transceiver Blocks

Figure 2–27 shows 16 channels located across four transceiver blocks. The incoming serial data to all 16 channels has a 0 PPM frequency difference with respect to each other. The `rx_coreclk` ports of all 16 channels are connected together and driven by a common clock driver. This common clock driver also latches the receiver data and status logic of all 16 receiver channels in the FPGA fabric. Only one clock resource (global, regional, or both) is used with this clocking scheme, compared to 16 clock resources (global, regional, or both) needed without the `rx_coreclk` ports (the Quartus II software-selected receiver phase compensation FIFO read clock).

Figure 2–27. Sixteen Identical Channels across Four Transceiver Blocks for Example 9



Common Clock Driver Selection Rules

The common clock driver driving the `rx_coreclk` ports of all channels must have a 0 PPM frequency difference with respect to the receiver phase compensation FIFO write clocks of these channels. If there is any frequency difference between the FIFO read clock (`rx_coreclk`) and the FIFO write clock, the FIFO overflows or under runs, resulting in corrupted data transfer between the FPGA fabric and the receiver.

Table 2–13 shows receiver phase compensation FIFO write clocks that the Quartus II software selects in various configurations.

Table 2–13. Receiver Phase Compensation FIFO Write Clocks

Configuration	Receiver Phase Compensation FIFO Write Clock	
	Without Byte De-Serializer	**With Byte De-Serializer**
Non-Bonded Channel Configuration with rate matcher	Low-speed parallel clock from the local clock divider in the associated channel (`tx_clkout`)	Divide-by-two version of the low-speed parallel clock from the local clock divider in the associated channel (`tx_clkout`)
Non-Bonded Channel Configuration without rate matcher	Parallel recovered clock from the receiver PMA in the associated channel (`rx_clkout`)	Divide-by-two version of the parallel recovered clock from the receiver PMA in the associated channel (`rx_clkout`)
×4 Bonded Channel Configuration	Low-speed parallel clock from the `CMU0` clock divider of the associated transceiver block (`coreclkout`)	Divide-by-two version of the low-speed parallel clock from the `CMU0` clock divider of the associated transceiver block (`coreclkout`)
×8 Bonded Channel Configuration	Low-speed parallel clock from the `CMU0` clock divider of the master transceiver block (`coreclkout` from the master transceiver block)	Divide-by-two version of the low-speed parallel clock from the `CMU0` clock divider of the master transceiver block (`coreclkout` from the master transceiver block)

To ensure that you understand the 0 PPM clock driver rule, the Quartus II software expects the following set of user assignments whenever the `rx_coreclk` port is used to drive the receiver phase compensation FIFO read clock:

- GXB 0 PPM Core Clock Setting

Failing to make this assignment correctly when using the `rx_coreclk` port results in a Quartus II compilation error.

GXB 0 PPM Core Clock Setting

The GXB 0 PPM core clock setting is intended for advanced users who know the clocking configuration of the entire system and want to reduce the FPGA fabric global and regional clock resource utilization. The GXB 0 PPM core clock setting allows the following clock drivers to drive the `rx_coreclk` ports:

- `tx_clkout` in non-bonded channel configurations with rate matcher
- `tx_clkout` and `rx_clkout` in non-bonded configurations without rate matcher
- `coreclkout` in bonded channel configurations
- FPGA CLK input pins
- Transceiver `REFCLK` pins
- Clock output from the left corner PLLs (PLL_1 and PLL_4)

The Quartus II software does not allow gated clocks or clocks generated in FPGA logic to drive the `tx_coreclk` ports.

Since the 0 PPM clock group assignment allows FPGA CLK input pins and transceiver REFCLK pins as clock drivers, the Quartus II compiler cannot determine if there is a 0 PPM difference between the FIFO write clock and read clock for each channel.

☞ You must ensure that the clock driver for all connected `rx_coreclk` ports has a 0 PPM difference with respect to the FIFO write clock in those channels.

Table 2–14 shows the Quartus II assignments that you must make for the clocking scheme shown in Figure 2–27.

Table 2–14. Quartus II Assignments

Assignment	**Description**
From:	Full design hierarchy name of one of the following clock drivers that you choose to drive the `rx_coreclk` ports of all identical channels *(1)*: ■ `tx_clkout` ■ `rx_clkout` ■ `coreclkout` ■ FPGA CLK input pins ■ Transceiver REFCLK pins ■ Clock output from left and right or top and bottom PLLs
To:	`rx_datain` pins of all channels whose `rx_coreclk` ports are connected together and driven by the 0 PPM clock driver.
Assignment Name:	GXB 0 PPM Core Clock Setting
Value:	ON

Note to Table 2–14:

(1) You can find the full hierarchy name of the 0 PPM clock driver using the **Node Finder** feature in the Quartus II Assignment Editor.

Example 10: Sixteen Channels Across Four Transceiver Blocks

Figure 2–28 shows 16 non-bonded channels without rate matcher located across four transceiver blocks. The incoming serial data to all 16 channels have a 0 PPM frequency difference with respect to each other. The `rx_coreclk` ports of all 16 channels are connected together and driven by `rx_clkout[9]` in transceiver block GXBL2. The `rx_clkout[9]` also clocks the receiver data and status signals of all 16 channels in the FPGA fabric. Only one global or regional clock resource is used by `rx_clkout[9]` with this clocking scheme.

Figure 2–28. Sixteen Channels across Four Transceiver Blocks for Example 10

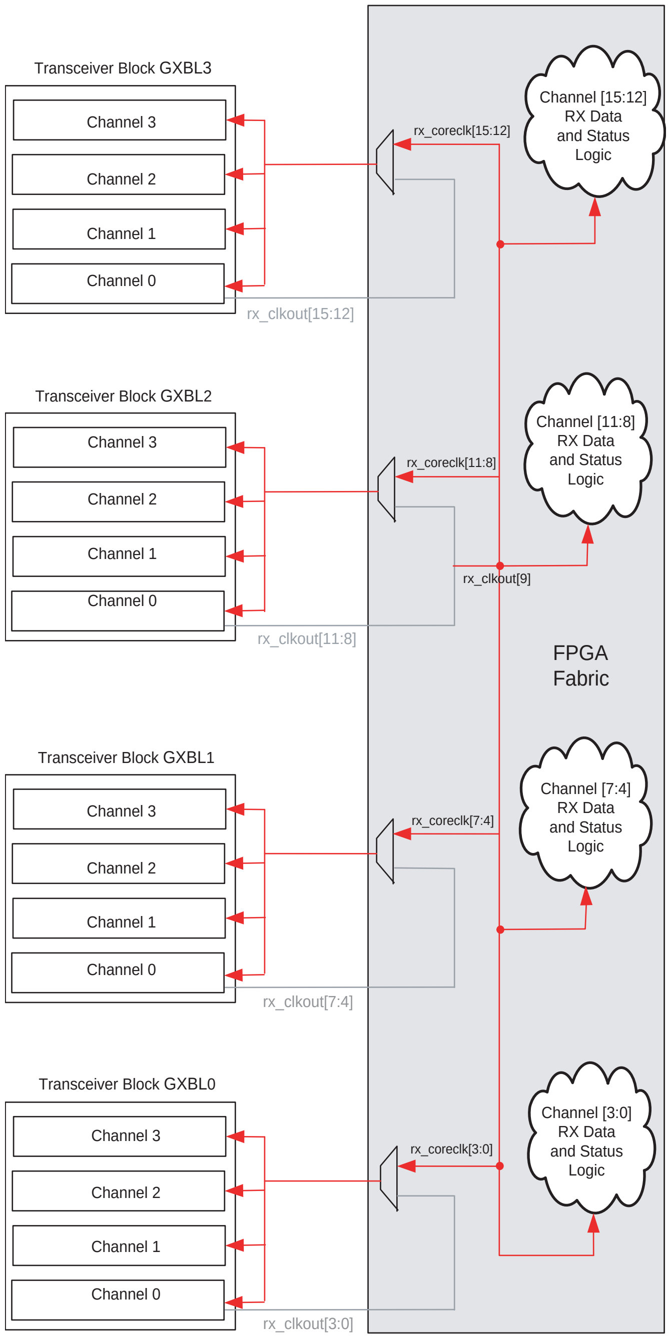


Table 2–15 shows the Quartus II assignments that you must make for the clocking scheme shown in Figure 2–28.

Table 2–15. Quartus II Assignments for Example 10

Assignment	Description
From:	`top_level/top_xcvr_instance1/altgx_component/rx_clkout[9]` *(1)*
To:	`rx_datain[15..0]`
Assignment Name:	GXB 0 PPM Core Clock Setting
Value:	ON

Note to Table 2–15:

(1) This is an example design hierarchy path for the `rx_clkout[9]` signal.

FPGA Fabric PLLs-Transceiver PLLs Cascading

The CMU PLL synthesizes the input reference clock to generate the high-speed serial clock used in the transmitter PMA. The receiver CDR synthesizes the input reference clock in lock-to-reference mode to generate the high-speed serial clock.

This high-speed serial clock output from the CMU PLL and receiver CDR runs at a frequency that is half the configured data rate. The CMU PLLs and receiver CDRs only support multiplication factors (M) of 2, 4, 5, 8, 10, 16, 20, and 25. If you use an on-board crystal oscillator to provide the input reference clock through the dedicated `REFCLK` pins or ITB lines, the allowed crystal frequencies are limited by the CMU PLL and receiver CDR multiplication factors. The input reference clock frequencies are also limited by the allowed phase frequency detector (PFD) frequency range between 50 MHz and 325 MHz.

Example 11: Channel Configuration for 3 Gbps Data Rate

Consider a channel configured for 3 Gbps data rate. The high-speed serial clock output from the CMU PLL and receiver CDR must run at 1.5 Gbps. Table 2–16 shows the allowed input reference clock frequencies for Example 11.

Table 2–16. Allowed Input Reference Clock Frequencies for Example 11 (Part 1 of 2)

Multiplication Factor (M)	On-Board Crystal Reference Clock Frequency (MHz)		Allowed
	with /N = 1	With /N = 2	
2	750	1500	No. Violates the PFD frequency limit of 325 MHz.
4	375	750	No. Violates the PFD frequency limit of 325 MHz.
5	300	600	Yes.
8	187.5	375	Yes.
10	150	300	Yes.
16	93.75	187.5	Yes.

Table 2–16. Allowed Input Reference Clock Frequencies for Example 11 (Part 2 of 2)

Multiplication Factor (M)	On-Board Crystal Reference Clock Frequency (MHz)		Allowed
	with /N = 1	With /N = 2	
20	75	150	Yes.
25	60	120	Yes.

For a 3-Gbps data rate, the Quartus II software only allows an input reference clock frequency of 60, 75, 93.75, 150, 187.5, 300, 375, and 750 MHz. To overcome this limitation, Arria II GX devices allow the synthesized clock output from left corner PLLs in the FPGA fabric to drive the CMU PLL and receiver CDR input reference clock. The additional clock multiplication factors available in the left corner PLLs allow more options for on-board crystal oscillator frequencies.

Dedicated Left PLL Cascade Lines Network

Arria II GX devices have a dedicated PLL cascade network on the left side of the device that connects to the input reference clock selection circuitry of the CMU PLLs and receiver CDRs. The dedicated PLL cascade network on the left side of the device connects to the input reference clock selection circuitry of the CMU PLLs and receiver CDRs in transceiver blocks located on the left side of the device.

The dedicated PLL cascade networks are segmented by bidirectional tri-state buffers located along the clock line. Segmentation of the dedicated PLL cascade network allows two left PLLs to drive the cascade clock line simultaneously to provide the input reference clock to the CMU PLLs and receiver CDRs in different transceiver blocks.

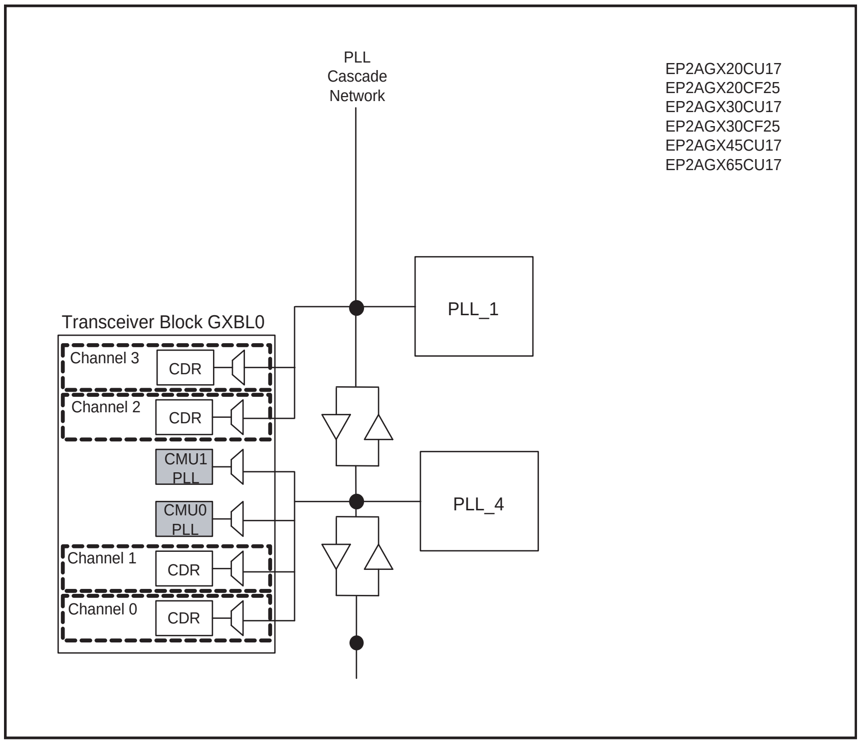
The following sections describe the dedicated PLL cascade networks available in the Arria II GX device family.

The following are the FPGA fabric PLLs-transceiver PLLs cascading option available for Arria II GX devices with four channels:

- EP2AGX20CU17
- EP2AGX20CF25
- EP2AGX30CU17
- EP2AGX30CF25
- EP2AGX45CU17
- EP2AGX65CU17

Figure 2–29 shows the FPGA fabric PLLs-transceiver PLLs cascading option allowed in the EP2AGX20CU17, EP2AGX20CF25, EP2AGX30CF25, EP2AGX45CU17, and EP2AGX65CU17 devices.

Figure 2–29. FPGA Fabric PLLs-Transceiver PLLs Cascading Option Allowed in the EP2AGX20CU17, EP2AGX20CF25, EP2AGX30CF25, EP2AGX45CU17, and EP2AGX65CU17 Devices



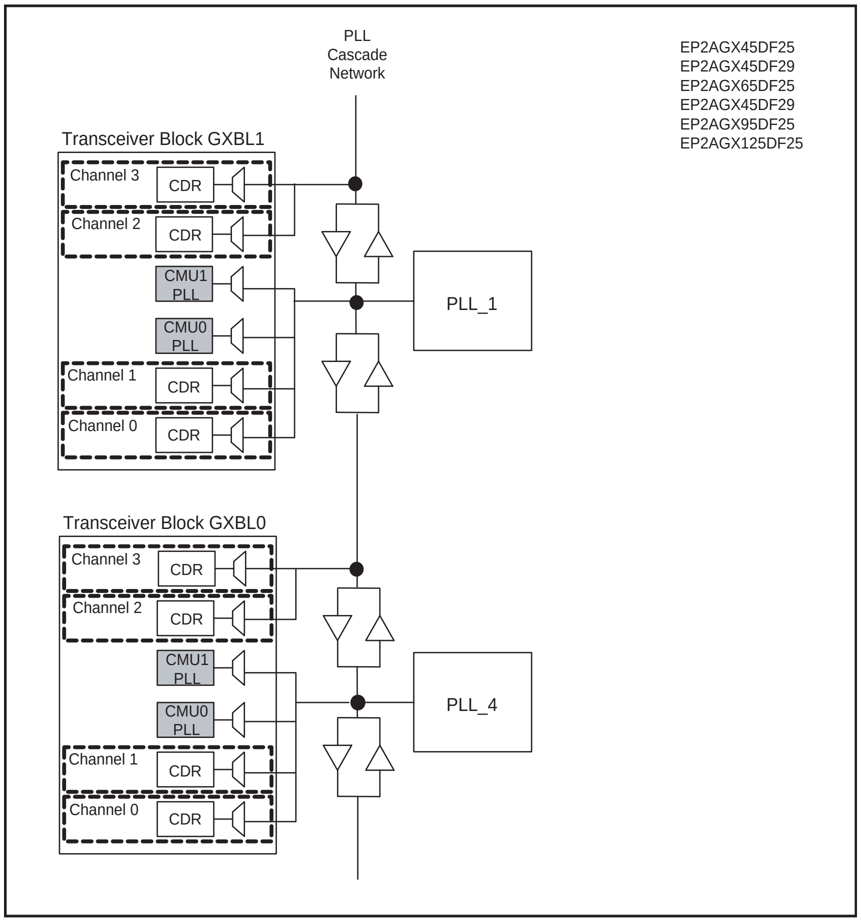
The following are the FPGA fabric PLLs-transceiver PLLs cascading option available for Arria II GX devices with eight channels:

- EP2AGX45DF25
- EP2AGX45DF29
- EP2AGX65DF25
- EP2AGX45DF29

- EP2AGX95DF25
- EP2AGX125DF25

Figure 2–30 shows the FPGA fabric PLLs-transceiver PLLs cascading option allowed in the EP2AGX45DF25, EP2AGX45DF29, EP2AGX65DF25, EP2AGX45DF29, EP2AGX95DF25, and EP2AGX125DF25 devices.

Figure 2–30. FPGA Fabric PLLs-Transceiver PLLs Cascading Option Allowed in the EP2AGX45DF25, EP2AGX45DF29, EP2AGX65DF25, EP2AGX45DF29, EP2AGX95DF25, and EP2AGX125DF25 Devices

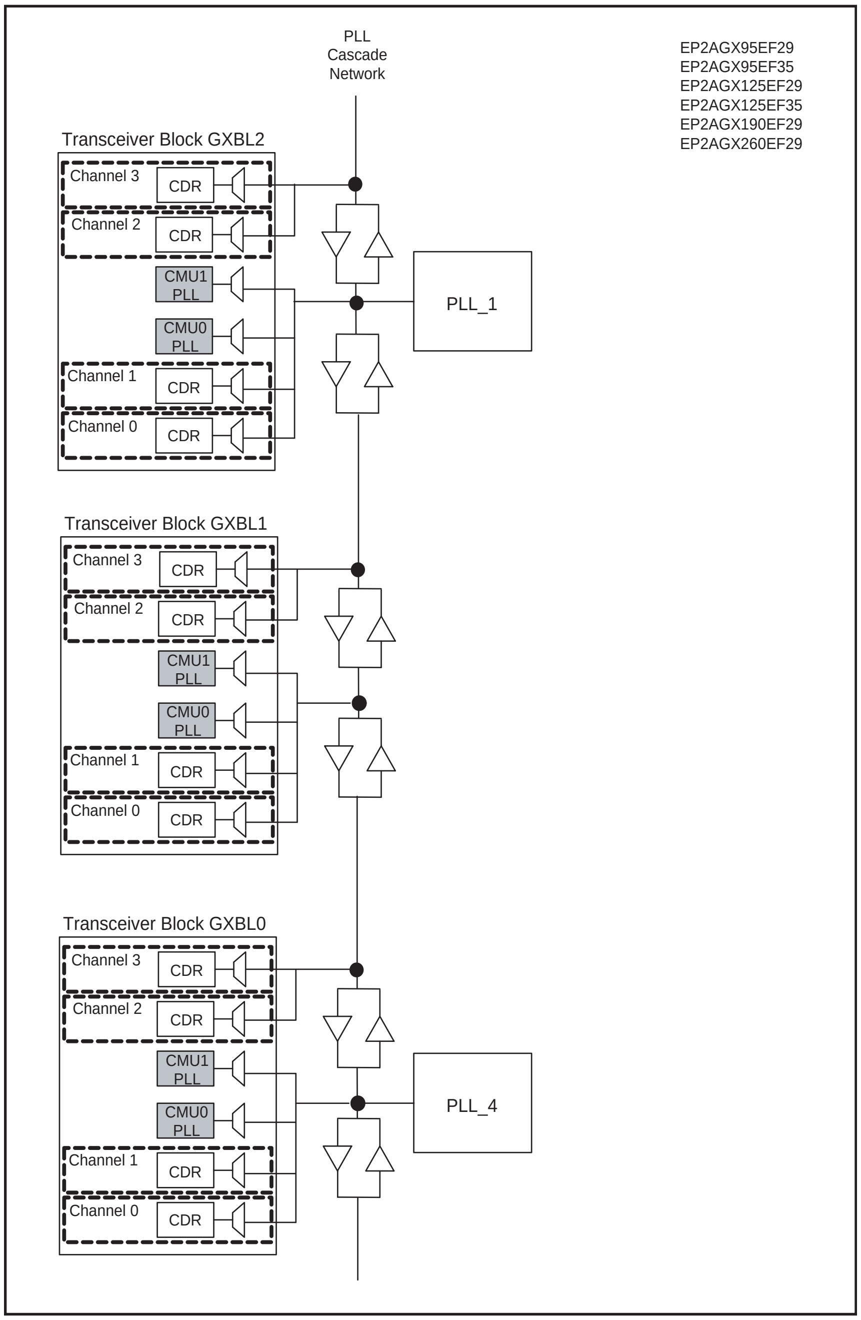


The following are the FPGA fabric PLLs-transceiver PLLs cascading option available for Arria II GX devices with twelve channels:

- EP2AGX95EF29
- EP2AGX95EF35
- EP2AGX125EF29
- EP2AGX125EF35
- EP2AGX190EF29
- EP2AGX260EF29

Figure 2–31 shows the FPGA fabric PLLs-transceiver PLLs cascading option allowed in the EP2AGX95EF29, EP2AGX95EF35, EP2AGX125EF29, EP2AGX125EF35, EP2AGX190EF29, and EP2AGX260EF29 devices.

Figure 2–31. FPGA Fabric PLLs Transceiver PLLs Cascading Option Allowed in the EP2AGX95EF29, EP2AGX95EF35, EP2AGX125EF29, EP2AGX125EF35, EP2AGX190EF29, and EP2AGX260EF29 Devices

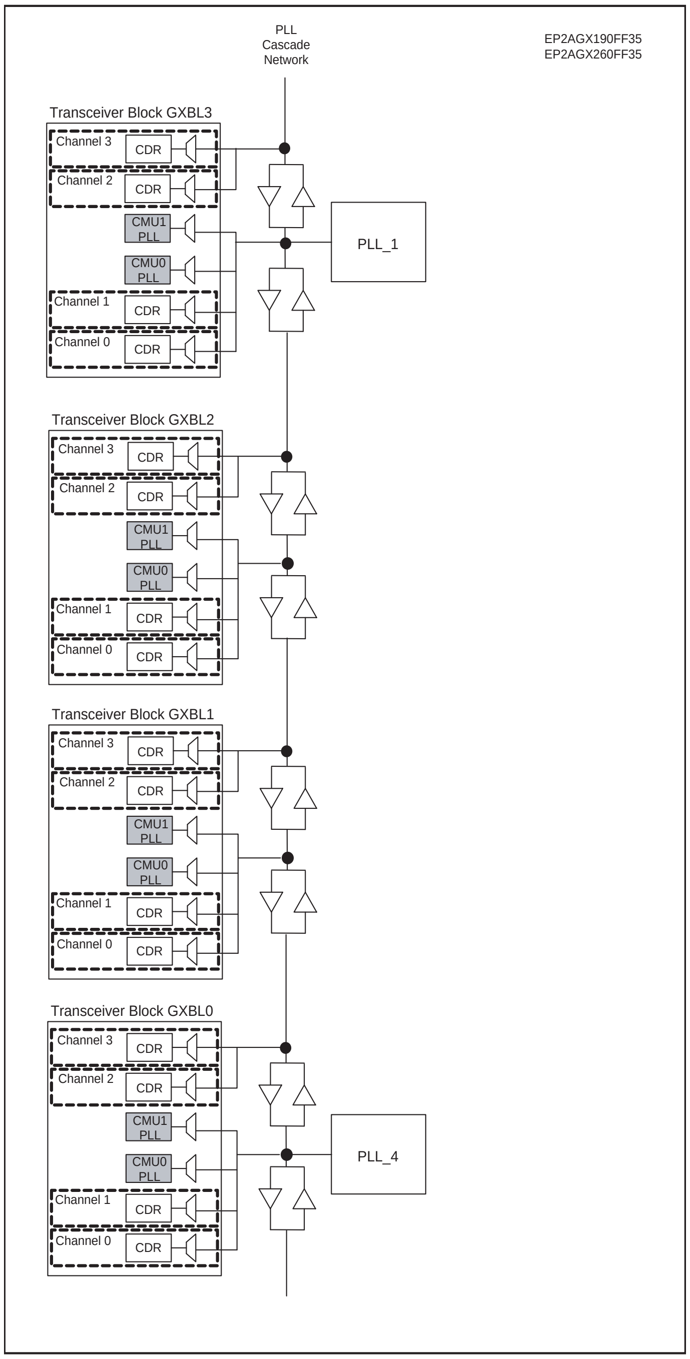


The following are the FPGA fabric PLLs-transceiver PLLs cascading option available for Arria II GX devices with sixteen channels:

- EP2AGX190FF35
- EP2AGX260FF35

Figure 2–32 shows the FPGA fabric PLLs-transceiver PLLs cascading option allowed in the EP2AGX190FF35 and EP2AGX260FF35 devices.

Figure 2–32. FPGA Fabric PLLs-Transceiver PLLs Cascading Option Allowed in the EP2AGX190FF35 and EP2AGX260FF35 Devices



FPGA Fabric PLLs-Transceiver PLLs Cascading Rules

PLL cascade networks are single clock lines segmented by bidirectional tri-state buffers located along the clock line. Segmentation of the PLL cascade network allows two left PLLs to drive the cascade clock line simultaneously to provide two input reference clocks to the CMU PLLs and receiver CDRs in different transceiver blocks. When cascading two or more FPGA fabric PLLs to the CMU PLLs and receiver CDRs, there must be no crossover in the cascaded clock paths on the PLL cascade network.

Example 12: Design Target-EP2AGX190FF35 Device

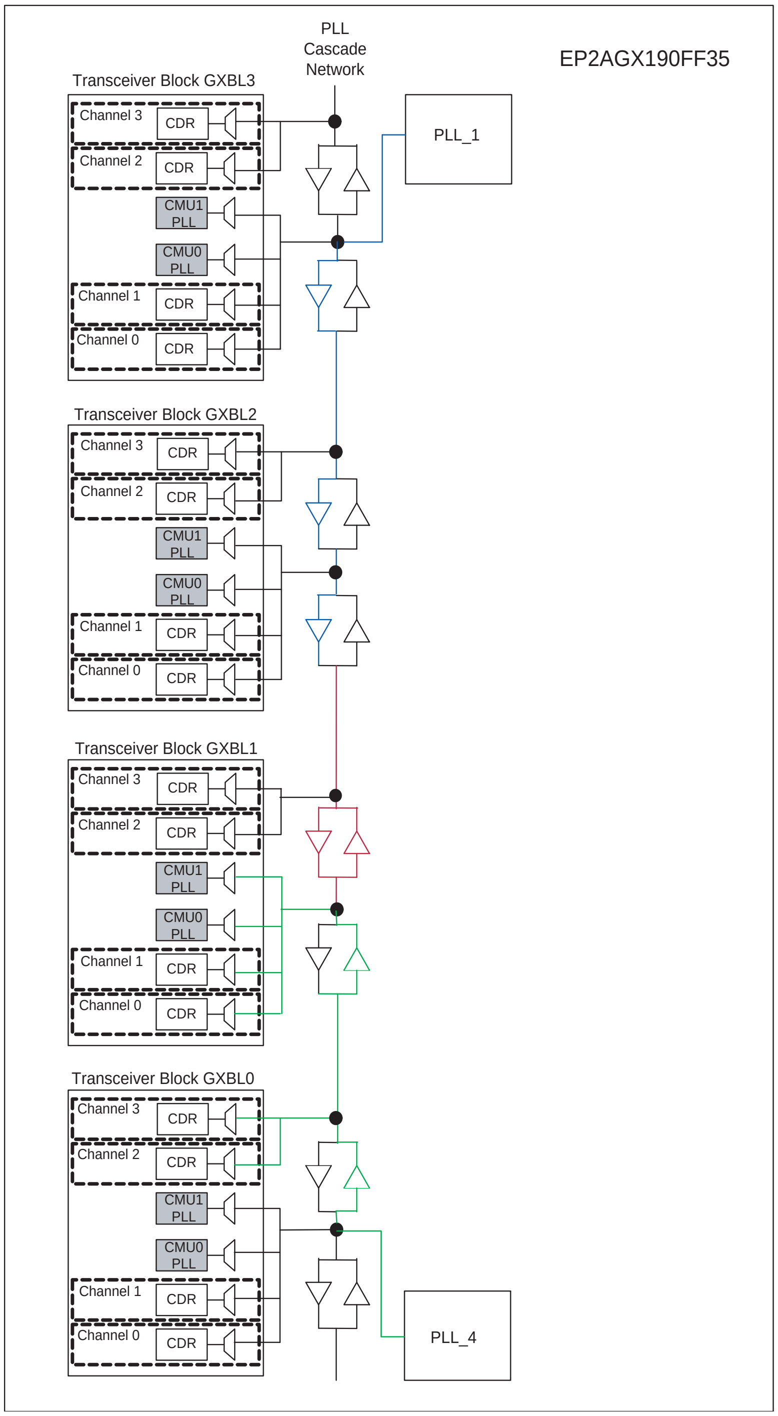
Consider a design targeting the EP2AGX190FF35 device and requiring input reference clocks to the following CMU PLLs and receiver CDRs from two left PLLs in the FPGA fabric:

- `CMU0 PLL` in transceiver block `GXBL1`
- Receiver CDRs in channel 2 and channel 3 in transceiver block `GXBL1`

Case 1: `PLL_4` is used to provide the input reference clock to the receiver CDRs in channel 2 and channel 3 (shown in green). `PLL_1` is used to provide the input reference clock to the `CMU0 PLL` (shown in blue) in transceiver block `GXBL1`.

Figure 2–33 shows that this FPGA fabric-transceiver PLL cascading configuration is illegal due to crossover (shown in red) of cascade clock paths on the PLL cascade network.

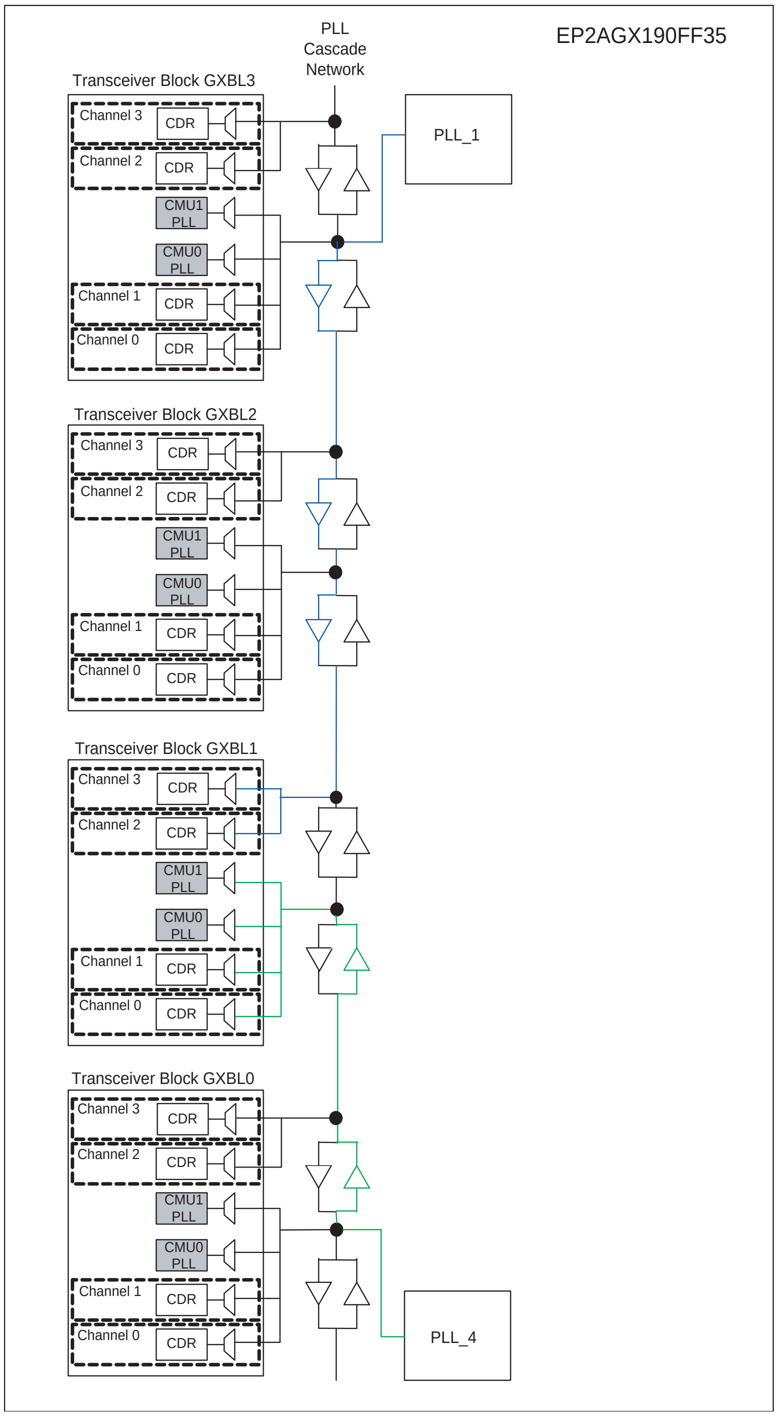
Figure 2–33. Illegal FPGA Fabric-Transceiver PLL Cascading Configuration



Case 2: `PLL_1` is used to provide the input reference clock to the receiver CDRs in channel 2 and channel 3 (shown in blue). `PLL_4` is used to provide the input reference clock to the `CMU0 PLL` (shown in green) in transceiver block `GXBL1`.

Figure 2–34 shows that this FPGA fabric-transceiver PLL cascading configuration is legal because there is no crossover of the cascade clock paths on the PLL cascade network.

Figure 2–34. Legal FPGA Fabric-Transceiver PLL Cascading Configuration



Document Revision History

Table 2–17 shows the revision history for this chapter.

Table 2–17. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial release.	—

3. Configuring Multiple Protocols and Data Rates

AIIGX52003-1.0

Introduction

This chapter describes the multiple protocols and data rates for Arria® II GX devices. Each transceiver channel in an Arria II GX device can run at an independent data rate or protocol mode. Within each transceiver channel, the transmitter and receiver channel can run at different data rates. Each transceiver block consists of two clock multiplier unit (CMU) phase-locked loops (PLLs) that provide clocks to all the transmitter channels within the transceiver block. Each receiver channel contains a dedicated clock data recovery (CDR).

This chapter includes the following sections:

Transceiver PLL Configurations

You can configure each transmitter channel to use one of the two CMU PLLs in the transceiver block. In addition, each transmitter channel has a local divider (/1, /2, or /4) that divides the high-clock output of the CMU PLL to provide high-speed serial and low-speed parallel clocks for its physical coding sublayer (PCS) and physical medium attachment (PMA) functional blocks.

You can configure the Rx CDR present in the receiver channel to a distinct data rate and provide separate input reference clocks. Each receiver channel also contains a local divider that divides the high-speed clock output of the Rx CDR and provides clocks for its PCS and PMA functional blocks. To enable transceiver channel settings, the Quartus® II software provides the ALTGX MegaWizard™ Plug-In Manager interface. The ALTGX MegaWizard Plug-In Manager allows you to instantiate a single transceiver channel or multiple transceiver channels in **Receiver and Transmitter**, **Receiver Only**, and **Transmitter Only** configurations.

Creating Transceiver Channel Instances

You can instantiate multiple transceiver channels in the **General** screen of the ALTGX MegaWizard Plug-In Manager in the following ways:

- For the **What is the number of channels?** option, select the required value. This method creates the transceiver channels with identical configurations. For examples, refer to "Combining Transceiver Instances in Multiple Transceiver Blocks" on page 3–13.
- For the **What is the number of channels?** option, select **1** and create a single channel transceiver instance. To instantiate additional transceiver channels with an identical configuration, stamp the created ALTGX instance multiple times. If you need additional transceiver channels with different configurations, create separate ALTGX megafunction instances with different settings and use them in your design.

When you create instances using the above methods, you can force the placement of up to four transceiver channels within the same transceiver block. This is done by assigning the `tx_dataout` and `rx_datain` ports of the channel instances to a single transceiver bank. If you do not assign pins to the `tx_dataout` and `rx_datain` ports, the Quartus II software chooses default pin assignments. When you compile the design, the Quartus II software combines multiple channel instances within the same transceiver block if the instances meet specific requirements. The following sections explain these requirements for different transceiver configurations.

General Requirements to Combine Channels

When you create multiple ALTGX instances, the Quartus II software requires that you set identical values on the following parameters and signals to combine the ALTGX instances within the same transceiver block or in the transceiver blocks on the same side of the device. The following sections describe these requirements.

Control Signals

The `gxb_powerdown` port is an optional port that you can enable in the ALTGX MegaWizard Plug-In Manager. If enabled, you must drive the `gxb_powerdown` port in the ALTGX instances from the same logic or the same input pin to enable the Quartus II software to assign them in the same transceiver block. If the `gxb_powerdown` port is disabled, the Quartus II software ties the port to ground.

Calibration Clock and Power Down

Each calibration block in an Arria II GX device is shared by multiple transceiver blocks.

If your design uses multiple transceiver blocks, depending on the transceiver banks selected, you must connect the `cal_blk_clk` and `cal_blk_powerdown` ports of all channel instances to the same input pin or logic.

For more information about the calibration block and transceiver banks that are connected to a specific calibration block, refer to the "Calibration Block" section in the *Arria II GX Transceiver Architecture* chapter in volume 2 of the *Arria II GX Device Handbook*.

☞ Asserting the `cal_blk_powerdown` port affects the calibration circuit on all transceiver channels connected to the calibration block.

Sharing CMU PLLs

☞ Each Arria II GX transceiver block contains two CMU PLLs. When you create multiple transceiver channel instances and intend to combine them in the same transceiver block, the Quartus II software checks whether a single CMU PLL can be used to provide clock outputs for the transmitter side of the channel instances. If a single CMU PLL is not sufficient, the Quartus II software attempts to combine the channel instances using two CMU PLLs. Otherwise, the Quartus II software issues a Fitter error.

The following two sections describe the ALTGX instance requirements to enable the Quartus II software to share the CMU PLL.

☞ Only channels combined within the same transceiver block can share the two CMU PLLs available in a transceiver block.

Multiple Channels Sharing a CMU PLL

To enable the Quartus II software to share the same CMU PLL for multiple channels, the following parameters in the channel instantiations must be identical:

- Base data rate (the CMU PLL is configured for this data rate)
- CMU PLL bandwidth setting
- Reference clock frequency
- Input reference clock pin
- `pll_powerdown` port of the ALTGX instances must be driven from the same logic

Each channel instance can have a different local divider setting. This is a useful option when you intend to run each channel within the transceiver block at different data rates that are derived from the same base data rate using the local divider values /1, /2, and /4. This is shown in Example 1.

Example 1

Consider an example design with four instances in a **Receiver and Transmitter** configuration in the same transceiver block at the following serial data rates. Assume that each instance contains a channel, is driven from the same clock source, and has the same CMU PLL bandwidth settings.

Table 3–1 shows the configuration for Example 1.

Table 3–1. Configuration for Example 1

User-Created Instance Name	ALTGX MegaWizard Plug-In Manager Settings		
	Number of Channels	**Configuration**	**Effective Data Rate**
inst0	1	Receiver and Transmitter	3.75 Gbps
inst1	1	Receiver and Transmitter	0.9375 Gbps
inst2	1	Receiver and Transmitter	1.875 Gbps
inst3	1	Receiver and Transmitter	3.75 Gbps

For Example 1, you can share a single CMU PLL for all four channels:

- One CMU PLL can be configured to run at 3.75 Gbps.
- Each channel can divide the CMU PLL clock output using the local divider and achieve the required data rates of 3.75 Gbps, 1.875 Gbps, and 0.9375 Gbps. Because each receiver channel has a dedicated CDR, the receiver side in each instance can be set up for these three data rates without restrictions.

The following steps show you how to achieve the configuration.

To enable the Quartus II software to share a single CMU PLL for all four channels, set the following values in the **General** screen of the ALTGX MegaWizard Plug-In Manager.

- For inst0:
  - Set **What is the effective data rate?** to 3.75 Gbps
  - Set **Specify base data rate** to 3.75 Gbps
- For inst1:
  - Set **What is the effective data rate?** to 1.875 Gbps
  - Set **Specify base data rate** to 3.75 Gbps
- For inst2:
  - Set **What is the effective data rate?** to 0.9375 Gbps
  - Set **Specify base data rate** to 3.75 Gbps
- For inst3:
  - Set **What is the effective data rate?** to 3.75 Gbps
  - Set **Specify base data rate** to 3.75 Gbps

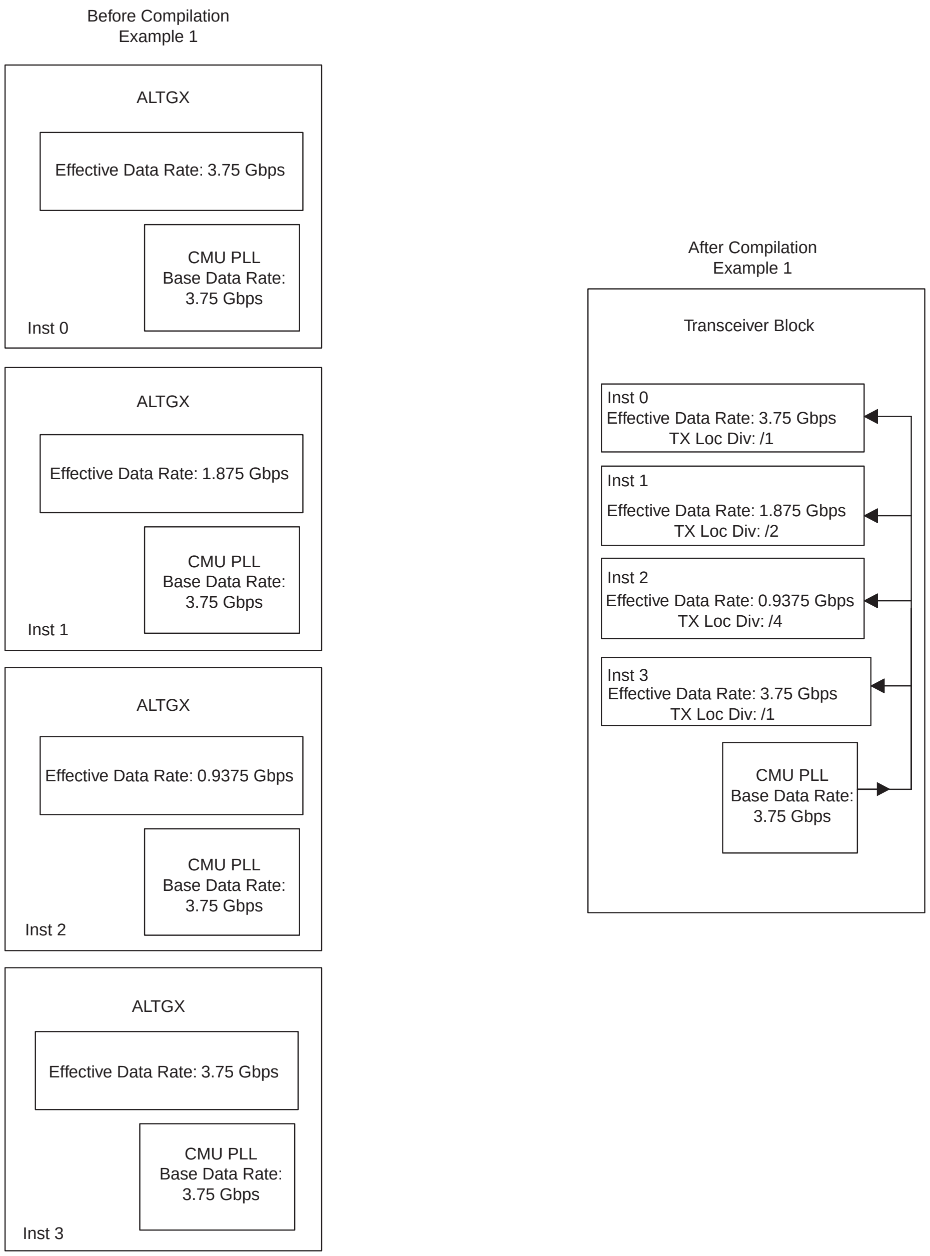
☞ The **Specify base data rate** option is 3.75 Gbps for all four instances. Because the CMU PLL bandwidth setting and input reference clock are the same and the `pll_powerdown` ports are driven from the same logic or pin, the Quartus II software shares a single CMU PLL that runs at 3.75 Gbps.

You can force the placement of transceiver channels to a specific transceiver block by assigning pins to `tx_dataout` and `rx_datain`. Otherwise, the Quartus II software selects a transceiver block.

Figure 3–1 shows the scenario before and after the Quartus II software combines the transceiver channel instances. Because the Rx CDR is not shared between channels, only the CMU PLL is shown.

Each ALTGX instance has a `pll_powerdown` port. You must drive the `pll_powerdown` ports of all instances from the same logic to allow the Quartus II software to share the same CMU PLL. If you drive the `pll_powerdown` ports of the ALTGX instance using different logic, the Quartus II software does not use the same CMU PLL even if all the other required parameters of all the ALTGX instances are identical.

Figure 3–1. ALTGX Instances before and after Compilation for Example 1



Multiple Channels Sharing Two CMU PLLs

In some cases, a single CMU PLL is not sufficient to run the transmitter channels within a transceiver block at the desired data rates.

Using a second CMU PLL is useful if you want to combine channels that require different configurations, such as:

- Quartus II software-defined protocols (for example, Basic, Gigabit Ethernet [GIGE], SONET/synchronous digital hierarchy [SDH], Serial Digital Interface [SDI], or PCI Express [PIPE] modes)
- CMU PLL bandwidth settings
- Different input reference clocks

Example 2

Consider a design that requires four channels set up in a **Receiver and Transmitter** configuration in the same transceiver block at the serial data rates shown in Table 3–2.

Table 3–2. Configuration for Example 2

User-Created Instance Name	ALTGX MegaWizard Plug-In Manager Settings		
	Number of Channels	**Configuration**	**Effective Data Rate**
inst0	1	Receiver and Transmitter	3.75 Gbps
inst1	1	Receiver and Transmitter	1.875 Gbps
inst2	1	Receiver and Transmitter	0.9375 Gbps
inst3	1	Receiver and Transmitter	2 Gbps

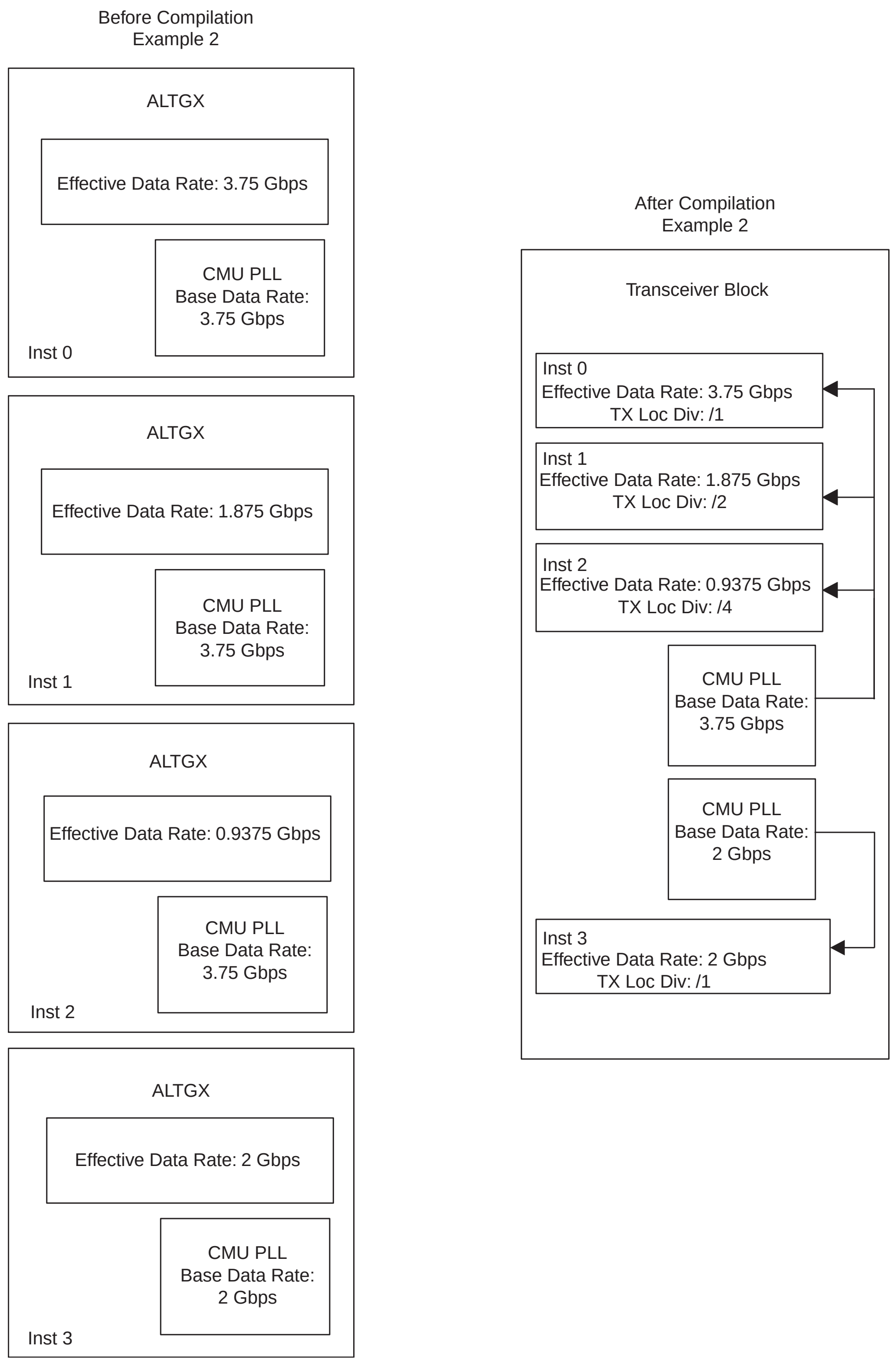
Assume that instance 0, 1, and 2 are driven from the same clock source and have the same CMU PLL bandwidth settings. In this case, you can use one CMU PLL for instance 0, 1, and 2. Refer to "Example 1" on page 3–3 for the ALTGX MegaWizard Plug-In Manager settings that enable the Quartus II software to share the same CMU PLL. A second CMU PLL is required for instance 3.

You can force the placement of transceiver channels to a specific transceiver block by assigning pins to the `tx_dataout` and `rx_datain` pins of the four ALTGX instances. Otherwise, the Quartus II software selects a transceiver block.

Figure 3–2 shows the transceiver configuration before and after the Quartus II software combines the transceiver channels within the same transceiver block. Because the Rx CDR is not shared between channels, only the CMU PLLs are shown.

☞ You must connect the `pll_powerdown` port of instance 0, 1, and 2 to the same logic output to share the same CMU PLL for these instances.

Figure 3–2. ALTGX Transceiver Channel Instances before and after Compilation for Example 2



In cases where you have two instances with the same serial data rate but with different CMU PLL data rates, the Quartus II software creates a separate CMU PLL for the two instances. For example, consider the configuration shown in Table 3–3.

Table 3–3. Sample Configuration Where Instances Cannot Be Combined in a Single Transceiver Block

User-Created Instance Name	ALTGX MegaWizard Plug-In Manager Settings			
	Number of Channels	**Configuration**	**Effective Data Rate**	**Base Data Rate**
inst0	1	Receiver and Transmitter	1.5 Gbps	1.5 Gbps
inst1	1	Receiver and Transmitter	1.5 Gbps	3.0 Gbps
inst2	1	Receiver and Transmitter	1 Gbps	1 Gbps

Even though the effective data rate of inst1 is 1.5 Gbps (3 Gbps/2 = 1.5 Gbps), the same as inst0, when you compile the design, the Quartus II software requires two CMU PLLs to provide clocks for the transmitter side of the two instances because their base data rates are different. In this example, you have the third instance, inst2, that requires a third CMU PLL. Therefore, the Quartus II software cannot combine the above three instances within the same transceiver block.

Combining Receiver Only Channels

You can selectively use the receiver in the transceiver channel by selecting the **Receiver Only** configuration in the **What is the Operating Mode?** option on the **General** screen of the ALTGX MegaWizard Plug-In Manager.

You can combine **Receiver Only** channel instances of different configurations and data rates into the same transceiver block. Since each receiver channel contains its own dedicated CDR, each **Receiver Only** instance (assuming one receiver channel per instance) can have different data rates.

For the Quartus II software to combine the **Receiver Only** instances within the same transceiver block, you must connect `gxb_powerdown` (if used) of all the channel instances from the same logic or input pin. For more information, refer to "General Requirements to Combine Channels" on page 3–2.

It is possible to have up to four receiver channels that can run at different data rates by using separate input reference clocks if there are enough clock routing resources available.

If you instantiate the **Receiver Only** configuration, the ALTGX MegaWizard Plug-In Manager does not allow you to enable the rate matching FIFO (clock rate compensation FIFO) in the receiver channel PCS because `tx_clkout` is not available in a **Receiver Only** instance to clock the read side of the rate matching FIFO. If you need to perform clock rate compensation, implement the rate matching FIFO in the FPGA fabric.

If you create a **Receiver Only** instance and do not use the transmitter channel that is present in the same physical transceiver channel, the Quartus II software automatically powers down the unused transmitter channel.

Combining Transmitter Channel and Receiver Channel Instances

You can create a separate transmitter channel instance and a separate receiver channel instance and assign the `tx_dataout` and `rx_datain` pins of the transmitter and receiver instance, respectively, in the same physical transceiver channel. This configuration is useful in cases where you intend to run the transmitter and receiver channel at different serial data rates. To create a transmitter channel instance and a receiver channel instance, select the **Transmitter Only** and **Receiver Only** options in the operating mode (**General** screen) of the ALTGX MegaWizard Plug-In Manager.

Multiple Transmitter Channel and Receiver Channel Instances

The Quartus II software allows you to combine multiple **Transmitter Only** channel and **Receiver Only** channel instances within the same transceiver block. Based on the pin assignments, the Quartus II software combines the corresponding **Transmitter Only** and **Receiver Only** channels in the same physical channel. To enable the Quartus II software to combine the transmitter channel and receiver channel instances in the same transceiver block, follow the rules and requirements outlined in the following sections:

- "General Requirements to Combine Channels" on page 3–2
- "Multiple Channels Sharing a CMU PLL" on page 3–3
- "Multiple Channels Sharing Two CMU PLLs" on page 3–6
- "Combining Receiver Only Channels" on page 3–8

Example 3

Consider that you create four ALTGX instances, as shown in Table 3–4.

Table 3–4. Four ALTGX Instances for Example 3

Instance Name	Configuration	Effective Data Rate	Input Reference Clock Frequency
inst0	Transmitter only	3.125 Gbps	156.25 MHz
inst1	Receiver only	2.5 Gbps	156.25 MHz
inst2	Transmitter only	1.25 Gbps	125 MHz
inst3	Receiver only	2 Gbps	125 MHz

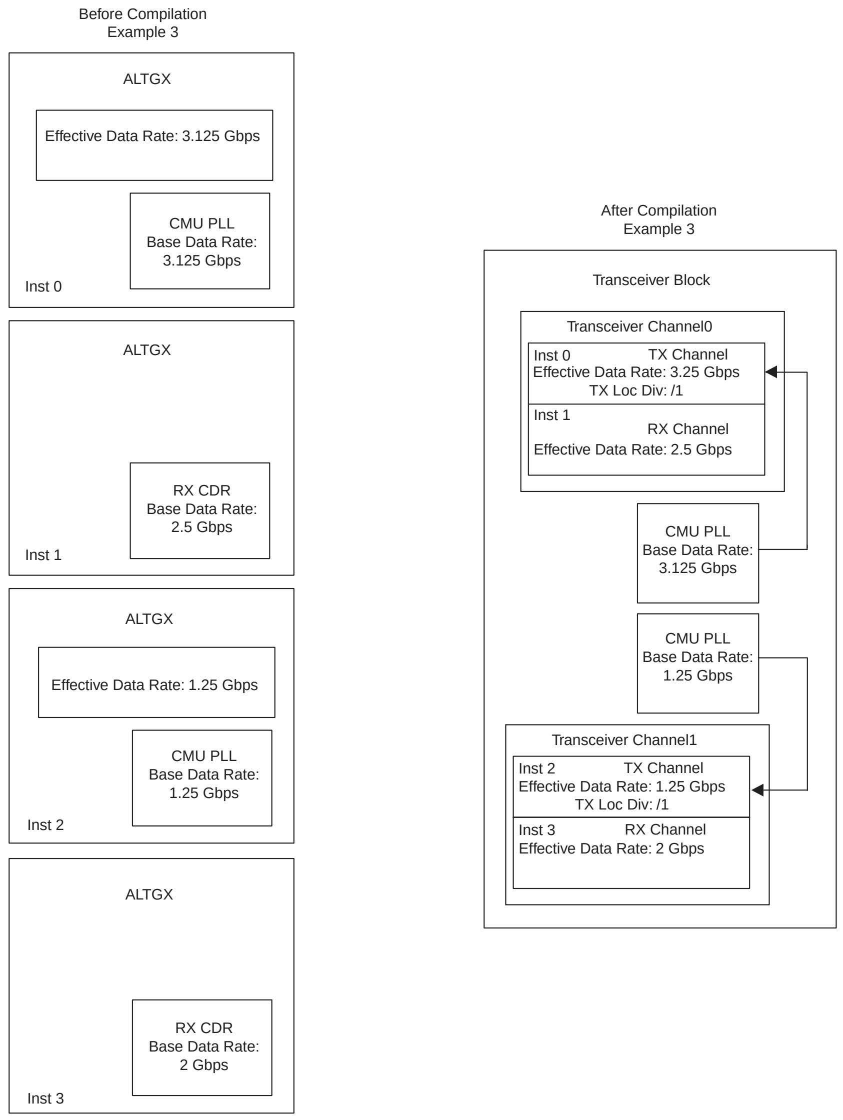
After you create the above instances, if you force the placement of the instances shown in Table 3–5, the Quartus II software combines inst0 and inst1 into physical channel 0 and inst2 and inst3 into physical channel 1.

Table 3–5. Forced Placement of the Instances for Example 3

Instance Name	Physical Channel Pin Assignments in the Same Transceiver Block
inst0	Tx pin of channel 0
inst1	Rx pin of channel 0
inst2	Tx pin of channel 1
inst3	Rx pin of channel 1

Figure 3–3 shows the transceiver channel instances before and after compilation.

Figure 3–3. ALTGX Transceiver Channel Instances before and after Compilation for Example 3



Combining Channels Configured in Protocol Functional Modes

The following sections describe combining channels that are configured in protocol functional modes.

Basic x4 Mode

The ALTGX MegaWizard Plug-In Manager provides a Basic mode with a ×4 option in the **Which sub-protocol will you be using?** option on the **General** screen. If you select this option, all the transmitter channels within the transceiver block receive the high-speed serial and low-speed parallel clock from the CMU0 clock divider block (present in the CMU0 channel). Each receiver channel within the transceiver block is clocked independently by the recovered clock from its receiver CDR.

When you use this mode, the ALTGX MegaWizard Plug-In Manager allows you to select one or more channels from the **What is the number of channels?** option on the **General** screen.

If you select the number of channels to be less than four, the remaining transmitter channels within the transceiver block cannot be used. Therefore, if you have more than one instance configured in **Transmit Only** or **Receiver and Transmitter** mode with the ×4 option enabled, the Quartus II software cannot combine the instances within the same transceiver block.

Only the transmitter channels share a common clock. The receiver channels are clocked independently. Therefore, you can configure the unused receiver channels within a transceiver block in any allowed configuration. For example, assume that you configure the ALTGX MegaWizard Plug-In Manager with the options shown in Table 3–6.

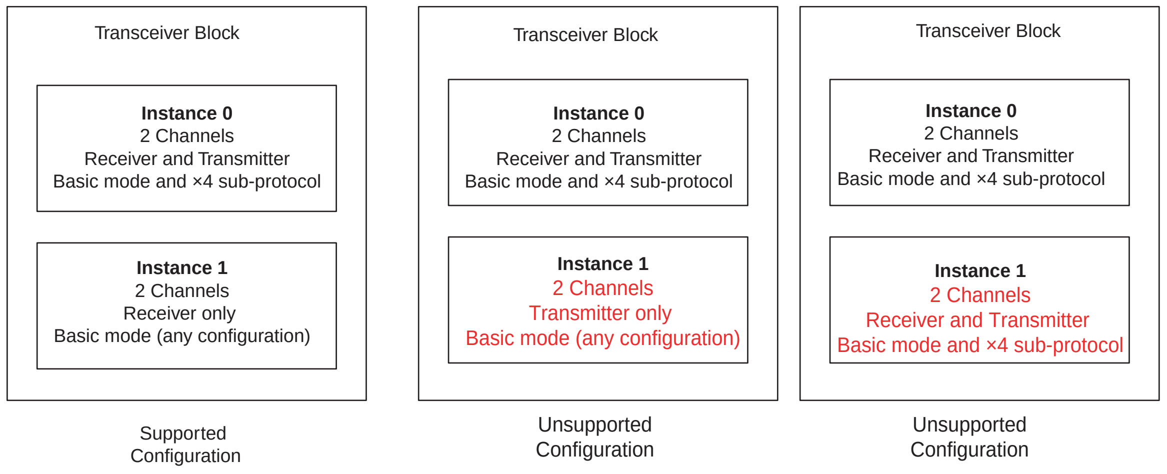
Table 3–6. General Screen Options in Basic ×4 Mode

Option	Value
What protocol will you be using?	Basic mode
Which subprotocol will you be using?	×4
What is the operating mode?	Receiver and Transmitter
What is the number of channels?	2

If you create the instance with the above selection, you cannot use the remaining two transmitter channels in the transceiver block. However, you can use the remaining two receiver channels in a different configuration.

Figure 3–4 shows examples of supported and unsupported configurations.

Figure 3–4. Examples of Supported and Unsupported Configurations to Combine Instances in Basic ×4 Mode



Combining Channels Using the PCI Express hard IP Block with Other Channels

The Arria II GX device contains an embedded PCI Express hard IP block that performs the phyMAC, datalink, and transaction layer functionality specified by PCI Express base specification 2.0. Each PCI Express hard IP block is shared by two transceiver blocks. The PCI Express compiler wizard provides you the options to configure the PCI Express hard IP block. When enabled, the transceiver channels associated with this block are enabled.

There are restrictions on combining transceiver channels with different functional and/or protocol modes (for example, Basic mode) within two contiguous transceiver blocks with the channels that use PCI Express hard IP block. The restrictions depend on the number of channels used (×1 or ×4) and the number of virtual channels (VC) selected in the PCI Express compiler MegaWizard Plug-In Manager. Table 3–7 shows the restrictions.

Table 3–7. PCI Express hard IP Block Restrictions When Combining Transceiver Channels with Different Functional and/or Protocol Modes *(Note 1)*, *(2)*

PCI Express Configuration (PCI Express hard IP Options Enabled in the PCIe Compiler Wizard)			Transceiver Block 0 *(3)*				Transceiver Block 1 *(4)*			
Link Width	Lane (Data Interface Width)	Virtual Channel (VC)	Ch0 *(5)*	Ch1	Ch2	Ch3	CH4	Ch5	Ch6	Ch7
1	64 bit	1	PCIe ×1	Avail.	Avail.	Avail.	Avail.	Avail.	Avail.	Avail.
4	64 bit	1	PCIe ×4	N/A	N/A	N/A	Avail.	Avail.	Avail.	Avail.
8	128 bit	1	PCIe ×8	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Notes to Table 3–7:

(1) Avail.—the channels can be used in other configurations.

(2) N/A—the channels are NOT available for use.

(3) Transceiver block 0—the master transceiver block that provides high-speed serial and low-speed parallel clocks in a PCI Express (PIPE) ×4 or ×8 configuration.

(4) Transceiver block 1—the adjacent transceiver block that shares the same PCI Express hard IP block with transceiver block 0.

(5) The physical channel 0 in the transceiver block. For more information about physical-to-logical channel mapping in PCI Express (PIPE) functional mode, refer to the “×8 Channel Configuration” section in the *Arria II GX Transceiver Clocking* chapter in volume 2 of the *Arria II GX Device Handbook*.

For more information about the PCI Express Compiler MegaCore Functions and hard IP implementation, refer to the *PCI Express Compiler User Guide*.

Combining Transceiver Instances Using PLL Cascade Clocks

The Arria II GX device provides multiple input reference clock sources to clock the CMU PLLs and Rx CDRs in each transceiver block. The following are the input reference clock sources that can clock the CMU PLLs and Rx CDRs:

- `refclks` from the same transceiver block
- Global clock lines
- `refclks` from transceiver blocks on the same side of the device using the inter-transceiver block (ITB) lines
- PLL cascade clock (this is the cascaded clock output from the PLLs in the FPGA fabric)

If you use the PLL cascade clock to provide input reference clocks to the CMU PLLs or Rx CDRs, there are requirements for combining transceiver channels (as described in the following sections).

The Arria II GX transceiver has the ability to cascade the output of the general purpose PLLs to the CMU PLLs and receiver CDRs. The left side of the Arria II GX device contains a PLL cascade clock network—a single line network that connects the PLL cascade clock to the transceiver block. This clock line is segmented to allow different PLL cascade clocks to drive the transceiver CMU PLLs and Rx CDRs.

The segmentation locations differ based on the device family. Therefore, there are restrictions when you want to combine transceiver channels that use different PLL cascade clocks as input reference clocks.

For more information about using the PLL cascade clock and segmentation, refer to the "PLL Cascading" section in the *Arria II GX Transceiver Clocking* chapter in volume 2 of the *Arria II GX Device Handbook*.

Combining Transceiver Instances in Multiple Transceiver Blocks

"Creating Transceiver Channel Instances" on page 3–2 describes the method to instantiate multiple transceiver channels using a single ALTGX instance. The following section describes the method to instantiate multiple transceiver channels using multiple transceiver blocks.

When you create a transceiver instance that has more than four transceiver channels, the Quartus II software attempts to combine the transceiver channels in multiple transceiver blocks. This is shown in Example 4.

Example 4

Consider that you create two ALTGX instances with the configuration shown in Table 3–8.

Table 3–8. Two ALTGX Instances for Example 4

Instance Name	Number of Transceiver Channels	Configuration	Effective Data Rate	Input Reference Clock
inst0	7	Receiver and Transmitter	3.125 Gbps	156.25 Gbps
inst1	1	Receiver and Transmitter	3.125 Gbps	156.25 Gbps

In this case, assuming that all the required parameters specified in "Multiple Channels Sharing a CMU PLL" on page 3–3 are identical in inst0 and inst1, the Quartus II software fits inst0 and inst1 in two transceiver blocks.

Figure 3–5 shows the transceiver instances before compilation for Example 4.

Figure 3–5. Transceiver Channel Instances before Compilation for Example 4

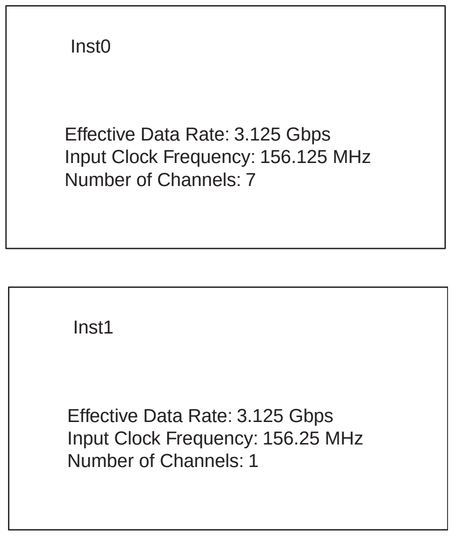
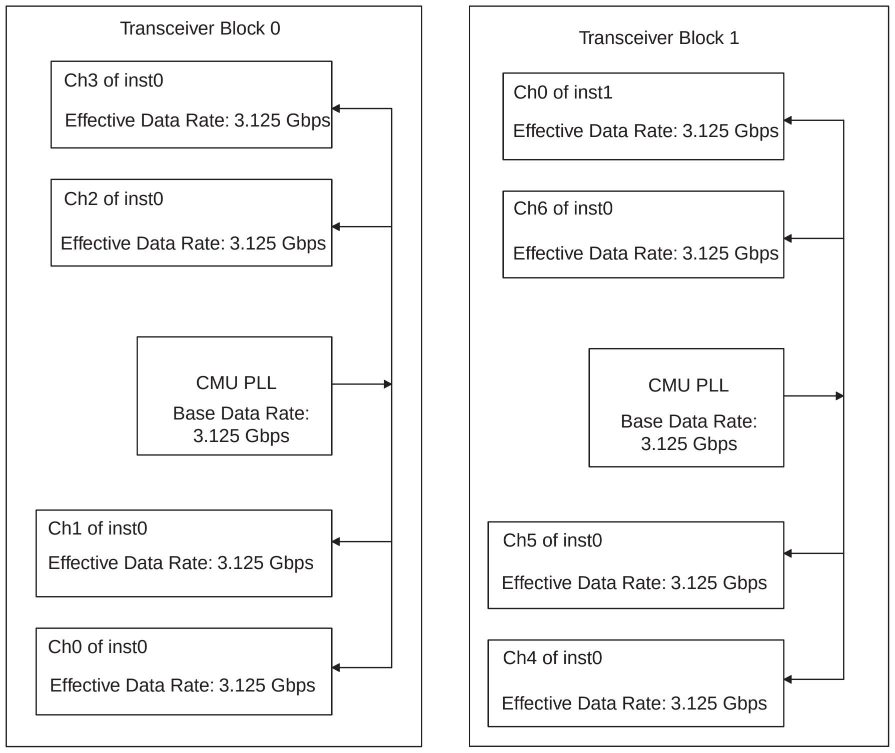

Inst0
Effective Data Rate: 3.125 Gbps
Input Clock Frequency: 156.125 MHz
Number of Channels: 7
Inst1
Effective Data Rate: 3.125 Gbps
Input Clock Frequency: 156.25 MHz
Number of Channels: 1

Figure 3–6 shows the transceiver instances after compilation for Example 4.

Figure 3–6. Combined Transceiver Instances after Compilation for Example 4


Transceiver Block 0
Ch3 of inst0
Effective Data Rate: 3.125 Gbps
Ch2 of inst0
Effective Data Rate: 3.125 Gbps
CMU PLL
Base Data Rate: 3.125 Gbps
Ch1 of inst0
Effective Data Rate: 3.125 Gbps
Ch0 of inst0
Effective Data Rate: 3.125 Gbps
Transceiver Block 1
Ch0 of inst1
Effective Data Rate: 3.125 Gbps
Ch6 of inst0
Effective Data Rate: 3.125 Gbps
CMU PLL
Base Data Rate: 3.125 Gbps
Ch5 of inst0
Effective Data Rate: 3.125 Gbps
Ch4 of inst0
Effective Data Rate: 3.125 Gbps

You can force the placement of the transceiver channels in specific transceiver banks by assigning pins to the `tx_dataout` and `rx_datain` ports of inst0 and inst1.

☞ Even though inst0 instantiates seven transceiver channels, the ALTGX MegaWizard Plug-In Manager provides only one bit for the `pll_inclk` port for inst0. In your design, provide only one clock input for the `pll_inclk` port. The Quartus II software uses two transceiver blocks to fit the seven channels and internally connects the input reference clock (connected to the `pll_inclk` port in your design) to the CMU PLLs of two transceiver blocks.

☞ For inst1, the ALTGX MegaWizard Plug-In Manager provides a `pll_inclk` port. In this example, it is assumed that a single reference clock is provided for inst0 and inst1. Therefore, connect the `pll_inclk` port of inst0 and inst1 to the same input reference clock pin. This enables the Quartus II software to share a single CMU PLL in transceiver block1 that has three channels of inst0 and one channel of inst1 (shown as ch5, ch6, and ch7 in transceiver block 1) in Figure 3–6.

For the Rx CDRs in inst0, the ALTGX MegaWizard Plug-In Manager provides seven bits for the `rx_cruclk` port (if you do not select the **Train Receiver CDR from pll_inclk** option in the **PLL/Ports** screen). This allows separate input reference clocks to the Rx CDRs of each channel.

Summary

The following summarizes how to configure multiple protocols and data rates in a transceiver block:

- You can run each transceiver channel at independent data rates or protocol functional modes.
- Each transceiver block consists of two CMU PLLs that provide clocks to run the transmitter channels within the transceiver block.
- To enable the Quartus II software to combine multiple instances of transceiver channels within a transceiver block, follow the rules specified in "General Requirements to Combine Channels" on page 3–2 and "Sharing CMU PLLs" on page 3–3.
- You can reset each CMU PLL within a transceiver block using a `pll_powerdown` signal. For each transceiver instance, the ALTGX MegaWizard Plug-In Manager provides an option to select the `pll_powerdown` port. If you want to share the same CMU PLL between multiple transceiver channels, connect the `pll_powerdown` ports of the instances and drive the signal from the same logic.
- If you enable the PCI Express hard IP block using the PCI Express compiler, the Quartus II software has certain requirements about using the remaining transceiver channels within the transceiver block in other configurations. For more information, refer to "Combining Channels Using the PCI Express hard IP Block with Other Channels" on page 3–12.

Document Revision History

Table 3–9 shows the revision history for this chapter.

Table 3–9. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial release.	—

4. Reset Control and Power Down

AIIGX52004-1.0

Introduction

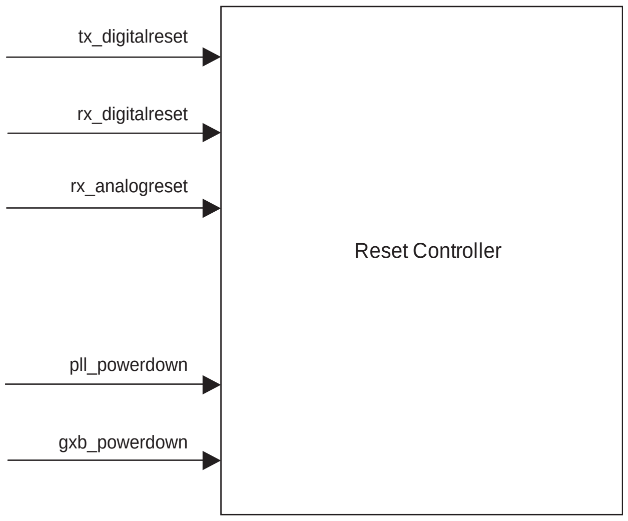
Arria® II GX devices offer multiple reset signals to control transceiver channels and clock multiplier unit (CMU) phase-locked loops (PLLs) independently. The ALTGX Transceiver MegaWizard™ Plug-In Manager provides individual reset signals for each channel instantiated in the design. It also provides one power-down signal for each transceiver block.

This chapter includes the following sections:

- "User Reset and Power-Down Signals" on page 4–1
- "Transceiver Reset Sequences" on page 4–4
- "Power Down" on page 4–17
- "Simulation Requirements" on page 4–18

Figure 4–1 shows the reset control and power-down block for an Arria II GX device.

Figure 4–1. Reset Control and Power-Down Block



User Reset and Power-Down Signals

Each transceiver channel in the Arria II GX device has individual reset signals to reset its physical coding sublayer (PCS) and physical medium attachment (PMA) blocks. Each CMU PLL in the transceiver block has a dedicated reset signal. The transceiver block also has a power-down signal that affects all the channels and CMU PLLs in the transceiver block.

☞ All reset and power-down signals are asynchronous.

The following reset signals are available for each transceiver channel:

- `tx_digitalreset`—provides asynchronous reset to all digital logic in the transmitter PCS, including the XAUI transmit state machine, the built-in self test (BIST) pseudo-random binary sequence (PRBS) generator, and the BIST pattern generator. This signal is available in the ALTGX MegaWizard Plug-In Manager in **Transmitter Only** and **Receiver and Transmitter** configurations. The minimum pulse width for this signal is two parallel clock cycles.
- `rx_digitalreset`—resets all digital logic in the receiver PCS, including the XAUI and GIGE receiver state machine, the XAUI channel alignment state machine, the BIST-PRBS verifier, and the BIST-incremental verifier. This signal is available in the ALTGX MegaWizard Plug-In Manager in **Receiver Only** and **Receiver and Transmitter** configurations. The minimum pulse width for this signal is two parallel clock cycles.

  The `tx_digitalreset` and `rx_digitalreset` signals must be asserted until the clocks out of the transmitter PLL and receiver CDR are stabilized. Stable parallel clocks are essential for proper operation of transmitter and receiver phase compensation FIFOs in the PCS.

- `rx_analogreset`—resets the receiver CDR present in the receiver channel. This signal is available in the ALTGX MegaWizard Plug-In Manager in **Receiver Only** and **Receiver and Transmitter** configurations. The minimum pulse width is two parallel clock cycles.

The following power-down signal is available for each CMU PLL in the transceiver block:

- `pll_powerdown`—each transceiver block has two CMU PLLs. Each CMU PLL has a dedicated power-down signal called `pll_powerdown`. The `pll_powerdown` signal powers down the CMU PLLs that provide high-speed serial and low-speed parallel clocks to the transceiver channels.

The following power-down signal is common to the transceiver block:

- `gxb_powerdown`—powers down the entire transceiver block. When this signal is asserted, the PCS and PMA in all the transceiver channels and the CMU PLLs are powered down. This signal operates independently from the other reset signals.

The `refclk` (`refclk0` or `refclk1`) buffer is not powered down by any of the above signals.

The following status signals are available:

- `pll_locked`—indicates the status of the transmitter PLL. A high on this signal shows that the transmitter PLL is locked to the incoming reference clock frequency.
- `rx_pll_locked`—a high on this signal shows that the receiver CDR is locked to the incoming reference clock frequency.

- `rx_freqlocked`—indicates the status of the receiver CDR lock mode. A high level indicates that the receiver is in lock-to-data mode. A low level indicates that the receiver CDR is in lock-to-reference mode.
- `busy`—indicates the status of the dynamic reconfiguration controller. The busy signal remains low for the first `reconfig_clk` clock cycle after power up. It then gets asserted from the second `reconfig_clk` clock cycle. Assertion on this signal indicates that the offset cancellation process is being executed on the receiver buffer as well as the receiver CDR. When this signal is de-asserted, it indicates that the offset cancellation is complete.

For more information, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II GX Devices*. This application note will be available in March, 2009. To review this document when it is available, go to the Literature page of the Altera website (www.altera.com).

If none of the channels is instantiated in a transceiver block, the Quartus® II software automatically powers down the entire transceiver block.

Blocks Affected by Reset and Power Down Signals

Table 4–1 shows the blocks that are affected by specific reset and power-down signals.

Table 4–1. Blocks Affected by Reset and Power-Down Signals

Transceiver Block	rx_digitalreset	rx_analogreset	tx_digitalreset	pll_powerdown	gxb_powerdown
CMU PLLs				✓	✓
Transmitter Phase Compensation FIFO			✓		✓
Byte Serializer			✓		✓
8B/10B Encoder			✓		✓
Serializer			✓		✓
Transmitter Buffer					✓
Transmitter XAUI State Machine			✓		✓
Receiver Buffer					✓
Receiver CDR		✓			✓
Receiver Deserializer					✓
Receiver Word Aligner	✓				✓
Receiver Deskew FIFO	✓				✓
Receiver Clock Rate Compensation FIFO	✓				✓
Receiver 8B/10B Decoder	✓				✓
Receiver Byte Deserializer	✓				✓
Receiver Byte Ordering	✓				✓
Receiver Phase Compensation FIFO	✓				✓
Receiver XAUI State Machine	✓				✓

Transceiver Reset Sequences

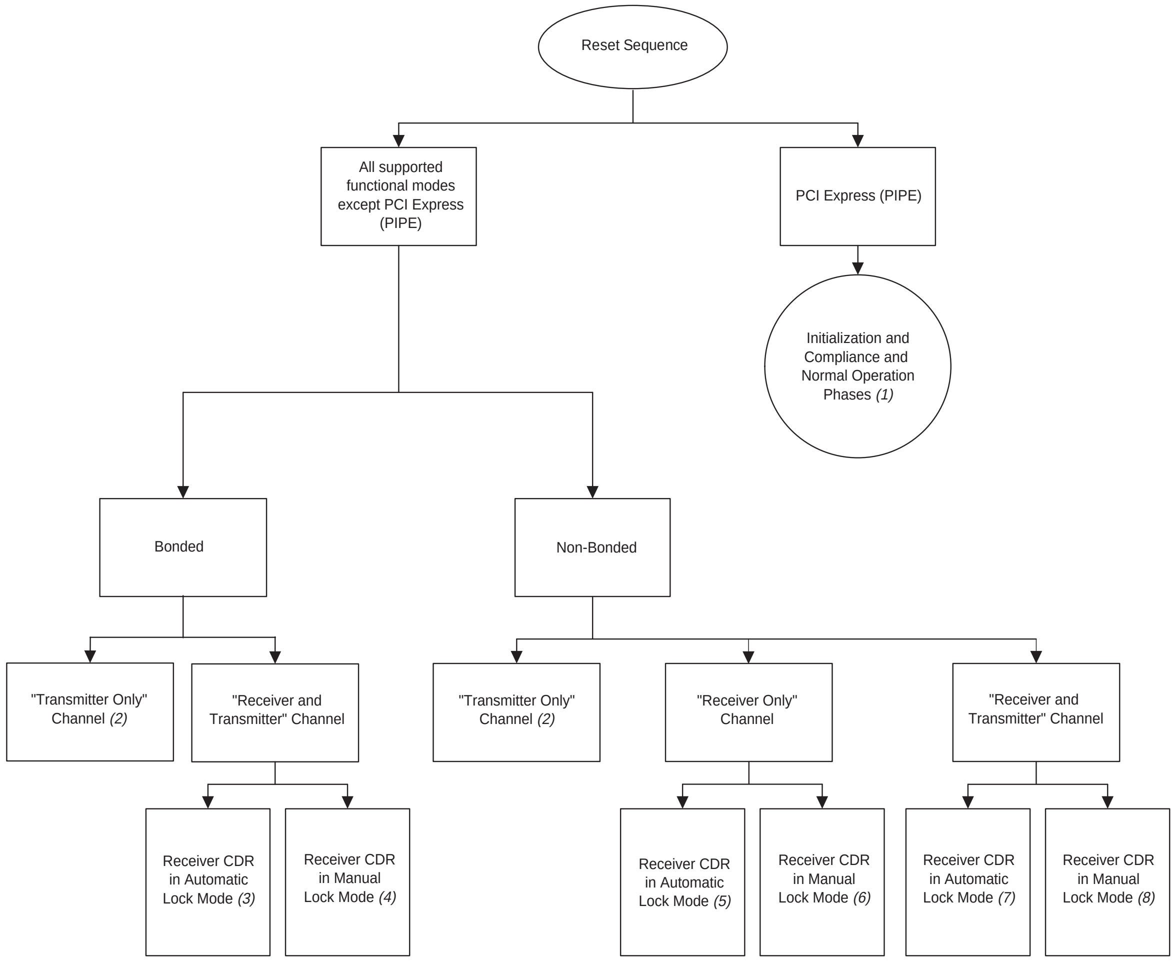
You can configure transceiver channels in Arria II GX devices in various configurations. In all functional modes except XAUI functional mode, transceiver channels can be either bonded or non-bonded. In XAUI functional mode, transceiver channels must be bonded. In PCI Express (PIPE) functional mode, transceiver channels can be either bonded or non-bonded and need to follow a specific reset sequence.

The two categories of reset sequences for Arria II GX devices described in this chapter are:

- "All Supported Functional Modes Except PCI Express (PIPE) Functional Mode" on page 4–5—describes the reset sequences in bonded and non-bonded configurations.
- "PCI Express (PIPE) Functional Mode" on page 4–15—describes the reset sequence for the initialization/compliance phase and normal operation phase in PCI Express (PIPE) functional modes.

☞ The `busy` signal remains low for the first `reconfig_clk` clock cycle. It then gets asserted from the second `reconfig_clk` clock cycle. Subsequent de-assertion of the `busy` signal indicates the completion of the offset cancellation process. This `busy` signal is required in transceiver reset sequences except for Transmitter Only channel configurations. Refer to the reset sequences shown in Figure 4–2 and the associated references listed in the notes.

Figure 4–2. Transceiver Reset Sequences Chart


Reset Sequence
All supported functional modes except PCI Express (PIPE)
PCI Express (PIPE)
Initialization and Compliance and Normal Operation Phases (1)
Bonded
Non-Bonded
"Transmitter Only" Channel (2)
"Receiver and Transmitter" Channel
"Transmitter Only" Channel (2)
"Receiver Only" Channel
"Receiver and Transmitter" Channel
Receiver CDR in Automatic Lock Mode (3)
Receiver CDR in Manual Lock Mode (4)
Receiver CDR in Automatic Lock Mode (5)
Receiver CDR in Manual Lock Mode (6)
Receiver CDR in Automatic Lock Mode (7)
Receiver CDR in Manual Lock Mode (8)

Note to Figure 4–2:

(1) Refer to the Timing Diagram in Figure 4–10.
(2) Refer to the Timing Diagram in Figure 4–3.
(3) Refer to the Timing Diagram in Figure 4–4.
(4) Refer to the Timing Diagram in Figure 4–5.
(5) Refer to the Timing Diagram in Figure 4–6.
(6) Refer to the Timing Diagram in Figure 4–7.
(7) Refer to the Timing Diagram in Figure 4–8.
(8) Refer to the Timing Diagram in Figure 4–9.

Altera strongly recommends adhering to these reset sequences for proper operation of the Arria II GX transceiver.

All Supported Functional Modes Except PCI Express (PIPE) Functional Mode

This section describes the reset sequences for transceiver channels in bonded and non-bonded configurations. Timing diagrams of some typical configurations are shown to facilitate proper reset sequence implementation. In these functional modes, you can set the receiver CDR either in automatic lock or manual lock mode.

In manual lock mode, the receiver CDR locks to the reference clock (lock-to-reference) or the incoming serial data (lock-to-data), depending on the logic levels on the `rx_locktorefclk` and `rx_locktodata` signals. With the receiver CDR in manual lock mode, you can either configure the transceiver channels in the Arria II GX device in a non-bonded configuration or a bonded configuration. In a bonded configuration, such as XAUI mode, four channels are bonded together.

Table 4–2 shows the lock-to-reference (LTR) and lock-to-data (LTD) controller lock modes for the `rx_locktorefclk` and `rx_locktodata` signals.

Table 4–2. Lock-To-Reference and Lock-To-Data Modes

rx_locktorefclk	rx_locktodata	LTR/LTD Controller Lock Mode
1	0	Manual, LTR Mode
—	1	Manual, LTD Mode
0	0	Automatic Lock Mode

Bonded Channel Configuration

In a bonded channel configuration, you can reset all the bonded channels simultaneously. Examples of bonded channel configurations are XAUI, PCI Express (PIPE), and Basic ×4 functional modes. In Basic ×4 functional mode, you can bond **Transmitter Only** channels together.

In XAUI mode, the receiver and transmitter channels are bonded. Each of the receiver channels in this mode has its own output status signals, `rx_pll_locked` and `rx_freqlocked`. The timing of these signals is considered in the reset sequence.

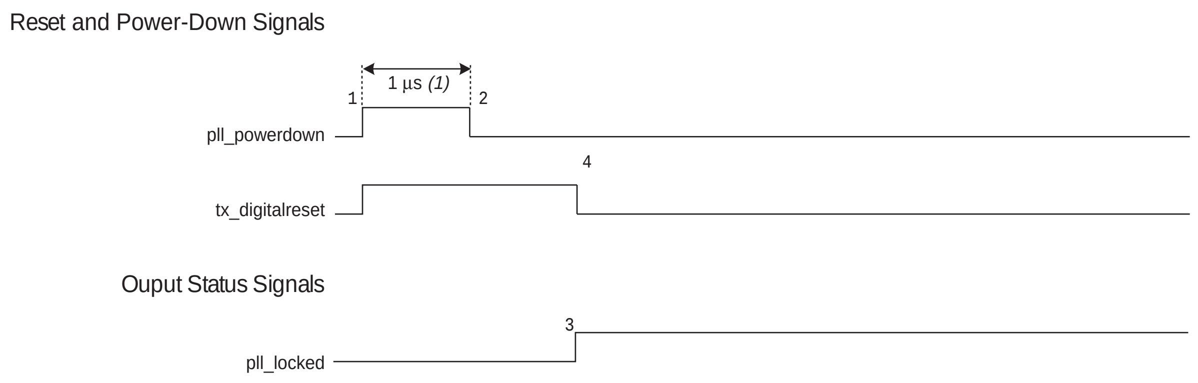
The following timing diagrams describe the reset and power-down sequences for bonded configurations under the following set-ups:

- **Transmitter Only** channel set-up—applicable to Basic ×4 functional mode
- **Receiver and Transmitter** channel set-up—receiver CDR in automatic lock mode; applicable to XAUI functional mode
- **Receiver and Transmitter** channel set-up—receiver CDR in manual lock mode; applicable to XAUI functional mode

Transmitter Only Channel

This configuration contains only a transmitter channel. If you create a **Transmitter Only** instance in the ALTGX MegaWizard Plug-In Manager in Basic ×4 functional mode, use the reset sequence shown in Figure 4–3.

Figure 4–3. Sample Reset Sequence for Four Transmitter Only Channels



Note to Figure 4–3:

(1) To be characterized.

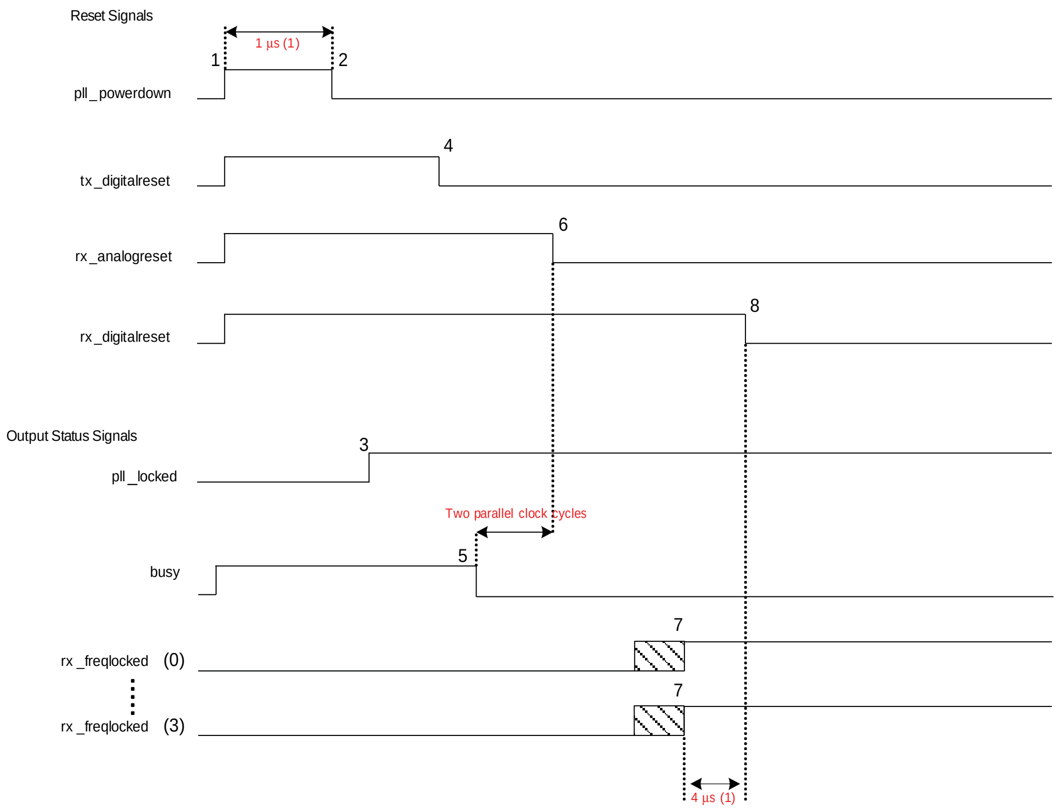
As shown in Figure 4–3, perform the following reset sequence steps for the **Transmitter Only** channel configuration:

1. After power up, assert `pll_powerdown` for a minimum period of 1 µs (the time between markers 1 and 2).
2. Keep the `tx_digitalreset` signal asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. After the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), de-assert the `tx_digitalreset` signal (marker 4). The transmitter is ready for transmitting data.

Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode

This configuration contains both a transmitter and receiver channel. For XAUI functional mode, with the receiver CDR in automatic lock mode, use the reset sequence shown in Figure 4–4.

Figure 4–4. Sample Reset Sequence for Four Receiver and Transmitter Channels—Receiver CDR in Automatic Lock Mode



Note to Figure 4–4:

(1) To be characterized.

As shown in Figure 4–4, perform the following reset sequence steps for the receiver CDR in automatic lock mode configuration:

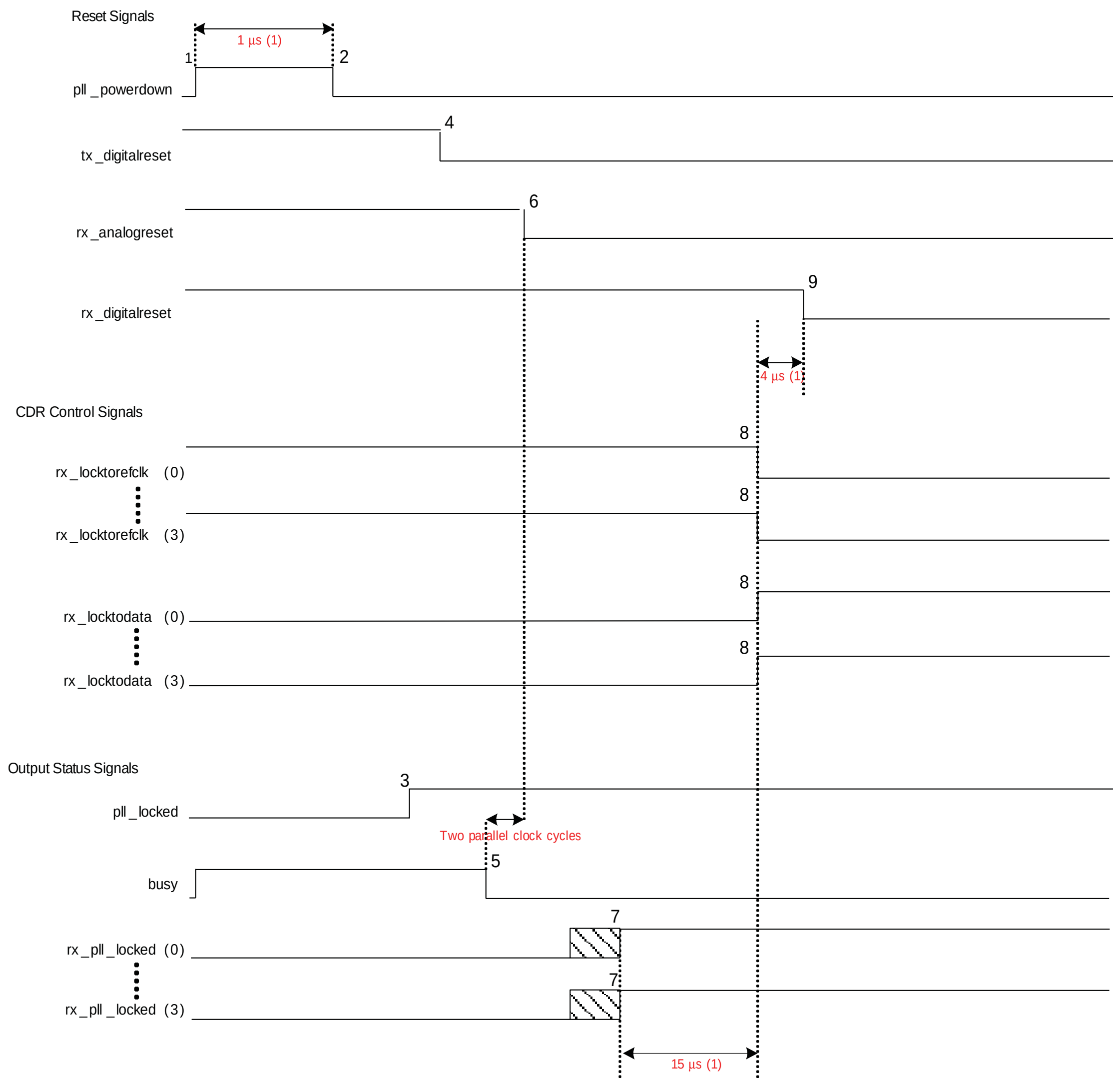
1. After power up, assert `pll_powerdown` for a minimum period of 1 µs (the time between markers 1 and 2).
2. Keep the `tx_digitalreset`, `rx_analogreset`, and `rx_digitalreset` signals asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. After the transmitter PLL locks, as indicated by the `pll_locked` signal going high, de-assert the `tx_digitalreset` signal. At this point, the transmitter is ready for data traffic.
4. For the receiver operation, after de-assertion of the `busy` signal, wait for two parallel clock cycles to de-assert the `rx_analogreset` signal. After `rx_analogreset` is de-asserted, the receiver CDR of each channel starts locking to the receiver input reference clock.
5. Wait for the `rx_freqlocked` signal from each channel to go high. The `rx_freqlocked` signal of each channel may go high at different times (indicated by the slashed pattern at marker 7).

6. In a bonded channel group, when the `rx_freqlocked` signals of all the channels have gone high, from that point onwards wait for at least 4 µs for the receiver parallel clock to be stable, then de-assert the `rx_digitalreset` signal (marker 8). At this point, all the receivers are ready for data traffic.

Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode

This configuration contains both a transmitter and receiver channel. For XAUI functional mode, with the receiver CDR in manual lock mode, use the reset sequence shown in Figure 4–5.

Figure 4–5. Sample Reset Sequence of Four Receiver and Transmitter Channels—Receiver CDR in Manual Lock Mode



Note to Figure 4–5:

(1) To be characterized.

As shown in Figure 4–5, perform the following reset sequence steps for the receiver CDR in manual lock mode configuration:

1. After power up, assert `pll_powerdown` for a minimum period of 1 µs (the time between markers 1 and 2).
2. Keep the `tx_digitalreset`, `rx_analogreset`, `rx_digitalreset`, and `rx_locktorefclk` signals asserted and the `rx_locktodata` signal de-asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. After the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), de-assert the `tx_digitalreset` signal (marker 4). For the receiver operation, after de-assertion of the `busy` signal, wait for two parallel clock cycles to de-assert the `rx_analogreset` signal. After the `rx_analogreset` signal is de-asserted, the receiver CDR of each channel starts locking to the receiver input reference clock because `rx_locktorefclk` is asserted.
4. Wait for the `rx_pll_locked` signal from each channel to go high. The `rx_pll_locked` signal of each channel may go high at different times with respect to each other (indicated by the slashed pattern at the marker 7).
5. In a bonded channel group, when the last `rx_pll_locked` signal goes high, from that point onwards, wait at least 15 µs and then de-assert `rx_locktorefclk` and assert `rx_locktodata` (marker 8). At this point, the receiver CDR enters lock-to-data mode and the receiver PLL starts locking to the received data.
6. De-assert `rx_digitalreset` at least 4 µs (the time between markers 8 and 9) after asserting the `rx_locktodata` signal.

Non-Bonded Channel Configuration

In non-bonded channels, each channel in the ALTGX megafunction instance contains its own `tx_digitalreset`, `rx_analogreset`, `rx_digitalreset`, `rx_pll_locked`, and `rx_freqlocked` signals.

You can reset each channel independently. For example, if there are four non-bonded channels, the ALTGX MegaWizard Plug-In Manager provides five signals: `tx_digitalreset`, `rx_analogreset`, `rx_digitalreset`, `rx_pll_locked`, and `rx_freqlocked`.

The following timing diagrams describe the reset and power-down sequences for one channel in a non-bonded configuration, under five different set-ups:

- **Transmitter Only** channel set-up
- **Receiver Only** channel set-up—receiver CDR in automatic lock mode
- **Receiver Only** channel set-up—receiver CDR in manual lock mode
- **Receiver and Transmitter** channel set-up—receiver CDR in automatic lock mode
- **Receiver and Transmitter** channel set-up—receiver CDR in manual lock mode

☞ Follow the same reset sequence for all the other channels in the non-bonded configuration.

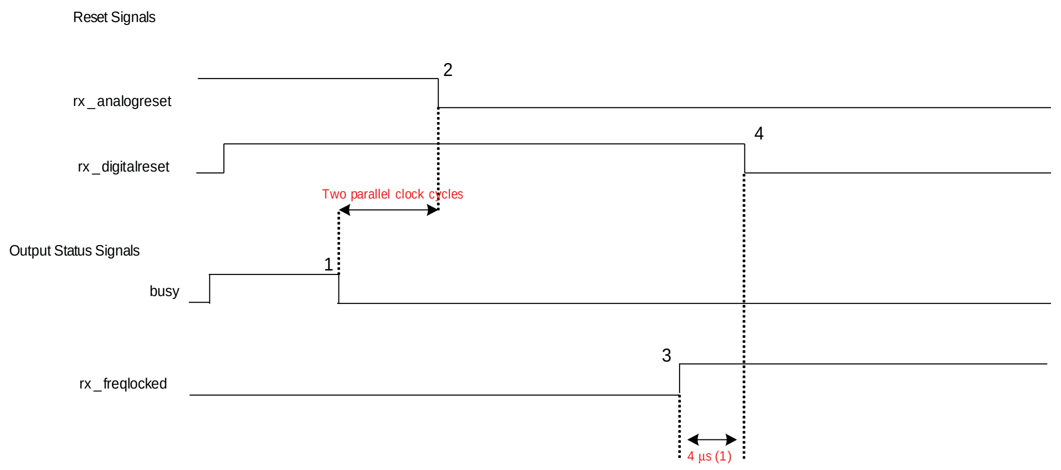
Transmitter Only Channel

This configuration contains only a transmitter channel. If you create a **Transmitter Only** instance in the ALTGX MegaWizard Plug-In Manager, use the same reset sequence as shown in Figure 4–2.

Receiver Only Channel—Receiver CDR in Automatic Lock Mode

This configuration contains only a receiver channel. If you create a **Receiver Only** instance in the ALTGX MegaWizard Plug-In Manager with the receiver CDR in automatic lock mode, use the reset sequence shown in Figure 4–6.

Figure 4–6. Sample Reset Sequence of Receiver-Only Channel—Receiver CDR in Automatic Lock Mode



Note to Figure 4–6:

(1) To be characterized.

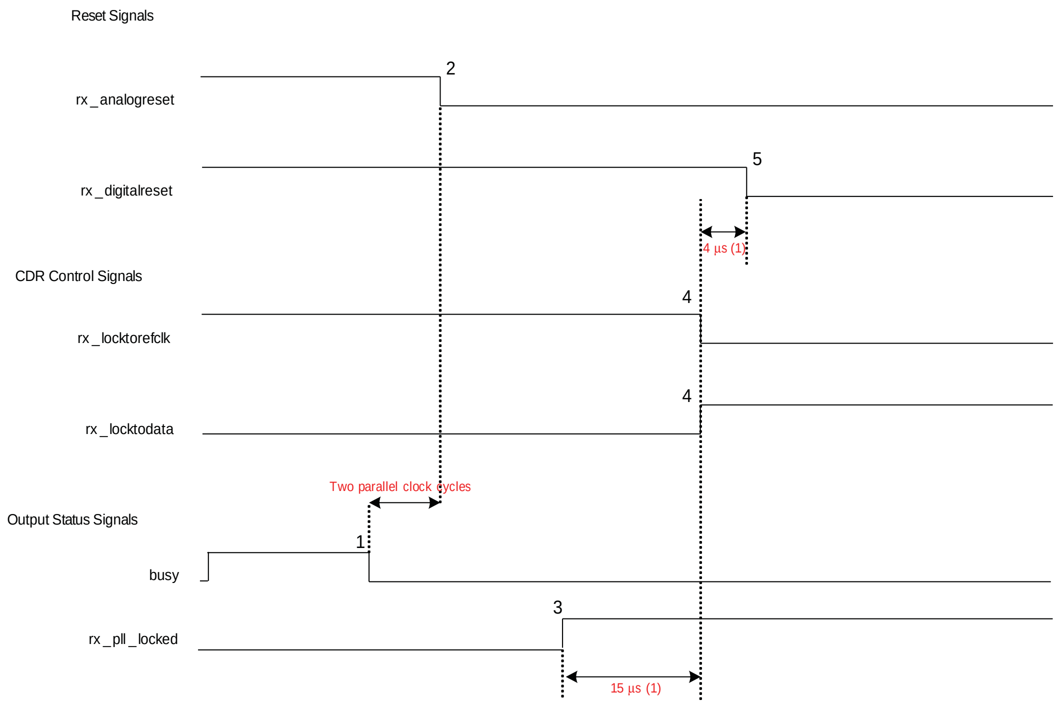
As shown in Figure 4–6, perform the following reset sequence steps for the receiver CDR in automatic lock mode configuration:

1. After power up, wait for the `busy` signal to be de-asserted (marker 1).
2. De-assert the `rx_analogreset` signal (marker 2).
3. Keep the `rx_digitalreset` signal asserted during this time period. After you de-assert the `rx_analogreset` signal, the receiver PLL starts locking to the receiver input reference clock.
4. Wait for the `rx_freqlocked` signal to go high (marker 3).
5. After `rx_freqlocked` goes high, wait at least 4 µs and then de-assert the `rx_digitalreset` signal (marker 4). At this point, the receiver is ready to receive data.

Receiver Only Channel—Receiver CDR in Manual Lock Mode

This configuration contains only a receiver channel. If you create a **Receiver Only** instance in the ALTGX MegaWizard Plug-In Manager with the receiver CDR in manual lock mode, use the reset sequence shown in Figure 4–7.

Figure 4–7. Sample Reset Sequence of Receiver-Only Channel—Receiver CDR in Manual Lock Mode



Note to Figure 4–7:

(1) To be characterized.

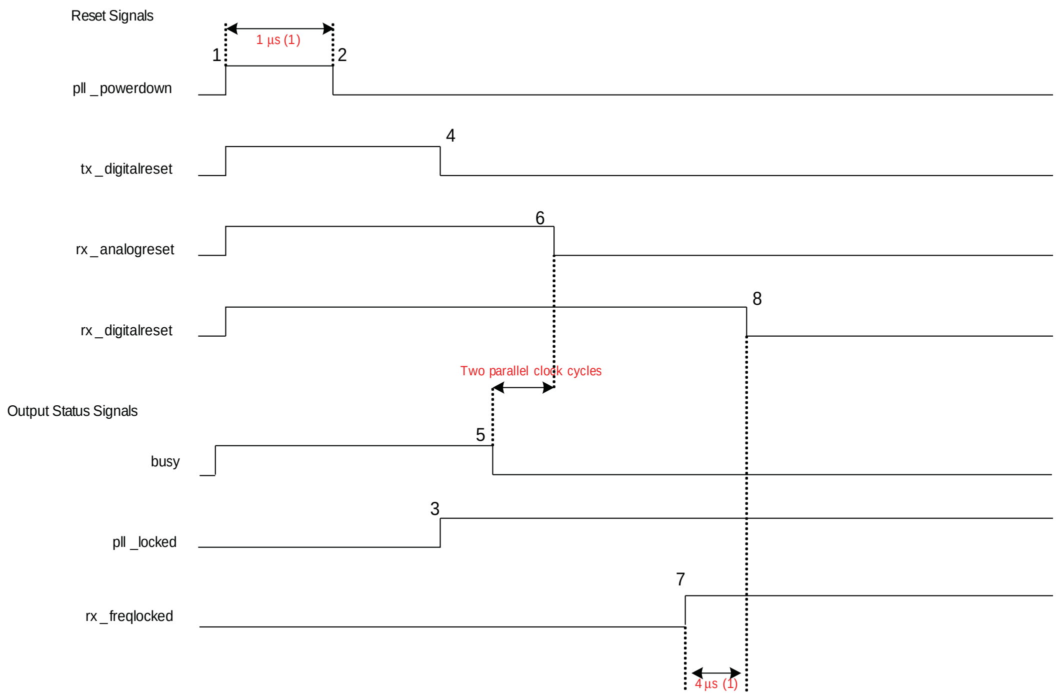
As shown in Figure 4–7, perform the following reset sequence steps for the receiver CDR in manual lock mode:

1. After power up, assert `rx_analogreset` for a minimum period of two parallel clock cycles (the time between markers 1 and 2).
2. Keep the `rx_digitalreset` and `rx_locktorefclk` signals asserted and the `rx_locktodata` signal de-asserted during this time period.
3. After de-assertion of the `busy` signal, de-assert the `rx_analogreset` signal, after which the receiver CDR starts locking to the receiver input reference clock because the `rx_locktorefclk` signal is asserted.
4. Wait at least 15 µs (the time between markers 3 and 4) after the `rx_pll_locked` signal goes high and then de-assert the `rx_locktorefclk` signal. At the same time, assert the `rx_locktodata` signal (marker 4). At this point, the receiver CDR enters lock-to-data mode and the receiver PLL starts locking to the received data.
5. De-assert `rx_digitalreset` at least 4 µs (the time between markers 4 and 5) after asserting the `rx_locktodata` signal.

Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode

This configuration contains both a transmitter and receiver channel. If you create a **Receiver and Transmitter** instance in the ALTGX MegaWizard Plug-In Manager with the receiver CDR in automatic lock mode, use the reset sequence shown in Figure 4–8.

Figure 4–8. Sample Reset Sequence of Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode



Note to Figure 4–8:

(1) To be characterized.

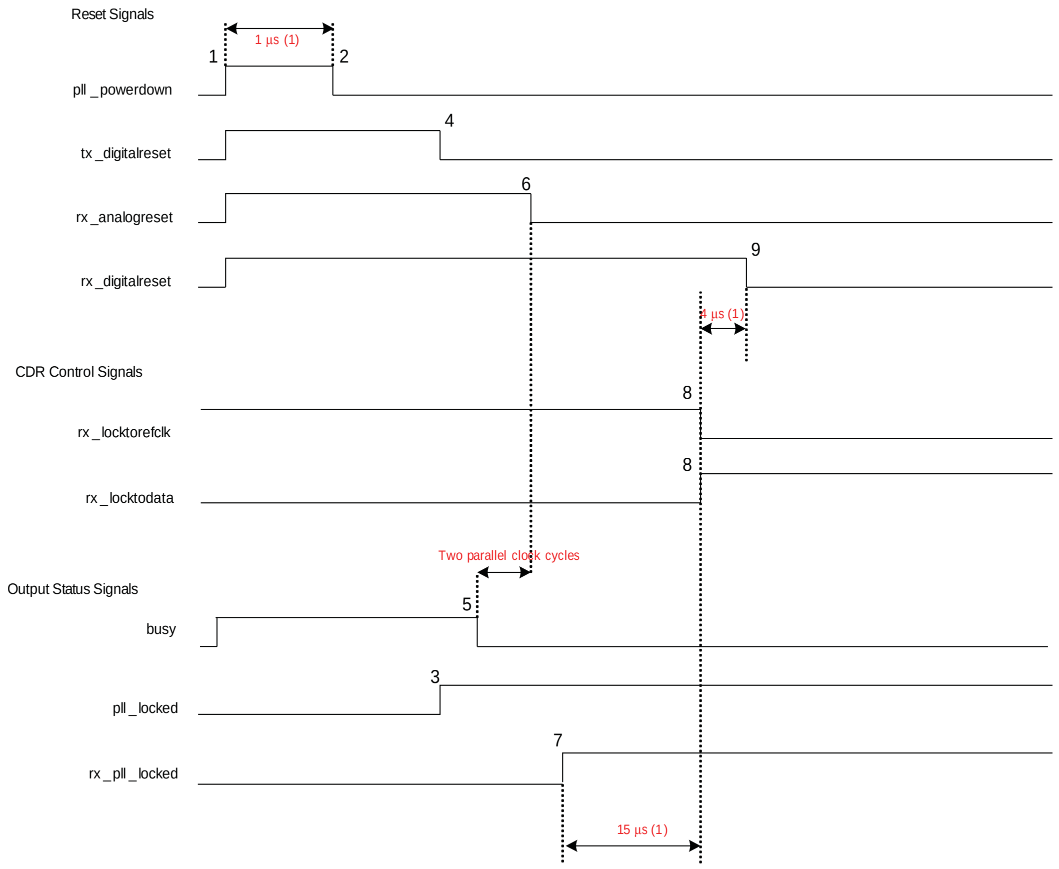
As shown in Figure 4–8, perform the following reset sequence steps for the receiver CDR in automatic lock mode:

1. After power up, assert `pll_powerdown` for a minimum period of 1 µs (the time between markers 1 and 2).
2. Keep the `tx_digitalreset`, `rx_analogreset`, and `rx_digitalreset` signals asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. After the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), de-assert `tx_digitalreset`. For receiver operation, wait for the `busy` signal to be de-asserted, after which `rx_analogreset` is de-asserted. After you de-assert `rx_analogreset`, the receiver CDR starts locking to the receiver input reference clock.
4. Wait for the `rx_freqlocked` signal to go high (marker 7).
5. After the `rx_freqlocked` signal goes high, wait at least 4 µs and then de-assert the `rx_digitalreset` signal (marker 8). The transmitter and receiver are ready for data traffic.

Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode

This configuration contains both a transmitter and receiver channel. If you create a **Receiver and Transmitter** instance in the ALTGX MegaWizard Plug-In Manager with the receiver CDR in manual lock mode, use the reset sequence shown in Figure 4–9.

Figure 4–9. Sample Reset Sequence of Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode



Note to Figure 4–9:

(1) To be characterized.

As shown in Figure 4–9, perform the following reset sequence steps for the receiver in manual lock mode:

1. After power up, assert `pll_powerdown` for a minimum period of 1 µs (the time between markers 1 and 2).
2. Keep the `tx_digitalreset`, `rx_analogreset`, `rx_digitalreset`, and `rx_locktorefclk` signals asserted and the `rx_locktodata` signal de-asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. After the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), de-assert `tx_digitalreset`. For receiver operation, wait for the `busy` signal to be de-asserted. At this point, `rx_analogreset` is de-asserted. After `rx_analogreset` is de-asserted, the receiver CDR starts locking to the receiver input reference clock because `rx_locktorefclk` is asserted.
4. Wait for at least 15 µs (the time between markers 7 and 8) after the `rx_pll_locked` signal goes high, then de-assert the `rx_locktorefclk` signal. At the same time, assert the `rx_locktodata` signal (marker 8). At this point, the receiver CDR enters lock-to-data mode and the receiver CDR starts locking to the received data.
5. De-assert `rx_digitalreset` at least 4 µs (the time between markers 8 and 9) after asserting the `rx_locktodata` signal.

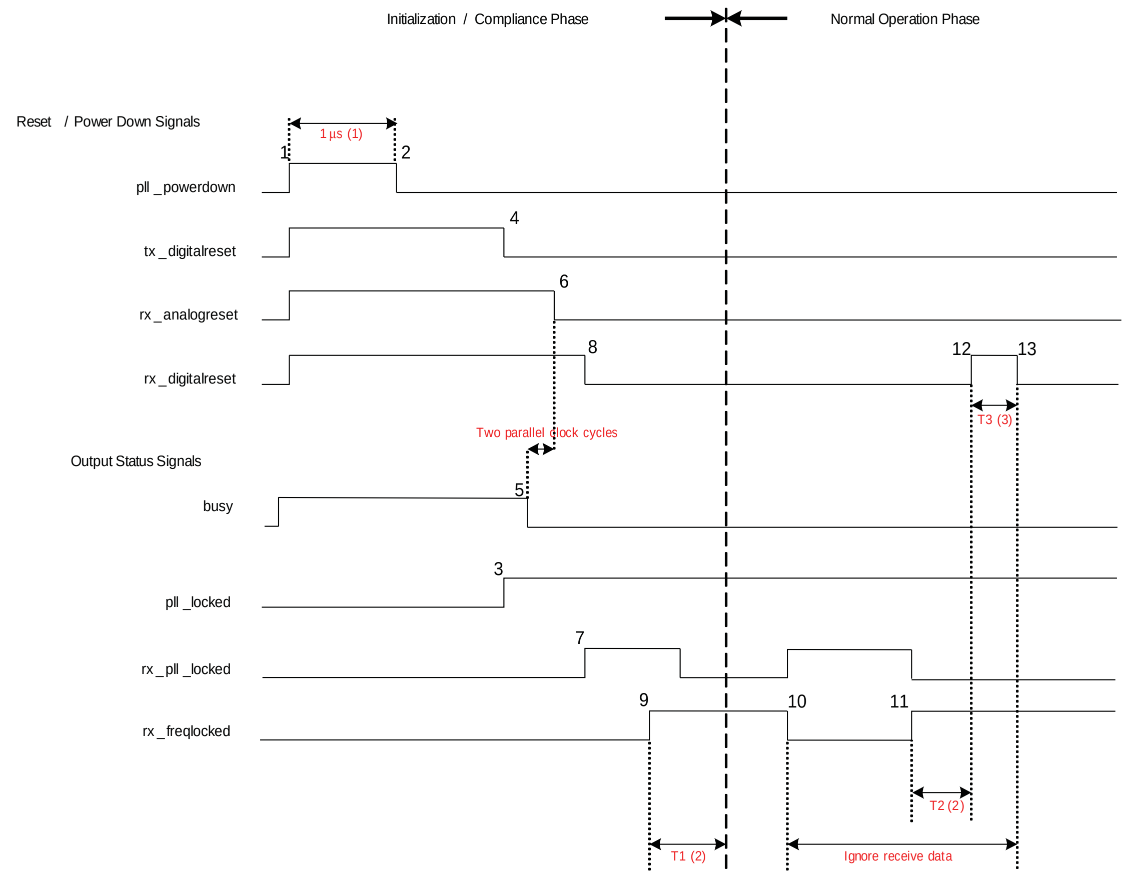
PCI Express (PIPE) Functional Mode

You can configure PCI Express (PIPE) functional mode with or without the receiver clock rate compensation FIFO in the Arria II GX device. The reset sequence remains the same of whether or not you use the receiver clock rate compensation FIFO.

PCI Express (PIPE) Reset Sequence

PCI Express (PIPE) protocol consists of initialization/compliance phase and normal operation phase. The reset sequences for these two phases are discussed based on the timing diagram in Figure 4–10.

Figure 4–10. Reset Sequence of PCI Express (PIPE) Functional Mode



Notes to Figure 4–10:

(1) To be characterized.

(2) The minimum T1 and T2 period is 4 µs.

(3) The minimum T3 period is two parallel clock cycles.

PCI Express (PIPE) Initialization/Compliance Phase

After the device is powered up, a PCI Express (PIPE)-compliant device goes through the compliance phase during initialization. The `rx_digitalreset` signal must be de-asserted during this compliance phase to achieve transitions on the `pipephydonestatus` signal, as expected by the link layer.

The `rx_digitalreset` signal is de-asserted based on the assertion of the `rx_freqlocked` signal.

During the initialization/compliance phase, do not use the `rx_freqlocked` signal to trigger a de-assertion of the `rx_digitalreset` signal. Instead, perform the following reset sequence:

1. After power up, assert `pll_powerdown` for a minimum period of 1 µs (the time between markers 1 and 2). Keep the `tx_digitalreset`, `rx_analogreset`, and `rx_digitalreset` signals asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.

2. After the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), de-assert `tx_digitalreset`. For receiver operation, wait for the `busy` signal to be de-asserted and for `rx_analogreset` to be de-asserted. After `rx_analogreset` is de-asserted, the receiver CDR starts locking to the receiver input reference clock.
3. When the receiver CDR locks to the input reference clock, as indicated by the `rx_pll_locked` signal going high at marker 7 in Figure 4–10, de-assert the `rx_digitalreset` signal (marker 8). After de-asserting `rx_digitalreset`, the `pipephydonestatus` signal transitions from the transceiver channel to indicate the status to the link layer. Depending on its status, `pipephydonestatus` helps with the continuation of the compliance phase. After successful completion of this phase, the device enters into the normal operation phase.

PCI Express (PIPE) Normal Phase

1. After completion of the Initialization/Compliance phase when the `rx_freqlocked` signal is de-asserted, (marker 10 in Figure 4–10), wait for the `rx_pll_locked` signal assertion signifying the lock-to-reference clock.
2. Next, wait for the `rx_freqlocked` signal to go high again. In this phase, the received data is valid (not electrical idle) and the receiver CDR locks to the incoming data.
3. Proceed with the reset sequence after assertion of the `rx_freqlocked` signal.
4. After the `rx_freqlocked` signal goes high, wait for at least 4 µs before asserting `rx_digitalreset` (marker 12 in Figure 4–10) for two parallel receive clock cycles so that the receiver phase compensation FIFO is initialized.

Data from the transceiver block is not valid from the time the `rx_freqlocked` signal goes low (marker 10 in Figure 4–10) to the time the `rx_digitalreset` is de-asserted (marker 13 in Figure 4–10). The PLD logic ignores the data during this period (between markers 10 and 13 in Figure 4–10).

☞ You can configure the Arria II GX device in ×1, ×2, ×4, and ×8 PIPE lane configurations. The reset sequence described in "PCI Express (PIPE) Reset Sequence" on page 4–15 applies to all these multi-lane configurations.

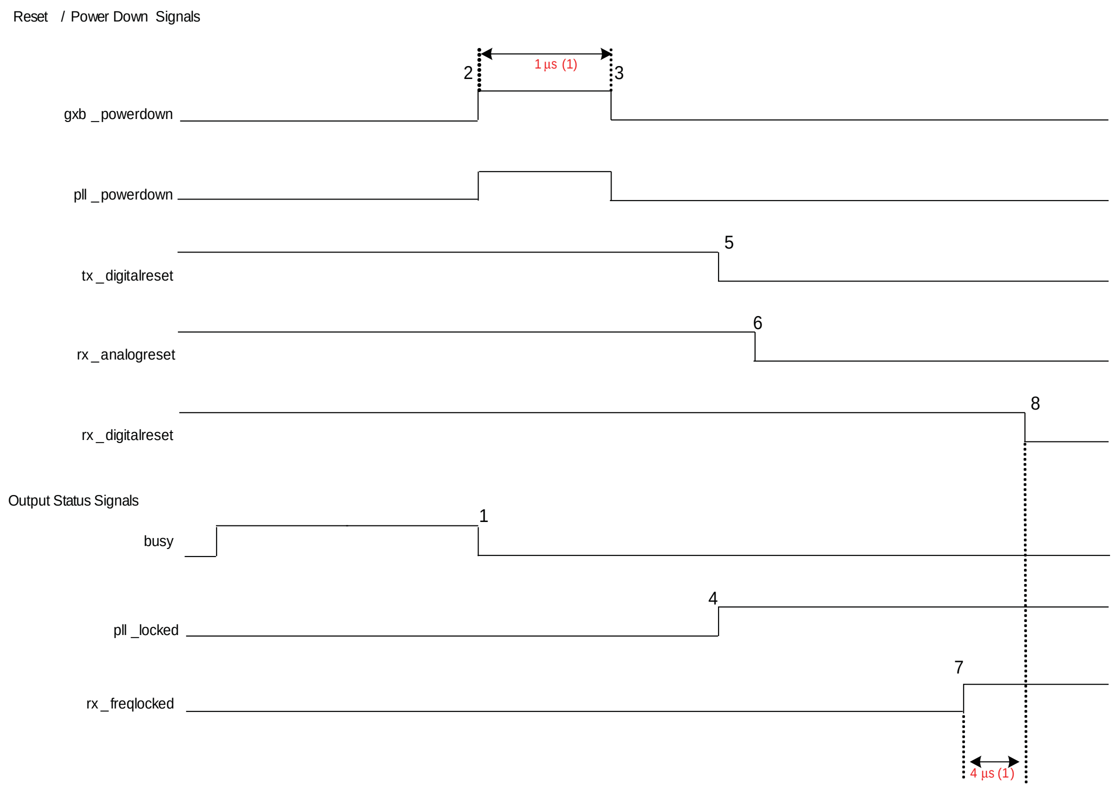
Power Down

The Quartus II software automatically selects the power down channel feature, which takes effect when you configure the Arria II GX device. All unused transceiver channels and blocks are powered down to reduce overall power consumption.

The `gxb_powerdown` signal is an optional transceiver block signal. It powers down all the blocks in the transceiver block. The minimum pulse width for this signal is 1 µs.

After power up, if you use the `gxb_powerdown` signal, wait for de-assertion of the `busy` signal, then assert the `gxb_powerdown` signal for a minimum of 1 µs. To finish, follow the sequence in Figure 4–11.

Figure 4–11. Sample Reset Sequence of Four Receiver and Transmitter Channels—Receiver CDR in Automatic Lock Mode with Optional gxb_powerdown Signal



Note to Figure 4–11:

(1) To be characterized.

Simulation Requirements

The following are simulation requirements:

- The `gxb_powerdown` port is optional. In simulation, if the `gxb_powerdown` port is not instantiated, you must assert the `tx_digitalreset`, `rx_digitalreset`, and `rx_analogreset` signals appropriately for correct simulation behavior.
- If the `gxb_powerdown` port is instantiated, and the other reset signals are not used, you must assert the `gxb_powerdown` signal for at least one parallel clock cycle for correct simulation behavior.
- You can de-assert the `rx_digitalreset` signal immediately after the `rx_freqlocked` signal goes high to reduce the simulation run time. It is not necessary to wait 4 µs (as suggested in the actual reset sequence).
- The `busy` signal is de-asserted after about 20 parallel `reconfig_clk` clock cycles in order to reduce the simulation run time. For silicon behavior in hardware, follow the reset sequences described in this chapter.
- In PCI Express (PIPE) mode simulation, you must assert the `tx_forceelecidle` signal for at least one parallel clock cycle before transmitting normal data for correct simulation behavior.

Document Revision History

Table 4–3 shows the revision history for this chapter.

Table 4–3. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial release.	—

About this Handbook

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

How to Contact Altera

For the most up-to-date information about Altera products, see the following table.

Contact *(Note 1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Altera literature services	Email	literature@altera.com
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions that this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicates command names and dialog box titles. For example, **Save As** dialog box.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, dialog box options, software utility names, and other GUI labels. For example, **\qdesigns** directory, **d:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicates document titles. For example, *AN 519: Stratix IV Design Guidelines.*
Italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, <*file name*> and <*project name*>**.pof** file.
Initial Capital Letters	Indicates keyboard keys and menu names. For example, Delete key and the Options menu.
"Subheading Title"	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, "Typographic Conventions."

Visual Cue	Meaning
`Courier type`	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. Active-low signals are denoted by suffix `n`. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
1., 2., 3., and a., b., c., and so on.	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■	Bullets indicate a list of items when the sequence of the items is not important.
☞	The hand points to information that requires special attention.
⚠ CAUTION	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
⚠ WARNING	A warning calls attention to a condition or possible situation that can cause you injury.
↵	The angled arrow instructs you to press **Enter**.
👣	The feet direct you to more information about a particular topic.

Arria II GX Device Handbook

Volume 3

101 Innovation Drive
San Jose, CA 95134
www.altera.com

AIIGX5V3-1.0

Contents

Chapter Revision Dates

The chapters in this book, *Arria II 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	Arria II GX Device Data Sheet	
	Revised:	*February 2009*
	Part Number:	*AIIGX53001-1.0*

Section I. Arria II GX Device Data Sheet

This section provides information about the Arria® II GX device data sheet. This section includes the following chapters:

- Chapter 1, Arria II GX Device Data Sheet

Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in this volume.

1. Arria II GX Device Data Sheet

AIIGX53001-1.0

Introduction

This chapter describes the electrical and switching characteristics of the Arria® II GX device family.

This chapter contains the following sections:

- "Electrical Characteristics" on page 1–1
- "Switching Characteristics" on page 1–11
- "Glossary" on page 1–30

Electrical Characteristics

The following sections describe the electrical characteristics.

Operating Conditions

When Arria II GX devices are implemented in a system, they are rated according to a set of defined parameters. To maintain the highest possible performance and reliability of Arria II GX devices, system designers must consider the following operating requirements:

- Arria II GX devices are offered in both commercial and industrial grades.
- Commercial devices are offered in –4 (fastest), –5, and –6 (slowest) speed grades.
- Industrial device is only offered in –5 speed grade.

☞ In this chapter, a prefix associated with the operating temperature range is attached to the speed grades; commercial with the "C" prefix, and industrial with the "I" prefix. Commercial devices are therefore indicated as C4, C5, and C6 speed grade respectively, while the industrial device is indicated as I5.

Absolute Maximum Ratings

Absolute maximum ratings define the maximum operating conditions for Arria II GX devices. The values are based on experiments conducted with the device and theoretical modeling of breakdown and damage mechanisms. The functional operation of the device is not implied under these conditions.

☞ Conditions beyond those listed in Table 1–1 may cause permanent damage to the device. Additionally, device operation at the absolute maximum ratings for extended periods of time may have adverse effects on the device.

Table 1–1. Arria II GX Device Absolute Maximum Ratings

Symbol	Description	Minimum	Maximum	Unit
V_{CC}	Supplies power to the core, periphery, I/O registers, PCIe HIP block, and transceiver PCS	-0.5	1.35	V
V_{CCCB}	Supplies power to the configuration RAM bits	-0.5	1.65	V
V_{CCBAT}	Battery back-up power supply for design security volatile key register	-0.5	3.75	V
V_{CCPD}	Supplies power to the I/O pre-drivers, differential input buffers, and MSEL circuitry	-0.5	3.75	V
V_{CCIO}	Supplies power to the I/O banks	-0.5	3.9	V
V_{CCD_PLL}	Supplies power to the digital portions of the PLL	-0.5	1.35	V
V_{CCA_PLL}	Supplies power to the analog portions of the PLL and device-wide power management circuitry	-0.5	3.75	V
V_I	DC input voltage	-0.5	4.0	V
V_{CCA}	Supplies power to the transceiver PMA regulator	—	2.625	V
V_{CCL_GXB}	Supplies power to the transceiver PMA TX, PMA RX, and clocking	—	1.21	V
V_{CCH_GXB}	Supplies power to the transceiver PMA output (TX) buffer	—	1.54	V
T_J	Operating junction temperature	-40	100	°C
T_{STG}	Storage temperature (no bias)	-65	150	°C

Maximum Allowed Overshoot and Undershoot Voltage

During transitions, input signals may overshoot to the voltage shown in Table 1–2 and undershoot to -2.0 V for magnitude of currents less than 100 mA and periods shorter than 20 ns.

Table 1–2 lists the maximum allowed input overshoot voltage and the duration of the overshoot voltage as a percentage over the device lifetime. The maximum allowed overshoot duration is specified as a percentage of high-time over the lifetime of the device. A DC signal is equivalent to 100% duty cycle. For example, a signal that overshoots to 4.3 V can only be at 4.3 V for 5.41% over the lifetime of the device: for a device lifetime of 10 years, this amounts to 5.41/10ths of a year.

Table 1–2. Maximum Allowed Overshoot During Transitions

Symbol	Description	Condition	Overshoot Duration as % of High Time	Unit
V_I (AC)	AC Input Voltage	4.0 V	100.000	%
		4.05 V	79.330	%
		4.1 V	46.270	%
		4.15 V	27.030	%
		4.2 V	15.800	%
		4.25 V	9.240	%
		4.3 V	5.410	%
		4.35 V	3.160	%
		4.4 V	1.850	%
		4.45 V	1.080	%
		4.5 V	0.630	%
		4.55 V	0.370	%
		4.6 V	0.220	%

Maximum Allowed I/O Operating Frequency

Table 1–3 defines the maximum allowed I/O operating frequency for I/Os using the specified I/O standards to ensure device reliability.

Table 1–3. Maximum Allowed I/O Operating Frequency

I/O Standard	I/O Frequency (MHz)
SSTL-18, SSTL -15 HSTL-18, HSTL-15	300
3.3-V and 3.0-V LVTTL 3.3-V, 3.0-V, 2.5-V, 1.8-V, 1.5-V LVCMOS PCI and PCI-X SSTL-2	250
1.2-V LVCMOS HSTL-12	200

Recommended Operating Conditions

This section lists the functional operation limits for AC and DC parameters for Arria II GX devices. The steady-state voltage and current values expected from Arria II GX devices are provided in Table 1–4. All supplies are required to monotonically reach their full-rail values without plateaus within t_{RAMP}.

Table 1–4 shows the recommended operating conditions for Arria II GX device.

Table 1–4. Arria II GX Device Recommended Operating Conditions

Symbol	Description	Condition	Minimum	Typical	Maximum	Unit
V_{CC}	Supplies power to the core, periphery, I/O registers, PCIe HIP block, and transceiver PCS	—	0.87	0.90	0.93	V
V_{CCCB}	Supplies power to the configuration RAM bits	—	1.425	1.50	1.575	V
V_{CCBAT} *(2)*	Battery back-up power supply for design security volatile key registers	—	1.2	—	3.3	V
V_{CCPD}	Supplies power to the I/O pre-drivers, differential input buffers, and MSEL circuitry	—	3.135	3.3	3.465	V
		—	2.85	3.0	3.15	V
		—	2.375	2.5	2.625	V
V_{CCIO}	Supplies power to the I/O banks *(1)*	—	3.135	3.3	3.465	V
		—	2.85	3.0	3.15	V
		—	2.375	2.5	2.625	V
		—	1.71	1.8	1.89	V
		—	1.425	1.5	1.575	V
		—	1.14	1.2	1.26	V
V_{CCD_PLL}	Supplies power to the digital portions of the PLL	—	0.87	0.90	0.93	V
V_{CCA_PLL}	Supplies power to the analog portions of the PLL and device-wide power management circuitry	—	2.375	2.5	2.625	V
V_I	DC Input voltage	—	–0.5	—	3.6	V
V_O	Output voltage	—	0	—	V_{CCIO}	V
V_{CCA}	Supplies power to the transceiver PMA regulator	—	2.375	2.5	2.625	V
V_{CCL_GXB}	Supplies power to the transceiver PMA TX, PMA RX, and clocking	—	1.045	1.1	1.155	V
V_{CCH_GXB}	Supplies power to the transceiver PMA output (TX) buffer	—	1.425	1.5	1.575	V
T_J	Operating junction temperature	Commercial	0	—	85	C
		Industrial	–40	—	100	C
t_{RAMP}	Power Supply Ramp time	Normal POR	0.05	—	100	ms
		Fast POR	0.05	—	4	ms

Note to Table 1–4:

(1) V_{CCIO} for 3C and 8C I/O banks where the configuration pins reside only supports 3.3-, 3.0-, 2.5-, or 1.8-V voltage levels.

(2) Altera recommends a 3.0-V nominal battery voltage when connecting V_{CCBAT} to a battery for volatile key backup. If you do not use the volatile security key, you may connect the V_{CCBAT} to either GND or a 3.0-V power supply.

DC Characteristics

This section lists the supply current, I/O pin leakage current, on-chip termination (OCT) accuracy and variation, input pin capacitance, internal weak pull-up and pull-down resistance, hot socketing, and Schmitt trigger input specifications.

Supply Current

Standby current is the current the device draws after the device is configured with no inputs or outputs toggling and no activity in the device. Since these currents vary largely with resources used, use the Excel-based Early Power Estimator (EPE) to get supply current estimates for your design.

I/O Pin Leakage Current

Table 1–5 defines the Arria II GX I/O pin leakage current specifications.

Table 1–5. Arria II GX I/O Pin Leakage Current

Symbol	Description	Conditions	Min	Typ	Max	Unit
I_I	Input pin	V_I = 0 V to $V_{CCIOMAX}$	–10	—	10	µA
I_{OZ}	Tri-stated I/O pin	V_O = 0 V to $V_{CCIOMAX}$	–10	—	10	µA

OCT Specifications

Table 1–6 lists the Arria II GX series OCT with and without calibration accuracy.

Table 1–6. OCT With and Without Calibration Specification for I/Os *(Note 1)*

Symbol	Description	Conditions	Calibration Accuracy	Unit
			Commercial	
25-Ω R_S 3.0/2.5	25-Ωseries OCT without calibration	V_{CCIO} = 3.0/2.5 V	± 30	%
50-Ω R_S 3.0/2.5	50-Ωseries COT without calibration	V_{CCIO} = 3.0/2.5 V	± 30	%
25-Ω R_S 1.8	25-Ωseries OCT without calibration	V_{CCIO} = 1.8 V	± 40	%
50-Ω R_S 1.8	50-Ωseries OCT without calibration	V_{CCIO} = 1.8 V	± 40	%
25-Ω R_S 1.5/1.2	25-Ωseries OCT without calibration	V_{CCIO} = 1.5/1.2 V	± 50	%
50-Ω R_S 1.5/1.2	50-Ωseries OCT without calibration	V_{CCIO} = 1.5/1.2 V	± 50	%
25-Ω R_S 3.0/2.5/1.8/ 1.5/1.2	25-Ω series OCT with calibration	V_{CCIO} = 3.0/2.5/1.8/ 1.5/1.2 V	± 10	%
50-Ω R_S 3.0/2.5/1.8/ 1.5/1.2	50-Ω series OCT with calibration	V_{CCIO} = 3.0/2.5/1.8/ 1.5/1.2 V	± 10	%

Note to Table 1–6:

(1) OCT with calibration accuracy is valid at the time of calibration only.

OCT calibration is automatically performed at power-up for OCT-enabled I/Os. When voltage and temperature conditions change after calibration, the resistance may change. Use Equation 1–1 to determine the OCT variation when voltage and temperature vary after power-up calibration.

Equation 1–1. OCT Variation *(Note 1)*

$$R_{OCT} = R_{CAL}\left(1 + \frac{dR}{dT} \times \Delta T + \frac{dR}{dV} \times \Delta V\right)$$

Note to Equation 1–1:

(1) R_{CAL} is calibrated OCT at power up. ΔT and ΔV are variations in temperature and voltage with respect to temperature and V_{CCIO} values, respectively, at power up.

Table 1–7 lists OCT variation with temperature and voltage after power-up calibration.

Table 1–7. OCT Variation after Power-up Calibration

Nominal Voltage	dR/dT (%ΔΩ/°C)	dR/dT(%ΔΩ/mV)
3.0	0.262	–0.026
2.5	0.234	–0.039
1.8	0.219	–0.086
1.5	0.199	–0.136
1.2	0.161	–0.288

Pin Capacitance

Table 1–8 shows the Arria II GX device family pin capacitance.

Table 1–8. Arria II GX Device Capacitance

Symbol	Description	Typical	Unit
C_{IODIFF}	Input capacitance on dual-purpose differential I/O pins	7.5	pF
C_{IOCLK}	Input capacitance on dual-purpose clock output/feedback pins and dedicated clock input pins	7	pF
C_{IOOCT}	Input capacitance on dual-purpose R_{UP} and R_{DN} pins	7	pF

Internal Weak Pull-Up and Weak Pull-Down Resistors

Table 1–9 lists the Arria II GX devices weak pull-up and pull-down resistor values.

Table 1–9. Arria II GX Internal Weak Pull-up and Weak Pull-Down Resistors (Part 1 of 2) *(Note 1)*

Symbol	Description	Conditions	Min.	Typ.	Max.	Unit
R_{PU}	Value of I/O pin pull-up resistor before and during configuration, as well as user mode if the programmable pull-up resistor option is enabled.	V_{CCIO} = 3.3 V ±5% *(2)*, *(3)*	7	25	41	kΩ
		V_{CCIO} = 3.0 V ±5% *(2)*, *(3)*	7	28	47	kΩ
		V_{CCIO} = 2.5 V ±5% *(2)*, *(3)*	8	35	61	kΩ
		V_{CCIO} = 1.8 V ±5% *(2)*, *(3)*	10	57	108	kΩ
		V_{CCIO} = 1.5 V ±5% *(2)*, *(3)*	13	82	163	kΩ
		V_{CCIO} = 1.2 V ±5% *(2)*, *(3)*	19	143	351	kΩ

Table 1–9. Arria II GX Internal Weak Pull-up and Weak Pull-Down Resistors (Part 2 of 2) *(Note 1)*

Symbol	Description	Conditions	Min.	Typ.	Max.	Unit
R_{PD}	Value of TCK pin pull-down resistor	V_{CCIO} = 3.3 V ±5% *(4)*	6	19	29	kΩ
		V_{CCIO} = 3.0 V ±5% *(4)*	6	22	32	kΩ
		V_{CCIO} = 2.5 V ±5% *(4)*	6	25	42	kΩ
		V_{CCIO} = 1.8 V ±5% *(4)*	7	35	70	kΩ
		V_{CCIO} = 1.5 V ±5% *(4)*	8	50	112	kΩ

Notes to Table 1–9:

(1) All I/O pins have an option to enable weak pull-up except configuration, test, and JTAG pins. The weak pull-down feature is only available for JTAG TCK.

(2) Pin pull-up resistance values may be lower if an external source drives the pin higher than V_{CCIO}.

(3) $R_{PU} = (V_{CCIO} - V_I)/IR_{PU}$. Minimum condition: -40°C; V_{CCIO} = VCC + 5%, V_I = VCC + 5% - 50 mV. Typical condition: 25°C; V_{CCIO} = VCC, V_I = 0 V. Maximum condition: 100°C; V_{CCIO} = VCC - 5%, V_I = 0 V; in which V_I refers to the voltage input at the I/O pin.

(4) $R_{PD} = V_I/I_{RPD}$. Minimum condition: -40°C; V_{CCIO} = VCC + 5%, V_I = 50 mV. Typical condition: 25°C; V_{CCIO} = VCC, V_I = VCC - 5%. Maximum condition: 100°C; V_{CCIO} = VCC - 5%, V_I = VCC - 5%; in which V_I refers to the voltage input at the I/O pin.

Hot Socketing

Table 1–10 defines the hot socketing specification for Arria II GX devices.

Table 1–10. Arria II GX Hot Socketing Specifications

Symbol	Description	Maximum
$I_{IIOPIN(DC)}$	DC current per I/O pin	300 µA
$I_{IOPIN(AC)}$	AC current per I/O pin	8 mA *(1)*
$I_{XCVRTX(DC)}$	DC current per transceiver TX pin	100 mA
$I_{XCVRRX(DC)}$	DC current per transceiver RX pin	50 mA

Notes to Table 1–10:

(1) The I/O ramp rate is 10 ns or more. For ramp rates faster than 10 ns, |IIOPIN| = C dv/dt, in which “C” is I/O pin capacitance and “dv/dt” is slew rate.

Schmitt Trigger Input

The Arria II GX device supports Schmitt trigger input on `TDI`, `TMS`, `TCK`, `nSTATUS`, `nCONFIG`, `nCE`, `CONF_DONE`, and `DCLK` pins. A Schmitt trigger introduces hysteresis to the input signal for improved noise immunity, especially for signals with slow edge rates. Table 1–11 lists the hysteresis specifications across the supported V_{CCIO} range for Schmitt trigger inputs in Arria II GX devices.

Table 1–11. Arria II GX Schmitt Trigger Input Hysteresis Specifications

Symbol	Description	Condition	Minimum	Unit
$V_{Schmitt}$	Hysteresis for Schmitt trigger input	V_{CCIO} = 3.3 V	220	mV
		V_{CCIO} = 2.5 V	180	mV
		V_{CCIO} = 1.8 V	110	mV
		V_{CCIO} = 1.5 V	70	mV

I/O Standard Specifications

Table 1–12 through Table 1–17 list input voltage (V_{IH} and V_{IL}), output voltage (V_{OH} and V_{OL}), and current drive characteristics (I_{OH} and I_{OL}) for various I/O standards supported by Arria II GX devices. They also show the Arria II GX device family I/O standard specifications. V_{OL} and V_{OH} values are valid at the corresponding I_{OH} and I_{OL}, respectively.

For an explanation of terms used in Table 1–12 through Table 1–17, refer to "Glossary" on page 1–30.

Table 1–12 lists the Arria II GX single-ended I/O standards.

Table 1–12. Single-Ended I/O Standards

I/O Standard	V_{CCIO} (V)			V_{IL} (V)		V_{IH} (V)		V_{OL} (V)	V_{OH} (V)	I_{OL} (mA)	I_{OH} (mA)
	Min	Typ	Max	Min	Max	Min	Max	Max	Min		
3.3 V LVTTL	3.135	3.3	3.465	-0.3	0.8	1.7	V_{CCIO} + 0.3	0.4	2.4	2	-2
3.3 V LVCMOS	3.135	3.3	3.465	-0.3	0.8	1.7	V_{CCIO} + 0.3	0.2	V_{CCIO} - 0.2	0.1	-0.1
3.0 V LVTTL	2.85	3	3.15	-0.3	0.8	1.7	V_{CCIO} + 0.3	0.4	2.4	2	-2
3.0 V LVCMOS	2.85	3	3.15	-0.3	0.8	1.7	V_{CCIO} + 0.3	0.2	V_{CCIO} - 0.2	0.1	-0.1
2.5 V LVCMOS	2.375	2.5	2.625	-0.3	0.7	1.7	V_{CCIO} + 0.3	0.4	2	1	-1
1.8 V LVCMOS	1.71	1.8	1.89	-0.3	0.35 × V_{CCIO}	0.65 × V_{CCIO}	V_{CCIO} + 0.3	0.45	V_{CCIO} - 0.45	2	-2
1.5 V LVCMOS	1.425	1.5	1.575	-0.3	0.35 × V_{CCIO}	0.65 × V_{CCIO}	V_{CCIO} + 0.3	0.25 * V_{CCIO}	0.75 × V_{CCIO}	2	-2
1.2 V LVCMOS	1.14	1.2	1.26	-0.3	0.35 × V_{CCIO}	0.65 × V_{CCIO}	V_{CCIO} + 0.3	0.25 * V_{CCIO}	0.75 × V_{CCIO}	2	-2
3.0-V PCI	2.85	3	3.15	—	0.3 × V_{CCIO}	0.5 × V_{CCIO}	3.6	0.1 × V_{CCIO}	0.9 × V_{CCIO}	1.5	-0.5
3.0-V PCI-X	2.85	3	3.15	—	0.35 × V_{CCIO}	0.5 × V_{CCIO}	—	0.1 × V_{CCIO}	0.9 × V_{CCIO}	1.5	-0.5

Table 1–13 lists the Arria II GX single-ended SSTL and HSTL I/O reference voltage specifications.

Table 1–13. Single-Ended SSTL and HSTL I/O Reference Voltage Specifications (Part 1 of 2)

I/O Standard	V_{CCIO} (V)			V_{REF} (V)			V_{TT} (V)		
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
SSTL-2 Class I, II	2.375	2.5	2.625	0.49 × V_{CCIO}	0.5 × V_{CCIO}	0.51 × V_{CCIO}	V_{REF} - 0.04	V_{REF}	V_{REF} + 0.04
SSTL-18 Class I, II	1.71	1.8	1.89	0.49 × V_{CCIO}	0.5 × V_{CCIO}	0.51 × V_{CCIO}	V_{REF} - 0.04	V_{REF}	V_{REF} + 0.04

Table 1–13. Single-Ended SSTL and HSTL I/O Reference Voltage Specifications (Part 2 of 2)

I/O Standard	V_{CCIO} (V)			V_{REF} (V)			V_{TT} (V)		
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
SSTL-15 Class I, II	1.425	1.5	1.575	0.49 × V_{CCIO}	0.5 × V_{CCIO}	0.51 × V_{CCIO}	V_{REF} - 0.04	V_{REF}	V_{REF} + 0.04
HSTL-18 Class I, II	1.71	1.8	1.89	0.85	0.9	0.95	—	V_{CCIO}/2	—
HSTL-15 Class I, II	1.425	1.5	1.575	0.68	0.75	0.9	—	V_{CCIO}/2	—
HSTL-12 Class I, II	1.14	1.2	1.26	0.48 × V_{CCIO}	0.5 × V_{CCIO}	0.52 × V_{CCIO}	—	V_{CCIO}/2	—

Table 1–14 lists the Arria II GX single-ended SSTL and HSTL I/O standard signal specifications.

Table 1–14. Single-Ended SSTL and HSTL I/O Standard Signal Specifications

I/O Standard	$V_{IL(DC)}$ (V)		$V_{IH(DC)}$ (V)		$V_{IL(AC)}$ (V)	$V_{IH(AC)}$ (V)	V_{OL} (V)	V_{OH} (V)	I_{ol} (mA)	I_{oh} (mA)
	Min	Max	Min	Max	Max	Min	Max	Min		
SSTL-2 Class I	-0.3	V_{REF} - 0.15	V_{REF} + 0.15	V_{CCIO} + 0.3	V_{REF} - 0.31	V_{REF} + 0.31	V_{TT} - 0.57	V_{TT} + 0.57	8.1	-8.1
SSTL-2 Class II	-0.3	V_{REF} - 0.15	V_{REF} + 0.15	V_{CCIO} + 0.3	V_{REF} - 0.31	V_{REF} + 0.31	V_{TT} - 0.76	V_{TT} + 0.76	16.2	-16.2
SSTL-18 Class I	-0.3	V_{REF} - 0.125	V_{REF} + 0.125	V_{CCIO} + 0.3	V_{REF} -0.25	V_{REF} + 0.25	V_{TT} - 0.475	V_{TT} + 0.475	6.7	-6.7
SSTL-18 Class II	-0.3	V_{REF} - 0.125	V_{REF} + 0.125	V_{CCIO} + 0.3	V_{REF} -0.25	V_{REF} + 0.25	0.28	V_{CCIO} - 0.28	13.4	-13.4
SSTL-15 Class I	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	V_{CCIO} + 0.3	V_{REF} - 0.175	V_{REF} + 0.175	0.2 × V_{CCIO}	0.8 × V_{CCIO}	8	-8
SSTL-15 Class II	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	V_{CCIO} + 0.3	V_{REF} - 0.175	V_{REF} + 0.175	0.2 × V_{CCIO}	0.8 × V_{CCIO}	16	-16
HSTL-18 Class I	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	V_{CCIO} + 0.3	V_{REF} -0.2	V_{REF} + 0.2	0.4	V_{CCIO} - 0.4	8	-8
HSTL-18 Class II	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	V_{CCIO} + 0.3	V_{REF} -0.2	V_{REF} + 0.2	0.4	V_{CCIO} - 0.4	16	-16
HSTL-15 Class I	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	V_{CCIO} + 0.3	V_{REF} -0.2	V_{REF} + 0.2	0.4	V_{CCIO} - 0.4	8	-8
HSTL-15 Class II	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	V_{CCIO} + 0.3	V_{REF} -0.2	V_{REF} + 0.2	0.4	V_{CCIO} - 0.4	16	-16
HSTL-12 Class I	-0.15	V_{REF} - 0.08	V_{REF} + 0.08	V_{CCIO} + 0.15	V_{REF} -0.15	V_{REF} + 0.15	0.25 × V_{CCIO}	0.75 × V_{CCIO}	8	-8
HSTL-12 Class II	-0.15	V_{REF} - 0.08	V_{REF} + 0.08	V_{CCIO} + 0.15	V_{REF} -0.15	V_{REF} + 0.15	0.25 × V_{CCIO}	0.75 × V_{CCIO}	16	-16

Table 1–15 lists the Arria II GX differential SSTL I/O standards.

Table 1–15. Differential SSTL I/O Standards

I/O Standard	V_{CCIO} (V)			$V_{SWING(DC)}$ (V)		$V_{X(AC)}$ (V)			$V_{SWING(AC)}$ (V)		$V_{OX(AC)}$ (V)		
	Min	Typ	Max	Min	Max	Min	Typ	Max	Min	Max	Min	Typ	Max
SSTL-2 Class I, II	2.375	2.5	2.625	0.36	V_{CCIO}	$V_{CCIO}/2 - 0.2$	—	$V_{CCIO}/2 + 0.2$	0.7	V_{CCIO}	*(1)*	—	*(1)*
SSTL-18 Class I, II	1.71	1.8	1.89	0.25	V_{CCIO}	$V_{CCIO}/2 - 0.175$	—	$V_{CCIO}/2 + 0.175$	0.5	V_{CCIO}	*(1)*	—	*(1)*
SSTL-15 Class I, II	1.425	1.5	1.575	0.2	—	*(1)*	—	*(1)*	0.35	—	*(1)*	—	*(1)*

Note to Table 1–15:

(1) Pending silicon characterization.

Table 1–16 lists the Arria II GX HSTL I/O standards.

Table 1–16. Differential HSTL I/O Standards

I/O Standard	V_{CCIO} (V)			$V_{DIF(DC)}$ (V)		$V_{X(AC)}$ (V)			$V_{CM(DC)}$ (V)			$V_{DIF(AC)}$ (V)	
	Min	Typ	Max	Min	Max	Min	Typ	Max	Min	Typ	Max	Min	Max
HSTL-18 Class I	1.71	1.8	1.89	0.2	—	0.85	—	0.95	0.88	—	0.95	0.4	—
HSTL-15 Class I, II	1.425	1.5	1.575	0.2	—	0.71	—	0.79	0.71	—	0.79	0.4	—
HSTL-12 Class I, II	1.14	1.2	1.26	0.16	—	—	0.5 × V_{CCIO}	—	0.48 × V_{CCIO}	0.5 × V_{CCIO}	0.52 × V_{CCIO}	0.3	—

Table 1–17 lists the Arria II GX differential I/O standard specifications.

Table 1–17. Differential I/O Standard Specifications

I/O Standard	V_{CCIO} (V)			V_{TH} (mV)			V_{ICM} (V) *(4)*			V_{OD} (V) *(1)*			V_{OS} (V)		
	Min	Typ	Max	Min	Cond.	Max	Min	Cond.	Max	Min	Typ	Max	Min	Typ	Max
2.5V LVDS	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.05	D_{max} <= 700 Mbps	1.80	0.247	—	0.6	1.125	1.25	1.375
						—	1.05	D_{max} > 700 Mbps	1.55						
RSDS *(3)*	2.375	2.5	2.625	—	—	—	—	—	—	0.1	0.2	0.6	0.5	1.2	1.4
Mini-LVDS *(3)*	2.375	2.5	2.625	—	—	—	—	—	—	0.25	—	0.6	1	1.2	1.4
LVPECL *(2)*	2.375	2.5	2.625	300	—	—	0.6	D_{max} <= 700 Mbps	1.8	—	—	—	—	—	—
							1.0	D_{max} > 700 Mbps	1.6						

Notes to Table 1–17*:*

(1) R_L range: 90 <= RL <= 110 Ω.

(2) LVPECL input standard is supported at the dedicated clock input pins (GCLK) only.

(3) RSDS and mini-LVDS I/O standards are only supported for differential outputs.

(4) V_{IN} range: 0 <= V_{IN} <= 1.85 V.

Power Consumption for Arria II GX Devices

Altera® offers two ways to estimate power for a design: the Excel-based Early Power Estimator and the Quartus® II PowerPlay Power Analyzer feature.

The interactive Excel-based Early Power Estimator is typically used prior to designing the FPGA in order to get a magnitude estimate of the device power. The Quartus II PowerPlay Power Analyzer provides better quality estimates based on the specifics of the design after place-and-route is complete. The PowerPlay Power Analyzer can apply a combination of user-entered, simulation-derived, and estimated signal activities which, when combined with detailed circuit models, can yield very accurate power estimates.

For more information about power estimation tools, refer to the *Early Power Estimator User Guide* and the *PowerPlay Power Analysis* chapter in the *Quartus II Handbook*.

Switching Characteristics

This section provides performance characteristics of the Arria II GX core and periphery blocks for commercial grade devices.

These characteristics can be designated as Preliminary and Final. Preliminary characteristics are created using simulation results, process data, and other known parameters. Final characteristics are based on actual silicon characterization and testing. These numbers reflect the actual performance of the device under worst-case silicon process, voltage, and junction temperature conditions.

The table title shows the designations as “Preliminary” or “Final” for each table.

Transceiver Performance Specifications

Table 1–18 lists the Arria II GX transceiver specifications.

Table 1–18. Arria II GX Transceiver Specification—Preliminary (Part 1 of 4)

Symbol/ Description	Conditions	C4			C5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Reference Clock											
Input frequency from REFCLK input pins	—	50	—	622.08	50	—	622.08	50	—	622.08	MHz
Input frequency from PLD input	—	50	—	200	50	—	200	50	—	200	MHz
Absolute V_{MAX} for a REFCLK pin	—	—	—	2.2	—	—	2.2	—	—	2.2	V
Absolute V_{MIN} for a REFCLK pin	—	-0.3	—	—	-0.3	—	—	-0.3	—	—	V
Rise/fall time	—	—	—	0.2	—	—	0.2	—	—	0.2	UI
Duty cycle	—	45	—	55	45	—	55	45	—	55	%
Peak-to-peak differential input voltage	—	200	—	2000	200	—	2000	200	—	2000	mV

Table 1–18. Arria II GX Transceiver Specification—Preliminary (Part 2 of 4)

Symbol/ Description	Conditions	C4			C5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Spread-spectrum modulating clock frequency	PCI Express	30	—	33	30	—	33	30	—	33	kHz
Spread-spectrum downspread	PCI Express	—	0 to -0.5%	—	—	0 to -0.5%	—	—	0 to -0.5%	—	
On-chip termination resistors	—	—	100	—	—	100	—	—	100	—	Ω
V_{ICM} (AC coupled)	—	—	1200	—	—	1200	—	—	1200	—	mV
V_{ICM} (DC coupled)	HCSL I/O standard for PCI Express reference clock	250	—	550	250	—	550	250	—	550	mV
R_{ref}	— —	—	2000 ± 1%	—	—	2000 ± 1%	—	—	2000 ± 1%	—	Ω
Transceiver Clocks											
Calibration block clock frequency	—	10	—	125	10	—	125	10	—	125	MHz
`fixedclk` clock frequency	PCI Express Receiver Detect	—	125	—	—	125	—	—	125	—	MHz
Transceiver block minimum power-down pulse width	—	—	1	—	—	1	—	—	1	—	µs
Receiver											
Data rate	—	600	—	3750	600	—	3125	600	—	3125	Mbps
Absolute V_{MAX} for a receiver pin *(1)*	—	—	—	1.5	—	—	1.5	—	—	1.5	V
Absolute V_{MIN} for a receiver pin	—	-0.4	—	—	-0.4	—	—	-0.4	—	—	V
Maximum peak-to-peak differential input voltage V_{ID} (diff p-p)	V_{ICM} = 0.82 V setting	—	—	2.7	—	—	2.7	—	—	2.7	V
	V_{ICM} =1.1 V setting *(7)*	—	—	1.6	—	—	1.6	—	—	1.6	V
Minimum peak-to-peak differential input voltage V_{ID} (diff p-p)	Data Rate = 600 Mbps to 3.75 Gbps.	100	—	—	100	—	—	100	—	—	mV
V_{ICM}	V_{ICM} = 0.82 V setting	—	820	—	—	820	—	—	820	—	mV
	V_{ICM} =1.1 V setting *(7)*	—	1100	—	—	1100	—	—	1100	—	mV

Table 1–18. Arria II GX Transceiver Specification—Preliminary (Part 2 of 4)

Symbol/ Description	Conditions	C4			C5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Spread-spectrum modulating clock frequency	PCI Express	30	—	33	30	—	33	30	—	33	kHz
Spread-spectrum downspread	PCI Express	—	0 to -0.5%	—	—	0 to -0.5%	—	—	0 to -0.5%	—	
On-chip termination resistors	—	—	100	—	—	100	—	—	100	—	Ω
V_{ICM} (AC coupled)	—	—	1200	—	—	1200	—	—	1200	—	mV
V_{ICM} (DC coupled)	HCSL I/O standard for PCI Express reference clock	250	—	550	250	—	550	250	—	550	mV
R_{ref}	— —	—	2000 ± 1%	—	—	2000 ± 1%	—	—	2000 ± 1%	—	Ω
Transceiver Clocks											
Calibration block clock frequency	—	10	—	125	10	—	125	10	—	125	MHz
`fixedclk` clock frequency	PCI Express Receiver Detect	—	125	—	—	125	—	—	125	—	MHz
Transceiver block minimum power-down pulse width	—	—	1	—	—	1	—	—	1	—	µs
Receiver											
Data rate	—	600	—	3750	600	—	3125	600	—	3125	Mbps
Absolute V_{MAX} for a receiver pin (1)	—	—	—	1.5	—	—	1.5	—	—	1.5	V
Absolute V_{MIN} for a receiver pin	—	-0.4	—	—	-0.4	—	—	-0.4	—	—	V
Maximum peak-to-peak differential input voltage V_{ID} (diff p-p)	V_{ICM} = 0.82 V setting	—	—	2.7	—	—	2.7	—	—	2.7	V
	V_{ICM} =1.1 V setting (7)	—	—	1.6	—	—	1.6	—	—	1.6	V
Minimum peak-to-peak differential input voltage V_{ID} (diff p-p)	Data Rate = 600 Mbps to 3.75 Gbps.	100	—	—	100	—	—	100	—	—	mV
V_{ICM}	V_{ICM} = 0.82 V setting	—	820	—	—	820	—	—	820	—	mV
	V_{ICM} =1.1 V setting (7)	—	1100	—	—	1100	—	—	1100	—	mV

Table 1–18. Arria II GX Transceiver Specification—Preliminary (Part 3 of 4)

Symbol/ Description	Conditions	C4			C5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Differential on-chip termination resistors	100-Ω setting	—	100	—	—	100	—	—	100	—	Ω
Return loss differential mode	PCI Express	50 MHz to 1.25 GHz: -10dB									
	XAUI	100 MHz to 2.5 GHz: -10dB									
Return loss common mode	PCI Express	50 MHz to 1.25 GHz: -6dB									
	XAUI	100 MHz to 2.5 GHz: -6dB									
Programmable PPM detector *(2)*	—	± 62.5, 100, 125, 200, 250, 300, 500, 1000									ppm
Run length	—	—	80	—	—	80	—	—	80	—	UI
Programmable equalization	—	—	—	7	—	—	7	—	—	7	dB
Signal detect/loss threshold	PCI Express (PIPE) Mode	65	—	175	65	—	175	65	—	175	mV
CDR LTR time *(3)*	—	—	—	75	—	—	75	—	—	75	µs
CDR minimum T1b *(4)*	—	15	—		15	—	—	15	—	—	µs
LTD lock time *(5)*	—	0	100	4000	0	100	4000	0	100	4000	ns
Data lock time from `rx_freqlocked` *(6)*	—	—	—	4000	—	—	4000	—	—	4000	ns
Programmable DC gain	DC Gain Setting = 0	—	0	—	—	0	—	—	0	—	dB
	DC Gain Setting = 1	—	3	—	—	3	—	—	3	—	dB
	DC Gain Setting = 2	—	6	—	—	6	—	—	6	—	dB
Transmitter											
Data rate		—	—	—	—	—	—	—	—	—	Mbps
V_{OCM}	0.65 V setting	—	650	—	—	650	—	—	650	—	mV
Differential on-chip termination resistors	100-Ω setting	—	100	—	—	100	—	—	100	—	Ω
Return loss differential mode	PCI Express	50 MHz to 1.25 GHz: -10dB									
	XAUI	312 MHz to 625 MHz: -10dB 625 MHz to 3.125 GHz: -10dB/decade slope									
Return loss common mode	PCI Express	50 MHz to 1.25 GHz: -6dB									
Rise time	—	50	—	200	50	—	200	50	—	200	ps
Fall time	—	50	—	200	50	—	200	50	—	200	ps
Intra-differential pair skew	—	—	—	15	—	—	15	—	—	15	ps

Table 1–18. Arria II GX Transceiver Specification—Preliminary (Part 4 of 4)

Symbol/ Description	Conditions	C4			C5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Intra-transceiver block skew	PCI Express (PIPE) ×4	—	—	120	—	—	120	—	—	120	ps
Inter-transceiver block skew	PCI Express (PIPE) ×8	—	—	300	—	—	300	—	—	300	ps
CMU PLL0 and CMU PLL1											
CMU PLL lock time from `CMUPLL_reset` deassertion	—	—	—	100	—	—	100	—	—	100	µs
PLD-Transceiver Interface											
Interface speed	—	25	—	200	25	—	160	25	—	130	MHz
Digital reset pulse width	—	Minimum is 2 parallel clock cycles									

Notes to Table 1–18:

(1) The device cannot tolerate prolonged operation at this absolute maximum.

(2) The rate matcher supports only up to +/-300 parts per million (ppm).

(3) Time taken to `rx_pll_locked` goes high from `rx_analogreset` deassertion. Refer to Figure 1–1.

(4) Time for which the CDR must be kept in lock-to-reference mode after `rx_pll_locked` goes high and before `rx_locktodata` is asserted in manual mode. Refer to Figure 1–1.

(5) Time taken to recover valid data after the `rx_locktodata` signal is asserted in manual mode. Refer to Figure 1–1.

(6) Time taken to recover valid data after the `rx_freqlocked` signal goes high in automatic mode. Refer to Figure 1–2.

(7) The 1.1-V RX V_{ICM} setting must be used if the input serial data standard is LVDS and the link is DC-coupled.

Figure 1–1 shows the lock time parameters in manual mode.

LTD = lock-to-data. LTR = lock-to-reference.

Figure 1–1. Lock Time Parameters for Manual Mode

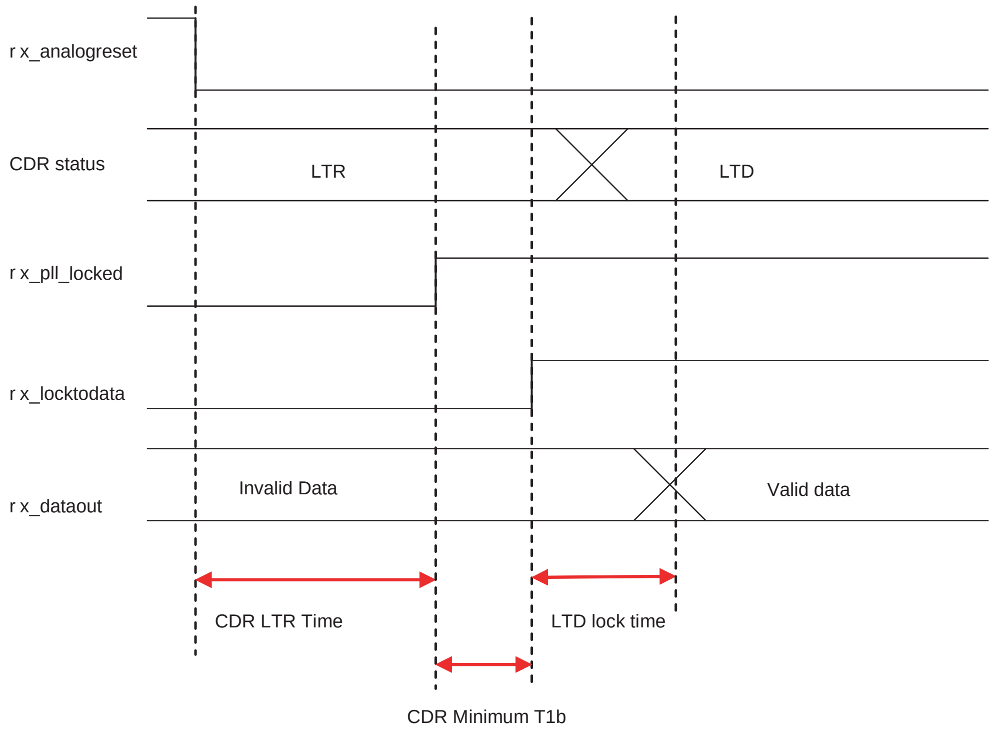


Figure 1–2 shows the lock time parameters in automatic mode.

Figure 1–2. Lock Time Parameters for Automatic Mode

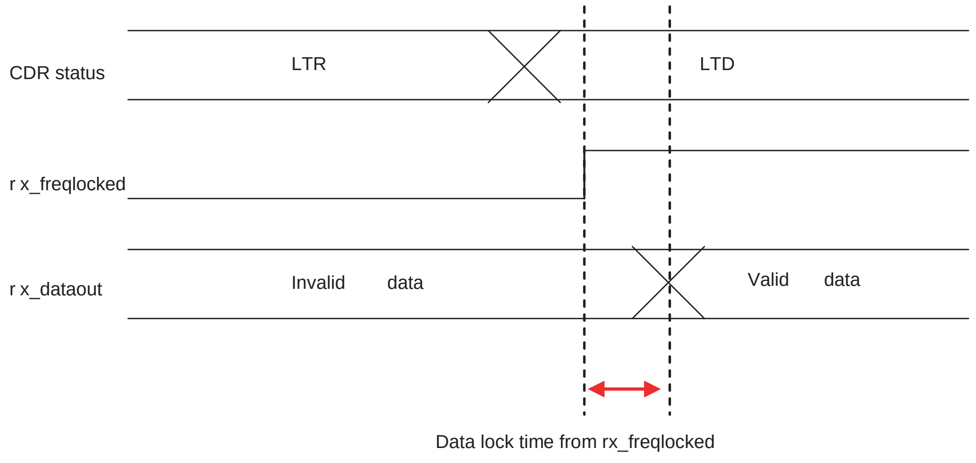


Figure 1–3 shows the differential receiver input waveform.

Figure 1–3. Receiver Input Waveform

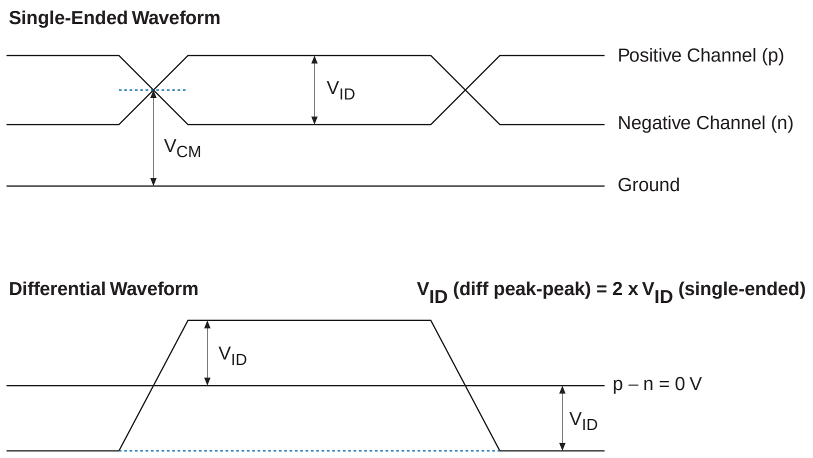


Figure 1–4 shows the transmitter output waveform.

Figure 1–4. Transmitter Output Waveform—Preliminary

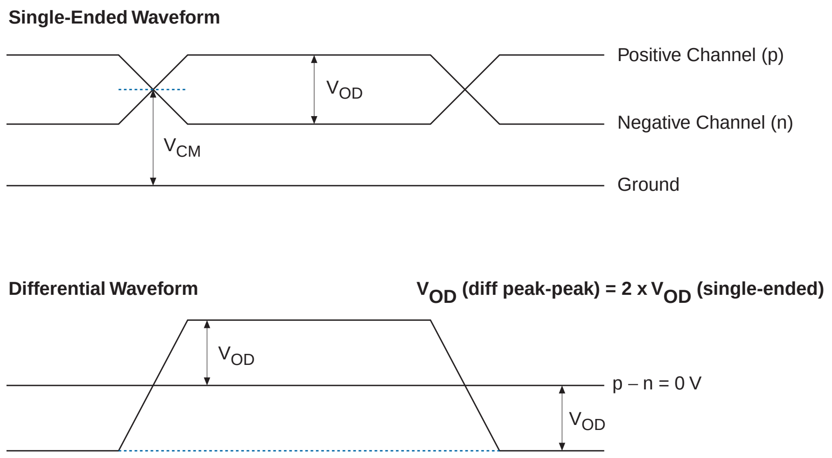


Table 1–19 shows the typical V_{OD} for TX term that equals 100 Ω .

Table 1–19. Typical V_{OD} Setting, TX Term = 100 Ω—Preliminary

Symbol	V_{OD} Setting (mV)							
	200	400	600	700	800	900	1000	1200
V_{OD} Typical (mV)	200	400	600	700	800	900	1000	1200

Table 1–20 shows the typical V_{OD} for TX term that equals 150 Ω .

Table 1–20. Typical V_{OD} Setting, TX Term = 150 Ω—Preliminary

Symbol	V_{OD} Setting (mV)				
	300	600	900	1050	1200
V_{OD} Typical (mV)	300	600	900	1050	1200

Table 1–21 shows the Arria II GX transceiver block AC specifications.

Table 1–21. Arria II GX Transceiver Block AC Specification—Preliminary *(Note 1), (2)* (Part 1 of 6)

Symbol/ Description	Conditions	C4 Min	C4 Typ	C4 Max	C5 Min	C5 Typ	C5 Max	C6 Min	C6 Typ	C6 Max	Unit
SONET/SDH Receiver Jitter Tolerance											
Jitter tolerance at 622.08 Mbps	Jitter frequency = 0.03 KHz Pattern = PRBS23	> 15			> 15			> 15			UI
	Jitter frequency = 25 KHZ Pattern = PRBS23	> 1.5			> 1.5			> 1.5			UI
	Jitter frequency = 250 KHz Pattern = PRBS23	> 0.15			> 0.15			> 0.15			UI
Jitter tolerance at 2488.32 Mbps	Jitter frequency = 0.06 KHz Pattern = PRBS23	> 15			> 15			> 15			UI
	Jitter frequency = 100 KHZ Pattern = PRBS23	> 1.5			> 1.5			> 1.5			UI
	Jitter frequency = 1 MHz Pattern = PRBS23	> 0.15			> 0.15			> 0.15			UI
	Jitter frequency = 10 MHz Pattern = PRBS23	> 0.15			> 0.15			> 0.15			UI
Fibre Channel Transmit Jitter Generation *(3)*, *(10)*											
Total jitter FC-1	Pattern = CRPAT	—	—	0.23	—	—	0.23	—	—	0.23	UI
Deterministic jitter FC-1	Pattern = CRPAT	—	—	0.11	—	—	0.11	—	—	0.11	UI
Total jitter FC-2	Pattern = CRPAT	—	—	0.33	—	—	0.33	—	—	0.33	UI
Deterministic jitter FC-2	Pattern = CRPAT	—	—	0.2	—	—	0.2	—	—	0.2	UI
Fibre Channel Receiver Jitter Tolerance *(3)*, *(11)*											
Deterministic jitter FC-1	Pattern = CJTPAT	> 0.37			> 0.37			> 0.37			UI
Random jitter FC-1	Pattern = CJTPAT	> 0.31			> 0.31			> 0.31			UI

Table 1–21. Arria II GX Transceiver Block AC Specification—Preliminary *(Note 1), (2)* (Part 2 of 6)

Symbol/ Description	Conditions	C4			C5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Sinusoidal jitter FC-1	Fc/25000		> 1.5			> 1.5			> 1.5		UI
	Fc/1667		> 0.1			> 0.1			> 0.1		UI
Deterministic jitter FC-2	Pattern = CJTPAT		> 0.33			> 0.33			> 0.33		UI
Random jitter FC-2	Pattern = CJTPAT		> 0.29			> 0.29			> 0.29		UI
Sinusoidal jitter FC-2	Fc/25000		> 1.5			> 1.5			> 1.5		UI
	Fc/1667		> 0.1			> 0.1			> 0.1		UI
XAUI Transmit Jitter Generation *(4)*											
Total jitter at 3.125 Gbps	Pattern = CJPAT	—	—	0.3	—	—	0.3	—	—	0.3	UI
Deterministic jitter at 3.125 Gbps	Pattern = CJPAT	—	—	0.17	—	—	0.17	—	—	0.17	UI
XAUI Receiver Jitter Tolerance *(4)*											
Total jitter			> 0.65			> 0.65			> 0.65		UI
Deterministic jitter			> 0.37			> 0.37			> 0.37		UI
Peak-to-peak jitter	Jitter frequency = 22.1 KHz		> 8.5			> 8.5			> 8.5		UI
Peak-to-peak jitter	Jitter frequency = 1.875 MHz		> 0.1			> 0.1			> 0.1		UI
Peak-to-peak jitter	Jitter frequency = 20 MHz		> 0.1			> 0.1			> 0.1		UI
PCI Express Transmit Jitter Generation *(5)*											
Total jitter at 2.5 Gbps (Gen1)	Compliance pattern	—	—	0.25	—	—	0.25	—	—	0.25	UI
PCI Express Receiver Jitter Tolerance *(5)*											
Total jitter at 2.5 Gbps (Gen1)	Compliance pattern		> 0.6			> 0.6			> 0.6		UI
Serial RapidIO Transmit Jitter Generation *(6)*											
Deterministic jitter (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	—	—	0.17	—	—	0.17	—	—	0.17	UI
Total jitter (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	—	—	0.35	—	—	0.35	—	—	0.35	UI
Serial RapidIO Receiver Jitter Tolerance *(6)*											
Deterministic jitter tolerance (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT		> 0.37			> 0.37			> 0.37		UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT		> 0.55			> 0.55			> 0.55		UI

Table 1–21. Arria II GX Transceiver Block AC Specification—Preliminary *(Note 1), (2)* (Part 3 of 6)

Symbol/ Description	Conditions	C4			C5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 22.1 KHz Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 8.5			> 8.5			> 8.5			UI
	Jitter Frequency = 1.875 MHz Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.1			> 0.1			> 0.1			UI
	Jitter Frequency = 20 MHz Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.1			> 0.1			> 0.1			UI
GIGE Transmit Jitter Generation *(7)*											
Deterministic jitter (peak-to-peak)	Pattern = CRPAT	—	—	0.14	—	—	0.14	—	—	0.14	UI
Total jitter (peak-to-peak)	Pattern = CRPAT	—	—	0.279	—	—	0.279	—	—	0.279	UI
GIGE Receiver Jitter Tolerance *(7)*											
Deterministic jitter tolerance (peak-to-peak)	Pattern = CJPAT	> 0.4			> 0.4			> 0.4			UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Pattern = CJPAT	> 0.66			> 0.66			> 0.66			UI
HiGig Transmit Jitter Generation *(8)*											
Deterministic jitter (peak-to-peak)	Data Rate = 3.75 Gbps Pattern = CJPAT	—	—	0.17	—	—	—	—	—	—	UI
Total jitter (peak-to-peak)	Data Rate = 3.75 Gbps Pattern = CJPAT	—	—	0.35	—	—	—	—	—	—	UI
HiGig Receiver Jitter Tolerance *(8)*											
Deterministic jitter tolerance (peak-to-peak)	Data Rate = 3.75 Gbps Pattern = CJPAT	> 0.37			—	—	—	—	—	—	UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Data Rate = 3.75 Gbps Pattern = CJPAT	> 0.65			—	—	—	—	—	—	UI

Table 1–21. Arria II GX Transceiver Block AC Specification—Preliminary *(Note 1), (2)* (Part 4 of 6)

Symbol/ Description	Conditions	C4			C5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 22.1 KHz Data Rate = 3.75 Gbps Pattern = CJPAT	> 8.5			—	—	—	—	—	—	UI
	Jitter Frequency = 1.875MHz Data Rate = 3.75 Gbps Pattern = CJPAT	> 0.1			—	—	—	—	—	—	UI
	Jitter Frequency = 20 MHz Data Rate = 3.75 Gbps Pattern = CJPAT	> 0.1			—	—	—	—	—	—	UI
SDI Transmitter Jitter Generation *(9)*											
Alignment jitter (peak-to-peak)	Data Rate = 1.485 Gbps (HD) Pattern = Color Bar Low-Frequency Roll-Off = 100 KHz	0.2	—	—	0.2	—	—	0.2	—	—	UI
	Data Rate = 2.97 Gbps (3G) Pattern = Color Bar Low-Frequency Roll-Off = 100 KHz	0.3	—	—	0.3	—	—	0.3	—	—	UI

Table 1–21. Arria II GX Transceiver Block AC Specification—Preliminary *(Note 1), (2)* (Part 5 of 6)

Symbol/ Description	Conditions	C4			C5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
SDI Receiver Jitter Tolerance *(9)*											
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 15 KHz Data Rate = 2.97 Gbps (3G) Pattern = Single Line Scramble Color Bar	> 2			> 2			> 2			UI
	Jitter Frequency = 100 KHz Data Rate = 2.97 Gbps (3G) Pattern = Single Line Scramble Color Bar	> 0.3			> 0.3			> 0.3			UI
	Jitter Frequency = 148.5 MHz Data Rate = 2.97 Gbps (3G) Pattern = Single Line Scramble Color Bar	> 0.3			> 0.3			> 0.3			UI

Table 1–21. Arria II GX Transceiver Block AC Specification—Preliminary *(Note 1), (2)* (Part 6 of 6)

Symbol/ Description	Conditions	C4			C5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Sinusoidal Jitter Tolerance (peak-to-peak)	Jitter Frequency = 20 KHz Data Rate = 1.485 Gbps (HD) Pattern = 75% Color Bar	> 1			> 1			> 1			UI
	Jitter Frequency = 100 KHz Data Rate = 1.485 Gbps (HD) Pattern = 75% Color Bar	> 0.2			> 0.2			> 0.2			UI
	Jitter Frequency = 148.5 MHz Data Rate = 1.485 Gbps (HD) Pattern =75% Color Bar	> 0.2			> 0.2			> 0.2			UI

Notes to Table 1–21:

(1) Dedicated `refclk` pins were used to drive the input reference clocks.
(2) The Jitter numbers specified are valid for the stated conditions only.
(3) The jitter numbers for Fibre Channel are compliant to the FC-PI-4 Specification revision 6.10.
(4) The jitter numbers for XAUI are compliant to the IEEE802.3ae-2002 Specification.
(5) The jitter numbers for PCI Express (PIPE) are compliant to the PCIe Base Specification 2.0.
(6) The jitter numbers for Serial RapidIO are compliant to the RapidIO Specification 1.3.
(7) The jitter numbers for GIGE are compliant to the IEEE802.3-2002 Specification.
(8) The jitter numbers for HiGig are compliant to the IEEE802.3ae-2002 Specification.
(9) The HD-SDI and 3G-SDI jitter numbers are compliant to the SMPTE292M and SMPTE424M Specifications.
(10) The Fibre Channel transmitter jitter generation numbers are compliant to the specification at δ_T inter operability point.
(11) The Fibre Channel receiver jitter tolerance numbers are compliant to the specification at δ_R interpretability point.

Core Performance Specifications for Arria II GX Devices

This section describes the clock tree, phase-locked loop (PLL), digital signal processing (DSP), embedded memory, configuration, and JTAG specifications.

Clock Tree Specifications

Table 1–22 lists the clock tree specifications for Arria II GX devices.

Table 1–22. Arria II GX Clock Tree Performance—Preliminary (Part 1 of 2)

Device	Performance *(1)*			Unit
	C4	C5	C6	
EP2AGX20	500	500	400	MHz
EP2AGX30	500	500	400	MHz
EP2AGX45	500	500	400	MHz
EP2AGX65	500	500	400	MHz

Table 1–22. Arria II GX Clock Tree Performance—Preliminary (Part 2 of 2)

Device	Performance (1)			Unit
	C4	C5	C6	
EP2AGX95	500	500	400	MHz
EP2AGX125	500	500	400	MHz
EP2AGX190	500	500	400	MHz
EP2AGX260	500	500	400	MHz

Note to Table 1–22:

(1) The performance specifications are applicable to GCLK and RCLK networks.

PLL Specifications

Table 1–23 describes the PLL specifications for Arria II GX devices.

Table 1–23. Arria II GX PLL Specifications—Preliminary (Part 1 of 2)

Symbol	Description	Min	Typ	Max	Unit
f_{IN}	Input clock frequency (from clock input pins residing in top/bottom banks)	5	—	472.5 (1)	MHz
	Input clock frequency (from clock input pins residing in right banks)	5	—	500	MHz
f_{INPFD}	Input frequency to the PFD	5	—	325	MHz
f_{VCO}	PLL VCO operating Range (2)	600	—	1300	MHz
f_{INDUTY}	Input clock duty cycle	40	—	60	%
$f_{EINDUTY}$	External feedback clock input duty cycle	40	—	60	%
t_{INCCJ}	Input clock cycle to cycle jitter	—	—	(4)	ps
f_{OUT}	Output frequency for internal global or regional clock	—	—	472.5	MHz
f_{OUT_EXT}	Output frequency for external clock output	—	—	472.5 (3)	MHz
$t_{OUTDUTY}$	Duty cycle for external clock output (when set to 50%)	45	50	55	%
t_{OUTPJ_DC}	Dedicated clock output period jitter	—	—	(4)	ps
t_{OUTPJ_IO}	Regular I/O clock output period jitter	—	—	(4)	ps
t_{FCOMP}	External feedback clock compensation time	—	—	10	ns
$t_{CONFIGPLL}$	Time required to reconfigure PLL scan chains	—	(4)	—	SCANCLK cycles
$t_{CONFIGPHASE}$	Time required to reconfigure phase shift	1	—	1	SCANCLK cycles
$f_{SCANCLK}$	SCANCLK frequency	—	—	100	MHz
t_{LOCK}	Time required to lock from end of device configuration	—	—	(4)	ms
t_{DLOCK}	Time required to lock dynamically (after switchover or reconfiguring any non-post-scale counters/delays)	—	—	(4)	ms
$f_{CL\ B\ W}$	PLL closed-loop low bandwidth range	—	(4)	—	MHz
	PLL closed-loop medium bandwidth range	—	(4)	—	MHz
	PLL closed-loop high bandwidth range	—	(4)	—	MHz
t_{PLL_PSERR}	Accuracy of PLL phase shift	—	(4)	—	ps

Table 1–23. Arria II GX PLL Specifications—Preliminary (Part 2 of 2)

Symbol	Description	Min	Typ	Max	Unit
t_{ARESET}	Minimum pulse width on `areset` signal	10	—	—	ns

Notes to Table 1–23:

(1) F_{IN} is limited by I/O F_{MAX}.
(2) The VCO frequency reported by the Quartus II software in the PLL summary section of the compilation report takes into consideration the VCO post-scale counter K value. Therefore, if the counter K has a value of 2, the frequency reported can be lower than the f_{VCO} specification.
(3) This specification is limited by the lower of the two: I/O F_{max} or F_{out} of the PLL.
(4) Pending silicon characterization.

DSP Block Specifications

Table 1–24 describes the Arria II GX DSP block performance specifications.

Table 1–24. Arria II GX DSP Block Performance Specifications—Preliminary *(Note 1)*

Mode	Resources Used	Performance			Unit
	Number of Multipliers	C4	C5	C6	
9 x 9-bit multiplier	1	315	245	200	MHz
12 x 12-bit multiplier	1	315	245	200	MHz
18 x 18-bit multiplier	1	350	275	225	MHz
36 x 36-bit multiplier	1	285	220	180	MHz
18 x 18-bit Multiply Accumulator	4	315	245	200	MHz
18 × 18-bit Multiply Adder	4	315	245	200	MHz
18 × 18-bit Multiply Adder-Signed Full Precision	2	315	245	200	MHz
18 × 18-bit Multiply Adder with loopback *(2)*	2	250	195	160	MHz
36-bit Shift (32-bit data)	1	285	220	180	MHz
Double mode	1	285	220	180	MHz

Notes to Table 1–24:

(1) Maximum is for fully-pipelined block with **Round** and **Saturation** disabled.
(2) Maximum is for non-pipelined block with loopback input registers disabled and **Round** and **Saturation** disabled.

Embedded Memory Block Specifications

Table 1–25 describes the Arria II GX embedded memory block specifications.

Table 1–25. Arria II GX Embedded Memory Block Performance Specifications—Preliminary (Part 1 of 2)

Memory	Mode	Resources Used		Performance			Unit
		ALUTs	Embedded Memory	C4	C5	C6	
Memory Logic Array Block (MLAB)	Single port 64 × 10	0	1	500	500	400	MHz
	Simple dual-port 32 × 20 single clock	0	1	500	500	400	MHz
	Simple dual-port 64 × 10 single clock	0	1	500	500	400	MHz

Table 1–25. Arria II GX Embedded Memory Block Performance Specifications—Preliminary (Part 2 of 2)

Memory	Mode	Resources Used		Performance			Unit
		ALUTs	Embedded Memory	C4	C5	C6	
M9K Block	Single-port 256 × 36	0	1	390	350	295	MHz
	Single-port 256 × 36, with the **read-during-write** option set to **Old Data**	0	1	250	225	190	MHz
	Simple dual-port 256 × 36 single CLK	0	1	390	350	295	MHz
	Single-port 256 × 36 single CLK, with the **read-during-write** option set to **Old Data**	0	1	250	225	190	MHz
	True dual port 512 × 18 single CLK	0	1	390	350	295	MHz
	True dual-port 512 × 18 single CLK, with the **read-during-write** option set to **Old Data**	0	1	250	225	190	MHz

Configuration

Table 1–26 lists the Arria II GX configuration mode specifications.

Table 1–26. Arria II GX Configuration Mode Specifications—Preliminary

Programming Mode	DCLK F_{MAX}	Unit
Passive Serial	125	MHz
Fast Passive Parallel	125	MHz
Fast Active Serial	40	MHz
Remote Update only in Fast AS mode	10	MHz

JTAG Specifications

Table 1–27 lists the JTAG timing parameters and values for Arria II GX devices.

Table 1–27. Arria II GX JTAG Timing Parameters and Values—Preliminary

Symbol	Description	Min	Max	Unit
t_{JCP}	TCK clock period	30	—	ns
t_{JCH}	TCK clock high time	14	—	ns
t_{JCL}	TCK clock low time	14	—	ns
$t_{JPSU\ (TDI)}$	TDI JTAG port setup time	1	—	ns
$t_{JPSU\ (TMS)}$	TMS JTAG port setup time	3	—	ns
t_{JPH}	JTAG port hold time	5	—	ns
t_{JPCO}	JTAG port clock to output	—	11 *(1)*	ns
t_{JPZX}	JTAG port high impedance to valid output	—	14 *(1)*	ns
t_{JPXZ}	JTAG port valid output to high impedance	—	14 *(1)*	ns

Note to Table 1–27:

(1) A 1-ns adder is required for each V_{CCIO} voltage step down from 3.3 V. For example, t_{JPCO} = 12 ns if V_{CCIO} of the TDO I/O bank = 2.5 V, or 13 ns if it equals to 1.8 V.

Periphery Performance

This section describes periphery performance, including high-speed I/O and external memory interface.

I/O performance supports several system interfacing, for example the high-speed I/O interface, external memory interface, and the PCI/PCI-X bus interface. I/O using SSTL-18 Class I termination standard can achieve up to the stated DDR2 SDRAM interfacing speed as indicated in Table 1–30 with typical DDR2 SDRAM memory interface setup. I/O using general-purpose I/O standards such as 3.0, 2.5, 1.8, or 1.5 LVTTL/LVCMOS are capable of typical 200 MHz interfacing frequency with 10pF load.

☞ Actual achievable frequency depends on design- and system-specific factors. You should perform HSPICE/IBIS simulations based on your specific design and system setup to determine the maximum achievable frequency in your system.

High-Speed I/O Specification

Table 1–28 lists the high-speed I/O timing for Arria II GX devices.

Table 1–28. High-Speed I/O Specifications

Symbol	Conditions	C4		C5		C6		Unit
		Min	Max	Min	Max	Min	Max	
Clock								
f_{INCLK} (input clock frequency)	Clock boost factor, W = 1 to 40 *(1)*	5	500	5	472.5	5	472.5	MHz
Transmitter								
f_{HSDR_TX} (true differential output data rate)	SERDES factor, J = 3 to 10	150	1000 *(2)*	150	840	150	740	Mbps
$f_{HSDR\_TX\_E3R}$ (emulated differential 3R output data rate)	SERDES factor, J = 3 to 10	*(3)*	640	*(3)*	*(4)*	*(3)*	*(4)*	Mbps
Receiver								
f_{HSDR_RX}	DPA mode *(5)*	150	1000	150	840	150	740	Mbps
	Non-DPA mode	*(3)*	945 *(6)*	*(3)*	740	*(3)*	640	Mbps

Notes to Table 1–28:

(1) $f_{INCLK} = f_{HSDR} / W$. Use W to determine the supported selection of input reference clock frequencies for the desired data rate.

(2) This applies to interfacing with DPA receivers. For interfacing with non-DPA receivers, maximum supported data rate is 945 Mbps. Beyond 840 Mbps, PCB trace compensation is required. PCB trace compensation refers to the adjustment of PCB trace length for LVDS channels to improve channel-to-channel skews, and is optional when interfacing with DPA receivers.

(3) The minimum specification is dependent on the clock source (for example, PLL and clock pin) and the clock routing resource you use (global, regional, or local). The I/O differential buffer and input register do not have a minimum toggle rate.

(4) Pending silicon characterization.

(5) Dedicated SERDES and DPA features are only available on right banks.

(6) PCB trace compensation refers to the adjustment of PCB trace length for LVDS channels to improve channel-to-channel skews, and is required to support data rate beyond 840 Mbps.

Table 1–29 shows DPA lock time specifications for Arria II GX devices.

Table 1–29. DPA Lock Time Specifications—Preliminary

Standard	Training Pattern	Transition Density	Min	Unit
SPI-4	00000000001111111111	10%	256	Number of repetitions
Parallel Rapid I/O	00001111	25%	256	Number of repetitions
	10010000	50%	256	Number of repetitions
Miscellaneous	10101010	100%	256	Number of repetitions
	01010101	—	256	Number of repetitions

External Memory Interface Specifications

Table 1–30 lists the maximum clock rate support for external memory interfaces with the half-rate controller for the Arria II GX device family.

Use Table 1–30 for memory interface timing analysis.

Table 1–30. Arria II GX Maximum Clock Rate Support for External Memory Interfaces with Half-Rate Controller—Preliminary *(Note 1)*, *(2)*

Memory Standards	C4 (MHz)	C5 (MHz)	C6 (MHz)
DDR3 SDRAM *(3)*	300	N/A	N/A
DDR2 SDRAM *(4)*	300 *(5)*	267 *(6)*	200
DDR SDRAM *(4)*	200	200	200
QDRII SRAM *(7)*, *(8)*	250	250	200

Notes to Table 1–30:

(1) These numbers are preliminary until characterization is final. The supported operating frequencies listed here are memory interface maximums for the FPGA device family. Your design's actual achievable performance is based on design- and system-specific factors, as well as static timing analysis of the completed design.

(2) The maximum clock rates are applicable to Class I termination for interfaces using either row I/Os or column I/Os. The maximum clock rates can be lower for Class II termination or interfaces using a combination of row and column I/Os.

(3) Arria II GX devices support DDR3 SDRAM components only if no levelling support is required for DDR3 DIMMs. Interfaces with multiple DDR3 SDRAM components require component arrangement according to DDR2 DIMM tree topology.

(4) This applies to interfaces with both components and single rank unbuffered modules.

(5) The 300-MHz DDR2 interface requires the use of 400-MHz DDR2 SDRAM modules or components.

(6) The 267-MHz DDR2 interface requires the use of 333-MHz DDR2 SDRAM modules or components.

(7) QDRII SRAM supports 1.8-V and 1.5-V HSTL I/O standards. However, Altera recommends using the 1.8-V HSTL I/O standard for maximum performance because of the higher I/O drive strength.

(8) Arria II GX devices feature I/Os capable of electrical support for QDRII. However, Altera does not currently supply a controller or PHY megafunction for QDRII interfaces.

DLL and DQS Logic Block Specifications

Table 1–31 lists DLL frequency range specifications for Arria II GX devices.

Table 1–31. Arria II GX DLL Frequency Range Specifications—Preliminary

Frequency Mode	Frequency Range (MHz)			Resolution (°)
	C4	C5	C6	
0	90 – 150	90 – 140	90 – 120	22.5
1	120 – 200	120 – 190	120 – 170	30
2	150 – 240	150 – 230	150 – 200	36
3	180 – 300	180 – 290	180 – 250	45
4	240 – 370	240 – 350	240 – 310	30
5	290 – 450	290 – 420	290 – 370	36

Table 1–32 lists the DQS phase offset delay per stage for Arria II GX devices.

Table 1–32. DQS Phase Offset Delay Per Setting—Preliminary *(Note 1), (2), (3)*

Speed Grade	Min	Max	Unit
C4	7.0	13.0	ps
C5	*(4)*	*(4)*	ps
C6	8.5	15.5	ps

Notes to Table 1–32:

(1) The valid settings for phase offset are -64 to +63 for frequency modes 0 to 3 and -32 to +31 for frequency modes 4 to 5.

(2) The typical value equals the average of the minimum and maximum values.

(3) Delay settings are linear.

(4) Pending silicon characterization.

Duty Cycle Distortion (DCD) Specifications

Table 1–33 lists the worst-case DCD for Arria II GX devices.

Table 1–33. Duty Cycle Distortion on Arria II GX I/O Pins—Preliminary *(Note 1)*

Symbol	C4		C5		C6		Unit
	Min	Max	Min	Max	Min	Max	
Output Duty Cycle	45	55	45	55	45	55	%

Note to Table 1–33:

(1) Preliminary DCD specification applies to clock outputs from PLLs, global clock tree, and I/O elements (IOE) driving dedicated and general purpose I/O pins.

Glossary

Table 1–34 shows the glossary for this chapter.

Table 1–34. Glossary

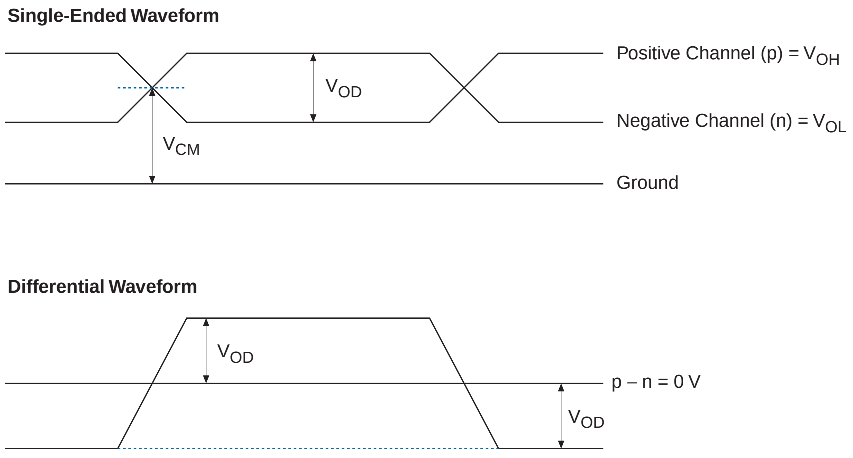
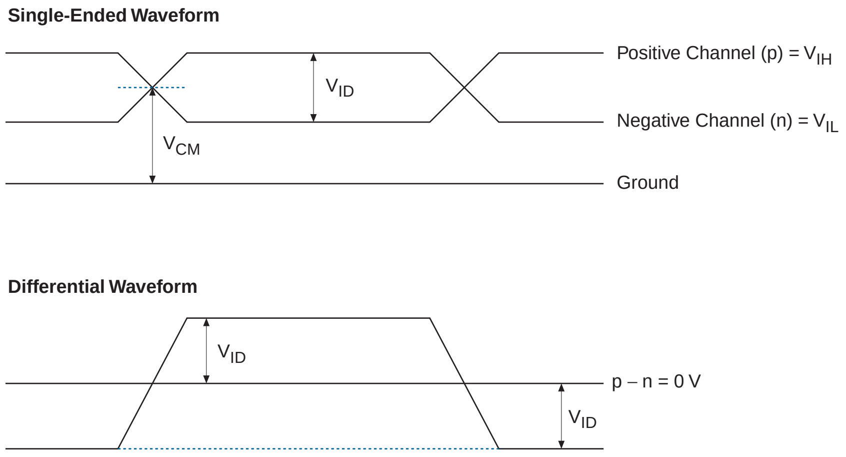
Letter	Subject	Definitions
A	—	—
B	—	—
C	—	—
D	Differential I/O Standards	*Receiver Input Waveforms*  Single-Ended Waveform Positive Channel (p) = V_{OH} V_{OD} Negative Channel (n) = V_{OL} V_{CM} Ground Differential Waveform V_{OD} p – n = 0 V V_{OD} *Transmitter Output Waveforms*  Single-Ended Waveform Positive Channel (p) = V_{IH} V_{ID} Negative Channel (n) = V_{IL} V_{CM} Ground Differential Waveform V_{ID} p – n = 0 V V_{ID}
E	—	—
F	f_{HSCLK}	Left/Right PLL input clock frequency.
	f_{HSDR}	High-Speed I/O Block: Maximum/minimum LVDS data transfer rate (f_{HSDR} = 1/TUI), non-DPA.
	$f_{HSDRDPA}$	High-Speed I/O Block: Maximum/minimum LVDS data transfer rate ($f_{HSDRDPA}$ = 1/TUI), DPA.

Table 1–34. Glossary (Continued)

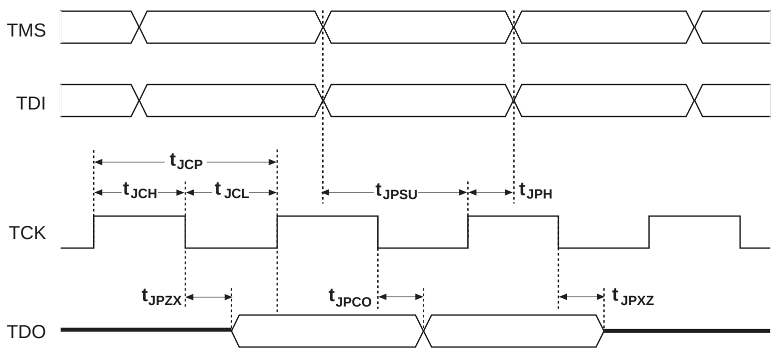
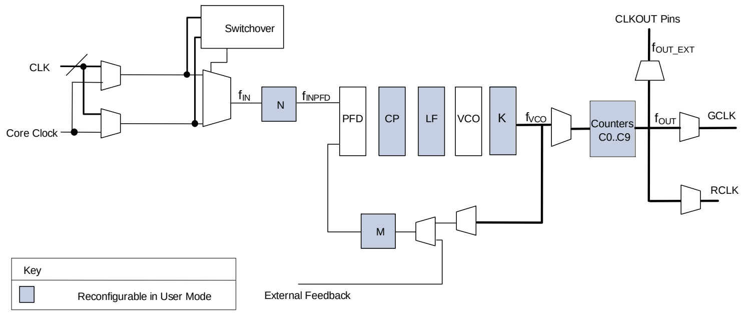
Letter	Subject	Definitions
G	—	—
H	—	—
I	—	—
J	J	High-Speed I/O Block: Deserialization factor (width of parallel data bus).
	JTAG Timing Specifications	JTAG Timing Specifications are in the following figure:  TMS TDI t_{JCP} t_{JCH} t_{JCL} t_{JPSU} t_{JPH} TCK t_{JPZX} t_{JPCO} t_{JPXZ} TDO
K	—	—
L	—	—
M	—	—
N	—	—
O	—	—
P	PLL Specifications	The block diagram shown in the following figure highlights the PLL Specification parameters: **Diagram of PLL Specifications (1)**  Switchover CLKOUT Pins f_{OUT_EXT} CLK f_{IN} N f_{INPFD} PFD CP LF VCO K f_{VCO} Counters C0..C9 f_{OUT} GCLK Core Clock RCLK M Key Reconfigurable in User Mode External Feedback **Note:** (1) CoreClock can only be fed by dedicated clock input pins or PLL outputs.
Q	—	—
R	R_L	Receiver differential input discrete resistor (external to the Arria II GX device).

Table 1–34. Glossary (Continued)

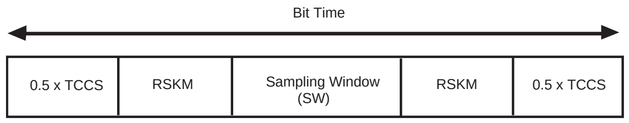
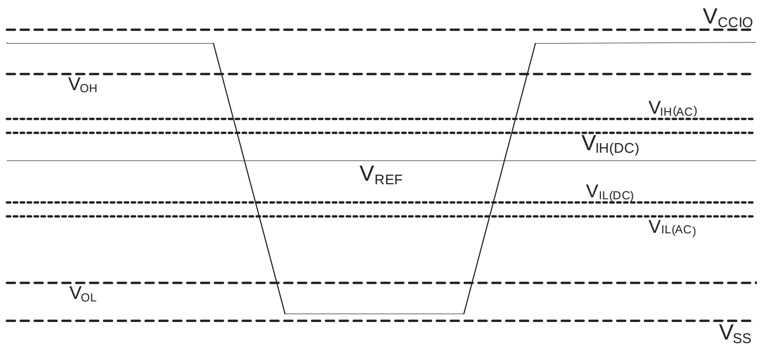
Letter	Subject	Definitions
S	SW (sampling window)	The period of time during which the data must be valid in order to capture it correctly. The setup and hold times determine the ideal strobe position within the sampling window: *Timing Diagram*  Bit Time 0.5 x TCCS RSKM Sampling Window (SW) RSKM 0.5 x TCCS
	Single-ended Voltage Referenced I/O Standard	The JEDEC standard for SSTL and HSTL I/O standards define both the AC and DC input signal values. The AC values indicate the voltage levels at which the receiver must meet its timing specifications. The DC values indicate the voltage levels at which the final logic state of the receiver is unambiguously defined. Once the receiver input has crossed the AC value, the receiver changes to the new logic state. The new logic state is then maintained as long as the input stays beyond the AC threshold. This approach is intended to provide predictable receiver timing in the presence of input waveform ringing: *Single-Ended Voltage Referenced I/O Standard*  V_{CCIO} V_{OH} $V_{IH(AC)}$ $V_{IH(DC)}$ V_{REF} $V_{IL(DC)}$ $V_{IL(AC)}$ V_{OL} V_{SS}
T	t_C	High-speed receiver and transmitter input and output clock period.
	TCCS (channel-to-channel-skew)	The timing difference between the fastest and slowest output edges, including t_{CO} variation and clock skew, across channels driven by the same PLL. The clock is included in the TCCS measurement (refer to the *Timing Diagram* figure under **S** in this table).
	$\mathbf{t}_{DUTY}$	High-speed I/O Block: Duty cycle on high-speed transmitter output clock. **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)
	$\mathbf{t}_{FALL}$	Signal high-to-low transition time (80-20%)
	$\mathbf{t}_{INCCJ}$	Cycle-to-cycle jitter tolerance on PLL clock input
	$\mathbf{t}_{OUTPJ\_IO}$	Period jitter on general purpose I/O driven by a PLL
	$\mathbf{t}_{OUTPJ\_DC}$	Period jitter on dedicated clock output driven by a PLL
	$\mathbf{t}_{RISE}$	Signal low-to-high transition time (20-80%)
U	—	—

Table 1–34. Glossary (Continued)

Letter	Subject	Definitions
V	$V_{CM(DC)}$	DC Common Mode Input Voltage.
	V_{ICM}	Input Common Mode Voltage: The common mode of the differential signal at the receiver.
	V_{ID}	Input differential Voltage Swing: The difference in voltage between the positive and complementary conductors of a differential transmission at the receiver.
	$V_{DIF(AC)}$	AC differential Input Voltage: Minimum AC input differential voltage required for switching.
	$V_{DIF(DC)}$	DC differential Input Voltage: Minimum DC input differential voltage required for switching.
	V_{IH}	Voltage Input High: The minimum positive voltage applied to the input which will be accepted by the device as a logic high.
	$V_{IH(AC)}$	High-level AC input voltage
	$V_{IH(DC)}$	High-level DC input voltage
	V_{IL}	Voltage Input Low: The maximum positive voltage applied to the input which will be accepted by the device as a logic low.
	$V_{IL(AC)}$	Low-level AC input voltage
	$V_{IL(DC)}$	Low-level DC input voltage
	V_{OCM}	Output Common Mode Voltage: The common mode of the differential signal at the transmitter.
	V_{OD}	Output differential Voltage Swing: The difference in voltage between the positive and complementary conductors of a differential transmission at the transmitter.
W	W	High-Speed I/O BLOCK: Clock Boost Factor
X	—	—
Y	—	—
Z	—	—

Document Revision History

Table 1–35 shows the revision history for this chapter.

Table 1–35. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
February 2009, v1.0	Initial release.	—

About this Handbook

This handbook provides comprehensive information about the Altera Arria II GX family of devices.

How to Contact Altera

For the most up-to-date information about Altera products, see the following table.

Contact *(Note 1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Altera literature services	Email	literature@altera.com
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions that this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicates command names and dialog box titles. For example, **Save As** dialog box.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, dialog box options, software utility names, and other GUI labels. For example, **\qdesigns** directory, **d:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicates document titles. For example, *AN 519: Stratix IV Design Guidelines*.
Italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, <*file name*> and <*project name*>**.pof** file.
Initial Capital Letters	Indicates keyboard keys and menu names. For example, Delete key and the Options menu.
“Subheading Title”	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, “Typographic Conventions.”

Visual Cue	Meaning
`Courier type`	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. Active-low signals are denoted by suffix `n`. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
1., 2., 3., and a., b., c., and so on.	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■	Bullets indicate a list of items when the sequence of the items is not important.
☞	The hand points to information that requires special attention.
⚠ CAUTION	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
⚠ WARNING	A warning calls attention to a condition or possible situation that can cause you injury.
↵	The angled arrow instructs you to press **Enter**.
👣	The feet direct you to more information about a particular topic.