Arria II Device Handbook

Volume 1: Device Interfaces and Integration

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

AIIGX5V1-4.6

Document last updated for Altera Complete Design Suite version: 13.1
Document publication date: February 2014

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

QUALITY
ISO 9001:2008
NSAI Certified

Contents

Chapter Revision Dates xi

Section I. Device Core for Arria II Devices

Revision History 1–1

Chapter 1. Overview for the Arria II Device Family

Arria II Device Feature 1–1
Arria II Device Architecture 1–6
High-Speed Transceiver Features 1–7
PCIe Hard IP Block 1–9
Logic Array Block and Adaptive Logic Modules 1–9
Embedded Memory Blocks 1–9
DSP Resources 1–10
I/O Features 1–10
High-Speed LVDS I/O and DPA 1–11
Clock Management 1–11
Auto-Calibrating External Memory Interfaces 1–12
Nios II 1–12
Configuration Features 1–12
SEU Mitigation 1–13
JTAG Boundary Scan Testing 1–13
Reference and Ordering Information 1–14
Document Revision History 1–14

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

Logic Array Blocks 2–1
LAB Interconnects 2–3
LAB Control Signals 2–4
Adaptive Logic Modules 2–5
ALM Operating Modes 2–7
Normal Mode 2–8
Extended LUT Mode 2–10
Arithmetic Mode 2–11
Shared Arithmetic Mode 2–13
LUT-Register Mode 2–15
Register Chain 2–16
ALM Interconnects 2–17
Clear and Preset Logic Control 2–17
LAB Power Management Techniques 2–17
Document Revision History 2–18

Chapter 3. Memory Blocks in Arria II Devices

Memory Features 3–2
Memory Block Types 3–3
Parity Bit Support 3–3
Byte Enable Support 3–3
Packed Mode Support 3–5
Address Clock Enable Support 3–5

Mixed Width Support . . . 3–8
Asynchronous Clear . . . 3–8
Error Correction Code Support . . . 3–8
Memory Modes . . . 3–10
Single-Port RAM Mode . . . 3–10
Simple Dual-Port Mode . . . 3–12
True Dual-Port Mode . . . 3–15
Shift-Register Mode . . . 3–17
ROM Mode . . . 3–18
FIFO Mode . . . 3–18
Clocking Modes . . . 3–19
Independent Clock Mode . . . 3–19
Input and Output Clock Mode . . . 3–19
Read and Write Clock Mode . . . 3–19
Single Clock Mode . . . 3–20
Design Considerations . . . 3–20
Selecting Memory Block . . . 3–20
Conflict Resolution . . . 3–20
Read-During-Write Behavior . . . 3–21
Same-Port Read-During-Write Mode . . . 3–21
Mixed-Port Read-During-Write Mode . . . 3–23
Power-Up Conditions and Memory Initialization . . . 3–26
Power Management . . . 3–26
Document Revision History . . . 3–27

Chapter 4. DSP Blocks in Arria II Devices

DSP Block Overview . . . 4–2
Simplified DSP Operation . . . 4–4
Operational Modes Overview . . . 4–7
DSP Block Resource Descriptions . . . 4–8
Input Registers . . . 4–9
Multiplier and First-Stage Adder . . . 4–11
Pipeline Register Stage . . . 4–12
Second-Stage Adder . . . 4–12
Rounding and Saturation Stage . . . 4–12
Second Adder and Output Registers . . . 4–13
Arria II Operational Mode Descriptions . . . 4–14
Independent Multiplier Modes . . . 4–14
9-Bit, 12-Bit, and 18-Bit Multiplier . . . 4–14
36-Bit Multiplier . . . 4–17
Double Multiplier . . . 4–17
Two-Multiplier Adder Sum Mode . . . 4–20
18 × 18 Complex Multiplier . . . 4–22
Four-Multiplier Adder . . . 4–23
High-Precision Multiplier Adder Mode . . . 4–24
Multiply Accumulate Mode . . . 4–25
Shift Modes . . . 4–26
Rounding and Saturation Mode . . . 4–28
DSP Block Control Signals . . . 4–30
Software Support for Arria II Devices . . . 4–31
Document Revision History . . . 4–32

Chapter 5. Clock Networks and PLLs in Arria II Devices

Clock Networks in Arria II Devices 5–1
Global Clock Networks 5–3
Regional Clock Networks 5–4
Periphery Clock Networks 5–6
Clock Sources Per Quadrant 5–8
Clock Regions 5–9
Clock Network Sources 5–11
Dedicated Clock Inputs Pins 5–11
Logic Array Blocks 5–11
PLL Clock Outputs 5–11
Clock Input Connections to PLLs 5–13
Clock Output Connections 5–14
Clock Control Block 5–15
Clock Enable Signals 5–18
Clock Source Control for PLLs 5–19
Cascading PLLs 5–21
PLLs in Arria II Devices 5–21
PLL Hardware Overview in Arria II Devices 5–23
PLL Clock I/O Pins 5–23
PLL Control Signals 5–27
pfdena 5–27
areset 5–27
locked 5–27
Clock Feedback Modes 5–28
Source-Synchronous Mode 5–29
Source-Synchronous Mode for LVDS Compensation 5–30
No-Compensation Mode 5–30
Normal Mode 5–31
Zero-Delay Buffer Mode 5–32
External Feedback Mode 5–33
Clock Multiplication and Division 5–34
Post-Scale Counter Cascading 5–35
Programmable Duty Cycle 5–36
Programmable Phase Shift 5–36
Programmable Bandwidth 5–38
Spread-Spectrum Tracking 5–38
Clock Switchover 5–38
Automatic Clock Switchover Mode 5–39
Manual Clock Switchover Mode 5–42
Clock Switchover Guidelines 5–42
PLL Reconfiguration 5–43
PLL Reconfiguration Hardware Implementation 5–44
Post-Scale Counters (C0 to C9) 5–46
Scan Chain Description 5–47
Charge Pump and Loop Filter 5–50
Bypassing PLL 5–51
Dynamic Phase-Shifting 5–51
PLL Specifications 5–54
Document Revision History 5–54

Section II. I/O Interfaces for Arria II Devices

Revision History 5–1

Chapter 6. I/O Features in Arria II Devices

I/O Standards Support 6–2
I/O Banks 6–5
Modular I/O Banks 6–7
I/O Structure 6–10
3.3-V I/O Interface 6–13
External Memory Interfaces 6–13
High-Speed Differential I/O with DPA Support 6–14
Programmable Current Strength 6–14
Programmable Slew Rate Control 6–16
Open-Drain Output 6–16
Bus Hold 6–17
Programmable Pull-Up Resistor 6–17
Programmable Pre-Emphasis 6–17
Programmable Differential Output Voltage 6–17
MultiVolt I/O Interface 6–18
OCT Support 6–19
R_S OCT Without Calibration for Arria II Devices 6–19
R_S OCT with Calibration for Arria II Devices 6–20
Left-Shift R_S OCT Control for Arria II GZ Devices 6–21
Expanded R_S OCT with Calibration for Arria II GZ Devices 6–22
R_D OCT for Arria II LVDS Input I/O Standard 6–23
R_T OCT with Calibration for Arria II GZ Devices 6–23
Dynamic R_S and R_T OCT for Single-Ended I/O Standard for Arria II GZ Devices 6–24
Arria II OCT Calibration 6–26
OCT Calibration Block 6–26
Termination Schemes for I/O Standards 6–28
Single-Ended I/O Standards Termination 6–28
Differential I/O Standards Termination 6–30
LVDS 6–32
Differential LVPECL 6–33
RSDS 6–33
mini-LVDS 6–34
Design Considerations 6–35
I/O Termination 6–35
Single-Ended I/O Standards 6–35
Differential I/O Standards 6–35
I/O Bank Restrictions 6–36
Non-Voltage-Referenced Standards 6–36
Voltage-Referenced Standards 6–36
Mixing Voltage-Referenced and Non-Voltage-Referenced Standards 6–36
I/O Placement Guidelines 6–37
3.3-V, 3.0-V, and 2.5-V LVTTL/LVCMOS Tolerance Guidelines 6–37
Pin Placement Guideline 6–37
Document Revision History 6–37

Chapter 7. External Memory Interfaces in Arria II Devices

Memory Interfaces Pin Support for Arria II Devices 7–3
Using the R_{UP} and R_{DN} Pins in a DQ/DQS Group Used for Memory Interfaces in Arria II GZ Devices 7–21
Combining ×16/×18 DQ/DQS Groups for ×36 QDR II+/QDR II SRAM Interface 7–21
Rules to Combine Groups 7–22
Arria II External Memory Interface Features 7–24

DQS Phase-Shift Circuitry 7–24
DLL 7–27
Phase Offset Control 7–32
DQS Logic Block 7–34
DQS Delay Chains 7–34
Update Enable Circuitry 7–35
DQS Postamble Circuitry 7–35
Arria II GZ Dynamic On-Chip Termination Control 7–37
I/O Element Registers 7–37
Document Revision History 7–42

Chapter 8. High-Speed Differential I/O Interfaces and DPA in Arria II Devices

LVDS Channels 8–2
Locations of the I/O Banks 8–3
LVDS SERDES and DPA Block Diagram 8–7
Differential Transmitter 8–8
Serializer 8–8
Programmable Pre-Emphasis and Programmable V_{OD}. 8–10
Differential Receiver 8–11
Receiver Hardware Blocks 8–12
DPA 8–12
Synchronizer 8–13
Data Realignment Block (Bit Slip) 8–14
Deserializer 8–15
Receiver Datapath Modes 8–16
Non-DPA 8–16
DPA Mode 8–18
Soft CDR Mode 8–19
Differential I/O Termination 8–20
PLLs 8–21
LVDS and DPA Clock Networks 8–21
Source-Synchronous Timing Budget 8–23
Differential Data Orientation 8–23
Differential I/O Bit Position 8–23
Transmitter Channel-to-Channel Skew 8–25
Receiver Skew Margin for Non-DPA Mode 8–25
Differential Pin Placement Guidelines 8–27
DPA-Enabled Channels and Single-Ended I/Os 8–27
Guidelines for DPA-Enabled Differential Channels 8–27
DPA-Enabled Channel Driving Distance 8–27
Using Center and Corner Left and Right PLLs in Arria II GX Devices 8–27
Using Both Center PLLs 8–29
Using Both Corner PLLs in Arria II GX Devices 8–31
Guidelines for DPA-Disabled Differential Channels 8–33
DPA-Disabled Channel Driving Distance 8–33
Using Corner and Center PLLs in Arria II GX Devices 8–33
Using Both Center PLLs 8–35
Using Both Corner PLLs in Arria II GX Devices 8–36
Setting Up an LVDS Transmitter or Receiver Channel 8–36
Document Revision History 8–36

Section III. System Integration for Arria II Devices

Revision History 8–1

Chapter 9. Configuration, Design Security, and Remote System Upgrades in Arria II Devices

Configuration Devices . . . 9–2
Configuration Features . . . 9–2
Power-On Reset Circuit and Configuration Pins Power Supply . . . 9–4
Power-On Reset Circuit . . . 9–4
Configuration Pins Power Supply . . . 9–5
V_{CCPD} Pins . . . 9–6
Configuration Process . . . 9–7
Power Up . . . 9–7
Reset . . . 9–7
Configuration . . . 9–7
Configuration Error . . . 9–8
Initialization . . . 9–8
User Mode . . . 9–9
Configuration Schemes . . . 9–9
MSEL Pin Settings . . . 9–9
Raw Binary File Size . . . 9–11
Fast Passive Parallel Configuration . . . 9–11
FPP Configuration Using a MAX II Device as an External Host . . . 9–11
FPP Configuration Timing . . . 9–15
AS and Fast AS Configuration (Serial Configuration Devices) . . . 9–19
Guidelines for Connecting Serial Configuration Device to Arria II Devices on an AS Interface . . 9–23
Estimating the AS Configuration Time . . . 9–23
Programming Serial Configuration Devices . . . 9–24
PS Configuration . . . 9–26
PS Configuration Using a MAX II Device as an External Host . . . 9–26
PS Configuration Timing . . . 9–29
PS Configuration Using a Download Cable . . . 9–30
JTAG Configuration . . . 9–33
Jam STAPL . . . 9–38
Device Configuration Pins . . . 9–39
Configuration Data Decompression . . . 9–46
Remote System Upgrades . . . 9–48
Functional Description . . . 9–49
Enabling Remote Update . . . 9–51
Configuration Image Types . . . 9–52
Remote System Upgrade Mode . . . 9–52
Remote Update Mode . . . 9–52
Dedicated Remote System Upgrade Circuitry . . . 9–55
Remote System Upgrade Registers . . . 9–56
Remote System Upgrade Control Register . . . 9–56
Remote System Upgrade Status Register . . . 9–57
Remote System Upgrade State Machine . . . 9–58
User Watchdog Timer . . . 9–59
Quartus II Software Support . . . 9–60
ALTREMOTE_UPDATE Megafunction . . . 9–60
Design Security . . . 9–61
Arria II Security Protection . . . 9–62
Security Against Copying . . . 9–62
Security Against Reverse Engineering . . . 9–62
Security Against Tampering . . . 9–62
AES Decryption Block . . . 9–62
Flexible Security Key Storage . . . 9–63
Arria II Design Security Solution . . . 9–64

Security Modes Available 9–65
Volatile Key 9–65
Non-Volatile Key 9–65
Volatile Key with Tamper Protection Bit Set 9–65
Non-Volatile Key with Tamper Protection Bit Set 9–65
No Key Operation 9–66
Supported Configuration Schemes 9–66
Document Revision History 9–69

Chapter 10. SEU Mitigation in Arria II Devices
Error Detection Fundamentals 10–1
Configuration Error Detection 10–2
User Mode Error Detection 10–2
Automated Single Event Upset Detection 10–4
Error Detection Pin Description 10–5
Error Detection Block 10–5
Error Detection Registers 10–6
Error Detection Timing 10–7
Software Support 10–9
Recovering From CRC Errors 10–10
Document Revision History 10–10

Chapter 11. JTAG Boundary-Scan Testing in Arria II Devices
BST Architecture for Arria II Devices 11–1
IEEE Std. 1149.6 Boundary-Scan Register for Arria II GX Devices 11–1
BST Operation Control 11–3
EXTEST_PULSE Instruction Mode 11–4
EXTEST_TRAIN Instruction Mode 11–5
I/O Voltage Support in a JTAG Chain 11–5
Disabling IEEE Std. 1149.1 BST Circuitry 11–6
Boundary-Scan Description Language Support 11–7
Document Revision History 11–8

Chapter 12. Power Management in Arria II Devices
External Power Supply Requirements 12–1
Power-On Reset Circuitry 12–1
Hot Socketing 12–2
Devices Can Be Driven Before Power-Up 12–2
I/O Pins Remain Tri-Stated During Power-Up 12–2
Insertion or Removal of an Arria II Device from a Powered-Up System 12–3
Hot-Socketing Feature Implementation 12–3
Document Revision History 12–4

Additional Information
About this Handbook Info–1
How to Contact Altera Info–1
Typographic Conventions Info–1

Chapter Revision Dates

The chapters in this document, Arria II Device Handbook Volume 1: Device Interfaces and Integration, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1. Overview for the Arria II Device Family
Revised: *July 2012*
Part Number: *AIIGX51001-4.4*

Chapter 2. Logic Array Blocks and Adaptive Logic Modules in Arria II Devices
Revised: *December 2010*
Part Number: *AIIGX51002-2.0*

Chapter 3. Memory Blocks in Arria II Devices
Revised: *December 2011*
Part Number: *AIIGX51003-3.2*

Chapter 4. DSP Blocks in Arria II Devices
Revised: *December 2010*
Part Number: *AIIGX51004-4.0*

Chapter 5. Clock Networks and PLLs in Arria II Devices
Revised: *July 2012*
Part Number: *AIIGX51005-4.2*

Chapter 6. I/O Features in Arria II Devices
Revised: *December 2011*
Part Number: *AIIGX51006-4.2*

Chapter 7. External Memory Interfaces in Arria II Devices
Revised: *June 2011*
Part Number: *AIIGX51007-4.1*

Chapter 8. High-Speed Differential I/O Interfaces and DPA in Arria II Devices
Revised: *July 2012*
Part Number: *AIIGX51008-4.3*

Chapter 9. Configuration, Design Security, and Remote System Upgrades in Arria II Devices
Revised: *July 2012*
Part Number: *AIIGX51009-4.3*

Chapter 10. SEU Mitigation in Arria II Devices
Revised: *February 2014*
Part Number: *AIIGX51010-4.3*

Chapter 11. JTAG Boundary-Scan Testing in Arria II Devices
Revised: *December 2013*
Part Number: *AIIGX51011-4.1*

Chapter 12. Power Management in Arria II Devices

Revised: *June 2011*
Part Number: *AIIGX51012-3.1*

Section I. Device Core for Arria II Devices

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

- Chapter 1, Overview for the Arria II Device Family
- Chapter 2, Logic Array Blocks and Adaptive Logic Modules in Arria II Devices
- Chapter 3, Memory Blocks in Arria II Devices
- Chapter 4, DSP Blocks in Arria II Devices
- Chapter 5, Clock Networks and PLLs in Arria II 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.

1. Overview for the Arria II Device Family

AIIGX51001-4.4

The Arria® II 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 Physical Interface for PCI Express® (PCIe®), Ethernet, and DDR3 memory are easily implemented in your design with the Quartus® II software, the SOPC Builder design software, and a broad library of hard and soft intellectual property (IP) solutions from Altera. The Arria II device family makes designing for applications requiring transceivers operating at up to 6.375 Gbps fast and easy.

This chapter contains the following sections:

- "Arria II Device Feature" on page 1–1
- "Arria II Device Architecture" on page 1–6
- "Reference and Ordering Information" on page 1–14

Arria II Device Feature

The Arria II 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 550 MHz
  - Configurable as 9 x 9-bit, 12 x 12-bit, 18 x 18-bit, and 36 x 36-bit full-precision multipliers as well as 18 x 36-bit high-precision multiplier
  - Hardcoded adders, subtractors, accumulators, and summation functions
  - Fully-integrated design flow with the MATLAB and DSP Builder software from Altera
- Maximum system bandwidth
  - Up to 24 full-duplex clock data recovery (CDR)-based transceivers supporting rates between 600 Mbps and 6.375 Gbps
  - Dedicated circuitry to support physical layer functionality for popular serial protocols, including PCIe Gen1 and PCIe Gen2, Gbps Ethernet, Serial RapidIO® (SRIO), Common Public Radio Interface (CPRI), OBSAI, SD/HD/3G/ASI Serial Digital Interface (SDI), XAUI and Reduced XAUI (RXAUI), HiGig/HiGig+, SATA/Serial Attached SCSI (SAS), GPON, SerialLite II, Fiber Channel, SONET/SDH, Interlaken, Serial Data Converter (JESD204), and SFI-5.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

- Complete PIPE protocol solution with an embedded hard IP block that provides physical interface and media access control (PHY/MAC) layer, Data Link layer, and Transaction layer functionality
- Optimized for high-bandwidth system interfaces
  - Up to 726 user I/O pins arranged in up to 20 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 1.25 Gbps
- Low power
  - Architectural power reduction techniques
  - Typical physical medium attachment (PMA) power consumption of 100 mW 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 (R_S) and on-chip parallel (R_T) termination with auto-calibration for single-ended I/Os and on-chip differential (R_D) termination for differential I/O
  - 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)
- Emulated LVDS output support with a data rate of up to 1152 Mbps

Table 1–1 lists the Arria II device features.

Table 1–1. Features in Arria II Devices

Feature	Arria II GX Devices						Arria II GZ Devices		
	EP2AGX45	EP2AGX65	EP2AGX95	EP2AGX125	EP2AGX190	EP2AGX260	EP2AGZ225	EP2AGZ300	EP2AGZ350
Total Transceivers *(1)*	8	8	12	12	16	16	16 or 24	16 or 24	16 or 24
ALMs	18,050	25,300	37,470	49,640	76,120	102,600	89,600	119,200	139,400
LEs	42,959	60,214	89,178	118,143	181,165	244,188	224,000	298,000	348,500
PCIe hard IP blocks	1	1	1	1	1	1	1	1	1
M9K Blocks	319	495	612	730	840	950	1,235	1,248	1,248
M144K Blocks	—	—	—	—	—	—	—	24	36
Total Embedded Memory in M9K Blocks (Kbits)	2,871	4,455	5,508	6,570	7,560	8,550	11,115	14,688	16,416
Total On-Chip Memory (M9K +M144K + MLABs) (Kbits)	3,435	5,246	6,679	8,121	9,939	11,756	13,915	18,413	20,772
Embedded Multipliers (18 x 18) *(2)*	232	312	448	576	656	736	800	920	1,040
General Purpose PLLs	4	4	6	6	6	6	6 or 8	4, 6, or 8	4, 6, or 8
Transceiver TX PLLs *(3)*, *(4)*	2 or 4	2 or 4	4 or 6	4 or 6	6 or 8	6 or 8	8 or 12	8 or 12	8 or 12
User I/O Banks *(5)*, *(6)*	6	6	8	8	12	12	16 or 20	8, 16, or 20	8, 16, or 20
High-Speed LVDS SERDES (up to 1.25 Gbps) *(7)*	8, 24, or 28	8, 24, or 28	24, 28, or 32	24, 28, 32	28 or 48	24 or 48	42 or 86	0 *(8)*, 42, or 86	0 *(8)*, 42, or 86

Notes to Table 1–1:

(1) The total number of transceivers is divided equally between the left and right side of each device, except for the devices in the F780 package. These devices have eight transceiver channels located only on the right side of the device.

(2) This is in four multiplier adder mode.

(3) The FPGA fabric can use these phase locked-loops (PLLs) if they are not used by the transceiver.

(4) The number of PLLs depends on the package. Transceiver transmitter (TX) PLL count = (number of transceiver blocks) × 2.

(5) Banks 3C and 8C are dedicated configuration banks and do not have user I/O pins.

(6) For Arria II GZ devices, the user I/Os count from pin-out files includes all general purpose I/O, dedicated clock pins, and dual purpose configuration pins. Transceiver pins and dedicated configuration pins are not included in the pin count.

(7) For Arria II GZ devices, total pairs of high-speed LVDS SERDES take the lowest channel count of RX/TX. For more information, refer to the *High-Speed I/O Interfaces and DPA in Arria II Devices* chapter.

(8) The smallest pin package (780-pin package) does not support high-speed LVDS SERDES.

Table 1–2 and Table 1–3 list the Arria II device package options and user I/O pin counts, high-speed LVDS channel counts, and transceiver channel counts for Ultra FineLine BGA (UBGA) and FineLine BGA (FBGA) devices.

Table 1–2. Package Options and I/O Information for Arria II GX Devices *(Note 1), (2), (3), (4), (5), (6), (7)*

Device	358-Pin Flip Chip UBGA 17 mm x 17 mm			572-Pin Flip Chip FBGA 25 mm x 25 mm			780-Pin Flip Chip FBGA 29 mm x 29 mm			1152-Pin Flip Chip FBGA 35 mm x 35 mm		
	I/O	LVDS *(8)*	XCVRs	I/O	LVDS *(8)*	XCVRs	I/O	LVDS *(8)*	XCVRs	I/O	LVDS *(8)*	XCVRs
EP2AGX45	156	33(R_D or eTX) + 32(RX, TX, or eTX)	4	252	57(R_D or eTX) + 56(RX, TX, or eTX)	8	364	85(R_D or eTX) + 84(RX, TX, or eTX)	8	—	—	—
EP2AGX65	156	33(R_D or eTX) + 32(RX, TX, or eTX)	4	252	57(R_D or eTX) + 56(RX, TX, or eTX)	8	364	85(R_D or eTX) +84(RX,TX, eTX)	8	—	—	—
EP2AGX95	—	—	—	260	57(R_D or eTX) + 56(RX, TX, or eTX)	8	372	85(R_D or eTX) +84(RX, TX, or eTX)	12	452	105(R_D or eTX) + 104(RX, TX, or eTX)	12
EP2AGX125	—	—	—	260	57(R_D or eTX) + 56(RX,TX, or eTX)	8	372	85(R_D or eTX) +84(RX,TX, or eTX)	12	452	105(R_D or eTX) + 104(RX, TX, or eTX)	12
EP2AGX190	—	—	—	—	—	—	372	85(R_D or eTX) +84(RX, TX, or eTX)	12	612	145(R_D or eTX) + 144(RX, TX, or eTX)	16
EP2AGX260	—	—	—	—	—	—	372	85(R_D, eTX) +84(RX, TX, or eTX)	12	612	145(R_D, eTX) + 144(RX, TX, or eTX)	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) R_D = True LVDS input buffers with on-chip differential termination (R_D OCT) support.

(4) RX = True LVDS input buffers without R_D OCT support.

(5) TX = True LVDS output buffers.

(6) eTX = Emulated-LVDS output buffers, either `LVDS_E_3R` or `LVDS_E_1R`.

(7) The LVDS channel count does not include dedicated clock input pins and PLL clock output pins.

(8) These numbers represent the accumulated LVDS channels supported in Arria II GX row and column I/O banks.

Table 1–3. Package Options and I/O Information for Arria II GZ Devices *(Note 1)*, *(2)*, *(3)*, *(4)*, *(5)*

Device	780-Pin Flip Chip FBGA 29 mm x 29 mm			1152-Pin Flip Chip FBGA 35 mm x 35 mm			1517-Pin Flip Chip FBGA 40 mm x 40 mm		
	I/O	LVDS *(6)*	XCVRs	I/O	LVDS *(7)*	XCVRs	I/O	LVDS *(7)*	XCVRs
EP2AGZ225	—	—	—	554	135 (RX or eTX) + 140 (TX or eTX)	16	734	179 (RX or eTX) + 184 (TX or eTX)	24
EP2AGZ300	281	68 (RX or eTX) + 72 eTX	16	554	135 (RX or eTX) + 140 (TX or eTX)	16	734	179 (RX or eTX) + 184 (TX or eTX)	24
EP2AGZ350	281	68 (RX or eTX) + 72 eTX	16	554	135 (RX or eTX) + 140 (TX or eTX)	16	734	179 (RX or eTX) + 184 (TX or eTX)	24

Notes to Table 1–3:

(1) The user I/O counts include clock pins.

(2) RX = True LVDS input buffers without R_D OCT support for row I/O banks, or true LVDS input buffers without R_D OCT support for column I/O banks.

(3) eTX = Emulated-LVDS output buffers, either `LVDS_E_3R` or `LVDS_E_1R`.

(4) The LVDS RX and TX channels are equally divided between the left and right sides of the device.

(5) The LVDS channel count does not include dedicated clock input pins.

(6) For Arria II GZ 780-pin FBGA package, the LVDS channels are only supported in column I/O banks.

(7) These numbers represents the accumulated LVDS channels supported in Arria II GZ device row and column I/O banks.

Arria II devices are available in up to four speed grades: –3 (fastest), –4, –5, and –6 (slowest). Table 1–4 lists the speed grades for Arria II devices.

Table 1–4. Speed Grades for Arria II Devices

Device	358-Pin Flip Chip UBGA	572-Pin Flip Chip FBGA	780-Pin Flip Chip FBGA	1152-Pin Flip Chip FBGA	1517-Pin Flip Chip FBGA
EP2AGX45	C4, C5, C6, I3, I5	C4, C5, C6, I3, I5	C4, C5, C6, I3, I5	—	—
EP2AGX65	C4, C5, C6, I3, I5	C4, C5, C6, I3, I5	C4, C5, C6, I3, I5	—	—
EP2AGX95	—	C4, C5, C6, I3, I5	C4, C5, C6, I3, I5	C4, C5, C6, I3, I5	—
EP2AGX125	—	C4, C5, C6, I3, I5	C4, C5, C6, I3, I5	C4, C5, C6, I3, I5	—
EP2AGX190	—	—	C4, C5, C6, I3, I5	C4, C5, C6, I3, I5	—
EP2AGX260	—	—	C4, C5, C6, I3, I5	C4, C5, C6, I3, I5	—
EP2AGZ225	—	—	—	C3, C4, I3, I4	C3, C4, I3, I4
EP2AGZ300	—	—	C3, C4, I3, I4	C3, C4, I3, I4	C3, C4, I3, I4
EP2AGZ350	—	—	C3, C4, I3, I4	C3, C4, I3, I4	C3, C4, I3, I4

Arria II Device Architecture

Arria II devices 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 devices support external memory interfaces and I/O protocols required by wireless, wireline, broadcast, computer, storage, and military markets. They inherit the 8-input ALM, M9K and M144K 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 6.375 Gbps speeds.

Figure 1–1 and Figure 1–2 show an overview of the Arria II GX and Arria II GZ device architecture, respectively.

Figure 1–1. Architecture Overview for Arria II GX Devices

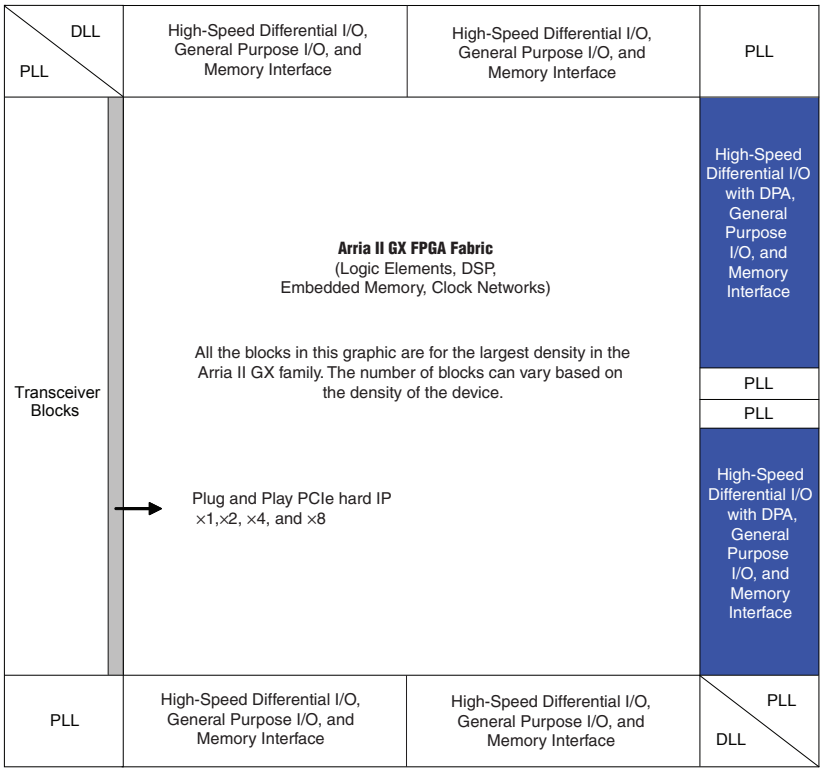
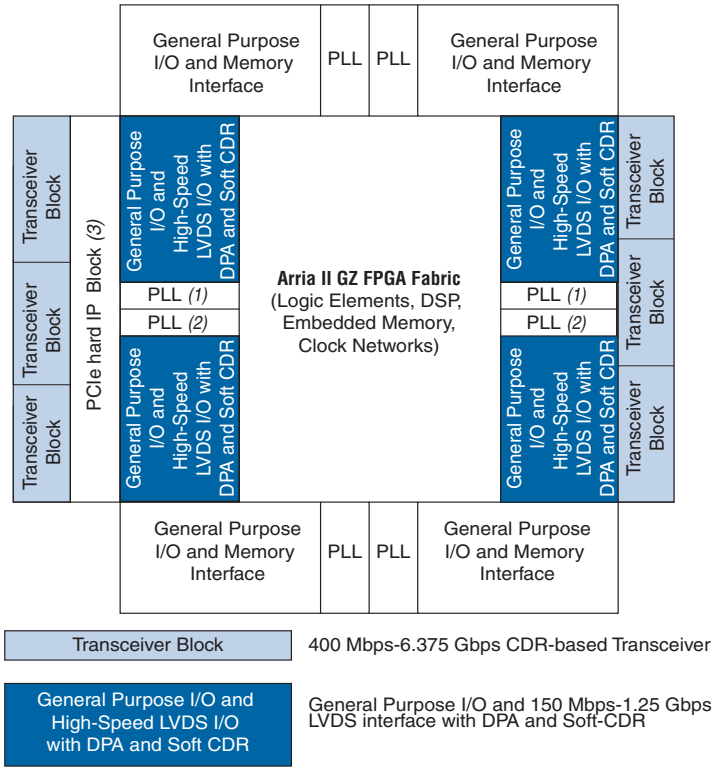


Figure 1–2. Architecture Overview for Arria II GZ Device



Notes to Figure 1–2:

(1) Not available for 780-pin FBGA package.

(2) Not available for 780-pin and 1152-pin FBGA packages.

(3) The PCIe hard IP block is located on the left side of the device only (IOBANK_QL).

High-Speed Transceiver Features

Arria II GX devices integrate up to 16 transceivers and Arria II GZ devices up to 24 transceivers on a single device. The transceiver block is optimized for cost and power consumption. Arria II transceivers support the following features:

- Configurable pre-emphasis and equalization, and adjustable output differential voltage
- Flexible and easy-to-configure transceiver datapath to implement proprietary protocols
- Signal integrity features
  - Programmable transmitter pre-emphasis to compensate for inter-symbol interference (ISI)
  - User-controlled receiver equalization with up to 7 dB (Arria II GX) and 16 dB (Arria II GZ) 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 physical coding sublayer (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 PCIe hard IP blocks
  - Support for protocol features such as MSB-to-LSB transmission in a SONET/SDH configuration and spread-spectrum clocking in a PCIe configuration

Table 1–5 lists common protocols and the Arria II dedicated circuitry and features for implementing these protocols.

Table 1–5. Sample of Supported Protocols and Feature Descriptions for Arria II Devices

Supported Protocols	Feature Descriptions

PCIe	■ Complete PCIe Gen1 and Gen2 protocol stack solution compliant to PCIe Base Specification 2.0 that includes PHY/MAC, Data Link, and Transaction layer circuitry embedded in the PCIe hard IP blocks.
	■ PCIe Gen1 has x1, x2, x4, and x8 lane configurations. PCIe Gen2 has x1, x2, and x4 lane configurations. PCIe Gen2 does not support x8 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
GbE	■ 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	■ Transmit 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 devices, such as SONET/SDH, SDI, SATA and SRIO, refer to the *Transceiver Architecture in Arria II Devices* chapter.

PCIe Gen2 protocol is only available in Arria II GZ devices.

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

PCIe Hard IP Block

Every Arria II device includes an integrated hard IP block which implements PCIe PHY/MAC, data link, and transaction layers. This PCIe hard IP block is highly configurable to meet the requirements of the majority of PCIe applications. PCIe hard IP makes implementing PCIe Gen1 and PCIe Gen2 solution in your Arria II design simple and easy.

You can instantiate PCIe hard IP block 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 ensure correct operation of the core. Table 1–6 lists the PCIe hard IP block support for Arria II GX and GZ devices.

Table 1–6. PCIe Hard IP Block Support

Support	Arria II GX Devices	Arria II GZ Devices
PCIe Gen1	x1, x4, x8	x1, x4, x8
PCIe Gen2	—	x1, x4
Root Port and endpoint configurations	Yes	Yes
Payloads	128-byte to 256-byte	128-byte to 2K-byte

Logic Array Block and Adaptive Logic Modules

- Logic array blocks (LABs) consists of 10 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 logic elements (LEs), logic levels, and associated routing
- LABs have a derivative called MLAB, which 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

- MLABs, M9K, and M144K embedded memory blocks provide up to 20,836 Kbits of on-chip memory capable of up to 540-MHz performance. The embedded memory structure consists of columns of embedded 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.

- The Quartus® II software allows you to take advantage of MLABs, M9K, and M144K memory blocks by instantiating memory using a dedicated megafunction wizard or by inferring memory directly from VHDL or Verilog source code.

Table 1–7 lists the Arria II device memory modes.

Table 1–7. Memory Modes for Arria II Devices

Port Mode	Port Width Configuration
Single Port	x1, x2, x4, x8, x9, x16, x18, x32, x36, x64, and x72
Simple Dual Port	x1, x2, x4, x8, x9, x16, x18, x32, x36, x64, and x72
True Dual Port	x1, x2, x4, x8, x9, x16, x18, x32, and x36

DSP Resources

- Fulfills the DSP requirements of 3G and Long Term Evolution (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
- The Quartus II software includes megafunctions you can use to control the mode of operation of the DSP blocks based on user-parameter settings
- You can directly infer multipliers from the VHDL or Verilog HDL source code

I/O Features

- Contains up to 20 modular I/O banks
- All I/O banks support a wide range of single-ended and differential I/O standards listed in Table 1–8.

Table 1–8. I/O Standards Support for Arria II Devices

Type	I/O Standard
Single-Ended I/O	LVTTL, LVCMOS, SSTL, HSTL, PCIe, and PCI-X
Differential I/O	SSTL, HSTL, LVPECL, LVDS, mini-LVDS, Bus LVDS (BLVDS) *(1)*, and RSDS

Note to Table 1–8:

(1) BLVDS is only available for Arria II GX devices.

- Supports programmable bus hold, programmable weak pull-up resistors, and programmable slew rate control
- For Arria II devices, calibrates OCT or driver impedance matching for single-ended I/O standards with one OCT calibration block on the I/O banks listed in Table 1–9.

Table 1–9. Location of OCT Calibration Block in Arria II Devices

Device	Package Option	I/O Bank
Arria II GX	All pin packages	Bank 3A, Bank 7A, and Bank 8A
Arria II GZ	780-pin flip chip FBGA	Bank 3A, Bank 4A, Bank 7A, and Bank 8A
	1152-pin flip chip FBGA	Bank 1A, Bank 3A, Bank 4A, Bank 6A, Bank 7A, and Bank 8A
	1517-pin flip chip FBGA	Bank 1A, Bank 2A, Bank 3A, Bank 4A, Bank 5A, Bank 6A, Bank 7A, and Bank 8A

- Arria II GX devices have dedicated configuration banks at Bank 3C and 8C, which support dedicated configuration pins and some of the dual-purpose pins with a configuration scheme at 1.8, 2.5, 3.0, and 3.3 V. For Arria II GZ devices, the dedicated configuration pins are located in Bank 1A and Bank 1C. However, these banks are not dedicated configuration banks; therefore, user I/O pins are available in Bank 1A and Bank 1C.
- Dedicated VCCIO, VREF, and VCCPD pin per I/O bank to allow voltage-referenced I/O standards. Each I/O bank can operate at independent V_{CCIO}, V_{REF}, and V_{CCPD} levels.

High-Speed LVDS I/O and DPA

- Dedicated circuitry for implementing LVDS interfaces at speeds from 150 Mbps to 1.25 Gbps
- R_D OCT 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 up to 1.25 Gbps data rate (SGMII and GbE)
- Emulated LVDS output buffers use two single-ended output buffers with an external resistor network to support LVDS, mini-LVDS, BLVDS (only for Arria II GZ devices), and RSDS standards.

Clock Management

- Provides dedicated global clock networks, regional clock networks, and periphery clock networks that are organized into a hierarchical structure that provides up to 192 unique clock domains
- Up to eight PLLs with 10 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
- FPGA fabric can use the unused transceiver PLLs 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 OCT and DQ/DQS 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, DDR3 SDRAM, RLDRAM II memory interface PHYs; the megafunction takes advantage of the PLL dynamic reconfiguration feature to calibrate based on the changes of process, voltage, and temperature (PVT).

For the maximum clock rates supported in Altera's FPGA devices, refer to the External Memory Interface Spec Estimator online tool.

For more information about the external memory interfaces support, refer to the *External Memory Interfaces in Arria II Devices* chapter.

Nios II

- Arria II 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
  - Supports active serial (AS), passive serial (PS), fast passive parallel (FPP), and JTAG configuration schemes.
- 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 FPP configuration mode with an external host (such as a MAX® II device or microprocessor), or when using the AS, FAS, or 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, 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 cyclic redundancy check (CRC) value
- You can identify and read out the bit location and the type of soft error through the JTAG or the core interface

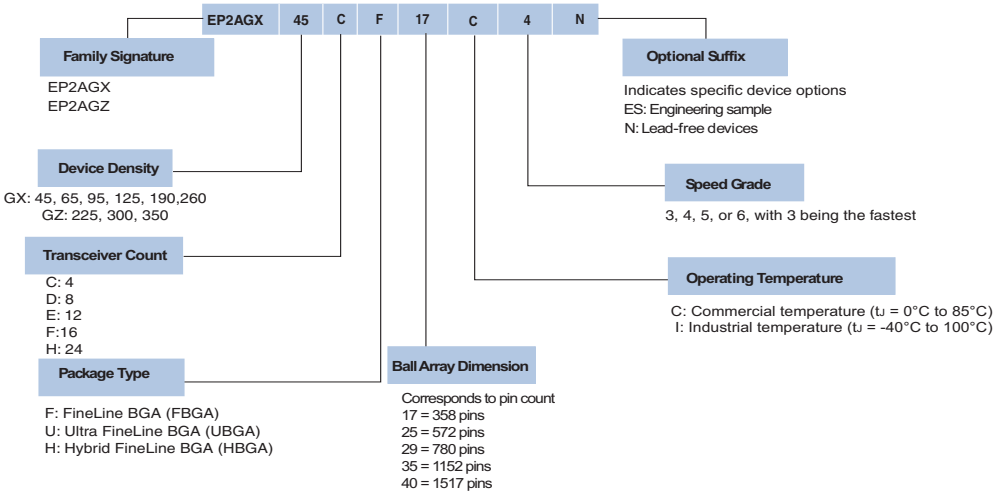
JTAG Boundary Scan Testing

- 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 1–3 shows the ordering codes for Arria II devices.

Figure 1–3. Packaging Ordering Information for Arria II Devices



Document Revision History

Table 1–10 lists the revision history for this chapter.

Table 1–10. Document Revision History (Part 1 of 2)

Date	Version	Changes
July 2012	4.4	Replaced Table 1-10. External Memory Interface Maximum Performance for Arria II Devices with link to the External Memory Interface Spec Estimator online tool.
December 2011	4.3	Updated Table 1–4 and Table 1–9.
June 2011	4.2	Updated Table 1–2.
June 2011	4.1	■ Updated Figure 1–2. ■ Updated Table 1–10. ■ Updated the “Arria II Device Feature” section. ■ Added Table 1–6. ■ Minor text edits.
December 2010	4.0	■ Updated for the Quartus II software version 10.0 release ■ Added information about Arria II GZ devices ■ Updated Table 1–1, Table 1–4, Table 1–5, Table 1–6, Table 1–7, and Table 1–9 ■ Added Table 1–3 ■ Added Figure 1–2 ■ Updated Figure 1–3 ■ Updated “Arria II Device Feature” and “Arria II Device Architecture” section

Table 1–10. Document Revision History (Part 2 of 2)

Date	Version	Changes
July 2010	3.0	Updated for the Quartus II software version 10.0 release: ■ Added information about –I3 speed grade ■ Updated Table 1–1, Table 1–3, and Table 1–7 ■ Updated Figure 1–2 ■ Updated "Highlights" and "High-Speed LVDS I/O and DPA"section ■ Minor text edits
November 2009	2.0	■ Updated Table 1–1, Table 1–2, and Table 1–3 ■ Updated "Configuration Features" section
June 2009	1.1	■ Updated Table 1–2. ■ Updated "I/O Features" section.
February 2009	1.0	Initial release.

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

AIIGX51002-2.0

This chapter describes the features of the logic array block (LAB) in the Arria® II core fabric. The LAB 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:

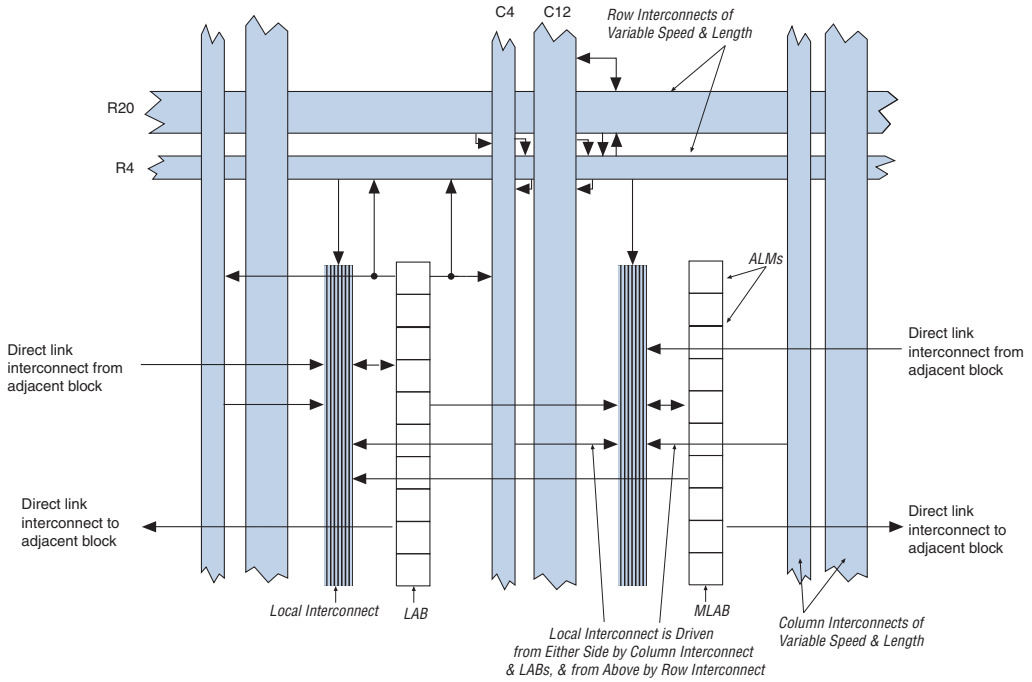
- "Logic Array Blocks" on page 2–1
- "Adaptive Logic Modules" on page 2–5

Logic Array Blocks

Each LAB consists of ten ALMs, various carry chains, shared arithmetic chains, LAB control signals, local interconnect, and register chain connection lines. The local interconnect transfers signals between ALMs in the same LAB. The direct link interconnect allows the LAB to drive into the local interconnect of its left and right neighbors. Register chain connections transfer the output of the ALM register to the adjacent ALM register in the LAB. The Quartus® II Compiler places associated logic in the LAB or the adjacent LABs, allowing the use of local, shared arithmetic chain, and register chain connections for performance and area efficiency.

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

Figure 2–1. LAB Structure in Arria II Devices

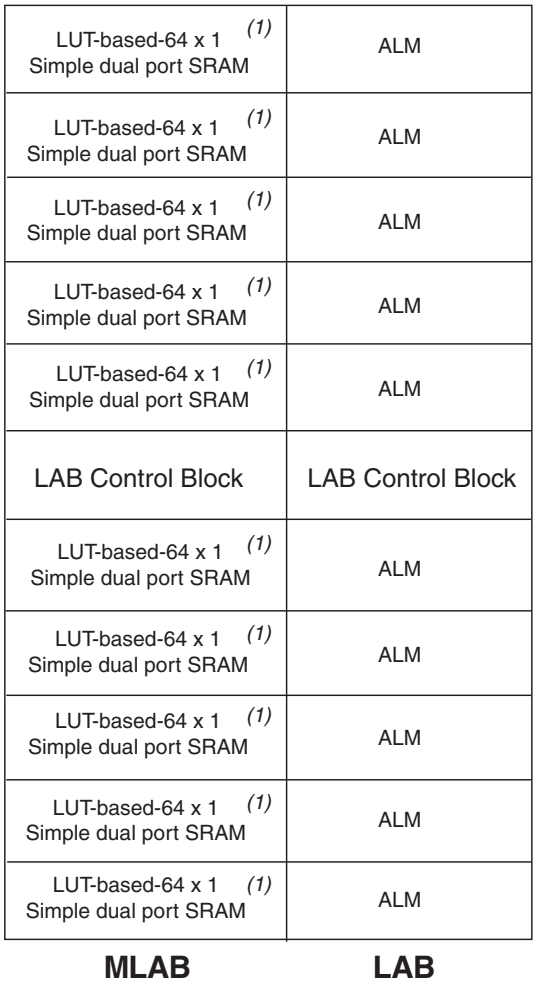


© 2010 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

The LAB of the Arria II device has a derivative called memory LAB (MLAB), which adds look-up table (LUT)-based SRAM capability to the LAB. The MLAB supports a maximum of 640 bits of simple dual-port 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 Arria II devices. MLAB is a superset of the LAB and includes all LAB features. Figure 2–2 shows an overview of LAB and MLAB topology.

For more information about MLABs, refer to the *TriMatrix Memory Blocks in Arria II Devices* chapter.

Figure 2–2. LAB and MLAB Structure in Arria II Devices



Note to Figure 2–2:

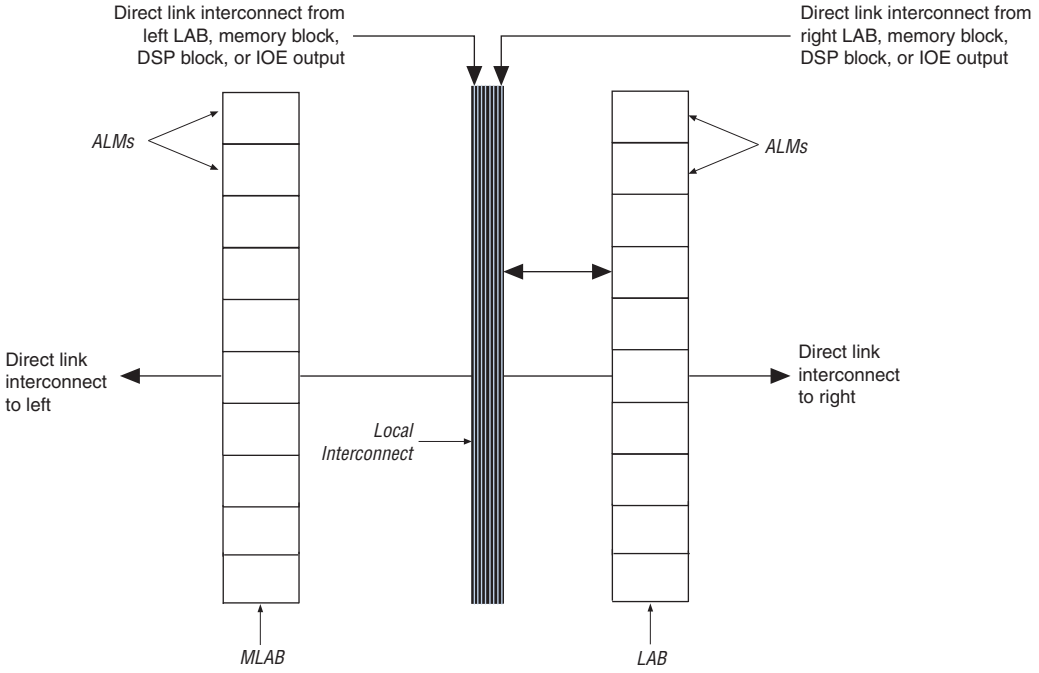
(1) You can use an MLAB ALM as a regular LAB ALM or configure it as a dual-port SRAM.

LAB Interconnects

The LAB local interconnect drives the ALMs in the same LAB using column and row interconnects and the ALM outputs in the same LAB. The direct link connection feature minimizes the use of row and column interconnects, providing higher performance and flexibility. Adjacent LABs/MLABs, memory blocks, or DSP blocks from the left or right can also drive the LAB's local interconnect through the direct link connection. 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.

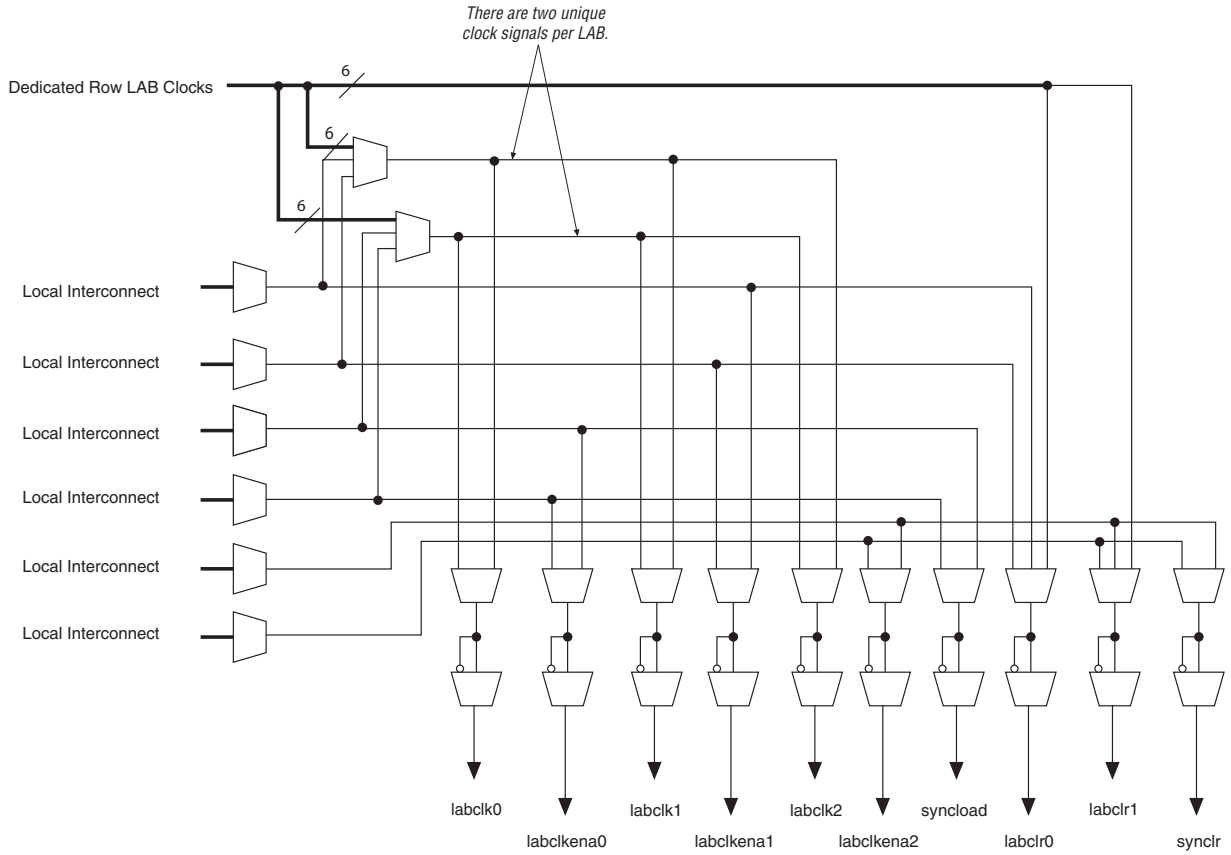
Figure 2–3. Direct Link Connection



LAB Control Signals

Each LAB contains dedicated logic for driving a maximum of 10 control signals to its ALMs at a time. Control signals include three clocks, three clock enables, two asynchronous clears, a synchronous clear, and synchronous load control signals. Although you generally use synchronous-load and clear signals when implementing counters, you can also use them with other functions. Each LAB 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 two clock sources and three clock enable signals. Each clock and clock enable signals are linked. For example, any ALM in a particular LAB using the `labclk1` signal also uses the `labclkena1` signal. If the LAB uses both the rising and falling edges of a clock, it also uses two LAB-wide clock signals. De-asserting the clock enable signal turns off the corresponding LAB-wide clock. The LAB row clocks [5..0] and LAB local interconnects generate the LAB-wide control signals. In addition to data, the inherent low skew of the MultiTrack interconnect allows clock and control signal distribution.

Figure 2–4. LAB-Wide Control Signals



Adaptive Logic Modules

The ALM is the basic building block of logic in the Arria II device architecture. Each ALM contains a variety of LUT-based resources that can be divided between two combinational adaptive LUTs (ALUTs) and two registers. With up to eight inputs for the two combinational ALUTs, one ALM can implement various combinations of two functions. This adaptability allows an ALM to be completely backward-compatible with 4-input LUT architectures. One ALM can also implement any function with up to 6-input and certain 7-input functions. In addition to the ALUT-based resources, each ALM contains two programmable registers, two dedicated full adders, a carry chain, a shared arithmetic chain, and a register chain. Through these dedicated resources, an ALM can efficiently implement various arithmetic functions and shift registers. Each ALM drives all types of interconnects: local, row, column, carry chain, shared arithmetic chain, register chain, and direct link. Figure 2–5 shows a high-level block diagram of the Arria II ALM.

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

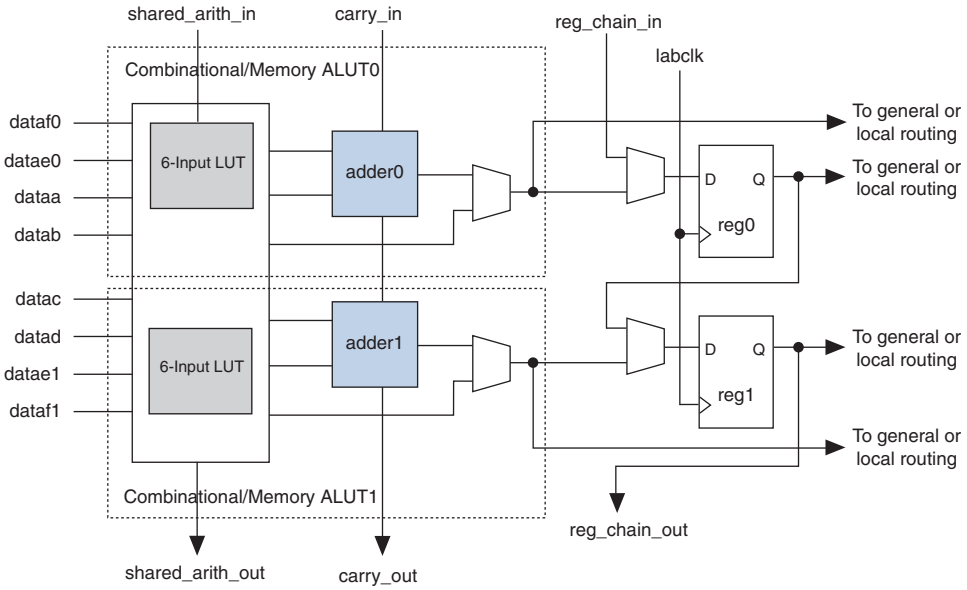
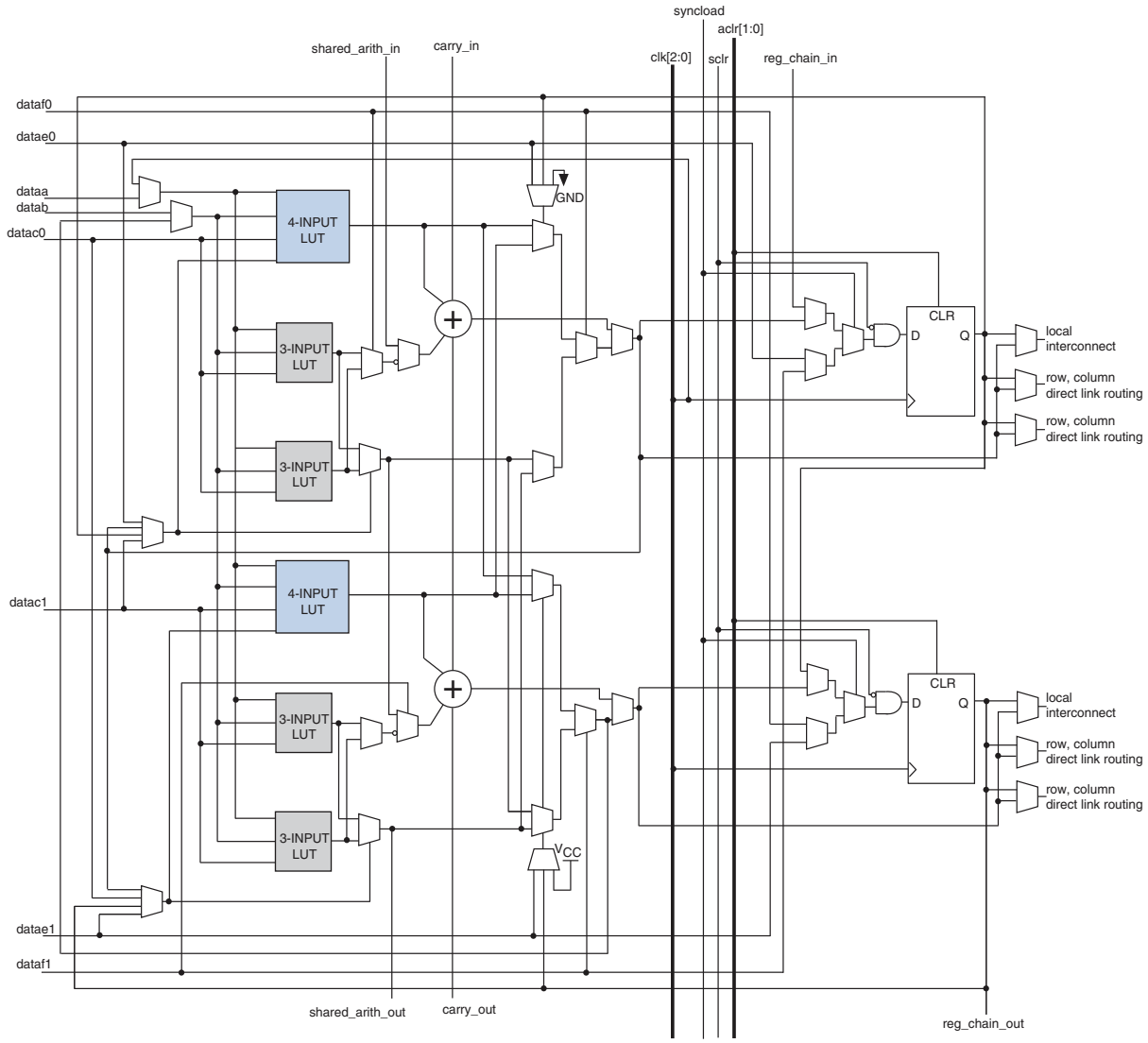


Figure 2–6 shows a detailed view of all the connections in an ALM.

Figure 2–6. Connection Details of the Arria II ALM



One ALM contains two programmable registers. Each register has data, clock, clock enable, synchronous and asynchronous clear, and synchronous load and clear inputs. Global signals, general purpose I/O (GPIO) pins, or any internal logic can drive the register's clock and clear-control signals. Either GPIO pins or internal logic can drive the clock enable. For combinational functions, the register is bypassed and the output of the 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 the ALM 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. The LUT or adder can drive one output while the register drives another output.

This feature is called register packing. It 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 ALM can operate in any of the following modes:

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

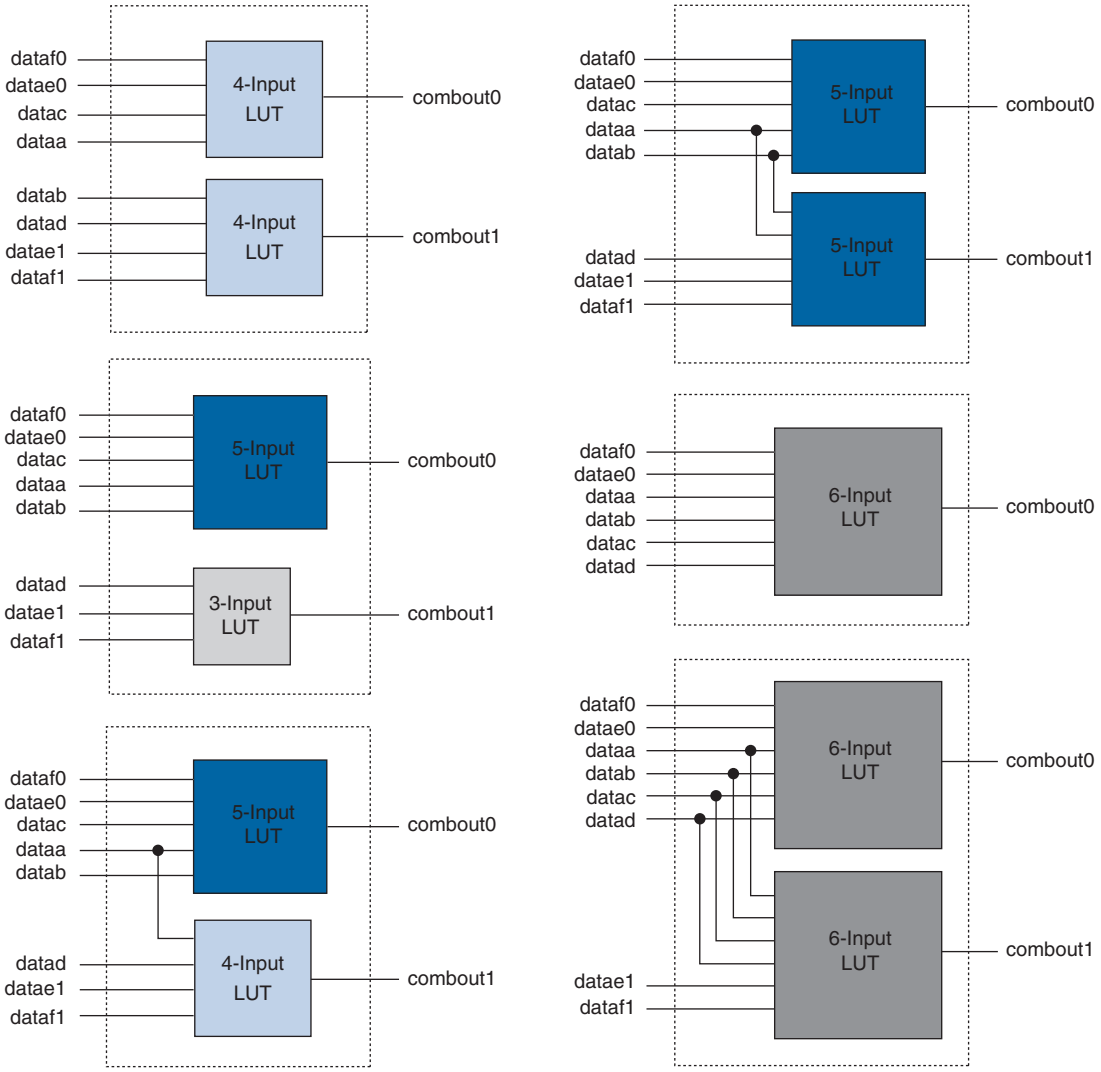
The Quartus II software and other supported third-party synthesis tools, in conjunction with parameterized functions such as the library of parameterized modules (LPM) functions, automatically choose the appropriate mode for common functions such as counters, adders, subtractors, and arithmetic functions. Each mode uses the ALM resources differently. In each mode, eleven available inputs to an ALM—the eight data inputs from the LAB local interconnect, carry-in from the previous ALM or LAB, the shared arithmetic chain connection from the previous ALM or LAB, and the register chain connection—are directed to different destinations to implement the desired logic function. LAB-wide signals provide clock, asynchronous clear, synchronous clear, synchronous load, and clock enable control for the register. These LAB-wide signals are available in all ALM modes. For more information on the LAB-wide control signals, refer to "LAB Control Signals" on page 2–4.

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 ALM, or 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.

Figure 2–7 shows the supported LUT combinations in normal mode.

Figure 2–7. ALM in Normal Mode ***(Note 1)***



Note to Figure 2–7:

(1) Combinations of functions with fewer inputs than those shown are also supported. For example, combinations of functions with the following number of inputs are supported: 4 and 3, 3 and 3, 3 and 2, and 5 and 2.

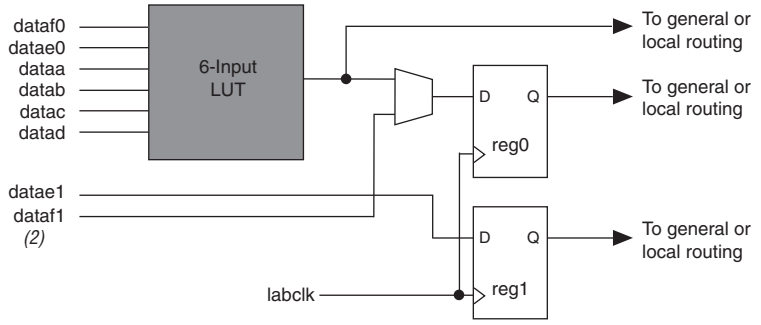
Normal mode provides complete backward-compatibility with 4-input LUT architectures.

For the packing of two 5-input functions into one ALM, the functions must have at least two common inputs. The common inputs are `dataa` and `datab`. The combination of a 4-input function with a 5-input function requires one common input (either `dataa` or `datab`).

In the case of implementing two 6-input functions in one ALM, four inputs must be shared and the combinational function must be the same. In a sparsely used device, functions that could be placed in one ALM may be implemented in separate ALMs by the Quartus II software to achieve the best possible performance. As a device begins to fill up, the Quartus II software automatically utilizes the full potential of the Arria II ALM. The Quartus II Compiler automatically searches for functions using common inputs or completely independent functions to be placed in one ALM to make efficient use of device resources. In addition, you can manually control resource usage by setting location assignments.

Any 6-input function can be implemented using inputs `dataa`, `datab`, `datac`, `datad`, and either `datae0` and `dataf0` or `datae1` and `dataf1`. If `datae0` and `dataf0` are utilized, the output is driven to `register0`, and/or `register0` is bypassed and the data drives out to the interconnect using the top set of output drivers (refer to Figure 2–8). If `datae1` and `dataf1` are used, the output either drives to `register1` or bypasses `register1` and drives to the interconnect using the bottom set of output drivers. The Quartus II Compiler automatically selects the inputs to the LUT. ALMs in normal mode support register packing.

Figure 2–8. Input Function in Normal Mode ***(Note 1)***



Notes to Figure 2–8:

(1) If `datae1` and `dataf1` are used as inputs to a 6-input function, `datae0` and `dataf0` are available for register packing.

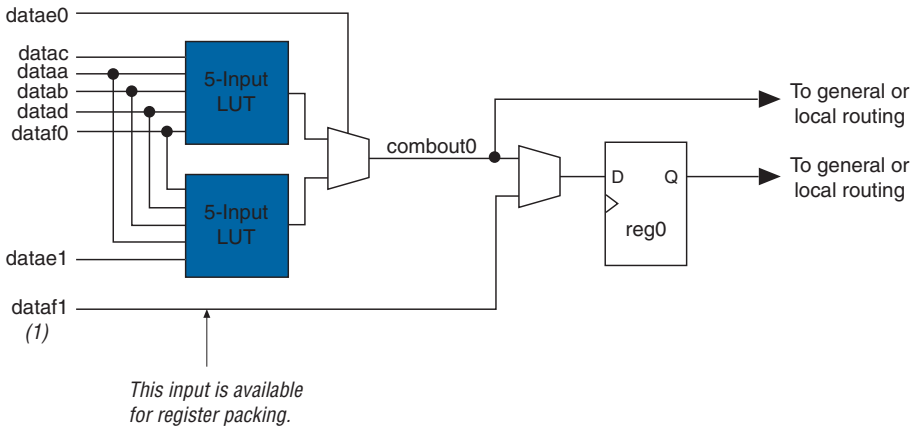
(2) The `dataf1` input is available for register packing only if the 6-input function is unregistered.

Extended LUT Mode

Use extended LUT mode to implement a specific set of 7-input functions. The set must be a 2-to-1 multiplexer fed by two arbitrary 5-input functions sharing four inputs. Figure 2–9 shows the template of supported 7-input functions using extended LUT mode. In this mode, if the 7-input function is unregistered, the unused eighth input is available for register packing.

Functions that fit into the template, as shown in Figure 2–9, often appear in designs as "if-else" statements in Verilog HDL or VHDL code.

Figure 2–9. Template for Supported 7-Input Functions in Extended LUT Mode



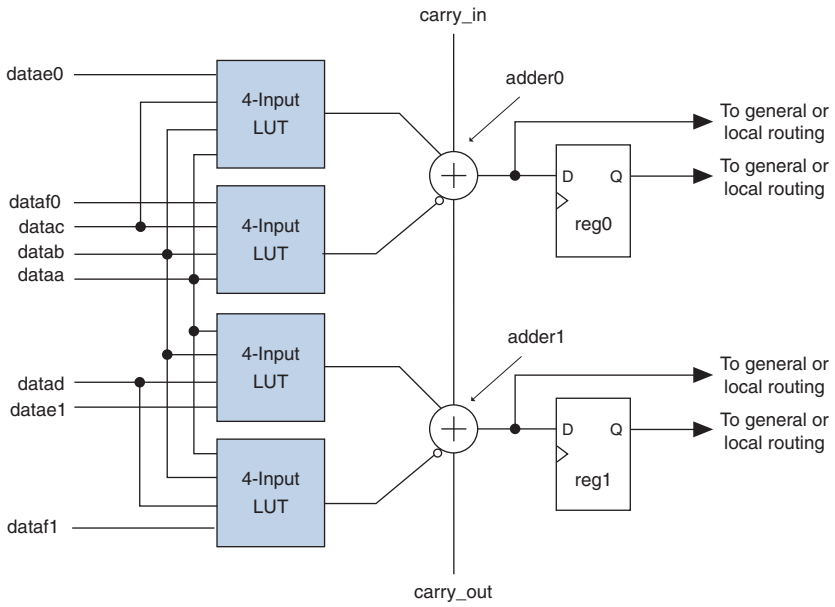
Note to Figure 2–9:

(1) If the 7-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 two 4-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 two 4-input functions. The four LUTs share `dataa` and `datab` inputs. As shown in Figure 2–10, the carry-in signal feeds to `adder0` and the carry-out from `adder0` feeds to the carry-in of `adder1`. The carry-out from `adder1` drives to `adder0` of the next ALM in the LAB. ALMs in arithmetic mode can drive out registered and unregistered versions of the adder outputs.

Figure 2–10. ALM in Arithmetic Mode



In arithmetic mode, the ALM supports simultaneous use of the adder's carry output along with combinational logic outputs. The adder output is ignored in this operation. Using the adder with combinational logic output provides resource savings of up to 50% for functions that can use this mode.

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 share 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 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 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. To enhance 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.

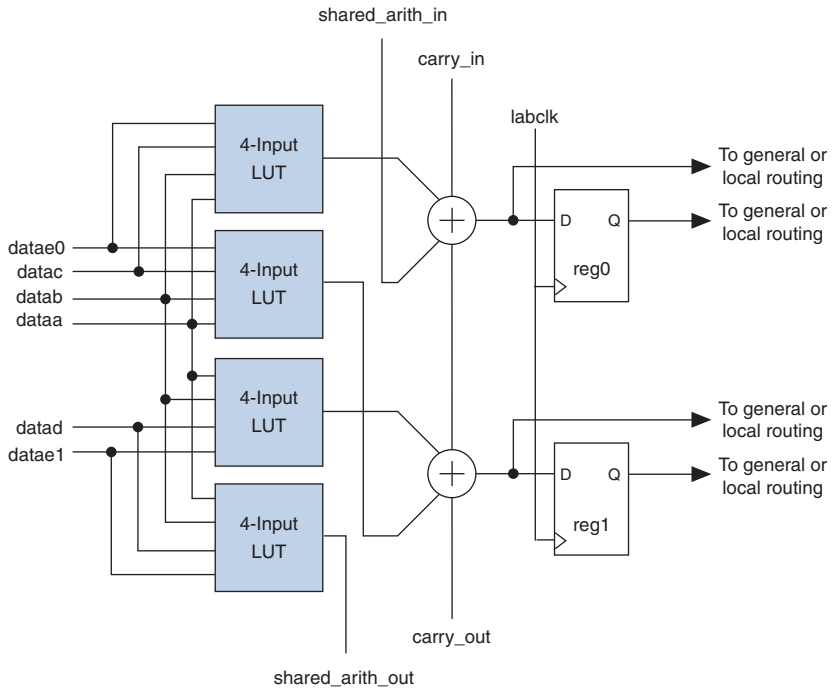


For more information on carry chain interconnect, refer to "ALM Interconnects" on page 2–17.

Shared Arithmetic Mode

In shared arithmetic mode, the ALM can implement a 3-input add in an ALM. In this mode, the ALM is configured with four 4-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 2–11 shows the ALM using this feature.

Figure 2–11. ALM in Shared Arithmetic Mode



You can find adder trees in many different applications. For example, the summation of the partial products in a logic-based multiplier can be implemented in a tree structure. Another example is a correlator function that can use a large adder tree to sum filtered data samples in a given time frame to recover or de-spread data that was transmitted using spread-spectrum technology.

Shared Arithmetic Chain

The shared arithmetic chain available in enhanced arithmetic mode allows the ALM to implement a 3-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. To enhance fitting, a long shared arithmetic chain runs vertically, allowing fast horizontal connections to the TriMatrix 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 2–17.

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 2–12 shows the register constructed using two combinational blocks in the ALM.

Figure 2–12. LUT Register from Two Combinational Blocks

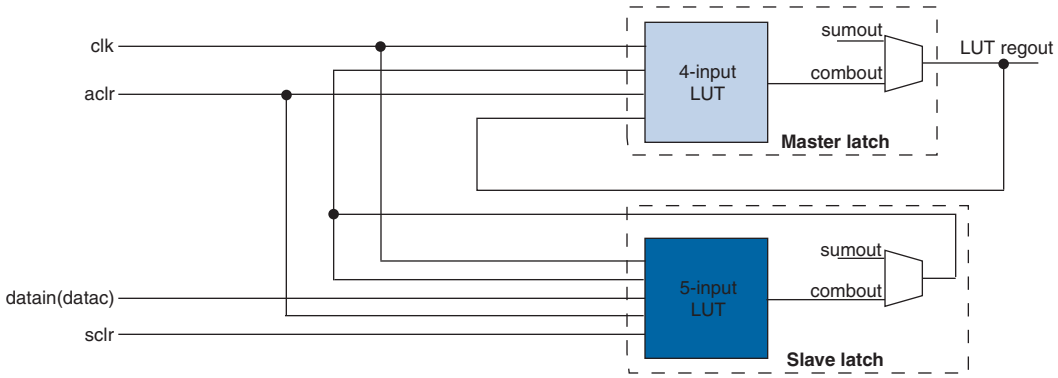
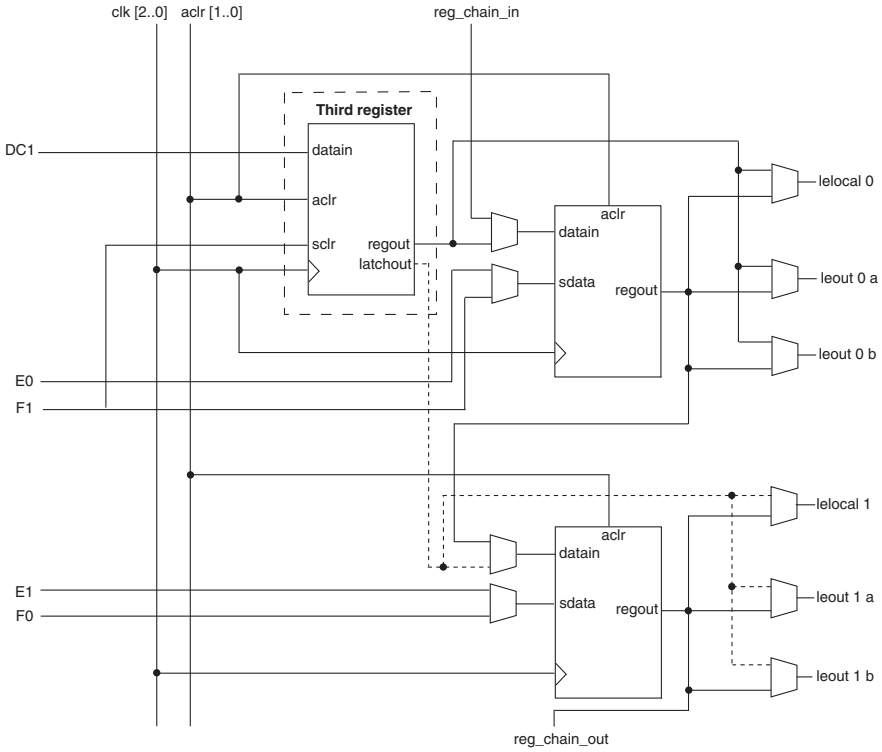


Figure 2–13 shows the ALM in LUT-Register mode.

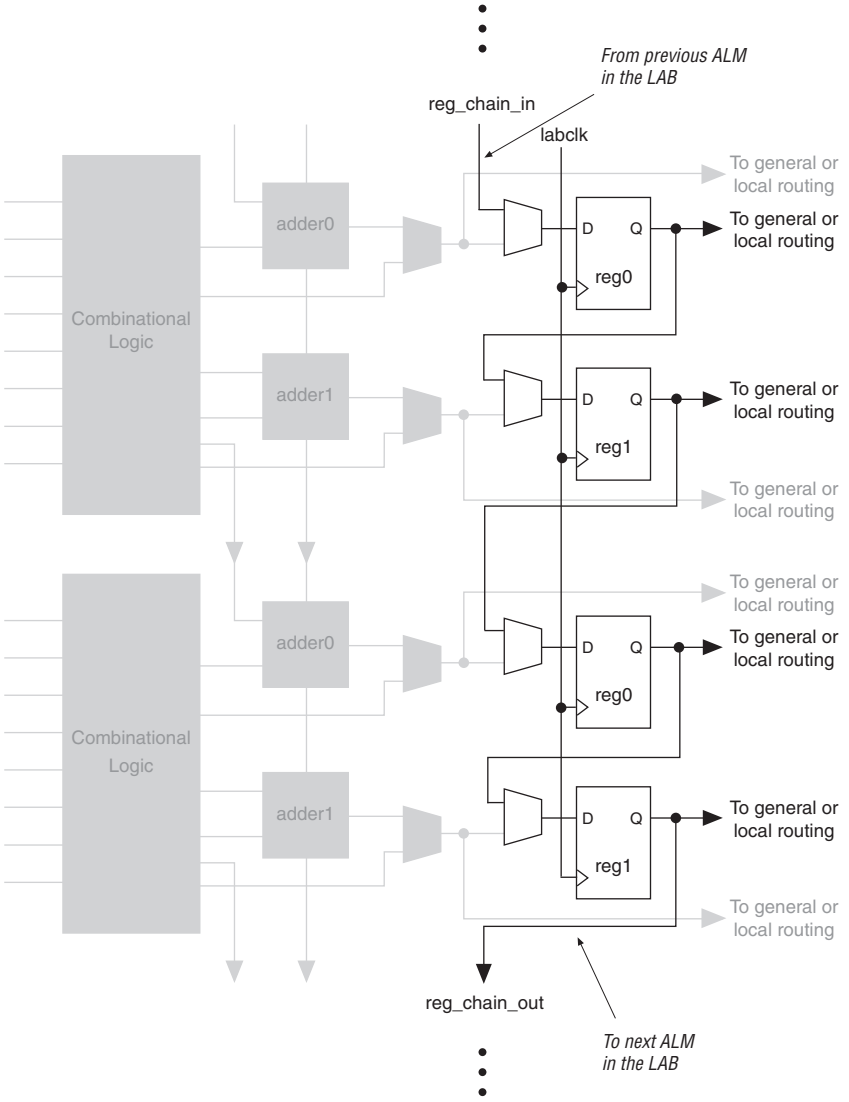
Figure 2–13. ALM in LUT-Register Mode with 3-Register Capability



Register Chain

In addition to general routing outputs, the ALMs in any given LAB have register chain outputs to allow 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 2–14). The Quartus II Compiler automatically takes advantage of these resources to improve utilization and performance.

Figure 2–14. Register Chain in an LAB ***(Note 1)***



Note to Figure 2–14:

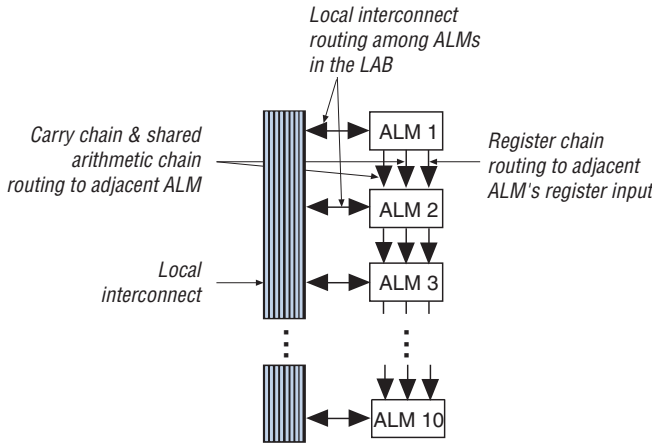
(1) You can use the combinational or adder logic to implement an unrelated, un-registered function.

For more information about register chain interconnect, refer to "ALM Interconnects" on page 2–17.

ALM Interconnects

There are three dedicated paths between ALMs: Register Cascade, Carry-chain, and Shared Arithmetic chain. Arria II 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 2–15 shows the shared arithmetic chain, carry chain, and register chain interconnects.

Figure 2–15. Shared Arithmetic Chain, Carry Chain, and Register Chain Interconnects



Clear and Preset Logic Control

LAB-wide signals control the logic for the register's clear signal. 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 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.

LAB Power Management Techniques

The following techniques are used to manage static and dynamic power consumption within the LAB:

- The Quartus II software forces all adder inputs low when ALM adders are not in use to save AC power.
- Arria II LABs operate in high-performance mode or low-power mode. The Quartus II software automatically chooses the appropriate mode for the LAB, based on the design, to optimize speed versus leakage trade-offs.

- Clocks represent a significant portion of dynamic power consumption due to their high switching activity and long paths. The LAB clock that distributes a clock signal to registers within an LAB is a significant contributor to overall clock power consumption. Each LAB's clock and clock enable signal are linked. For example, a combinational ALUT or register in a particular LAB using the `labclk1` signal also uses the `labclkena1` signal. To disable an LAB-wide clock power consumption without disabling the entire clock tree, use the LAB-wide clock enable to gate the LAB-wide clock. The Quartus II software automatically promotes register-level clock enable signals to the LAB-level. All registers within the LAB that share a common clock and clock enable are controlled by a shared, gated clock. To take advantage of these clock enables, use a clock-enable construct in your HDL code for the registered logic.

For more information about implementing static and dynamic power consumption within the LAB, refer to the *Power Optimization* chapter in volume 2 of the *Quartus II Handbook*.

Document Revision History

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

Table 2–1. Document Revision History

Date	Version	Changes
December 2010	2.0	Updated for the Quartus II software version 10.1 release: ■ Added Arria II GZ device information. ■ Updated "Logic Array Blocks", "LAB Interconnects", "LAB Control Signals", "Adaptive Logic Modules", "ALM Operating Modes", "Normal Mode" sections. ■ Added Figure 2–7 and Figure 2–8. ■ Added "LAB Power Management Techniques" section.
June 2009	1.1	Updated Figure 2–6.
February 2009	1.0	Initial Release.

3. Memory Blocks in Arria II Devices

AIIGX51003-3.2

This chapter describes the Arria® II device memory blocks that include 640-bit memory logic array blocks (MLABs), 9-Kbit M9K blocks, and 144-Kbit M144K blocks. MLABs are optimized to implement filter delay lines, small FIFO buffers, and shift registers. You can use the M9K blocks for general purpose memory applications and the M144K blocks for processor code storage, packet buffering, and video frame buffering.

☞ M144K block is only available for Arria II GZ devices.

You can configure each embedded memory block independently with the Quartus® II MegaWizard™ Plug-In Manager to be a single- or dual-port RAM, FIFO, ROM, or shift register. You can stitch together multiple blocks of the same type to produce larger memories with a minimal timing penalty.

This chapter contains the following sections:

- "Memory Features" on page 3–2
- "Memory Modes" on page 3–10
- "Clocking Modes" on page 3–19
- "Design Considerations" on page 3–20

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Memory Features

Table 3–1 lists the features supported by the embedded memory blocks.

Table 3–1. Summary of Memory Features in Arria II Devices (Part 1 of 2)

Feature	MLABs		M9K Blocks		M144K Blocks
	Arria II GX	Arria II GZ	Arria II GX	Arria II GZ	Arria II GZ
Maximum performance	500 MHz	500 MHz	390 MHz	540 MHz	500 MHz
Total RAM bits (including parity bits)	640	640	9,216	9,216	147,456
Configurations (depth × width)	64 × 8 64 × 9 64 × 10 32 × 16 32 × 18 32 × 20	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	8K × 1 4K × 2 2K × 4 1K × 8 1K × 9 512 × 16 512 × 18 256 × 32 256 × 36	16K × 8 16K × 9 8K × 16 8K × 18 4K × 32 4K × 36 2K × 64 2K × 72
Parity bits	✓	✓	✓	✓	✓
Byte enable	✓	✓	✓	✓	✓
Packed mode	—	—	✓	✓	✓
Address clock enable	✓	✓	✓	✓	✓
Single-port memory	✓	✓	✓	✓	✓
Simple dual-port memory	✓	✓	✓	✓	✓
True dual-port memory	—	—	✓	✓	✓
Embedded shift register	✓	✓	✓	✓	✓
ROM	✓	✓	✓	✓	✓
FIFO buffer	✓	✓	✓	✓	✓
Simple dual-port mixed width support	—	—	✓	✓	✓
True dual-port mixed width support	—	—	✓	✓	✓
Memory initialization file (**.mif**)	✓	✓	✓	✓	✓
Mixed-clock mode	✓	✓	✓	✓	✓
Power-up condition	Outputs cleared if registered, otherwise reads memory contents.		Outputs cleared		Outputs cleared
Register clears	Output registers		Output registers		Output registers
Write/Read operation triggering	Write: Falling clock edges. Read: Rising clock edges		Write and Read: Rising clock edges		Write and Read: Rising clock edges
Same-port read-during-write	Outputs set to **old data**	Outputs set to **don't care**	Outputs set to **old data** or **new data**		Outputs set to **old data** or **new data**

Table 3–1. Summary of Memory Features in Arria II Devices (Part 2 of 2)

Feature	MLABs		M9K Blocks		M144K Blocks
	Arria II GX	Arria II GZ	Arria II GX	Arria II GZ	Arria II GZ
Mixed-port read-during-write	Outputs set to **old data**, **new data**, or **don't care**		Outputs set to **old data** or **don't care**		Outputs set to **old data** or **don't care**
ECC Support	Soft IP support using the Quartus II software		Soft IP support using the Quartus II software		Built-in support in ×64-wide simple dual-port mode or soft IP support using the Quartus II software

Table 3–2 lists the capacity and distribution of the memory blocks in each Arria II device.

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

Device	MLABs	M9K Blocks	M144K	Total RAM Bits (including MLABs) (Kbits)
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
EP2AGZ225	4,480	1,235	—	13,915
EP2AGZ300	5,960	1,248	24	18,413
EP2AGZ350	6,970	1,248	36	20,772

Memory Block Types

M9K and M144K memory blocks are dedicated resources. MLABs are dual-purpose blocks. You can configure the MLABs as regular logic array blocks (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 write operations of the RAM blocks.

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 and M144K blocks, the byte enable controls all 9 bits (8 bits of data plus 1 parity bit). When using parity bits on the MLAB, the byte-enable controls all 10 bits in the widest mode.

Byte enables are only supported for true dual-port memory configurations when both the PortA and PortB data widths of the individual M9K memory blocks are multiples of 8 or 9 bits. For example, you cannot use byte enable for a mixed data width memory configured with portA=32 and portB=8 because the mixed data width memory is implemented as 2 separate 16 x 4 bit memories.

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 you use 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.



You cannot use the byte enable feature when using the error correction coding (ECC) feature on M144K blocks.

Figure 3–1 shows how the write enable (`wren`) and byte enable (`byteena`) signals control the operations of the M9K and M144K memory blocks.

When a byte-enable bit is deasserted 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. Byte Enable Functional Waveform for M9K and M144K

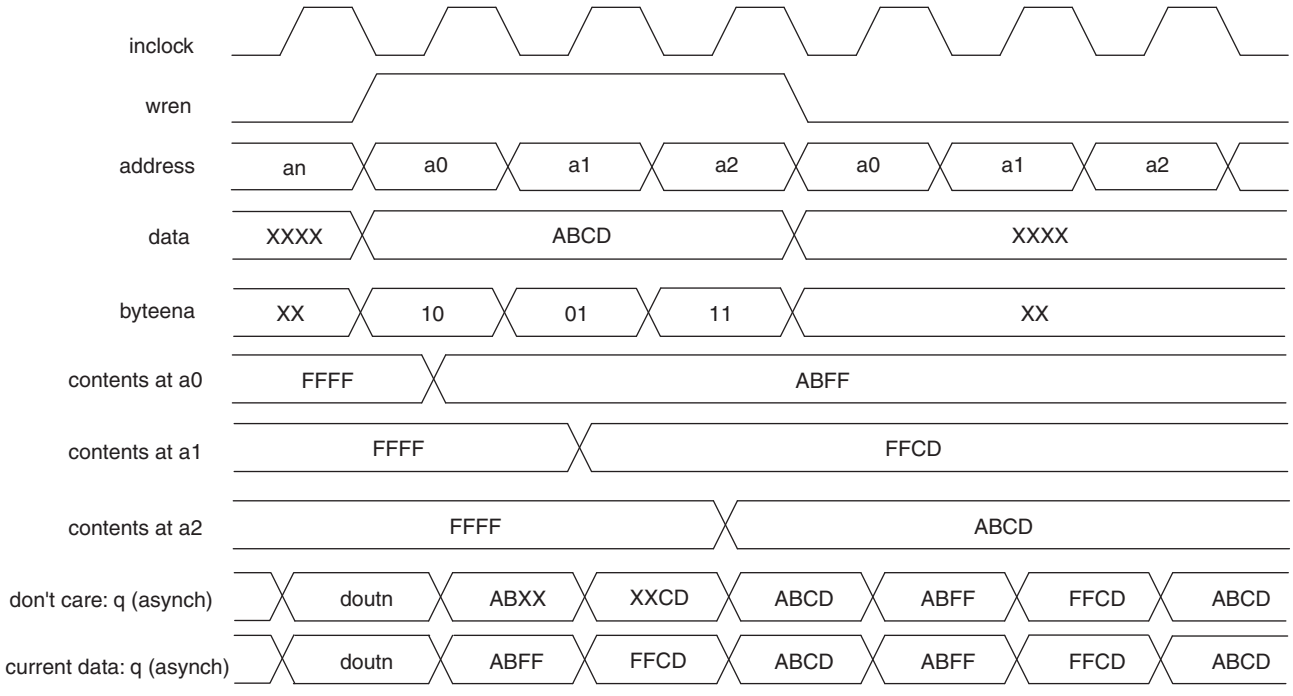
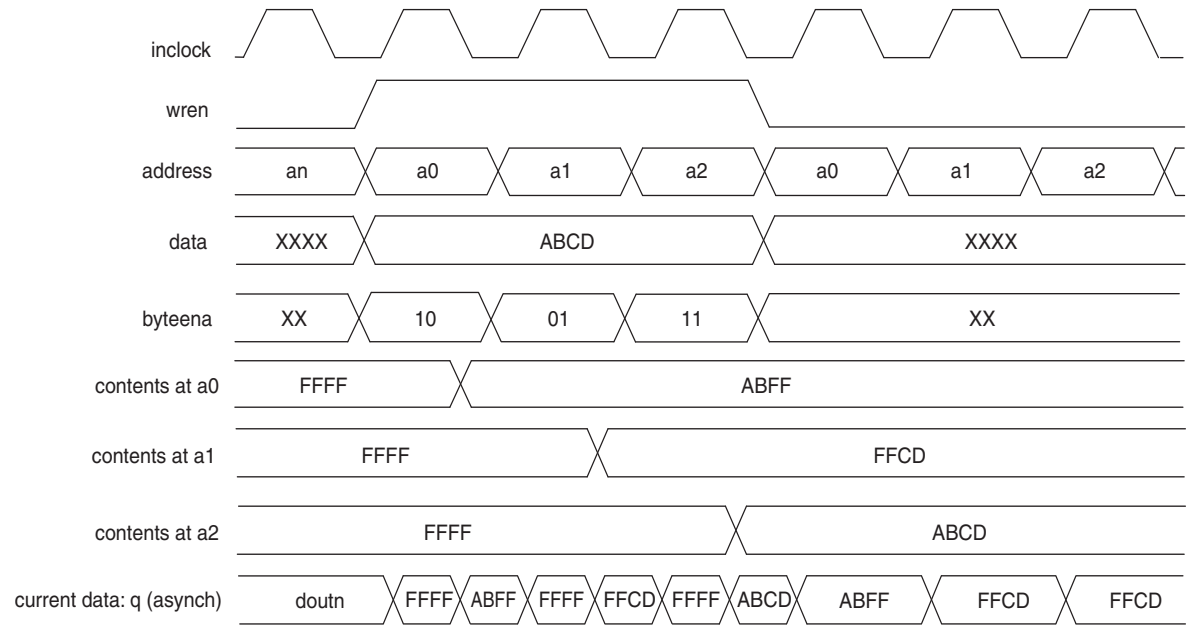


Figure 3–2 shows how the wren and byteena signals control the operations of the MLABs. Falling clock edges triggers the write operation in MLABs.

Figure 3–2. Byte Enable Functional Waveform for MLABs



Packed Mode Support

Arria II M9K and M144K blocks support packed mode. The packed mode feature packs two independent single-port RAMs into one memory block. The Quartus II software automatically implements the 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

Arria II 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 signal is low (disabled).

Figure 3–3 shows an address clock enable block diagram. The port name `addressstall` refers to the address clock enable.

Figure 3–3. Address Clock Enable

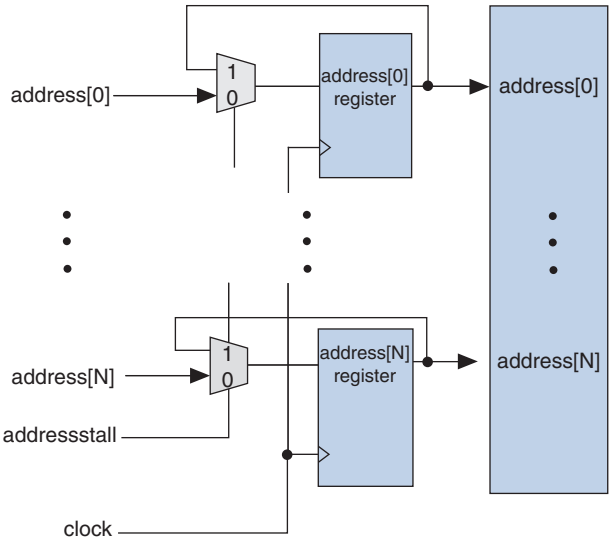


Figure 3–4 shows the address clock enable waveform during the read cycle.

Figure 3–4. Address Clock Enable During Read Cycle Waveform

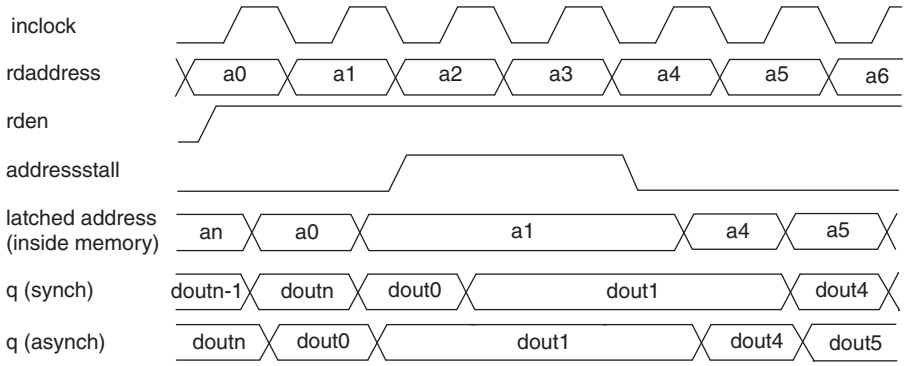


Figure 3–5 shows the address clock enable waveform during write cycle for M9K and M144K blocks.

Figure 3–5. Address Clock Enable During Write Cycle Waveform for M9K and M144K Blocks

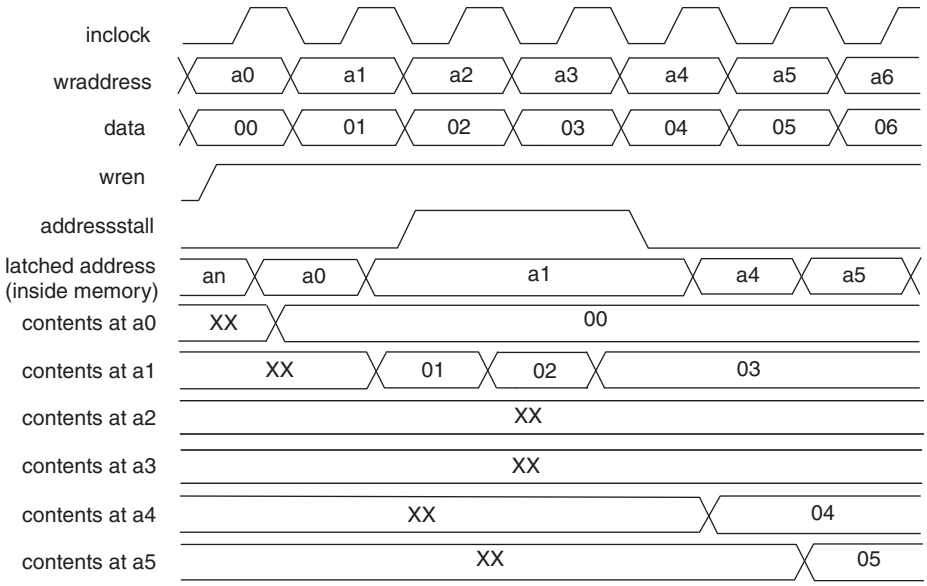
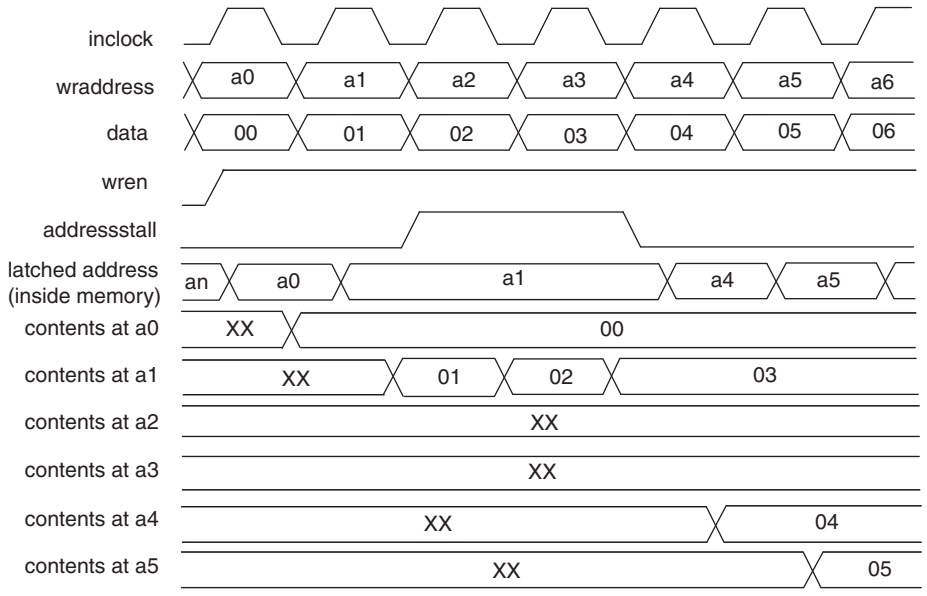


Figure 3–6 shows the address clock enable waveform during the write cycle for MLABs.

Figure 3–6. Address Clock Enable During Write Cycle Waveform for MLABs



Mixed Width Support

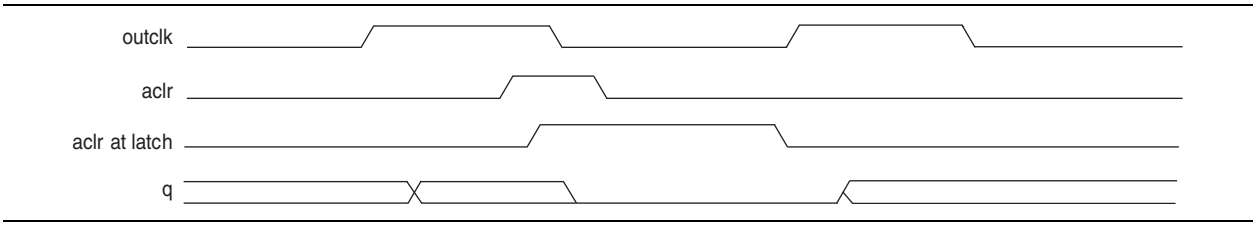
M9K and M144K blocks support mixed data widths inherently. MLABs can support mixed data widths through emulation with the Quartus II software. When using simple dual-port, true dual-port, or FIFO modes, mixed width support allows you to read and write different data widths to a memory block. For more information about the different widths supported per memory mode, refer to "Memory Modes" on page 3–10.

MLABs do not support mixed-width FIFO mode.

Asynchronous Clear

Arria II 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 3–7 shows a functional waveform showing this functionality.

Figure 3–7. Output Latch Asynchronous Clear Waveform



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

For more information about the RAM MegaWizard Plug-In Manager, refer to the *Internal Memory (RAM and ROM) Megafunction User Guide*.

Error Correction Code Support

Arria II GZ M144K blocks have built-in support for ECC when in ×64-wide simple dual-port mode. ECC allows you to detect and correct data errors in the memory array. The M144K blocks have a single-error-correction double-error-detection (SECDED) implementation. SECDED can detect and fix a single bit error in a 64-bit word, or detect two bit errors in a 64-bit word. It cannot detect three or more errors.

The M144K ECC status is communicated using a three-bit status flag (`eccstatus[2..0]`). The status flag can be either registered or unregistered. When registered, it uses the same clock and asynchronous clear signals as the output registers. When unregistered, it cannot be asynchronously cleared.

Table 3–3 lists the truth table for the ECC status flags.

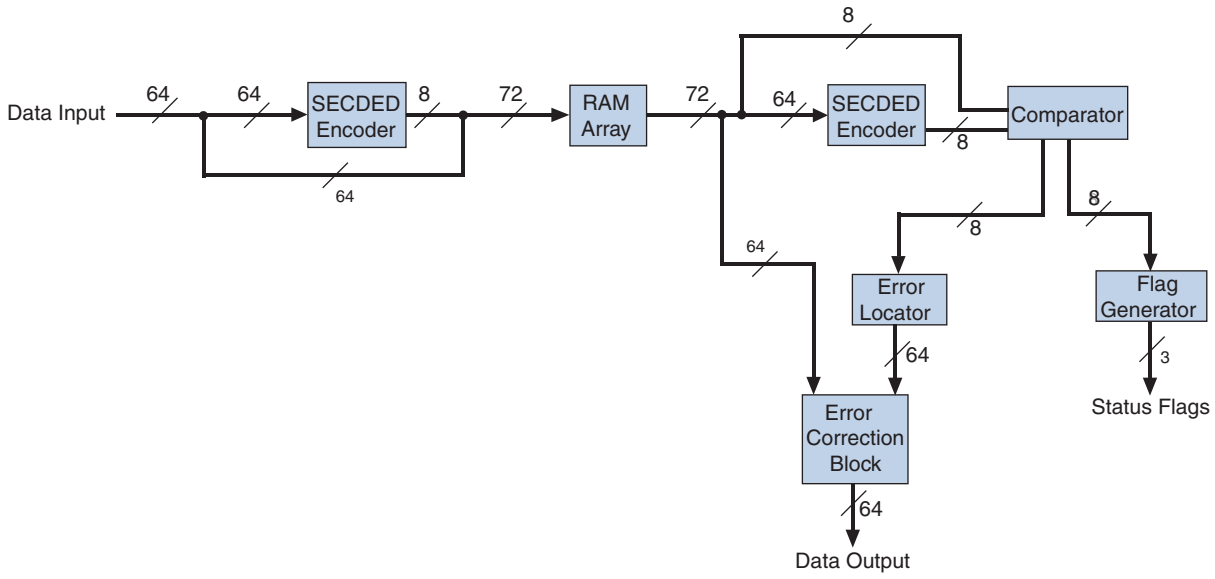
Table 3–3. Truth Table for ECC Status Flags in Arria II Devices

Status	eccstatus[2]	eccstatus[1]	eccstatus[0]
No error	0	0	0
Single error and fixed	0	1	1
Double error and no fix	1	0	1
Illegal	0	0	1
Illegal	0	1	0
Illegal	1	0	0
Illegal	1	1	X

- You cannot use the byte enable feature when ECC is engaged.
- Read-during-write old data mode is not supported when ECC is engaged.

Figure 3–8 shows a diagram of the ECC block of the M144K block.

Figure 3–8. ECC Block Diagram of the M144K Block



Memory Modes

Arria II memory blocks allow you to implement fully synchronous SRAM memory in multiple modes of operation. M9K and M144K 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 RAM Mode" on page 3–10
- "Simple Dual-Port Mode" on page 3–12
- "True Dual-Port Mode" on page 3–15
- "Shift-Register Mode" on page 3–17
- "ROM Mode" on page 3–18
- "FIFO Mode" on page 3–18

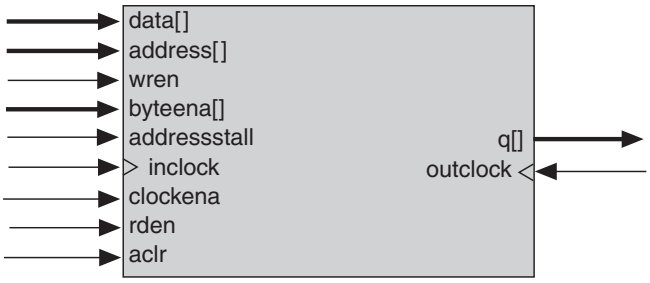
To choose the desired read-during-write behavior, set the read-during-write behavior to either **new data**, **old data**, or **don't care** in the RAM MegaWizard Plug-In Manager in the Quartus II software. For more information about this behavior, refer to "Read-During-Write Behavior" on page 3–21.

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 Mode

All memory blocks support single-port mode. Single-port mode allows you to do either a one-read or a one-write operation at a time. Simultaneous reads and writes are not supported in single-port mode. Figure 3–9 shows the single-port RAM configuration.

Figure 3–9. Single-Port Memory ***(Note 1)***



Note to Figure 3–9:

(1) You can implement two single-port memory blocks in a single M9K and M144K blocks. For more information, refer to "Packed Mode Support" on page 3–5.

During a write operation, the RAM output behavior is configurable. If you use the read-enable signal and perform a write operation with the read enable deactivated, 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 do not use the read-enable signal at all, the RAM outputs show the "new data" being written, the "old data" at that address, or a "don't care" value.

Table 3–4 lists the possible port width configurations for memory blocks in single-port mode.

Table 3–4. Port Width Configurations for MLABs, M9K, and M144K Blocks (Single-Port Mode)

Port Width Configurations		
MLABs	**M9K Blocks**	**M144K Blocks**
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	16K × 8 16K × 9 8K × 16 8K × 18 4K × 32 4K × 36 2K × 64 2K × 72

Figure 3–10 shows timing waveforms for read and write operations in single-port mode with unregistered outputs for M9K and M144K blocks. Registering the M9K and M144K block outputs delay the q output by one clock cycle.

Figure 3–10. Timing Waveform for Read-Write Operations for M9K and M144K Blocks (Single-Port Mode)

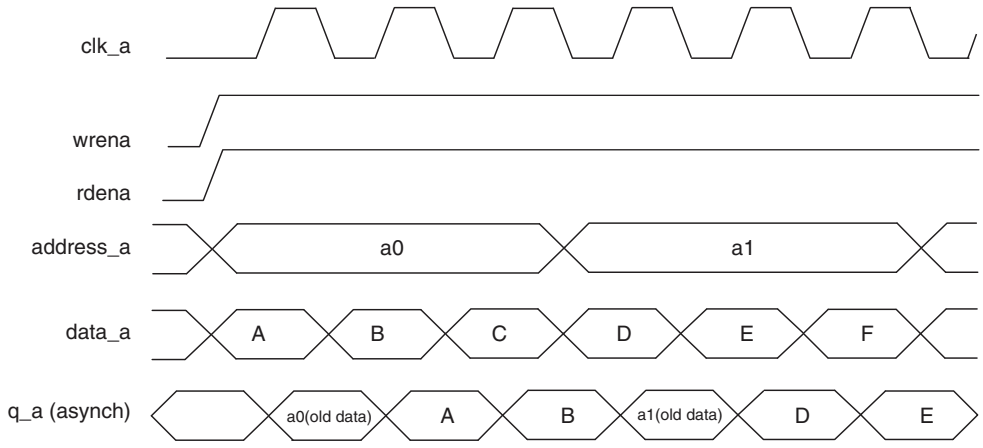
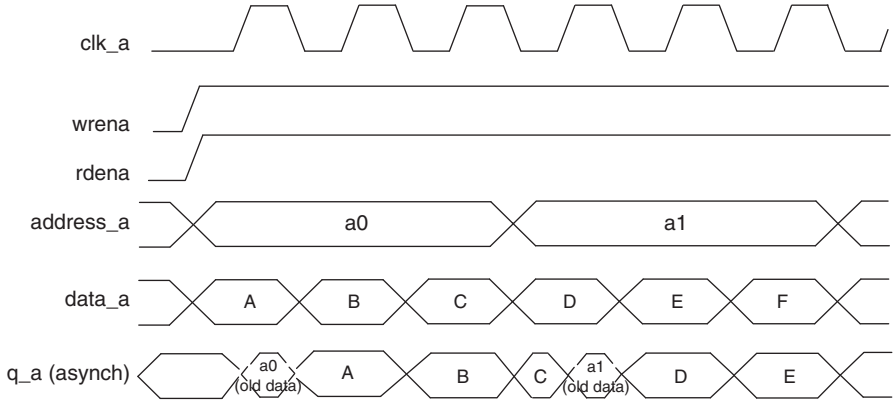


Figure 3–11 shows the timing waveforms for read and write operations in single-port mode with unregistered outputs for the MLAB. The rising clock edges trigger the read operation whereas the falling clock edges triggers the write operation.

Figure 3–11. Timing Waveform for Read-Write Operations for MLABs (Single-Port Mode)

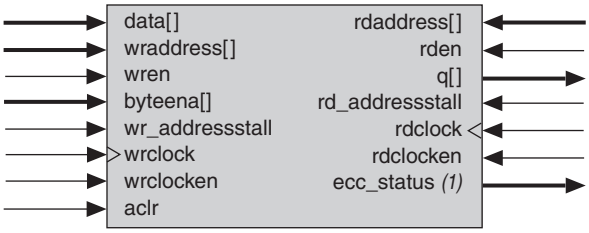


Simple Dual-Port Mode

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. The write operation occurs on port A; the read operation occurs on port B.

Figure 3–12 shows a simple dual-port configuration. Simple dual-port RAM supports input and output clock mode in addition to the read and write clock mode.

Figure 3–12. Arria II Simple Dual-Port Memory



Note to Figure 3–12:

(1) Only available for Arria II GZ devices.

Simple dual-port mode supports different read and write data widths (mixed width support). Table 3–5 lists the mixed width configurations for the M9K blocks in simple dual-port mode. MLABs do not have native support for mixed width operations. The Quartus II software can implement mixed width memories in MLABs with more than one MLAB.

Table 3–5. M9K Block Mixed-Width Configurations (Simple Dual-Port Mode)

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

Table 3–6 lists the mixed-width configurations for M144K blocks in simple dual-port mode.

Table 3–6. M144K Block Mixed-Width Configurations (Simple Dual-Port Mode)

Read Port	Write Port							
	16K × 8	8K × 16	4K × 32	2K × 64	16K × 9	8K × 18	4K × 36	2K × 72
16K × 8	✓	✓	✓	✓	—	—	—	—
8K × 16	✓	✓	✓	✓	—	—	—	—
4K × 32	✓	✓	✓	✓	—	—	—	—
2K × 64	✓	✓	✓	✓	—	—	—	—
16K × 9	—	—	—	—	✓	✓	✓	✓
8K × 18	—	—	—	—	✓	✓	✓	✓
4K × 36	—	—	—	—	✓	✓	✓	✓
2K × 72	—	—	—	—	✓	✓	✓	✓

In simple dual-port mode, M9K and M144K blocks support separate write-enable and read-enable signals. Read-during-write operations to the same address can either output a “don’t care” or “old data” value.

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

Figure 3–13 shows timing waveforms for read and write operations in simple dual-port mode with unregistered outputs for M9K and M144K blocks. Registering the M9K and M144K block outputs delay the q output by one clock cycle.

Figure 3–13. Simple Dual-Port Timing Waveforms for M9K and M144K Blocks

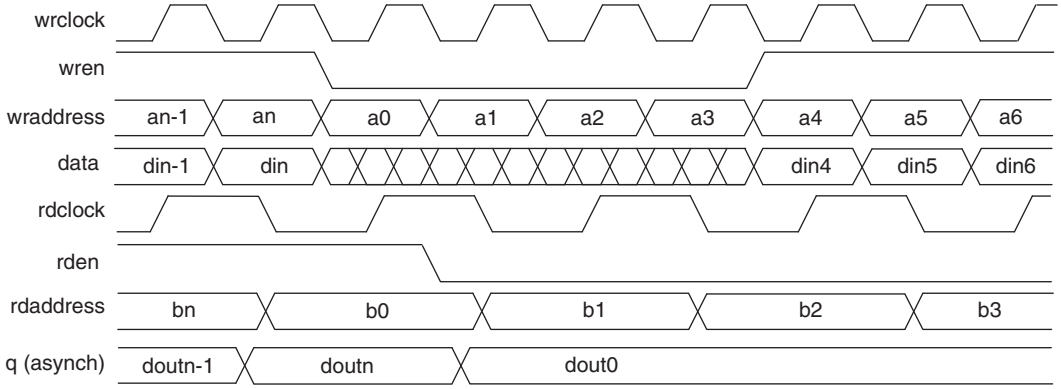


Figure 3–14 shows the timing waveforms for read and write operations in simple dual-port mode with unregistered outputs in the MLAB. The write operation is triggered by the falling clock edges.

Figure 3–14. Simple Dual-Port Timing Waveforms for MLABs

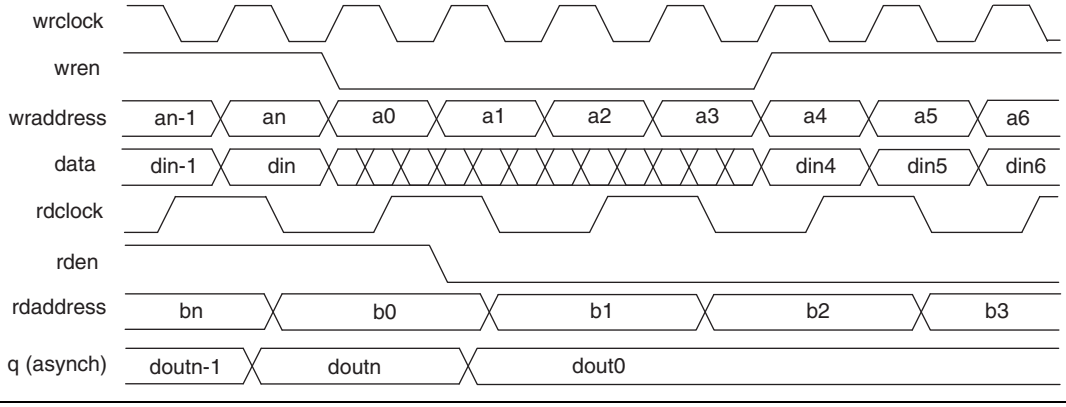
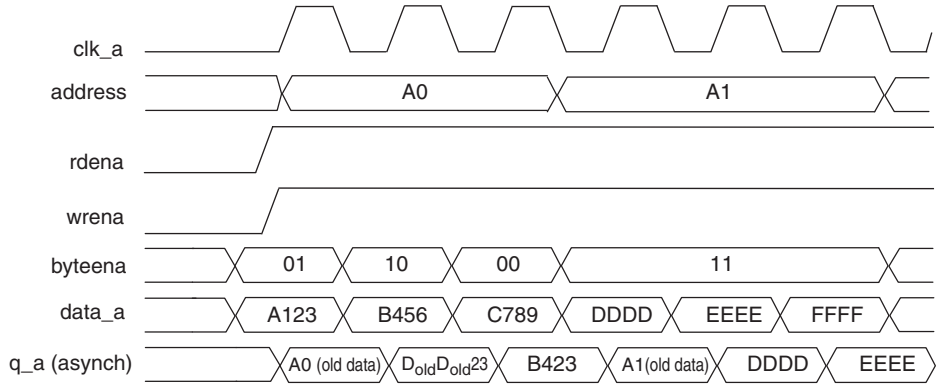


Figure 3–15 shows timing waveforms for read and write operations in mixed-port mode with unregistered outputs.

Figure 3–15. Mixed-Port Read-During-Write Timing Waveforms

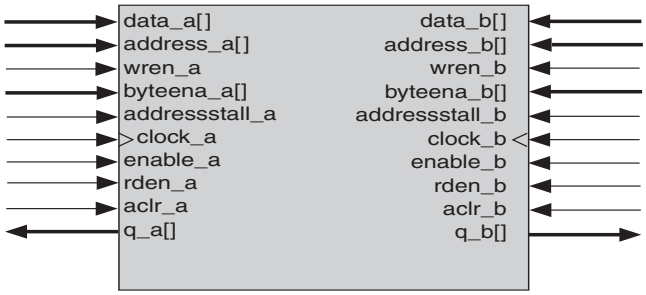


True Dual-Port Mode

Arria II M9K and M144K blocks support true dual-port mode. Sometimes called bidirectional 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. True dual-port memory supports input and output clock mode in addition to the independent clock mode.

Figure 3–16 shows the true dual-port RAM configuration.

Figure 3–16. Arria II True Dual-Port Memory



The widest bit configuration of the M9K and M144K blocks in true dual-port mode are:

- M9K: 512 × 16-bit (or 512 × 18-bit with parity)
- M144K: 4K × 32-bit (or 4K × 36-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 3–7 lists the possible M9K block mixed-port width configurations in true dual-port mode.

Table 3–7. M9K Block Mixed-Width Configuration (True-Dual Port Mode)

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	—	—	—	—	—	✓	✓

Table 3–8 lists the possible M144K block mixed-port width configurations in true dual-port mode.

Table 3–8. M144K Block Mixed-Width Configurations (True Dual-Port Mode)

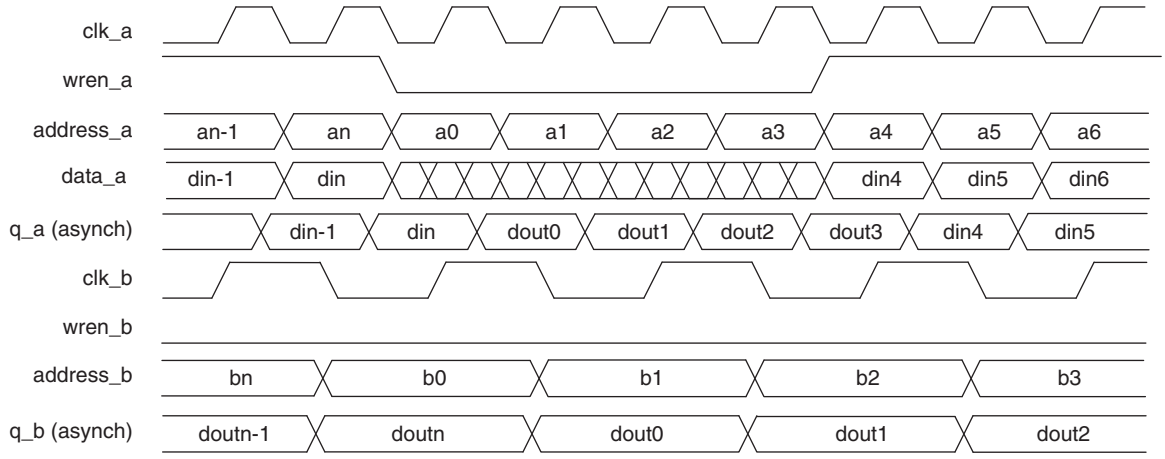
Read Port	Write Port					
	16K × 8	8K × 16	4K × 32	16K × 9	8K × 18	4K × 36
16K × 8	✓	✓	✓	—	—	—
8K × 16	✓	✓	✓	—	—	—
4K × 32	✓	✓	✓	—	—	—
16K × 9	—	—	—	✓	✓	✓
8K × 18	—	—	—	✓	✓	✓
4K × 36	—	—	—	✓	✓	✓

In true dual-port mode, M9K and M144K blocks support separate write-enable and read-enable signals. You can save power by keeping the read-enable signal low (inactive) when not reading. 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. Conflict resolution circuitry is not built into the Arria II memory blocks. You must handle address conflicts external to the RAM block.

Figure 3–17 shows true dual-port timing waveforms for the write operation at port A and the read operation at port B with the read-during-write behavior set to **new data**. Registering the RAM outputs delay the q outputs by one clock cycle.

Figure 3–17. True Dual-Port Timing Waveform



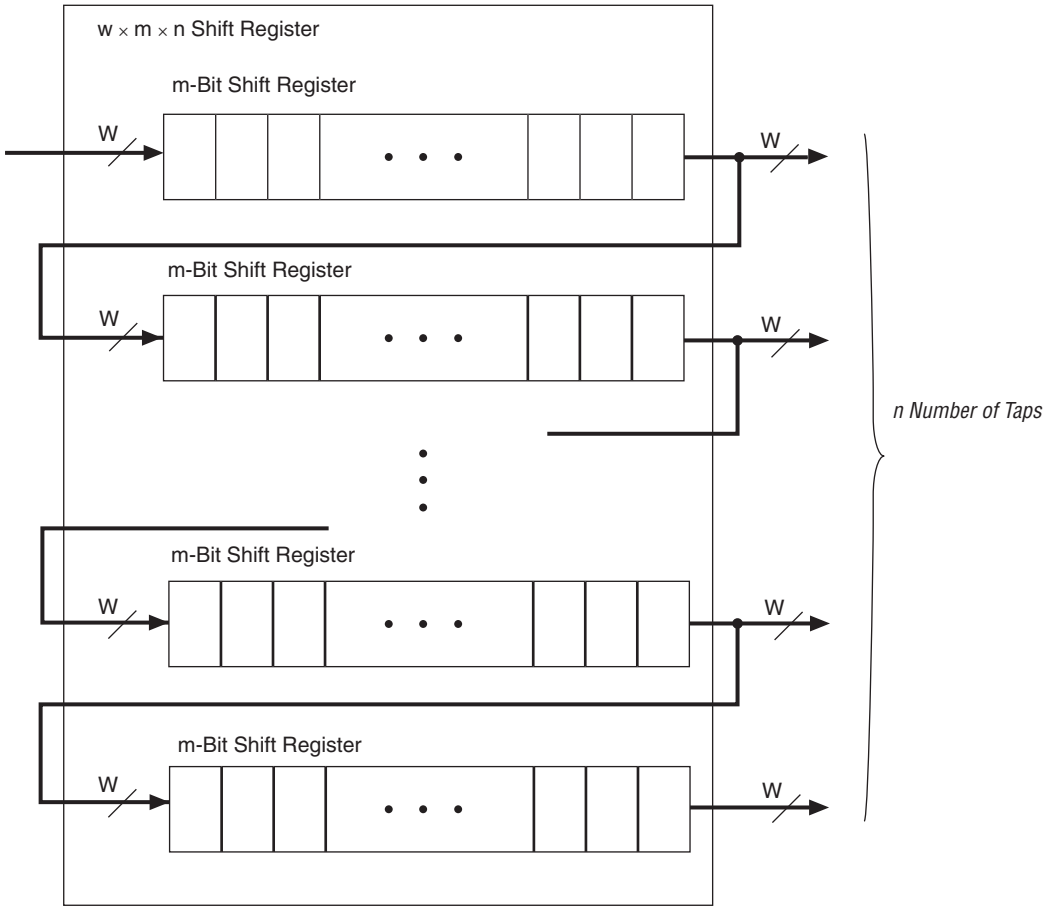
Shift-Register Mode

All Arria II 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 3–18 shows the memory block in shift-register mode.

Figure 3–18. Shift-Register Memory Configuration



ROM Mode

All Arria II memory blocks support ROM mode. A **.mif** initializes the ROM contents of these blocks. The address lines of the ROM are registered on M9K and M144K blocks; however, they 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 *SCFIFO and DCFIFO Megafunctions User Guide*.

MLABs do not support mixed-width FIFO mode.

Clocking Modes

Arria II memory blocks support the following clocking modes:

- "Independent Clock Mode" on page 3–19
- "Input and Output Clock Mode" on page 3–19
- "Read and Write Clock Mode" on page 3–19
- "Single Clock Mode" on page 3–20



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 3–9 lists the supported clocking mode/memory mode combinations.

Table 3–9. Internal Memory Clock Modes for Arria II Devices

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 memory blocks can implement independent clock mode for true dual-port memories. In this mode, a separate clock is available for each port (clock A and clock B). Clock A controls all registers on the port A side; clock B controls all registers on the port B side. Each port also supports independent clock enables for both port A and port B registers, respectively. Asynchronous clears are supported only for output latches and output registers on both ports.

Input and Output Clock Mode

Arria II 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 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.

When using read and write clock mode, the output read data is unknown if you perform a simultaneous read and write to the same address location. If you require the output data to be a known value, use either single clock mode or input and output clock mode, and choose the appropriate read-during-write behavior in the MegaWizard Plug-In Manager.

Single Clock Mode

Arria II 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.

Selecting Memory Block

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 with the Quartus II software. Emulation results in minimal additional logic resources 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 Devices* chapter.

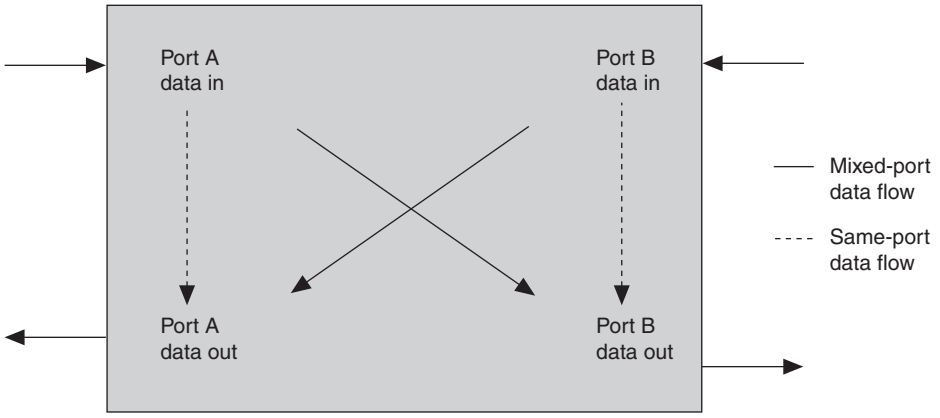
Conflict Resolution

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 Behavior

You can customize the read-during-write behavior of the Arria II memory blocks to suit your design requirements. The two types of read-during-write operations are same port and mixed port. Figure 3–19 shows the difference between the same port and mixed port.

Figure 3–19. 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 3–20 shows sample functional waveforms of same-port read-during-write behavior in don't care mode for MLABs.

Figure 3–20. MLABs Blocks Same Port Read-During Write: Don't Care Mode

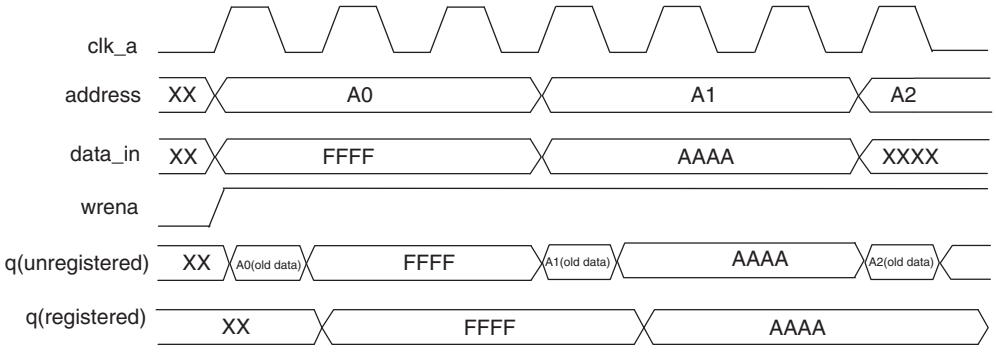


Figure 3–21 shows sample functional waveforms of same-port read-during-write behavior in new data mode.

Figure 3–21. M9K and M144K Blocks Same Port Read-During Write: New Data Mode

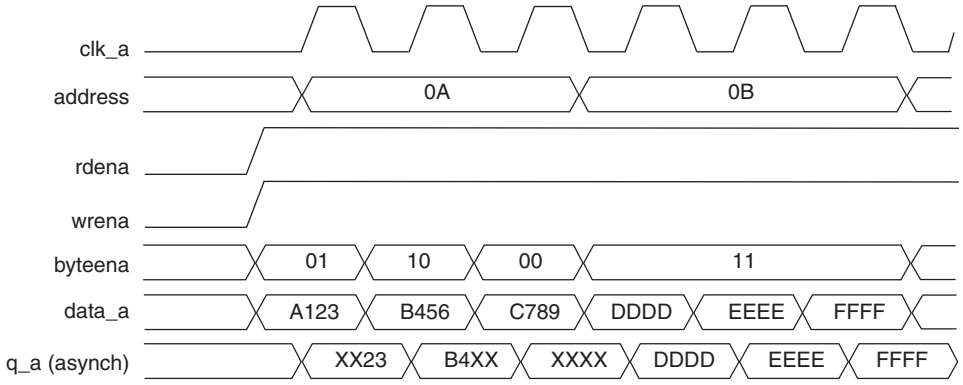
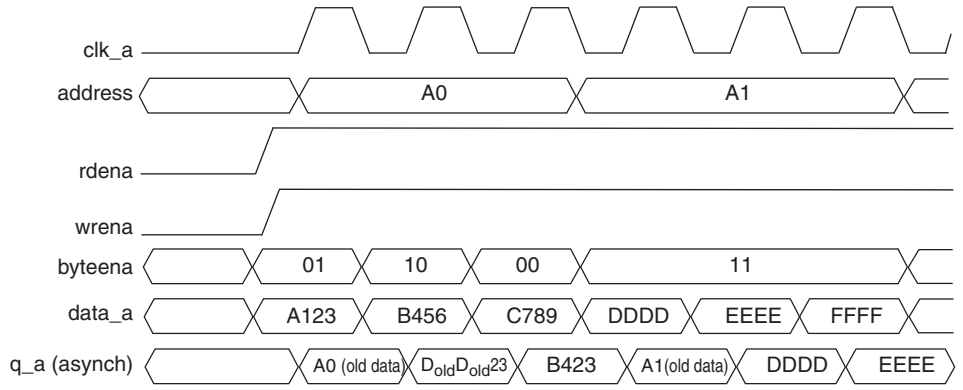


Figure 3–22 shows sample functional waveforms of same-port read-during-write behavior in old data mode.

Figure 3–22. M9K and M144K Blocks Same Port Read-During-Write: Old Data Mode



For MLABs, the output of the MLABs can only be set to **don't care** in same-port read-during-write mode. In this mode, the output of the MLABs is unknown during a write cycle. There is a window near the falling edge of the clock during which the output is unknown. Prior to that window, "old data" is read out; after that window, "new data" is seen at the output.

Mixed-Port Read-During-Write Mode

This mode applies to a RAM in simple or true dual-port mode that has one port reading from and the other port writing to the same address location with the same clock. In this mode, you can choose "old data", "new data" or "don't care" values as the output.

For old data mode, a read-during-write operation to different ports causes the RAM outputs to reflect the "old data" value at that address location.

For new data mode, a read-during-write operation to different ports causes the MLAB registered output to reflect the "new data" value on the next rising edge after the data is written to the MLAB memory.

For 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 *Internal Memory (RAM and ROM) Megafunction User Guide*.

Figure 3–23 shows a sample functional waveform of mixed-port read-during-write behavior for old data mode in MLABs.

Figure 3–23. MLABs Mixed-Port Read-During-Write: Old Data Mode

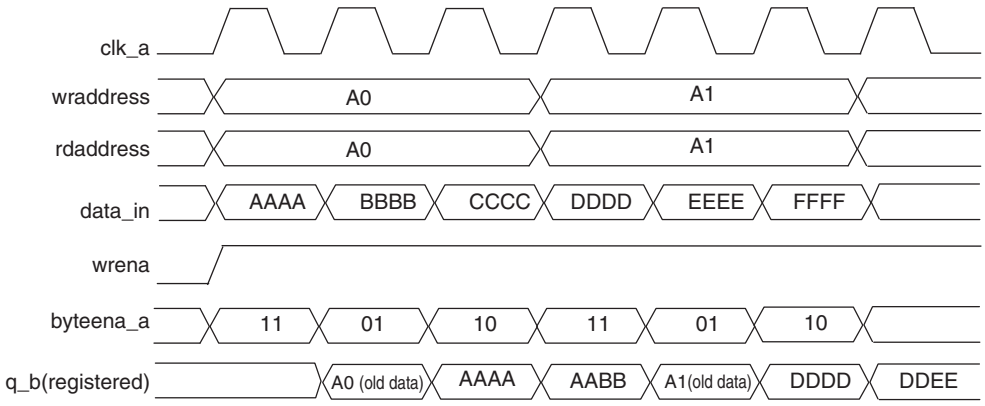


Figure 3–24 shows a sample functional waveform of mixed-port read-during-write behavior for new data mode in MLABs.

Figure 3–24. MLABs Mixed-Port Read-During-Write: New Data Mode

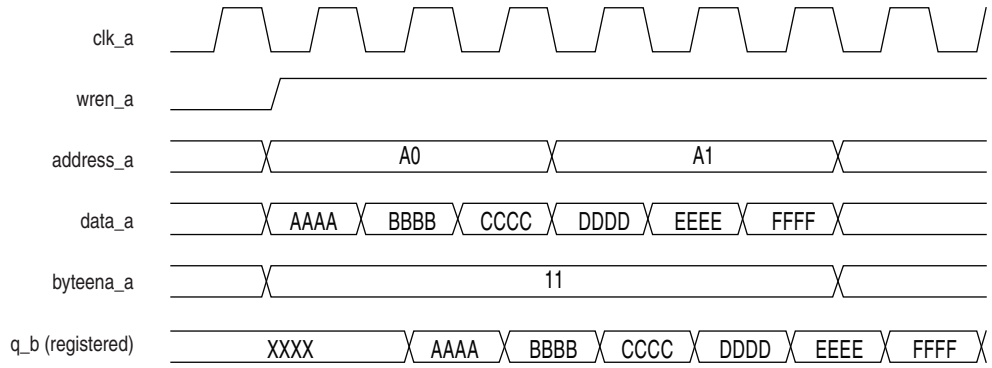


Figure 3–25 shows a sample functional waveform of mixed-port read-during-write behavior for don't care mode in MLABs.

Figure 3–25. MLABs Mixed-Port Read-During-Write: Don't Care Mode

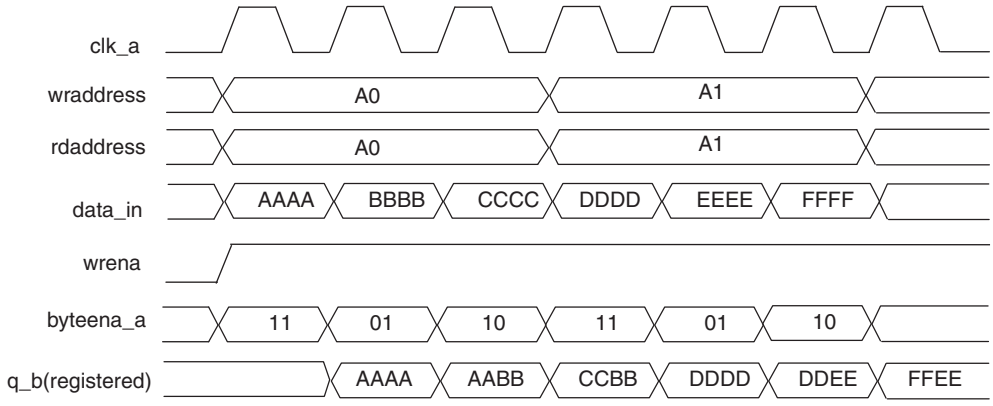


Figure 3–26 shows a sample functional waveform of mixed-port read-during-write behavior in old data mode.

Figure 3–26. M9K and M144K Mixed Port Read During Write: Old Data Mode

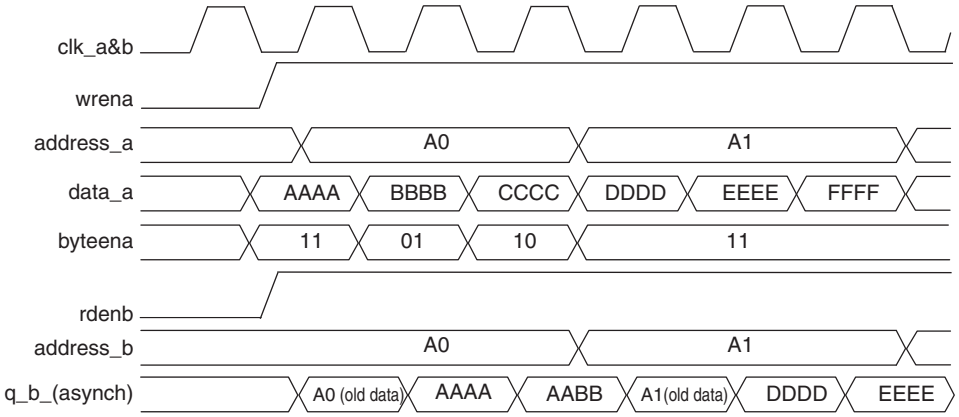
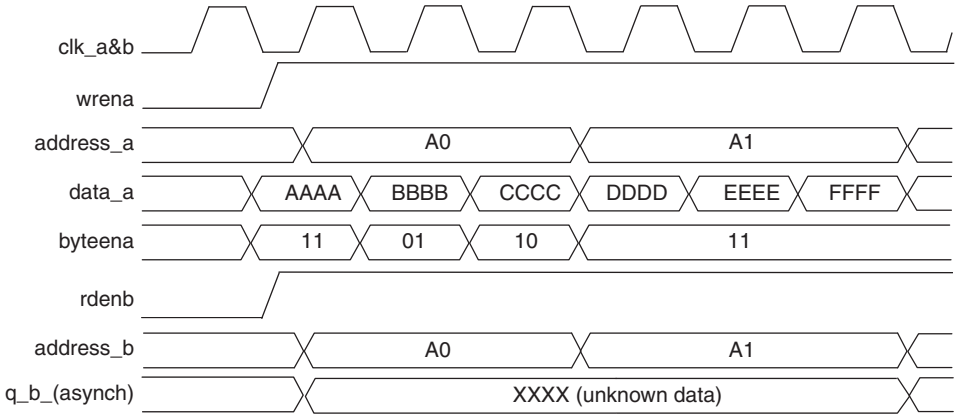


Figure 3–27 shows a sample functional waveform of mixed-port read-during-write behavior for don't care mode in M9K and M144K blocks.

Figure 3–27. M9K and M144K Mixed-Port Read-During-Write: Don't Care 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 and M144K block outputs power up to zero (cleared), regardless of whether the output registers are used or bypassed. MLABs power up to zero if the output registers are used and power up reading the memory contents if the output registers are not used. You must take this into consideration when designing logic that might evaluate the initial power-up values of the MLAB memory block. For Arria II devices, the Quartus II software initializes the RAM cells to zero unless there is a **.mif file** specified.

All memory blocks support initialization using a **.mif**. 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**), it still powers up with its outputs cleared.

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

Power Management

Arria II 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 deasserting the read-enable signal during write operations or any period when no memory operations occur.

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

Document Revision History

Table 3–10 lists the revision history for this chapter.

Table 3–10. Document Revision History

Date	Version	Changes
December 2011	3.2	■ Updated Table 3–1. ■ Updated "Byte Enable Support" and "Mixed-Port Read-During-Write Mode" sections.
June 2011	3.1	■ Updated Table 3–1. ■ Updated the "Mixed-Port Read-During-Write Mode" section. ■ Minor text edits.
December 2010	3.0	■ Updated for the Quartus II software version 10.1 release. ■ Added Arria II GZ devices information. ■ Updated Table 3–1 and Table 3–2. ■ Updated Figure 3–10, Figure 3–12, and Figure 3–16. ■ Added Table 3–6 and Table 3–8. ■ Added Figure 3–10, Figure 3–15, Figure 3–21, Figure 3–23, and Figure 3–24. ■ Added "Error Correction Code Support" section. ■ Minor text edit.
November 2009	2.0	Updated for Arria II GX v9.1 release: ■ Updated Table 3–2 ■ Updated Figure 3–16 ■ Minor text edit
June 2009	1.1	■ Updated Table 3–1 ■ Updated "Byte Enable Support", "Simple Dual-Port Mode", and "Read and Write Clock Mode" sections ■ Updated Figure 3–1, Figure 3–2, Figure 3–5, Figure 3–9, Figure 3–12, Figure 3–18, Figure 3–19, and Figure 3–20 ■ Added Figure 3–2, Figure 3–6, Figure 3–10, and Figure 3–13
February 2009	1.0	Initial release

4. DSP Blocks in Arria II Devices

AIIGX51004-4.0

This chapter describes how the dedicated high-performance digital signal processing (DSP) blocks in Arria® II device are optimized to support DSP applications requiring high data throughput, such as finite impulse response (FIR) filters, infinite impulse response (IIR) filters, fast Fourier transform (FFT) functions, and encoders. You can configure the DSP blocks to implement one of several operational modes to suit your application. The built-in shift register chain, multipliers, and adders/subtractors minimize the amount of external logic to implement these functions, resulting in efficient resource utilization and improved performance and data throughput 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 devices are ideally suited for these systems because the DSP blocks consist of a combination of dedicated elements that perform multiplication, addition, subtraction, accumulation, summation, and dynamic shift operations.

Along with the high-performance Arria II soft logic fabric and memory structures, you can configure DSP blocks to build sophisticated fixed-point and floating-point arithmetic functions. These can be manipulated easily to implement common, larger computationally intensive subsystems such as FIR filters, complex FIR filters, IIR filters, FFT functions, and discrete cosine transform (DCT) functions.

This chapter contains the following sections:

- "DSP Block Overview" on page 4–2
- "Simplified DSP Operation" on page 4–4
- "Operational Modes Overview" on page 4–7
- "DSP Block Resource Descriptions" on page 4–8
- "Arria II Operational Mode Descriptions" on page 4–14
- "Software Support for Arria II Devices" on page 4–31

© 2010 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

DSP Block Overview

Arria II GX devices have two to four columns of DSP blocks, while Arria II GZ devices have two to seven columns of DSP blocks. These DSP blocks implement multiplication, multiply-add, multiply-accumulate (MAC), and dynamic shift functions. Architectural highlights of the Arria II DSP block include:

- High-performance, power-optimized, fully registered, and pipelined multiplication operations
- Natively supported 9-bit, 12-bit, 18-bit, and 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 efficiently combine multiplication results
- 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 lists the number of DSP blocks in Arria II devices.

Table 4–1. Number of DSP Blocks in Arria II Devices ***(Note 1)***

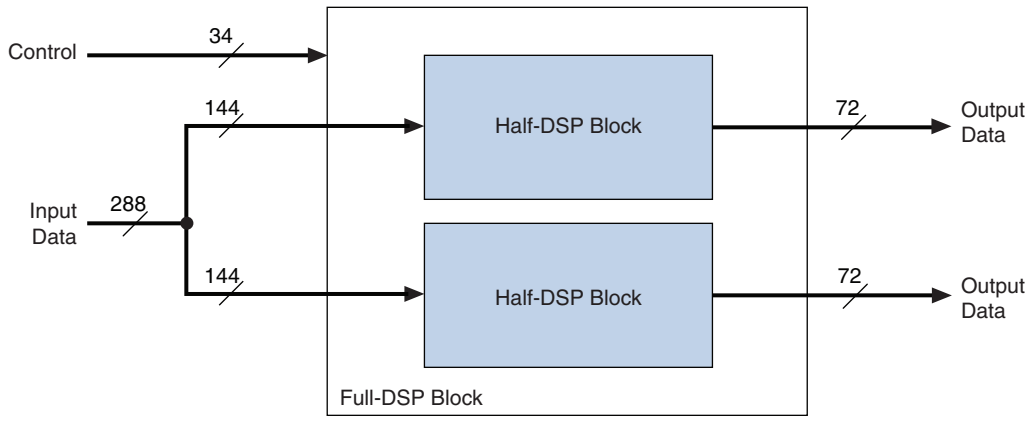
Family	Device	DSP Blocks	Independent Input and Output Multiplication Operators					High Precision Multiplier Adder Mode	Four Multiplier Adder Mode
			9 × 9 Multipliers	12 × 12 Multipliers	18 × 18 Multipliers	18 × 18 Complex	36 × 36 Multipliers	18 × 36 Multipliers	18 × 18 Multipliers
Arria II GX	EP2AGX45	29	232	174	116	58	58	116	232
	EP2AGX65	39	312	234	156	78	78	156	312
	EP2AGX95	56	448	336	224	112	112	224	448
	EP2AGX125	72	576	432	288	144	144	288	576
	EP2AGX190	82	656	492	328	164	164	328	656
	EP2AGX260	92	736	552	368	184	184	368	736
Arria II GZ	EP2AGZ225	100	800	600	400	200	200	400	800
	EP2AGZ300	115	920	690	460	230	230	460	920
	EP2AGZ350	130	1,040	780	520	260	260	520	1,040

Note to Table 4–1:

(1) The numbers in this table represents the numbers of multipliers in their respective mode.

Each DSP block occupies four logic array blocks (LABs) in height and you can divide further into two half blocks that share some common clocks signals, but are for all common purposes identical in functionality. Figure 4–1 shows the layout of each block.

Figure 4–1. Overview of DSP Block Signals



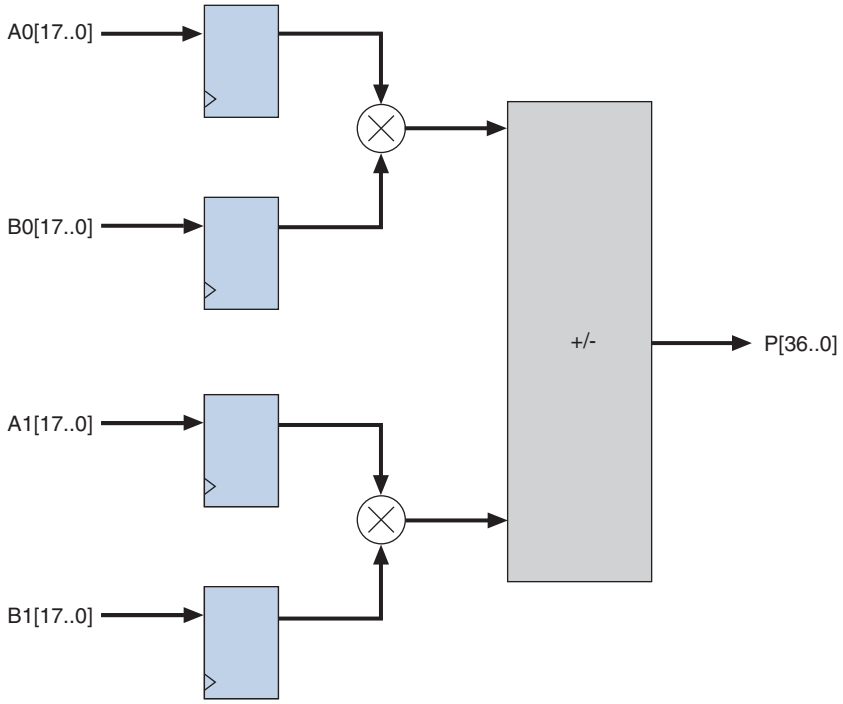
Simplified DSP Operation

In Arria II devices, the fundamental building block is a pair of 18 × 18-bit multipliers followed by a first-stage 37-bit addition and subtraction unit 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 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 4–5 on page 4–8.

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)

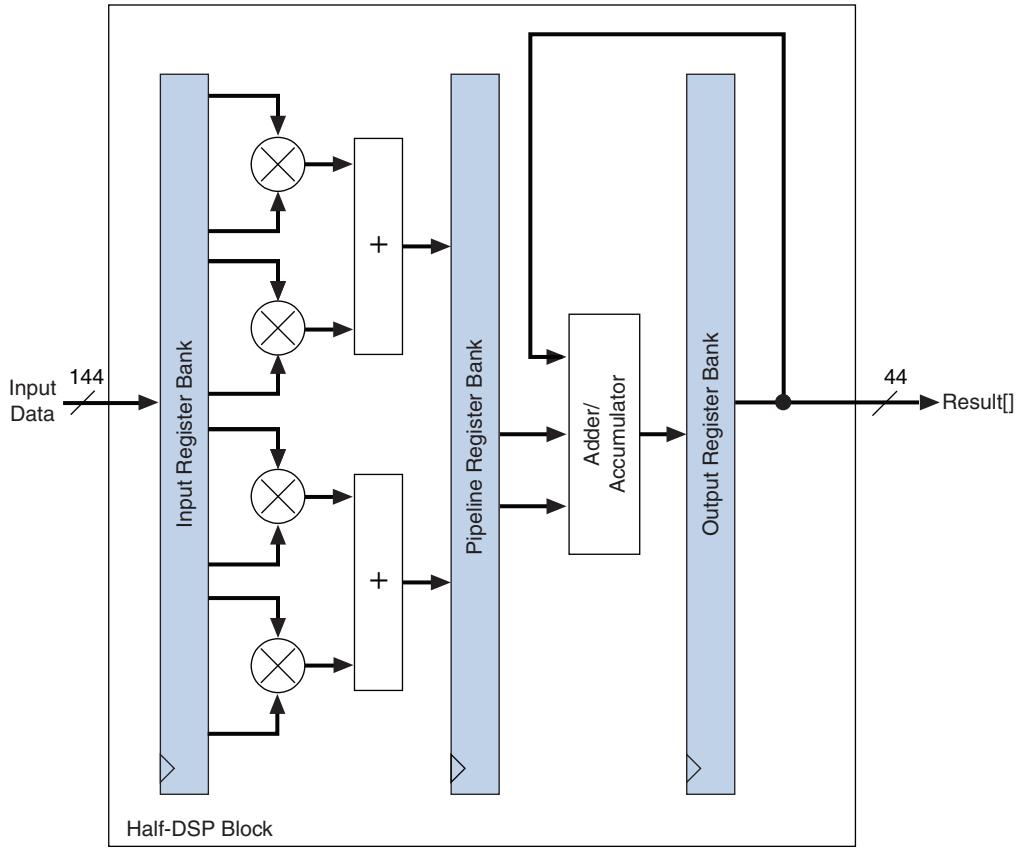
$$W_n[43..0] = W_{n-1}[43..0] \pm Z_n[37..0]$$

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, as shown in Figure 4–3.

You can bypass all register stages depending on which mode you select, except accumulation and loopback mode. In these two modes, you must enable at least one set of the registers. If the register is not enabled, an infinite loop occurs.

Figure 4–3. Four-Multiplier Adder and Accumulation Capability



To support FIR-like structures efficiently, a major addition to the DSP block in Arria II 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 with the output register stage shown in Figure 4–4. Detailed examples are described in later sections.

The combination of a fast, low-latency four-multiplier adder unit and the "chained cascade" capability of the output chaining adder provides the optimal FIR and vector multiplication capability.

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

Figure 4–4. Output Cascading Feature for FIR Structures

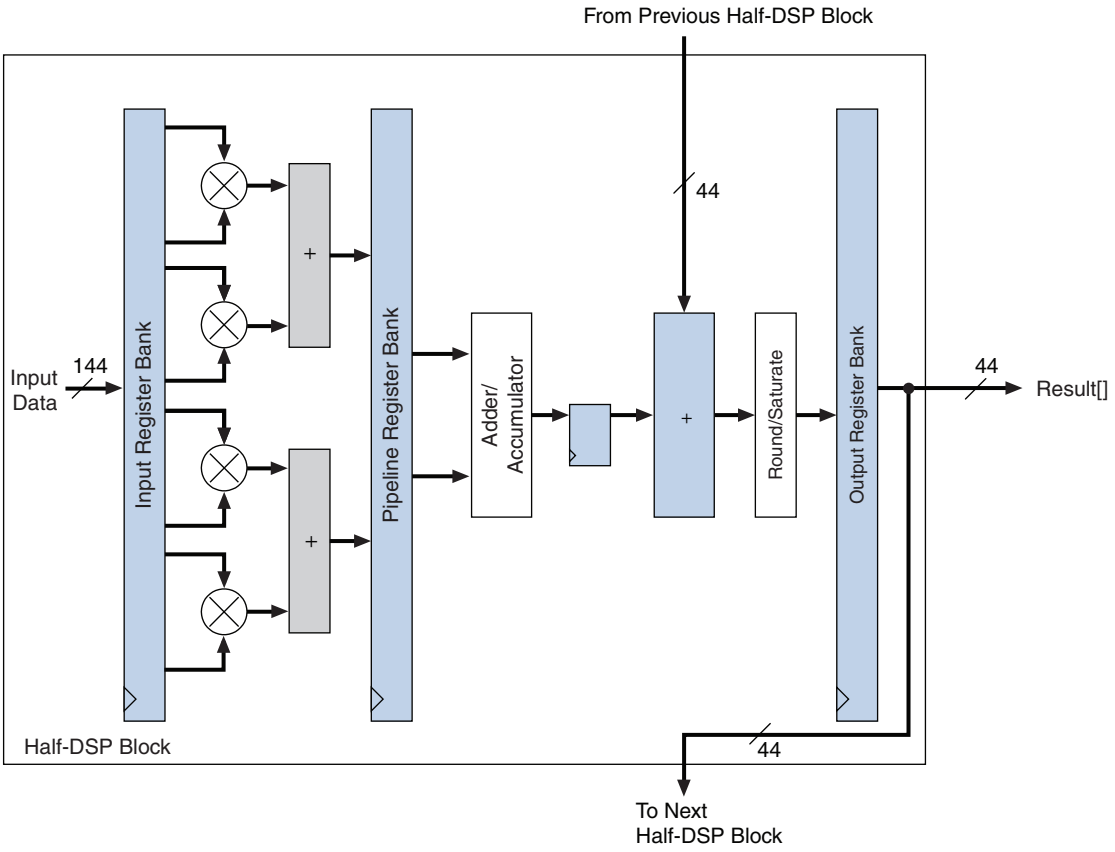


Figure 4–4 shows the optional rounding and saturation unit. This unit provides a set of commonly found arithmetic rounding and saturation functions in signal processing.

In addition to the independent multipliers and sum modes, you can use DSP blocks to perform shift operations. DSP blocks can dynamically switch between logical shift left/right, arithmetic shift left/right, and rotation operation in one clock cycle.

Operational Modes Overview

You can use each Arria II DSP block in one of six basic operational modes. Table 4–2 lists the six basic operational modes and the number of multipliers that you can implement in a single DSP block.

Table 4–2. DSP Block Operational Modes for Arria II Devices

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 *(2)*	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
Shift *(3)*	36 bits *(4)*	1	2	Both	No	No	—	—	—
High Precision Multiplier Adder	18 × 36	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) Unsigned value is also supported, but you must ensure that the result can be contained in 36 bits.

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

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

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

The Quartus® II software includes megafunctions that control the mode of operation of the multipliers. After making the appropriate parameter settings with the megafunction's MegaWizard™ Plug-In Manager, the Quartus II software automatically configures the DSP block.

Arria II DSP blocks can operate in different modes simultaneously. Each half block is fully independent except for the sharing of the `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 device. The Quartus II software automatically places multipliers that can share the same DSP block resources in the same block.

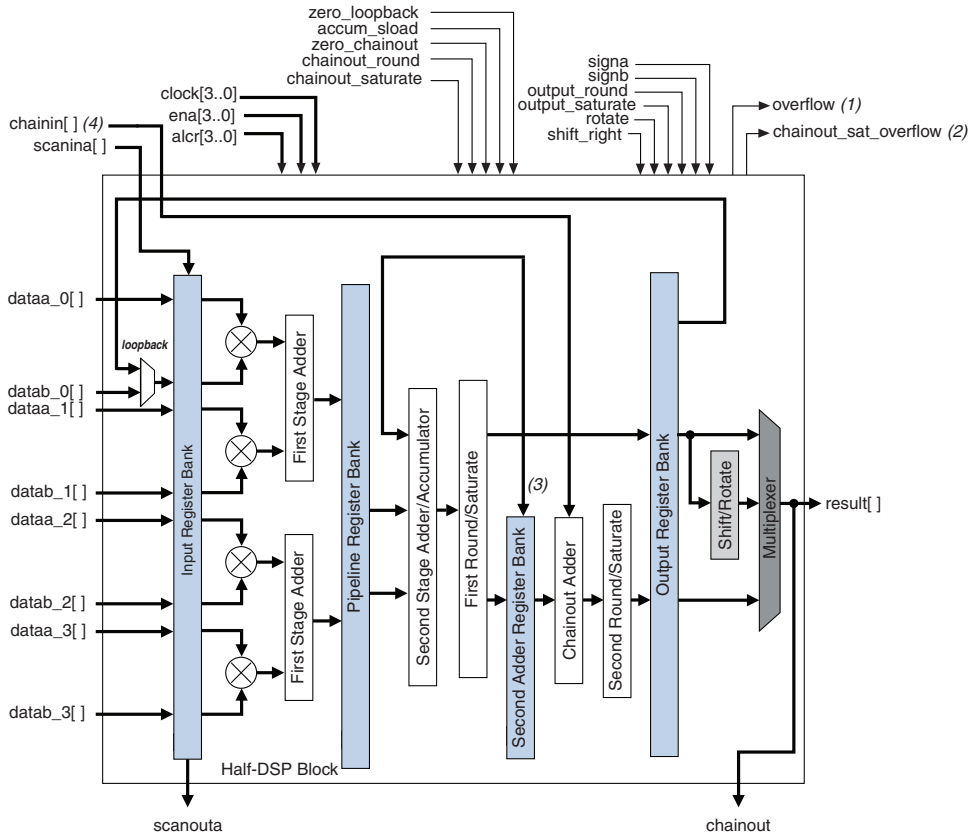
DSP Block Resource Descriptions

The DSP block consists of the following elements:

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

Figure 4–5 shows a detailed illustration of the overall architecture of the top half of the DSP block. Table 4–9 on page 4–30 lists the DSP block dynamic signals.

Figure 4–5. Half-DSP Block Architecture



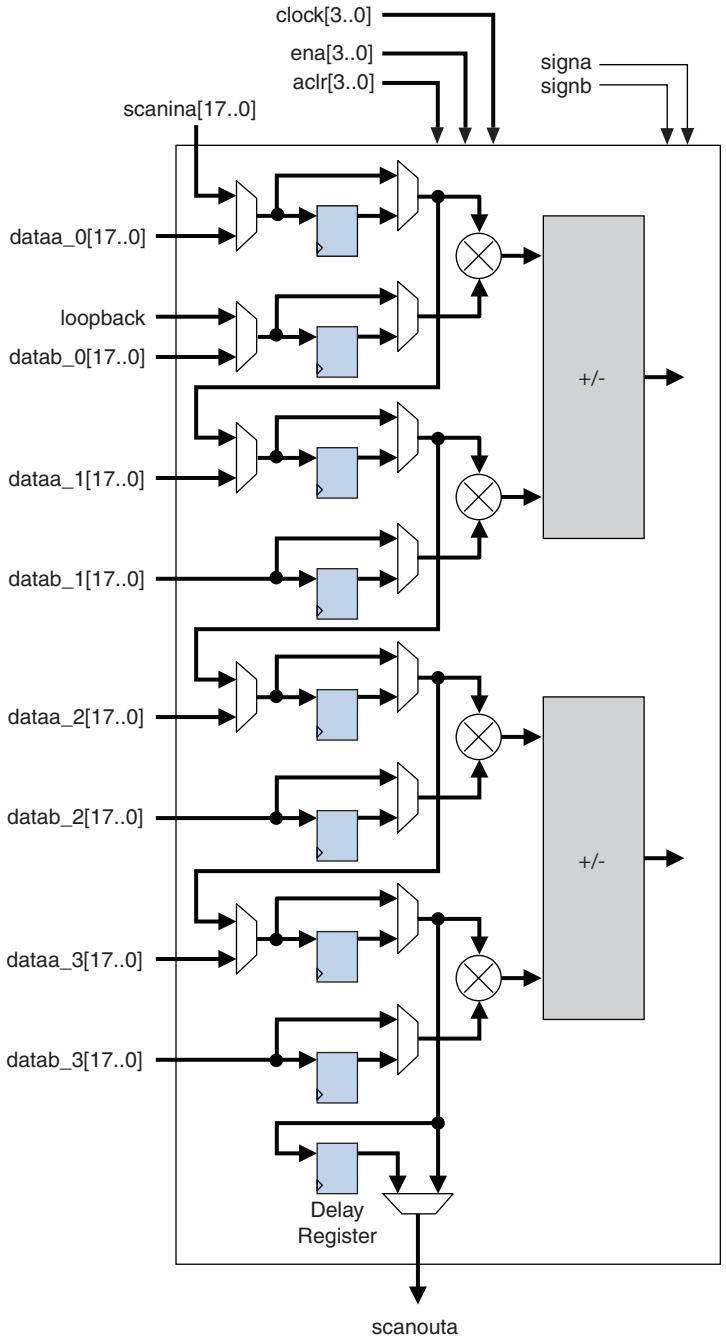
Notes to Figure 4–5:

(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.
(4) You must connect the `chainin` port to the `chainout` port of the previous DSP blocks; it must not be connected to general routings.

Input Registers

Figure 4–6 shows the input register of a half-DSP block.

Figure 4–6. Input Register of Half-DSP Block *(Note 1)*



Note to Figure 4–6:

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

All DSP block registers are triggered by the positive edge of the clock signal and are cleared after power up. Each multiplier operand can feed an input register or feed directly to the multiplier, bypassing the input registers. The `clock[3..0]`, `ena[3..0]`, and `aclr[3..0]` DSP block signals control the input registers in the DSP block.

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 to balance the latency requirements when using the chained cascade feature. A feature of the input register bank is to support a tap delay line. Therefore, you can drive the top leg of the multiplier input (A) from general routing or from the cascade chain, as shown in Figure 4–6.

At compile time, you must select the incoming data for multiplier input (A) from either general routing or from the cascade chain. 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 with the regular FPGA routing resources.

Shift registers are useful in DSP functions such as FIR filters. When implementing an 18 × 18 or smaller width multiplier, 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) has a multiplexer for the first multiplier B-input (lower-leg input) register to select between general routing and loopback, as shown in Figure 4–5 on page 4–8. 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 lists the summary of input register modes for the DSP block.

Table 4–3. Input Register Modes for Arria II Devices

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 word lengths 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 4–14 on page 4–21.

Multiplier and First-Stage Adder

The multiplier stage supports 9 × 9, 12 × 12, 18 × 18, or 36 × 36 multipliers. Other word lengths are padded up to the nearest appropriate native wordlength; for example, 16 × 16 is padded up to use 18 × 18. For more information, refer to "Independent Multiplier Modes" on page 4–14. 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/data B` is a signed number; a `logic 0` value indicates an unsigned number.

Table 4–4 lists the sign of the multiplication result for the various operand sign representations. If any one of the operands is a signed value, the result of the multiplication is signed.

Table 4–4. Multiplier Sign Representation for Arria II Devices

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 half-DSP block must have the same sign representation. Similarly, all `data B` inputs feeding the same half-DSP 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 × 18 loopback and two-multiplier adder modes. For more information, refer to "Two-Multiplier Adder Sum Mode" on page 4–20.



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

Figure 4–5 on page 4–8 shows that the outputs of the multipliers are the only outputs that can feed into the first-stage adder. 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 you must configure after compilation. 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, rounding and saturation unit, or the output registers.

Pipeline Register Stage

Figure 4–5 on page 4–8 shows that the output from the first-stage adder can either feed or bypass the pipeline registers. Pipeline registers increase the maximum performance (at the expense of extra cycles of latency) of the DSP block, 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 either:

- 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 rounding and saturation logic unit or the output register.



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

Rounding and Saturation Stage

Rounding and saturation logic units are located at the output of the 44-bit second-stage adder (the rounding logic unit followed by the saturation logic unit). There are two rounding and saturation logic units per half-DSP block. The input to the rounding 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 "Arria II Operational Mode Descriptions" on page 4–14.

The dynamic rounding and saturation signals control the rounding and saturation logic unit, respectively. A `logic 1` value on the round signal, saturate signal, or both enables the round logic unit, saturate logic unit, or both.



You can use the rounding 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 you can combine to form larger 72-bit banks to support 36 × 36 output results.

The outputs of the different stages in the Arria II 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 come from the outputs of the multiplier block, first-stage adder, pipeline registers, second-stage adder, or the rounding and saturation logic unit. Based on the DSP block operational mode you specify, the output selection unit is automatically set by the software, and has the option to either drive or bypass the output registers. The exception is when the block is used in shift mode, where you dynamically control 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 for performing a four-multiplier adder and the second is for the chainout adder. The outputs of the four-multiplier adder are routed to the second-stage adder registers before enters the chainout adder. The output of the chainout adder goes to the regular output register bank. Depending on the configuration, you can route the chainout results to the input of the next half block's chainout adder input or to the general fabric (functioning as regular output registers).

You can only connect the `chainin` port to the `chainout` port of the previous DSP block and must not be connected to general routings.

The second-stage and output registers are triggered by the positive edge of the clock signal and are cleared on power up. The `clock[3..0]`, `ena[3..0]`, and `aclr[3..0]` DSP block signals control the output registers in the DSP block.

Arria II Operational Mode Descriptions

This section describes the operation modes of Arria II devices.

Independent Multiplier Modes

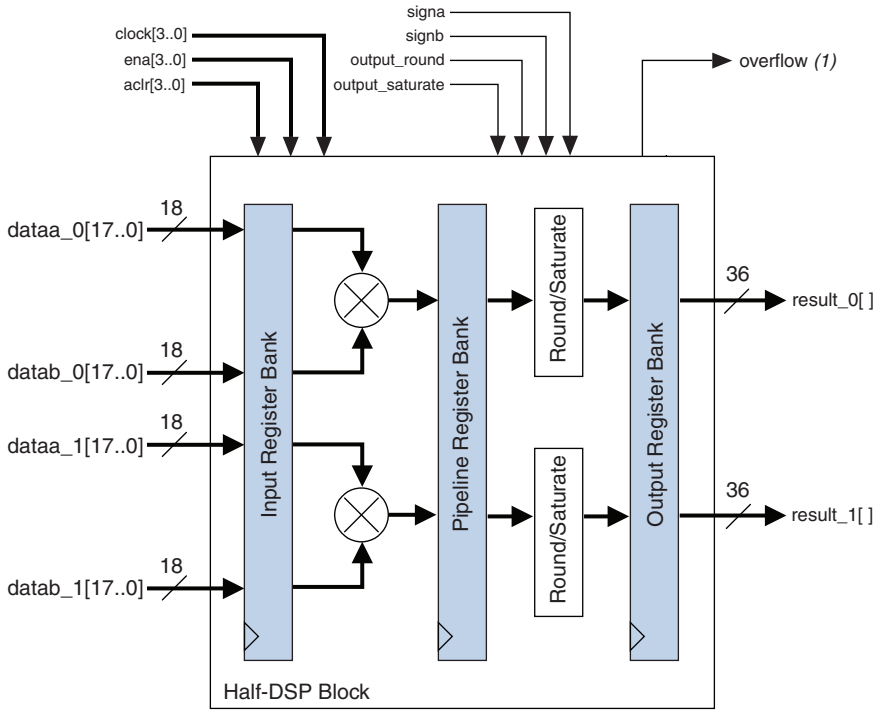
In the independent input and output multiplier mode, the DSP block performs individual multiplication operations for general-purpose multipliers.

9-Bit, 12-Bit, and 18-Bit Multiplier

You can configure each DSP block multiplier for 9-bit, 12-bit, 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 4–7, Figure 4–8, and Figure 4–9 show the DSP block in the independent multiplier operation mode. Table 4–9 on page 4–30 lists the DSP block dynamic signals.

Figure 4–7. 18-Bit Independent Multiplier Mode Shown for Half-DSP Block



Note to Figure 4–7:

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

Figure 4–8. 12-Bit Independent Multiplier Mode Shown for Half-DSP Block

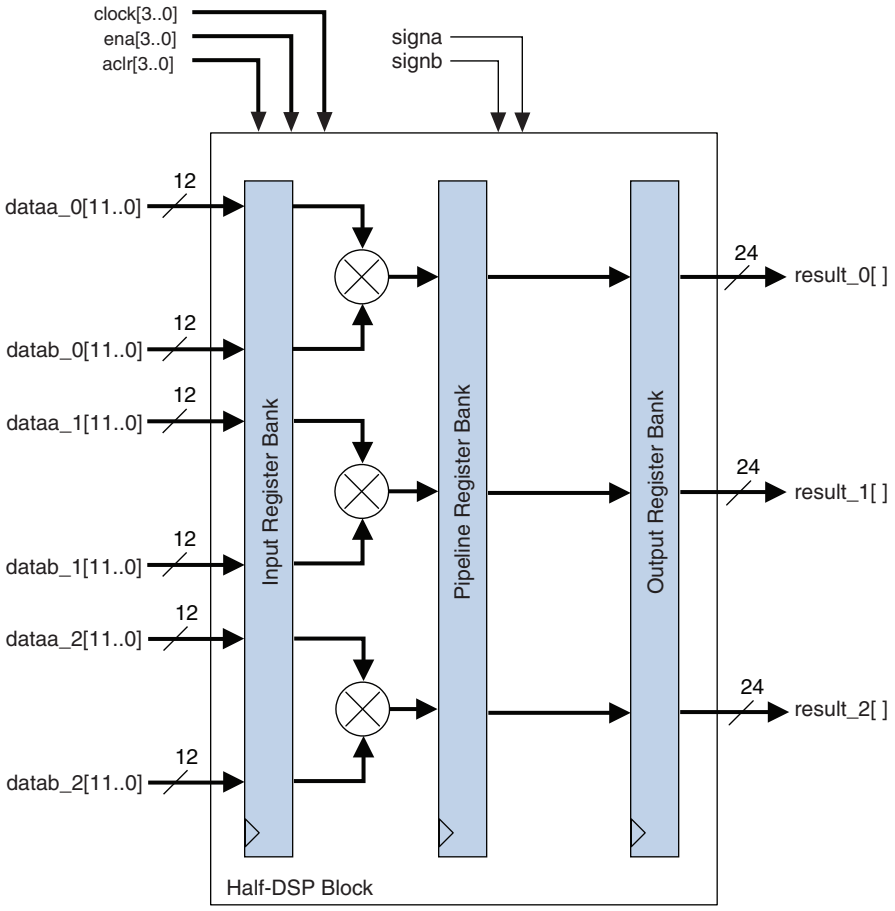
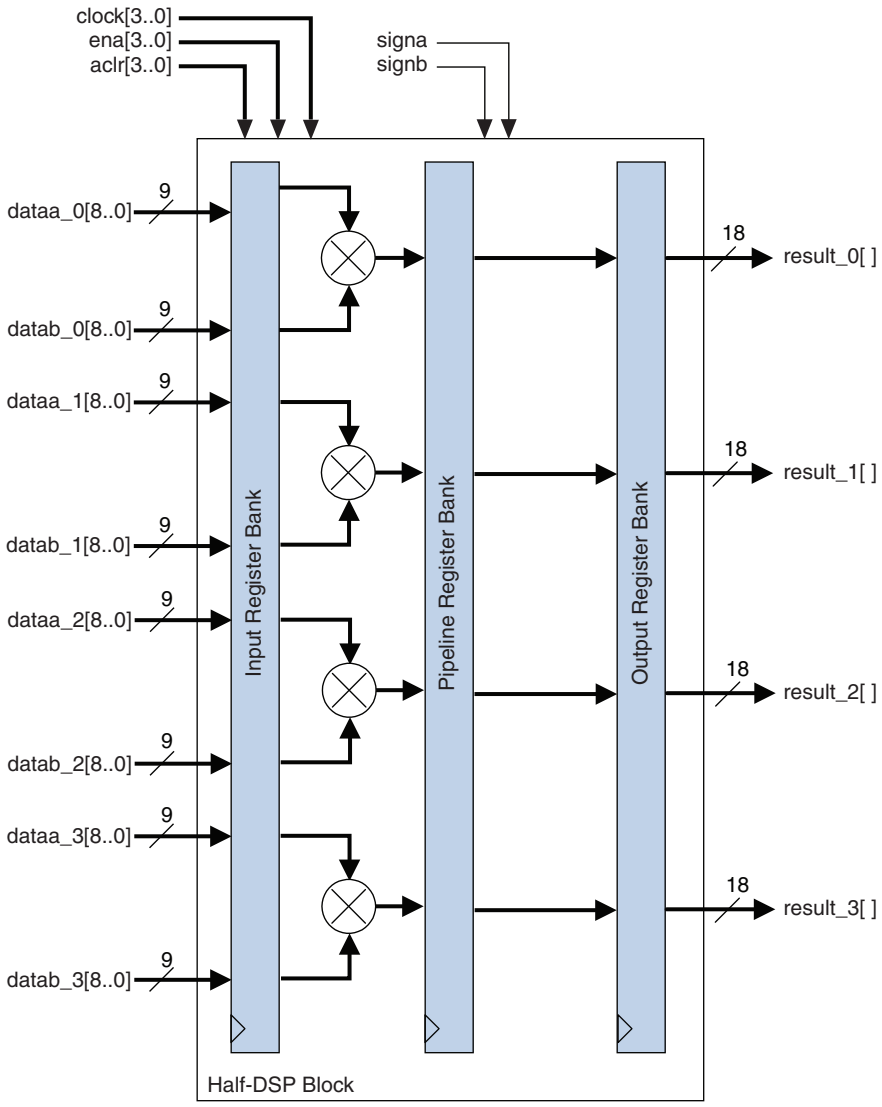


Figure 4–9. 9-Bit Independent Multiplier Mode Shown for Half-DSP Block



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



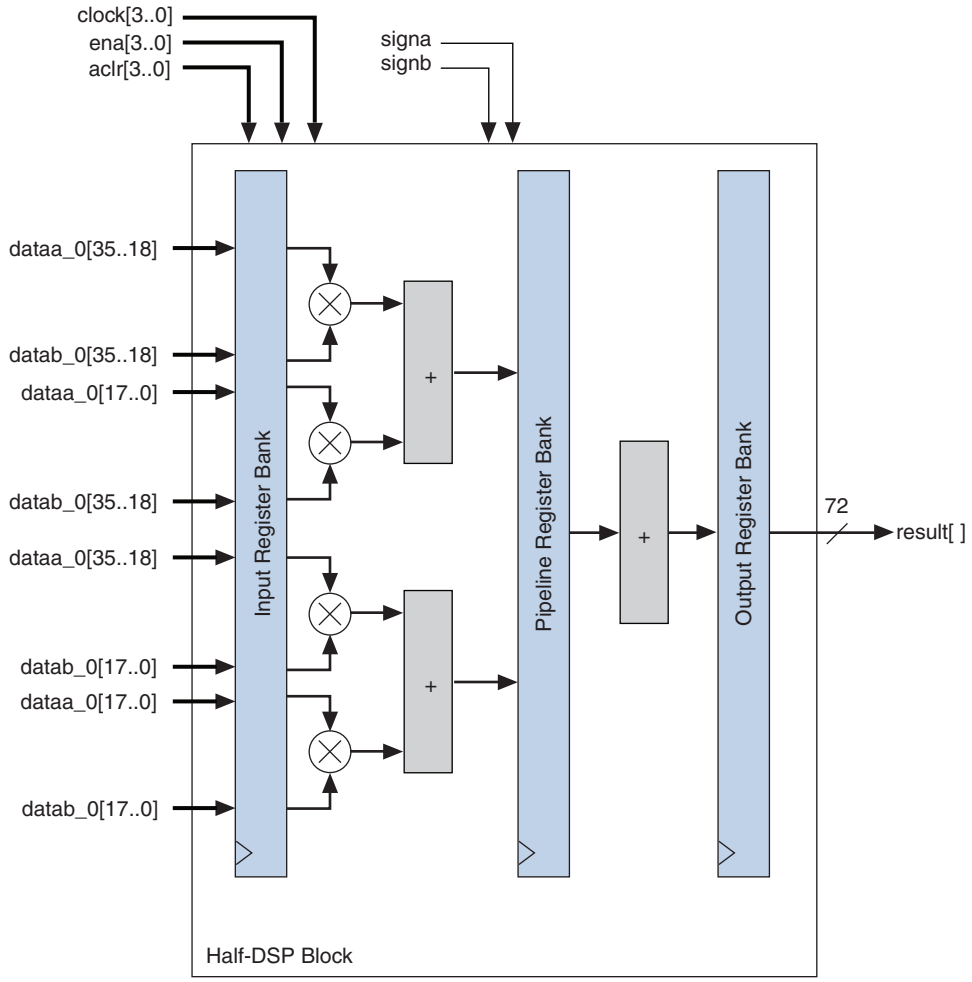
The rounding and saturation logic unit is supported for 18-bit independent multiplier mode only.

36-Bit Multiplier

You can construct a 36 × 36 multiplier with 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. Arria II devices can have up to two 36-bit multipliers per DSP block (one 36-bit multiplier per half DSP block). 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 4–10.

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 4–10. 36-Bit Independent Multiplier Mode Shown for Half-DSP Block



Double Multiplier

You can configure the Arria II 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 with basic 18 × 18 multipliers, shifters, and adders. To efficiently use built-in shifters and adders in the Arria II DSP block, a special double mode (partial 54 × 54 multiplier) is available that

is a slight modification to the basic 36 × 36 multiplier mode, as shown in Figure 4–11 and Figure 4–12.

Figure 4–11. Double Mode Shown for a Half DSP Block

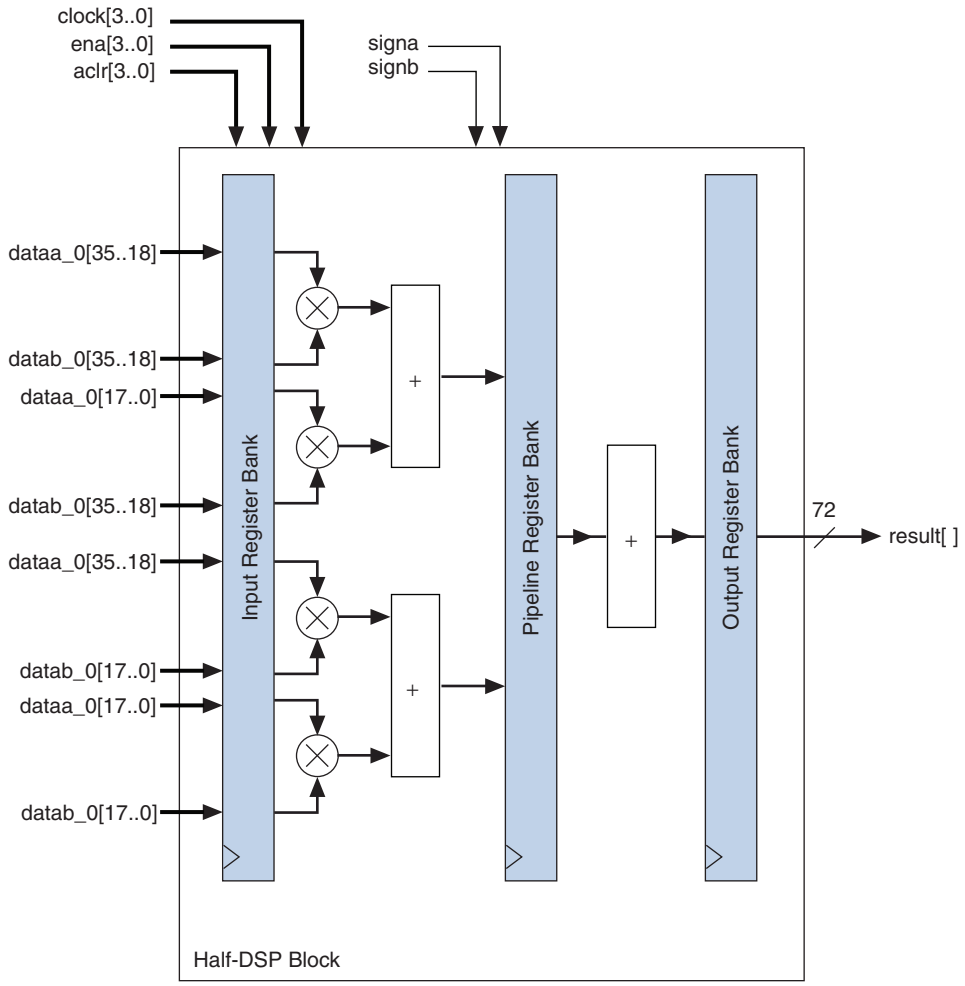
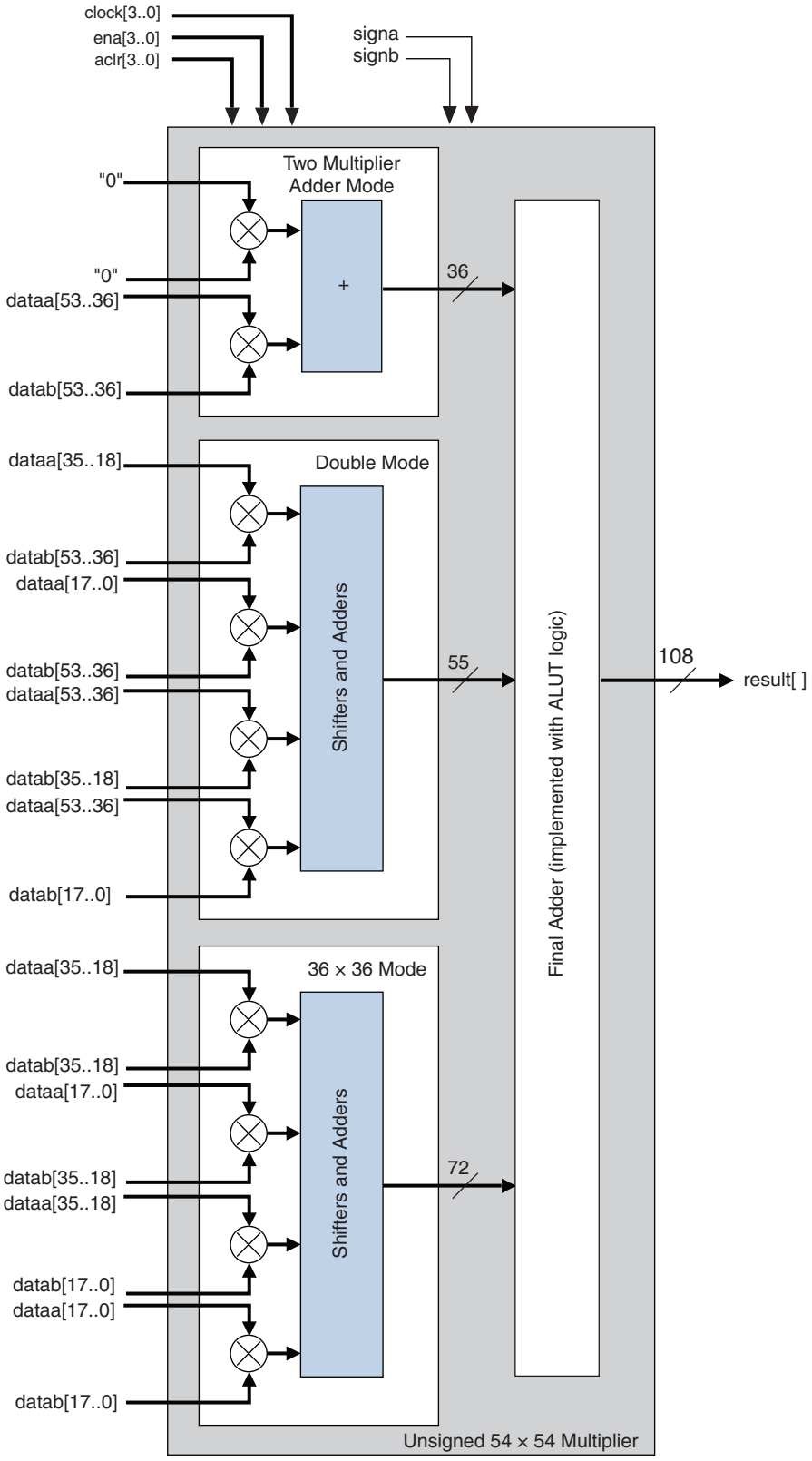


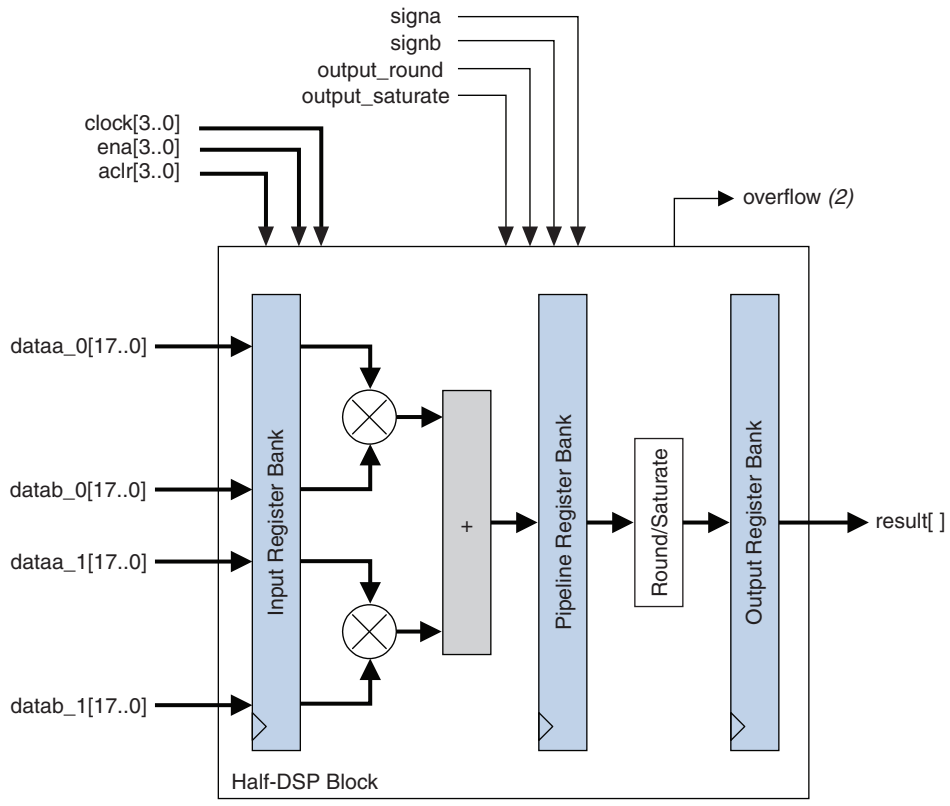
Figure 4–12. Unsigned 54 × 54-Bit Multiplier



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 must be selected at compile time. The two-multiplier adder function is useful for applications such as FFTs, complex FIR, and IIR filters. Figure 4–13 shows the DSP block configured in the two-multiplier adder mode.

Figure 4–13. Two-Multiplier Adder Mode Shown for Half-DSP Block *(Note 1)*



Notes to Figure 4–13:

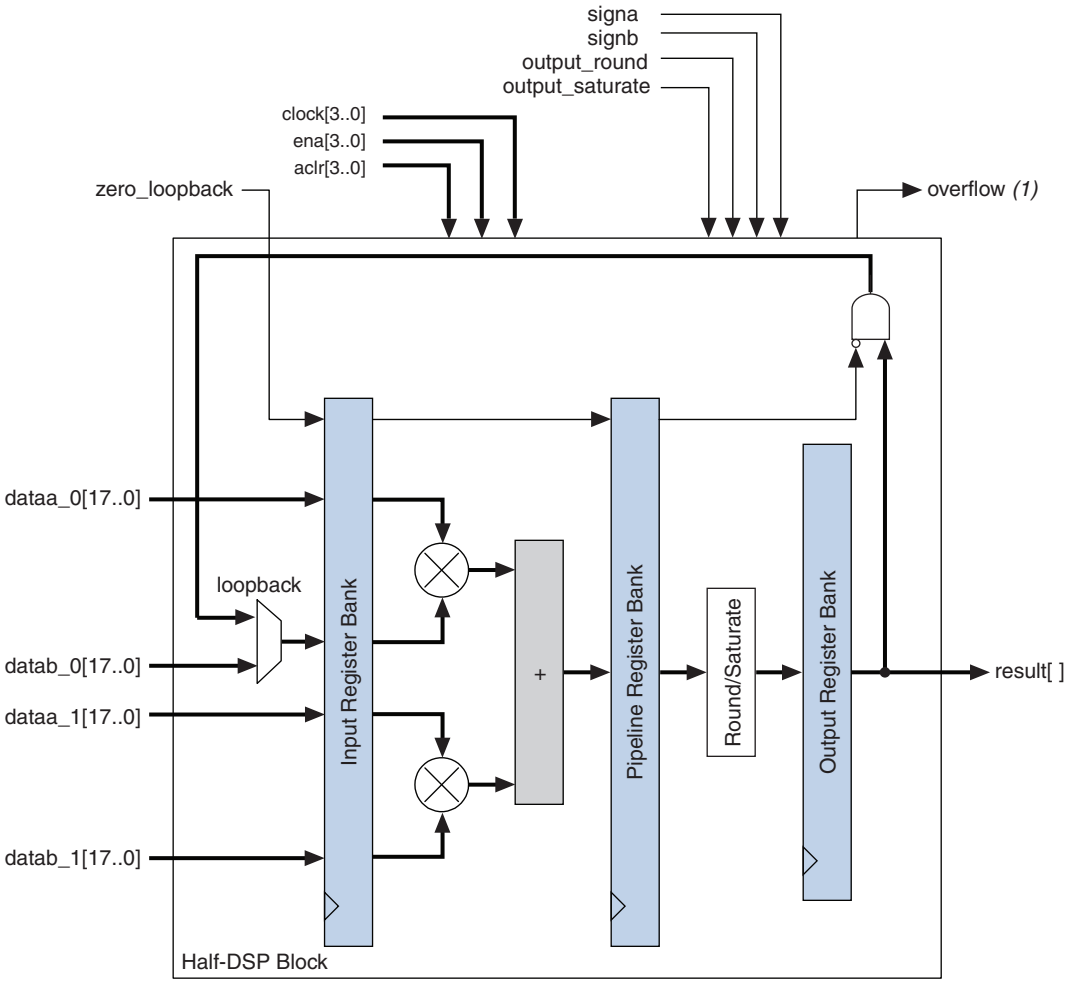
(1) In a half-DSP block, you can implement 2 two-multiplier adders.

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

The loopback mode is a sub-feature of the two-multiplier adder mode. Figure 4–14 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, and a `logic 0` selects the looped back data.

At compile time, you must select the option to use the loopback mode or the general two-multiplier adder mode.

Figure 4–14. Loopback Mode for Half-DSP Block



Note to Figure 4–14:

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

If all the inputs are full 18 bits and unsigned, the result requires 37 bits for two-muliplier adder mode. Because 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 rounding 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 Multiplier

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

Equation 4–4 shows how you can write a complex multiplication.

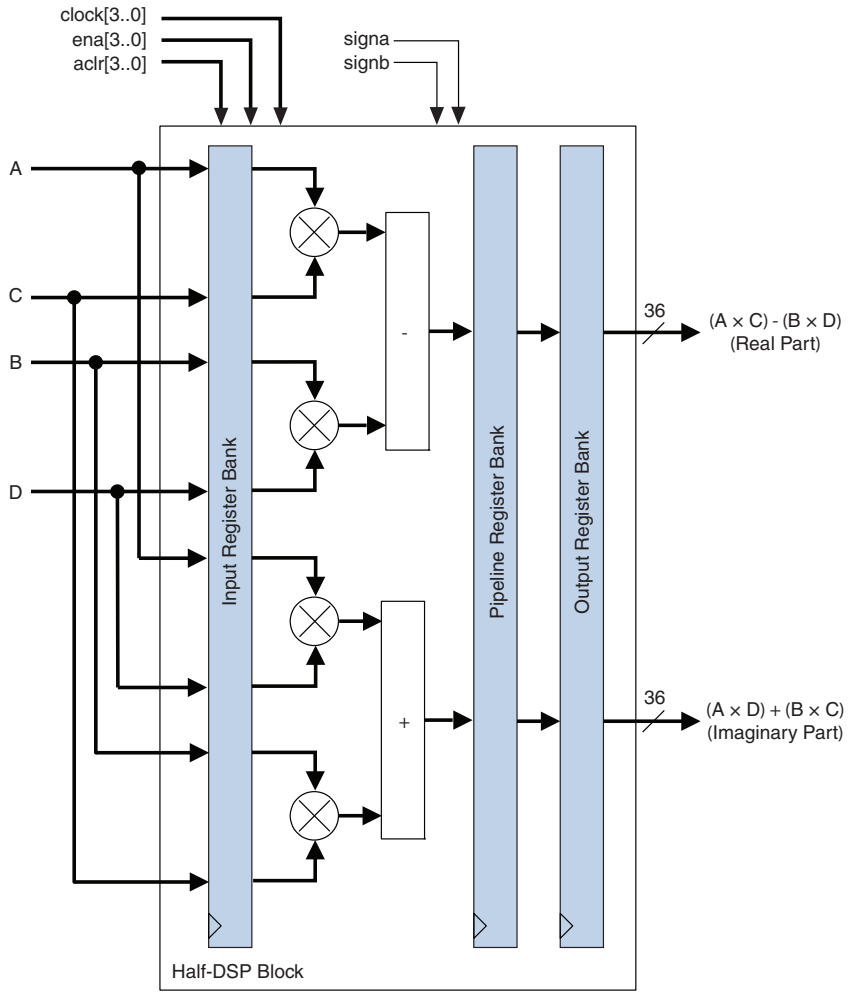
Equation 4–4. Complex Multiplication Equation

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

To implement this complex multiplication in the DSP block, the real part [(a × c) – (b × d)] is implemented with two multipliers feeding one subtractor block, and the imaginary part [(a × d) + (b × c)] is implemented with another two multipliers feeding an adder block. This mode automatically assumes all inputs are using signed numbers.

Figure 4–15 shows an 18-bit complex multiplication. This mode automatically assumes all inputs are using signed numbers.

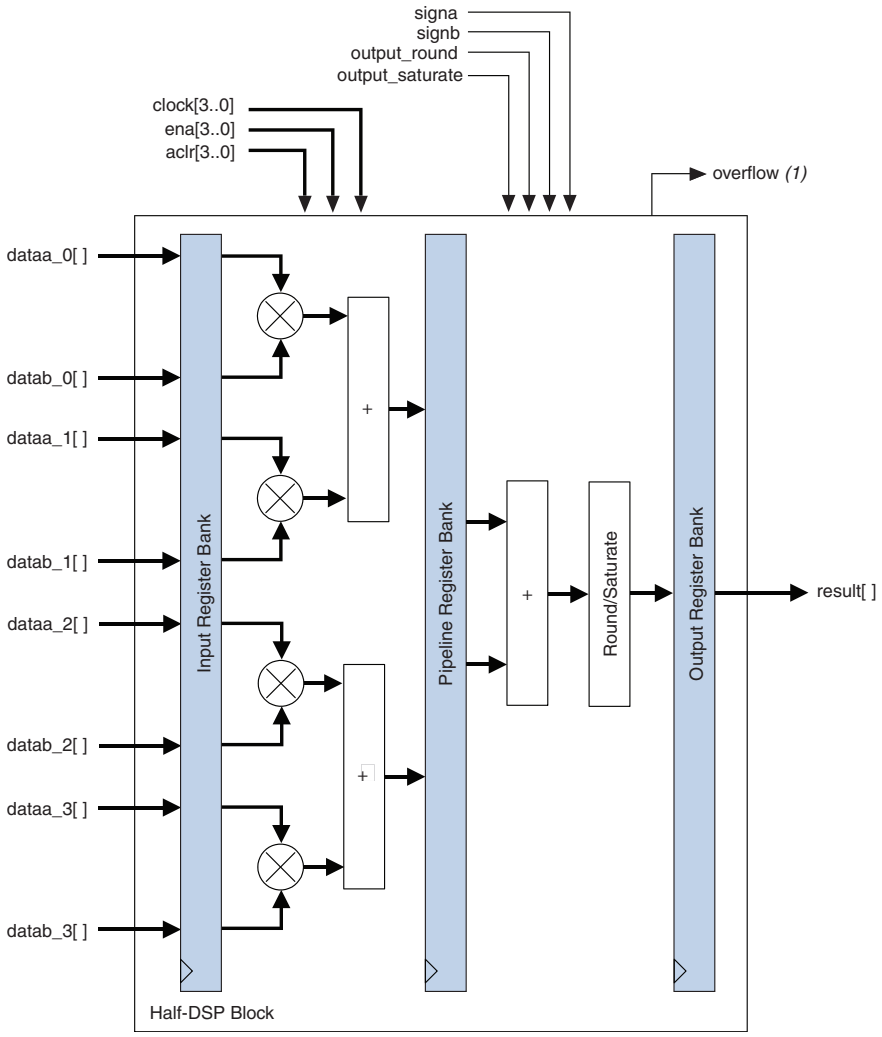
Figure 4–15. Complex Multiplier Using Two-Multiplier Adder Mode



Four-Multiplier Adder

In the four-multiplier adder configuration shown in Figure 4–16, 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 4–4 and Equation 4–3 on page 4–5.

Figure 4–16. Four-Multiplier Adder Mode Shown for Half-DSP Block



Note to Figure 4–16:

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

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

High-Precision Multiplier Adder Mode

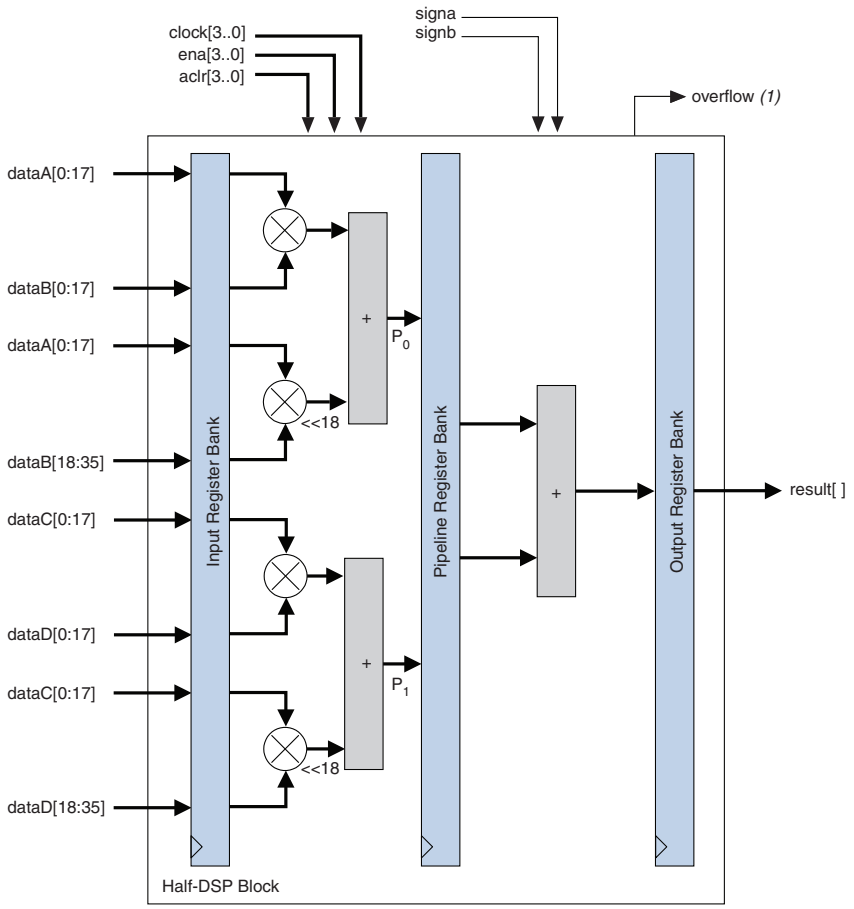
In the high-precision multiplier adder, the DSP block can implement 2 two-multiplier adders, with a multiplier precision of 18 × 36 (one two-multiplier adder per half-DSP block). This mode is useful in filtering or FFT applications where a datapath greater than 18 bits is required, yet 18 bits is sufficient for coefficient precision. This can occur if 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 datapath 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. Figure 4–17 shows 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 4–17. High-Precision Multiplier Adder Configuration for Half-DSP Block



Note to Figure 4–17:

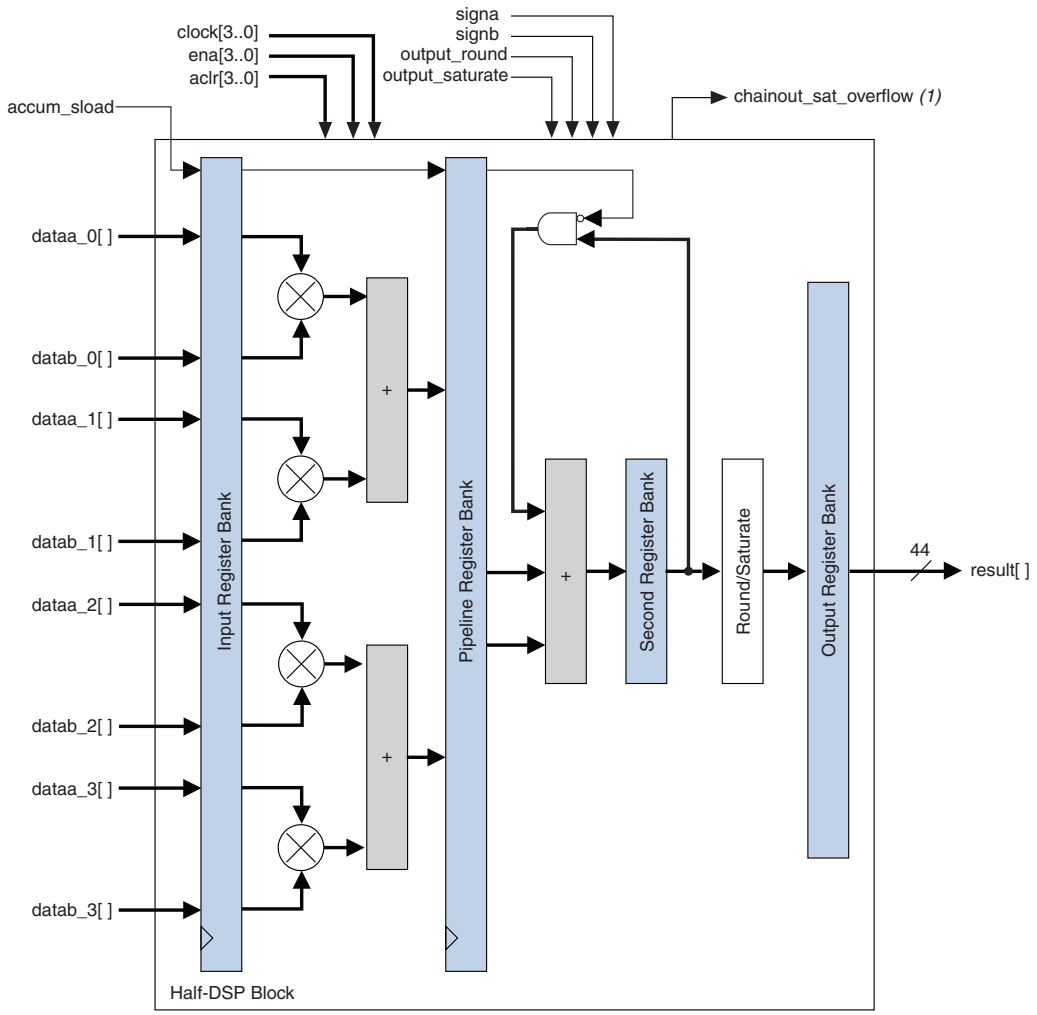
(1) Block output for accumulator overflow and saturate overflow.

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 4–5.

Figure 4–18 shows the DSP block configured to operate in multiply accumulate mode.

Figure 4–18. Multiply Accumulate Mode Shown for Half-DSP Block



Note to Figure 4–18:

(1) Block output for saturation overflow of chainout.

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

Use the dynamic `accum_sload` control signal to clear the accumulation. A `logic 1` value on the `accum_sload` signal synchronously loads the accumulator with the multiplier result only, and 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 you can configure it at compilation.

The multiply accumulate mode supports the rounding and saturation logic unit because it is configured as an 18-bit multiplier accumulator. You can use the pipeline registers and output registers within the DSP block to increase the performance of the DSP block.

Shift Modes

Arria II 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 with the dynamic rotate and shift control signals.

You can easily use the shift mode in an Arria II device with a soft embedded processor such as the Nios® II processor to perform the dynamic shift and rotate operation.

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.

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.

Figure 4–19 shows the shift mode configuration.

Figure 4–19. Shift Operation Mode Shown for Half-DSP Block

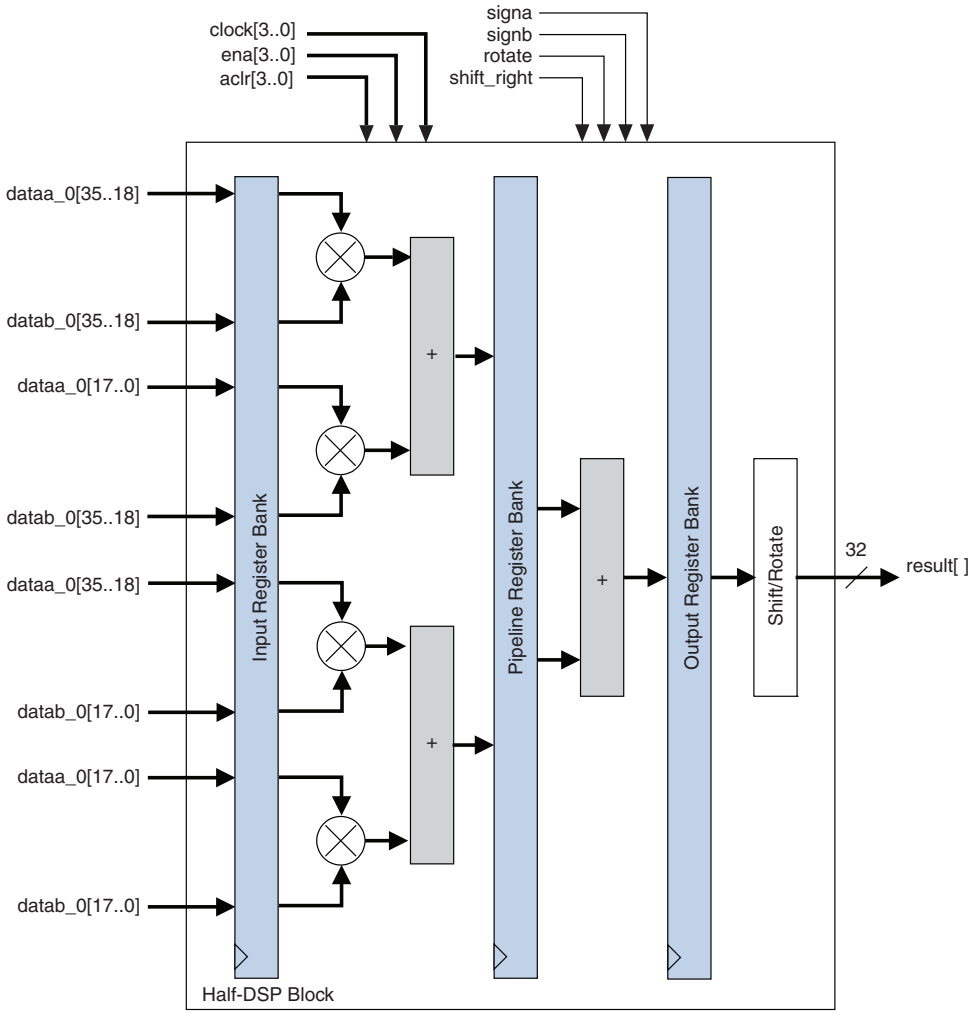


Table 4–5 lists examples of shift operations.

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

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

Two rounding modes are supported in Arria II devices:

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

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

The 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 mode provides unbiased rounding support and is used where DC offsets are a concern. Table 4–6 lists an example of how round-to-nearest-even mode. 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 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 $-2^{(n-1)}$, and 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

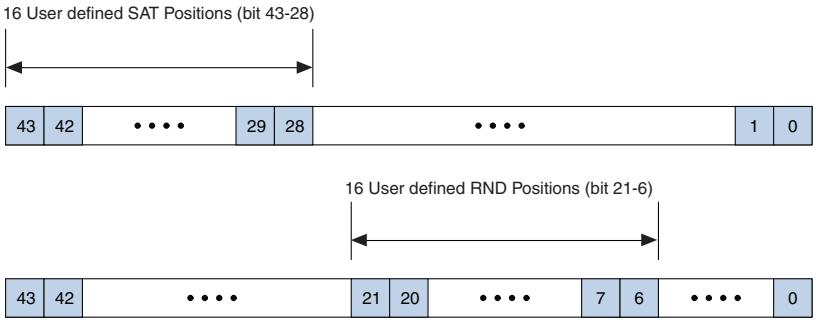
Table 4–8 lists how the saturation works. In this example, a 44-bit input is saturated to 36-bits.

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 devices have up to 16 configurable bit positions out of the 44-bit bus (`[43:0]`) for the rounding 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 4–20.

Figure 4–20. Rounding and Saturation Locations



For symmetric saturation, the RND bit position is 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 4–7. However, for accumulation type operations, the following convention is used.

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

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

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

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

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

Result = SAT[RND[$\sum$(A × B)]]

DSP Block Control Signals

You can configure the Arria II DSP block with a set of static and dynamic signals. At run time, you can configure the DSP block dynamic signals to toggle or not.

Table 4–9 shows a list of dynamic signals for the DSP block. Table 4–9 lists the DSP block dynamic signals.

Table 4–9. DSP Block Dynamic Signals for DSP Block in Arria II Devices (Part 1 of 2)

Signal Name	Function	Count
DSP Block Dynamic Signals per Half-DSP Block		
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 is enabled, saturation is done on the rounded result. ■ chainout_saturate = 1 for saturation support ■ chainout_saturate = 0 for no saturation support	1
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

Table 4–9. DSP Block Dynamic Signals for DSP Block in Arria II Devices (Part 2 of 2)

Signal Name	Function	Count
shift_right	shift_right = 1, shift right feature is enabled	1
DSP Block Dynamic Signals per Full-DSP Block		
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 Half- and Full-DSP Blocks		33

Software Support for Arria II 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 with 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. With either method, the Quartus II software maps the functionality to the DSP blocks during compilation.

For instructions about using the megafunctions and the MegaWizard Plug-In Manager, refer to the Quartus II Software Help.

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

Document Revision History

Table 4–10 shows the revision history for this document.

Table 4–10. Document Revision History

Date	Version	Changes
December 2010	4.0	■ Updated for the Quartus II software version 10.1 release. ■ Added Arria II GZ devices information. ■ Updated "DSP Block Overview", "Operational Modes Overview", and "DSP Block Resource Descriptions" sections. ■ Updated Table 4–1 ■ Added Figure 4–3, Figure 4–7, Figure 4–11, and Figure 4–15 ■ Minor text edits
July 2010	3.0	Updated for the Arria II GX v10.0 release: ■ Updated "DSP Block Resource Descriptions" and "Second-Stage Adder" sections ■ Minor text edits
November 2009	2.0	Updated for Arria II GX v9.1 release: ■ Updated Table 4–1 and Table 4–9 ■ Updated Figure 4–9 ■ Minor text edit
June 2009	1.1	Updated Table 4–1
February 2009	1.0	Initial release

5. Clock Networks and PLLs in Arria II Devices

AIIGX51005-4.2

This chapter describes the hierarchical clock networks and phase-locked loops (PLLs) which have advanced features in Arria® II devices that provide dedicated global clock networks (GCLKs), regional clock networks (RCLKs), and periphery clock networks (PCLKs). This chapter also includes details reconfiguring the PLL counter clock frequency and phase shift in real time, allowing you to sweep PLL output frequencies and dynamically adjust the output clock phase shift.

This chapter contains the following sections:

- "Clock Networks in Arria II Devices" on page 5–1
- "PLLs in Arria II Devices" on page 5–22

Clock Networks in Arria II Devices

The GCLKs, RCLKs, and PCLKs available in Arria II devices are organized into hierarchical clock structures that provide up to 192 unique clock domains (16 GCLK + 88 RCLK + 88 PCLK) and allow up to 60 unique GCLK, RCLK, and PCLK clock sources (16 GCLK + 22 RCLK + 22 PCLK) per device quadrant.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Table 5–1 lists the clock resources available in Arria II devices.

Table 5–1. Clock Resources in Arria II Devices

Clock Resource and Device	Number of Resources Available		Source of Clock Resource	
	Arria II GX	Arria II GZ	Arria II GX	Arria II GZ
Clock input pins	12 Single-ended (6 Differential)	32 Single-ended (16 Differential)	CLK[4..15], DIFFCLK_[0..5]p/n pins	CLK[0..15]p and CLK[0..15] n pins
GCLK networks	16	16	CLK[4..15] pins, PLL clock outputs, programmable logic device (PLD)-transceiver interface clocks, and logic array	CLK[0..15]p and CLK[0..15]n pins, PLL clock outputs, and logic array
RCLK networks	48	64/88 *(1)*	CLK[4..15] pins, PLL clock outputs, PLD-transceiver interface clocks, and logic array	CLK[0..15]p and CLK[0..15]n pins, PLL clock outputs, and logic array
PCLK networks	84 (24 per device quadrant) *(2)*	88 (22 per device quadrant)	Dynamic phase alignment (DPA) clock outputs, PLD-transceiver interface clocks, horizontal I/O pins, and logic array	DPA clock outputs, PLD-transceiver interface clocks, horizontal I/O pins, and logic array
GCLKs/RCLKs per quadrant	28	32/38 *(3)*	16 GCLKs + 12 RCLKs	16 GCLKs + 16 RCLKs 16 GCLKs + 22 RCLKs
GCLKs/RCLKs per device	64	80/104 *(4)*	16 GCLKs + 48 RCLKs	16 GCLKs + 64 RCLKs 16 GCLKs + 88 RCLKs

Notes to Table 5–1:

(1) There are 64 RCLKs in the EP2AGZ225 devices. There are 88 RCLKs in the EP2AGZ300 and EP2AGZ350 devices.

(2) There are 50 PCLKs in EP2AGX45 and EP2AGX65 devices, where 18 are on the left side and 32 on the right side. There are 59 PCLKs in EP2AGX95 and EP2AGX125 device, where 27 are on the left side and 32 on the right side. There are 84 PCLKs in EP2AGX190 and EP2AGX260 devices, where 36 are on the left side and 48 on the right side.

(3) There are 32 GCLKs/RCLKs per quadrant in the EP2AGZ225 devices. There are 38 GCLKs/RCLKs per quadrant in the EP2AGZ300 and EP2AGZ350 devices.

(4) There are 80 GCLKs/RCLKs per entire device in the EP2AGZ225 devices. There are 104 GCLKs/RCLKS per entire device in the EP2AGZ300 and EP2AGZ350 devices.

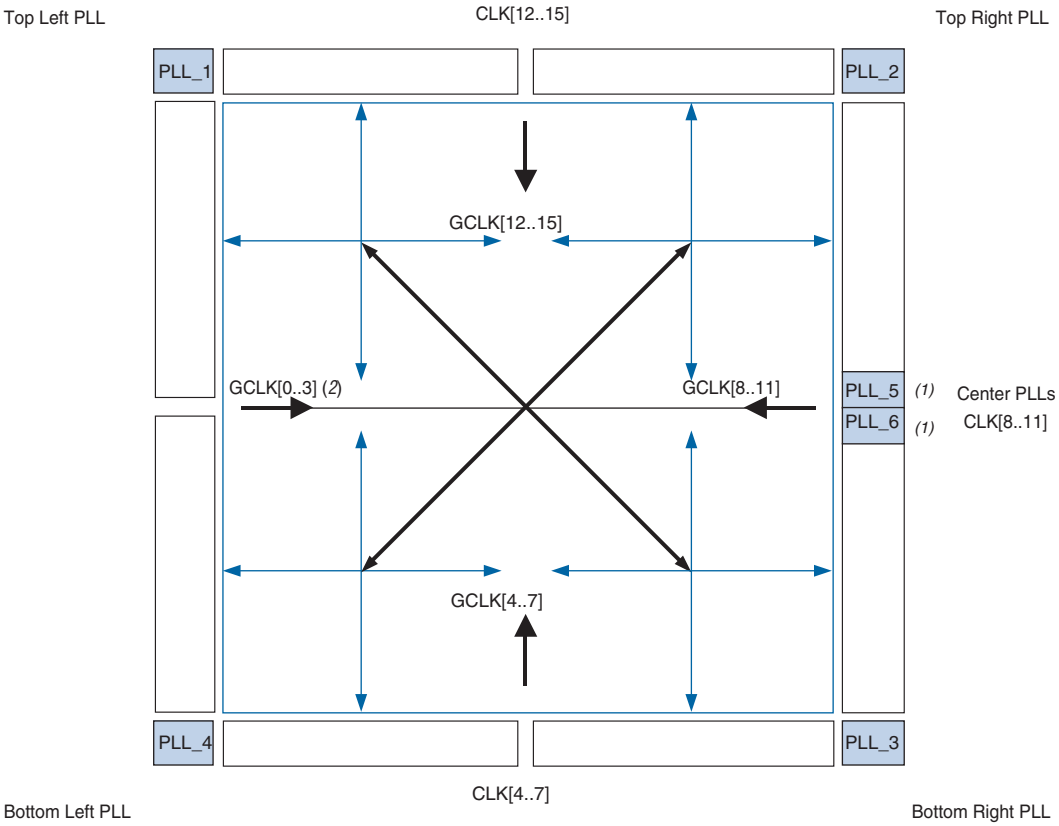
Arria II GX devices have up to 12 dedicated single-ended clock pins or six dedicated differential clock pins (DIFFCLK_[0..5]p and DIFFCLK_[0..5]n) that can drive either the GCLK or RCLK networks. These clock pins are arranged on the three sides (top, bottom, and right sides) of the Arria II GX device, as shown in Figure 5–1 on page 5–4 and Figure 5–3 on page 5–6.

Arria II GZ devices have up to 32 dedicated single-ended clock pins or 16 dedicated differential clock pins (CLK[0..15]p and CLK[0..15]n) that can drive either the GCLK or RCLK networks. These clock pins are arranged on the four sides of the Arria II GZ device, as shown in Figure 5–2 on page 5–5 and Figure 5–4 on page 5–6.

Global Clock Networks

Arria II devices provide up to 16 GCLKs that can drive throughout the device, serving as low-skew clock sources for functional blocks such as adaptive logic modules (ALMs), digital signal processing (DSP) blocks, embedded memory blocks, and PLLs. Arria II I/O elements (IOEs) and internal logic can drive GCLKs to create internally generated GCLKs and other high fan-out control signals; for example, synchronous or asynchronous clears and clock enables. Figure 5–1 and Figure 5–2 show CLK pins and PLLs that can drive GCLK networks in Arria II devices.

Figure 5–1. GCLK Networks in Arria II GX Devices

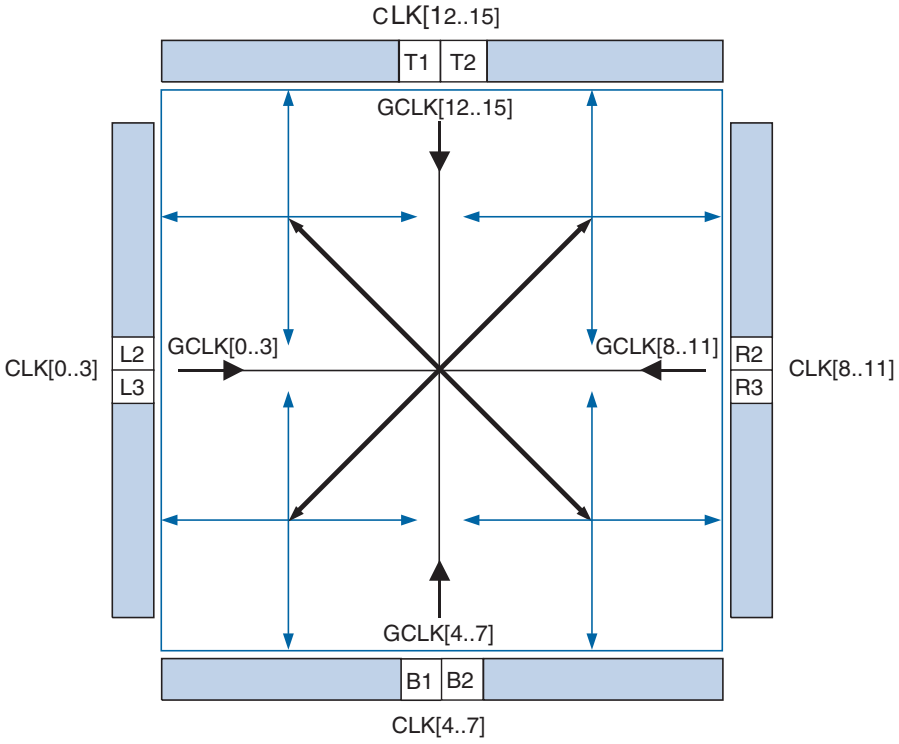


Notes to Figure 5–1:

(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.

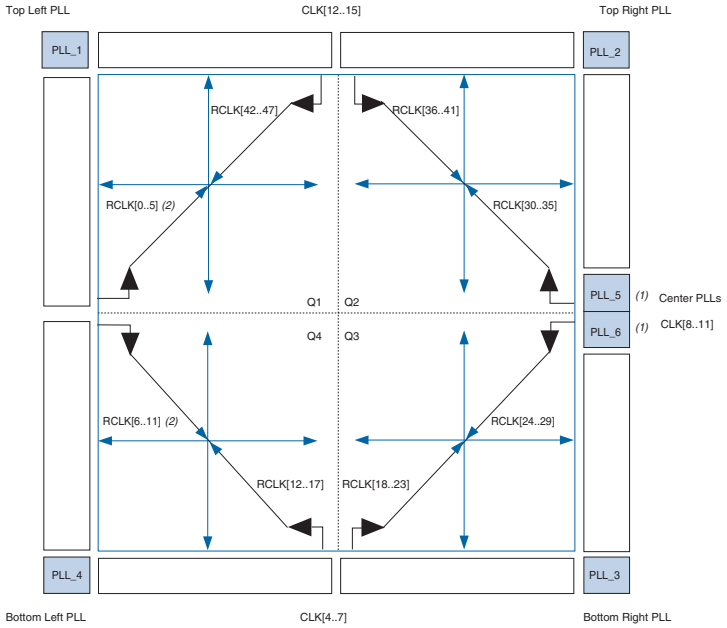
Figure 5–2. GCLK Networks in Arria II GZ Devices



Regional Clock Networks

For Arria II devices, the RCLK networks only pertain to the quadrant they drive into. RCLK networks provide the lowest clock delay and skew for logic contained in a single device quadrant. Arria II IOEs and internal logic in a given quadrant can also drive RCLKs to create internally generated RCLKs and other high fan-out control signals; for example, synchronous or asynchronous clears and clock enables. Figure 5–3 and Figure 5–4 show CLK pins and PLLs that can drive RCLK networks in Arria II devices.

Figure 5–3. RCLK Networks in Arria II GX Devices

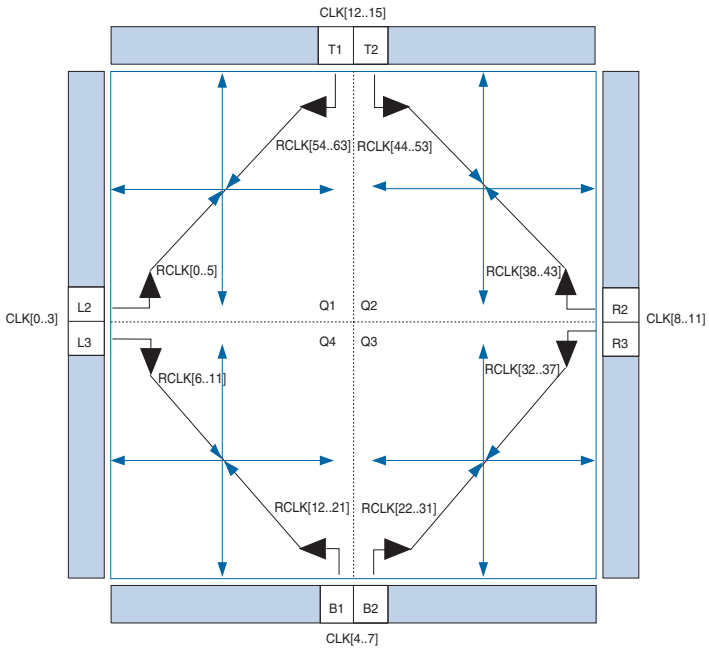


Notes to Figure 5–3:

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

(2) `RCLK[0..5]` is not driven by any clock pins because there are no dedicated clock pins on the left side of the Arria II GX devices.

Figure 5–4. RCLK Networks in Arria II GZ Devices *(Note 1)*



Note to Figure 5–4:

(1) A maximum of four signals from the core can drive into each group of RCLKs. For example, only four core signals can drive into `RCLK[0..5]` and another four core signals can drive into `RCLK[54..63]` at any one time.

Periphery Clock Networks

PCLK networks are a collection of individual clock networks driven from the periphery of the Arria II device. Clock outputs from the DPA block, PLD-transceiver interface clocks, I/O pins, and internal logic can drive the PCLK networks. Figure 5–5 through Figure 5–8 show CLK pins and PLLs that can drive PCLK networks in Arria II devices.

The number of PCLKs for each Arria II device are as follows:

- EP2AGX45 and EP2AGX65 devices contain 50 PCLKs
- EP2AGX95 and EP2AGX125 devices contain 59 PCLKs
- EP2AGX190 and EP2AGX260 devices contain 84 PCLKs
- EP2AGZ225, EP2AGZ300, and EP2AGZ350 devices contain 88 PCLKs

PCLKs have higher skew when compared with the GCLK and RCLK networks. You can use PCLKs instead of general purpose routing to drive signals into the Arria II device.

Figure 5–5. PCLK Networks (EP2AGX45 and EP2AGX65 Devices)

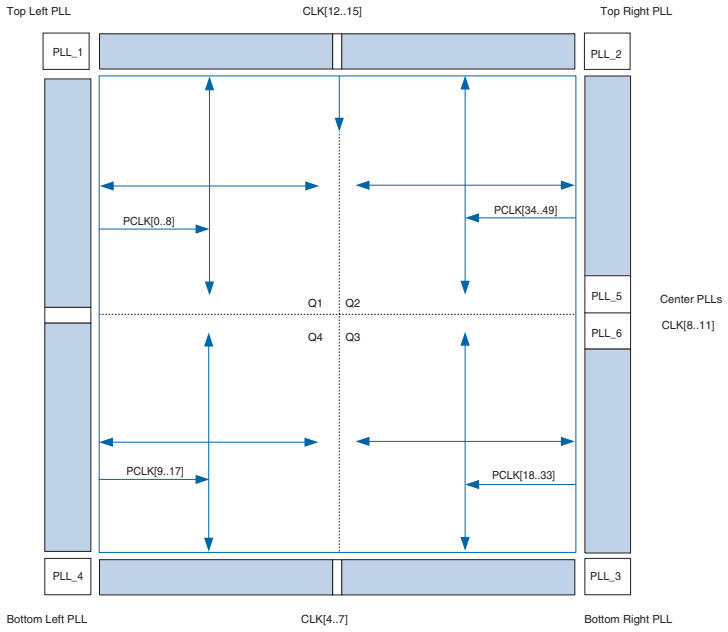


Figure 5–6. PCLK Networks in (EP2AGX95 and EP2AGX125 Devices)

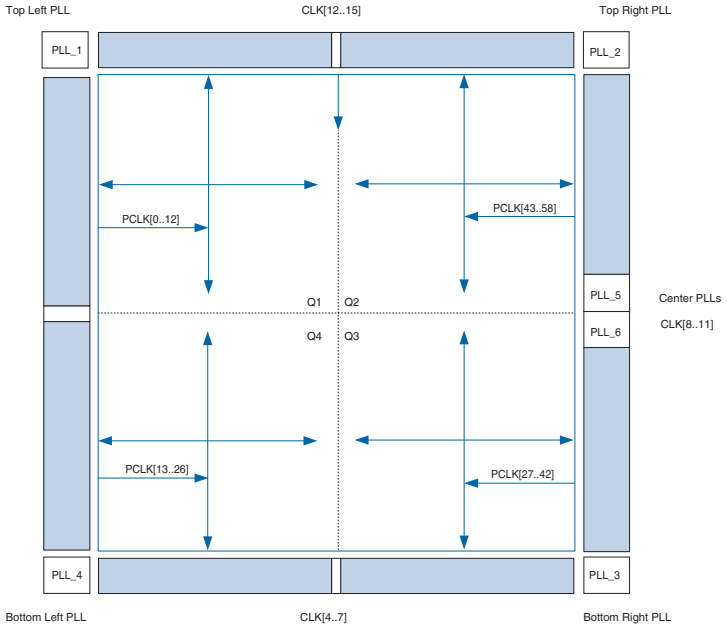


Figure 5–7. PCLK Networks in (EP2AGX190 and EP2AGX260 Devices)

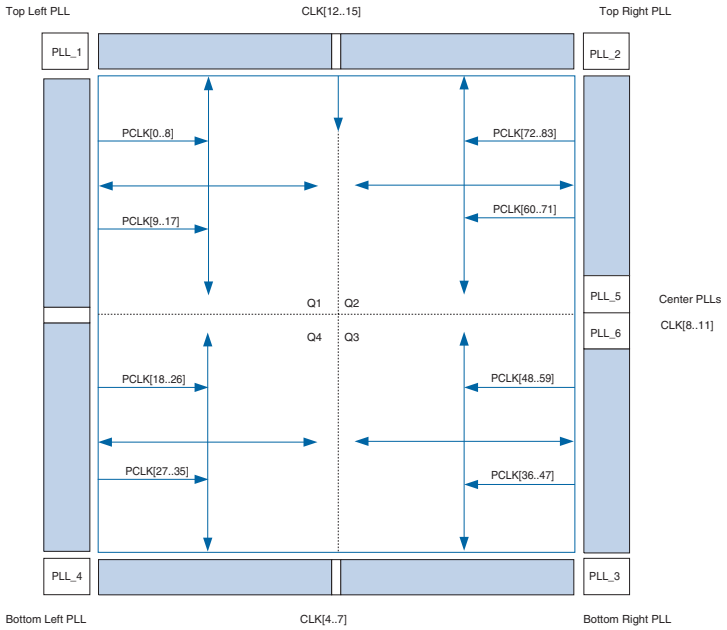
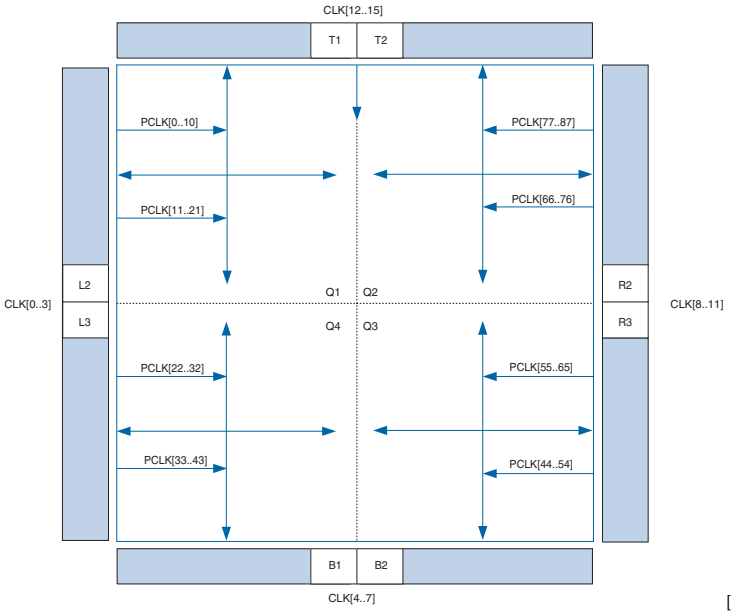


Figure 5–8. PCLK Networks in Arria II GZ Devices

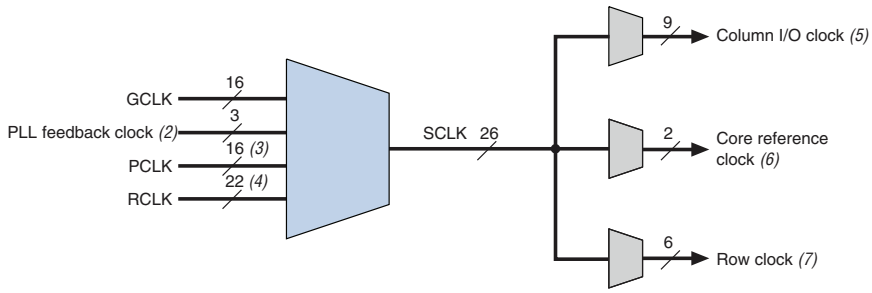


Clock Sources Per Quadrant

There are 26 section clock (SCLK) networks available in each spine clock that can drive six row clocks in each logic array block (LAB) row, nine column I/O clocks, and three core reference clocks. SCLKs are the clock resources to the core functional blocks, PLLs, and I/O interfaces of the device.

Figure 5–9 shows that the GCLK, RCLK, PCLK, or PLL feedback clock networks in each spine clock can drive the SCLKs.

Figure 5–9. Hierarchical Clock Networks per Spine Clock in Arria II Devices *(Note 1)*



Notes to Figure 5–9:

(1) The GCLK, RCLK, PCLK, and PLL feedback clocks share the same routing to the SCLKs. The total number of clock resources must not exceed the SCLK limits in each region to ensure successful design fitting in the Quartus® II software.

(2) There are up to three PLL feedback clocks which are from the PLL that drives into the SCLKs.

(3) There are up to 16 PCLKs that can drive the SCLKs in each spine clock in the largest device.

(4) There are up to 22 RCLKs (Arria II GZ) or 12 RCLKs (Arria II GX) that can drive the SCLKs in each spine clock in the largest device.

(5) The column I/O clock drives the column I/O core registers and I/O interfaces.

(6) The core reference clock feeds into the PLL as the PLL reference clock.

(7) The row clock is the clock source to the LAB, memory blocks, and row I/O interfaces in the core row.



A spine clock is another layer of routing below the GCLKs, RCLKs, and PCLKs before each clock is connected to the clock routing for each LAB row. The settings for spine clocks are transparent. The Quartus II software automatically routes the spine clock based on the GCLK, RCLK, and PCLKs.

Clock Regions

Arria II GX devices provide up to 64 distinct clock domains (16 GCLKs + 48 RCLKs) in the entire device, while Arria II GZ devices provide up to 104 distinct clock domains (16 GCLKs + 88 RCLKs). Use these clock resources to form the following three types of clock regions:

- Entire device
- Regional
- Dual regional

To form the entire device clock region, a source (not necessarily a clock signal) drives a GCLK network that can be routed through the entire device. This clock region has a higher skew when compared with other clock regions, but allows the signal to reach every destination in the device. This is a good option for routing global reset and 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 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. For Arria II GX devices, corner PLL outputs generate a dual-regional clock network through clock multiplexers that serve the two immediate quadrants of the device. For Arria II GZ devices, corner PLL outputs only span one quadrant, they cannot generate a dual-regional clock network.

Figure 5–10 and Figure 5–11 show the dual-regional clock region for Arria II devices.

Figure 5–10. Device Dual-Regional Clock Region for Arria II GX Devices

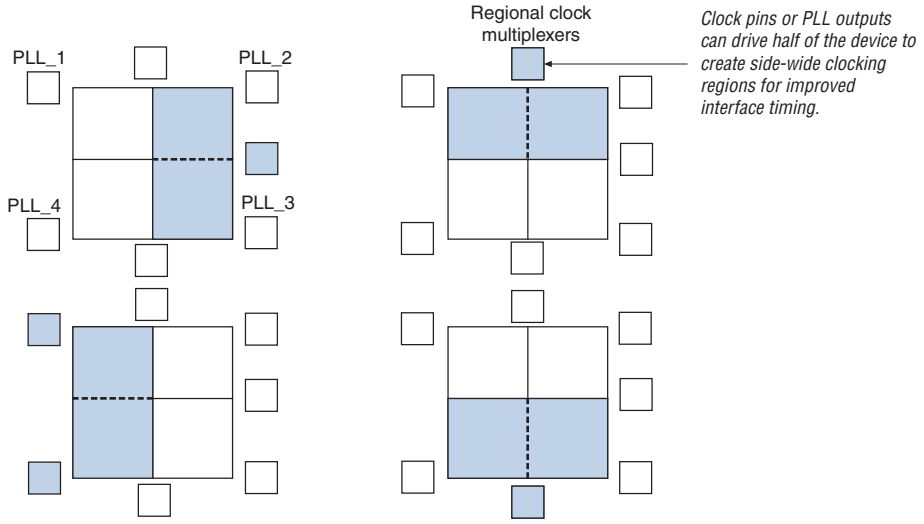
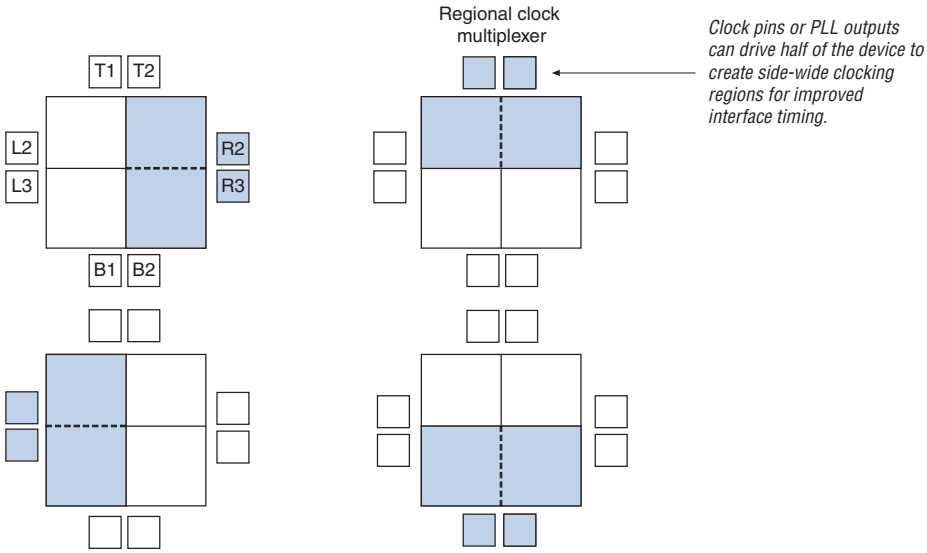


Figure 5–11. Device Dual-Regional Clock Region for Arria II GZ Devices



Clock Network Sources

In Arria II GX devices, clock input pins, internal logic, transceiver clocks, and PLL outputs can drive the GCLK and RCLK networks, while in Arria II GZ devices, clock input pins, PLL outputs, and internal logic can drive the GCLK and RCLK networks. Table 5–2 through Table 5–5 on page 5–13 list the connectivity between the dedicated clock pins and the GCLK and RCLK networks.

Dedicated Clock Inputs Pins

CLK pins can either be differential clocks or single-ended clocks. Arria II GX devices support six differential clock inputs or 12 single-ended clock inputs, while Arria II GZ devices support 16 differential clock inputs or 32 single-ended clock inputs. You can also use the dedicated clock input pins `CLK[4..15]` (for Arria II GX devices) and `CLK[15..0]` (for Arria II GZ devices) for high fan-out control signals such as asynchronous clears, presets, and clock enables for protocol signals such as `TRDY` and `IRDY` for PCI Express® (PCIe®) through GCLK or RCLK networks.

Logic Array Blocks

You can drive up to four signals into each GCLK and RCLK network with logic array block (LAB)-routing to allow internal logic to drive a high fan-out, low-skew signal.

☞ You cannot drive Arria II PLLs by internally generated GCLKs or RCLKs. The input clock to the PLL has to come from dedicated clock input pins or PLL-fed GCLKs and RCLKs only.

PLL Clock Outputs

Table 5–2 and Table 5–3 list the connection between the dedicated clock input pins and GCLKs.

Table 5–2. Clock Input Pin Connectivity to GCLK Networks for Arria II GX Devices

Clock Resources	CLK (p/n 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 device.

Table 5–3. Clock Input Pin Connectivity to the GCLK Networks for Arria II GZ Devices (Part 1 of 2)

Clock Resources	CLK (p/n Pins)															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
`GCLK[0..3]`	✓	✓	✓	✓	—	—	—	—	—	—	—	—	—	—	—	—
`GCLK[4..7]`	—	—	—	—	✓	✓	✓	✓	—	—	—	—	—	—	—	—

Table 5–3. Clock Input Pin Connectivity to the GCLK Networks for Arria II GZ Devices (Part 2 of 2)

Clock Resources	CLK (p/n Pins)															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
GCLK[8..11]	—	—	—	—	—	—	—	—	✓	✓	✓	✓	—	—	—	—
GCLK[12..15]	—	—	—	—	—	—	—	—	—	—	—	—	✓	✓	✓	✓

Table 5–4 and Table 5–5 list the connectivity between the dedicated clock input pins and RCLKs in Arria II devices. A given clock input pin can drive two adjacent RCLK networks to create a dual-RCLK network.

Table 5–4. Clock Input Pin Connectivity to RCLK Networks for Arria II GX Devices

Clock Resource	CLK (p/n 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]	—	—	—	—	—	—	—	—	—	✓	—	✓

Table 5–5. Clock Input Pin Connectivity to the RCLK Networks for Arria II GZ Devices (Part 1 of 2)

Clock Resource	CLK (p/n Pins)															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RCLK [0, 4, 6, 10]	✓	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [1, 5, 7, 11]	—	✓	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [2, 8]	—	—	✓	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [3, 9]	—	—	—	✓	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [13, 17, 21, 23, 27, 31]	—	—	—	—	✓	—	—	—	—	—	—	—	—	—	—	—
RCLK [12, 16, 20, 22, 26, 30]	—	—	—	—	—	✓	—	—	—	—	—	—	—	—	—	—
RCLK [15, 19, 25, 29]	—	—	—	—	—	—	✓	—	—	—	—	—	—	—	—	—
RCLK [14, 18, 24, 28]	—	—	—	—	—	—	—	✓	—	—	—	—	—	—	—	—
RCLK [35, 41]	—	—	—	—	—	—	—	—	✓	—	—	—	—	—	—	—
RCLK [34, 40]	—	—	—	—	—	—	—	—	—	✓	—	—	—	—	—	—
RCLK [33, 37, 39, 43]	—	—	—	—	—	—	—	—	—	—	✓	—	—	—	—	—
RCLK [32, 36, 38, 42]	—	—	—	—	—	—	—	—	—	—	—	✓	—	—	—	—
RCLK [47, 51, 57, 61]	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	—	—
RCLK [46, 50, 56, 60]	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	—

Table 5–5. Clock Input Pin Connectivity to the RCLK Networks for Arria II GZ Devices (Part 2 of 2)

Clock Resource	CLK (p/n Pins)															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RCLK [45, 49, 53, 55, 59, 63]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—
RCLK [44, 48, 52, 54, 58, 62]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓

Clock Input Connections to PLLs

Table 5–6 and Table 5–7 list dedicated clock input pin connectivity to Arria II PLLs.

Table 5–6. PLLs and PLL Clock Pin Drivers for Arria II GX Devices *(Note 1)*

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

Note to Table 5–6:

(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.

Table 5–7. PLLs and PLL Clock Pin Drivers for Arria II GZ Devices *(Note 1)*, *(2)*

Dedicated Clock Input Pin CLK (p/n Pins)	PLL Number							
	L2	L3	B1	B2	R2	R3	T1	T2
CLK[0..3]	✓	✓	—	—	—	—	—	—
CLK[4..7]	—	—	✓	✓	—	—	—	—
CLK[8..11]	—	—	—	—	✓	✓	—	—
CLK[12..15]	—	—	—	—	—	—	✓	✓

Notes to Table 5–7:

(1) For single-ended clock inputs, only the CLK<#>p pin has a dedicated connection to the PLL. If you use the CLK<#>n pin, a GCLK is used.

(2) For the availability of the clock input pins in each device density, refer to the "Arria II Device Pin-Out Files" section of the Pin-Out Files for Altera Devices.

Clock Output Connections

PLLs in Arria II GX devices can drive up to 24 RCLK networks and eight GCLK networks, while PLLs in Arria II GZ devices can drive up to 20 RCLK networks and four GCLK networks. The Quartus II software automatically assigns PLL clock outputs to RCLK or GCLK networks.

Table 5–8 and Table 5–9 list the Arria II PLL connectivity to GCLK networks.

Table 5–8. PLL Connectivity to GCLKs for Arria II GX Devices

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

Table 5–9. PLL Connectivity to the GCLK Networks for Arria II GZ Devices *(Note 1)*

Clock Network	PLL Number							
	L2	L3	B1	B2	R2	R3	T1	T2
GCLK[0..3]	✓	✓	—	—	—	—	—	—
GCLK[4..7]	—	—	✓	✓	—	—	—	—
GCLK[8..11]	—	—	—	—	✓	✓	—	—
GCLK[12..15]	—	—	—	—	—	—	✓	✓

Note to Table 5–9:

(1) Only PLL counter outputs C0 - C3 can drive the GCLK networks.

Table 5–10 and Table 5–11 list how the PLL clock outputs connect to RCLK networks.

Table 5–10. RCLK Outputs from PLLs for Arria II GX Devices

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

Table 5–11. RCLK Outputs From the PLL Clock Outputs for Arria II GZ Device (Part 1 of 2)

Clock Resource	PLL Number							
	L2	L3	B1	B2	R2	R3	T1	T2
RCLK[0..11]	✓	✓	—	—	—	—	—	—
RCLK[12..31]	—	—	✓	✓	—	—	—	—

Table 5–11. RCLK Outputs From the PLL Clock Outputs for Arria II GZ Device (Part 2 of 2)

Clock Resource	PLL Number							
	L2	L3	B1	B2	R2	R3	T1	T2
RCLK[32..43]	—	—	—	—	✓	✓	—	—
RCLK[44..63]	—	—	—	—	—	—	✓	✓

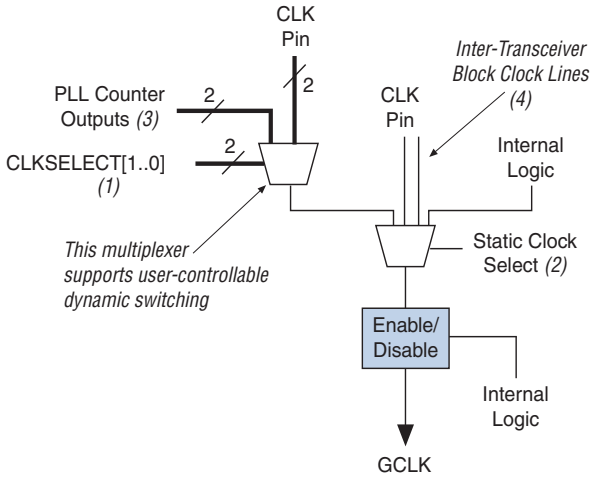
Clock Control Block

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

- Clock source selection (dynamic selection for GCLKs)
- GCLK multiplexing
- Clock power down (static or dynamic clock enable or disable)

Figure 5–12 shows the GCLK select blocks for Arria II devices.

Figure 5–12. GCLK Control Block for Arria II Devices



Notes to Figure 5–12:

(1) You can only dynamically control these clock select signals 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 GCLK network.

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

Select the clock source for the GCLK control block either statically with a setting in the Quartus II software or dynamically with an 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.

Table 5–12 lists the mapping between the input clock pins, PLL counter outputs, and clock control block inputs.

Table 5–12. Mapping Between Input Clock Pins, PLL Counter Outputs, and Clock Control Block Inputs for Arria II Devices

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

Note to Table 5–12:

(1) The left side of the Arria II GX device only allows PLL counter outputs as the dynamic clock source selection to the GCLK network. Therefore, 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, ensure that these clock sources are implemented on the same side of the device.

For all possible legal inclk sources for each GCLK and RCLK network, refer to Table 5–2 on page 5–12 through Table 5–10 on page 5–15.

You can statically control the clock source selection for the RCLK select block with configuration bit settings in the configuration file generated by the Quartus II software.

You can power down the Arria II clock networks both statically and dynamically. When a clock network is powered down, all the logic fed by the clock network is in an off-state, thereby reducing the overall power consumption of the device. The unused GCLK and RCLK networks are automatically powered down through configuration bit settings in the configuration file 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. This function is independent of the PLL and is applied directly on the clock network, as shown in Figure 5–12 on page 5–16 through Figure 5–14 on page 5–18.

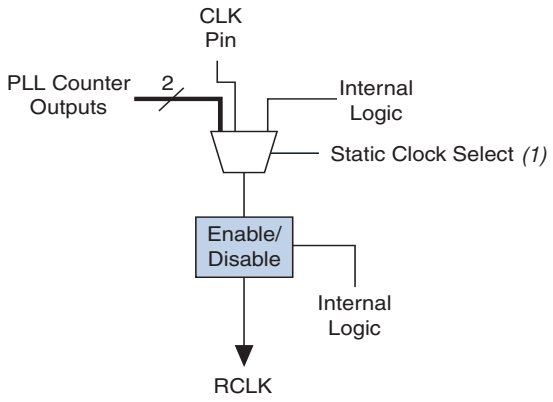
You can set the input clock sources and the clkena signals for the GCLK and RCLK clock network multiplexers through the Quartus II software with the ALTCLKCTRL megafunction. You can also enable or disable the dedicated external clock output pins with the ALTCLKCTRL megafunction.

When you use the ALTCLKCTRL megafunction to implement dynamic clock source selection in Arria II devices, the inputs from the clock pins, except for the left side of the Arria II GX device, feed the inclk[0..1] ports of the multiplexer, and the PLL outputs feed the inclk[2..3] ports. You can choose from among these inputs with the CLKSELECT[1..0]signal. For the connections between the PLL counter outputs to the clock control block, refer to Table 5–12 on page 5–17.

For more information, refer to the *Clock Control Block (ALTCLKCTRL) Megafunction User Guide*.

Figure 5–13 and Figure 5–14 show the RCLK select blocks.

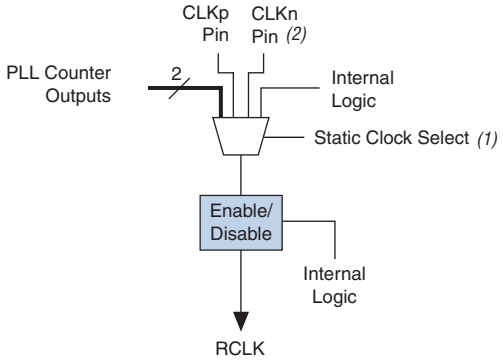
Figure 5–13. RCLK Control Block for Arria II GX Devices



Note to Figure 5–13:

(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.

Figure 5–14. RCLK Control Block for Arria II GZ Devices



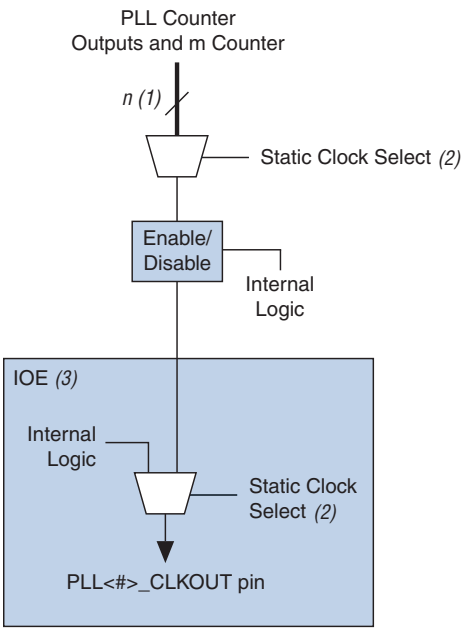
Notes to Figure 5–14:

(1) When the device is in user mode, you can only set the clock select signals through a configuration file (**.sof** or **.pof**). You cannot dynamically control the clock.

(2) The `CLKn` pin is not a dedicated clock input when used as a single-ended PLL clock input.

Figure 5–15 shows the external PLL output clock control block.

Figure 5–15. External PLL Output Clock Control Block Arria II Devices



Notes to Figure 5–15:

(1) For Arria II GX devices, n = 8; for Arria II GZ devices, n = 8 or 11.

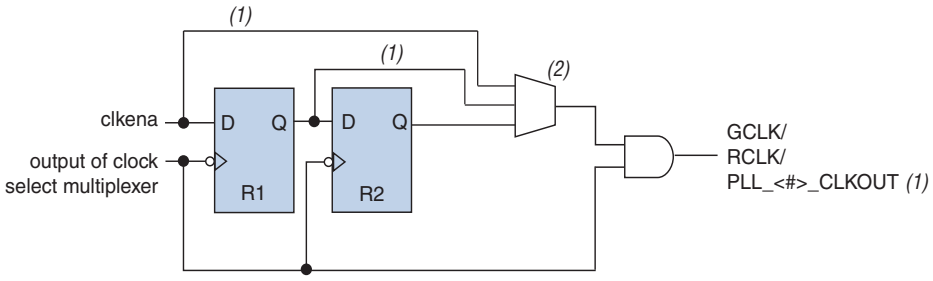
(2) When the device is in user mode, you can only set the clock select signals through a configuration file (**.sof** or **.pof**). You cannot dynamically control the clock.

(3) The clock control block feeds 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 5–16 shows how the clock enable/disable circuit of the clock control block is implemented in Arria II devices.

Figure 5–16. clkena Implementation for Arria II Devices



Notes to Figure 5–16:

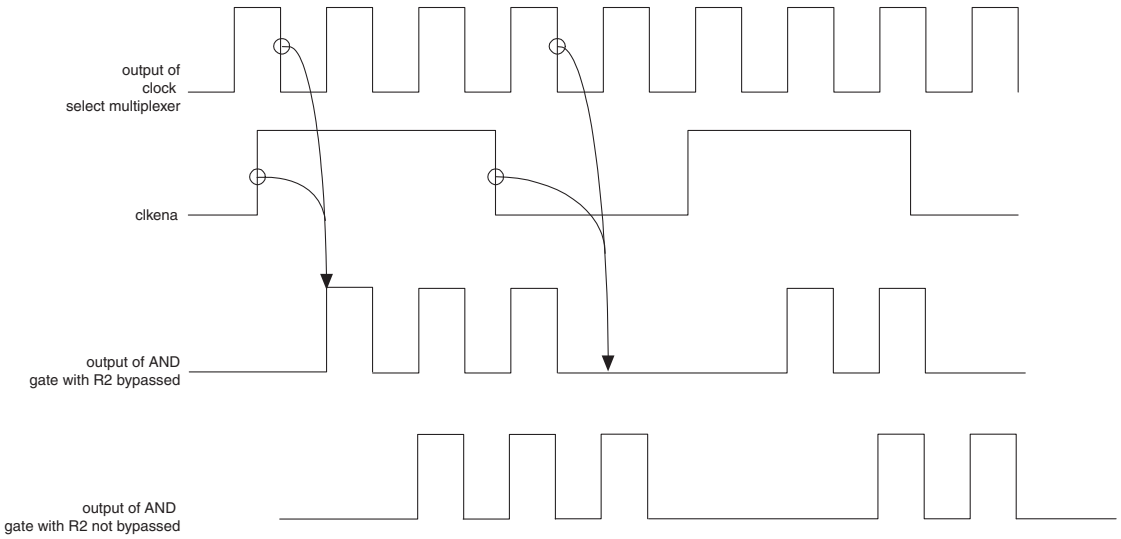
(1) The R1 and R2 bypass paths are not available for PLL external clock outputs.

(2) The select line is statically controlled by a bit setting in the configuration file (**.sof** or **.pof**).

In Arria II 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 used. You can also use the clkena signals to control the dedicated external clocks from the PLLs. Arria II devices also have an additional metastability register that aids in asynchronous enable or disable of the GCLK and RCLK networks. You can optionally bypass this register in the Quartus II software.

Figure 5–17 shows a waveform example for the clock output enable. The clkena signal is synchronous to the falling edge of the clock output.

Figure 5–17. clkena Signals for Arria II Devices



Note to Figure 5–17:

(1) You can use the clkena signals to enable or disable the GCLK and RCLK 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.

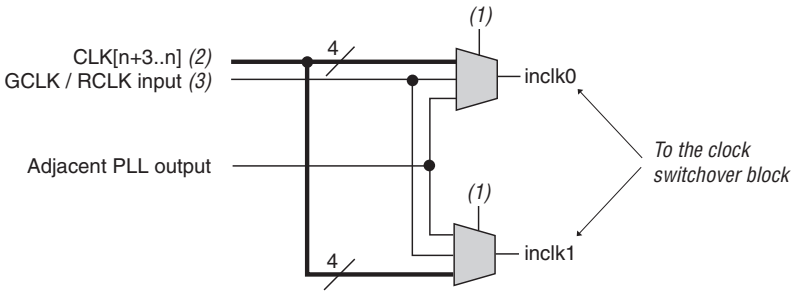
Clock Source Control for PLLs

The clock input to Arria II 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 (Arria II GX devices) or from dedicated connections between adjacent top/bottom and left/right PLLs (Arria II GZ devices). For Arria II GX devices, 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 5–18. For Arria II GZ devices, the clock input sources to top/bottom and left/right PLLs (L2, L3, T1, T2, B1, B2, R2, and R3) are shown in Figure 5–19.

The multiplexer select lines are set in the configuration file only. When configured, you cannot change this block 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 about the clock control block and its supported features in the Quartus II software, refer to the *Clock Control Block (ALTCLKCTRL) Megafunction User Guide*.

Figure 5–18. Clock Input Multiplexer Logic for Arria II GX PLLs



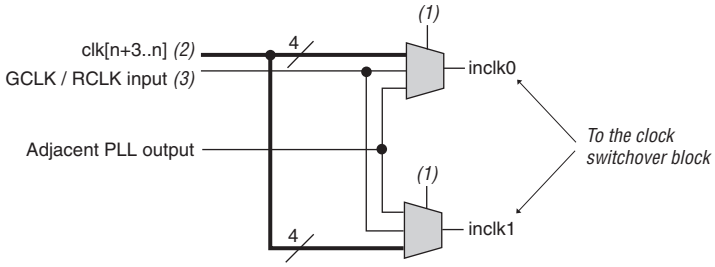
Notes to Figure 5–18:

(1) Input clock multiplexing is controlled through a configuration file (**.sof** or **.pof**) only; it cannot be dynamically controlled when the device is operating in user mode.

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

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

Figure 5–19. Clock Input Multiplexer Logic for Arria II GZ devices



Notes to Figure 5–19:

(1) When the device is operating in user mode, input clock multiplexing is controlled through a configuration file (**.sof** or **.pof**) only and cannot be dynamically controlled.

(2) n = 0 for L2 and L3 PLLs; n = 4 for B1 and B2 PLLs; n = 8 for R2 and R3 PLLs, and n = 12 for T1 and T2 PLLs.

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

Cascading PLLs

You can cascade the corner and center PLLs through the GCLK and RCLK networks (Arria II GX devices) or left/right and top/bottom PLLs through the GCLK and RCLK networks (Arria II GZ devices). In addition, where two PLLs exist next to each other, there is a direct connection between them that does not require the GCLK and RCLK network. By cascading PLLs, you can use this path to reduce clock jitter. For Arria II GX devices, the direct PLL cascading feature is available in `PLL_5` and `PLL_6` on the right side of EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260 devices. Arria II GX devices allow cascading of `PLL_1` and `PLL_4` to the transceiver PLLs (clock management unit PLLs and receiver clock data recoveries [CDRs]). Arria II GZ devices allows cascading the left and right PLLs to transceiver PLLs (CMU PLLs and receiver CDRs).

If your design cascades PLLs, the source (upstream) PLL must have a low-bandwidth setting, while the destination (downstream) PLL must have a high-bandwidth setting. Ensure that there is no overlap of the bandwidth ranges of the two PLLs.

For more information, refer to the "FPGA Fabric PLLs-Transceiver PLLs Cascading" section in the *Transceiver Clocking in Arria II Devices* chapter.

For more information about PLL cascading in external memory interfaces designs, refer to the *External Memory PHY Interface (ALTMEMPHY) (nonAFI) Megafunction User Guide*.

PLLs in Arria II Devices

Arria II GX devices offer up to six PLLs per device and seven outputs per PLL, while Arria II GZ devices offer up to eight PLLs 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. For the location and number of PLLs in Arria II devices, refer to Figure 5–1 on page 5–4 through Figure 5–4 on page 5–6.

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

For more information about the number of general-purpose and transceiver TX PLLs in each device density, refer to the *Overview for Arria II Device Family* chapter. For more information about using the transceiver TX PLLs in the transceiver block, refer to the *Transceiver Clocking in Arria II Devices* chapter.

All Arria II PLLs have the same core analog structure and support features with minor differences in the features that are supported for Arria II GZ devices.

Table 5–13 lists the PLL features in Arria II devices.

Table 5–13. PLL Features in Arria II Devices

Feature	Arria II GX PLLs	Arria II GZ PLLs	
		Top/Bottom PLLs	Left/Right PLLs
C (output) counters	7	10	7
M, N, C counter sizes	1 to 512	1 to 512	1 to 512
Dedicated clock outputs	1 single-ended or 1 differential pair 3 single-ended or 3 differential pairs *(1)*, *(2)*	6 single-ended or 4 single-ended and 1 differential pair	2 single-ended or 1 differential pair
Clock input pins	4 single-ended or 2 differential pin pairs	4 single-ended or 2 differential pin pairs	4 single-ended or 2 differential pin pairs
External feedback input pin	No	Single-ended or differential	Single-ended only
Spread-spectrum input clock tracking	Yes *(3)*	Yes *(3)*	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.	Through GCLK and RCLK and a dedicated path between adjacent PLLs	Through GCLK and RCLK and dedicated path between adjacent PLLs *(4)*
Compensation modes	All except external feedback mode when you use differential I/Os	All except LVDS clock network compensation	All except external feedback mode when you use differential I/Os
PLL drives DIFFCLK and LOADEN	Yes	No	Yes
VCO output drives DPA clock	Yes	No	Yes
Phase shift resolution	Down to 96.125 ps *(5)*	Down to 96.125 ps *(5)*	Down to 96.125 ps *(5)*
Programmable duty cycle	Yes	Yes	Yes
Output counter cascading	Yes	Yes	Yes
Input clock switchover	Yes	Yes	Yes

Notes to Table 5–13:

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

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

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

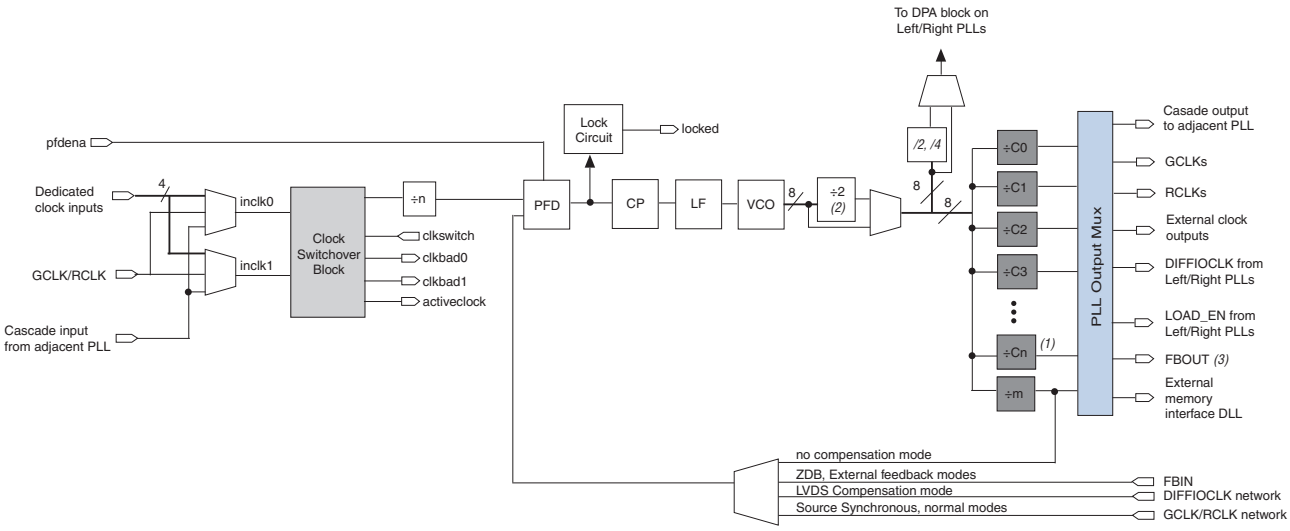
(4) The dedicated path between adjacent PLLs is not available on L1, L4, R1, and R4 PLLs.

(5) The smallest phase shift is determined by the voltage-controlled oscillator (VCO) period divided by eight. For degree increments, the Arria II 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.

PLL Hardware Overview in Arria II Devices

Figure 5–20 shows a simplified block diagram of the major components of the Arria II PLL.

Figure 5–20. PLL Block Diagram for Arria II Devices



Notes to Figure 5–20:

(1) The number of post-scale counters is seven for left and right PLLs and ten for top and bottom PLLs.
(2) This is the VCO post-scale counter K.
(3) The FBOUT port is fed by the M counter in Arria II PLLs. The FBOUT port is only available in Arria II GZ devices.



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

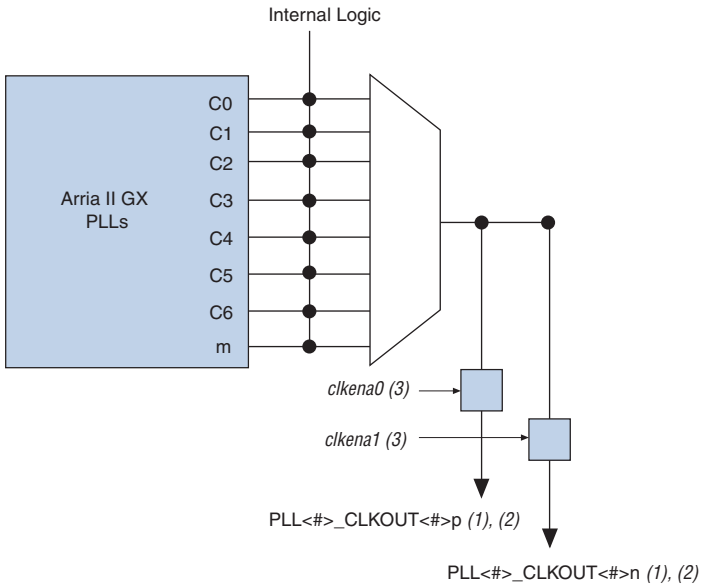
PLL Clock I/O Pins

For Arria II GX devices, each PLL supports one of the following clock I/O pin configurations:

- One single-ended I/O or one differential I/O pair.
- Three single-ended I/O or three differential I/O pairs (this is only supported in PLL_1 and PLL_3 of EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260 devices). You can only access one differential I/O pair or one single-ended pin at a time.

Figure 5–21 shows the clock I/O pins associated with Arria II GX PLLs.

Figure 5–21. External Clock Outputs for Arria II GX PLLs



Notes to Figure 5–21:

(1) You can feed these clock output pins with any one of the `C[6..0]`, or `m` counters.

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

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

For Arria II GX devices, any of the output counters (`C[6..0]`) or the `M` counter can feed the dedicated external clock outputs, as shown in Figure 5–21. Therefore, one counter or frequency can drive all the output pins available from a given PLL.

For Arria II GZ devices, each top and bottom PLL supports six clock I/O pins, organized as three pairs of pins:

- 1st pair—two single-ended I/O or one differential I/O
- 2nd pair—two single-ended I/O or one differential external feedback input (FBp/FBn)
- 3rd pair—two single-ended I/O or one differential input

Figure 5–22 shows the clock I/O pins associated with the top and bottom PLLs.

Figure 5–22. External Clock Outputs for Top and Bottom PLLs in Arria II GZ Devices



Notes to Figure 5–22:

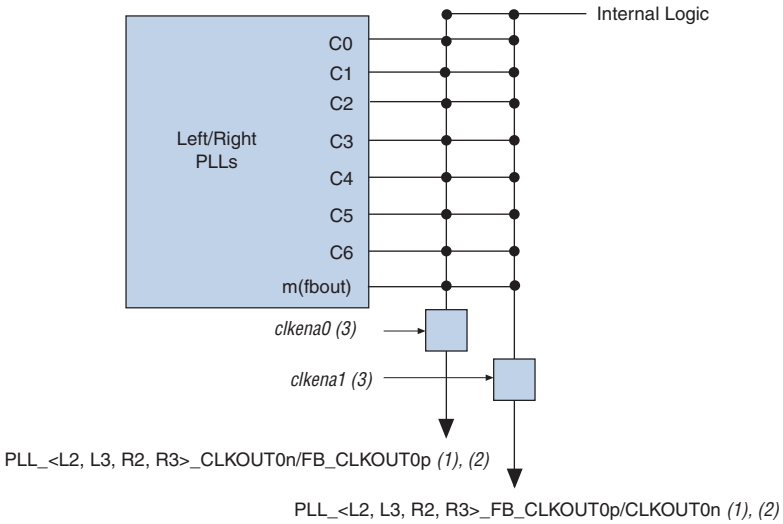
(1) You can feed these clock output pins using any one of the C[9..0], or m counters.

(2) The CLKOUT0p and CLKOUT0n pins can be either single-ended or differential clock outputs. The CLKOUT1 and CLKOUT2 pins are dual-purpose I/O pins that you can use as two single-ended outputs or one differential external feedback input pin. The CLKOUT3 and CLKOUT4 pins are two single-ended output pins.

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

For Arria II GZ devices, any of the output counters (C[9..0] on the top and bottom PLLs and C[6..0] on the left and right PLLs) or the M counter can feed the dedicated external clock outputs, as shown in Figure 5–22 and Figure 5–23. Therefore, one counter or frequency can drive all the output pins available from a given PLL. Each left and right PLL supports two clock I/O pins, configured as either two single-ended I/Os or one differential I/O pair. When using both pins as single-ended I/Os, one of them can be the clock output while the other pin is the external feedback input (FB) pin. Therefore, for single-ended I/O standards, the left and right PLLs only support external feedback mode.

Figure 5–23. External Clock Outputs for Left and Right PLLs in Arria II GZ Devices



Notes to Figure 5–23:

(1) You can feed these clock output pins using any one of the `C[6..0]`, or `m` counters.

(2) The `CLKOUT0p` and `CLKOUT0n` pins are dual-purpose I/O pins that you can use as two single-ended outputs or one single-ended output and one external feedback input pin.

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

Each pin of a single-ended output pair can either be in-phase or 180° out-of-phase. The Quartus II software places the NOT gate in your design into the IOE to implement a 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_E_3R, LVPECL, differential high-speed transceiver logic (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 Devices* chapter.

Arria II PLLs can also drive out to any regular I/O pin through the GCLK or RCLK network. You can also use the external clock output pins as user I/O pins if you do not require external PLL clocking. However, external clock output pins can support a differential I/O standard that is only driven by a PLL.

Regular I/O pins cannot drive the PLL clock input pins.

PLL Control Signals

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

pfdena

Use the pfdena signal to maintain the most recent locked frequency to allow your system to store its current settings before shutting down. The pfdena signal controls the phase frequency detector (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 relocks.

You must 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 must be maintained after a loss-of-lock condition.

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

locked

The locked signal 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.

Altera recommends using 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 PLLs support up to six different clock feedback modes. Each mode allows clock multiplication and division, phase shifting, and programmable duty cycle.

Table 5–14 lists the clock feedback modes supported by the Arria II PLLs.

Table 5–14. Clock Feedback Mode Availability for Arria II Devices

Clock Feedback Mode	Availability in Arria II GX Devices	Availability in Arria II GZ Devices	
		Top/Bottom PLLs	Left/Right PLLs
Source-synchronous mode	Yes	Yes	Yes
No-compensation mode	Yes	Yes	Yes
Normal mode	Yes	Yes	Yes
Zero-delay buffer (ZDB) mode *(1)*	Yes	Yes	Yes
External Feedback *(2)*	No	Yes	Yes *(3)*
LVDS compensation	Yes *(4)*	No	Yes

Notes to Table 5–14:

(1) ZDB mode uses 8 ns delay for compensation in Arria II GX devices.
(2) The high-bandwidth PLL setting is not supported in the external feedback mode.
(3) External feedback mode is supported for single-ended inputs and outputs only on the left and right PLLs.
(4) LVDS compensation mode is only supported on PLL_2, PLL_3, PLL_5, and PLL_6.

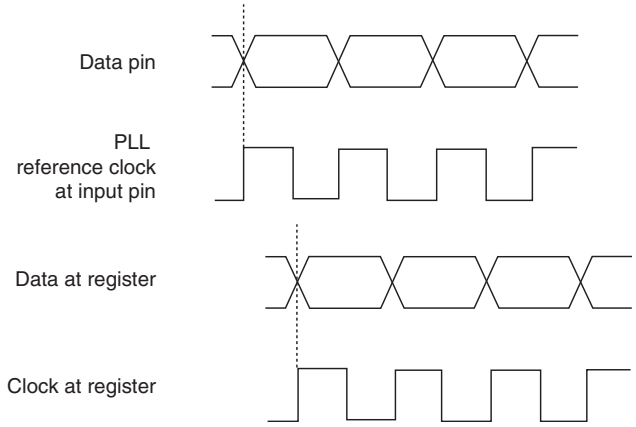


Input and output delays are fully compensated by a PLL only when you use the dedicated clock input pins associated with a given PLL as clock sources. For example, when you use PLL_1 (Arria II GX devices) or PLL_T1 (Arria II GZ devices) in normal mode, the clock delays from the input pin to the PLL clock 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 (Arria II GX devices) or PLL_T2 (Arria II GZ devices) 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), 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 5–24 shows an example waveform of the clock and data in source-synchronous 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 5–24. Phase Relationship Between Clock and Data in Source-Synchronous Mode in Arria II Devices



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 are 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. All input pins must be on the same side of the device for the clock delay to be properly compensated. The PLL compensates for the input pin with the longest pad-to-register delay among all input pins in the compensated bus.

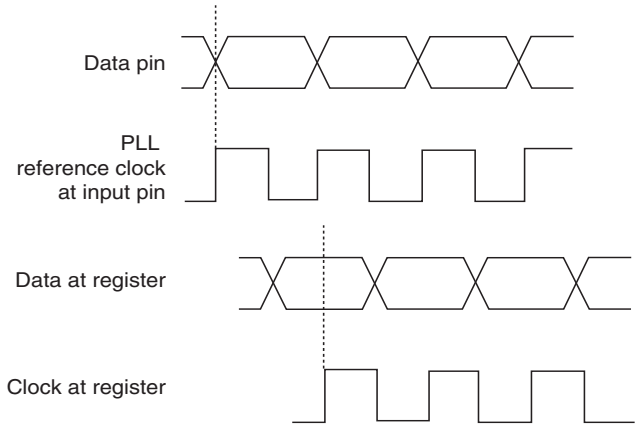
If you do not assign the **PLL Compensation** assignment, the Quartus II software automatically selects all pins driven by the compensated output of the PLL as the compensation target.

Source-Synchronous Mode for LVDS Compensation

The goal of source-synchronous mode for LVDS compensation is to maintain the same data and clock timing relationship seen at the pins at the internal serializer/deserializer (SERDES) capture register, except that the clock is inverted (180° phase shift), as shown in Figure 5–25. Thus, this mode ideally compensates for the delay of the LVDS clock network plus any difference in the delay between these two paths:

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

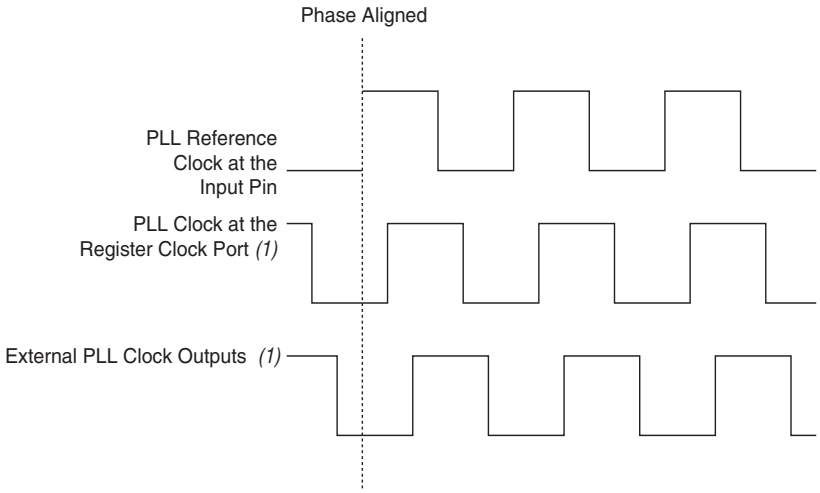
Figure 5–25. Source-Synchronous Mode for LVDS Compensation for Arria II Devices



No-Compensation Mode

In no-compensation mode, the PLL does not compensate for the 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 5–26 shows an example waveform of the PLL clocks' phase relationship in no-compensation mode.

Figure 5–26. Phase Relationship Between PLL Clocks in No-Compensation Mode for Arria II Devices



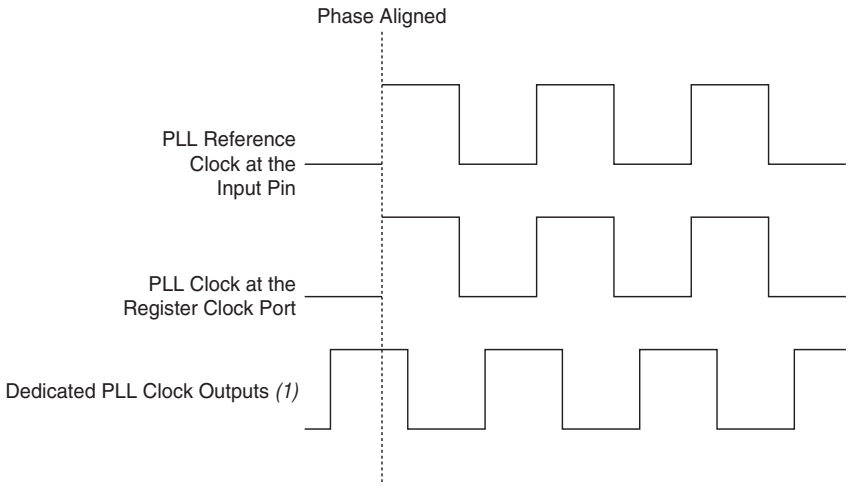
Note to Figure 5–26:

(1) The PLL clock outputs can 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 TimeQuest 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 5–27 shows an example waveform of the phase relationship of the PLL clocks in normal mode.

Figure 5–27. Phase Relationship Between PLL Clocks in Normal Mode for Arria II Devices



Note to Figure 5–27:

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

Zero-Delay Buffer Mode

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

You must instantiate a bidirectional I/O pin in the design to serve as the feedback path connecting the FBOUT and FBIN ports of the PLL when using Arria II GZ PLLs in ZDB mode, along with single-ended I/O standards, to ensure phase alignment between the CLK pin and the external clock output (CLKOUT) pin. The PLL uses this bidirectional I/O pin to mimic and compensate for the output delay from the clock output port of the PLL to the external clock output pin.

The bidirectional I/O pin that you instantiate in your design must always be assigned a single-ended I/O standard.

Do not place board traces on the bidirectional I/O pin when using ZDB mode, to avoid signal reflection.

Figure 5–28 shows ZDB mode in Arria II GZ PLLs. You cannot use differential I/O standards on the PLL clock input or output pins.

Figure 5–28. ZDB Mode in PLLs for Arria II GZ Devices

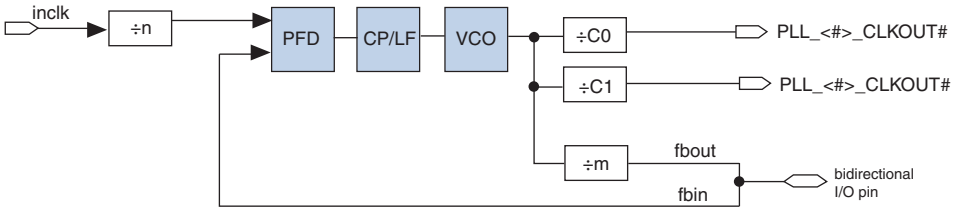
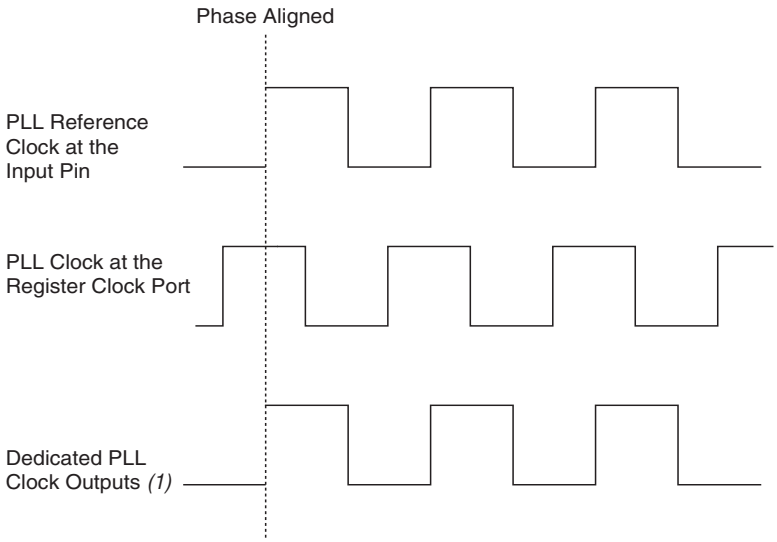


Figure 5–29 shows an example waveform of the PLL clocks' phase relationship in ZDB mode.

Figure 5–29. Phase Relationship Between PLL Clocks in Zero Delay Buffer Mode for Arria II Devices



Note to Figure 5–29:

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

External Feedback Mode

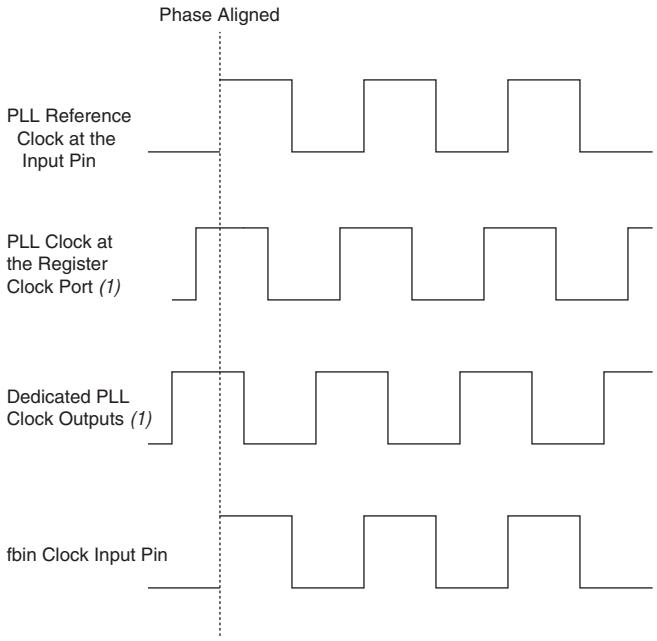
In external feedback mode, the external feedback input pin (`fbin`) is phase-aligned with the clock input pin, as shown in Figure 5–30. Aligning these clocks allows you to remove clock delay and skew between devices. This mode is supported on all Arria II GZ PLLs.

In external feedback mode, the output of the `M` counter (`FBOUT`) feeds back to the PLL `fbin` input (using a trace on the board) becoming part of the feedback loop. Also, use one of the dual-purpose external clock outputs as the `fbin` input pin in this mode.

You must use the same I/O standard on the input clock, feedback input, and output clocks. Left and right PLLs support this mode when using single-ended I/O standards only.

Figure 5–30 shows an example waveform of the phase relationship between the PLL clocks in external feedback mode.

Figure 5–30. Phase Relationship Between the PLL Clocks in External Feedback Mode for Arria II Devices

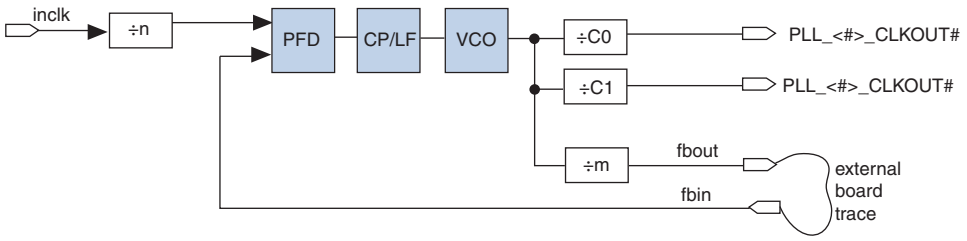


Note to Figure 5–30:

(1) The PLL clock outputs can lead or lag the `fbin` clock input.

Figure 5–31 shows external feedback mode implementation in Arria II GZ devices.

Figure 5–31. External Feedback Mode in Arria II GZ Devices



Clock Multiplication and Division

Each Arria II PLL provides clock synthesis for PLL output ports with $M/(N \times post\text{-}scale\ counter)$ 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, 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).

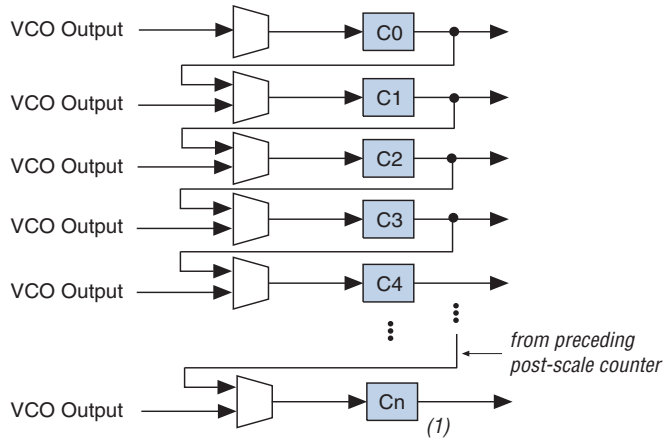
Each PLL has one pre-scale counter (N) and one multiply counter (M) with a range of 1 to 512 for both M and N. The n counter does not use duty-cycle control because the only purpose of this counter is to calculate frequency division. There are seven generic post-scale counters in each PLL that can feed 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 ranges 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

Arria II 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 5–32.

Figure 5–32. Counter Cascading for Arria II Devices



Note to Figure 5–32:

(1) For Arria II GX devices, n = 6. For Arria II GZ devices, n = 6 or 9.

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, the cascaded value is C0 × C1 = 800.

☞ Post-scale counter cascading is set in the configuration file. You cannot accomplish post-scale counter cascading with PLL reconfiguration.

Programmable Duty Cycle

The programmable duty cycle allows the 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, 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.

For Arria II GZ devices, if the PLL is in external feedback mode, set the duty cycle for the counter driving the `fbin` pin to 50%.

Programmable Phase Shift

Use phase shift to implement a robust solution for clock delays in Arria II devices. Implement phase shift with a combination of the VCO phase output and the counter starting time. A combination of 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 (PVT).

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

- Fine resolution with VCO phase taps
- Coarse resolution with 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 with this method is defined in Equation 5–1.

Equation 5–1. Fine-Resolution Phase Shifts for Arria II Devices

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

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. The PLL operating frequency, which is governed by the reference clock frequency and the counter settings, defines this phase shift.

Equation 5–2 shows the coarse-resolution phase shifts are implemented by delaying the start of the counters for a predetermined number of counter clocks.

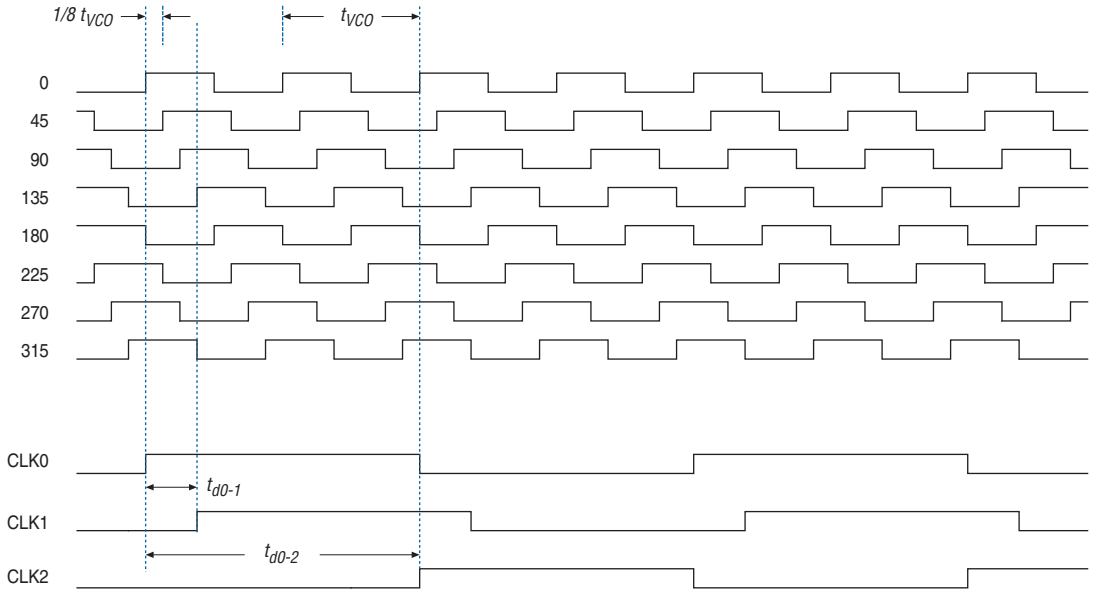
Equation 5–2. Coarse-Resolution Phase Shifts for Arria II Devices

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

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 5–33 shows an example of a phase-shift insertion with the fine resolution with 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 135x phase tap from the VCO and also has the C value for the counter set to one. The CLK1 signal is also divided by four. 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 5–33. Delay Insertion with VCO Phase Output and Counter Delay Time for Arria II Devices



Use the coarse- and fine-phase shifts to implement clock delays in Arria II devices. The ALTPLL megafunction allows you to enter the desired VCO phase taps and initial counter value settings through the MegaWizard™ Plug-In Manager in the Quartus II software.

Arria II 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 ability of the PLL to track the input clock and its associated jitter. Arria II PLLs provide advanced control of the PLL bandwidth with 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 devices can accept a spread-spectrum input with typical modulation frequencies. However, the device cannot automatically detect that the input is a spread-spectrum signal. Instead, the input signal looks like deterministic jitter at the input of the PLL. Arria II PLLs can track a spread-spectrum input clock as long as the input jitter is in the PLL input jitter tolerance specification. Arria II devices cannot internally generate spread-spectrum clocks.

Clock Switchover

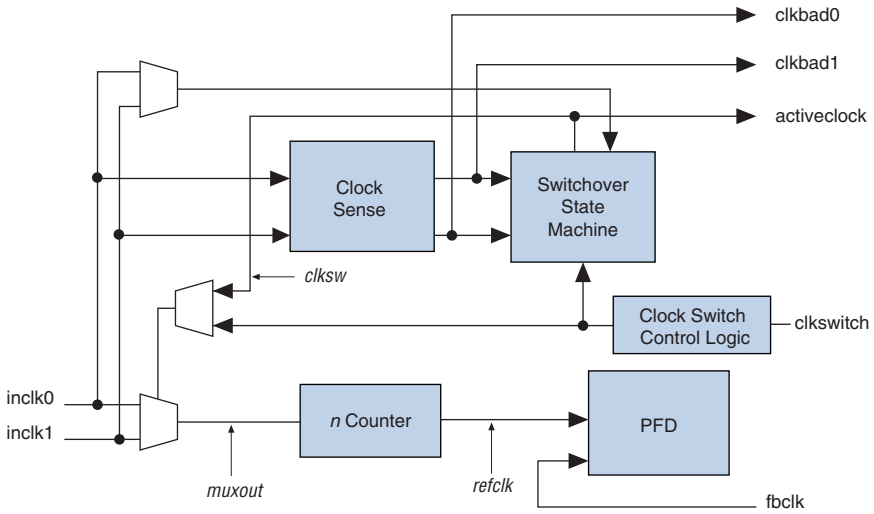
The clock switchover feature allows the PLL to switch between two reference input clocks. Use this feature for clock redundancy or for a dual-clock domain application such as in a system that turns on the redundant clock if the previous clock stops running. Your design can perform clock switchover automatically, 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 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 with 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 `clkswitch` = 1, it overrides automatic clock switchover function. As long as the `clkswitch` signal is high, further switchover action is blocked.

Arria II PLLs support a fully configurable clock switchover capability. Figure 5–34 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 5–34. Automatic Clock Switchover Circuit Block Diagram for Arria II Devices



Automatic Clock Switchover Mode

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 5–34. In this case, `inclk1` becomes the reference clock for the PLL. When you use 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 you use automatic clock switchover mode, the following requirements must be satisfied:

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

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



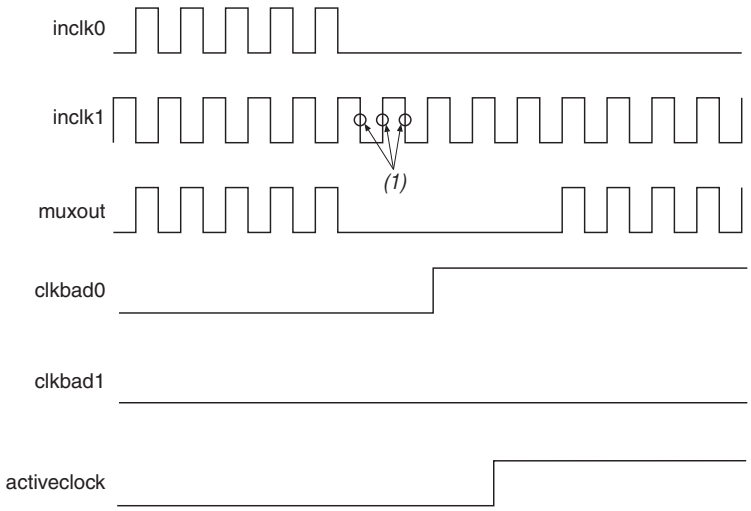
Altera recommends resetting the PLL with the `areset` signal to maintain the phase relationships between the PLL input and output clocks when you use clock switchover.

When you use 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 5–35 shows an example waveform of the switchover feature with 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 5–35. Automatic Switchover Upon Loss of Clock Detection for Arria II Devices



Note to Figure 5–35:

(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.

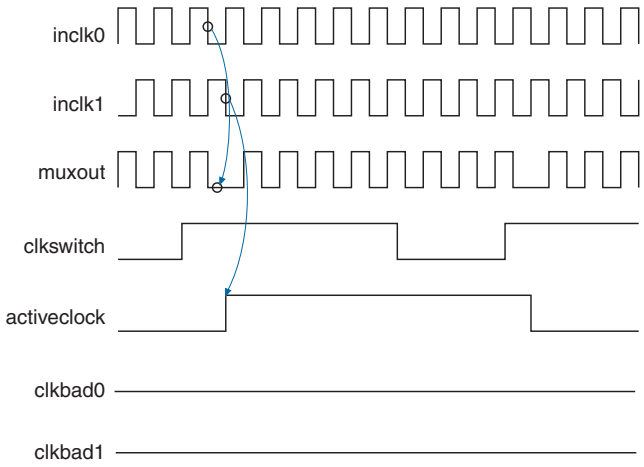
Manual Override Mode

In 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 when you use clkswitch because the automatic clock-sense circuitry cannot monitor clock input (inclk0 and inclk1) frequencies with a frequency difference of more than 100% (2x). 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 must 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 1,600 MHz. The ALTPLL MegaWizard Plug-In Manager interface notifies you if a given combination of inclk0 and inclk1 frequencies cannot meet this requirement.

Figure 5–36 shows an example waveform of the switchover feature when controlled by the clkswitch signal. 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 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 5–36. Clock Switchover with the clkswitch (Manual) Control for Arria II Devices ***(Note 1)***



Note to Figure 5–36:

(1) To start a manual clock switchover event, both inclk0 and inclk1 must be running when the clkswitch signal goes high.

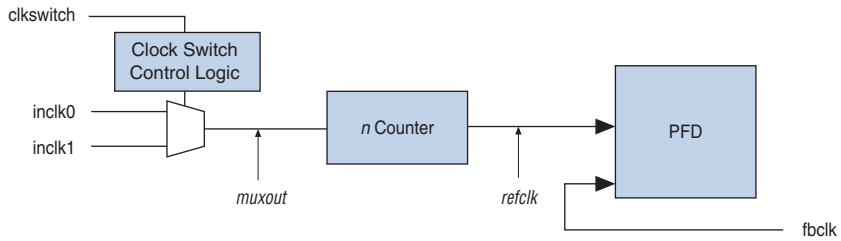
In automatic switchover with manual override mode, the activeclock signal mirrors the clkswitch signal. As both clocks are still functional during the manual switch, neither clkbad signal goes high. Because the switchover circuit is positive-edge sensitive, the falling edge of the clkswitch signal does 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 Mode

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 being held high for at least three inclk cycles begins a clock switchover event. You must bring the clkswitch signal 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 and inclk1 cycles.

Figure 5–37 shows a block diagram of the manual switchover circuit.

Figure 5–37. Manual Clock Switchover Circuitry in PLLs for Arria II Devices



For more information about PLL software support in the Quartus II software, refer to the *Phase-Locked Loops (ALTPLL) Megafunction User Guide*.

Clock Switchover Guidelines

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

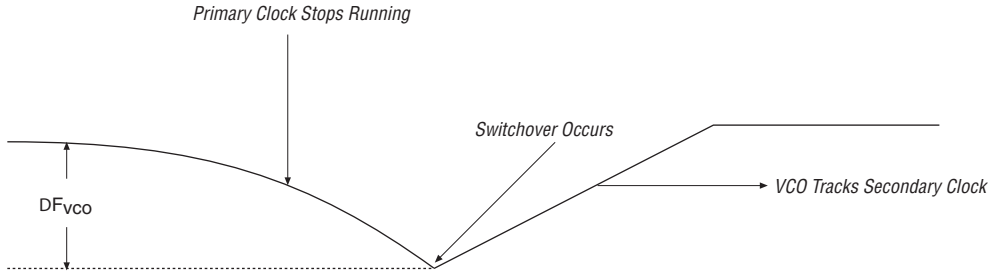
- Automatic clock switchover requires that the inclk0 and inclk1 frequencies be in 100% (2x) of each other. Failing to meet this requirement causes the clkbad[0] and clkbad[1] signals to not function properly.
- When you use manual clock switchover mode, the difference between inclk0 and inclk1 can be more than 100% (2x). However, differences in frequency, or phase of the two clock sources, or both, are likely to 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 must use a low-bandwidth PLL. The low-bandwidth PLL reacts more slowly than the high-bandwidth PLL to reference the input clock changes. When the switchover event occurs, a low-bandwidth PLL propagates the stopping of the clock to the output more slowly than the high-bandwidth PLL. However, be aware that the low-bandwidth PLL also increases lock time.

- After a switchover event 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 relock 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 become stable before re-enabling the output clocks from the PLL at the clock control block.
- Figure 5–38 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 5–38. VCO Switchover Operating Frequency for Arria II Devices



- 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, the output clock-enable signals (`clkena`) can disable the clock outputs during the switchover and resynchronization period. After the lock indication is stable, the system can re-enable the output clocks.

PLL Reconfiguration

PLLs use several divide counters and different VCO phase taps to perform frequency synthesis and phase shifts. In Arria II 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 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 the clock-to-out (t_{CO}) delays in real time by changing the PLL output clock phase shift. This approach eliminates the requirement to regenerate a configuration file with the new PLL settings.

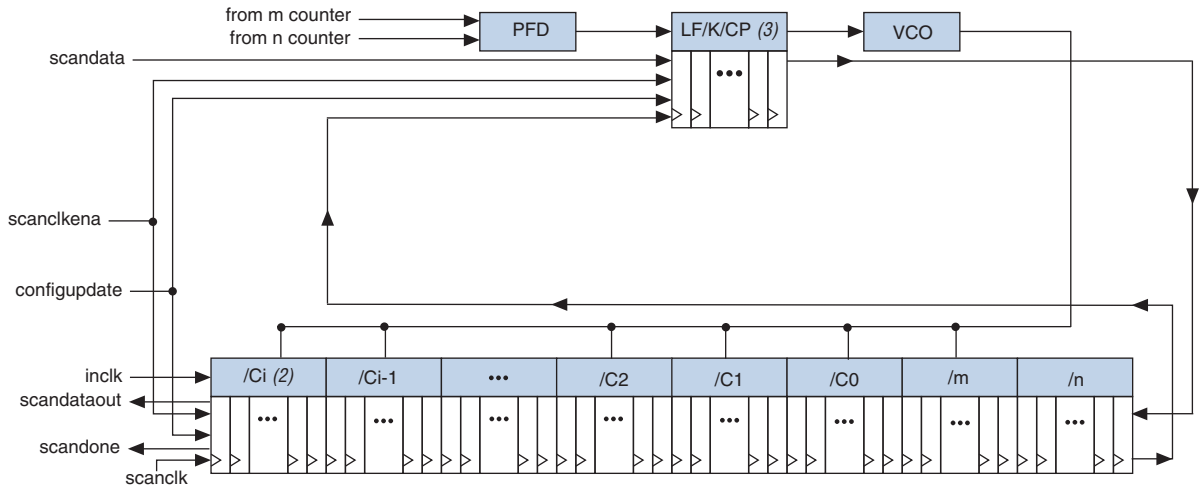
PLL Reconfiguration Hardware Implementation

The following PLL components are reconfigurable in real time:

- Pre-scale counter (N)
- Feedback counter (M)
- Post-scale output counters (C0 to C6 for Arria II GX devices and C0 to C9 for Arria II GZ devices)
- Post VCO divider (K)
- Dynamically adjust the charge pump current (Icp) and loop filter components (R and C) to facilitate reconfiguration of the PLL bandwidth

Figure 5–39 shows how you can dynamically adjust the PLL counter settings by shifting their new settings into a serial shift-register chain or scan chain. Serial data is the input to the scan chain with 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.

Figure 5–39. PLL Reconfiguration Scan Chain for Arria II Devices ***(Note 1)***



Notes to Figure 5–39:

(1) The Arria II GX PLLs and Arria II GZ left and right PLLs support C0 to C6 counters.

(2) For Arria II GX devices, i = 6. For Arria II GZ devices, i = 6 or 9.

(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.

For more information about the PLL reconfiguration port signals, refer to the *Phase Locked-Loops Reconfiguration (ALTPLL_RECONFIG) Megafunction User Guide*.

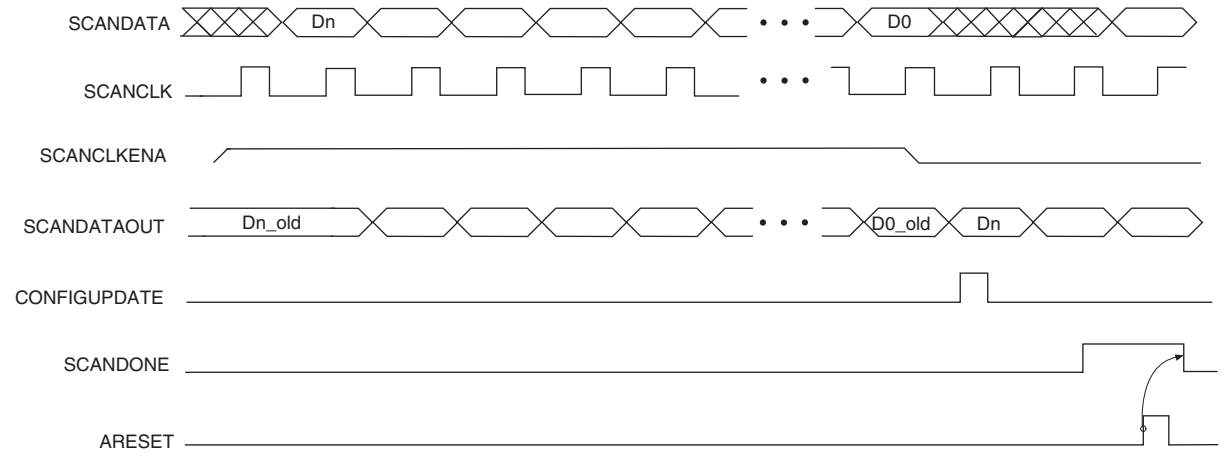
The counter settings are updated synchronously to the clock frequency of the individual counters. Therefore, all counters are not simultaneously updated.

To reconfigure the PLL counters, follow these steps:

1. Assert the SCANCLKENA signal at least one SCANCLK cycle prior to shifting in the first bit of SCANDATA (Dn for Arria II GX devices or D0 for Arria II GZ devices).
2. Serial data (SCANDATA) is shifted into the scan chain on the second rising edge of SCANCLK.
3. For Arria II GX devices, after all 180 bits are scanned into the scan chain, the SCANCLKENA signal is deasserted to prevent inadvertent shifting of bits in the scan chain. For Arria II GZ devices, after all 234 bits (top and bottom PLLs) or 180 bits (left and right PLLs) have been scanned into the scan chain, the SCANCLKENA signal is deasserted 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 with the ARESET signal if you make any changes to the M, N, or post-scale output C counters or the Icp, R, or C settings.
7. Repeat steps 1 through 5 to reconfigure the PLL any number of times.

Figure 5–40 shows a functional simulation of the PLL reconfiguration feature.

Figure 5–40. PLL Reconfiguration Waveform for Arria II Devices





When you reconfigure the counter clock frequency, you cannot reconfigure the corresponding counter phase shift settings with the same interface. Instead, reconfigure the phase shifts in real time with the dynamic phase shift reconfiguration interface. If you reconfigure the counter frequency, but want 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 C9)

You can configure the multiply or divide values and duty cycle of post-scale counters 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 require 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 5–15 lists the number of bits for each component of an Arria II GX PLL.

Table 5–15. PLL Reprogramming Bits for Arria II GX Devices

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 5–15:

(1) Includes two control bits: `rbypass` for bypassing the counter and `rselodd` to select the output clock duty cycle.
(2) The LSB for `C6` low-count value is the first bit shifted into the scan chain.
(3) The MSB for loop filter is the last bit shifted into the scan chain.

The length of the scan chain varies for different Arria II GZ PLLs. The top and bottom PLLs have ten post-scale counters and a 234-bit scan chain, while the left and right PLLs have seven post-scale counters and a 180-bit scan chain.

Table 5–16 lists the number of bits for each component of a Arria II GZ PLL. Table 5–16 also lists the scan chain order of PLL components for the top and bottom PLLs, which have 10 post-scale counters. The order of bits is the same for the left and right PLLs, but the reconfiguration bits start with the C6 post-scale counter.

Table 5–16. Top and Bottom PLL Reprogramming Bits for Arria II GZ Devices

Block Name	Number of Bits		Total
	Counter	Other *(1)*	
C9 *(2)*	16	2	18
C8	16	2	18
C7	16	2	18
C6 *(3)*	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 *(4)*	0	2	2
Loop Filter Resistor	0	5	5
Unused CP/LF	0	7	7
Total number of bits	—	—	234

Notes to Table 5–16:

(1) Includes two control bits, `rbypass` for bypassing the counter, and `rselodd` to select the output clock duty cycle.
(2) The LSB for the C9 low-count value is the first bit shifted into the scan chain for the top and bottom PLLs.
(3) The LSB for the C6 low-count value is the first bit shifted into the scan chain for the left and right PLLs.
(4) The MSB for the loop filter is the last bit shifted into the scan chain.

Figure 5–41 shows the scan chain order of Arria II GX PLL components which have seven post-scale counters. The reconfiguration bits start with the C6 post-scale counter.

Figure 5–41. Scan Chain Order of PLL Components for Arria II GX PLLs

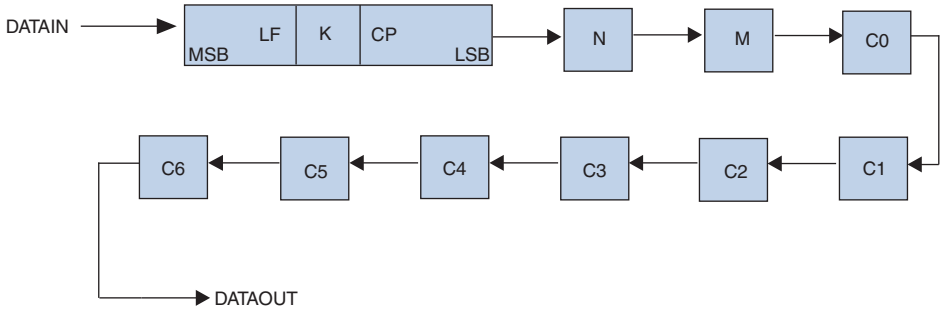
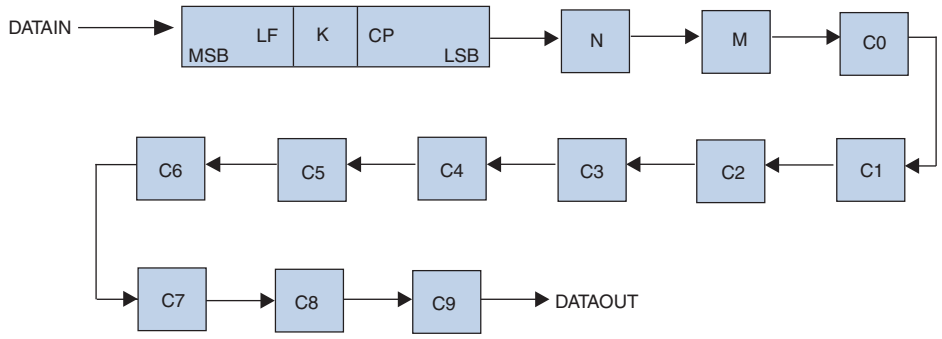


Figure 5–42 shows the scan chain order of PLL components for the top and bottom Arria II GZ PLLs.

Figure 5–42. Scan Chain Order of PLL Components for Top and Bottom of Arria II GZ PLLs *(Note 1)*

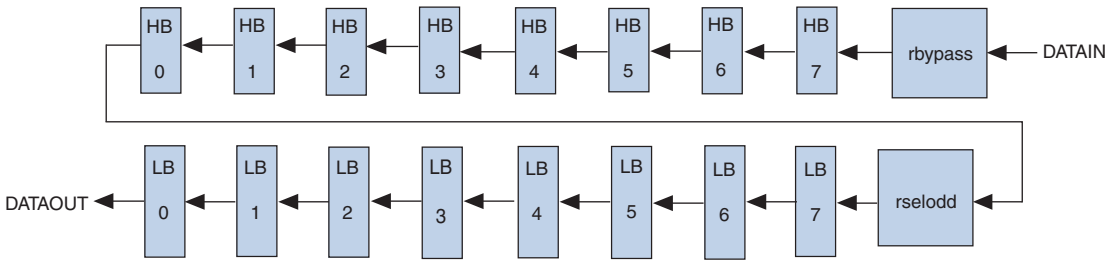


Note to Figure 5–43:

(1) The left and right PLLs have the same scan chain order. The post-scale counters end at C6.

Figure 5–43 shows the scan chain bit-order sequence for post-scale counters in all Arria II PLLs.

Figure 5–43. Scan Chain Bit-Order Sequence for Post-Scale Counters in Arria II 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 5–17 through Table 5–19 show the possible settings for charge pump current (Icp), loop filter resistor (R), and capacitor (C) values for Arria II PLLs.

Table 5–17. charge_pump_current Bit Settings for Arria II Devices

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 5–18. loop_filter_r Bit Settings for Arria II Devices

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
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 5–19. loop_filter_c Bit Settings for Arria II Devices

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 C9 counters) factor of one.

Table 5–20 lists the settings for bypassing the counters in Arria II PLLs.

Table 5–20. PLL Counter Settings for Arria II Devices

PLL Scan Chain Bits [0..8] Settings									
LSB								**MSB**	**Description**
0 *(1)*, X *(2)*	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 5–20:

(1) For Arria II GX devices.
(2) For Arria II GZ devices
(3) Counter-bypass bit.

For more information about how to use the PLL scan chain bit settings, refer to the *Phase Locked-Loops Reconfiguration (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 having 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 5–21 lists the control signals that are used for dynamic phase-shifting.

Table 5–21. Dynamic Phase-Shifting Control Signals for Arria II Devices (Part 1 of 2)

Signal Name	Description	Source	Destination
PHASECOUNTERSELECT[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

Table 5–21. Dynamic Phase-Shifting Control Signals for Arria II Devices (Part 2 of 2)

Signal Name	Description	Source	Destination
SCANCLK	Free running clock from core used in combination with PHASESTEP to enable, disable, or both dynamic phase shifting. Shared with scanclk for dynamic reconfiguration.	GCLK, RCLK, or I/O pin	PLL reconfiguration circuit
PHASEDONE	When asserted, this indicates to the core logic that the phase adjustment is complete and the PLL is ready to act on a possible second adjustment pulse. Asserts based on internal PLL timing. Deasserts on the rising edge of scanclk.	PLL reconfiguration circuit	Logic array or I/O pins

Table 5–22 lists the PLL counter selection based on the corresponding PHASECOUNTERSELECT setting.

Table 5–22. Phase Counter Select Mapping for Arria II Devices *(Note 1)*

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
1	0	0	1	C7 Counter
1	0	1	0	C8 Counter
1	0	1	1	C9 Counter

Note to Table 5–22:

(1) C7 to C9 counter are only available for Arria II GZ devices.

To perform one dynamic phase-shift, follow these steps:

1. Set PHASEUPDOWN and PHASECOUNTERSELECT as required.
2. Assert PHASESTEP for at least two SCANCLK cycles. Each PHASESTEP pulse allows one phase shift.
3. Deassert PHASESTEP after PHASEDONE goes low.
4. Wait for PHASEDONE to go high.
5. Repeat steps 1 through 4 as many times as required to perform multiple phase-shifts.

PHASEUPDOWN and PHASECOUNTERSELECT signals are synchronous to SCANCLK and must meet the t_{su} and t_h requirements with respect to the SCANCLK edges.

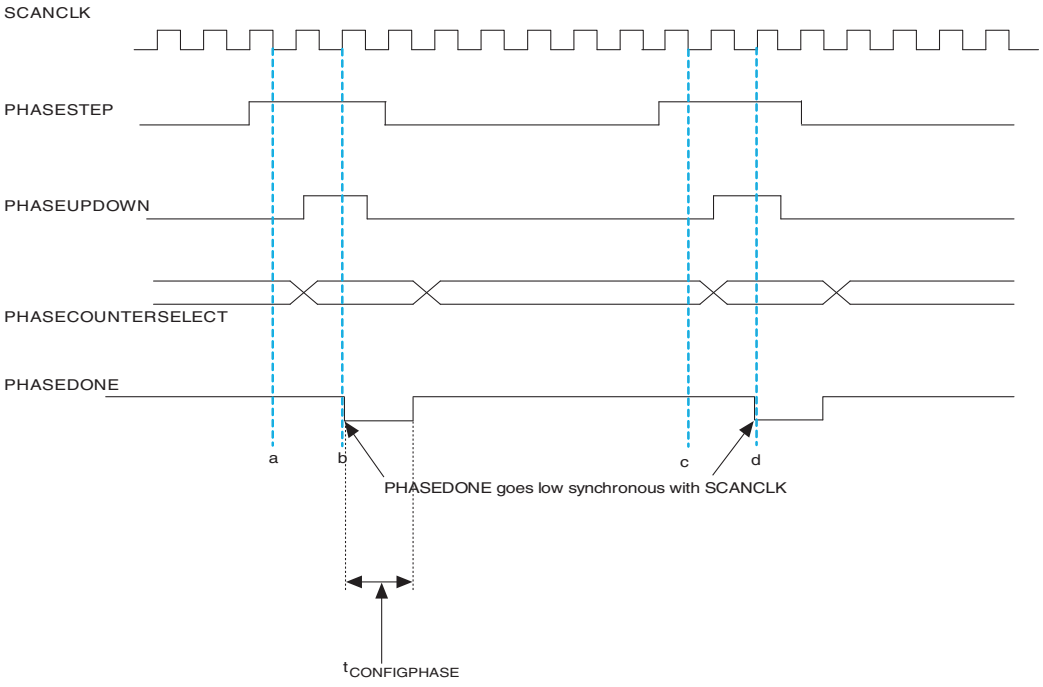
You can repeat dynamic phase-shifting indefinitely. For example, in a design where the VCO frequency is set to 1,000 MHz and the output clock frequency is set to 100 MHz, performing 40 dynamic phase shifts (each one yields 125 ps phase shift) results in shifting the output clock by 180°, in other words, a phase shift of 5 ns.

The PHASESTEP signal is latched on the negative edge of SCANCLK (a,c) and must remain asserted for at least two SCANCLK cycles. De-assert PHASESTEP after PHASEDONE goes low. On the second SCANCLK rising edge (b,d) after PHASESTEP is latched, the values of PHASEUPDOWN and PHASECOUNTERSELECT are latched and the PLL starts dynamic phase-shifting for the specified counters and in the indicated direction. PHASEDONE is de-asserted synchronous to SCANCLK at the second rising edge (b,d) and remains low until the PLL finishes dynamic phase-shifting. Depending on the VCO and SCANCLK frequencies, PHASEDONE low time may be greater than or less than one SCANCLK cycle.

You can perform another dynamic phase-shift after the PHASEDONE signal goes from low to high. Each PHASESTEP pulse enables one phase shift. PHASESTEP pulses must be at least one SCANCLK cycle apart.

Figure 5–44 shows the dynamic phase shifting waveform.

Figure 5–44. Dynamic Phase Shifting Waveform for Arria II Devices



For more information about the ALTPLL_RECONFIG MegaWizard Plug-In Manager interface, refer to the *Phase Locked-Loops Reconfiguration (ALTPLL_RECONFIG) Megafunction User Guide*.

PLL Specifications

For more information about PLL timing specifications, refer to the *Device Datasheet for Arria II Devices*.

Document Revision History

Table 5–23 lists the revision history for this chapter.

Table 5–23. Document Revision History

Date	Version	Changes
July 2012	4.2	Updated "Periphery Clock Networks" section.
June 2011	4.1	■ Updated Table 5–15. ■ Updated Figure 5–44. ■ Updated "Dynamic Phase-Shifting" section. ■ Added Figure 5–5, Figure 5–6, Figure 5–7, and Figure 5–8. ■ Minor text edits.
December 2010	4.0	■ Updated for the Quartus II software version 10.1 release. ■ Added Arria II GZ devices information. ■ Updated Table 5–1, Table 5–12, Table 5–20, and Table 5–21. ■ Added Figure 5–2, Figure 5–3, Figure 5–4, Figure 5–5, Figure 5–7, Figure 5–15, Figure 5–11, Figure 5–16, Figure 5–18, Figure 5–19, Figure 5–24, Figure 5–26, Figure 5–27, Figure 5–38, and Figure 5–39. ■ Added Table 5–5, Table 5–7, Table 5–9, Table 5–11, andTable 5–16. ■ Added "Clock Sources Per Quadrant" and "External Feedback Mode" sections. ■ Minor text edit.
July 2010	3.0	Updated for Arria II GX v10.0 release: ■ Updated "Clock Regions" and "Arria II PLL Hardware Overview" sections. ■ Updated Figure 5–44. ■ Removed sub-regional clock references. ■ Minor text edit.
November 2009	2.0	Updated for Arria II GX v9.1 release: ■ Updated Table 5–1. ■ Updated Figure 5–14. ■ Updated the "Periphery Clock (PCLK) Networks" and "Cascading PLLs" sections. ■ Minor text edit.
June 2009	1.1	■ Updated Table 5–8. ■ Updated Figure 5–13 and Figure 5–14. ■ Updated the "PLL Clock I/O Pins" and "PLL Reconfiguration Hardware Implementation" sections.
February 2009	1.0	Initial release

Section II. I/O Interfaces for Arria II Devices

This section provides information on Arria® II 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 Devices
- Chapter 7, External Memory Interfaces in Arria II Devices
- Chapter 8, High-Speed Differential I/O Interfaces and DPA in Arria II 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 Devices

AIIGX51006-4.2

This chapter describes how Arria® II devices provide I/O capabilities that allow you to work in compliance with current and emerging I/O standards and requirements. With these device features, you can reduce board design interface costs and increase development flexibility.

Package and die enhancements with dynamic termination and output control provide best-in-class signal integrity. Numerous I/O features assist high-speed data transfer into and out of the device, including:

- 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
- Bus LVDS (BLVDS) for Arria II GX devices
- Programmable output current strength
- Programmable slew rate
- Programmable bus-hold
- Programmable pull-up resistor
- Open-drain output
- On-chip series termination (R_S OCT)
- On-chip differential termination (R_D OCT)
- On-chip parallel termination (R_T OCT) for Arria II GZ devices
- Dynamic OCT for Arria II GZ devices
- Programmable pre-emphasis
- Programmable voltage output differential (V_{OD})

This chapter includes the following sections:

- "I/O Standards Support" on page 6–2
- "I/O Banks" on page 6–5
- "I/O Structure" on page 6–10
- "OCT Support" on page 6–19
- "Arria II OCT Calibration" on page 6–26
- "Termination Schemes for I/O Standards" on page 6–28
- "I/O Bank Restrictions" on page 6–36

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

I/O Standards Support

Table 6–1 lists 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. I/O Standards and Voltage Levels for Arria II GX Devices

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, II	JESD8-9B	*(2)*	2.5	2.5	1.25	1.25
SSTL-18 Class I, 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, II	JESD8-6	*(2)*	1.8	2.5	0.90	0.90
HSTL-15 Class I, II	JESD8-6	*(2)*	1.5	2.5	0.75	0.75
HSTL-12 Class I, 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)*	—	2.5	—	—
BLVDS	—	*(2)*	2.5	2.5	—	—

Notes to Table 6–1:

(1) PCI-X does not meet the PCI-X I-V curve requirement at the linear region.

(2) Single-ended SSTL/HSTL, differential SSTL/HSTL, LVDS, LVPECL, and BLVDS input buffers are powered by V_{CCPD}.

(3) Differential SSTL/HSTL inputs use LVDS differential input buffers without R_D OCT support.

Table 6–2 lists the supported I/O standards for Arria II GZ devices and the typical values for input and output V_{CCIO}, V_{CCPD}, V_{REF}, and board V_{TT}.

Table 6–2. I/O Standards and Voltage Levels for Arria II GZ Devices *(Note 1)* (Part 1 of 2)

I/O Standard	Standard Support	V_{CCIO} (V)				V_{CCPD} (V) (Pre-Driver Voltage)	V_{REF} (V) (Input Ref Voltage)	V_{TT} (V) (Board Termination Voltage)
		Input Operation		Output Operation				
		Column I/O Banks	Row I/O Banks	Column I/O Banks	Row I/O Banks			
3.3-V LVTTL	JESD8-B	3.0/2.5	3.0/2.5	3.0	3.0	3.0	—	—
3.3-V LVCMOS *(3)*	JESD8-B	3.0/2.5	3.0/2.5	3.0	3.0	3.0	—	—
2.5-V LVCMOS	JESD8-5	3.0/2.5	3.0/2.5	2.5	2.5	2.5	—	—
1.8-V LVCMOS	JESD8-7	1.8/1.5	1.8/1.5	1.8	1.8	2.5	—	—
1.5-V LVCMOS	JESD8-11	1.8/1.5	1.8/1.5	1.5	1.5	2.5	—	—
1.2-V LVCMOS	JESD8-12	1.2	1.2	1.2	1.2	2.5	—	—
3.0-V PCI	PCI Rev 2.1	3.0	3.0	3.0	3.0	3.0	—	—
3.0-V PCI-X	PCI-X Rev 1.0	3.0	3.0	3.0	3.0	3.0	—	—
SSTL-2 Class I, II	JESD8-9B	*(2)*	*(2)*	2.5	2.5	2.5	1.25	1.25
SSTL-18 Class I, II	JESD8-15	*(2)*	*(2)*	1.8	1.8	2.5	0.90	0.90
SSTL-15 Class I	—	*(2)*	*(2)*	1.5	1.5	2.5	0.75	0.75
SSTL-15 Class II	—	*(2)*	*(2)*	1.5	—	2.5	0.75	0.75
HSTL-18 Class I, II	JESD8-6	*(2)*	*(2)*	1.8	1.8	2.5	0.90	0.90
HSTL-15 Class I	JESD8-6	*(2)*	*(2)*	1.5	1.5	2.5	0.75	0.75
HSTL-15 Class II	JESD8-6	*(2)*	*(2)*	1.5	—	2.5	0.75	0.75
HSTL-12 Class I	JESD8-16A	*(2)*	*(2)*	1.2	1.2	2.5	0.6	0.6
HSTL-12 Class II	JESD8-16A	*(2)*	*(2)*	1.2	—	2.5	0.6	0.6
Differential SSTL-2 Class I, II	JESD8-9B	*(2)*	*(2)*	2.5	2.5	2.5	—	1.25
Differential SSTL-18 Class I, II	JESD8-15	*(2)*	*(2)*	1.8	1.8	2.5	—	0.90
Differential SSTL-15 Class I	—	*(2)*	*(2)*	1.5	1.5	2.5	—	0.75
Differential SSTL-15 Class II	—	*(2)*	*(2)*	1.5	—	2.5	—	0.75
Differential HSTL-18 Class I, II	JESD8-6	*(2)*	*(2)*	1.8	1.8	2.5	—	0.90
Differential HSTL-15 Class I	JESD8-6	*(2)*	*(2)*	1.5	1.5	2.5	—	0.75
Differential HSTL-15 Class II	JESD8-6	*(2)*	*(2)*	1.5	—	2.5	—	0.75
Differential HSTL-12 Class I	JESD8-16A	*(2)*	*(2)*	1.2	1.2	2.5	—	0.60
Differential HSTL-12 Class II	JESD8-16A	*(2)*	*(2)*	1.2	—	2.5	—	0.60

Table 6–2. I/O Standards and Voltage Levels for Arria II GZ Devices *(Note 1)* (Part 2 of 2)

I/O Standard	Standard Support	V_{CCIO} (V)				V_{CCPD} (V) (Pre-Driver Voltage)	V_{REF} (V) (Input Ref Voltage)	V_{TT} (V) (Board Termination Voltage)
		Input Operation		Output Operation				
		Column I/O Banks	Row I/O Banks	Column I/O Banks	Row I/O Banks			
LVDS *(4)*, *(5)*, *(8)*	ANSI/TIA/EIA-644	*(2)*	*(2)*	2.5	2.5	2.5	—	—
RSDS *(6)*, *(7)*, *(8)*	—	*(2)*	*(2)*	2.5	2.5	2.5	—	—
mini-LVDS *(6)*, *(7)*, *(8)*	—	*(2)*	*(2)*	2.5	2.5	2.5	—	—
LVPECL	—	*(4)*	2.5	—	—	2.5	—	—

Notes to Table 6–2:

(1) V_{CCPD} is either 2.5 or 3.0 V. For V_{CCIO} = 3.0 V, V_{CCPD} = 3.0 V. For V_{CCIO} = 2.5 V or less, V_{CCPD} = 2.5 V.

(2) Single-ended HSTL/SSTL, differential SSTL/HSTL, and LVDS input buffers are powered by V_{CCPD}. Row I/O banks support both true differential input buffers and true differential output buffers. Column I/O banks support true differential input buffers, but not true differential output buffers. I/O pins are organized in pairs to support differential standards. Column I/O differential HSTL and SSTL inputs use LVDS differential input buffers without R_D OCT support.

(3) For more information about the 3.3-V LVTTL/LVCMOS standard supported in Arria II devices, refer to "3.3-V I/O Interface" on page 6–13.

(4) Column I/O banks support LVPECL I/O standards for input clock operation. Clock inputs on column I/Os are powered by $V_{CCCLKIN}$ when configured as differential clock inputs. They are powered by V_{CCIO} when configured as single-ended clock inputs. Differential clock inputs in row I/Os are powered by V_{CCPD}.

(5) Column and row I/O banks support LVDS outputs using two single-ended output buffers, an external one-resistor (LVDS_E_1R), and a three-resistor (LVDS_E_3R) network.

(6) Row I/O banks support RSDS and mini-LVDS I/O standards using a true LVDS output buffer without a resistor network.

(7) Column and row I/O banks support RSDS and mini-LVDS I/O standards using two single-ended output buffers with one-resistor (RSDS_E_1R and mini-LVDS_E_1R) and three-resistor (RSDS_E_3R and mini-LVDS_E_3R) networks.

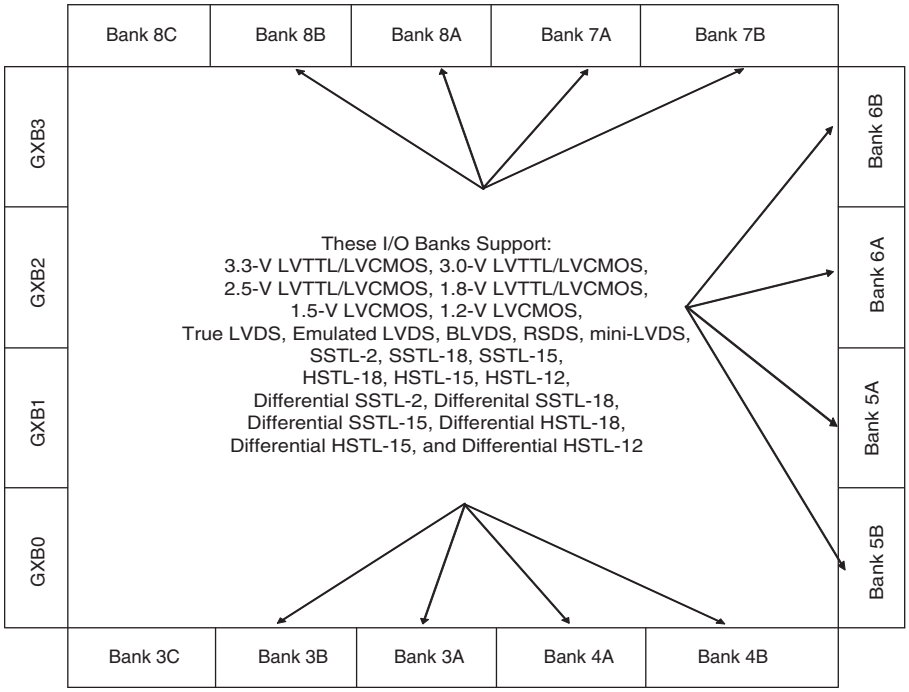
(8) The emulated differential output standard that supports the tri-state feature includes: LVDS_E_1R, LVDS_E_3R, RSDS_E_1R, RSDS_E_3R, Mini_LVDS_E_1R, and Mini_LVDS_E_3R. For more information, refer to the *I/O Buffer (ALTIOBUF) Megafunction User Guide*.

For detailed electrical characteristics of each I/O standard, refer to the *Device Datasheet for Arria II Devices*.

I/O Banks

Arria II GX devices contain up to 16 I/O banks as shown in Figure 6–1. The left I/O banks are dedicated for high-speed transceivers. Bank 3C and 8C are dedicated for configuration pins. The rest of the banks are user I/O banks that support all single-ended and differential I/O standards.

Figure 6–1. I/O Banks in Arria II GX Devices ***(Note 1)***, ***(2)***, ***(3)***, ***(4)***, ***(5)***, ***(6)***, ***(7)***



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 R_D OCT support.

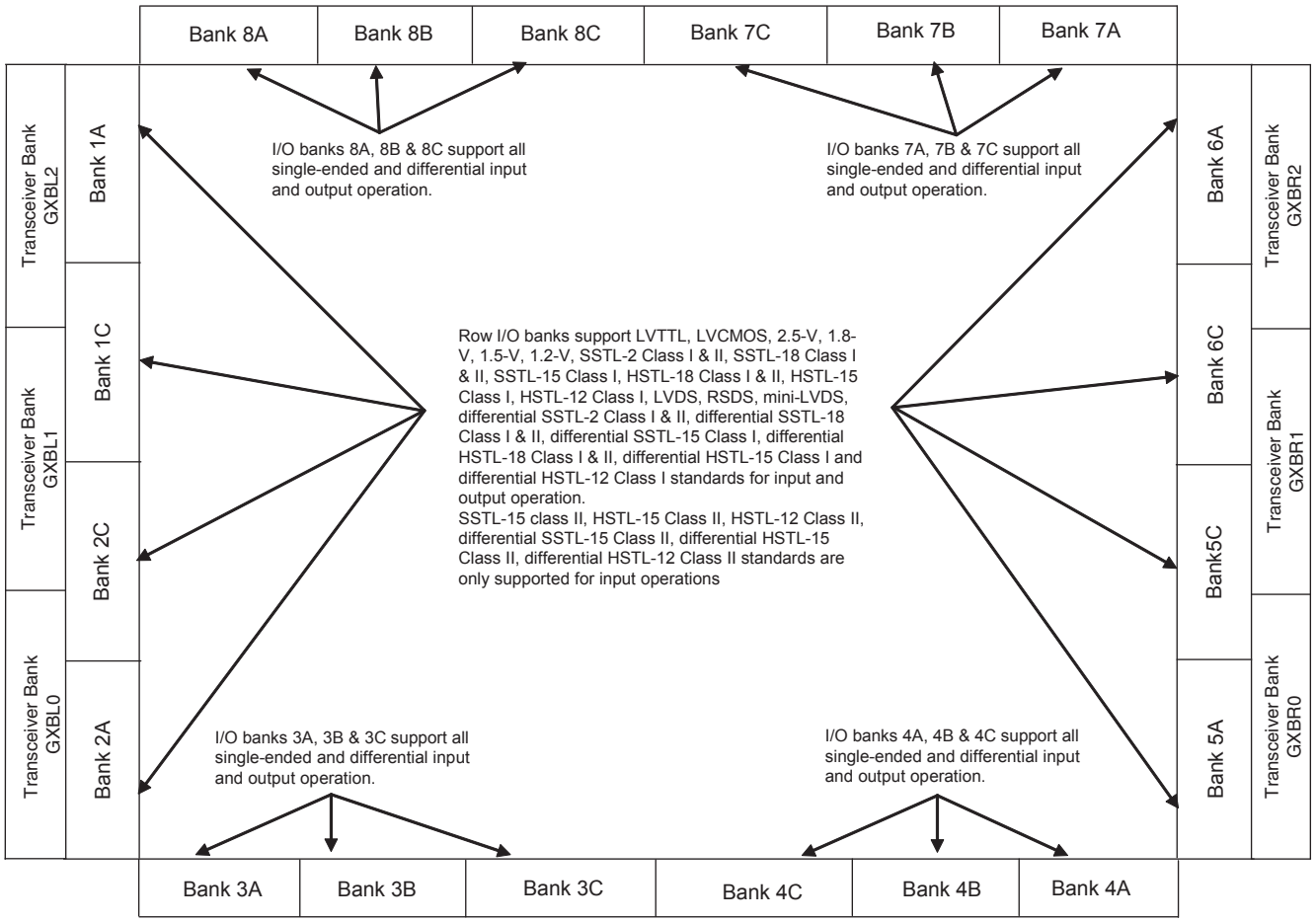
(5) Differential HSTL and SSTL outputs are not true differential outputs. They use two single-ended outputs with the second output programmed as inverted.

(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.

(7) The `PLL_CLKOUT` pin supports only emulated differential I/O standard but not true differential I/O standard.

Arria II GZ devices contain up to 20 I/O banks as shown in Figure 6–2. Each I/O bank can support high-performance external memory interfaces with dedicated circuitry. The I/O pins are organized in pairs to support differential standards. Each I/O pin pair can support both differential input and output buffers except the `clk[1,3,8,10]`, `PLL_L[1,4]_clk`, and `PLL_R[1,4]_clk` pins, which support differential input operations only.

Figure 6–2. I/O Banks in Arria II GZ Devices ***(Note 1)***, ***(2)***, ***(3)***, ***(4)***, ***(5)***, ***(6)***, ***(7)***, ***(8)***



Notes to Figure 6–2:

(1) Differential HSTL and SSTL outputs are not true differential outputs. They use two single-ended outputs with the second output programmed as inverted.

(2) Column I/O differential HSTL and SSTL inputs use LVDS differential input buffers without R_D OCT support.

(3) Column I/O supports LVDS outputs using single-ended buffers and external resistor networks.

(4) Column I/O supports PCI/PCI-X with an on-chip clamp diode. Row I/O supports PCI/PCI-X with an external clamp diode.

(5) Clock inputs on column I/Os are powered by $V_{CCCLKIN}$ when configured as differential clock inputs. They are powered by V_{CCIO} when configured as single-ended clock inputs. All outputs use the corresponding bank V_{CCIO}.

(6) Row I/O supports the true LVDS output buffer.

(7) Column and row I/O banks support LVPECL standards for input clock operation.

(8) Figure 6–2 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 devices are arranged in groups called modular I/O banks. Depending on the device densities, the number of I/O banks range from 6 to 20. Table 6–3 and Table 6–4 show the number of I/O pins available in each I/O bank.

Table 6–3. Available I/O Pins in Each Arria II GX 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	EP2AGX45	22	—	38	—	18	—	18	—	38	—	22	—	156
	EP2AGX65	22	—	38	—	18	—	18	—	38	—	22	—	156
572-pin Flip Chip FBGA	EP2AGX45	38	—	38	—	50	—	50	—	38	—	38	—	252
	EP2AGX65	38	—	38	—	50	—	50	—	38	—	38	—	252
	EP2AGX95	38	—	42	—	50	—	50	—	38	—	42	—	260
	EP2AGX125	38	—	42	—	50	—	50	—	38	—	42	—	260
780-pin Flip Chip FBGA	EP2AGX45	54	—	70	—	66	—	50	—	70	—	54	—	364
	EP24GX65	54	—	70	—	66	—	50	—	70	—	54	—	364
	EP2AGX95	54	—	74	—	66	—	50	—	70	—	58	—	372
	EP2AGX125	54	—	74	—	66	—	50	—	70	—	58	—	372
	EP2AGX190	54	—	74	—	66	—	50	—	70	—	58	—	372
	EP2AGX260	54	—	74	—	66	—	50	—	70	—	58	—	372
1152-pin Flip Chip FBGA	EP2AGX95	70	—	74	16	66	—	66	—	70	16	74	—	452
	EP2AGX125	70	—	74	16	66	—	66	—	70	16	74	—	452
	EP2AGX190	70	32	74	32	66	32	66	32	70	32	74	32	612
	EP2AGX260	70	32	74	32	66	32	66	32	70	32	74	32	612

Note to Table 6–3:

(1) The number of I/O pins include all general purpose I/Os, dedicated clock pins, and dual-purpose configuration pins. Transceiver pins and dedicated configuration pins are not included in the I/O pin count.

Table 6–4. Available I/O Pins in Each Arria II GZ I/O Bank ***(Note 1)***

Package	Device	Bank																			Total
		1A	1C	2A	2C	3A	3B	3C	4A	4B	4C	5A	5C	6A	6C	7A	7B	7C	8A	8B	8C	
780-pin Flip Chip FBGA	EP2AGZ300	—	1	—	—	40	—	28	40	—	30	—	—	—	—	40	—	30	40	—	32	281
	EP2AGZ350	—	1	—	—	40	—	28	40	—	30	—	—	—	—	40	—	30	40	—	32	281
1152-pin Flip Chip FBGA	EP2AGZ225	46	42	—	—	40	24	30	40	24	30	—	—	46	42	40	24	30	40	24	32	554
	EP2AGZ300	46	42	—	—	40	24	30	40	24	30	—	—	46	42	40	24	30	40	24	32	554
	EP2AGZ350	46	42	—	—	40	24	30	40	24	30	—	—	46	42	40	24	30	40	24	32	554
1517-pin Flip Chip FBGA	EP2AGZ225	46	42	48	42	40	24	30	40	24	30	48	42	46	42	40	24	30	40	24	32	734
	EP2AGZ300	46	42	48	42	40	24	30	40	24	30	48	42	46	42	40	24	30	40	24	32	734
	EP2AGZ350	46	42	48	42	40	24	30	40	24	30	48	42	46	42	40	24	30	40	24	32	734

Note to Table 6–4:

(1) The number of I/O pins include all general purpose I/Os, dedicated clock pins, and dual-purpose configuration pins. Transceiver pins and dedicated configuration pins are not included in the I/O pin count.

In Arria II devices, the maximum number of I/O banks per side, excluding the configuration banks, is either four or six, depending on the device density. All Arria II devices support migration across device densities and packages. When migrating between devices with a different number of I/O banks per side, it is the "B" bank that 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. Table 6–5 and Table 6–6 list the pin migration across device densities and packages.

Table 6–5. Pin Migration Across Densities in Arria II GX Devices *(Note 1)*

Package	Pin Type	Device					
		EP2AGX45	EP2AGX65	EP2AGX95	EP2AGX125	EP2AGX190	EP2AGX260
358-pin Flip Chip UBGA	I/O	144	144	—	—	—	—
	Clock	12	12	—	—	—	—
	XCVR channel	4	4	—	—	—	—
572-pin Flip Chip FBGA	I/O	240	240	248	248	—	—
	Clock	12	12	12	12	—	—
	XCVR channel	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 6–5:

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

Table 6–6. Pin Migration Across Densities in Arria II GZ Devices *(Note 1)* (Part 1 of 2)

Package	Pin Type	Device		
		EP2AGZ225	EP2AGZ300	EP2AGZ350
780-pin Flip Chip FBGA	I/O	—	280	280
	Clock	—	1	1
	XVCR channel	—	16	16
1152-pin Flip Chip FBGA	I/O	550	550	550
	Clock	4	4	4
	XVCR channel	16	16	16

Table 6–6. Pin Migration Across Densities in Arria II GZ Devices *(Note 1)* (Part 2 of 2)

Package	Pin Type	Device		
		EP2AGZ225	EP2AGZ300	EP2AGZ350
1517-pin Flip Chip FBGA	I/O	726	726	726
	Clock	8	8	8
	XVCR channel	24	24	24

Note to Table 6–6:

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

I/O Structure

The I/O element (IOE) in the Arria II devices contains a bidirectional I/O buffer and I/O registers to support a completely embedded bidirectional single data rate (SDR) or double data rate (DDR) transfer. The IOEs are located in I/O blocks around the periphery of the Arria II device. There are up to four IOEs per row I/O block and four IOEs per column I/O block. The row IOEs drive row, column, or direct link interconnects. The column IOEs drive column interconnects.

The Arria II bidirectional IOE supports the following features:

- Programmable input delay
- Programmable output-current strength
- Programmable slew rate
- Programmable bus-hold
- Programmable pull-up resistor
- Programmable output delay
- Open-drain output
- R_S OCT
- R_D OCT
- R_T OCT for Arria II GZ devices
- Dynamic OCT for Arria II GZ devices
- PCI clamping diode

I/O registers are composed of the input path for handling data from the pin to the core, the output path for handling data from the core to the pin, and the output enable path for handling the OE signal to the output buffer. These registers allow faster source-synchronous register-to-register transfers and resynchronization. You can bypass each block of the output and output enable paths. Figure 6–3 and Figure 6–4 show the Arria II IOE structure.

Figure 6–3. IOE Structure for Arria II GX Devices

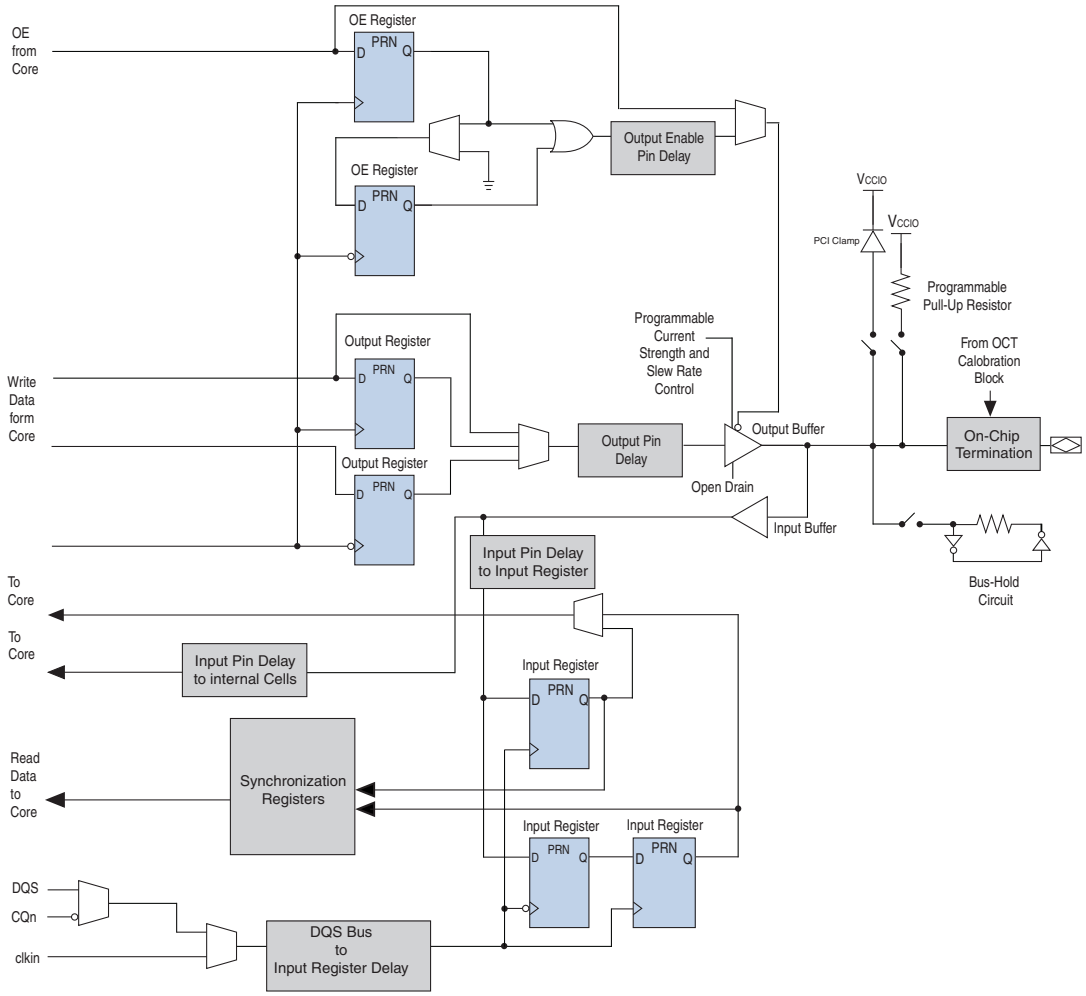
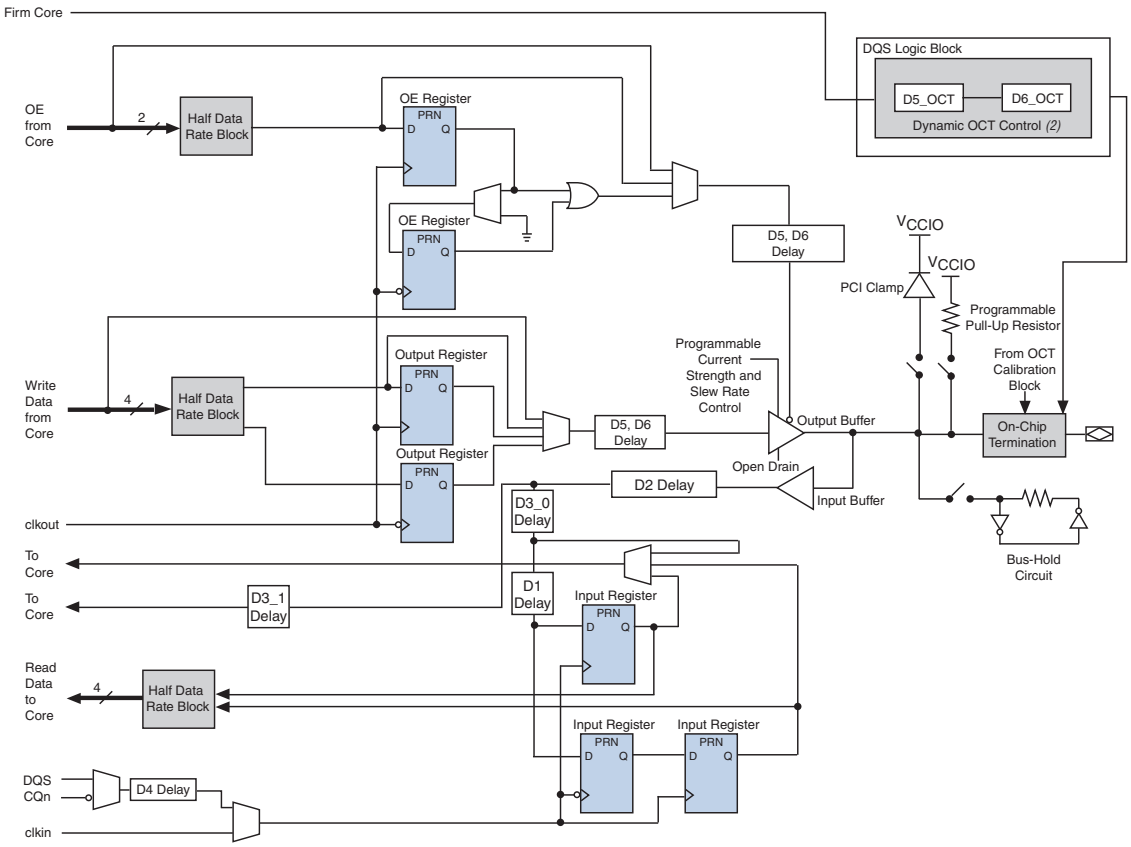


Figure 6–4. IOE Structure for Arria II GZ Devices *(Note 1)*, *(2)*



Notes to Figure 6–4:

(1) The `D3_0` and `D3_1` delays have the same available settings in the Quartus® II software.

(2) One dynamic OCT control is available per DQ/DQS group.

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

3.3-V I/O Interface

Arria II I/O buffers support 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 V_{CCIO} voltage is powered by 3.3 V or 3.0 V for Arria II GX devices and 3.0 V only for Arria II GZ devices.

To ensure device reliability and proper operation when interfacing a 3.3-V I/O system with Arria II devices, do not exceed the absolute maximum ratings. Altera recommends performing IBIS simulation to determine that the overshoot and undershoot voltages are within the guidelines.

When you use the Arria II 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 devices support R_S OCT for all LVTTL/LVCMOS I/O standards in all I/O banks.

When you use the Arria II device as a receiver, use a clamping diode (on-chip or off-chip) to limit overshoot. Arria II 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 to use a 3.0-V V_{CCIO} bank supply voltage. In this method, the clamp diode (on-chip or off-chip), can sufficiently clamp overshoot voltage 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 using the V_{CCIO} at 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 the absolute maximum rating and maximum allowed overshoot during transitions, refer to the *Devices Datasheet for Arria II Devices* chapter.

External Memory Interfaces

In addition to I/O registers in each IOE, Arria II 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 Devices* chapter.

High-Speed Differential I/O with DPA Support

Arria II 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 and DPA in Arria II Devices* chapter.

Programmable Current Strength

The output buffer for each Arria II 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 6–7 and Table 6–8 list the programmable current strength settings for Arria II devices.

Table 6–7. Programmable Current Strength for Arria II GX Devices *(Note 1)* (Part 1 of 2)

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**

Table 6–7. Programmable Current Strength for Arria II GX Devices *(Note 1)* (Part 2 of 2)

I/O Standard	I_{OL} / I_{OH} Current Strength Setting (mA) for Top, Bottom, and Right I/O Pins
HSTL-12 Class I	12, 10, 8
HSTL-12 Class II	**16**
BLVDS	8, 12, 16

Notes to Table 6–7:

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

Table 6–8. Programmable Current Strength for Arria II GZ Devices *(Note 1)*, *(2)*

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

Notes to Table 6–8:

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

(2) The 3.3-V LVTTL and 3.3-V LVCMOS are supported using V_{CCIO} and V_{CCPD} at 3.0 V.



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 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 with R_S OCT.

Table 6–9 lists the default slew rate settings from the Quartus II software.

Table 6–9. Default Slew Rate Settings for Arria II Devices

I/O Standard	Arria II GX Device		Arria II GZ Device	
	Slew Rate Option	Default Slew Rate (Fast)	Slew Rate Option	Default Slew Rate (Fast)
1.2-V, 1.5-V, 1.8-V, 2.5-V LVCMOS, and 3.3-V LVTTL/LVCMOS *(1)*	0, 1	1	0, 1, 2, 3	3
SSTL-2, SSTL-18, SSTL-15, HSTL-18, HSTL-15, and HSTL-12	1	1	0, 1, 2, 3	3
3.0-V PCI/PCI-X	0, 1	1	0, 1, 2, 3	3
LVDS_E_1R, mini-LVDS_E_1R, and RSDS_E_1R *(2)*	1	1	0, 1, 2, 3	3
LVDS_E_3R, mini-LVDS_E_3R, and RSDS_E_3R	1	1	0, 1, 2, 3	3

Notes to Table 6–9:

(1) Programmable slew rate is not supported for 3.3-V LVTTL/LVCMOS in Arria II GX devices.

(2) LVDS_E_1R and mini-LVDS_E_1R is not supported in Arria II GX devices.

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 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. You must use an external pull-up resistor to pull the high-Z output to logic high.

Bus Hold

Each Arria II 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, an external pull-up or pull-down resistor is not required 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 and 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.

For more information about the specific sustaining current driven through this resistor and the overdrive current used to identify the next-driven input level, refer to *Device Datasheet for Arria II Devices* chapter.

Programmable Pull-Up Resistor

Each Arria II 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 LVDS transmitters support programmable pre-emphasis to compensate the frequency dependent attenuation of the transmission line. For programmable pre-emphasis control, the Quartus II software allows two settings for Arria II GX devices and four settings for Arria II GZ devices.

For more information about programmable pre-emphasis, refer to the *High-Speed Differential I/O Interfaces and DPA in Arria II Devices* chapter.

Programmable Differential Output Voltage

Arria II 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.

For more information about programmable V_{OD}, refer to the *High-Speed Differential I/O Interfaces and DPA in Arria II Devices* chapter.

MultiVolt I/O Interface

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

You can connect the VCCIO pins to a power supply voltage level listed in Table 6–10, 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).

You must connect the Arria II GX VCCPD power pins to a 2.5-, 3.0-, or 3.3-V power supply and the Arria II GZ VCCPD power pins to a 2.5- or 3.0-V power supply. Using these power pins to supply the pre-driver power to the output buffers increases the performance of the output pins. Table 6–10 lists the Arria II MultiVolt I/O support.

Table 6–10. MultiVolt I/O Support for Arria II Devices ***(Note 1)***

VCCIO (V) *(2)*	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) (4)*	✓ *(3) (4)*	—	—	—	✓	—	—
3.0	—	—	—	✓	✓ *(4)*	✓ *(4)*	—	—	—	—	✓	—
3.3 *(5)*	—	—	—	✓	✓ *(4)*	✓ *(4)*	—	—	—	—	—	✓

Notes to Table 6–10:

(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 V_{IL} maximum and V_{IH} minimum voltage specifications.

(2) Each I/O bank of an Arria II device 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. The LVDS I/O standard is not supported when V_{CCIO} is 3.0 or 3.3 V. The LVDS input operations are supported when V_{CCIO} is 1.2, 1.5, 1.8, or 2.5 V. The LVDS output operations are only supported when V_{CCIO} is 2.5 V.

(3) Altera recommends using an external clamp diode when V_{CCIO} is 2.5 V and the input signal is 3.0 or 3.3 V.

(4) Altera recommends using an external clamp diode on the row I/O pins when the input signal is 3.0 or 3.3 V for Arria II GZ devices.

(5) Not applicable for Arria II GZ devices.

OCT Support

Arria II devices feature OCT to provide I/O impedance matching and termination capabilities. OCT maintains signal quality, saves board space, and reduces external component costs.

Arria II devices support the following features:

- "R_S OCT Without Calibration for Arria II Devices"
- "R_S OCT with Calibration for Arria II Devices"
- "Left-Shift R_S OCT Control for Arria II GZ Devices"
- "Expanded R_S OCT with Calibration for Arria II GZ Devices"
- "R_D OCT for Arria II LVDS Input I/O Standard"
- "R_T OCT with Calibration for Arria II GZ Devices"
- "Dynamic R_S and R_T OCT for Single-Ended I/O Standard for Arria II GZ Devices"

Arria II devices support OCT in all user I/O banks by selecting one of the OCT I/O standards. Arria II devices support OCT in the same I/O bank with different I/O standards if they use the same VCCIO supply voltage. You can independently configure each I/O buffer in an I/O bank to support OCT or programmable current strength. However, you cannot configure both R_S OCT and programmable current strength for the same I/O buffer.

A pair of RUP and RDN pins are available in a given I/O bank for Arria II GX series-calibrated termination and shared for Arria II GZ series- and parallel-calibrated termination. RUP and RDN pins share the same V_{CCIO} and GND, respectively, with the I/O bank where they are located. RUP and RDN pins are dual-purpose I/Os, and function as regular I/Os if you do not use the calibration circuit.

For R_S OCT, the connections are as follows:

- The RUP pin is connected to V_{CCIO} 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.
- The RDN pin is connected to GND through an external 25-Ω ±1% or 50-Ω ±1% resistor for an R_S OCT value of 25-Ω or 50-Ω, respectively.

For R_T OCT, the connections are as follows:

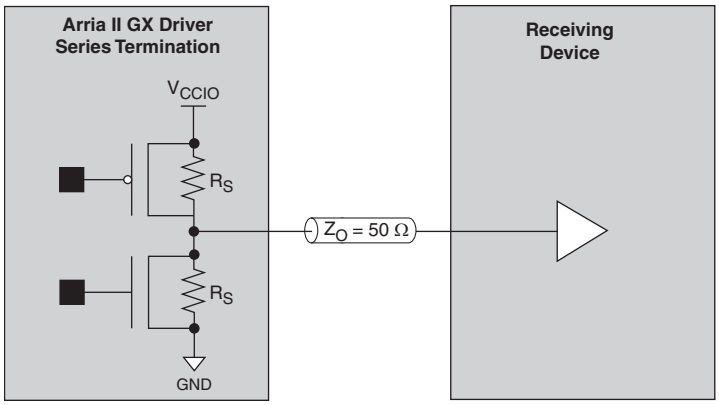
- The RUP pin is connected to V_{CCIO} through an external 50-Ω ±1% resistor.
- The RDN pin is connected to GND through an external 50-Ω ±1% resistor.

R_S OCT Without Calibration for Arria II Devices

Arria II 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 devices support R_S OCT for single-ended I/O standards.

The R_S shown in Figure 6–5 is the intrinsic impedance of output transistors. The typical R_S values are 25 Ω and 50 Ω.

Figure 6–5. R_S OCT without Calibration for Arria II Devices



To use OCT for:

- SSTL Class I standard—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—select the **25-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}).

R_S OCT with Calibration for Arria II Devices

Arria II devices support R_S OCT with calibration in all I/O banks. The R_S OCT 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 `RUP` and `RDN` pins, and dynamically enables or disables the transistors until they match.

The R_S shown in Figure 6–6 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 6–6. R_S OCT with Calibration for Arria II Devices

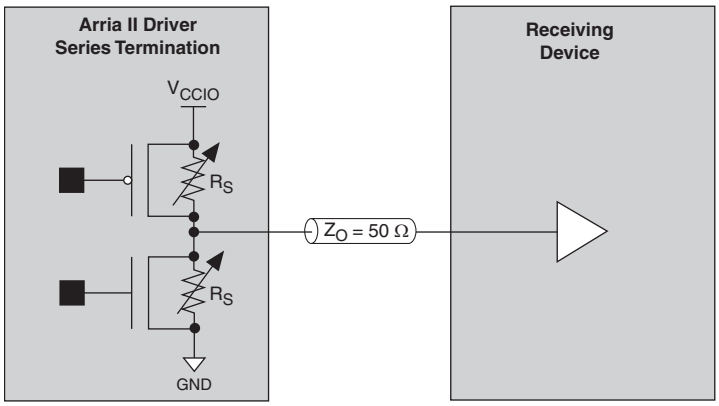


Table 6–11 lists the I/O standards that support R_S OCT with and without calibration.

Table 6–11. R_S OCT Selectable I/O Standards With and Without Calibration for Arria II Devices

I/O Standard	R_S OCT Termination Setting	
	Row I/O (Ω)	Column I/O (Ω)
3.3-V LVTTL/LVCMOS *(1)*, *(2)*	50	50
	25	25
3.0-V LVTTL/LVCMOS	50	50
	25	25
2.5-V LVTTL/LVCMOS	50	50
	25	25
1.8-V LVTTL/LVCMOS	50	50
	25	25
1.5-V LVCMOS	50	50
	25 *(3)*	25
1.2-V LVCMOS	50	50
	25 *(3)*	25
SSTL-2 Class I	50	50
SSTL-2 Class II	25	25
SSTL-18 Class I	50	50
SSTL-18 Class II	25	25
SSTL-15 Class I	50	50
SSTL-15 Class II *(2)*	—	25
HSTL-18 Class I	50	50
HSTL-18 Class II	25	25
HSTL-15 Class I	50	50
HSTL-15 Class II	25 *(3)*	25
HSTL-12 Class I	50	50
HSTL-12 Class II	25 *(3)*	25

Notes to Table 6–11:

(1) The 3.3-V LVTTL/LVCMOS standard is supported using V_{CCIO} at 3.0 V.
(2) Applicable for Arria II GZ devices only.
(3) Applicable for Arria II GX devices only.

Left-Shift R_S OCT Control for Arria II GZ Devices

Arria II GZ devices support left-shift series termination control. You can use left-shift series termination control to get the calibrated R_S OCT with half of the impedance value of the external reference resistors connected to the RUP and RDN pins. This feature is useful in applications that require both 25-Ω and 50-Ω calibrated R_S OCT at the same V_{CCIO}. For example, if your application requires 25-Ω and 50-Ω calibrated R_S OCT for SSTL-2 Class I and Class II I/O standards, you only need one OCT calibration block with 50-Ω external reference resistors.

You can enable the left-shift series termination control feature in the ALTIOBUF megafunction in the Quartus II software. The Quartus II software only allows left-shift series termination control for 25-Ω calibrated R_S OCT with 50-Ω external reference resistors connected to the RUP and RDN pins. You can only use left-shift series termination control for the I/O standards that support 25-Ω calibrated R_S OCT.



This feature is automatically enabled if you are using a bidirectional I/O with 25-Ω calibrated R_S OCT and 50-Ω R_T OCT.



For more information about how to enable the left-shift series termination feature in the ALTIOBUF megafunction, refer to the *I/O Buffer (ALTIOBUF) Megafunction User Guide*.

Expanded R_S OCT with Calibration for Arria II GZ Devices

OCT calibration circuits always adjust R_S OCT to match the external resistors connected to the RUP and RDN pin; however, it is possible to achieve R_S OCT values other than the 25-Ω and 50-Ω resistors. Theoretically, if you need a different R_S OCT value, you can change the resistance connected to the RUP and RDN pins accordingly. Practically, the R_S OCT range that Arria II GZ devices support is limited because of output buffer size and granularity limitations.

The Quartus II software only allows discrete R_S OCT calibration settings of 25, 40, 50, and 60 Ω . You can select the closest discrete value of R_S OCT with calibration settings in the Quartus II software to your system to achieve the closest timing. For example, if you are using 20-Ω R_S OCT with calibration in your system, you can select the **25-Ω R_S OCT with calibration** setting in the Quartus II software to achieve the closest timing.

Table 6–12 lists expanded R_S OCT with calibration supported in Arria II devices. Use expanded R_S OCT with calibration of SSTL and HSTL for impedance matching to improve signal integrity but do not use it to meet the JEDEC standard.

Table 6–12. Selectable I/O Standards with Expanded R_S OCT with Calibration Range for Arria II GZ Devices

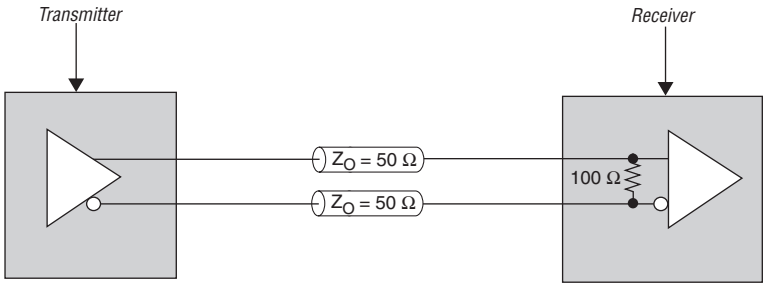
I/O Standard	Expanded R_S OCT Range	
	Row I/O (Ω)	Column I/O (Ω)
3.3-V LVTTL/LVCMOS	20–60	20–60
2.5-V LVTTL/LVCMOS	20–60	20–60
1.8-V LVTTL/LVCMOS	20–60	20–60
1.5-V LVTTL/LVCMOS	40–60	20–60
1.2-V LVTTL/LVCMOS	40–60	20–60
SSTL-2	20–60	20–60
SSTL-18	20–60	20–60
SSTL-15	40–60	20–60
HSTL-18	20–60	20–60
HSTL-15	40–60	20–60
HSTL-12	40–60	20–60

R_D OCT for Arria II LVDS Input I/O Standard

All I/O banks in Arria II GX devices support input R_D OCT with a nominal resistance value of 100 Ω, as shown in Figure 6–7. However, not all input differential pins support R_D OCT. You can enable R_D OCT when both the V_{CCIO} and V_{CCPD} is set to 2.5 V.

Arria II GZ column I/O banks and dedicated clock input pairs on the row I/O banks do not support R_D OCT. You can enable the Arria II GZ R_D OCT in row I/O banks when both the V_{CCIO} and V_{CCPD} is set to 2.5 V.

Figure 6–7. Differential Input On-Chip Termination for Arria II Devices

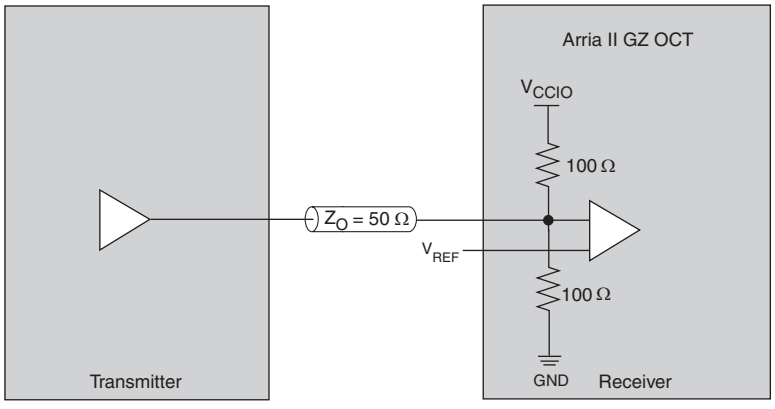


For more information about R_D OCT, refer to the *High-Speed Differential I/O Interfaces and DPA in Arria II Devices* chapter.

R_T OCT with Calibration for Arria II GZ Devices

Arria II GZ devices support R_T OCT with calibration in all banks. R_T OCT with calibration is only supported for input configuration of input and bidirectional pins. Output pin configurations do not support R_T OCT with calibration. Figure 6–8 shows R_T OCT with calibration. When you use R_T OCT, the V_{CCIO} of the bank must match the I/O standard of the pin where the R_T OCT is enabled.

Figure 6–8. R_T OCT with Calibration for Arria II GZ Devices



The R_T OCT calibration circuit compares the total impedance of the I/O buffer to the external 50-Ω ±1% resistors connected to the RUP and RDN pins and dynamically enables or disables the transistors until they match. 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. Table 6–13 lists the I/O standards that support R_T OCT with calibration.

Table 6–13. Selectable I/O Standards with R_T OCT with Calibration for Arria II GZ Devices

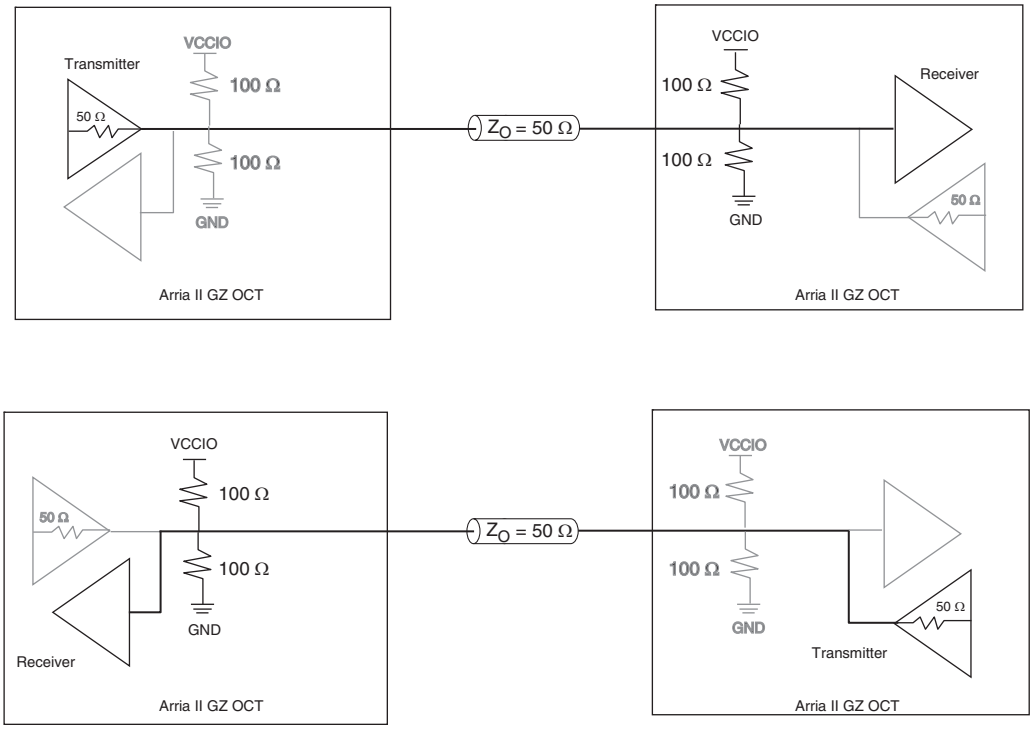
I/O Standard	R_T OCT Setting (Column I/O) (Ω)	R_T OCT Setting (Row I/O) (Ω)
SSTL-2 Class I, II	50	50
SSTL-18 Class I, II	50	50
SSTL-15 Class I, II	50	50
HSTL-18 Class I, II	50	50
HSTL-15 Class I, II	50	50
HSTL-12 Class I, II	50	50
Differential SSTL-2 Class I, II	50	50
Differential SSTL-18 Class I, II	50	50
Differential SSTL-15 Class I, II	50	50
Differential HSTL-18 Class I, II	50	50
Differential HSTL-15 Class I, II	50	50
Differential HSTL-12 Class I, II	50	50

Dynamic R_S and R_T OCT for Single-Ended I/O Standard for Arria II GZ Devices

Arria II GZ devices support on and off dynamic termination, both series and parallel, for a bidirectional I/O in all I/O banks. Figure 6–9 shows the termination schemes supported in Arria II GZ devices. Dynamic parallel termination is enabled only when the bidirectional I/O acts as a receiver and is disabled when it acts as a driver. Similarly, dynamic series termination is enabled only when the bidirectional I/O acts as a driver and is disabled when it acts as a receiver. This feature is useful for terminating any high-performance bidirectional path because signal integrity is optimized depending on the direction of the data.

Using dynamic OCT helps save power because device termination is internal instead of external. Termination only switches on during input operation, thus drawing less static power.

Figure 6–9. Dynamic R_T OCT in Arria II GZ Devices



For more information about tolerance specifications for OCT with calibration, refer to the *Device Datasheet for Arria II Devices* chapter.

Arria II OCT Calibration

Arria II GX devices support calibrated R_S OCT and Arria II GZ devices support calibrated R_S and R_T OCT on all I/O pins. You can calibrate the I/O banks with any of the OCT calibration blocks available in the device provided the V_{CCIO} of the I/O bank with the pins using calibrated OCT matches the V_{CCIO} of the I/O bank with the calibration block and its associated RUP and RDN pins.

For more information about the location of the OCT calibration blocks in Arria II devices, refer to the *Arria II Device Family Connection Guidelines* and *Arria II Device Pin-Outs*.

OCT Calibration Block

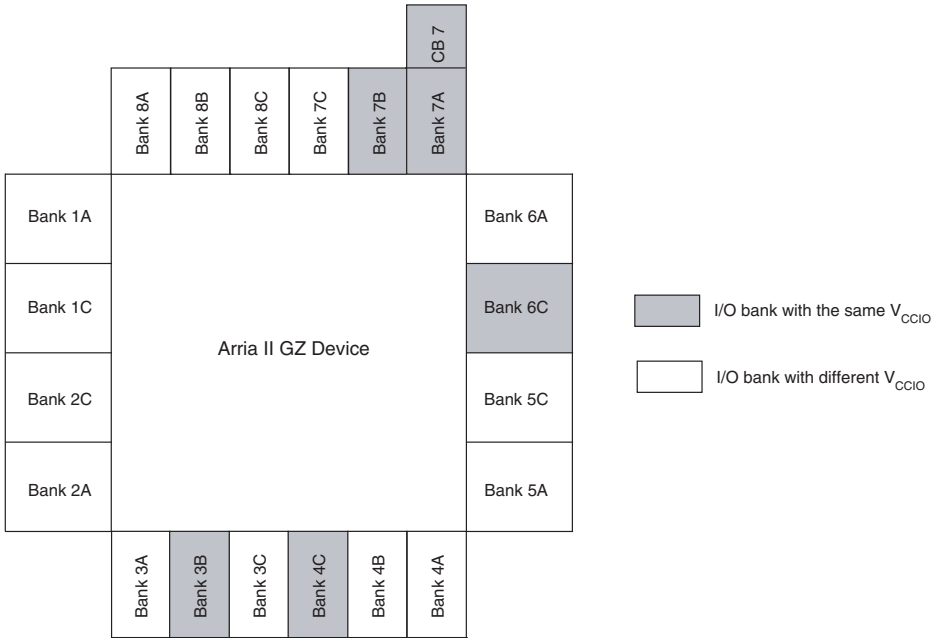
An OCT calibration block has the same V_{CCIO} as the I/O bank that contains the block. R_S OCT calibration is supported on all user I/O banks with different V_{CCIO} voltage standards, up to the number of available OCT calibration blocks. You can configure I/O banks to receive calibrated codes from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

For example, Figure 6–10 shows a group of I/O banks that has the same V_{CCIO} voltage. If a group of I/O banks has the same V_{CCIO} voltage, you can use one OCT calibration block to calibrate the group of I/O banks placed around the periphery. Because banks 3B, 4C, 6C, and 7B have the same V_{CCIO} as bank 7A, you can calibrate all four I/O banks (3B, 4C, 6C, and 7B) with the OCT calibration block (CB7) located in bank 7A. You can enable this by serially shifting out R_S OCT calibration codes from the OCT calibration block located in bank 7A to the I/O banks located around the periphery.

I/O banks that do not contain calibration blocks share calibration blocks with I/O banks that do contain calibration blocks.

Figure 6–10 is a top view example of the Arria II GZ silicon die that corresponds to a reverse view for flip chip packages. It is a graphical representation only. This figure does not show transceiver banks and transceiver calibration blocks.

Figure 6–10. Example of Calibrating Multiple I/O Banks with One Shared OCT Calibration Block in Arria II GZ Devices



For more information about the OCT calibration block, refer to the *ALT_OCT Megafunction User Guide*.

Termination Schemes for I/O Standards

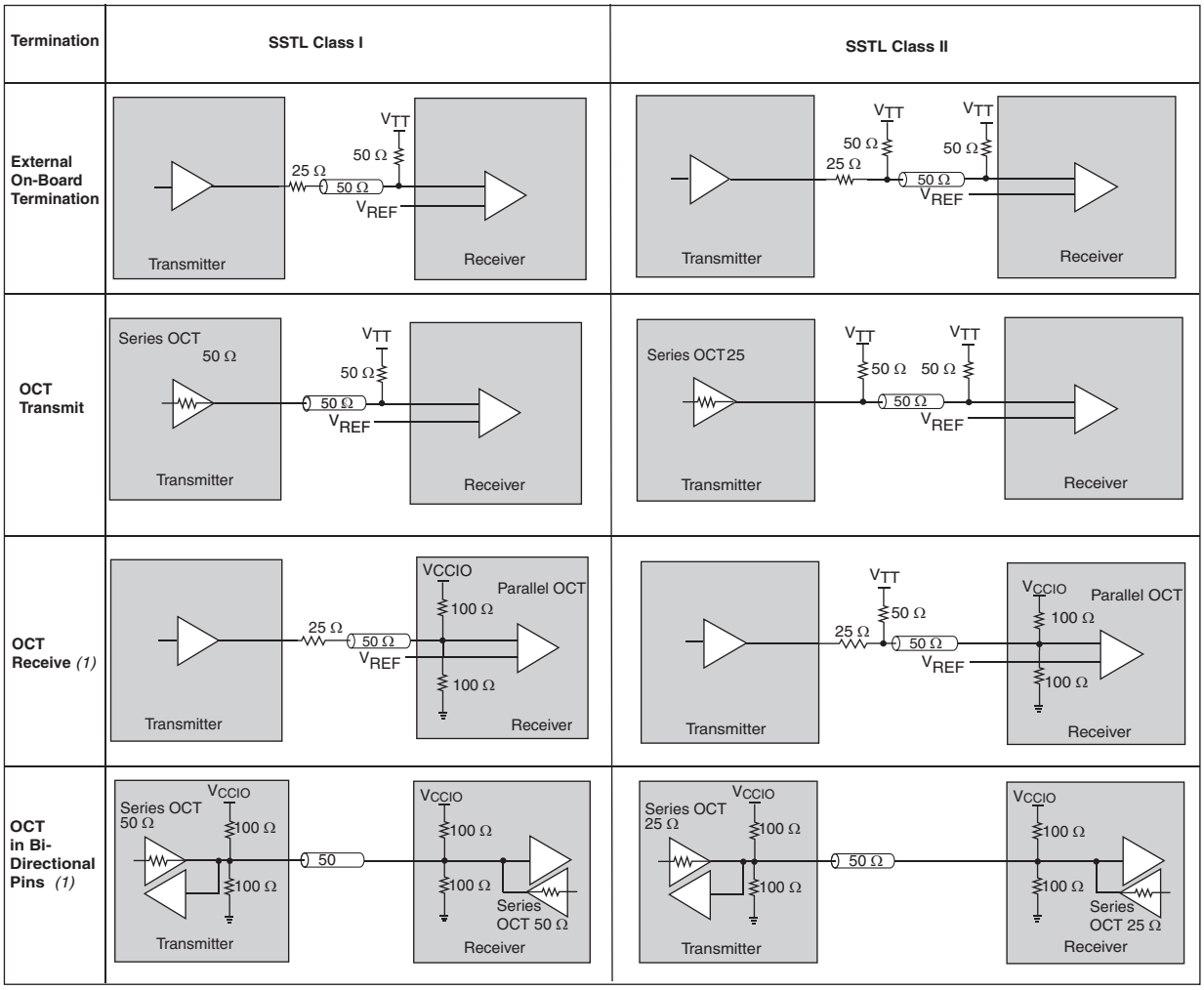
The following section describes the different termination schemes for I/O standards used in Arria II devices.

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 6–11 shows the details of SSTL I/O termination on Arria II devices.

Figure 6–11. SSTL I/O Standard Termination for Arria II Devices

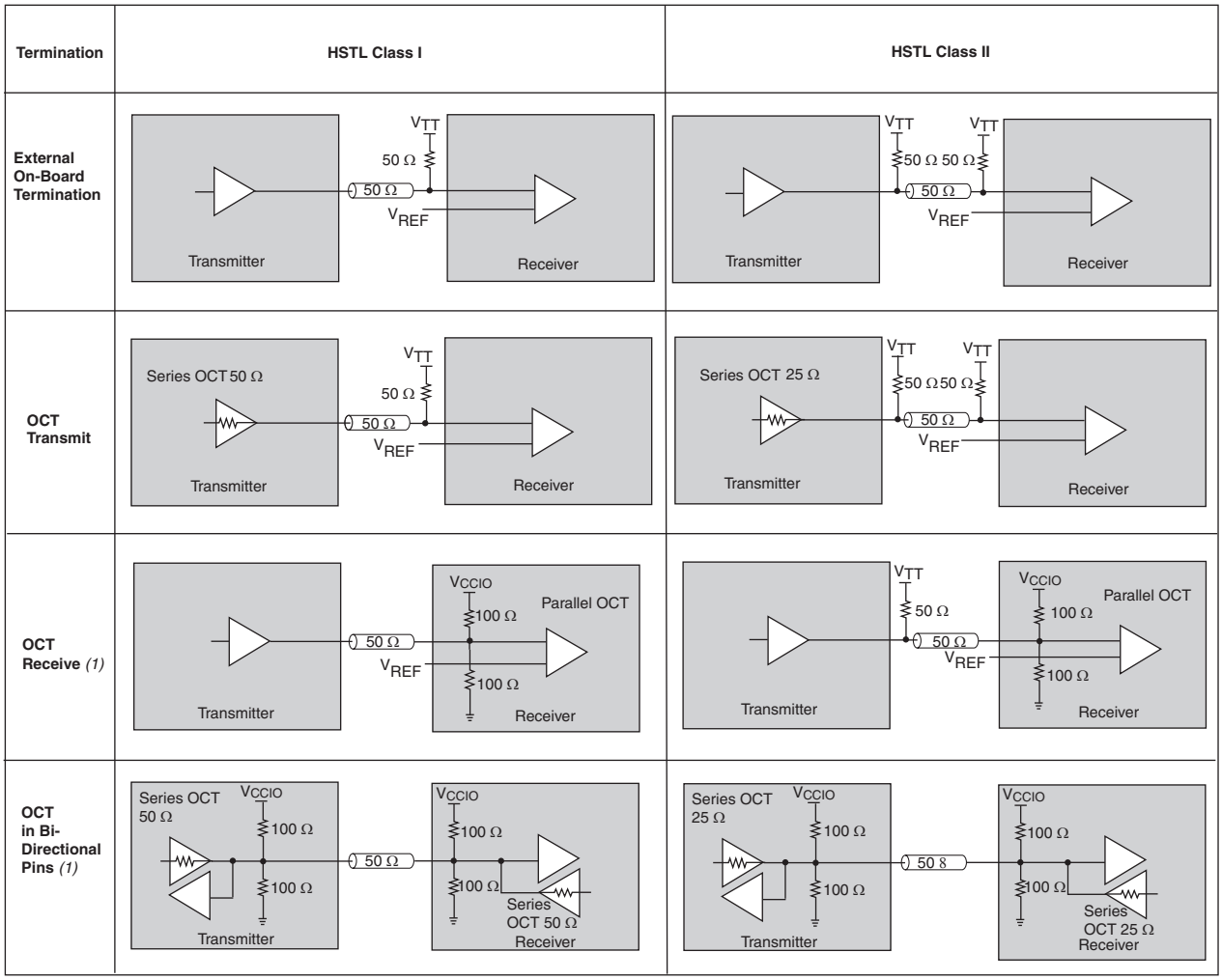


Note to Figure 6–11:

(1) Applicable to Arria II GZ devices only.

Figure 6–12 shows the details of HSTL I/O termination on Arria II devices.

Figure 6–12. HSTL I/O Standard Termination for Arria II Devices



Note to Figure 6–12:

(1) Applicable to Arria II GZ devices only.

Differential I/O Standards Termination

Arria II devices support differential SSTL-2 and SSTL-18, differential HSTL-18, HSTL-15, HSTL-12, LVDS, LVPECL, RSDS, and mini-LVDS. Figure 6–13 through Figure 6–14 show the details of various differential I/O terminations on Arria II devices.



Differential HSTL and SSTL outputs are not true differential outputs. They use two single-ended outputs with the second output programmed as inverted.

Figure 6–13 shows the details of differential SSTL I/O standard termination on Arria II devices.

Figure 6–13. Differential SSTL I/O Standard Termination for Arria II Devices

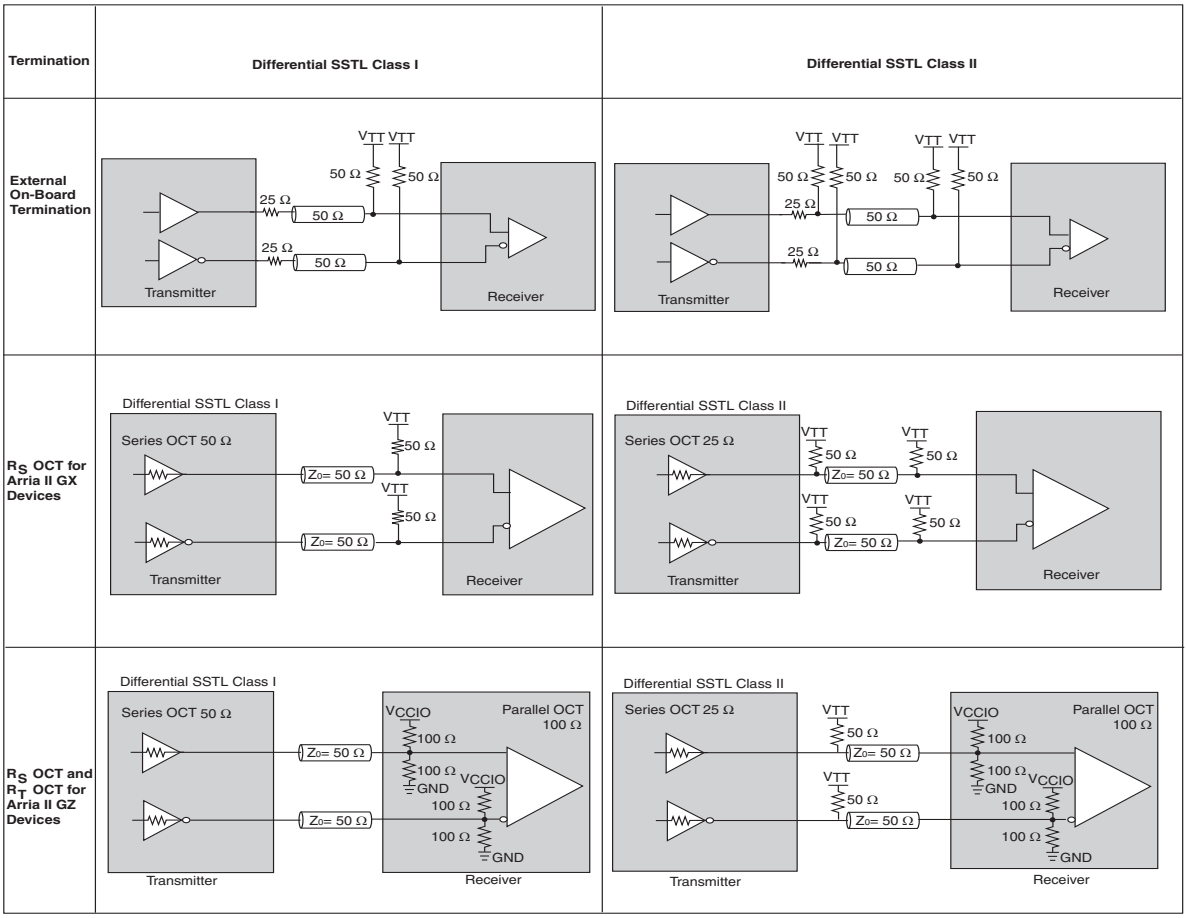
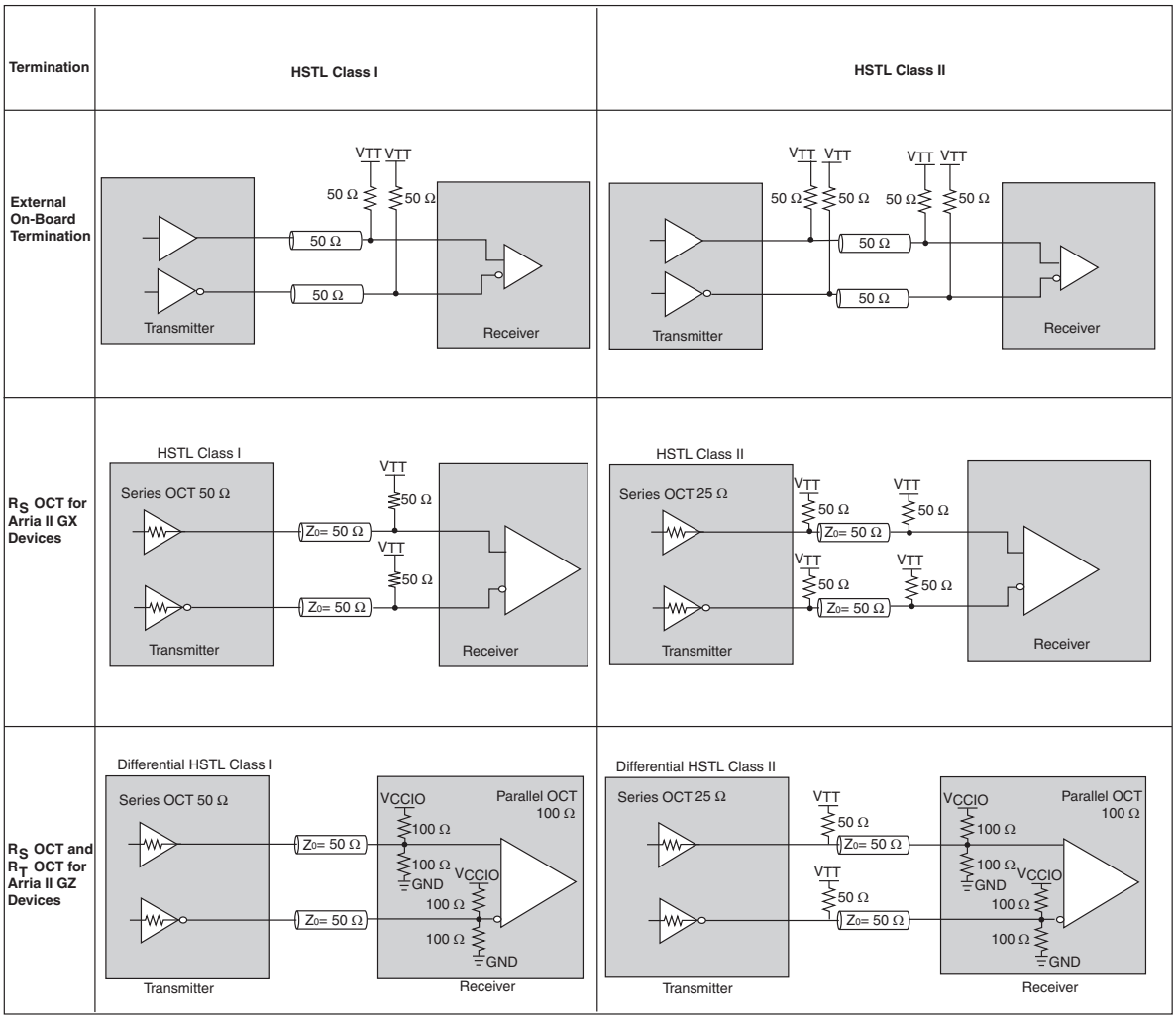


Figure 6–14 shows the details of differential HSTL I/O standard termination on Arria II devices.

Figure 6–14. Differential HSTL I/O Standard Termination for Arria II Devices

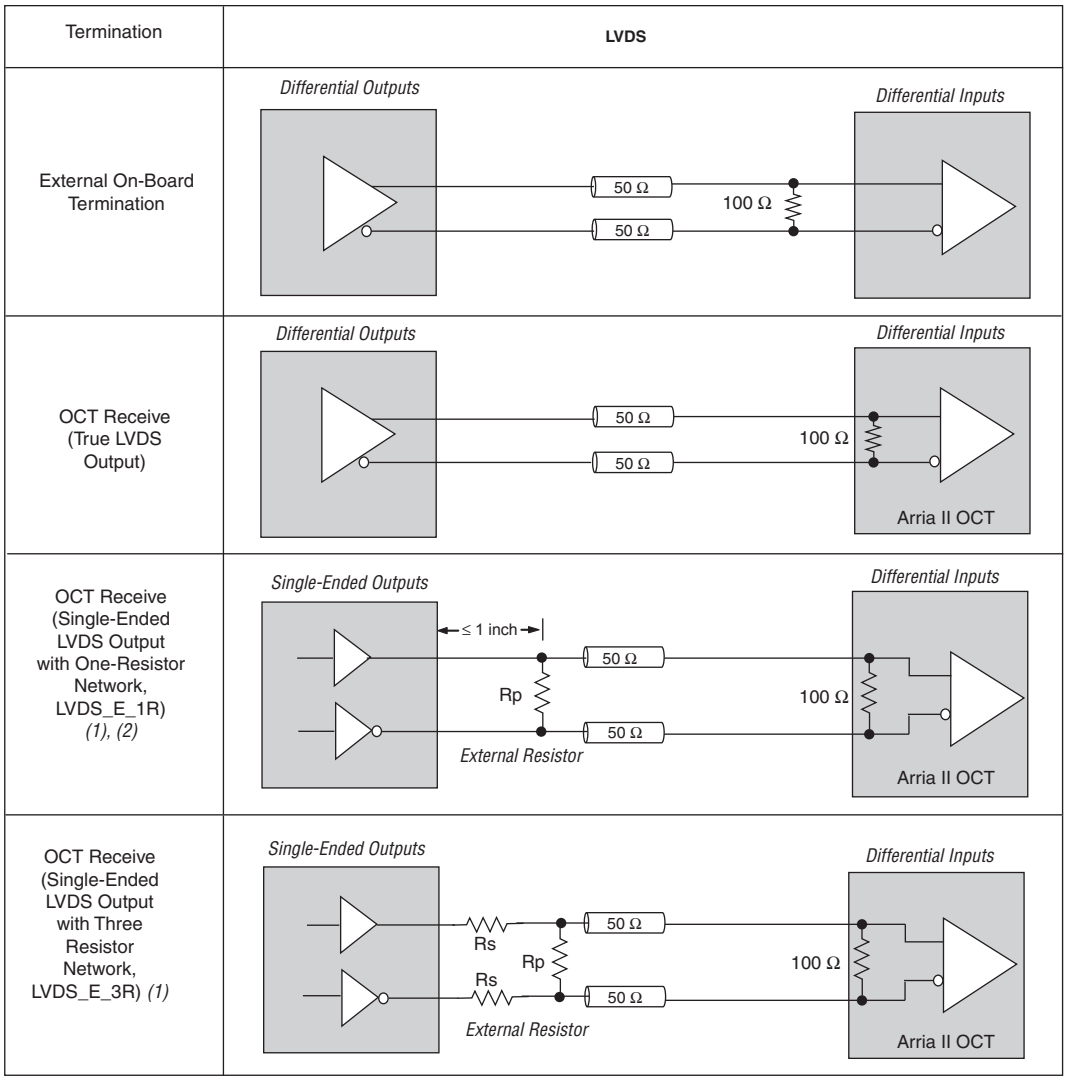


LVDS

The LVDS I/O standard is a differential high-speed, low-voltage swing, low-power, general-purpose I/O (GPIO) interface standard. Arria II 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 devices provide an optional 100-Ω differential termination resistor in the device with R_D OCT.

Figure 6–15 shows the details of LVDS termination in Arria II devices. The Arria II GZ R_D OCT is only available in the row I/O banks.

Figure 6–15. LVDS I/O Standard Termination for Arria II Devices ***(Note 1)***



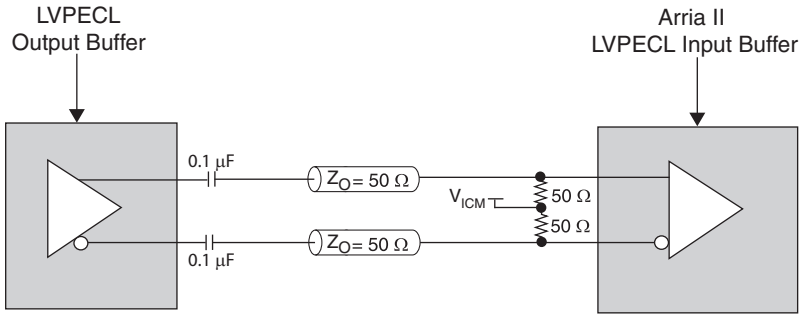
Notes to Figure 6–15:

(1) For LVDS output with a three-resistor network, the R_S and R_P values are 120 and 170 Ω, respectively. For LVDS output with a one-resistor network, the R_P value is 120 Ω.
(2) LVDS_E_1R is available for Arria II GZ devices only.

Differential LVPECL

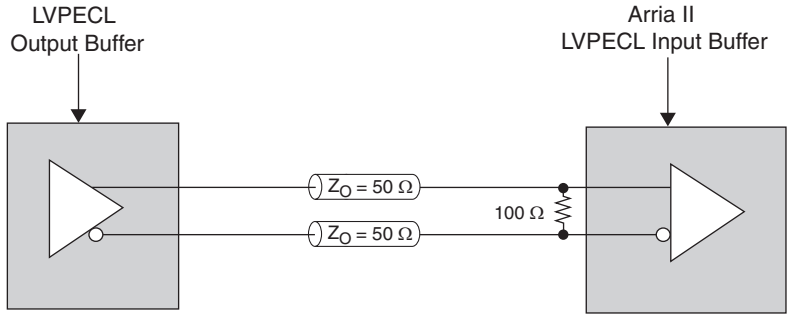
Arria II devices support the LVPECL I/O standard on input clock pins only. LVPECL output operation is not supported. LVDS input buffers are used to support LVPECL input operation. AC-coupling is required when the LVPECL common mode voltage of the output buffer is higher than Arria II LVPECL input common mode voltage. Figure 6–16 shows the AC-coupled termination scheme. The 50-Ω resistors used at the receiver end are external to the device.

Figure 6–16. LVPECL AC-Coupled Termination



Arria II devices support DC-coupled LVPECL if the LVPECL output common mode voltage is within the Arria II LVPECL input buffer specification (Figure 6–17).

Figure 6–17. LVPECL DC-Coupled Termination



RSDS

Arria II devices supports true 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 6–18. Only Arria II GZ row I/O banks support RSDS output using true LVDS output buffers without an external resistor network.

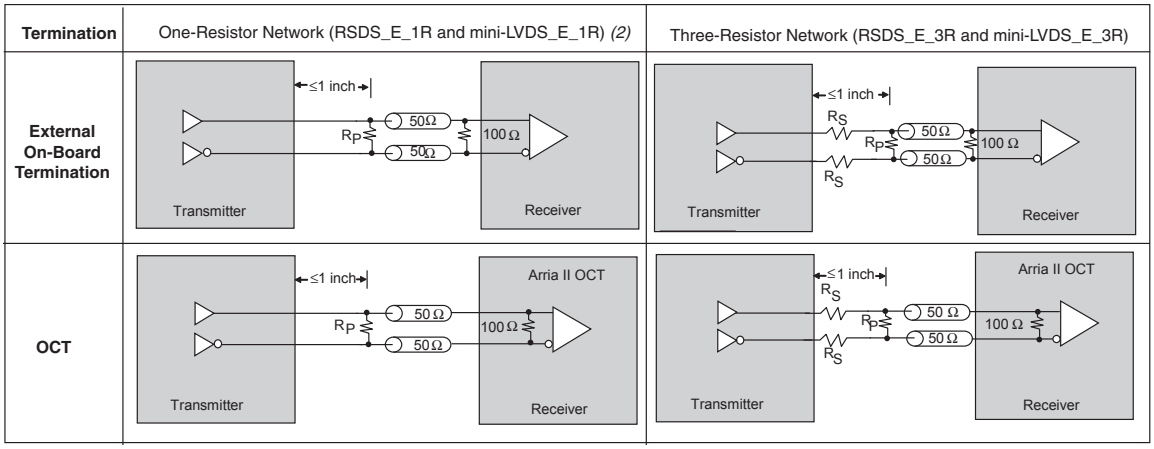
mini-LVDS

Arria II GX devices support true mini-LVDS with a three-resistor network using two single-ended output buffers for external three-resistor networks.

For Arria II GZ devices, use two single-ended output buffers with external one- or three-resistor networks (mini-LVDS_E_1R or mini-LVDS_E_3R). Arria II GZ row I/O banks support mini-LVDS output using true LVDS output buffers without an external resistor network.

Figure 6–18 shows the one-resistor and three-resistor topology for RSDS and mini-LVDS I/O standard termination.

Figure 6–18. RSDS and mini-LVDS I/O Standard Termination for Arria II Devices *(Note 1)*



Notes to Figure 6–18:

(1) R_p = 170 Ω and R_s= 120 Ω

(2) mini-LVDS_E_1R is applicable for Arria II GZ devices only.

A resistor network is required to attenuate the LVDS output-voltage swing to meet RSDS and mini-LVDS 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. Resistor Network

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

To validate that custom resistor values meet the RSDS requirements, Altera recommends performing additional simulations with IBIS models.

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

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

Design Considerations

Although Arria II devices feature various I/O capabilities for high-performance and high-speed system designs, there are several other design considerations that require your attention to ensure the success of your designs.

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 a superior noise margin.

Arria II R_S OCT provides the convenience of not using external components. When optimizing OCT for use in typical transmission line environments, the R_S OCT impedance must be equal to or less than the transmission line impedance for optimal performance. In ideal applications, setting the R_S OCT impedance to match the transmission line impedance avoids reflections. You can also use external pull-up resistors to terminate the voltage-referenced I/O standards such as SSTL and HSTL I/O standards.

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 devices provide an optional differential on-chip resistor when you use LVDS.

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 devices.

Non-Voltage-Referenced Standards

Each Arria II device I/O bank has its own VCCIO pins and supports only one V_{CCIO}. 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 I/O bank has a dedicated VREF pin while Arria II GZ I/O banks supports multiple VREF pins feeding a common VREF bus. The number of available VREF pins increases as device density increases. For Arria II GZ devices, if these pins are not used as VREF pins, they cannot be used as generic I/O pins and must be tied to V_{CCIO} or GND. Each bank can only have a single V_{CCIO} voltage level and a single V_{REF} voltage level at a given time.

Arria II GX I/O banks featuring single-ended or differential standards can support voltage-referenced standards as long as all voltage-referenced standards use the same V_{REF} setting.

For Arria II GZ devices, voltage-referenced input standards use their own V_{CCPD} level as the power source. This feature allows you to place voltage-referenced input signals in an I/O bank with a V_{CCIO} of 2.5 V or below. For example, you can place HSTL-15 input pins in an I/O bank with 2.5-V V_{CCIO}. However, the voltage-referenced input with R_T OCT enabled requires the V_{CCIO} of the I/O bank to match the voltage of the input standard.

Voltage-referenced bidirectional and output signals must be the same as the V_{CCIO} voltage of the I/O bank. 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 devices and includes essential information for designing systems with an Arria II 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 impedance of the I/O driver 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 required, Altera recommends series termination at the driver end (on-chip or off-chip).
- Output slew rate—Arria II GX devices have two levels and Arria II GZ devices have four 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 I/Os have on-chip clamping diodes. These clamping diodes are required for PCI/PCI-X standards and recommended for 3.3-V LVTTL/CMOS standards.
- When you use clamping diodes, the floating well of the I/O is clamped to V_{CCIO}. As a result, the Arria II 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 device supports a maximum DC current of 8 mA.

Pin Placement Guideline

To validate your pin placement, Altera recommends creating a Quartus II design, entering in your device I/O assignments, and compiling your design. 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.

Document Revision History

Table 6–14 lists the revision history for this chapter.

Table 6–14. Document Revision History (Part 1 of 2)

Date	Version	Changes
December 2011	4.2	■ Updated Table 6–2 and Table 6–11. ■ Minor text edits.
June 2011	4.1	■ Updated Table 6–9 and Table 6–10. ■ Updated Figure 6–3 and Figure 6–4. ■ Minor text edits.

Table 6–14. Document Revision History (Part 2 of 2)

Date	Version	Changes
December 2010	4.0	Updated for the Quartus II software version 10.1 release: ■ Added Arria II GZ device information. ■ Added "Left-Shift RS OCT Control for Arria II GZ Devices", "Expanded RS OCT with Calibration for Arria II GZ Devices", "RT OCT with Calibration for Arria II GZ Devices", and "Dynamic RS and RT OCT for Single-Ended I/O Standard for Arria II GZ Devices" sections. ■ Added Figure 6–1.
July 2010	3.0	Updated for Arria II GX v10.0 release: ■ Updated Table 6–4, Table 6–5, and Table 6–6. ■ Updated Figure 6–1. ■ Updated "Overview" section.
October 2009	2.0	Updated for Arria II GX v9.1 release: ■ Updated Table 6–2 and Table 6–3. ■ Updated Figure 6–2, Figure 6–13, and Figure 6–14 ■ Minor text edits.
June 2009	1.1	■ Updated Table 6–1, Table 6–4 and Table 6–5. ■ Updated "Programmable Slew Rate Control", "Programmable Differential Output Voltage", "Mini-LVDS", "RSDS", "OCT Calibration Block", and "I/O Placement Guidelines" sections. ■ Updated Figure 6–1, Figure 6–6, Figure 6–7, Figure 6–8, Figure 6–9, Figure 6–10, and Figure 6–14.
February 2009	1.0	Initial release.

7. External Memory Interfaces in Arria II Devices

AIIGX51007-4.1

This chapter describes the hardware features in Arria® II devices that facilitate high-speed memory interfacing for the double data rate (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.

Arria II devices 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 DDR memory standards, such as DDR3, DDR2, DDR SDRAM, QDR II, QDR II+ SRAM, and RLDRAM II. The Arria II FPGA supports DDR external memory on the top, bottom, left, and right I/O banks.

The high-performance memory interface solution includes the self-calibrating ALTMEMPHY megafunction and UniPHY Intellectual Property (IP) core, optimized to take advantage of the Arria II I/O structure and the Quartus® II TimeQuest Timing Analyzer. The ALTMEMPHY megafunction and UniPHY IP core provide the total solution for the highest reliable frequency of operation across process, voltage, and temperature (PVT) variations.

The ALTMEMPHY megafunction and UniPHY IP core instantiate a phase-locked loop (PLL) and PLL reconfiguration logic to adjust the resynchronization phase shift based on PVT variation.

This chapter includes the following sections:

- "Memory Interfaces Pin Support for Arria II Devices" on page 7–3
- "Combining ×16/×18 DQ/DQS Groups for ×36 QDR II+/QDR II SRAM Interface" on page 7–21
- "Arria II External Memory Interface Features" on page 7–24

Arria II GZ devices only support the UniPHY IP core. Arria II GX devices support the QDR II and QDR II + SRAM controller with the UniPHY IP core, and DDR3, DDR2, and the DDR SDRAM controller with the ALTMEMPHY megafunction.

RLDRAM II is only available in Arria II GZ devices.

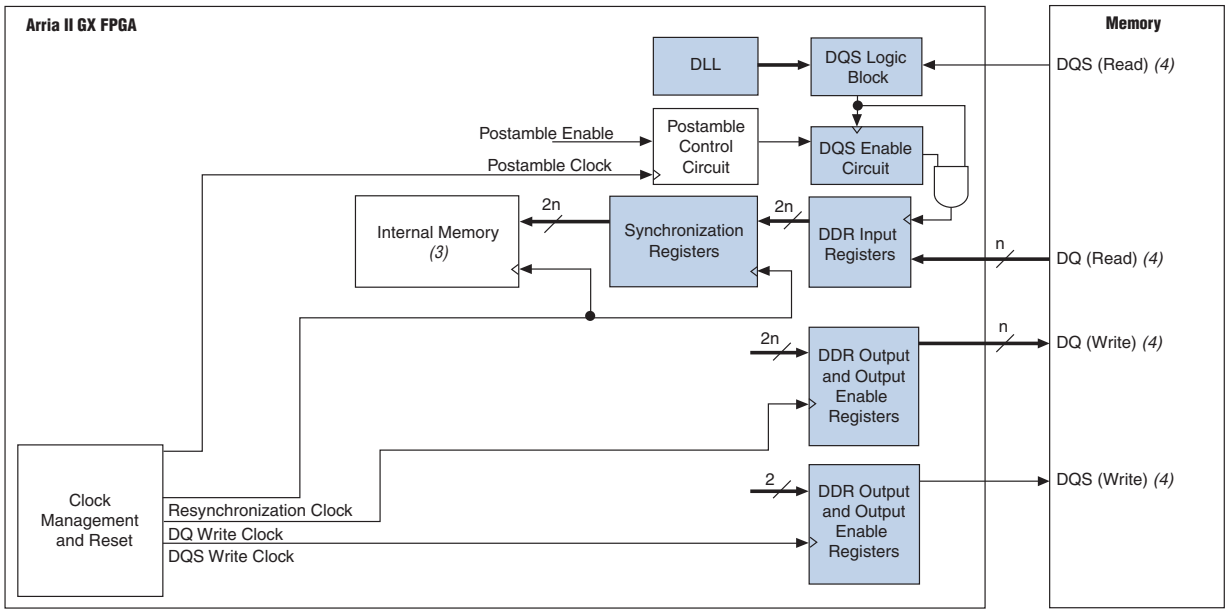
For more information about any of the above-mentioned features, refer to the *I/O Features in Arria II Devices* or the *Clock Networks and PLLs in Arria II Devices* chapter.

For more information about external memory system specifications, implementation, board guidelines, timing analysis, simulation, debug information, ALTMEMPHY megafunction and UniPHY IP core support for Arria II devices, refer to the *External Memory Interface Handbook*.

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Figure 7–1 and Figure 7–2 show the memory interface datapath overview for Arria II GX and Arria II GZ devices, respectively.

Figure 7–1. External Memory Interface Datapath Overview for Arria II GX Devices ***(Note 1)*** **,** ***(2)***



Notes to Figure 7–1:

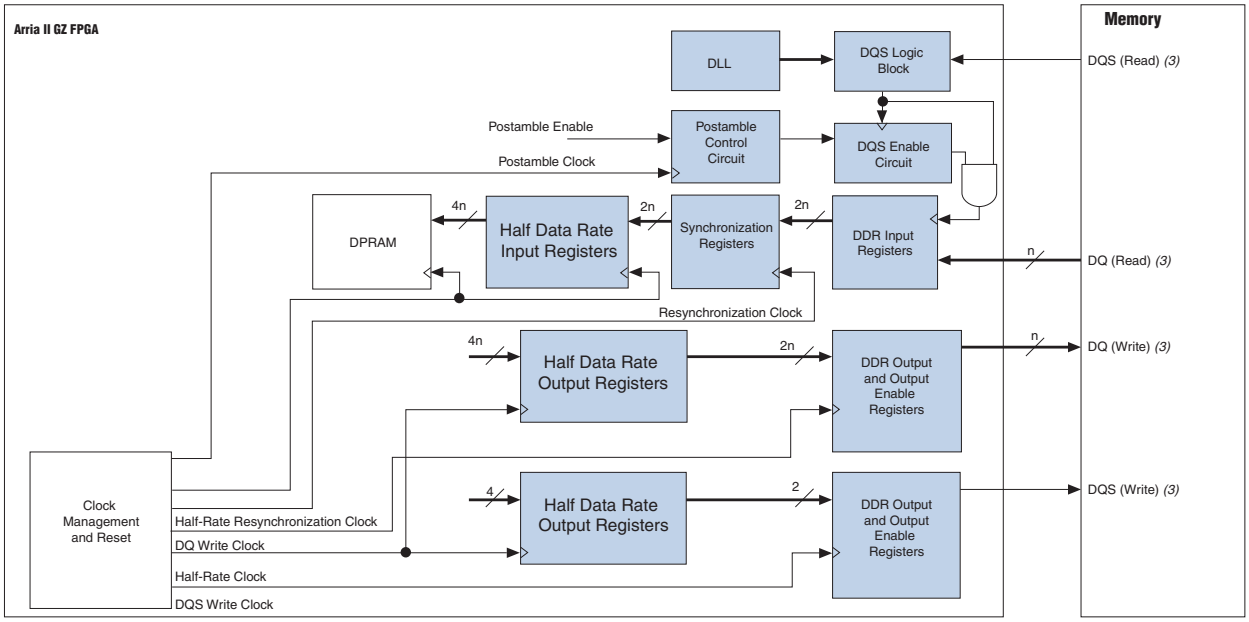
(1) You can bypass each register block.

(2) Shaded blocks are implemented in the I/O element (IOE).

(3) The memory blocks used for each memory interface may differ slightly.

(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.

Figure 7–2. External Memory Interface Datapath Overview for Arria II GZ Devices *(Note 1), (2)*



Notes to Figure 7–2:

(1) You can bypass each register block.

(2) The blocks used for each memory interface may differ slightly. The shaded blocks are part of the Arria II GZ IOE.

(3) These signals may be bidirectional or unidirectional, depending on the memory standard. When bidirectional, the signal is active during both read and write operations.

Memory Interfaces Pin Support for Arria II Devices

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 devices support all these pins.

If you have more than one clock pair, you must place them in the same DQ group. For example, if you have two clock pairs, you must place both of them in the same ×4 DQS group.

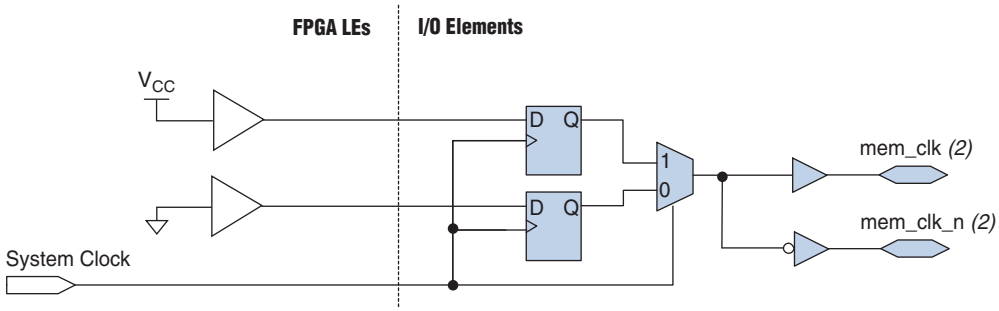
For more information about pin connections, refer to the *Arria II Device Family Pin Connection Guidelines*.

The DDR3, DDR2, DDR SDRAM, and RLDRAM II devices use CK and CK# signals to capture the address and command signals. You can generate these signals to mimic the write-data strobe with Arria II DDR I/O registers (DDIOs) to ensure that timing relationships between the CK/CK# and DQS signals (t_{DQSS}, t_{DSS}, and t_{DSH} in DDR3, DDR2, and DDR SDRAM devices) are met. The QDR II+/QDR II SRAM devices use the same clock (K/K#) to capture the write data, address, and command signals.

For more information about pin location requirements, which pins to use as memory clock pins, and pin connections between an Arria II device and an external memory device, refer to *Section I. Device and Pin Planning* in volume 2 of the *External Memory Interface Handbook*.

Memory clock pins in Arria II devices are generated with a DDIO register going to differential output pins (refer to Figure 7–3), marked in the pin table with DIFFIN or DIFFIO_RX prefixes (Arria II GX devices) and DIFFOUT, DIFFIO_TX, or DIFFIO_RX prefixes (Arria II GZ devices). These pins support the differential output function and you can use them as memory clock pins.

Figure 7–3. Memory Clock Generation for Arria II Devices *(Note 1)*



Notes to Figure 7–3:

(1) Global or regional clock networks are required for memory output clock generation to minimize jitter.

(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 with 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 VREF pins.

Arria II devices offer differential input buffers for differential read-data strobe and clock operations. In addition, Arria II devices also provide an independent DQS logic block for each CQn pin for complementary read-data strobe and clock operations. In the Arria II pin tables, the differential DQS pin pairs are denoted as DQS and DQSn pins, and 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 IOE registers.

Use differential DQS signaling for DDR2 SDRAM interfaces running at 333 MHz.

DQ pins can be bidirectional signals, as in DDR3, DDR2, and DDR SDRAM, and RLDRAM II common I/O (CIO) interfaces or unidirectional signals, as in QDR II+, QDR II SRAM, and RLDRAM II separate I/O (SIO) devices. Connect the unidirectional read-data signals to Arria II DQ pins and the unidirectional write-data signals to a different DQ/DQS group than the read DQ/DQS group. The write clocks must be assigned to the DQS/DQSn pins associated to this write DQ/DQS group. Do not use the CQ/CQn pin-pair for write clocks.

Using a DQ/DQS group for the write-data signals minimizes output skew and allows vertical migration. Arria II GX devices do not support vertical migration with Arria II GZ devices.

The DQ and DQS pin locations are fixed in the pin table. Memory interface circuitry is available in every Arria II I/O bank that does not support transceivers. All memory interface pins support the I/O standards required to support DDR3, DDR2, DDR SDRAM, QDR II+ and QDR II SRAM, and RLDRAM II devices.

Arria II devices support DQ and DQS signals with DQ bus modes of ×4, ×8/×9, ×16/×18, or ×32/×36, although not all devices support DQS bus mode in ×32/×36. The DDR, DDR2, and DDR3 SDRAM interfaces use one DQS pin for each ×8 group; for example, an interface with a ×72 wide interface requires nine DQS pins. When any of these pins are not used for memory interfacing, you can use these pins as user I/Os. Additionally, you can use any DQSn or CQn pins not used for clocking as DQ (data) pins.

Table 7–1 lists pin support per DQ/DQS bus mode, including the DQS/CQ and DQSn/CQn pin pair, for Arria II devices.

Table 7–1. DQ/DQS Bus Mode Pins for Arria II Devices

Mode	DQSn Support	CQn Support	Parity or DM (Optional)	QVLD (Optional) *(1)*	Typical Number of Data Pins per Group	Maximum Number of Data Pins per Group *(2)*
×4	Yes	No	No *(6)*	No	4	5
×8/×9 *(3)*	Yes	Yes	Yes	Yes	8 or 9	11
×16/×18 *(4)*	Yes	Yes	Yes	Yes	16 or 18	23
×32/×36 *(5)*	Yes	Yes	Yes	Yes	32 or 36	47
×32/×36 *(7)*	Yes	Yes	No *(8)*	Yes	32 or 36	39

Notes to Table 7–1:

(1) The QVLD pin is not used in the ALTMEMPHY megafunction and it is only applicable for Arria II GZ devices.

(2) This represents the maximum number of DQ pins (including parity, data mask, and QVLD 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 DQ/DQS group in a particular device. Check the pin table for the exact number per group. For DDR3, DDR2, and DDR interfaces, the number of pins is further reduced for an interface larger than ×8 due to the need of one DQS pin for each ×8/×9 group that is used to form the x16/×18 and ×32/×36 groups.

(3) Two ×4 DQ/DQS groups are stitched to make a ×8/×9 group so there are a total of 12 pins in this group.

(4) Four ×4 DQ/DQS groups are stitched to make a ×16/×18 group.

(5) Eight ×4 DQ/DQS groups are stitched to make a ×32/×36 group.

(6) The DM pin can be supported if differential DQS is not used and the group does not have additional signals.

(7) These ×32/×36 DQ/DQS groups are available in EP2AGZ300 and EP2AGZ350 devices in 1152- and 1517-pin FineLine BGA packages. There are 40 pins in each of these DQ/DQS groups.

(8) There are 40 pins in each of these DQ/DQS groups. You cannot place the BWSn pins within the same DQ/DQS group as the write data pins because of insufficient pins availability.

Table 7–2 lists the number of I/O modules and DQ/DQS groups per side of the Arria II GX device. For a more detailed listing of the number of DQ/DQS groups available per bank in each Arria II GX device, refer to Figure 7–4 on page 7–7 through Figure 7–10 on page 7–13. These figures represent the die top view of the Arria II GX device.

For more information about DQ/DQS groups pin-out restriction format, refer to the *Arria II Device Family Pin Connection Guidelines*.

Table 7–2. Number of DQ/DQS Groups and I/O Modules per Side in Arria II GX Devices

Device	Package	Side	Number of I/O Module *(1)*	Number of DQ/DQS Groups				Refer to
				×4	×8/×9	×16/×18	×32/×36	
EP2AGX45 EP2AGX65	358-Pin Ultra FineLine BGA	Top/Bottom	3	6	3	1	0	Figure 7–4 on page 7–7
		Right	2	4	2	0	0	
EP2AGX45 EP2AGX65	572-Pin FineLine BGA	Top/Bottom	4	8	4	2	0	Figure 7–5 on page 7–8
EP2AGX95 EP2AGX125		Right	6	12	6	2	0	Figure 7–6 on page 7–9
EP2AGX45 EP2AGX65	780-Pin FineLine BGA	Top/Bottom/ Right	7	14	7	3	1	Figure 7–7 on page 7–10
EP2AGX95 EP2AGX125 EP2AGX190 EP2AGX260								Figure 7–8 on page 7–11
EP2AGX95 EP2AGX125	1152-Pin FineLine BGA	Top/Bottom	9	18	9	4	2	Figure 7–9 on page 7–12
		Right	8	16	8	4	2	
EP2AGX190 EP2AGX260	1152-Pin FineLine BGA	Top/Bottom/ Right	12	24	12	6	2	Figure 7–10 on page 7–13

Note to Table 7–2:

(1) Each I/O module consists of 16 I/O pins. 12 of the 16 pins are DQ/DQS pins.

Table 7–3 lists the number of DQ/DQS groups available per side in each Arria II GZ device. For a more detailed listing of the number of DQ/DQS groups available per bank in each Arria II GZ device, refer to Figure 7–11 through Figure 7–15. These figures represent the die top view of the Arria II GZ device.

Table 7–3. Number of DQ/DQS Groups per Side in Arria II GZ Devices (Part 1 of 2)

Device	Package	Side	Number of DQ/DQS Groups				Refer to
			×4 *(1)*	×8/×9	×16/×18	×32/×36 *(2)*	
EP2AGZ300 EP2AGZ350	780-pin FineLine BGA	Left/Right	0	0	0	0	Figure 7–11 on page 7–14
		Top/Bottom	18	8	2	0	
EPAGZ225	1152-pin FineLine BGA	Left/Right	13	6	2	0	Figure 7–12 on page 7–15
		Top/Bottom	26	12	4	0	
EP2AGZ300 EP2AGZ350	1152-pin FineLine BGA	Left/Right	13	6	2	0	Figure 7–13 on page 7–16
		Top/Bottom	26	12	4	2 *(3)*	

Table 7–3. Number of DQ/DQS Groups per Side in Arria II GZ Devices (Part 2 of 2)

Device	Package	Side	Number of DQ/DQS Groups				Refer to
			×4 *(1)*	×8/×9	×16/×18	×32/×36 *(2)*	
EP2AGZ225	1517-pin FineLine BGA	All sides	26	12	4	0	Figure 7–14 on page 7–17
EP2AGZ300 EP2AGZ350	1517-pin FineLine BGA	Left/Right	26	12	4	0	Figure 7–15 on page 7–18
		Top/Bottom	26	12	4	2 *(3)*	

Notes to Table 7–3:

(1) Some of the ×4 groups may use R_{UP} and R_{DN} pins. You cannot use these groups if you use the Arria II GZ calibrated OCT feature.

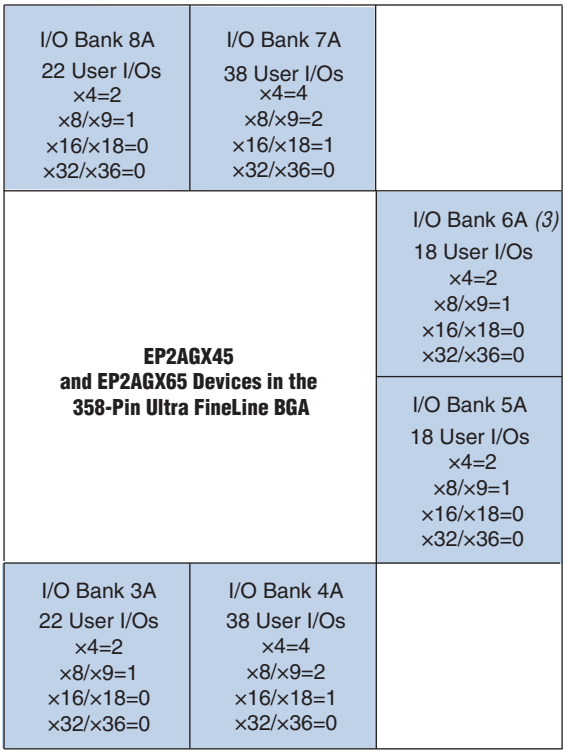
(2) To interface with a ×36 QDR II+/QDR II SRAM device in a Arria II GZ FPGA that does not support the ×32/×36 DQ/DQS group, refer to "Combining ×16/×18 DQ/DQS Groups for ×36 QDR II+/QDR II SRAM Interface" on page 7–21.

(3) These ×32/×36 DQ/DQS groups have 40 pins instead of 48 pins per group. You cannot place BWSn pins within the same DQ/DQS group as the write data pins because of insufficient pins available.

Figure 7–4 through Figure 7–10 show the maximum number of DQ/DQS groups per side of the Arria II GX device. These figures represent the die-top view of the Arria II GX device.

Figure 7–4 shows the number of DQ/DQS groups per bank in EP2AGX45 and EP2AGX65 devices in the 358-pin Ultra FineLine BGA (UBGA) package.

Figure 7–4. Number of DQ/DQS Groups per Bank in EP2AGX45 and EP2AGX65 Devices in the 358-Pin Ultra Fineline BGA Package *(Note 1)*, *(2)*



Notes to Figure 7–4:

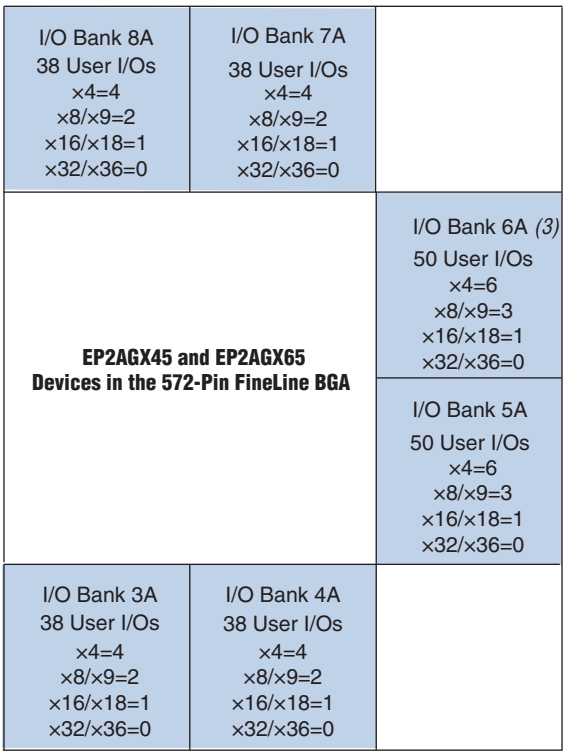
(1) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(2) Arria II GX devices in the 358-pin UBGA package do not support the ×36 QDR II+/QDR II SRAM interface.

(3) Several configuration pins in Bank 6A are shared with DQ/DQS pins. You cannot use a ×4 DQ/DQS group with any of their pin members used for configuration purposes. Ensure that the DQ/DQS groups you chose are not also used for configuration.

Figure 7–5 shows the number of DQ/DQS groups per bank in Arria II GX EP2AGX45 and EP2AGX65 devices in the 572-pin FineLine BGA package.

Figure 7–5. Number of DQ/DQS Groups per Bank in EP2AGX45 and EP2AGX65 Devices in the 572-Pin FineLine BGA Package ***(Note 1)***, ***(2)***



Notes to Figure 7–5:

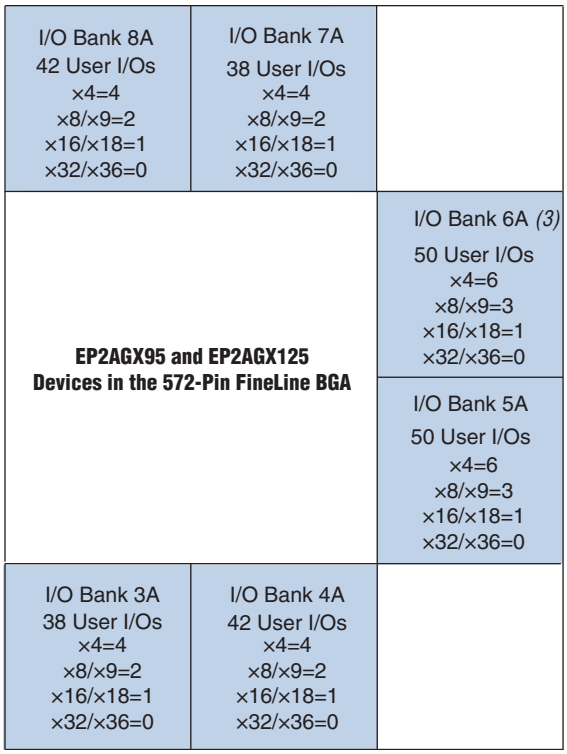
(1) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(2) Arria II GX devices in the 572-pin FineLine BGA Package do not support the ×36 QDR II+/QDR II SRAM interface.

(3) Several configuration pins in Bank 6A are shared with DQ/DQS pins. You cannot use a ×4 DQ/DQS group with any of their pin members used for configuration purposes. Ensure that the DQ/DQS groups you chose are not also used for configuration.

Figure 7–6 shows the number of DQ/DQS groups per bank in Arria II GX EP2AGX95 and EP2AGX125 devices in the 572-pin FineLine BGA package.

Figure 7–6. Number of DQ/DQS Groups per Bank in EP2AGX95 and EP2AGX125 Devices in the 572-Pin FineLine BGA Package ***(Note 1)***, ***(2)***



Notes to Figure 7–6:

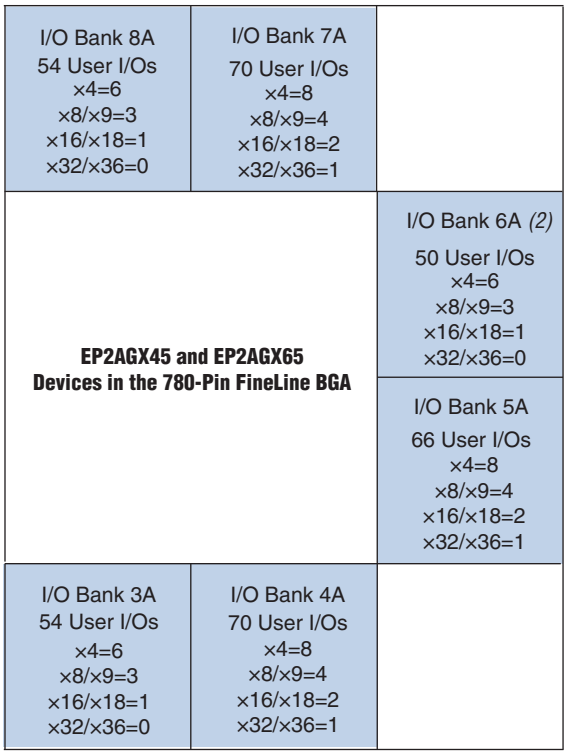
(1) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(2) Arria II GX devices in the 572-pin FineLine BGA Package do not support the ×36 QDR II+/QDR II SRAM interface.

(3) Several configuration pins in Bank 6A are shared with DQ/DQS pins. You cannot use a ×4 DQ/DQS group with any of their pin members used for configuration purposes. Ensure that the DQ/DQS groups you chose are not also used for configuration.

Figure 7–7 shows the number of DQ/DQS groups per bank in Arria II GX EP2AGX45 and EP2AGX65 devices in the 780-pin FineLine BGA package.

Figure 7–7. Number of DQ/DQS Groups per Bank in EP2AGX45 and EP2AGX65 Devices in the 780-Pin FineLine BGA Package ***(Note 1)***



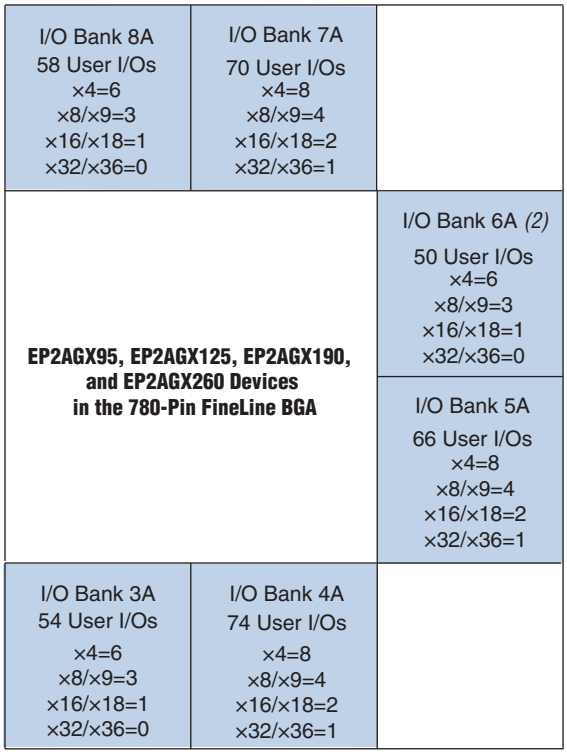
Notes to Figure 7–7:

(1) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(2) Several configuration pins in Bank 6A are shared with DQ/DQS pins. You cannot use a ×4 DQ/DQS group with any of their pin members used for configuration purposes. Ensure that the DQ/DQS groups you chose are not also used for configuration.

Figure 7–8 shows the number of DQ/DQS groups per bank in Arria II GX EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260 devices in the 780-pin FineLine BGA package.

Figure 7–8. Number of DQ/DQS Groups per Bank in EP2AGX95, EP2AGX125, EP2AGX190 and EP2AGX260 Devices in the 780-Pin FineLine BGA Package ***(Note 1)***



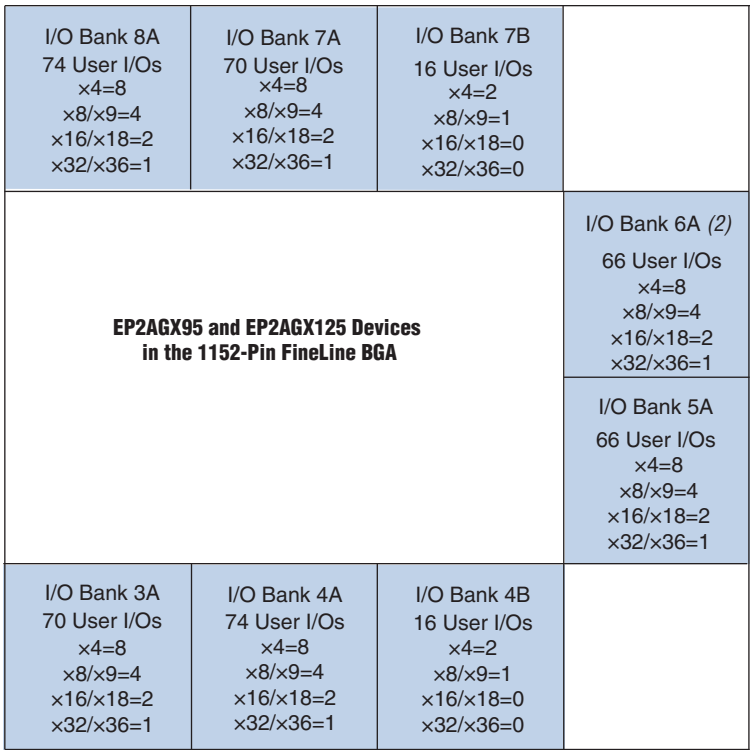
Notes to Figure 7–8:

(1) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(2) Several configuration pins in Bank 6A are shared with DQ/DQS pins. You cannot use a ×4 DQ/DQS group with any of their pin members used for configuration purposes. Ensure that the DQ/DQS groups you chose are not also used for configuration.

Figure 7–9 shows the number of DQ/DQS groups per bank in Arria II GX EP2AGX95 and EP2AGX125 devices in the 1152-pin FineLine BGA package.

Figure 7–9. Number of DQ/DQS Groups per Bank in EP2AGX95 and EP2AGX125 Devices in the 1152-Pin FineLine BGA Package ***(Note 1)***



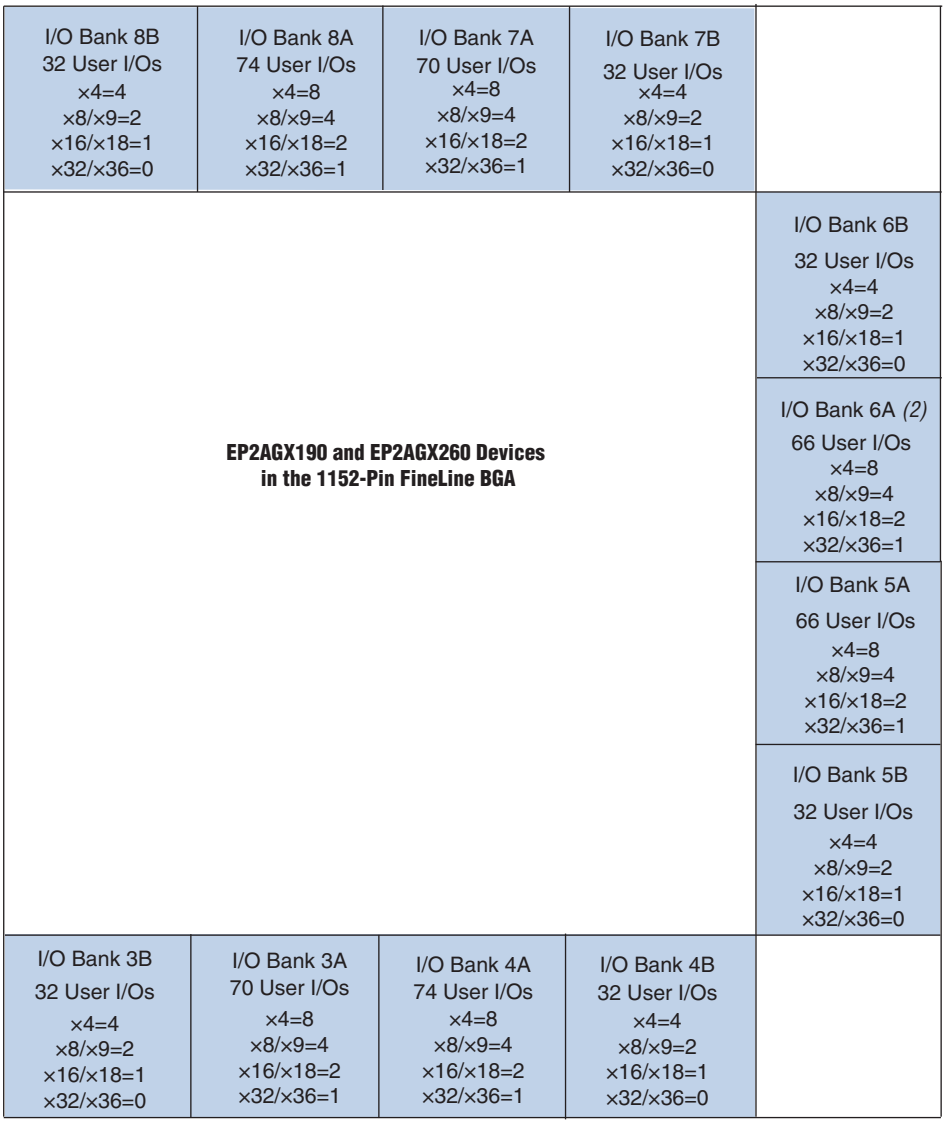
Notes to Figure 7–9:

(1) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(2) Several configuration pins in Bank 6A are shared with DQ/DQS pins. You cannot use a ×4 DQ/DQS group with any of their pin members used for configuration purposes. Ensure that the DQ/DQS groups you chose are not also used for configuration.

Figure 7–10 shows the number of DQ/DQS groups per bank in Arria II GX EP2AGX190 and EP2AGX260 devices in the 1152-pin FineLine BGA package.

Figure 7–10. Number of DQ/DQS Groups per Bank in EP2AGX190 and EP2AGX260 Devices in the 1152-Pin FineLine BGA Package ***(Note 1)***



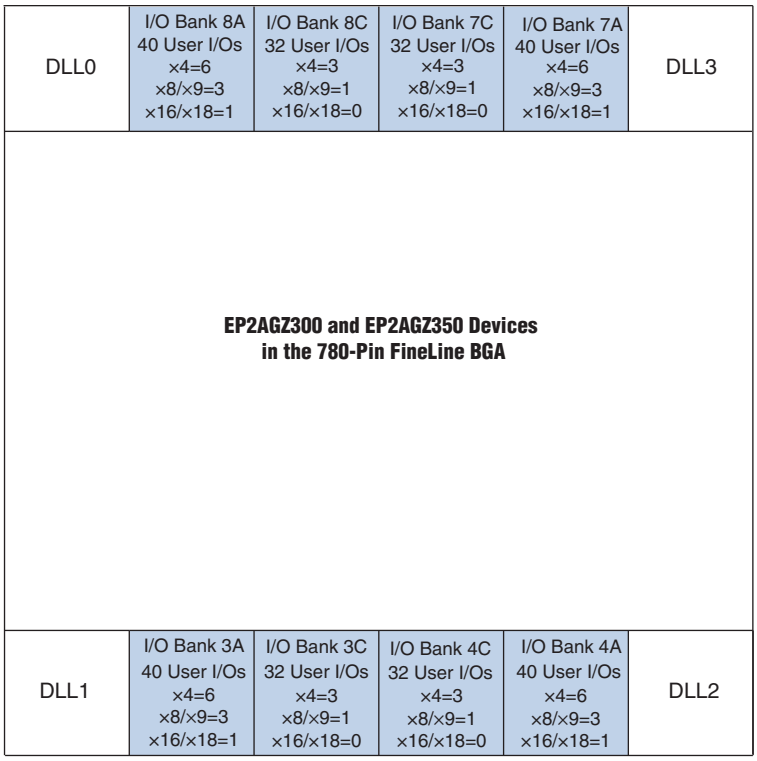
Notes to Figure 7–10:

(1) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(2) Several configuration pins in Bank 6A are shared with DQ/DQS pins. You cannot use a ×4 DQ/DQS group with any of their pin members used for configuration purposes. Ensure that the DQ/DQS groups you chose are not also used for configuration.

Figure 7–11 shows the number of DQ/DQS groups per bank in Arria II GZ EP2AGZ300 and EP2AGZ350 devices in the 780-pin FineLine BGA package.

Figure 7–11. Number of DQ/DQS Groups per Bank in EP2AGZ300 and EP2AGZ350 Devices in the 780-Pin FineLine BGA Package, *(Note 1)*

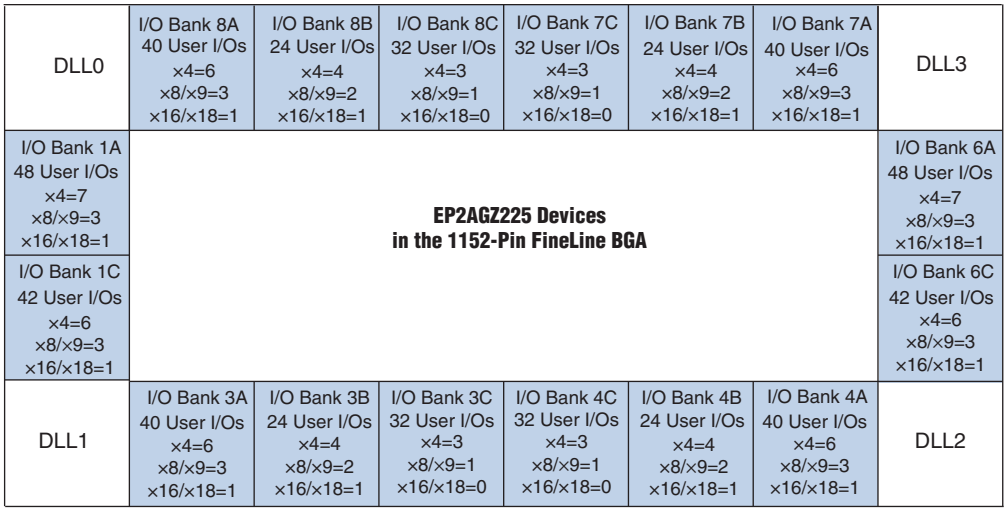


Note to Figure 7–11:

(1) EP2AGZ300 and EP2AGZ350 devices do not support ×32/×36 mode. To interface with a ×36 QDR II+/QDR II SRAM device, refer to "Combining ×16/×18 DQ/DQS Groups for ×36 QDR II+/QDR II SRAM Interface" on page 7–21.

Figure 7–12 shows the number of DQ/DQS groups per bank in Arria II GZ EP2AGZ225 devices in the 1152-pin FineLine BGA package.

Figure 7–12. Number of DQ/DQS Groups per Bank in EP2AGZ225 Devices in the 1152-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*



Notes to Figure 7–12:

(1) EP2AGZ225 devices do not support the ×32/×36 mode. To interface with a ×36 QDR II+/QDR II SRAM device, refer to "Combining ×16/×18 DQ/DQS Groups for ×36 QDR II+/QDR II SRAM Interface" on page 7–21.

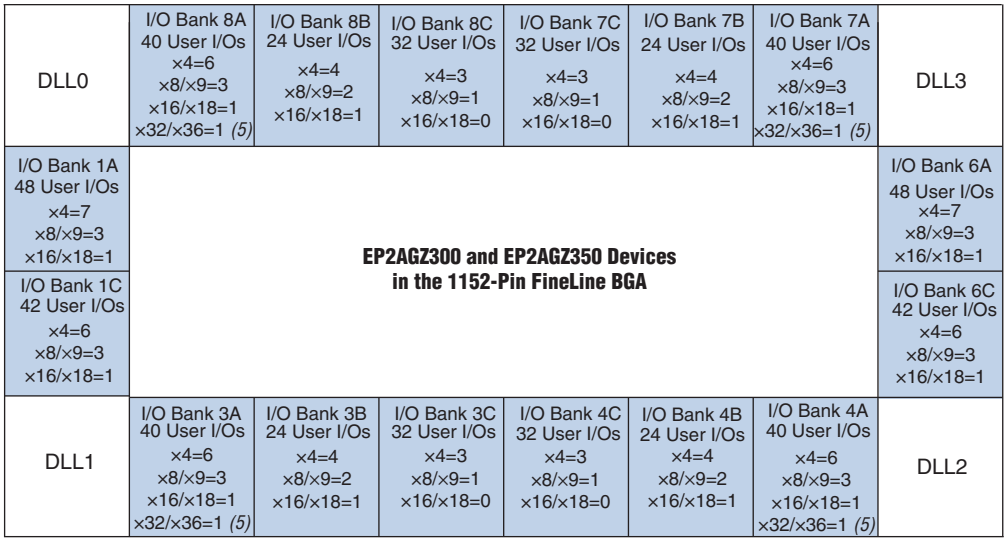
(2) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group; however, there are restrictions on using ×8/×9 groups that include that ×4 group.

(3) All I/O pin counts include dedicated clock inputs that you can use for data inputs.

(4) You can also use some of the DQ/DQS pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQ/DQS group with any of its pin members used for configuration purposes. Ensure that the DQ/DQS groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQ/DQS groups, depending on your configuration scheme.

Figure 7–13 shows the number of DQ/DQS groups per bank in Arria II GZ EP2AGZ300 and EP2AGZ350 devices in the 1152-pin FineLine BGA package.

Figure 7–13. Number of DQ/DQS Groups per Bank in EP2AGZ300 and EP2AGZ350 Devices in the 1152-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*



Notes to Figure 7–13:

(1) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group; however, there are restrictions on using ×8/×9 groups that include that ×4 group.

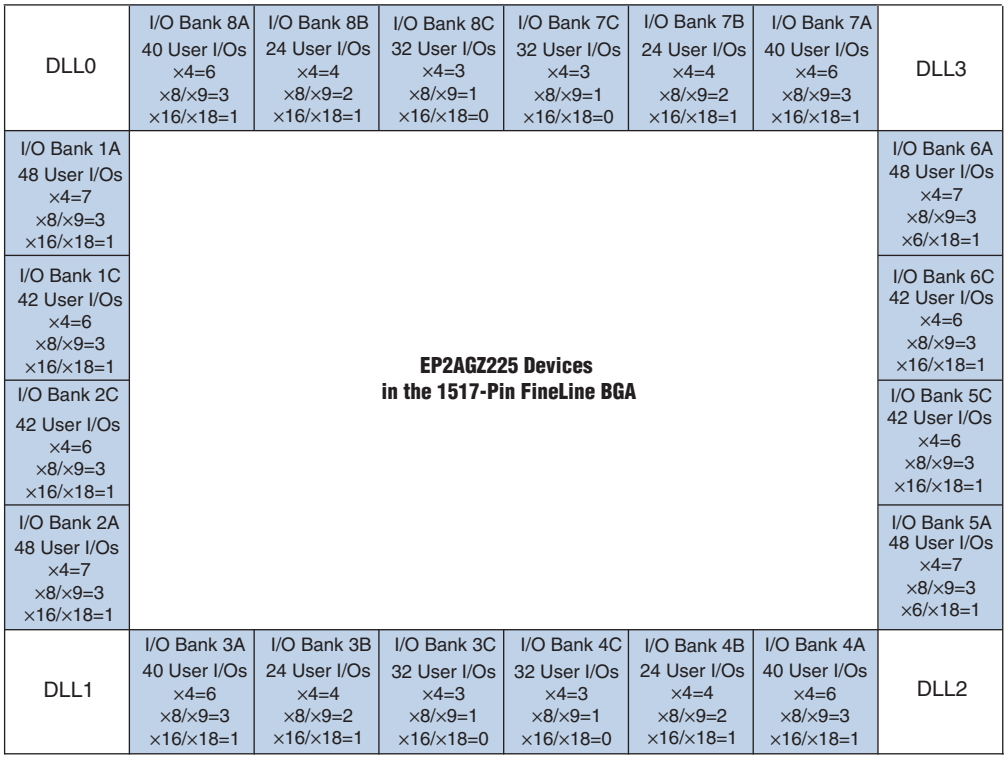
(2) All I/O pin counts include dedicated clock inputs that you can use for data inputs.

(3) You can also use some of the DQ/DQS pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQ/DQS group with any of its pin members used for configuration purposes. Ensure that the DQ/DQS groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQ/DQS groups, depending on your configuration scheme.

(4) These ×32/×36 DQ/DQS groups have 40 pins instead of 48 pins per group.

Figure 7–14 shows the number of DQ/DQS groups per bank in Arria II GZ EP2AGZ225 devices in the 1517-pin FineLine BGA package.

Figure 7–14. Number of DQ/DQS Groups per Bank in EP2AGZ225 Devices in the 1517-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*



Notes to Figure 7–14:

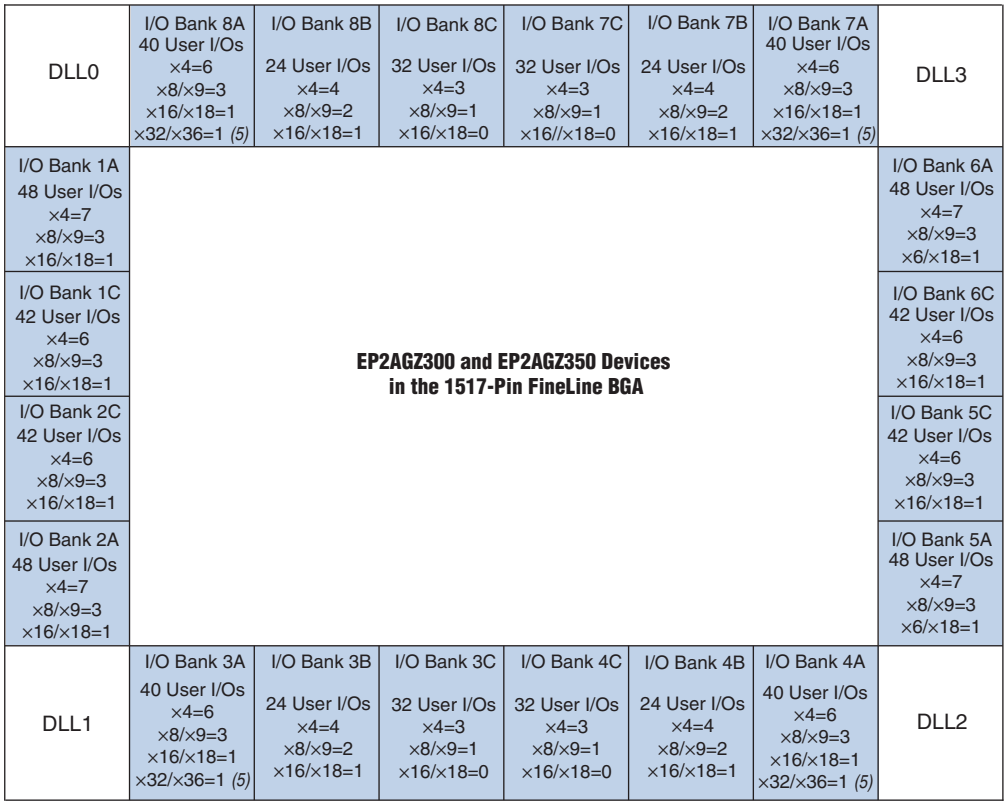
(1) EP2AGZ225 devices do not support ×32/×36 mode. To interface with a ×36 QDR II+/QDR II SRAM device, refer to "Combining ×16/×18 DQ/DQS Groups for ×36 QDR II+/QDR II SRAM Interface" on page 7–21.

(2) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group.

(3) All I/O pin counts include dedicated clock inputs that you can use for data inputs.

(4) You can also use some of the DQ/DQS pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQ/DQS group with any of its pin members used for configuration purposes. Ensure that the DQ/DQS groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQ/DQS groups, depending on your configuration scheme.

Figure 7–15. Number of DQ/DQS Groups per Bank in EP2AGZ300 and EP2AGZ350 Devices in the 1517-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*



Notes to Figure 7–15:

(1) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group.

(2) All I/O pin counts include dedicated clock inputs that you can use for data inputs.

(3) You can also use some of the DQ/DQS pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQ/DQS group with any of its pin members used for configuration purposes. Ensure that the DQ/DQS groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQ/DQS groups, depending on your configuration scheme.

(4) These ×32/×36 DQ/DQS groups have 40 pins instead of 48 pins per group.

The DQS and DQSn pins are listed in the Arria II pin tables as `DQSXY` and `DQSnXY`, respectively, where `X` denotes the DQ/DQS grouping number and `Y` denotes whether the group is located on the top (T), bottom (B), left (L), or right (R) side of the device. The DQ/DQS pin numbering is based on ×4 mode.

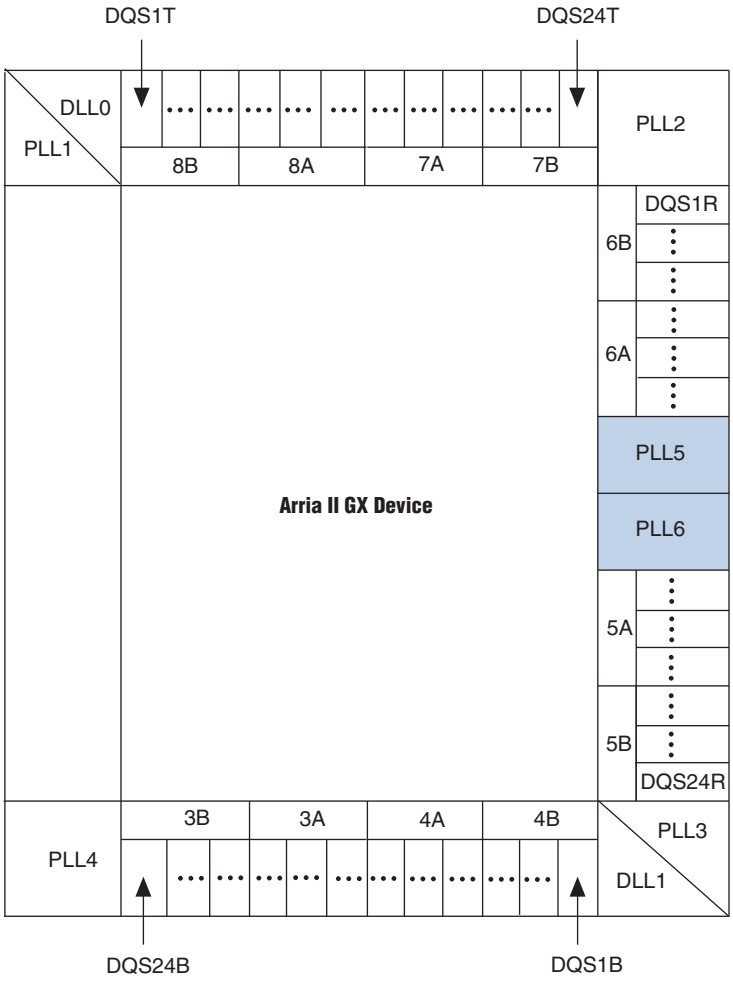
The corresponding DQ pins are marked as `DQXY`, where `X` indicates which DQS group the pins belong to and `Y` indicates whether the group is located on the top (T), bottom (B), left (L), or right (R) side of the device. For example, `DQS3B` indicates a DQS pin that is located on the bottom side of the device. The DQ pins belonging to that group are shown as `DQ3B` in the pin table. For DQS pins in Arria II GX I/O banks, refer to Figure 7–16. For DQS pins in Arria II GZ I/O banks, refer to Figure 7–17.



The parity, DM, BWSn, NWSn, QVLD, and ECC pins are shown as DQ pins in the pin table.

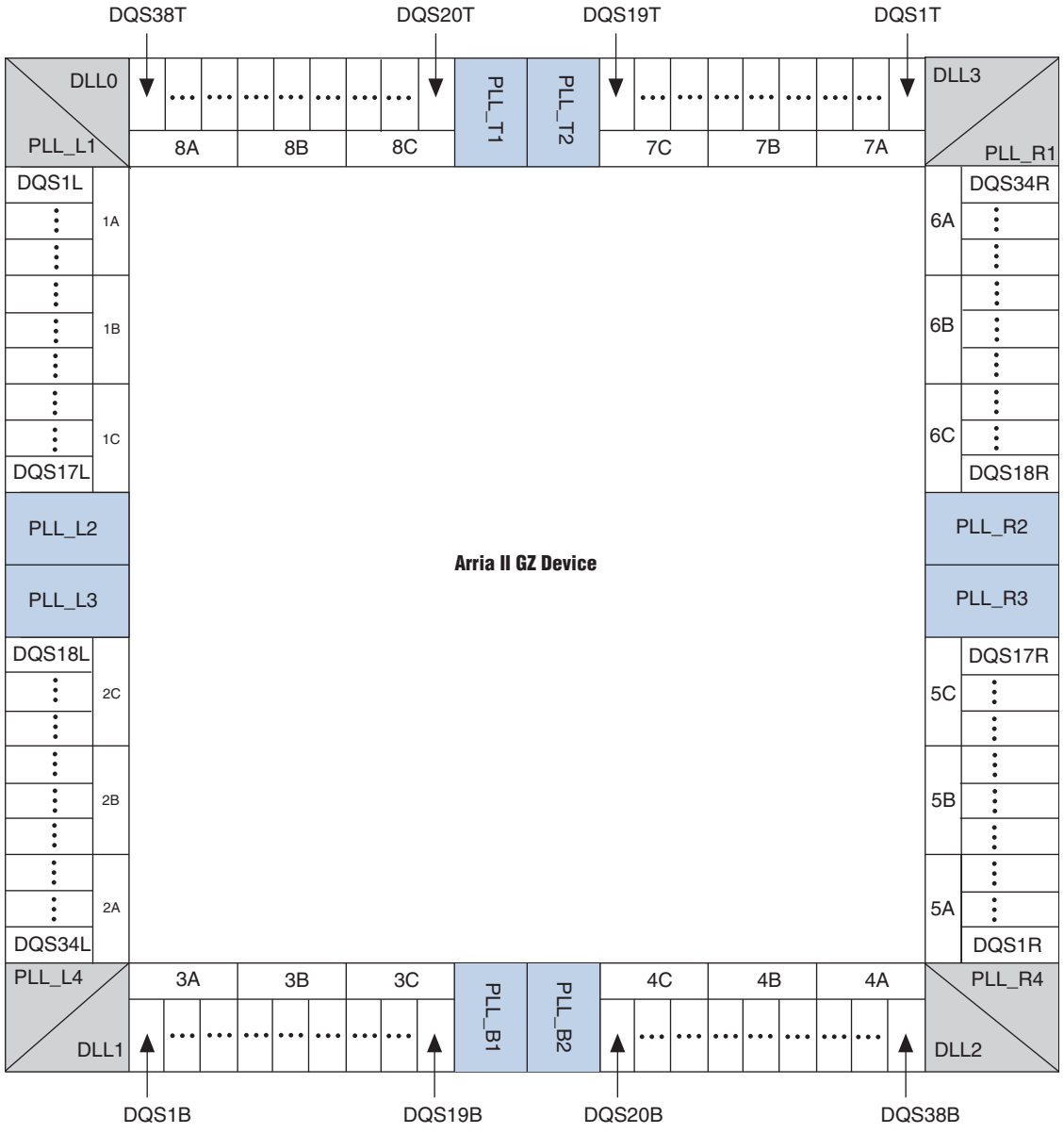
The numbering scheme starts from the top-left side of the device going clockwise in a die top view. Figure 7–16 shows how the DQ/DQS groups are numbered in a die top view of the largest Arria II GX device.

Figure 7–16. DQS Pins in Arria II GX I/O Banks



The numbering scheme starts from the top-left corner of the device going counter-clockwise in a die top view. Figure 7–17 shows how the DQ/DQS groups are numbered in a die top view of the device.

Figure 7–17. DQS Pins in Arria II GZ I/O Banks



Using the R_{UP} and R_{DN} Pins in a DQ/DQS Group Used for Memory Interfaces in Arria II GZ Devices

You can use the DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins (listed in the pin table). You cannot use a ×4 DQ/DQS group for memory interfaces if any of its pin members are used as R_{UP} and R_{DN} pins for OCT calibration. You may be able to use the ×8/×9 group that includes this ×4 DQ/DQS group, if either of the following applies:

- You are not using DM pins with your differential DQS pins
- You are not using complementary or differential DQS pins

You can use the ×8/×9 group because a DQ/DQS ×8/×9 group actually comprises 12 pins, because the groups are formed by stitching two DQ/DQS groups in ×4 mode with six pins each (refer to Table 7–1 on page 7–5). A typical ×8 memory interface consists of one DQS, one DM, and eight DQ pins that add up to 10 pins. If you choose your pin assignment carefully, you can use the two extra pins for R_{UP} and R_{DN}. In a DDR3 SDRAM interface, you must use differential DQS, which means that you only have one extra pin. In this case, pick different pin locations for the R_{UP} and R_{DN} pins (for example, in the bank that contains the address and command pins).

You cannot use the R_{UP} and R_{DN} pins shared with DQ/DQS group pins when using ×9 QDR II+/QDR II SRAM devices, because the R_{UP} and R_{DN} pins are dual purpose with the CQn pins. In this case, pick different pin locations for R_{UP} and R_{DN} pins to avoid conflict with memory interface pin placement. You have the choice of placing the R_{UP} and R_{DN} pins in the data-write group or in the same bank as the address and command pins.

There is no restriction on using ×16/×18 or ×32/×36 DQ/DQS groups that include the ×4 groups whose pins are being used as R_{UP} and R_{DN} pins, because there are enough extra pins that can be used as DQS pins.

☞ For ×8, ×16/×18, or ×32/×36 DQ/DQS groups whose members are used for R_{UP} and R_{DN}, you must assign DQS and DQ pins manually. The Quartus® II software might not be able to place DQS and DQ pins without manual pin assignments, resulting in a "no-fit".

Combining ×16/×18 DQ/DQS Groups for ×36 QDR II+/QDR II SRAM Interface

This implementation combines ×16/×18 DQ/DQS groups to interface with a ×36 QDR II+/QDR II SRAM device. The ×36 read data bus uses two ×16/×18 groups, and the ×36 write data uses another two ×16/×18 or four ×8/×9 groups. The CQ/CQn signal traces are split on the board trace to connect to two pairs of CQ/CQn pins in the FPGA. This is the only connection on the board that you must change for this implementation. Other QDR II+/QDR II SRAM interface rules for Arria II devices also apply for this implementation.

☞ The ALTMEMPHY megafunction and UniPHY IP core do not use the QVLD signal, so you can leave the QVLD signal unconnected as in any QDR II+/QDR II SRAM interfaces in Arria II devices.

For more information about the ALTMEMPHY megafunction and UniPHY IP core, refer to the *External Memory Interface Handbook*.

Use one side of the device with the ×36 mode emulation interface whenever possible, even though the ×36 group formed by a combination of DQ/DQS groups from the top and bottom I/O banks, or top/bottom I/O bank and left/right I/O banks is supported.

Rules to Combine Groups

In 572-, 780-, 1152-, and some 1517-pin package devices, there is at most one ×16/×18 group per I/O bank. You can combine two ×16/×18 groups from a single side of the device for a ×36 interface. 358-pin package devices have only one ×16/×18 group in each bank 4A and 7A. You can only form a ×36 interface with these two banks.

For devices that do not have four ×16/×18 groups in a single side of the device to form two ×36 groups for read and write data, you can form one ×36 group on one side of the device and another ×36 group on the other side of the device. Altera recommends forming two ×36 groups on column I/O banks (top and bottom) only, although forming a ×36 group from column I/O banks and another ×36 group from row I/O banks for the read and write data buses is supported. For vertical migration with the ×36 emulation implementation, you must check if migration is possible by enabling device migration in the Quartus II project. The Quartus II software also supports the use of four ×8/×9 DQ groups for write data pins and the migration of these groups across device density. 358-pin package devices can only form a ×36 group for write data pin with four ×8/×9 groups.

Table 7–4 lists the possible combinations to use two ×16/×18 DQ/DQS groups to form a ×32/×36 group on Arria II devices lacking a native ×32/×36 DQ/DQS group.

Table 7–4. Possible Group Combinations in Arria II Devices

Device	Package	Device Density	I/O Bank Combinations
Arria II GX	358-Pin Ultra FineLine BGA	■ EP2AGX45 ■ EP2AGX65	4A and 7A (Top and Bottom I/O banks) *(1)*
	572-Pin FineLine BGA	■ EP2AGX45 ■ EP2AGX65 ■ EP2AGX95 ■ EP2AGX125	7A and 8A (Top I/O banks) 5A and 6A (Right I/O banks) 3A and 4A (Bottom I/O banks)
	780-Pin FineLine BGA *(2)*	■ EP2AGX45 ■ EP2AGX65 ■ EP2AGX95 ■ EP2AGX125 ■ EP2AGX190 ■ EP2AGX260	7A and 8A (Top I/O banks) 5A and 6A (Right I/O banks) 3A and 4A (Bottom I/O banks)
	1152-Pin FineLine BGA *(2)*	■ EP2AGX95 ■ EP2AGX125	7A and 8A (Top I/O banks) 5A and 6A (Right I/O banks) 3A and 4A (Bottom I/O banks)
		■ EP2AGX190 ■ EP2AGX260	Combine any two banks from each side of I/O banks
Arria II GZ	780-Pin FineLine BGA	■ EP2AGZ300 ■ EP2AGZ350	3A and 4A, 7A and 8A (bottom and top I/O banks) *(3)*
	1152-Pin FineLine BGA	■ EP2AGZ225 ■ EP2AGZ300 *(4)* ■ EP2AGZ350 *(4)*	1A and 1C, 6A and 6C (left and right I/O banks) 3A and 3B, 4A and 4B (bottom I/O banks) 7A and 7B, 8A and 8B (top I/O banks)
	1517-Pin FineLine BGA	■ EP2AGZ225 ■ EP2AGZ300 *(4)* ■ EP2AGZ350 *(4)*	1A and 1C, 2A and 2C (left I/O banks) 3A and 3B, 4A and 4B (bottom I/O banks) 5A and 5C, 6A and 6C (right I/O banks) 7A and 7B, 8A and 8B (top I/O banks)

Notes to Table 7–4:

(1) Only one ×8/×9 group left in each of the remaining I/O banks. You can form only ×36 group write data with four ×8/×9 groups in these packages.

(2) This device supports ×36 DQ/DQS groups on each side of I/O banks.

(3) Each side of the device in these packages has four remaining ×8/×9 groups. You can combine them for the write side (only) if you want to keep the ×36 QDR II+/QDR II SRAM interface on one side of the device. In this case, you must change the **Memory Interface Data Group** default assignment from the default **18** to **9**.

(4) This device supports ×36 DQ/DQS groups on the top and bottom I/O banks natively.

Arria II External Memory Interface Features

Arria II devices are rich with features that allow robust high-performance external memory interfacing. The Altera® Memory IPs allow 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 devices feature that is used in external memory interfaces from the DQS phase-shift circuitry, dynamic OCT control block, and DQS logic block.



If you use the Altera memory controller MegaCore® functions, the ALTMEMPHY megafunction and UniPHY IP core are instantiated for you.



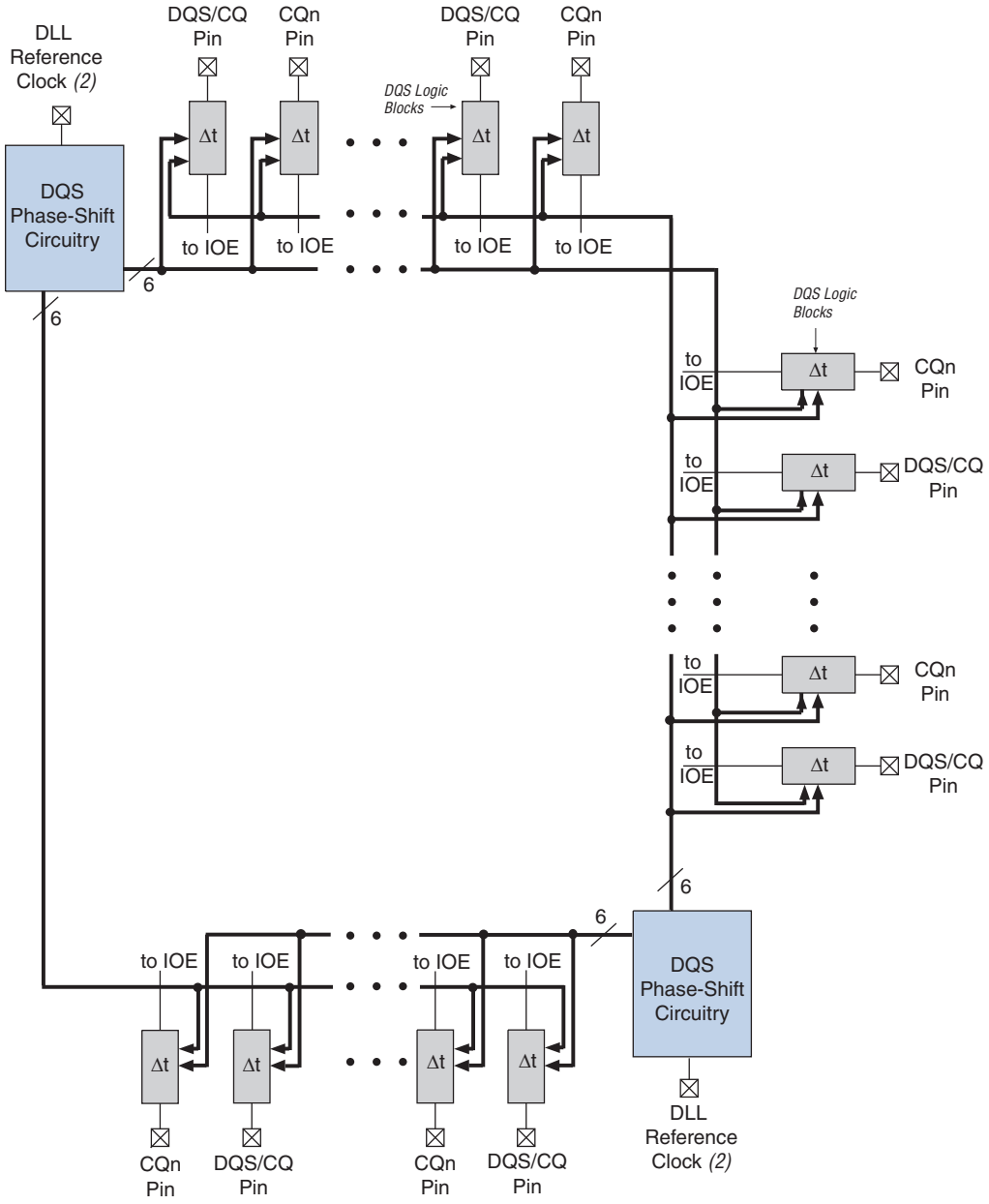
For more information about supported external memory IPs, refer to *Section III: External Memory Interface System Specification* in volume 1 of the *External Memory Handbook*.

DQS Phase-Shift Circuitry

Arria II 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 the multiple DQS pins and the phase-offset control module to further fine-tune the DQS phase shift for different sides of the device.

Figure 7–18 and Figure 7–19 show 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 and all sides of the Arria II GZ device.

Figure 7–18. DQS/CQ and CQn Pins and DQS Phase-Shift Circuitry for Arria II GX Devices ***(Note 1)***

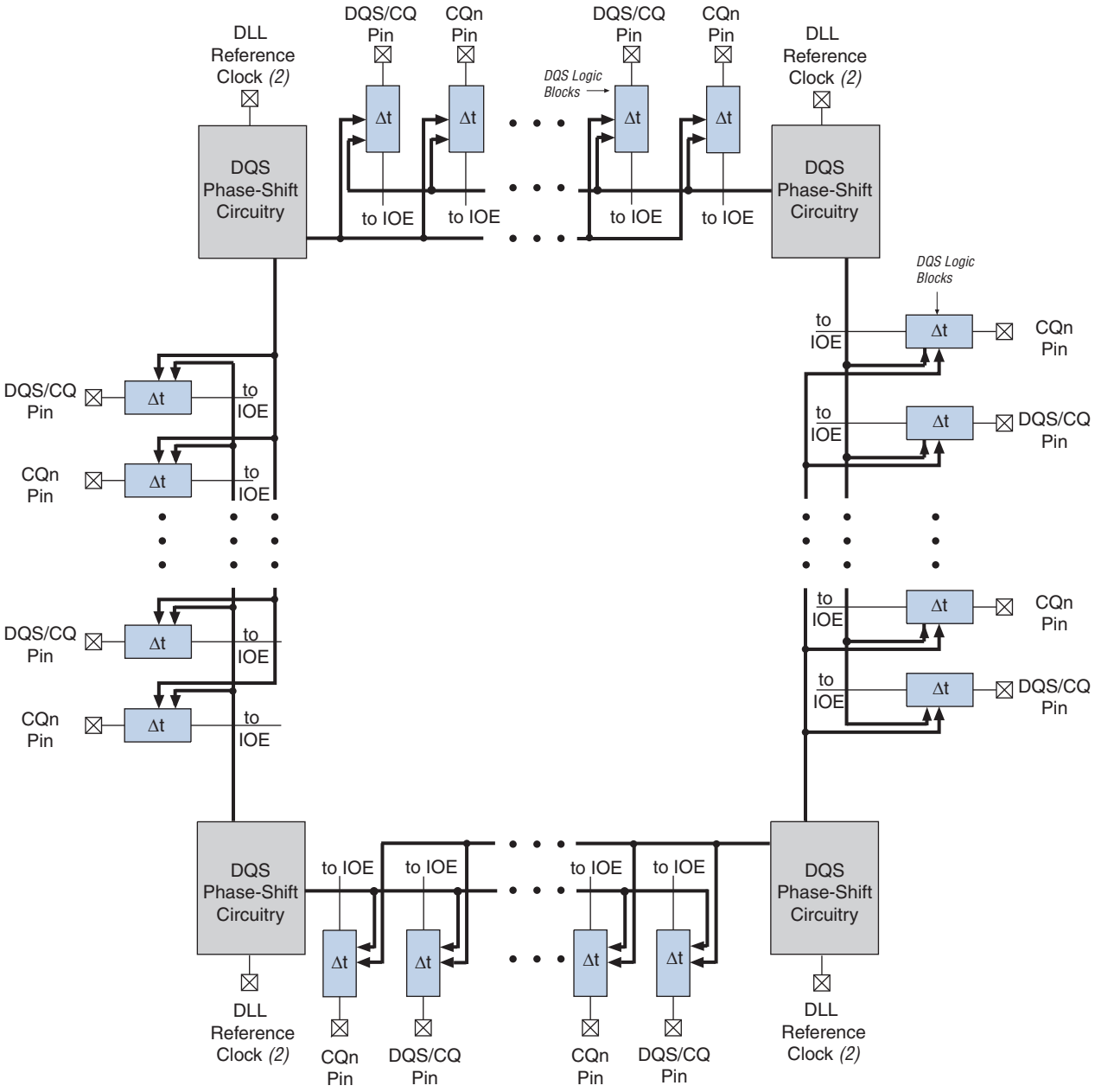


Notes to Figure 7–18:

(1) For possible reference input clock pins for each DLL, refer to "DLL" on page 7–27.

(2) You can configure each DQS/CQ and CQn pin with a phase shift based on one of two possible DLL output settings.

Figure 7–19. DQS/CQ and CQn Pins and DQS Phase-Shift Circuitry for Arria II GZ Devices ***(Note 1)***



Notes to Figure 7–19:

(1) For possible reference input clock pins for each DLL, refer to "DLL" on page 7–27.

(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 required 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 requires a maximum of 1,280 clock cycles to lock and calculate the correct input clock period when the DLL is in low jitter mode. Otherwise, only 256 clock cycles are required. Do not send data during these clock cycles because there is no guarantee that the data is properly captured. As the settings from the DLL may not be stable until this lock period has elapsed, be aware that anything with 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 must still exist for reliable operation.

There are two DLLs in an Arria II GX device and four DLLs in Arria II GZ device, located in the top-left and bottom-right corners of the Arria II GX device and each corner of the Arria II GZ device. These DLLs can support a maximum of two unique frequencies (Arria II GX devices) or four unique frequencies (Arria II GZ devices), 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 and Arria II GZ device to have eight different DLL phase shift settings.

For Arria II GX devices, 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.

For Arria II GZ devices, each DLL can access the two adjacent sides from its location within the device. For example, DLL0 on the top left of the device can access the top side (I/O banks 7A, 7B, 7C, 8A, 8B, and 8C) and the left side of the device (I/O banks 1A, 1B, 1C, 2A, 2B, and 2C). This means that each I/O bank is accessible by two DLLs, giving more flexibility to create multiple frequencies and multiple-type interfaces. You can have two different interfaces with the same frequency on the two sides adjacent to a DLL, where the DLL controls the DQS delay settings for both interfaces.



Interfaces that span across two sides of the device are not recommended for high-performance memory interface applications. However, Arria II GX devices support split interfaces (top and bottom I/O banks) and interfaces with multiple DQ/DQS groups wrapping over column and row I/Os from adjacent sides of the devices. Interfaces spanning “top and bottom I/O banks”, “right and bottom I/O banks”, or “top, bottom, and right I/O banks” are supported.

For Arria II GX devices, each bank can use settings from either one or both DLLs. For example, `DQS1R` can get its phase-shift settings from DLL0, and `DQS2R` can get its phase-shift settings from DLL1.

For Arria II GZ devices, each bank can use settings from either or both adjacent DLLs the bank. For example, DQS1L can get its phase-shift settings from DLL0, while DQS2L can get its phase-shift settings from DLL1.

If 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 automatically changes it. Because the PLL does not use any other outputs, it does not have to compensate for any clock paths.

Arria II devices support PLL cascading. If you cascade PLLs, you must use PLLs adjacent to each other (for example, PLL5 and PLL6 for Arria II GX devices) so that the dedicated path between the two PLLs is used instead of using a global clock (GCLK) or regional clock (RCLK) network that might be subjected to core noise. The TimeQuest Timing Analyzer takes PLL cascading into consideration for timing analysis.

Table 7–5 lists the DLL location and supported I/O banks for Arria II GZ devices.

Table 7–5. DLL Location and Supported I/O Banks for Arria II GZ Devices

DLL	Location	Accessible I/O Banks *(1)*
DLL0	Top-left corner	1A, 1B, 1C, 2A, 2B, 2C, 7A, 7B, 7C, 8A, 8B, 8C
DLL1	Bottom-left corner	1A, 1B, 1C, 2A, 2B, 2C, 3A, 3B, 3C, 4A, 4B, 4C
DLL2	Bottom-right corner	3A, 3B, 3C, 4A, 4B, 4C, 5A, 5B, 5C, 6A, 6B, 6C
DLL3	Top-right corner	5A, 5B, 5C, 6A, 6B, 6C, 7A, 7B, 7C, 8A, 8B, 8C

Note to Table 7–5:

(1) The DLL can access these I/O banks if they are available for memory interfacing.

Table 7–6 lists the reference clock for each DLL might come from PLL output clocks or dedicated clock input pins for Arria II GX devices.

Table 7–6. DLL Reference Clock Input for Arria II GX Devices ***(Note 1)***

DLL	CLKIN (Top/Bottom)	CLKIN (Right)	PLL
DLL0	CLK12 CLK13 CLK14 CLK15	—	PLL1
DLL1	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.

For Arria II GZ devices, the reference clock for each DLL may come from PLL output clocks or any of the two dedicated clock input pins located in either side of the DLL. Table 7–7 through Table 7–9 show the available DLL reference clock input resources for the Arria II GZ devices.

Table 7–7. DLL Reference Clock Input for EP2AGZ300 and EP2AGZ350 Devices in the 780-Pin FineLine BGA Package

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL0	CLK12P CLK13P CLK14P CLK15P	—	PLL_T1	—	—
DLL1	CLK4P CLK5P CLK6P CLK7P	—	PLL_B1	—	—
DLL2	CLK4P CLK5P CLK6P CLK7P	—	PLL_B2	—	—
DLL3	CLK12P CLK13P CLK14P CLK15P	—	PLL_T2	—	—

Table 7–8. DLL Reference Clock Input for EP2AGZ225, EP2AGZ300, and EP2AGZ350 Devices in the 1152-Pin FineLine BGA Package (Part 1 of 2)

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL0	CLK12P CLK13P CLK14P CLK15P	CLK0P CLK1P	PLL_T1	PLL_L2	—
DLL1	CLK4P CLK5P CLK6P CLK7P	CLK0P CLK1P	PLL_B1	—	—

Table 7–8. DLL Reference Clock Input for EP2AGZ225, EP2AGZ300, and EP2AGZ350 Devices in the 1152-Pin FineLine BGA Package (Part 2 of 2)

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL2	CLK4P CLK5P CLK6P CLK7P	CLK10P CLK11P	PLL_B2	—	—
DLL3	CLK12P CLK13P CLK14P CLK15P	CLK10P CLK11P	PLL_T2	PLL_R2	—

Table 7–9. DLL Reference Clock Input for EP2AGZ225, EP2AGZ300, and EP2AGZ350 Devices in the 1517-Pin FineLine BGA Package

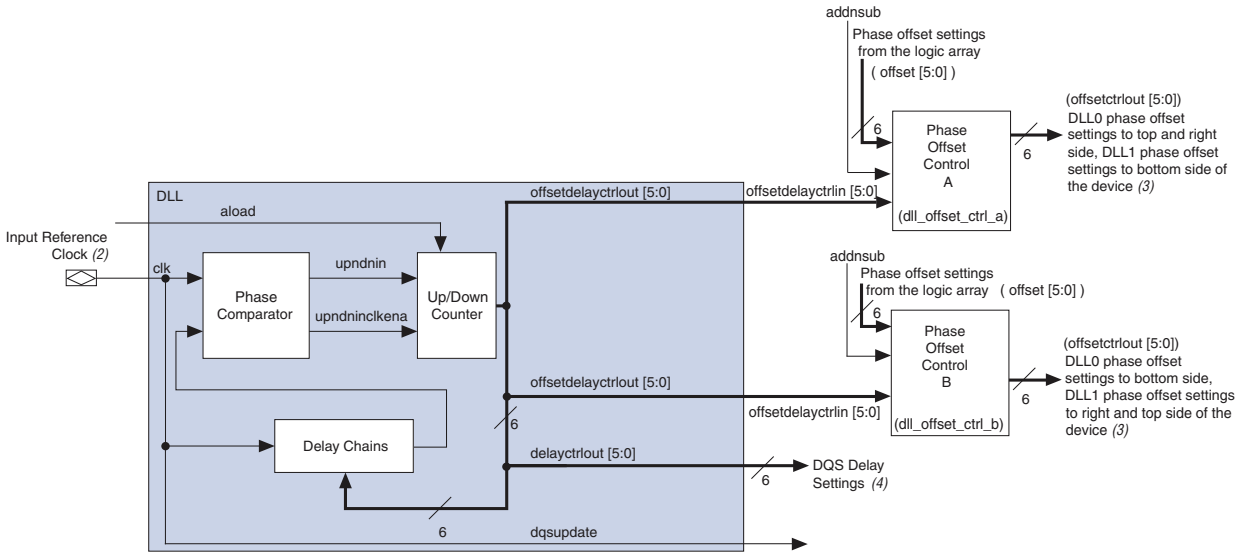
DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL0	CLK12P CLK13P CLK14P CLK15P	CLK0P CLK1P CLK2P CLK3P	PLL_T1	PLL_L2	—
DLL1	CLK4P CLK5P CLK6P CLK7P	CLK0P CLK1P CLK2P CLK3P	PLL_B1	PLL_L3	—
DLL2	CLK4P CLK5P CLK6P CLK7P	CLK8P CLK9P CLK10P CLK11P	PLL_B2	PLL_R3	—
DLL3	CLK12P CLK13P CLK14P CLK15P	CLK8P CLK9P CLK10P CLK11P	PLL_T2	PLL_R2	—



If you use the ALTMEMPHY megafunction or UniPHY IP core, Altera recommends using the dedicated PLL input pin for the PLL reference clock.

Figure 7–20 shows the DQS phase-shift circuitry for Arria II devices. 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 7–20. Simplified Diagram of the DQS Phase-Shift Circuitry for Arria II Devices ***(Note 1)***



Notes to Figure 7–20:

(1) All features of the DQS phase-shift circuitry are accessible from the UniPHY IP core and 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. For the exact PLL and input clock pin, refer to Table 7–6 and Table 7–10.

(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 1,280 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°, 180°, or 240°. 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°), 45° (up to 180°), or 60° (up to 240°).

There are seven different frequency modes for Arria II GX DLLs, and eight different frequency modes for Arria II GZ DLLs as shown in Table 7–10. 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, 5, 6, and 7 only 5 bits of the DQS delay settings vary with PVT to implement the phase-shift delay; the MSB of the DQS delay setting is set to **0**.

Table 7–10. DLL Frequency Modes for Arria II Devices

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
6	45, 90, 135, 180	8
7 *(1)*	60, 120, 180, 240	6

Note to Table 7–10:

(1) Frequency mode 7 is only available for Arria II GZ devices only.

For the frequency range of each mode, refer to the *Device Datasheet for Arria II Devices*.

For a 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 do not use 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 QDR II+/QDR II SRAM interfaces.

Phase Offset Control

Each DLL has two phase offset modules and can provide two separate DQS delay settings with independent offset; for Arria II GX devices, one offset goes clockwise half-way around the chip and the other goes counter-clockwise half-way around the chip and for Arria II GZ devices, one for the top and bottom I/O bank and one for the left and right I/O bank. Even though you have independent phase offset control, the frequency of the interface with the same DLL must 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 must have a 67.5° phase shift on the DQS signal, you can use two delay chains in the DQS logic blocks to give you a 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 the –64 to +63 settings for frequency mode 0, 1, 2, and 3, and between the –32 to +31 settings for frequency modes 4, 5, 6, and 7. An additional bit indicates whether the setting has a positive or negative value. The settings are linear and each phase offset setting adds a delay amount.

For more information about the specified phase-shift settings, refer to the *Device Datasheet for Arria II Devices*.

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; 32 for frequency modes 4, 5, 6, and 7. Therefore, the actual physical offset setting range is 64 or 32 subtracted by the DQS delay settings from the DLL.

If you use this feature, monitor the DQS delay settings to know how many offsets you can add and subtract in the system. 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 required to achieve a 30° phase shift in DLL frequency mode 1, you can subtract up to 28 phase offset settings and add up to 35 phase offset settings to achieve the optimal delay required. However, if the same DQS delay settings of 28 is required to achieve a 30° phase shift in DLL frequency mode 4, subtract up to 28 phase offset settings, but 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.

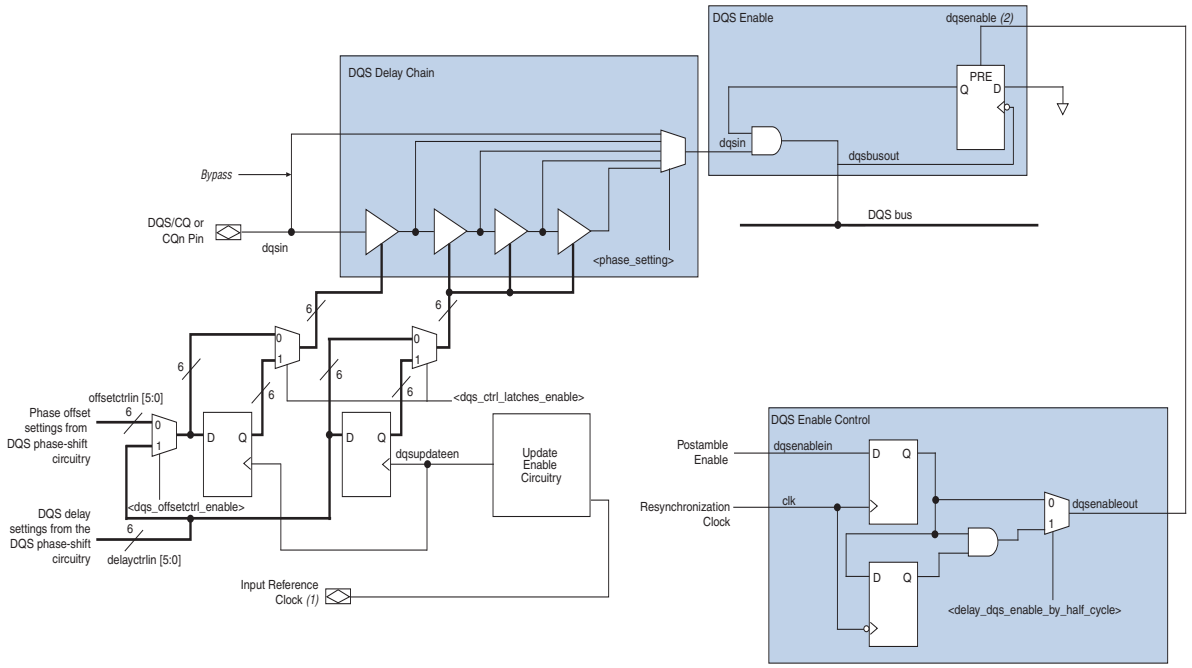
For more information about the value for each step, refer to the *Device Datasheet for Arria II Devices*.

When using static phase offset, specify the phase offset amount in the ALTMEMPHY megafunction as a positive number for addition or a negative number for subtraction. You can also have a dynamic phase offset that is always added to, subtracted from, or both added to and subtracted from the DLL phase shift. When you always add or subtract, you can dynamically input the phase offset amount into the `dll_offset[5..0]` port. When you want to both add and subtract dynamically, you control the `addnsub` signal in addition to the `dll_offset[5..0]` signals.

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 (refer to Figure 7–21).

Figure 7–21. DQS Logic Block for Arria II Devices



Notes to Figure 7–21:

(1) The input reference clock for the DQS phase-shift circuitry can come from a PLL output clock or an input clock pin. For the exact PLL and input clock pin, refer to Table 7–6 on page 7–28 and Table 7–10 on page 7–32.

(2) The `dqsenable` signal can also come from the Arria II GX FPGA fabric.

DQS Delay Chains

DQS delay chains consist of a set of variable delay elements to allow the DQS/CQ and CQn in out 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 DQS delay setting and the phase-offset setting. The number of delay chains required is transparent because the ALTMEMPHY megafunction and UniPHY IP core automatically set 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.

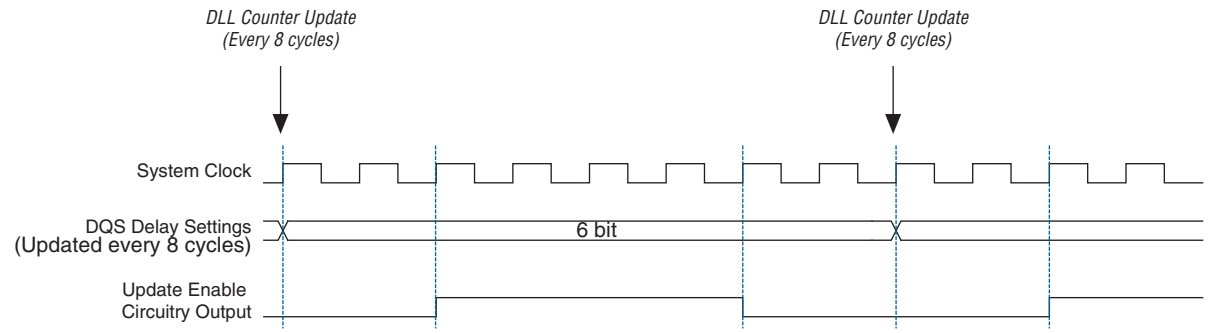
The delay elements in the DQS logic block have the same characteristics as the delay elements in the DLL. When the DLL is not used to control the DQS delay chains, you can input your own Gray-coded 6-bit or 5-bit settings with the `dqs_delayctrlin[5..0]` signals available in the ALTMEMPHY megafunction and UniPHY IP core. These settings control 1, 2, 3, or all 4 delay elements in the DQS delay chains. The ALTMEMPHY megafunction and UniPHY IP core can also dynamically choose the number of DQS delay chains required 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 a 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 and UniPHY IP core use this circuit by default. Figure 7–22 shows an example waveform of the update enable circuitry output.

Figure 7–22. 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 ensures that data is not lost if there is noise on the DQS line at the end of a read postamble time.

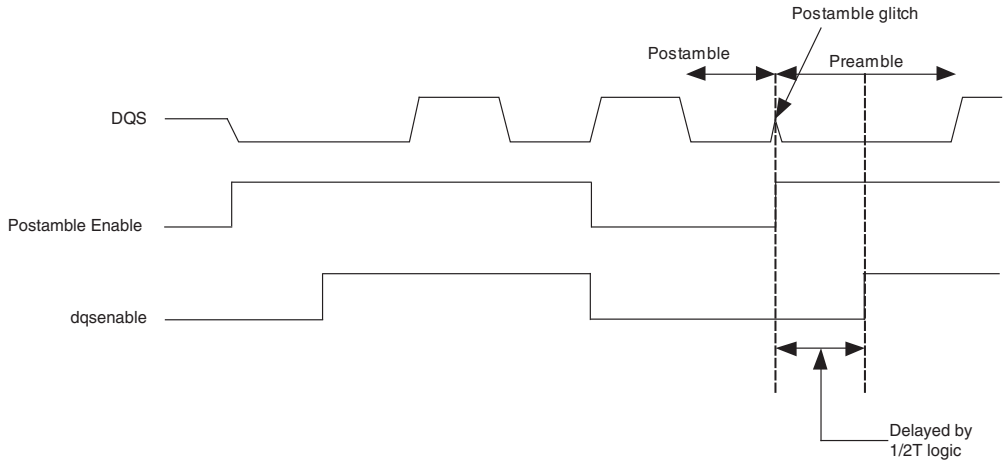
Arria II devices have dedicated postamble registers that you can control to ground the shifted DQS signal used to clock the DQ input registers at the end of a read operation. This ensures that any glitches on the DQS input signals at the end of the read postamble time do not affect the DQ IOE registers.

In addition to the dedicated postamble register, Arria II GZ devices also have a half-data rate (HDR) block inside the postamble enable circuitry. Use these registers if the controller is running at half the frequency of the I/Os.

Using the HDR block as the first stage capture register in the postamble enable circuitry block is optional. The HDR block is clocked by the half-rate resynchronization clock, which is the output of the I/O clock divider circuit (shown in Figure 7–26 on page 7–39).

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 shown in Figure 7–23.

Figure 7–23. Avoiding Glitch on a Non-Consecutive Read Burst Waveform

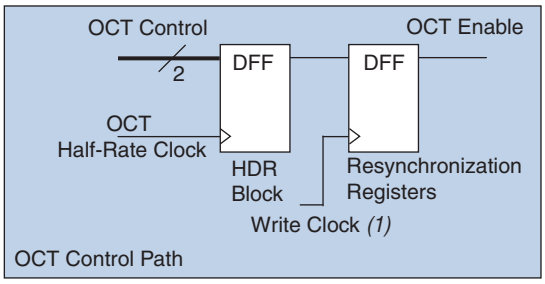


Arria II GZ Dynamic On-Chip Termination Control

Figure 7–24 shows the dynamic OCT control block. The block includes all the registers required to dynamically turn on the on-chip parallel termination (R_T OCT) during a read and turn R_T OCT off during a write.

For more information about the dynamic OCT control block, refer to the *I/O Features in Arria II Devices* chapter.

Figure 7–24. Dynamic OCT Control Block for Arria II GZ Devices



Note to Figure 7–24:

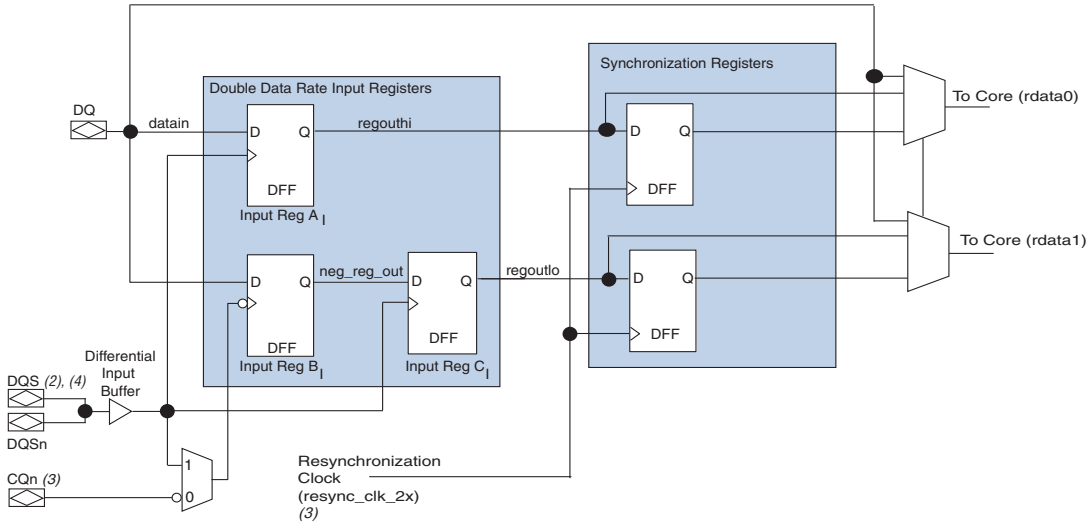
(1) The write clock comes from the PLL.

I/O Element Registers

IOE registers are expanded to allow source-synchronous systems to have faster register-to-register transfers and resynchronization. For Arria II GX devices, both top, bottom, and right IOEs have the same capability. Right IOEs have extra features to support LVDS data transfer. For Arria II GZ devices, both top and bottom, and left and right IOEs have the same capability. Left and right IOEs have extra features to support LVDS data transfer.

Figure 7–25 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 7–25. IOE Input Registers for Arria II GX Devices ***(Note 1)***



Notes to Figure 7–25:

(1) You can bypass each register block in this path.

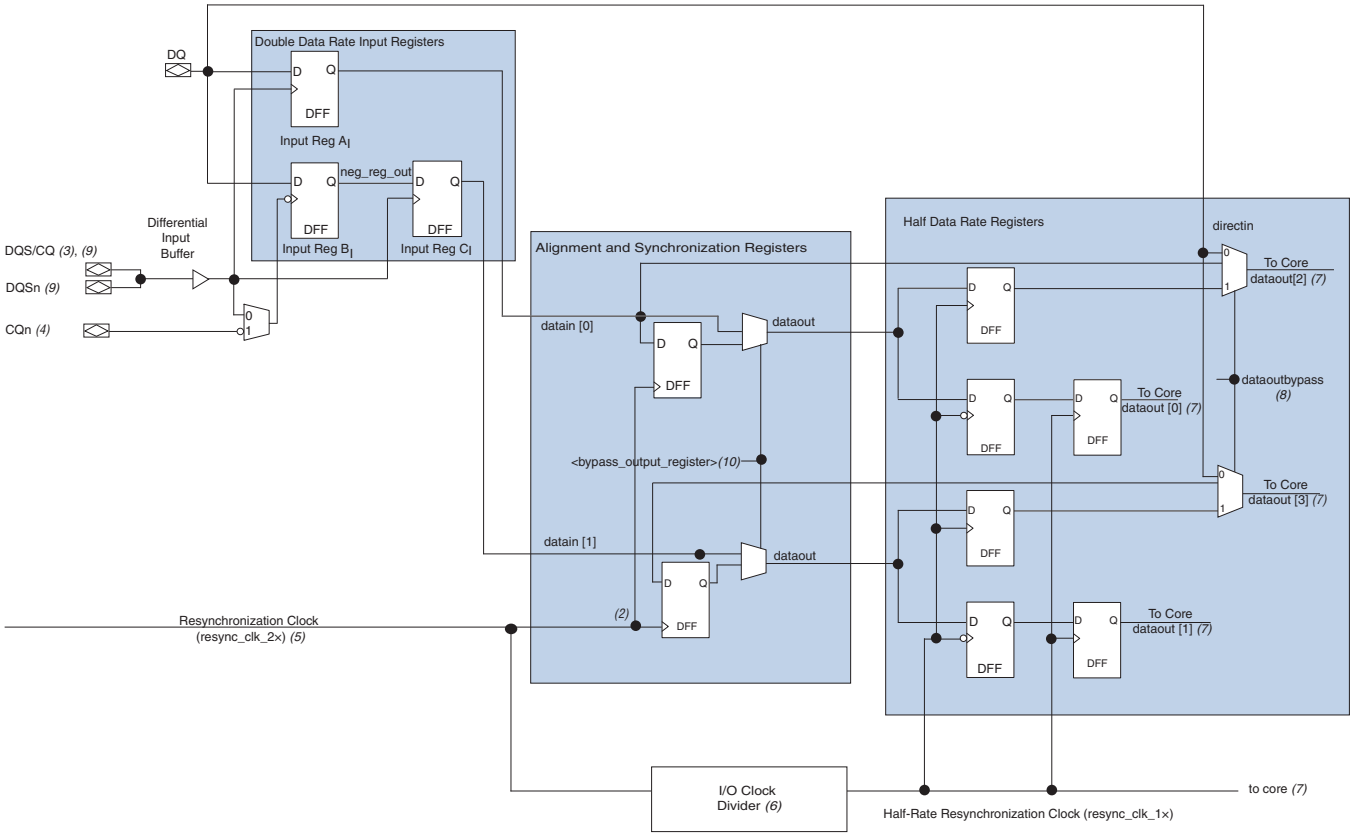
(2) The input clock can be from the DQS logic block (whether the postamble circuitry is bypassed or not) or from a global clock line.

(3) This input clock comes from the CQn logic block.

(4) The DQS signal must be inverted for DDR interfaces except for the QDR II+/QDR II SRAM interfaces. This inversion is done automatically if you use the Altera external memory interface IPs.

Figure 7–26 shows the registers available in the Arria II GZ input path. The input path consists of the DDR input registers, resynchronization registers, and HDR block. You can bypass each block of the input path.

Figure 7–26. IOE Input Registers for Arria II GZ Devices *(Note 1)*



Notes to Figure 7–26:

(1) You can bypass each register block in this path.

(2) This is the 0-phase resynchronization clock.

(3) The input clock can be from the DQS logic block (whether the postamble circuitry is bypassed or not) or from a GCLK line.

(4) This input clock comes from the CQn logic block.

(5) This resynchronization clock comes from a PLL through the clock network (`resync_ck_2x`).

(6) The I/O clock divider resides adjacent to the DQS logic block. In addition to the PLL, the I/O clock divider can also be fed by the DQS bus or CQn bus.

(7) The half-rate data and clock signals feed into a dual-port RAM in the FPGA core.

(8) You can dynamically change the `dataoutbypass` signal after configuration to select either the `directin` input or the output from the half data rate register to feed `dataout`.

(9) The DQS and DQSn signals must be inverted for DDR, DDR2, and DDR3 interfaces. When using Altera's memory interface IPs, the DQS, and DQSn signals are automatically inverted.

(10) The **bypass_output_register** option allows you to select either the output from the second mux or the output of the fourth alignment/ synchronization register to feed `dataout`.

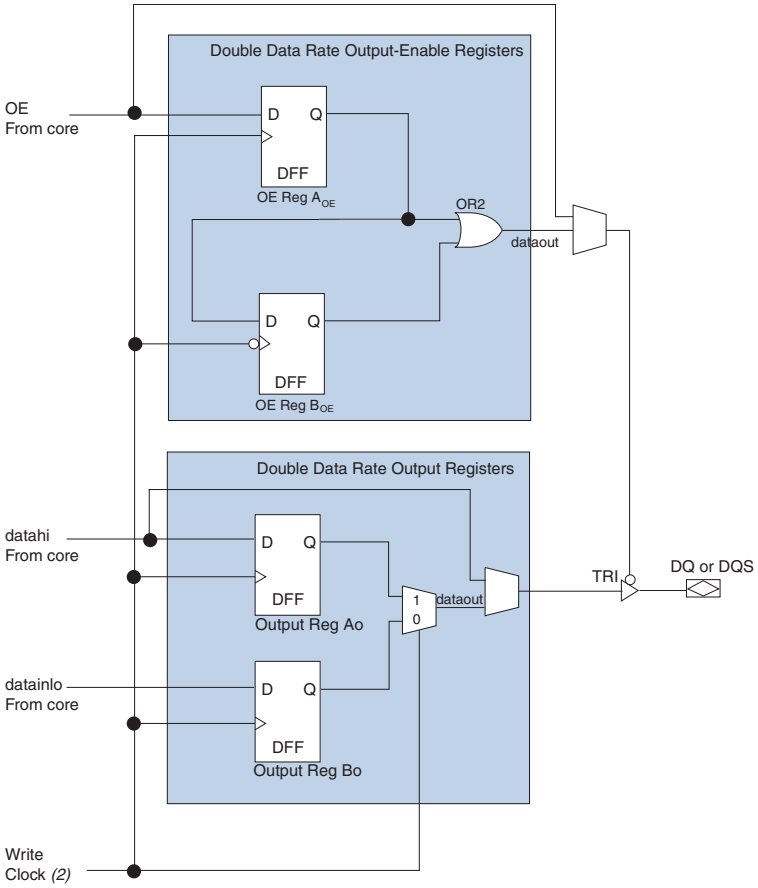
There are three registers in the DDR input registers block. Two registers capture data on the positive and negative edges of the clock, and the third register aligns the captured data. You can choose to use the same clock for the positive edge and negative edge registers, or two complementary clocks (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.

For Arria II GX devices, 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.

For Arria II GZ devices, the resynchronization registers resynchronize the data to the system clock domain. These registers are clocked by the resynchronization clock that is generated by the PLL. The outputs of the resynchronization registers can go straight to the core or to the HDR blocks, which are clocked by the divided-down resynchronization clock.

Figure 7–27 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 7–27. IOE Output and Output Enable Path Registers for Arria II GX Devices ***(Note 1)***



Notes to Figure 7–27:

(1) You can bypass each register block of the output and output-enable paths.

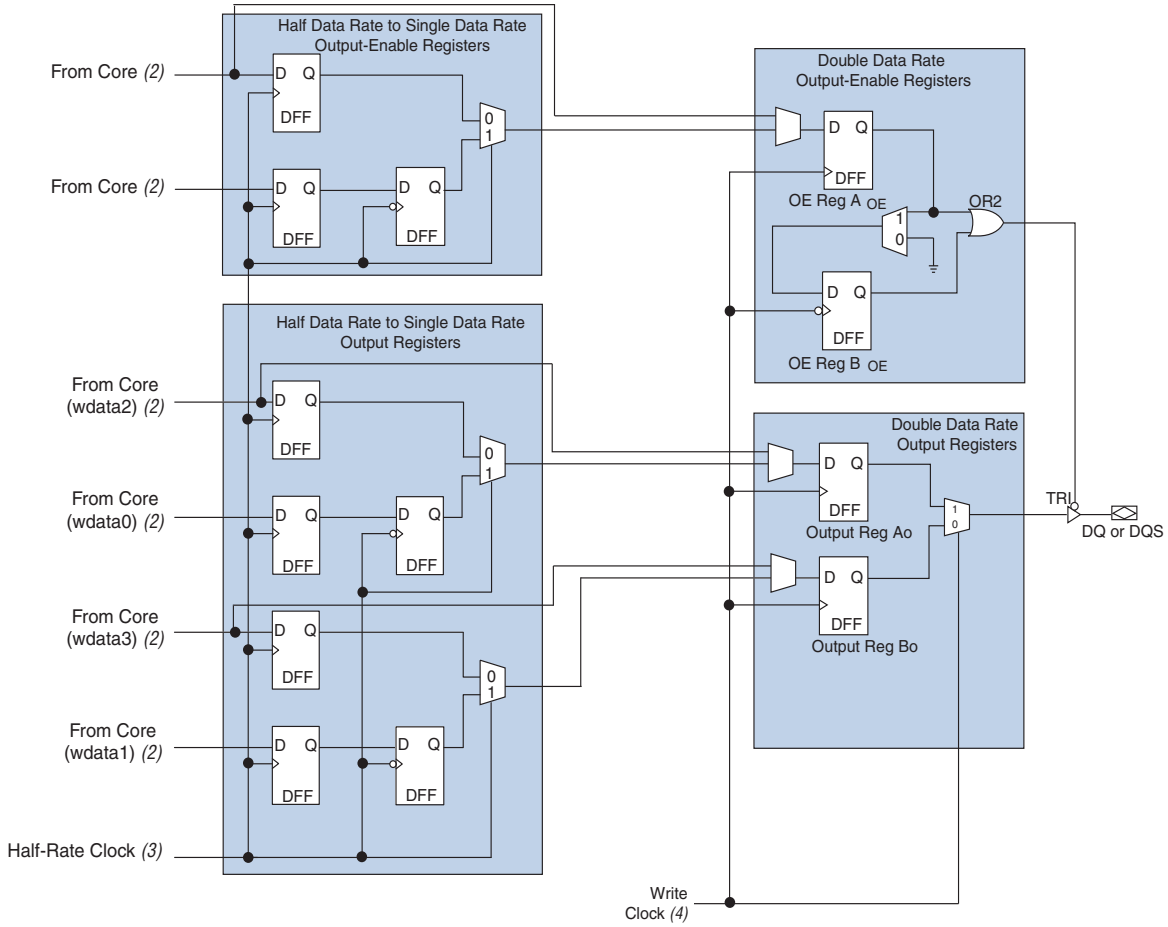
(2) The write clock comes from the PLL. The DQ write clock and DQS write clock have a 90° offset between them.

For Arria II GX devices, the output path is designed to route combinatorial or registered single data rate (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.

Figure 7–28 shows the registers available in the Arria II GZ output and output-enable paths. The path is divided into the HDR block, resynchronization registers, and output and output-enable registers. The device can bypass each block of the output and output-enable path.

Figure 7–28. IOE Output and Output-Enable Path Registers for Arria II GZ Devices ***(Note 1)***



Notes to Figure 7–28:

(1) You can bypass each register block of the output and output-enable paths.

(2) Data coming from the FPGA core are at half the frequency of the memory interface clock frequency in half-rate mode.

(3) The half-rate clock comes from the PLL.

(4) The write clock comes from the PLL. The DQ write clock and DQS write clock have a 90° offset between them.

For Arria II GZ devices, the output path is designed to route combinatorial or registered SDR outputs and full-rate or half-rate DDR outputs from the FPGA core. Half-rate data is converted to full-rate using the HDR block, clocked by the half-rate clock from the PLL.

The output-enable path has a structure similar to the output path. You can have a combinatorial or registered output in SDR applications and you can use half-rate or full-rate operation in DDR applications. Also, the ouput-enable path's resynchronization registers have a structure similar to the output path registers, ensuring that the output-enable path goes through the same delay and latency as the output path.

Document Revision History

Table 7–11 shows the revision history for this document.

Table 7–11. Document Revision History (Part 1 of 2)

Date	Version	Changes
June 2011	4.1	■ Updated Table 7–3. ■ Updated Figure 7–11, Figure 7–12, Figure 7–13, Figure 7–14, and Figure 7–15. ■ Minor text edits.
December 2010	4.0	■ Updated for the Quartus II software version 10.1 release. ■ Added Arria II GZ devices information. ■ Added Figure 7–2, Figure 7–10, Figure 7–11, Figure 7–12, Figure 7–13, Figure 7–14, Figure 7–15, Figure 7–17, Figure 7–19, Figure 7–24, Figure 7–26, and Figure 7–26. ■ Added Table 7–1, Table 7–3, Table 7–4, Table 7–5, Table 7–3, Table 7–4, Table 7–6, Table 7–7, Table 7–8, and Table 7–9. ■ Updated Table 7–10. ■ Added "Using the RUP and RDN Pins in a DQ/DQS Group Used for Memory Interfaces in Arria II GZ Devices" and "Arria II GZ Dynamic On-Chip Termination Control" sections. ■ Minor text edits.
July 2010	3.0	Updated for Arria II GX v10.0 release: ■ Updated *"Arria II Memory Interfaces Pin Support"* section by adding reference to the *Section I. Device and Pin Planning* in volume 2 of the *External Memory Interface Handbook* and removing "Table 7–1: Memory Interface Pin Utilization". ■ Update DLL numbering to match with the Quartus II software. ■ Minor text edits.
November 2009	2.0	Updated for Arria II GX v9.1 release: ■ Updated Table 7–1, Table 7–2, and Table 7–5. ■ Updated Figure 7–1, Figure 7–2, Figure 7–3, Figure 7–11, Figure 7–12, Figure 7–13, Figure 7–15, and Figure 7–16. ■ Updated the "Arria II GX External Memory Interface Features" section. ■ Added new "Combining ×16/×18 DQ/DQS Groups for ×36 QDR II+/QDR II SRAM Interface" section. ■ Minor text edits.

Table 7–11. Document Revision History (Part 2 of 2)

Date	Version	Changes
June 2009	1.2	■ Added Table 7–2. ■ Updated Table 7–1, Table 7–3, and Table 7–5. ■ Updated Figure 7–1, Figure 7–3, Figure 7–4, Figure 7–5, Figure 7–6, Figure 7–7, Figure 7–8, Figure 7–9, and Figure 7–11. ■ Updated "Introduction" and "DLL" sections.
February 2009	1.1	Updated Table 7–1 and Table 7–2.
February 2009	1.0	Initial release.

8. High-Speed Differential I/O Interfaces and DPA in Arria II Devices

AIIGX51008-4.3

This chapter describes the high-speed differential I/O features and resources, the functionality of the serializer/deserializer (SERDES), and the dynamic phase alignment (DPA) circuitry in Arria® II devices.

This chapter contains the following sections:

- "LVDS Channels" on page 8–2
- "LVDS SERDES and DPA Block Diagram" on page 8–7
- "Differential Transmitter" on page 8–8
- "Differential Receiver" on page 8–11
- "Programmable Pre-Emphasis and Programmable V_{OD}." on page 8–10
- "Differential I/O Termination" on page 8–20
- "PLLs" on page 8–21
- "LVDS and DPA Clock Networks" on page 8–21
- "Source-Synchronous Timing Budget" on page 8–23
- "Differential Pin Placement Guidelines" on page 8–27
- "Setting Up an LVDS Transmitter or Receiver Channel" on page 8–36

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

- Differential I/O buffer
- Transmitter serializer
- Receiver deserializer
- Data realignment block (bit slip)
- DPA block
- Synchronizer (FIFO buffer)

Arria II devices support the following high-speed differential I/O standards:

- LVDS
- mini-LVDS
- RSDS
- LVPECL
- Bus LVDS (BLVDS) for Arria II GX devices



True mini-LVDS and RSDS inputs are not supported. The LVPECL I/O standard is only used for phase-locked loop (PLL) clock inputs in differential mode.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

For specifications and features of the differential I/O standards supported in Arria II devices, refer to the *I/O Features in Arria II Devices* and *Arria II Devices Data Sheet* chapters.

LVDS Channels

In Arria II GX devices, there are true 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 on-chip differential termination (R_D OCT) support. You can configure the LVDS I/O buffers as either LVDS input (without R_D OCT) or true LVDS output buffers. You can also configure the LVDS pins on the top, bottom, and right sides of the device, as emulated LVDS output buffers, which use two single-ended output buffers with an external resistor network to support LVDS, mini-LVDS, and RSDS standards.

The Arria II GZ devices support LVDS on both row and column I/O banks. Row I/Os support true LVDS input with 100-Ω R_D OCT and true LVDS output buffers. Column I/Os supports true LVDS input buffers without R_D OCT. You can also configure the row and column LVDS pins as emulated LVDS output buffers that use two single-ended output buffers with an external resistor network to support LVDS, mini-LVDS, and RSDS standards. Arria II GZ devices offer single-ended I/O `refclk` support for the LVDS.

Dedicated SERDES and DPA circuitries are implemented on the right I/O banks of Arria II GX devices and row I/O banks of Arria II GZ devices to further enhance the LVDS interface performance in the device. For column I/O banks in Arria II devices, SERDES is implemented in the core logic because there is no dedicated SERDES circuitry.

When you configure the I/O buffers as LVDS input with R_D OCT enabled, you must set both V_{CCIO} and V_{CCPD} to **2.5 V**.

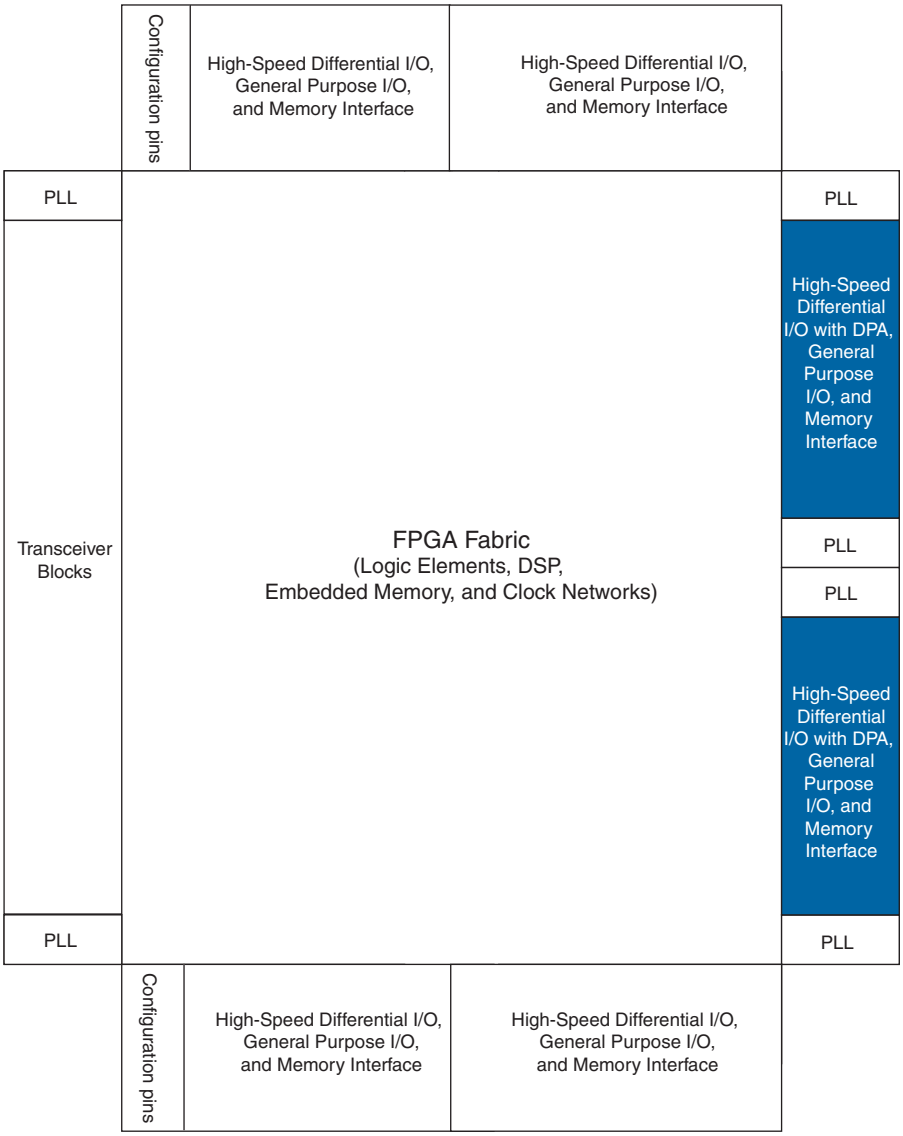
For more information about I/O banks, refer to the *I/O Features in Arria II Devices* chapter.

Locations of the I/O Banks

Arria II I/Os are divided into 16 to 20 I/O banks. For Arria II GX devices, the high-speed differential I/O s are located at the right side of the device. For Arria II GZ devices, the high-speed differential I/Os are located at the right and left sides of the device.

Figure 8–1 and Figure 8–2 show a high-level chip overview of Arria II devices.

Figure 8–1. High-Speed Differential I/Os with DPA Block Locations in an Arria II GX Device *(Note 1)*, *(2)*, *(3)*

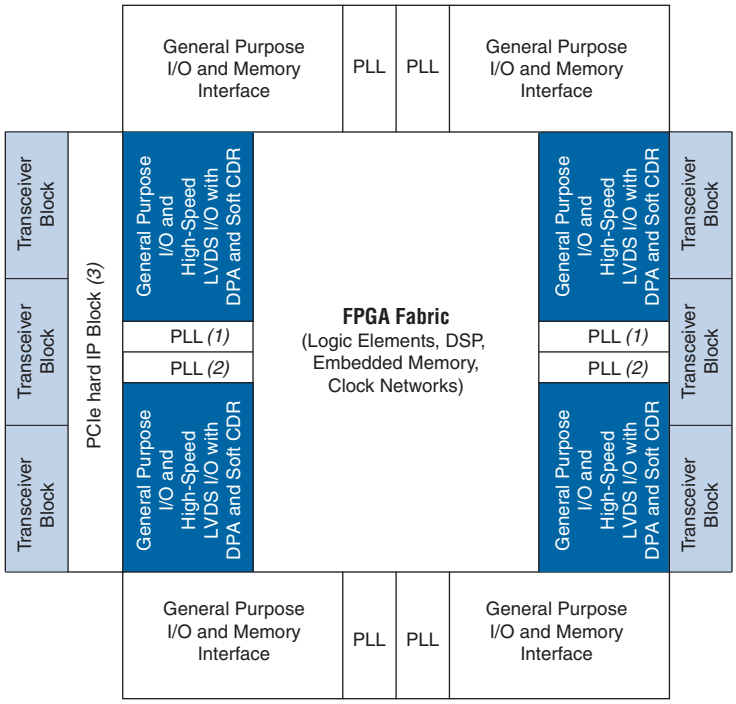


Notes to Figure 8–1:

(1) This figure 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 EP2AGX45 and EP2AGX65 devices.

Figure 8–2 shows a high-level chip overview of the Arria II GZ devices.

Figure 8–2. High-Speed Differential I/Os with DPA Block Locations in Arria II GZ Devices



Notes to Figure 8–2:

(1) Not available for F780 device package.
(2) Not available for F780 and F1152 device packages.
(3) The PCIe hard IP block is located on the left side of the device only (IOBANK_QL).

Table 8–1 to Table 8–4 list the maximum number of row and column LVDS I/Os supported in Arria II devices. You can design the LVDS I/Os as true LVDS input, output buffers, or emulated LVDS output buffers, if 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 8–1). You can design up to a maximum of either:

- 28 true LVDS input buffers with R_D OCT and 28 true LVDS output buffers
- 56 LVDS input buffers of which 28 are true LVDS input buffers with R_D OCT and 28 requires external 100-Ω termination
- 28 true LVDS output buffers and 28 emulated LVDS output buffers
- 56 emulated LVDS output buffers

☞ Dedicated SERDES and DPA circuitry are only available on the right side of the device in row I/O banks. SERDES with DPA receivers are only available on differential pins in the row I/O banks and SERDES transmitters are only available on transmit (Tx) pins in the row I/O banks. The receive (Rx) pins in row I/O banks are receiver channels without dedicated SERDES and DPA circuitry.

Table 8–1. LVDS Channels Supported in Arria II GX Device Row I/O Banks *(Note 1)*, *(2)*, *(3)*, *(4)*, *(5)*, *(6)*

Device	358-Pin FlipChip UBGA	572-Pin FlipChip FBGA	780-Pin FlipChip FBGA	1152-Pin FlipChip FBGA
EP2AGX45	8(R_D or eTx) + 8(Rx, Tx or eTx)	24(R_D or eTx) + 24(Rx, Tx, or eTx)	28(R_D or eTx) + 28(Rx, Tx, or eTx)	—
EP2AGX65	8(R_D or eTx) + 8(Rx, Tx, or eTx)	24(R_D or eTx) + 24(Rx, Tx, or eTx)	28(R_D or eTx) + 28(Rx, Tx or eTx)	—
EP2AGX95	—	24(R_D or eTx) + 24(Rx, Tx or eTx)	28(R_D or eTx) + 28(Rx, Tx or eTx)	32(R_D or eTx) + 32(Rx, Tx, or eTx)
EP2AGX125	—	24(R_D or eTx) + 24(Rx, Tx or eTx)	28(R_D or eTx) + 28((Rx, Tx or eTx)	32(R_D or eTx) + 32(Rx, Tx or eTx)
EP2AGX190	—	—	28(R_D or eTx)+ 28(Rx, Tx or eTx)	48(R_D or eTx) + 48(Rx, Tx or eTx)
EP2AGX260	—	—	28(R_D or eTx) + 28(Rx, Tx or eTx)	48(R_D or eTx) + 48(Rx, Tx or eTx)

Notes to Table 8–1:

(1) Dedicated SERDES and DPA circuitry only exist on the right side of the device in the Row I/O banks.
(2) R_D = True LVDS input buffers with R_D OCT support and dedicated SERDES receiver channel with DPA circuitry.
(3) Rx = True LVDS input buffers without R_D OCT support and dedicated SERDES receiver channel with DPA circuitry.
(4) Tx = True LVDS output buffers and dedicated SERDES transmitter channel.
(5) eTx = Emulated LVDS output buffers, either LVDS_E_3R or LVDS_E_1R.
(6) The LVDS channel count does not include dedicated clock input pins and PLL clock output pins.

Table 8–2. LVDS Channels Supported in Arria II GX Device Column I/O Banks *(Note 1)*, *(2)*, *(3)*, *(4)*, *(5)*, *(6)* **(Part 1 of 2)**

Device	358-Pin FlipChip UBGA	572-Pin FlipChip FBGA	780-Pin FlipChip FBGA	1152-Pin FlipChip FBGA
EP2AGX45	25(R_D or eTx) + 24(Rx, Tx, or eTx)	33(R_D or eTx) + 32(Rx, Tx, or eTx)	57(R_D or eTx) + 56(Rx, Tx, or eTx)	—
EP2AGX65	25(R_D or eTx) + 24(Rx, Tx, or eTx)	33(R_D or eTx) + 32(Rx, Tx, or eTx)	57(R_D or eTx) + 56(Rx, Tx, or eTx)	—
EP2AGX95	—	33(R_D or eTx) + 32(Rx, Tx, or eTx)	57(R_D or eTx) + 56(Rx, Tx, or eTx)	73(R_D or eTx) + 72(Rx, Tx, or eTx)
EP2AGX125	—	33(R_D or eTx) + 32(Rx, Tx, or eTx)	57(R_D or eTx) + 56(Rx, Tx, or eTx)	73(R_D or eTx) + 72(Rx, Tx, or eTx)
EP2AGX190	—	—	57(R_D or eTx) + 56(Rx, Tx, or eTx)	97(R_D or eTx) + 96(Rx, Tx, or eTx)

Table 8–2. LVDS Channels Supported in Arria II GX Device Column I/O Banks *(Note 1)*, *(2)*, *(3)*, *(4)*, *(5)*, *(6)* (Part 2 of 2)

Device	358-Pin FlipChip UBGA	572-Pin FlipChip FBGA	780-Pin FlipChip FBGA	1152-Pin FlipChip FBGA
EP2AGX260	—	—	57(R_D or eTx) + 56(Rx, Tx, or eTx)	97(R_D or eTx) + 96(Rx, Tx, or eTx)

Notes to Table 8–2:

(1) There are no dedicated SERDES and DPA circuitry in device column I/O banks.
(2) R_D = True LVDS input buffers with R_D OCT support.
(3) Rx = True LVDS input buffers without R_D OCT support.
(4) Tx = True LVDS output buffers.
(5) eTx = Emulated LVDS output buffers, either LVDS_E_3R or LVDS_E_1R.
(6) The LVDS channel count does not include dedicated clock input pins and PLL clock output pins.

Table 8–3 and Table 8–4 list the maximum number of row and column LVDS I/Os supported in Arria II GZ devices.

Table 8–3. LVDS Channels Supported in Arria II GZ Device Row I/O Banks *(Note 1)*, *(2)*, *(3)*

Device	780-Pin FineLine BGA	1152-Pin FineLine BGA	1517-Pin FineLine BGA
EP2AGZ225	—	42(Rx or eTx) + 44(Tx or eTx)	86(Rx or eTx) + 88(Tx or eTx)
EP2AGZ300	—	42(Rx or eTx) + 44(Tx or eTx)	86(Rx or eTx) + 88(Tx or eTx)
EP2AGZ350	—	42(Rx or eTx) + 44(Tx or eTx)	86(Rx or eTx) + 88(Tx or eTx)

Notes to Table 8–3:

(1) Rx = true LVDS input buffers with R_D OCT, Tx = true LVDS output buffers, eTx = emulated LVDS output buffers (either `LVDS_E_1R` or `LVDS_E_3R`).
(2) The LVDS Rx and Tx channels are equally divided between the left and right sides of the device, except for the devices in the 780-pin Fineline BGA. These devices have the LVDS Rx and Tx located on the left side of the device.
(3) The LVDS channel count does not include dedicated clock input pins.

Table 8–4. LVDS Channels Supported in Arria II GZ Device Column I/O Banks *(Note 1)*, *(2)*, *(3)*

Device	780-Pin FineLine BGA	1152-Pin FineLine BGA	1517-Pin FineLine BGA
EP2AGZ225	—	93(Rx or eTx) + 96 eTx	93(Rx or eTx) + 96 eTx
EP2AGZ300	68(Rx or eTx) + 72 eTx	93(Rx or eTx) + 96 eTx	93(Rx or eTx) + 96 eTx
EP2AGZ350	68(Rx or eTx) + 72 eTx	93(Rx or eTx) + 96 eTx	93(Rx or eTx) + 96 eTx

Notes to Table 8–4:

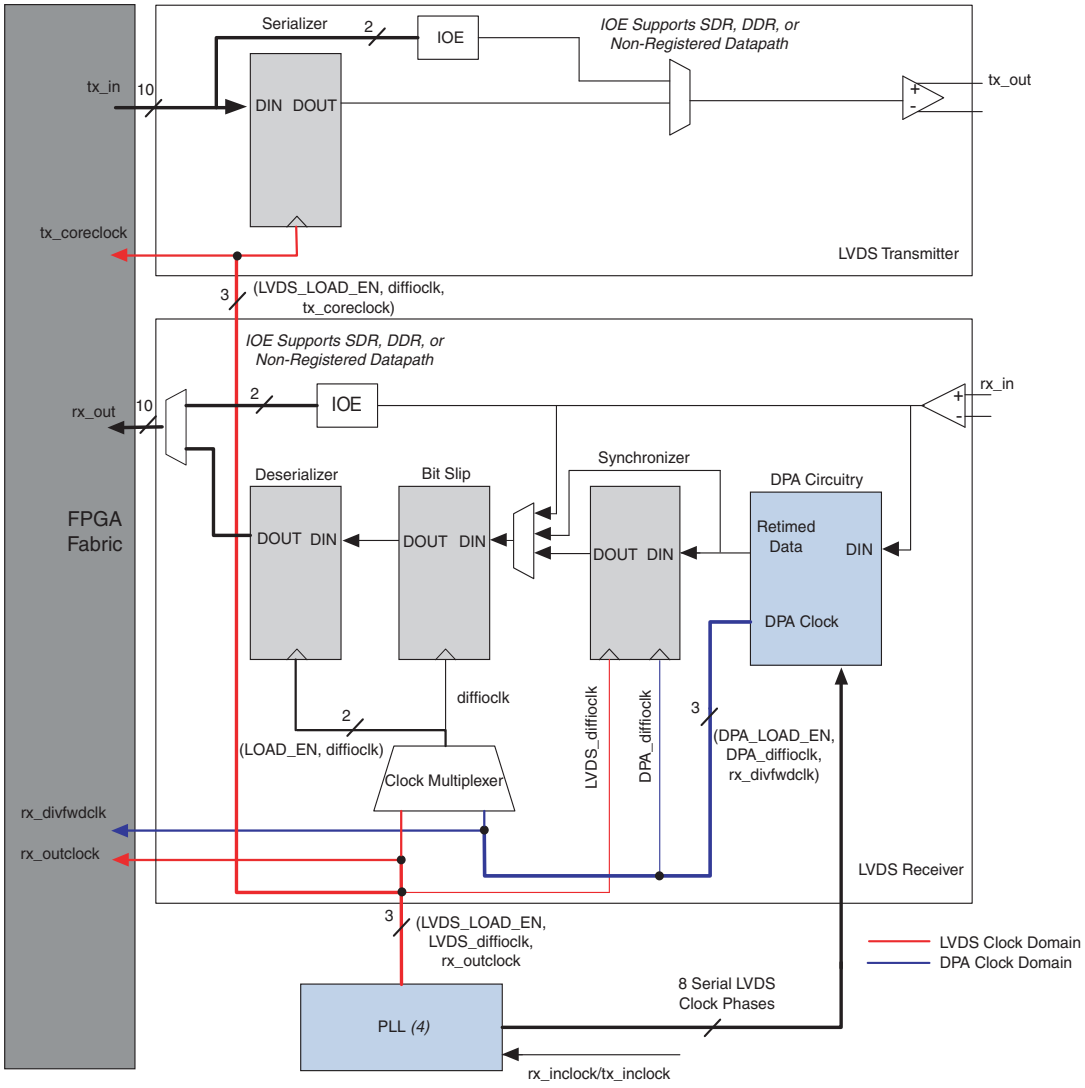
(1) Rx = true LVDS input buffers without R_D OCT, eTx = emulated LVDS output buffers (either `LVDS_E_1R` or `LVDS_E_3R`).
(2) The LVDS Rx and Tx channels are equally divided between the top and bottom sides of the device.
(3) The LVDS channel count does not include dedicated clock input pins.

LVDS SERDES and DPA Block Diagram

The Arria II GX devices have dedicated SERDES and DPA circuitry for LVDS transmitters and receivers on the right side. The Arria II GZ devices have dedicated SERDES and DPA circuitry for LVDS transmitters and receivers on the row I/O banks.

Figure 8–3 shows the LVDS SERDES and DPA block diagram. This diagram shows the interface signals for the transmitter and receiver datapaths. For more information, refer to "Differential Transmitter" on page 8–8 and "Differential Receiver" on page 8–11.

Figure 8–3. LVDS SERDES and DPA Block Diagram *(Note 1)*, *(2)*, *(3)*



Notes to Figure 8–3:

(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 modes, the data width is 1 and 2 bits, respectively.

(3) The `tx_in` and `rx_out` ports have a maximum data width of 10 bits.

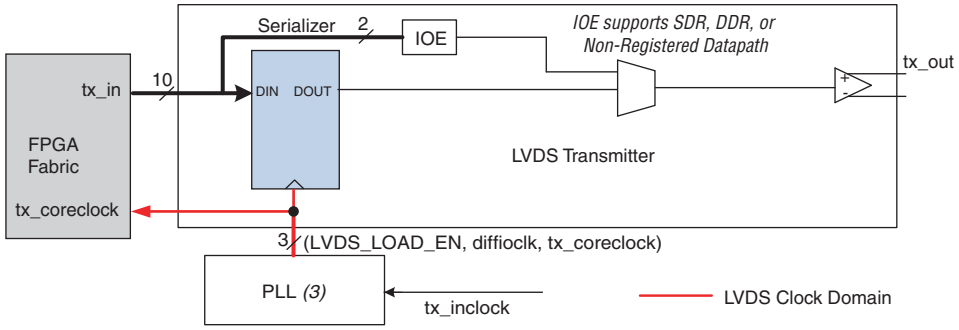
(4) Arria II GX center/corner PLL or Arria II GZ left/right PLL.

Differential Transmitter

The Arria II transmitter has a dedicated circuitry to provide support for LVDS signaling. The dedicated circuitry consists of a differential buffer, a serializer, and PLLs that can be shared between the transmitter and receiver. The differential buffer can drive out LVDS, mini-LVDS, and RSDS signaling levels. The differential output buffer supports programmable pre-emphasis and programmable voltage output differential (V_{OD}) controls, and can drive out mini-LVDS and RSDS signaling levels. Figure 8–4 is a block diagram of the LVDS transmitter.

When using emulated LVDS I/O standards at the differential transmitter, the SERDES circuitry must be implemented in logic cells but not hard SERDES.

Figure 8–4. LVDS Transmitter Block Diagram ***(Note 1)***, ***(2)***



Notes to Figure 8–4:

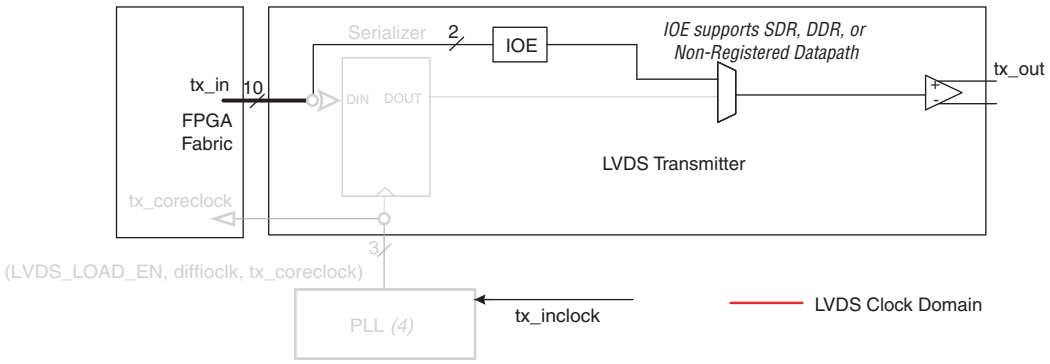
(1) In SDR and DDR modes, the data width is 1 and 2 bits, respectively.
(2) The `tx_in` port has a maximum data width of 10 bits.
(3) Arria II GX center/corner PLL or Arria II GZ left/right PLL.

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 x4, x6, x7, x8, or x10 using the ALTLVDS megafunction. The load enable signal is derived from the serialization factor setting.

You can bypass the serializer to support DDR (x2) and SDR (x1) 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 8–5 shows the serializer bypass path.

Figure 8–5. Serializer Bypass Path *(Note 1)*, *(2)*, *(3)*



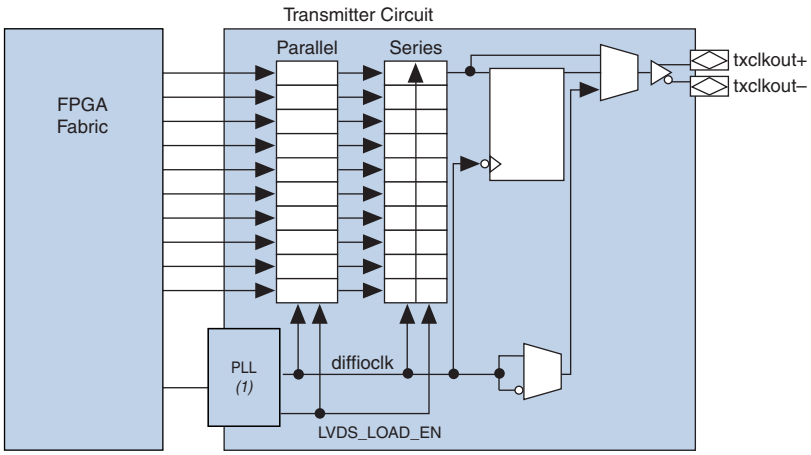
Notes to Figure 8–5:

(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 modes, the data width to the IOE is 1 and 2 bits, respectively.
(4) Arria II GX center/corner PLL or Arria II GZ left/right PLL.

Differential applications often require specific clock-to-data alignments or a specific data rate to clock rate factors. You can configure any Arria II 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 PLLs provide additional support for other phase shifts in 45° increments. These settings are made statically in the Quartus® II MegaWizard™ Plug-In Manager software.

Figure 8–6 shows the Arria II LVDS transmitter in clock output mode. In clock output mode, you can use an LVDS data channel as a clock output channel.

Figure 8–6. LVDS Transmitter in Clock Output Mode



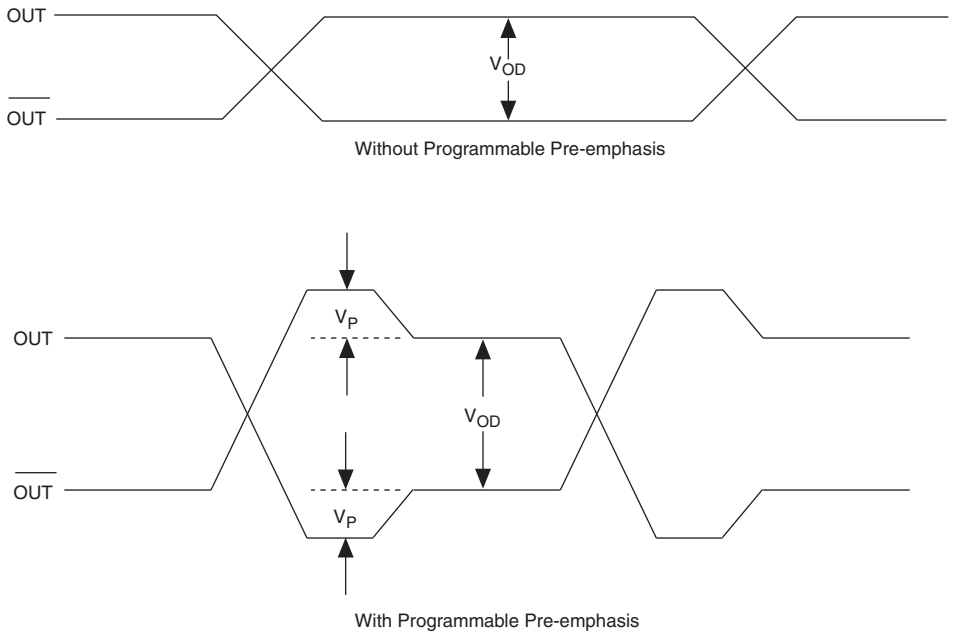
Note to Figure 8–6:

(1) Arria II GX center/corner PLL or Arria II GZ left/right PLL.

Programmable Pre-Emphasis and Programmable V_{OD}.

Pre-emphasis increases the amplitude of the high frequency component of the output signal, which helps to compensate for the frequency-dependent attenuation along the transmission line. Figure 8–7 shows the LVDS output single-ended waveform with and without pre-emphasis. The definition of V_{OD} is also shown.

Figure 8–7. LVDS Output Single-Ended Waveform with and without Programmable Pre-Emphasis *(Note 1)*



Note to Figure 8–7:

(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 must not be included in the V_{OD} voltage.

Table 8–5 lists the assignment name and its possible values for programmable pre-emphasis in the Quartus II software Assignment Editor.

Table 8–5. Programmable Pre-Emphasis Settings in Quartus II Software Assignment Editor

Assignment Name	Assignment Value	
	Arria II GX Device	Arria II GZ Device
Programmable Pre-Emphasis	0 (off), 1 (default on)	0 (default zero), 1 (medium low), 2 (medium high), 3 (high)

You can statically assign the V_{OD} settings from the Assignment Editor. Table 8–6 lists the assignment name for programmable V_{OD} and its possible values in the Quartus II software Assignment Editor.

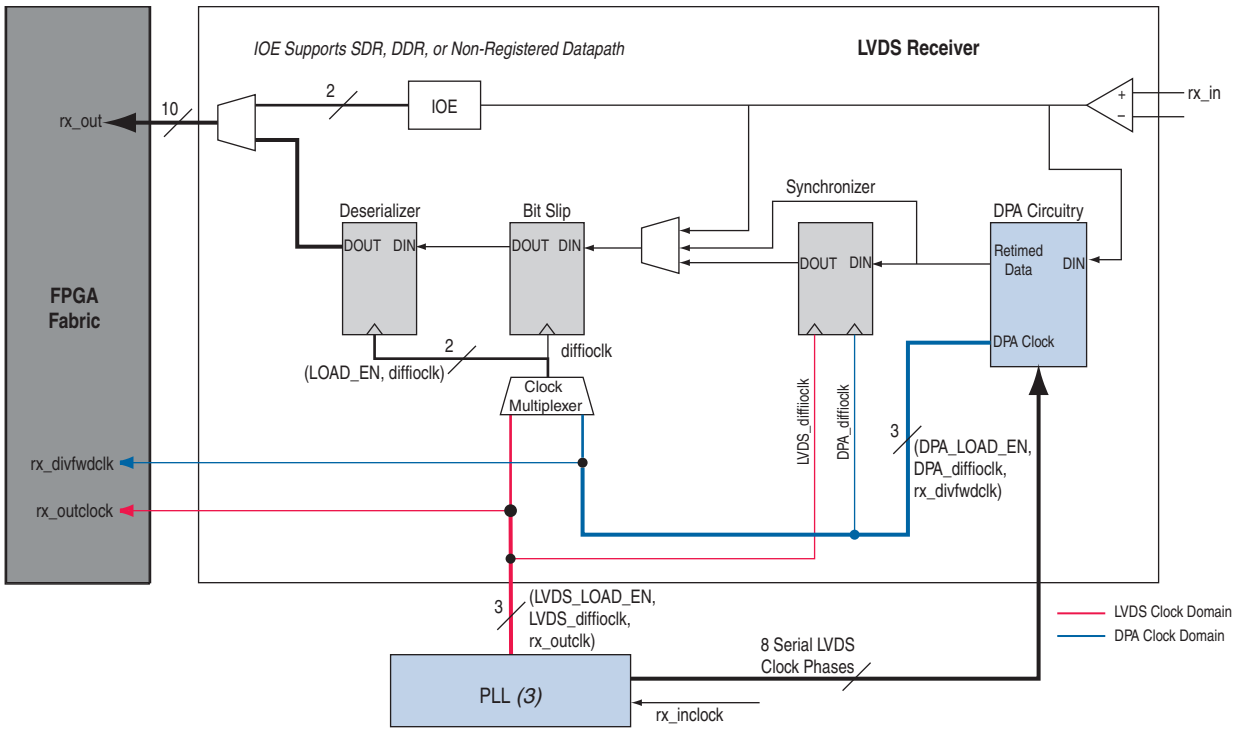
Table 8–6. Programmable V_{OD} Settings in Quartus II Software Assignment Editor

Assignment Name	Assignment Value	
	Arria II GX Device	Arria II GZ Device
Programmable Differential Output Voltage (V_{OD})	2	0, 1, 2, 3

Differential Receiver

The Arria II device family has a dedicated circuitry to receive high-speed differential signals in side or row I/Os. Figure 8–8 shows the hardware blocks of the Arria II receiver. The receiver has a differential buffer and PLLs that can be shared between the transmitter and receiver, a DPA block, a synchronizer, a data realignment block, and a deserializer. The differential buffer can receive LVDS signal levels, which are statically set in the Quartus II software Assignment Editor. Figure 8–8 shows a block diagram of an LVDS receiver in the right I/O bank.

Figure 8–8. LVDS Receiver Block Diagram ***(Note 1)***, ***(2)***



Notes to Figure 8–8:

(1) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.

(2) The `rx_out` port has a maximum data width of 10 bits.

(3) Arria II GX center/corner PLL or Arria II GZ left/right PLL.

The Arria II 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.

☞ Only non-DPA mode requires manual skew adjustment.

Arria II devices support 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 I/O banks only support non-DPA mode, in which the SERDES are implemented in the core logic.

Receiver Hardware Blocks

The differential receiver has the following hardware blocks:

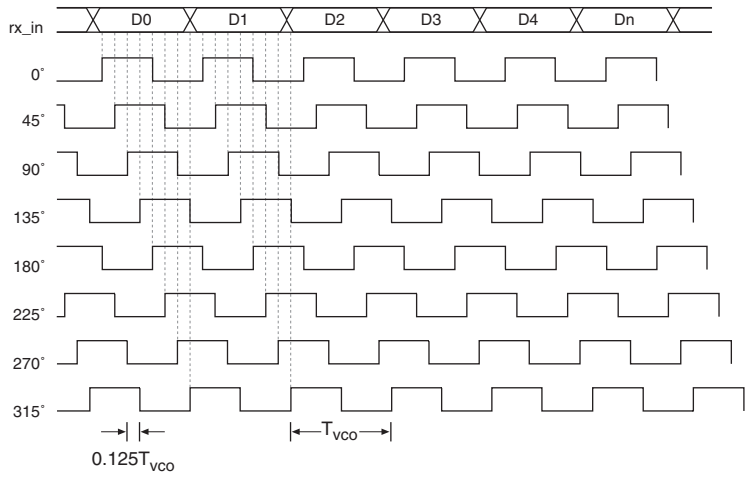
- "DPA" on page 8–12
- "Synchronizer" on page 8–13
- "Data Realignment Block (Bit Slip)" on page 8–14
- "Deserializer" on page 8–15

DPA

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 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 block. 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 8–9 shows the possible phase relationships between the DPA clocks and the incoming serial data.

Figure 8–9. DPA Clock Phase to Serial Data Timing Relationship ***(Note 1)***



Note to Figure 8–9:

(1) T_{VCO} is defined as the PLL serial clock period.

The DPA block requires a training pattern and 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 clock on its current phase when asserted. This signal is useful if you do not want the DPA circuitry to continuously adjust the phase after initial phase selection.

The DPA circuitry loses lock when it switches phases to maintain an optimal sampling phase. After it is locked, the DPA circuitry can lose the lock status 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 while in the 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 `DPA_diffioclk` and the high-speed clock (`LVDS_diffioclk`) produced by the PLL. Because every DPA channel might have a different phase selected to sample the data, you need the FIFO buffer 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 input reference clock of the receiver, and is automatically reset when the DPA circuitry 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 the data checker indicates corrupted received data.

Data Realignment Block (Bit Slip)

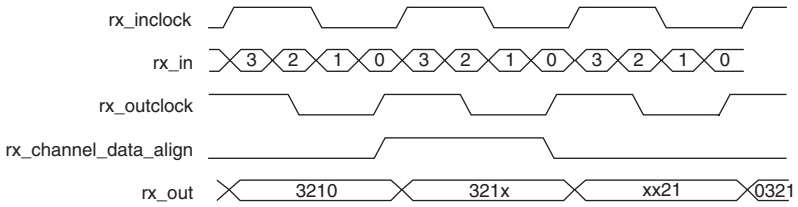
Skew in the transmitted data along with skew added by the link causes channel-to-channel skew on the received serial data streams. If you enabled the DPA block, the received data is captured with different clock phases on each channel and might cause the received data to be misaligned from channel to channel. To compensate for the 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 8–10 shows receiver output after a one bit-slip pulse with the deserialization factor set to **4**.

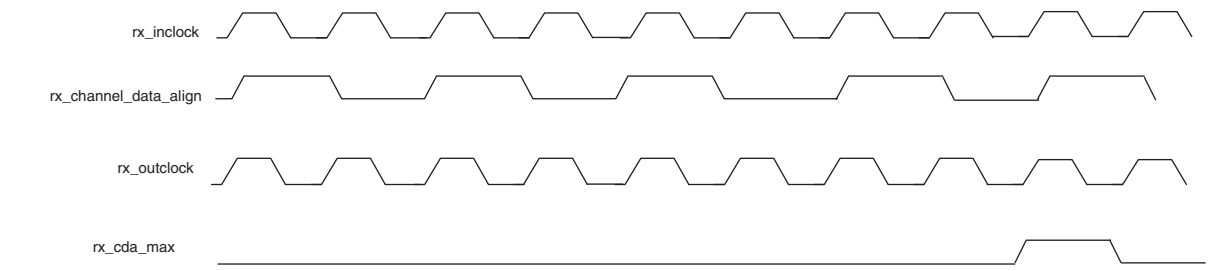
Figure 8–10. Data Realignment Timing



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 must 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 8–11 shows a preset value of 4-bit times before rollover occurs. The rx_cda_max signal pulses for one rx_outclock cycle to indicate that rollover has occurred.

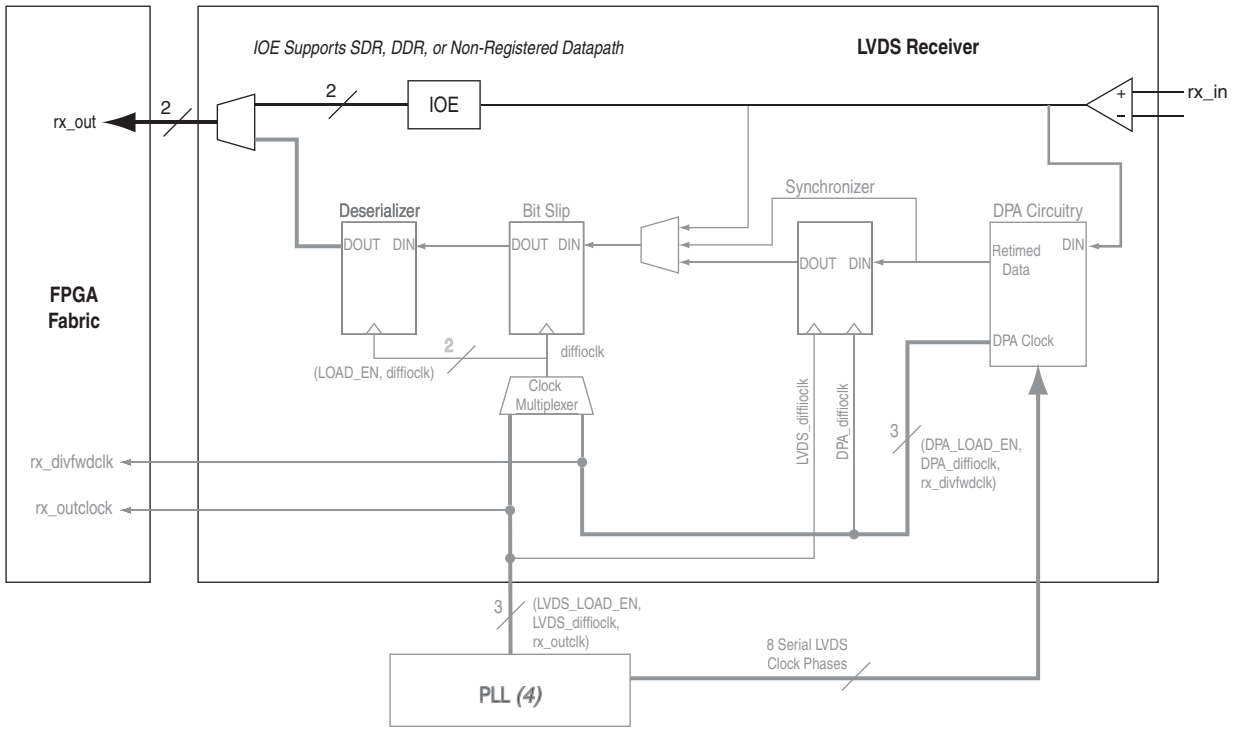
Figure 8–11. Receiver Data Re-Alignment Rollover



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. You can bypass the deserializer to support DDR (x2) and SDR (x1) operations, as shown in Figure 8–12. You cannot use the DPA and data realignment circuit when the deserializer is bypassed. The IOE contains two data input registers that can operate in DDR or SDR mode.

Figure 8–12. Deserializer Bypass *(Note 1)*, *(2)*, *(3)*



Notes to Figure 8–12:

(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 modes, the data width from the IOE is 1 and 2 bits, respectively.
(4) Arria II GX center/corner PLL or Arria II GZ left/right PLL.

Receiver Datapath Modes

Arria II devices support the following three receiver datapath modes:

- "Non-DPA"
- "DPA Mode"
- "Soft CDR Mode"

Non-DPA

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 8–13 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 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.

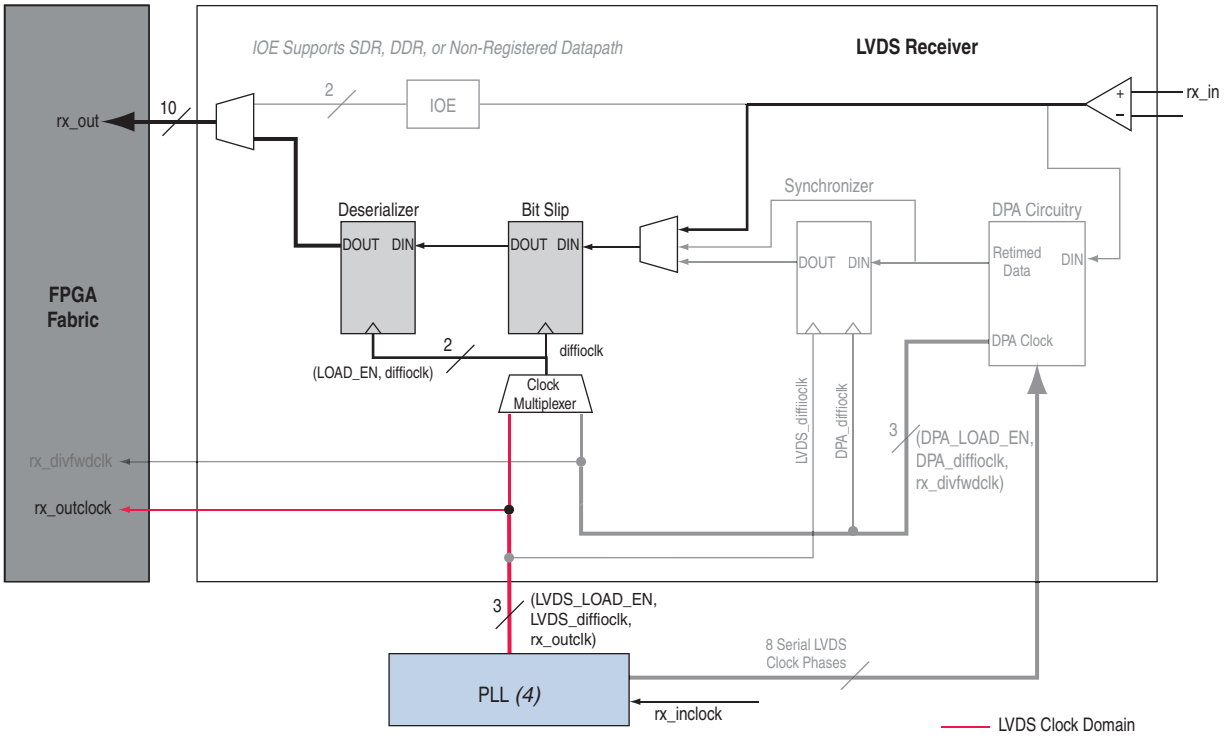
For Arria II GX devices, you must perform PCB trace compensation to adjust the trace length of each LVDS channel to improve channel-to-channel skew when interfacing with non-DPA receivers at data rate above 840 Mbps.

The Quartus II software Fitter Report panel reports the amount of delay you must add to each trace for the Arria II GX device. You can use the recommended trace delay numbers published under the LVDS Transmitter/Receiver Package Skew Compensation panel and manually compensate the skew on the PCB board trace to reduce channel-to-channel skew, thus meeting the timing budget between LVDS channels.



For more information about the LVDS Transmitter/Receiver Package Skew Compensation report panel, refer to the "Arria II GX LVDS Package Skew Compensation Report Panel" section in the *SERDES Transmitter/Receiver (ALTLVDS) Megafunction User Guide*.

Figure 8–13. Receiver Datapath in Non-DPA Mode ***(Note 1)***, ***(2)***, ***(3)***



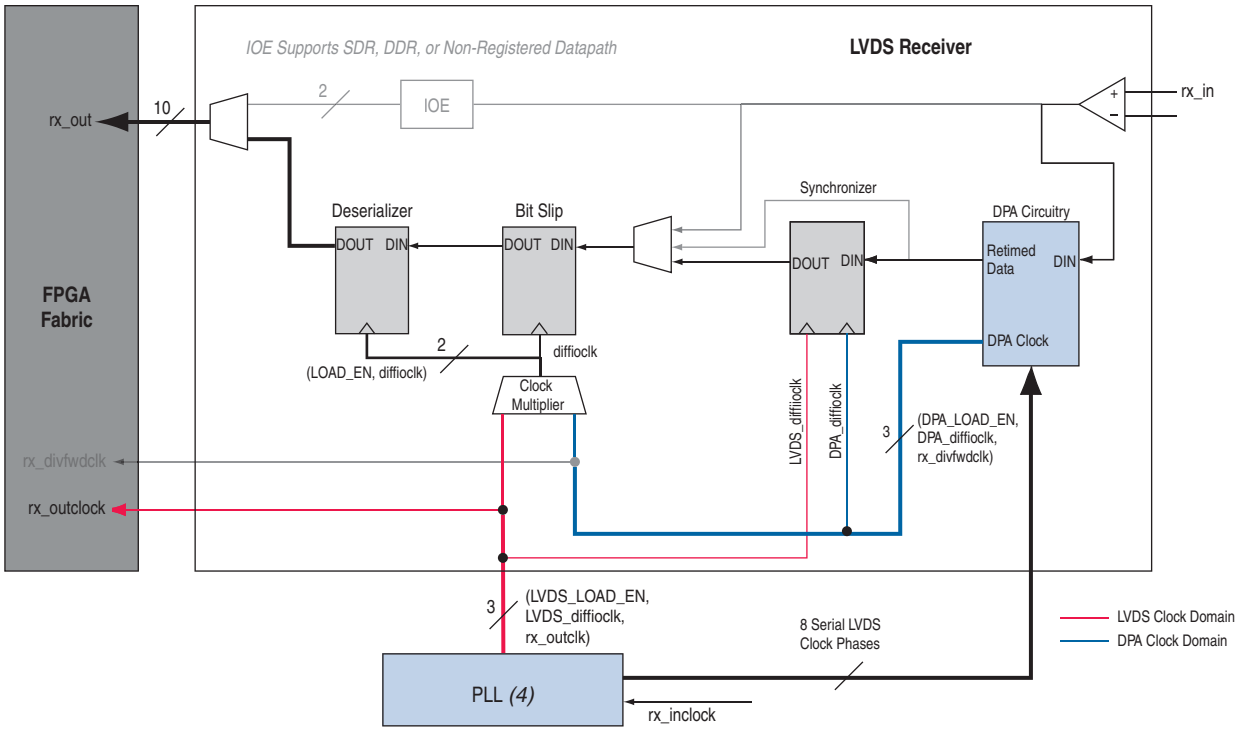
Notes to Figure 8–13:

(1) All disabled blocks and signals are grayed out.
(2) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.
(3) The `rx_out` port has a maximum data width of 10 bits.
(4) Arria II GX center/corner PLL or Arria II GZ left/right PLL.

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 8–14 shows the DPA mode datapath. Use the `DPA_diffioclk` clock to write serial data into the synchronizer. Use the `LVDS_diffioclk` clock to read the serial data from the synchronizer. Use the same `LVDS_diffioclk` clock in the data realignment and deserializer blocks.

Figure 8–14. Receiver Datapath in DPA Mode ***(Note 1), (2), (3)***



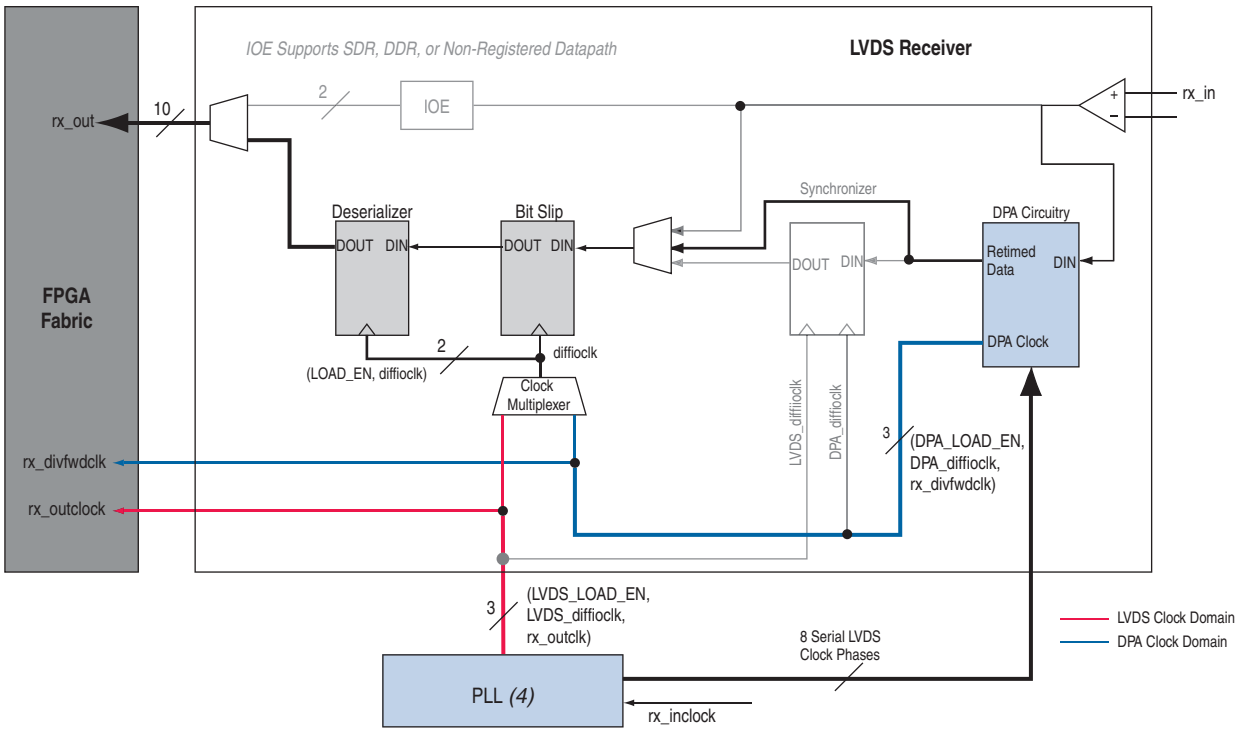
Notes to Figure 8–14:

(1) All disabled blocks and signals are grayed out.
(2) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.
(3) The `rx_out` port has a maximum data width of 10 bits.
(4) Arria II GX center/corner PLL or Arria II GZ left/right PLL.

Soft CDR Mode

Figure 8–15 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 circuitry continuously changes its phase to track the parts per million (ppm) difference between the upstream transmitter and the local receiver reference input clocks. Use the `DPA_diffioclk` clock 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 8–15. Receiver Datapath in Soft CDR Mode ***(Note 1)***, ***(2)***, ***(3)***



Notes to Figure 8–15:

(1) All disabled blocks and signals are grayed out.
(2) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.
(3) The `rx_out` port has a maximum data width of 10 bits.
(4) Arria II GX center/corner PLL or Arria II GZ left/right PLL.

Differential I/O Termination

The Arria II device family provides a 100-Ω R_D OCT option on each differential receiver channel for LVDS standards. OCT saves board space by eliminating the need to add external resistors on the board. You can enable OCT in the Quartus II software Assignment Editor.

For Arria II GX devices, OCT is supported in the top, right, and bottom I/O banks. Arria II GX clock input pins (`CLK[4..15]`) do not support OCT. For Arria II GZ devices, R_D OCT is supported on all row I/O pins and dedicated clock input pins (`CLK[0,2,9,11]`). It is not supported for column I/O pins and dedicated clock input pins (`CLK[1,3,8,10]`).

Figure 8–16 shows LVDS input OCT.

Figure 8–16. LVDS Input Buffer I/O R_D OCT

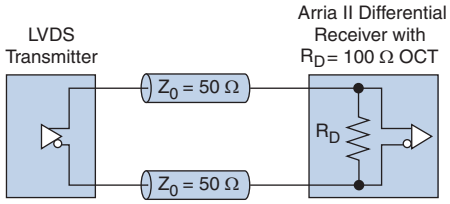


Table 8–7 lists the assignment name and its value for differential input OCT in the Quartus II software Assignment Editor.

Table 8–7. Differential Input OCT in Quartus II Software Assignment Editor

Assignment Name	Assignment Value
Input Termination (Accepts wildcards/groups)	Differential

For more information, refer to *I/O Features in Arria II Devices* chapter.

PLLs

Arria II GX devices contain up to six PLLs with up to four center and corner PLLs located on the right side of the device. Use the center/corner PLL on the right side of the device 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.

Arria II GZ devices contain up to four left and right PLLs with up to two PLLs located on the left side and two on the right side of the device. The left PLLs can support high-speed differential I/O banks on the left side; the right PLLs can support high-speed differential I/O banks on the right side of the device. The high-speed differential I/O receiver and transmitter channels use these left and right PLLs to generate the parallel clocks (`rx_outclock` and `tx_outclock`) and high-speed clocks (`diffioclk`). Figure 8–2 on page 8–4 shows the locations of the left and right PLLs for Arria II GZ devices. The PLL VCO operates at the clock frequency of the data rate. Clock switchover and dynamic reconfiguration are allowed using the left and right PLL in high-speed differential I/O support mode.

For more information about PLLs, refer to the *Clock Network and PLLs in Arria II Devices* chapter.

LVDS and DPA Clock Networks

Arria II GX devices only have 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 8–17 and Figure 8–18 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 8–17. LVDS and DPA Clock Networks in the Arria II GX Devices without Center PLLs

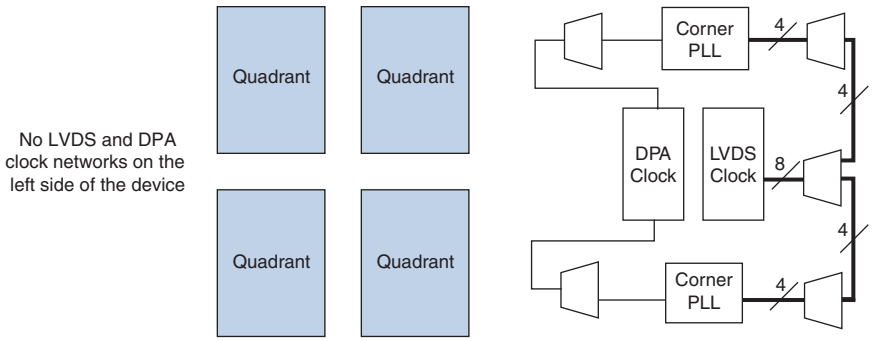
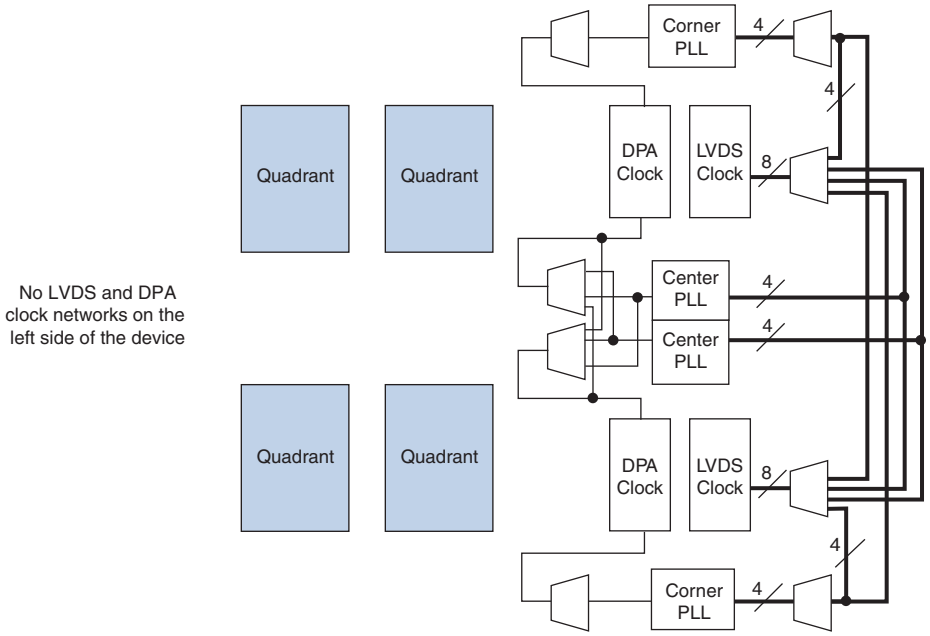


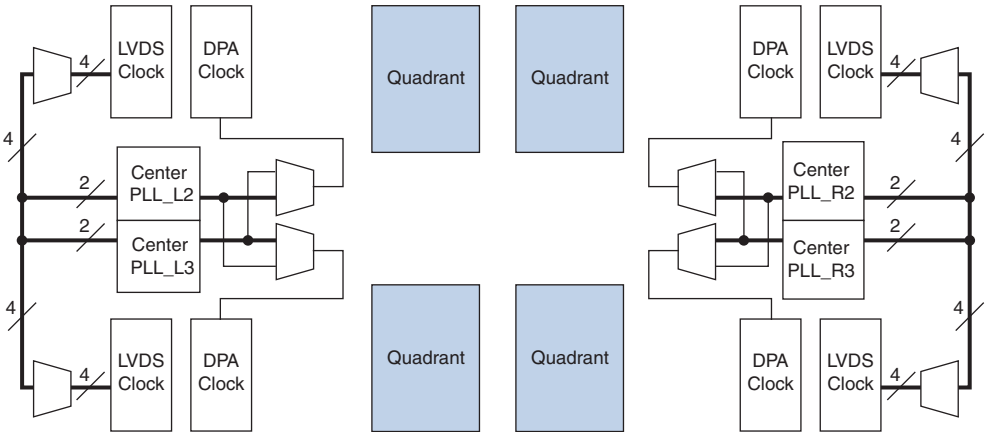
Figure 8–18. LVDS and DPA Clock Networks in the Arria II GX Devices with Center PLLs



Arria II GZ devices have left and right PLLs that feed into the differential transmitter and receive channels through the LVDS and DPA clock network. The center left and right PLLs can clock the transmitter and receive channels above and below them.

Figure 8–19 shows center PLL clocking in Arria II GZ devices.

Figure 8–19. LVDS/DPA Clocks in the Arria II GZ Devices with Center PLLs



For more information about Arria II devices PLL clocking restrictions, refer to "Differential Pin Placement Guidelines" on page 8–27.

Source-Synchronous Timing Budget

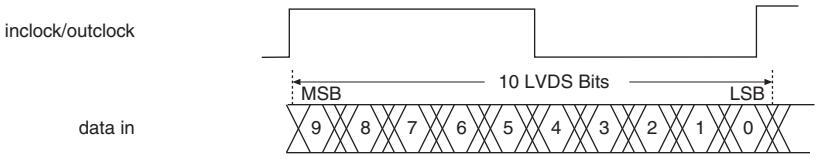
This section describes the timing budget, waveforms, and specifications for source-synchronous signaling in Arria II devices. Timing analysis for the differential block is different from traditional synchronous timing analysis techniques. Therefore, it is important to understand how to analyze timing 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 8–20 shows the data bit orientation of x10 mode.

Figure 8–20. Bit Orientation



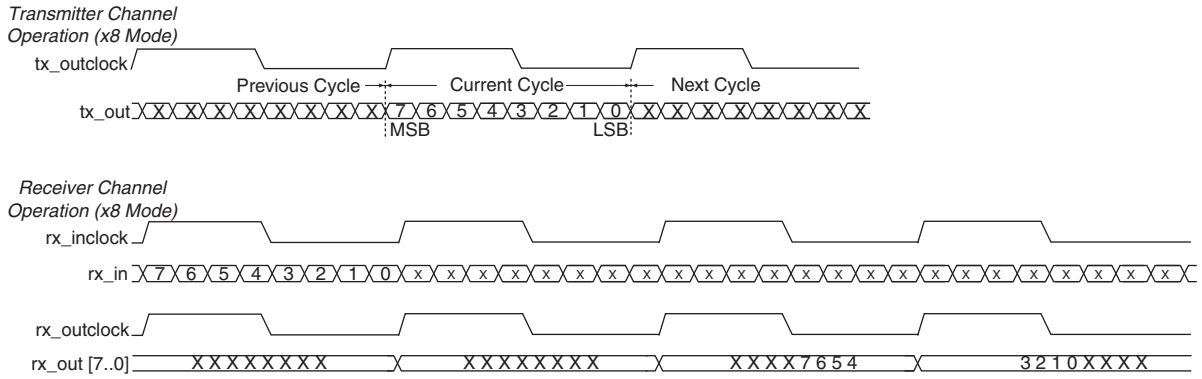
Differential I/O Bit Position

Data synchronization is necessary for successful data transmission at high frequencies. Figure 8–21 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 to find the bit position in the word. The bit positions after deserialization are listed in Table 8–8.

Figure 8–21. Bit Order and Word Boundary for One Differential Channel ***(Note 1)***



Note to Figure 8–21:

(1) These waveforms are only functional waveforms and are not intended to convey timing information.

Table 8–8 lists 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 8–8. 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

Transmitter Channel-to-Channel Skew

Transmitter channel-to-channel skew (TCCS) is an important parameter based on the Arria II transmitter in a source synchronous differential interface. This parameter is used in receiver skew margin calculation.

TCCS is the difference between the fastest and slowest data output transitions, including the TCO variation and clock skew. For LVDS transmitters, the TimeQuest Timing Analyzer provides a TCCS report, which shows TCCS values for serial output ports.

You can get the TCCS value from the TCCS report (`report_TCCS`) in the Quartus II compilation report under the TimeQuest analyzer or from the *Arria II Device Data Sheet* chapter.

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, use RSKM, TCCS, and sampling window (SW) specifications for high-speed source-synchronous differential signals in the receiver datapath. The relationship between RSKM, TCCS, and SW is expressed by the RSKM equation shown in Equation 8–1:

Equation 8–1.

$$RSKM = \frac{TUI - SW - TCCS}{2}$$

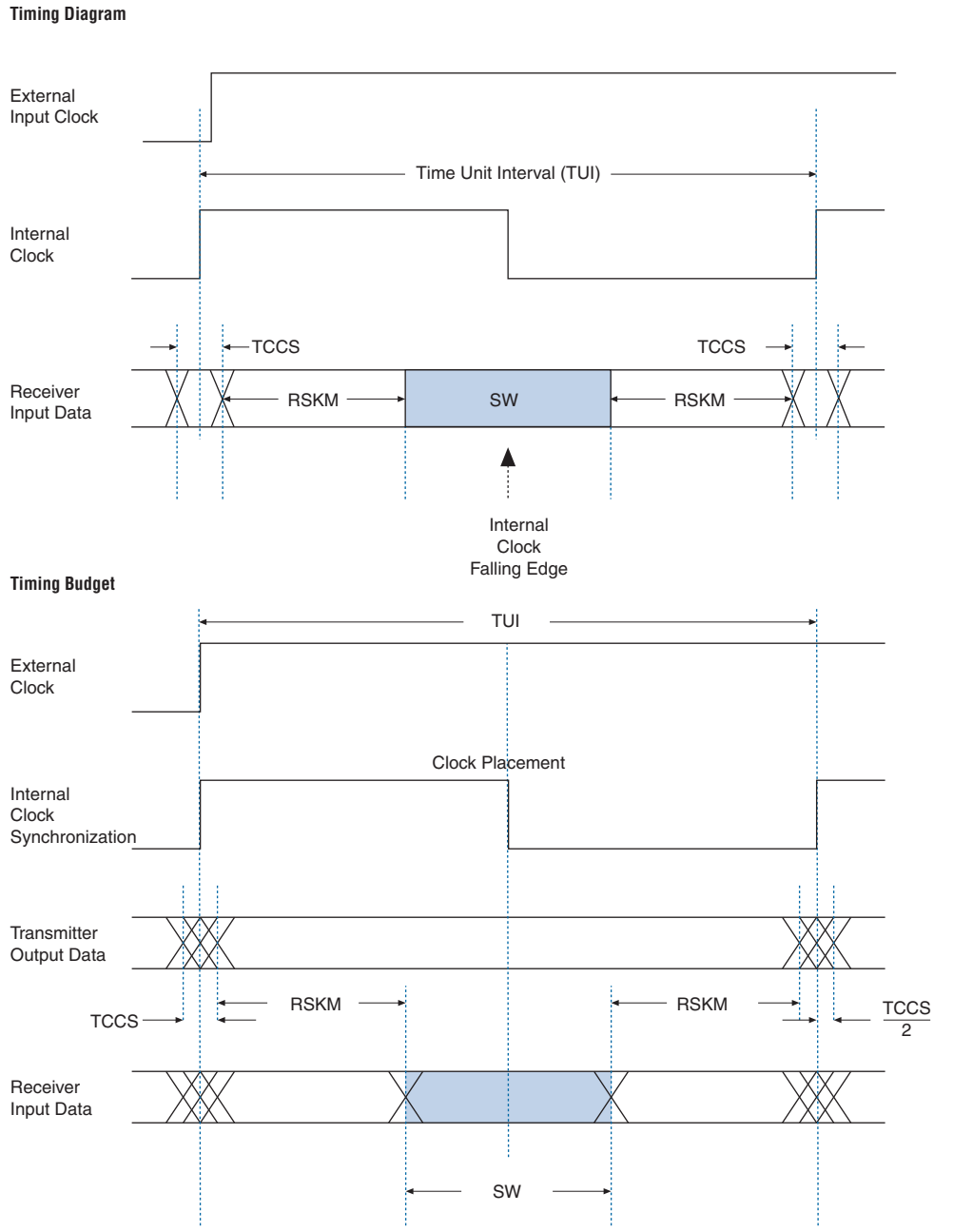
Where:

- 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 the data is successfully sampled by the LVDS receiver. The sampling window is the device property and varies with the 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 the 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 8–22 shows the relationship between the RSKM, TCCS, and SW.

Figure 8–22. Differential High-Speed Timing Diagram and Timing Budget for Non-DPA Mode



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 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 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.



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.

- Single-ended I/Os are allowed in the same I/O bank, if the single-ended I/O standard uses the same V_{CCIO} as the DPA-enabled differential I/O bank.
- Single-ended inputs can be in the same logic array block (LAB) row as a differential channel using the SERDES circuitry.
- Double data rate I/O (DDIO) can be placed within the same LAB row as a SERDES differential channel but half rate DDIO or single data rate (SDR) output pins cannot be placed within the same LAB row as a receiver SERDES differential channel. The input register must be implemented within the FPGA fabric logic.

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

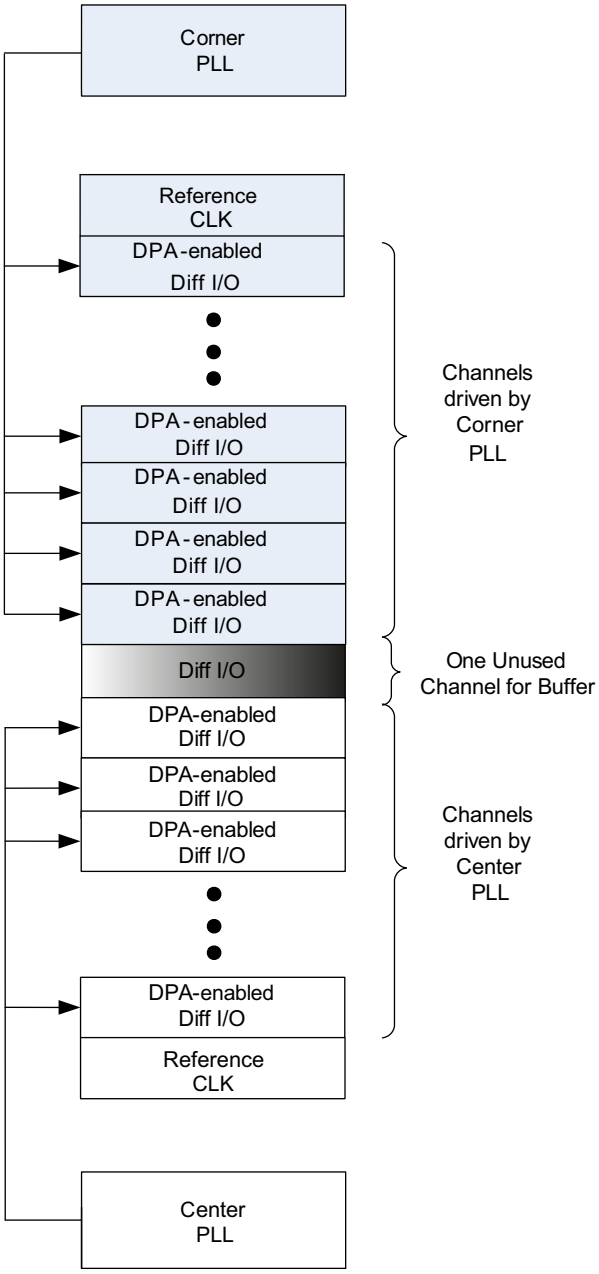
If the number of DPA-enabled channels driven by each center or corner PLL exceeds 25 LAB rows, Altera recommends implementing data realignment (bit slip) circuitry for all the DPA channels.

Using Center and Corner Left and Right PLLs in Arria II GX Devices

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 8–23. This separation prevents 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 8–23. Center and Corner PLLs Driving DPA-Enabled Differential I/Os in the Same Bank



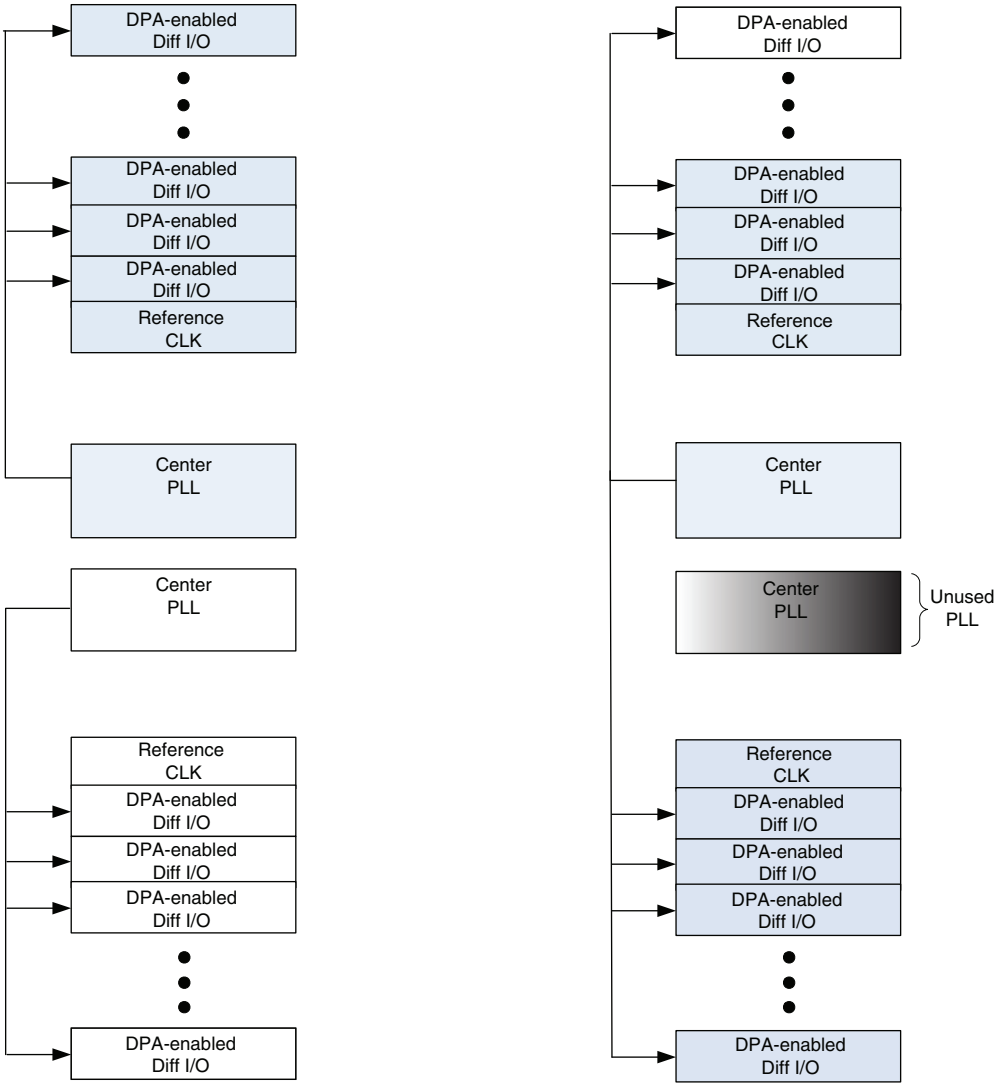
Using Both Center PLLs

You can use center PLLs to drive DPA-enabled channels simultaneously, if they drive these channels in their adjacent banks only, as shown in Figure 8–23.

☞ Center PLLs are available at the right I/O banks of Arria II GX devices and the right and left I/O banks of Arria II GZ devices.

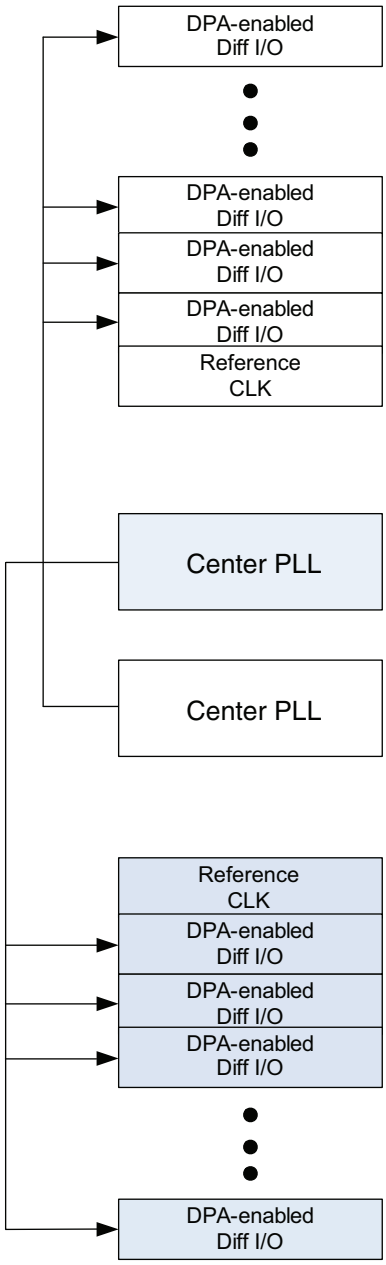
If one of the center PLLs drives the DPA-enabled channels in the upper and lower I/O banks, you cannot use the other center PLL for DPA-enabled channels, as shown in Figure 8–24.

Figure 8–24. 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 8–25.

Figure 8–25. Invalid Placement of DPA-Disabled Differential I/Os Driven by Both Center PLLs



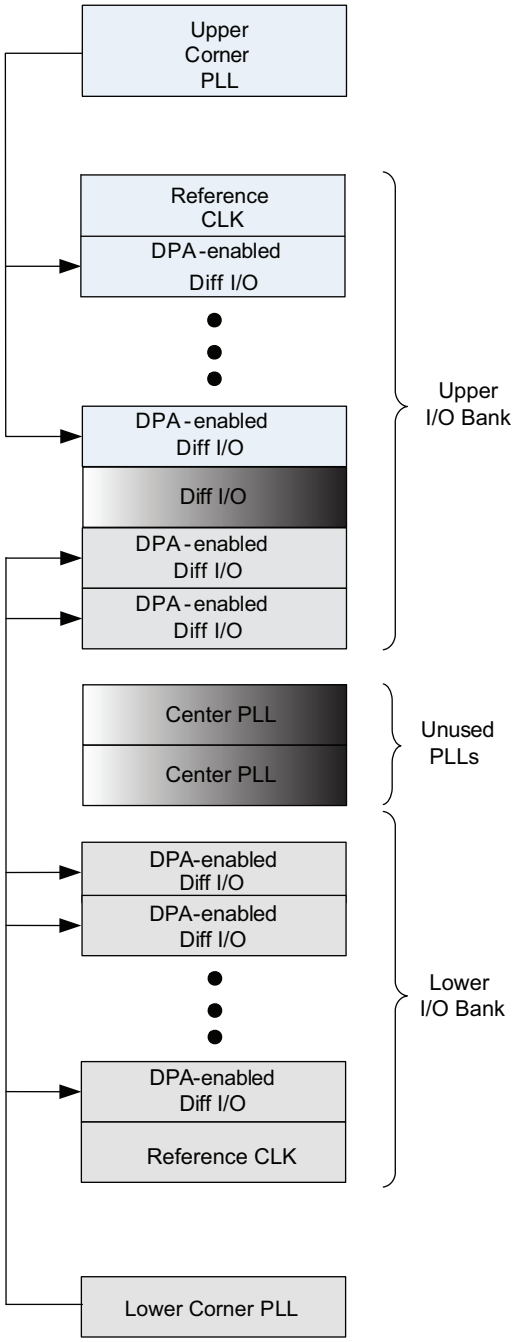
Using Both Corner PLLs in Arria II GX Devices

You can use both corner PLLs to drive DPA-enabled channels simultaneously, if 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, you cannot use the center PLLs. You can use the other corner PLL 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.

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 8–26.

Figure 8–26. 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.

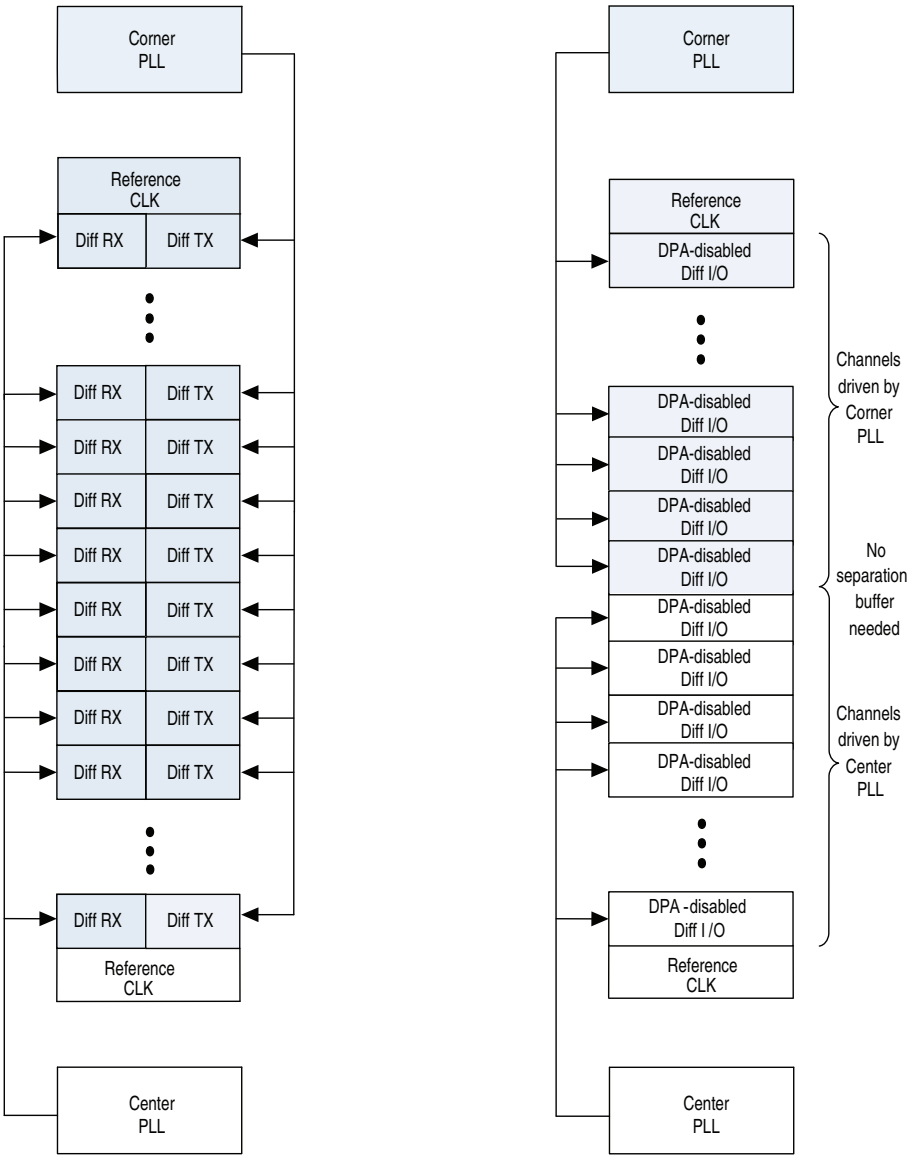
DPA-Disabled Channel Driving Distance

Each PLL can drive all the DPA-disabled channels in the entire bank.

Using Corner and Center PLLs in Arria II GX Devices

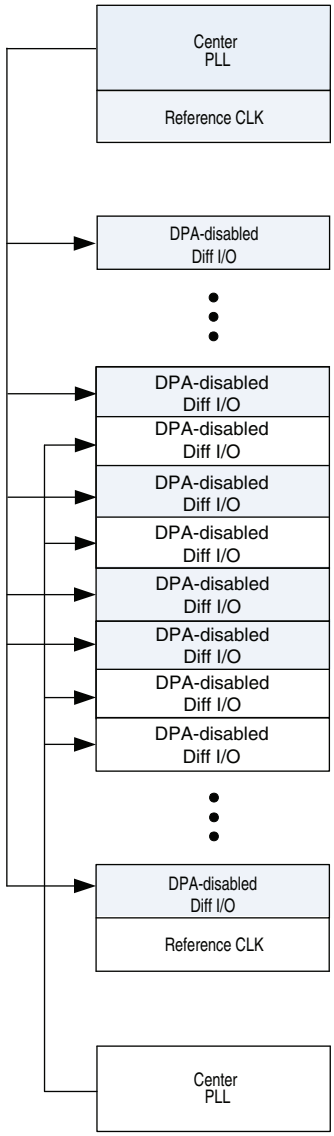
You can use a corner PLL to drive all transmitter channels and you can use a center PLL to drive all DPA-disabled receiver channels in the same I/O bank. In other words, you can drive a transmitter channel and a receiver channel in the same LAB row by two different PLLs, as shown in Figure 8–27.

Figure 8–27. Corner and Center PLLs Driving DPA-Disabled Differential I/Os in the Same Bank



A corner PLL and a center PLL can drive duplex channels in the same I/O bank, if the channels driven by each PLL are not interleaved. No separation is necessary between the group of channels driven by the corner and center left and right PLLs. Refer to Figure 8–27 and Figure 8–28.

Figure 8–28. Invalid Placement of DPA-Disabled Differential I/Os Due to Interleaving of Channels Driven by the Corner and Center PLLs

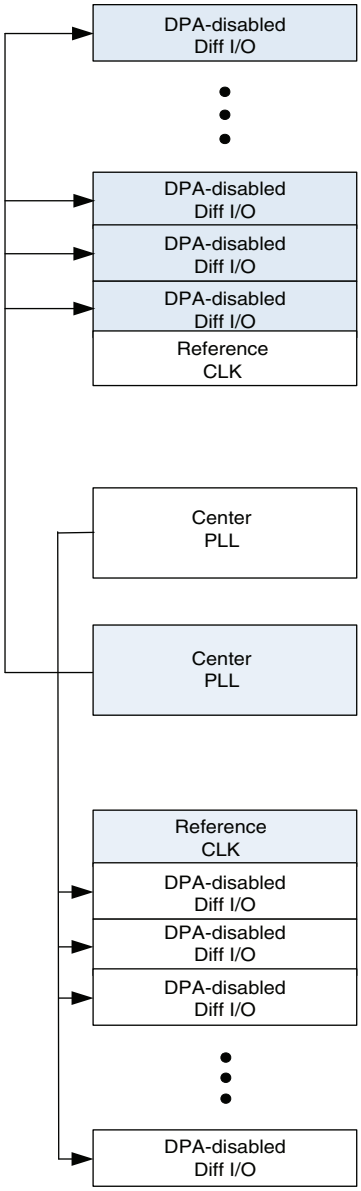


Using Both Center PLLs

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 8–29.

☞ Center PLLs are available at the right I/O banks of Arria II GX devices and the right and left I/O banks of Arria II GZ devices.

Figure 8–29. Both Center PLLs Driving Cross-Bank DPA-Disabled Channels Simultaneously



Using Both Corner PLLs in Arria II GX Devices

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 if 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

The 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 megafunction simplifies the clocking setup for the LVDS transmitter or receiver channels. However, the drawback is reduced flexibility when using the PLL.

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 block settings, and detailed descriptions of the LVDS transmitter and receiver interface signals, refer to the *SERDES Transmitter/Receiver (ALTLVDS) Megafunction User Guide*.

For more information about the ALTPLL megafunction, refer to the *Phase Locked-Loops (ALTPLL) Megafunction User Guide*.

Document Revision History

Table 8–9 lists the revision history for this chapter.

Table 8–9. Document Revision History (Part 1 of 2)

Date	Version	Changes Made
July 2012	4.3	Updated Figure 8–23.
December 2011	4.2	■ Updated “Differential Receiver” section. ■ Minor text edits.
June 2011	4.1	■ Updated Figure 8–2. ■ Minor text edits.
December 2010	4.0	Updated for the Quartus II software version 10.1 release: ■ Added Arria II GZ device information. ■ Updated Table 8–3 and Table 8–4. ■ Updated Figure 8–2.

Table 8–9. Document Revision History (Part 2 of 2)

Date	Version	Changes Made
July 2010	3.0	Updated for Arria II GX v10.0 release: ■ Updated Table 8–1 and Table 8–2. ■ Updated Figure 8–1 and Figure 8–5. ■ Updated "Non-DPA Mode" section. ■ Removed Table 8–1: Supported Data Range. ■ Minor text edit.
November 2009	2.0	Updated for Arria II GX v9.1 release: ■ Updated Table 8–1 and Table 8–2. ■ Updated Figure 8–1. ■ Updated "LVDS Channels" and "Non-DPA Mode" sections. ■ Minor text edit.
June 2009	1.1	■ Updated Table 8–2 and Table 8–3. ■ Updated "Programmable Pre-Emphasis and Programmable VOD." and "LVDS Channels" sections.
February 2009	1.0	Initial release

Section III. System Integration for Arria II Devices

This section provides information about Arria® II device configuration, design security, remote system upgrades, SEU mitigation, JTAG, and power requirements. This section includes the following chapters:

- Chapter 9, Configuration, Design Security, and Remote System Upgrades in Arria II Devices
- Chapter 10, SEU Mitigation in Arria II Devices
- Chapter 11, JTAG Boundary-Scan Testing in Arria II Devices
- Chapter 12, Power Management in Arria II 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.

9. Configuration, Design Security, and Remote System Upgrades in Arria II Devices

AIIGX51009-4.3

This chapter describes the supported configuration schemes for Arria® II devices, instructions for executing the required configuration schemes, and the necessary option pin settings. This chapter also reviews the different ways you can configure your device and explains the design security and remote system upgrade features for Arria II devices.

Arria II devices use SRAM cells to store configuration data. Because SRAM memory is volatile, you must download configuration data to the Arria II device each time the device powers up. All configuration schemes use either an external controller (for example, a MAX® II device or microprocessor), a configuration device, or a download cable.

This chapter includes the following sections:

- "Configuration Features"
- "Power-On Reset Circuit and Configuration Pins Power Supply" on page 9–4
- "Configuration Process" on page 9–7
- "Configuration Schemes" on page 9–9
- "Fast Passive Parallel Configuration" on page 9–11
- "AS and Fast AS Configuration (Serial Configuration Devices)" on page 9–19
- "PS Configuration" on page 9–26
- "JTAG Configuration" on page 9–33
- "Device Configuration Pins" on page 9–39
- "Configuration Data Decompression" on page 9–46
- "Remote System Upgrades" on page 9–48
- "Remote System Upgrade Mode" on page 9–52
- "Dedicated Remote System Upgrade Circuitry" on page 9–55
- "Quartus II Software Support" on page 9–60
- "Design Security" on page 9–61

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Configuration Devices

Altera® serial configuration devices support single- and multi-device configuration solutions for Arria II devices. Arria II GX devices use the active serial (AS) configuration scheme while Arria II GZ devices use the fast AS configuration scheme. Serial configuration devices offer a low-cost, low pin-count configuration solution.

For more information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Datasheet* in volume 2 of the *Configuration Handbook*.

All minimum timing information stated in this chapter covers the entire Arria II device family. Some devices may work at less than the minimum timing stated in this chapter due to process variations.

Configuration Features

Arria II devices offer decompression, design security, and remote system upgrade features. Arria II 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 protects your designs. You can make real-time system upgrades from remote locations of your Arria II designs with the remote system upgrade feature.

Table 9–1 lists the configuration features you can use in each configuration scheme for Arria II GX devices.

Table 9–1. Configuration Features for Arria II GX Devices

Configuration Scheme	Configuration Method	Decompression	Design Security	Remote System Upgrade
FPP	MAX II device or a microprocessor with flash memory	✓ *(1)*	✓ *(1)*	—
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 x4 the data rate.

(2) Remote system upgrade is only available in the AS configuration scheme. Local update mode is not supported in the AS configuration scheme.

Table 9–2 lists the configuration features you can use in each configuration scheme for Arria II GZ devices.

Table 9–2. Configuration Features for Arria II GZ Devices

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 9–2:

(1) In these modes, the host system must send a DCLK that is x4 the data rate.

(2) Remote system upgrade is only available in the fast AS configuration scheme. Local update mode is not supported in the fast AS configuration scheme.

Refer to the following for the configuration features supported in Arria II devices:

- For more information about the configuration data decompression feature, refer to "Configuration Data Decompression" on page 9–46.
- For more information about the remote system upgrade feature, refer to "Remote System Upgrades" on page 9–48.
- For more information about the design security feature, refer to the "Design Security" on page 9–61.
- For more information about the parallel flash loader (PFL), refer to *Parallel Flash Loader Megafunction User Guide*.

If your system already contains a common flash interface (CFI) flash memory device, you can also use it for the Arria II device configuration storage. The PFL feature in MAX II devices provides an efficient method to program CFI flash memory devices through the JTAG interface and provides the logic to control configuration from the flash memory device to the Arria II device. Both passive serial (PS) and fast passive parallel (FPP) configuration modes are supported using this PFL feature.

For more information about programming Altera serial configuration devices, refer to "Programming Serial Configuration Devices" on page 9–24.

Power-On Reset Circuit and Configuration Pins Power Supply

The following sections describe 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. After power-up, the device does not release nSTATUS until the voltage levels are above the POR trip point of the device. Table 9–3 lists the voltages required for power-up in Arria II devices.

Table 9–3. Required Voltages for Arria II Devices

Devices	Voltages
Arria II GX	V_{CCCB}, V_{CCA_PLL}, V_{CC}, V_{CCPD}, and V_{CCIO} for I/O banks 3C or 8C
Arria II GZ	V_{CC}, V_{CCAUX}, V_{CCCB}, V_{CCPGM}, and V_{CCPD}

On power down for Arria II GX devices, brown-out occurs if V_{CC} ramps down below the POR trip point and any of the V_{CC}, V_{CCPD}, or V_{CCIO} voltages for I/O banks 3C or 8C drops below the threshold. On power down for Arria II GZ devices, brown-out occurs if the V_{CC}, V_{CCAUX}, V_{CCCB}, V_{CCPGM}, or V_{CCPD} voltages drops below the threshold voltage.

In Arria II devices, you can select between a fast POR time or a standard POR time. For Arria II GX devices, selection depends on the MSEL pin settings. For Arria II GZ devices, selection depends on the PORSEL input pin. PORSEL = L is set as standard POR time. PORSEL = H is set as fast POR time. Fast POR time is **4 ms < T_{POR} < 12 ms** for a fast configuration time. Standard POR time is **100 ms < T_{POR} < 300 ms** for a lower power-ramp rate.

Configuration Pins Power Supply

Table 9–4 lists the configuration pins for Arria II devices.

Table 9–4. Configuration pins for Arria II Devices

Devices	Configuration Pins
Arria II GX	All dedicated configuration pins are supplied by 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. Use the V_{CCIO} pin for I/O banks 3C and 8C to power all the dedicated configuration inputs, dedicated configuration outputs, and dedicated configuration bidirectional pins that you used for configuration. With V_{CCIO} for I/O banks 3C and 8C, the configuration input buffers do not have to the share power lines with the regular I/O buffer. You must power the dual function configuration pins that you used for configuration with the V_{CCIO} power supply in which the configuration pins reside. For more information about the configuration voltage standard applied to the V_{CCIO} power supply, refer to Table 9–6 on page 9–9.
Arria II GZ	All dedicated configuration pins and dual-function pins are supplied by V_{CCPGM}. The supported configuration voltages are 1.8, 2.5, and 3.0 V. Use the V_{CCPGM} pin to power all the dedicated configuration inputs, dedicated configuration outputs, and dedicated configuration bidirectional pins that you used for configuration. With V_{CCPGM}, the configuration input buffers do not have to share the power lines with the regular I/O buffer.

Arria II devices do not support a 1.5-V configuration. The operating voltage for the configuration input pin is independent of the I/O banks power supply V_{CCIO} during configuration. Therefore, for Arria II devices, you do not require configuration voltage constraints on V_{CCIO}.

For more information, refer to the *Power Management in Arria II Devices* chapter.

For more information about the configuration pins connection recommendations, refer to the *Arria II Device Family Pin Connection Guidelines*.

V_{CCPD} Pins

Arria II devices have a dedicated programming power supply, the V_{CCPD} pins. Table 9–5 lists the power supply for Arria II devices.

Table 9–5. Power Supply for Arria II Devices

Devices	Programming Power Supply
Arria II GX	V_{CCPD} must be connected to 3.3, 3.0, or 2.5 V to power the I/O pre-drivers, HSTL/SSTL input buffers, and 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 within 100 ms when PORSEL is low or 4 ms when PORSEL is high. If these supplies are not ramped up in this specified time, your Arria II GX device will not configure successfully. If the system cannot ramp up the power supplies within 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.
Arria II GZ	V_{CCPD} must be connected to 3.0 or 2.5 V to power the I/O pre-drivers and JTAG I/O pins (TCK, TMS, TDI, TDO, and TRST). V_{CCPD} and V_{CCPGM} must ramp up from 0 V to the desired voltage level within 100 ms when PORSEL is low or 4 ms when PORSEL is high. If these supplies are not ramped up in this specified time, your Arria II GZ device will not configure successfully. If the system cannot ramp up the power supplies within 100 ms or 4 ms, you must hold nCONFIG low until all the power supplies are stable. V_{CCPD} must be greater than or equal to V_{CCIO} of the same bank: ■ If the V_{CCIO} of the bank is powered to 3.0 V, V_{CCPD} must be powered up to 3.0 V. ■ If the V_{CCIO} of the bank is powered to 2.5 V or lower, V_{CCPD} must be powered up to 2.5 V.

For more information about configuration pins power supply, refer to "Device Configuration Pins" on page 9–39.

Configuration Process

The following sections describe the general configuration process for FPP, standard AS, fast AS, and PS schemes.

Power Up

To begin the configuration process, you must fully power the relevant voltage supply to the appropriate voltage levels.



For an FPP configuration in Arria II GX devices, the DATA[7..1] pins are supplied by V_{CCIO} for I/O bank 6A. You must power up this bank when you use the FPP configuration. For Arria II GZ devices, the DATA[7..1] pins are powered up by V_{CCPGM} during configuration or by V_{CCIO} if they are used as regular I/Os in user mode.

Reset

After power up, the Arria II device goes through a POR. The POR delay depends on the MSEL pin settings. During POR, the device resets, holds nSTATUS low, clears the configuration RAM bits, and tri-states all user I/O pins. After 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. Use the dedicated DATA[0] pin for both PS and AS 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 the I/O pins are tri-stated. After nCONFIG and nSTATUS return to a logic-high level, configuration begins.

A pull-up or pull-down resistor helps keep the nCONFIG line in a known state when the external host (a Max® II CPLD or a microcontroller) is not driving the line. For example, during external host reprogramming or power-up where the I/O driving nCONFIG may be tri-stated). If a pull-up resistor is added to the nCONFIG line, the FPGA stays in user mode if the external host is being reprogrammed. If a pull-down resistor is added to the nCONFIG line, the FPGA goes into reset mode if the external host is being reprogrammed. Whenever the nCONFIG line is released high, ensure the first DCLK and DATA are not driven unintentionally.

☞ Altera recommends to keep the nCONFIG line low if the external host or the FPGA is not ready for configuration.

Configuration Error

If an error occurs during configuration, Arria II devices assert the nSTATUS signal low, indicating a data frame error; the CONF_DONE signal stays low. If you turn on 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), the Arria II device resets the configuration device and retries the configuration. If you turn off this option, the system must monitor nSTATUS for errors and then pulse nCONFIG low to restart the configuration.

Initialization

In Arria II 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 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 does 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 devices require a minimum number of clock cycles to initialize properly and enter user mode as specified in the t_{CD2UMC} parameter.

☞ Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device for both uncompressed and compressed bitstream in the FPP or PS configuration mode.

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. After 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 devices.

MSEL Pin Settings

Select the configuration scheme by driving the Arria II device MSEL pins either high or low, as listed in Table 9–6 and Table 9–7. The MSEL input buffers are powered by the V_{CCPD} and V_{CCPGM} power supplies for Arria II GX and GZ devices, respectively. Altera recommends hardwiring the MSEL[] pins to V_{CCPD}/V_{CCPGM} 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}/V_{CCPGM} or GND without pull-up or pull-down resistors. Do not drive the MSEL[] pins by a microprocessor or another device.



For Figure 9–1 on page 9–12 through Figure 9–30 on page 9–66, MSEL[n..0] represents MSEL[3..0] for Arria II GX devices and MSEL[2..0] for Arria II GZ devices as listed in Table 9–6 and Table 9–7, respectively.

Table 9–6. Configuration Schemes for Arria II GX Devices (Part 1 of 2)

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0	POR Delay	Configuration Voltage Standard (V) *(1)*
FPP	0	0	0	0	Fast	3.3, 3.0, 2.5
	0	1	1	1	Fast	1.8
FPP with design security feature, decompression, or both enabled *(2)*	0	0	0	1	Fast	3.3, 3.0, 2.5
	1	0	0	0	Fast	1.8
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

Table 9–6. Configuration Schemes for Arria II GX Devices (Part 2 of 2)

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0	POR Delay	Configuration Voltage Standard (V) *(1)*
AS with or without remote system upgrade	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 *(3)*	*(4)*	*(4)*	*(4)*	*(4)*	—	—

Notes to Table 9–6:

(1) Configuration voltage standard applied to the V_{CCIO} power 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 x4 the data rate.

(3) JTAG-based configuration takes precedence over other configuration schemes, which means the MSEL pin settings are ignored. JTAG-based configuration does not support the design security or decompression features.

(4) Do not leave the 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 connecting the MSEL pins to GND.

Table 9–7 lists the configuration schemes for Arria II GZ devices.

Table 9–7. Configuration Schemes for Arria II GZ Devices

Configuration Scheme	MSEL2	MSEL1	MSEL0	POR Delay	Configuration Voltage Standard (V)
FPP	0	0	0	Fast/Standard	3.0, 2.5, 1.8
PS	0	1	0	Fast/Standard	3.0, 2.5, 1.8
Fast AS (40 MHz) *(1)*	0	1	1	Fast/Standard	3.0, 2.5, 1.8
Remote system upgrade fast AS (40 MHz) *(1)*	0	1	1	Fast/Standard	3.0, 2.5, 1.8
FPP with design security feature and/or decompression enabled *(2)*	0	0	1	Fast/Standard	3.0, 2.5, 1.8
JTAG-based configuration *(3)*	*(4)*	*(4)*	*(4)*	—	—

Notes to Table 9–7:

(1) Arria II GZ devices only support fast AS configuration. You must use either EPCS64 or EPCS128 devices to configure an Arria II GZ device in fast AS mode.

(2) These modes are only supported when using a MAX II device or microprocessor with flash memory for configuration. In these modes, the host system must output a DCLK that is x4 the data rate.

(3) The JTAG-based configuration takes precedence over other configuration schemes, which means the MSEL pin settings are ignored. The JTAG-based configuration does not support the design security or decompression features.

(4) Do not leave the MSEL pins floating, connect them to V_{CCPGM} or GND. These pins support non-JTAG configuration scheme used in production. If you only use the JTAG configuration, Altera recommends connecting the MSEL pins to GND.

Raw Binary File Size

Table 9–8 lists the uncompressed raw binary file (**.rbf**) configuration file sizes for Arria II devices.

Table 9–8. Uncompressed .rbf Sizes for Arria II Devices

Device	Data Size (bits)
EP2AGX45	29,599,704
EP2AGX65	29,599,704
EP2AGX95	50,376,968
EP2AGX125	50,376,968
EP2AGX190	86,866,440
EP2AGX260	86,866,440
EP2AGZ225	94,557,472
EP2AGZ300	128,395,584
EP2AGZ350	128,395,584

Use the data in Table 9–8 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. For the different types of configuration file and file sizes, refer to the Quartus II software. However, for a specific version of the Quartus II software, any design targeted for the same device has the same uncompressed configuration file size. If you are using compression, the file size can vary after each compilation because the compression ratio depends 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

FPP configuration in Arria II devices is designed to meet the continuously increasing demand for faster configuration times. Arria II devices are designed with the capability of receiving byte-wide configuration data per clock cycle.

You can perform FPP configuration of Arria II devices using an intelligent host such as a MAX II device or microprocessor.

FPP Configuration Using a MAX II Device as an External Host

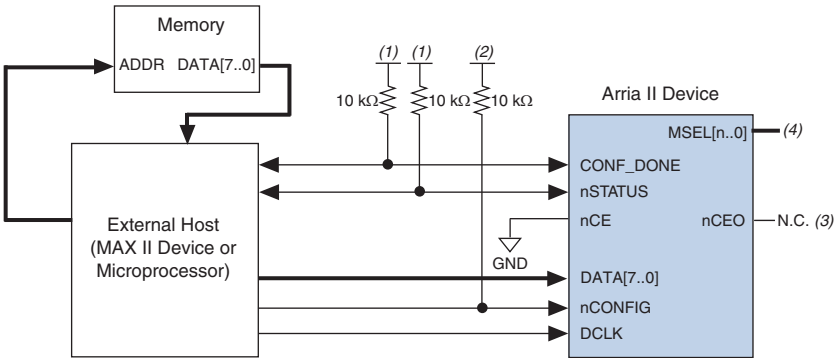
FPP configuration using an external host provides the fastest method to configure Arria II devices. In this configuration scheme, you can use a MAX II device or microprocessor as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Arria II 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 use the Arria II decompression and/or design security features, the external host must send a DCLK frequency that is x4 the data rate.

The x4 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. For Arria II GX devices, if you are not using the decompression or design security features, the data rate is x1 the DCLK frequency. For Arria II GZ devices, if you are not using the decompression or design security features, the data rate is x8 the DCLK frequency.

Figure 9–1 shows the configuration interface connections between an Arria II 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 pull-up resistor to a supply that provides an acceptable input signal for the Arria II device. For Arria II GX devices, use the V_{CCIO} pin. For Arria II GZ devices, use the V_{CCPGM} pin. V_{CCIO}/V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on both the device and the external host. Altera recommends powering up the configuration system I/Os with V_{CCIO}/V_{CCPGM}.

(2) A pull-up resistor to V_{CCIO}/V_{CCPGM} or a pull-down resistor keeps the nCONFIG line in a known state when the external host is not driving the line.

(3) You can leave the nCEO pin unconnected or used as a user I/O pin when it does not feed the nCE pin of the other device.

(4) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

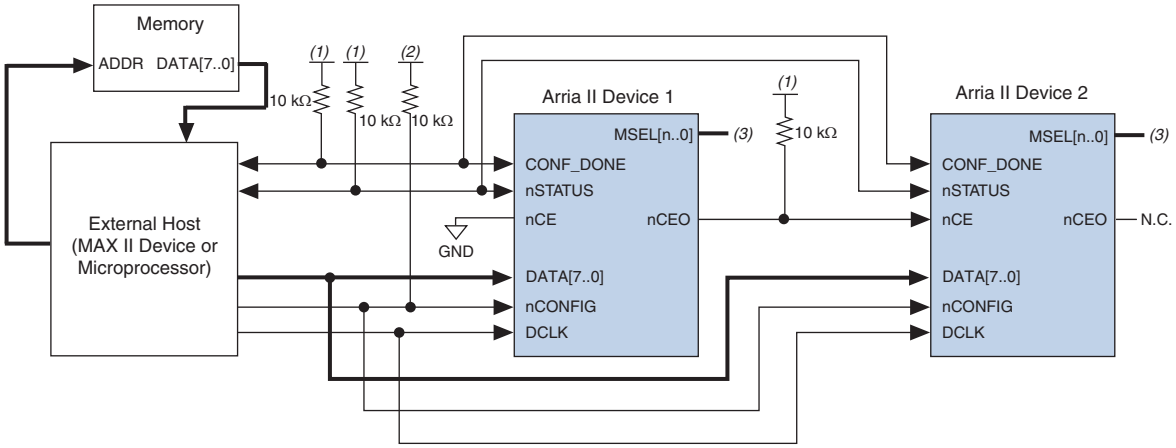
Arria II devices receive configuration data on the DATA[7..0] pins and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. If you are using the Arria II decompression, design security, or both features, configuration data is latched on the rising edge of every first DCLK cycle out of the four DCLK cycles. Altera recommends keeping the data on DATA[7..0] stable for the next three clock cycles while the data is being processed. You can only stop DCLK three clock cycles after the last data is latched.

In Arria II 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 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 does not affect the configuration process. Arria II devices support an f_{MAX} of 125 MHz.

Figure 9–2 shows how to configure multiple Arria II devices using a MAX II device. This circuit is similar to the FPP configuration circuit for a single device, except the Arria II 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 the Arria II device. For Arria II GX devices, use the V_{CCIO} pin. For Arria II GZ devices, use the V_{CCPGM} pin. V_{CCIO}/V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on both the device and the external host. Altera recommends powering up the configuration system I/Os with V_{CCIO}/V_{CCPGM}.

(2) A pull-up resistor to V_{CCIO}/V_{CCPGM} or a pull-down resistor keeps the nCONFIG line in a known state when the external host is not driving the line.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

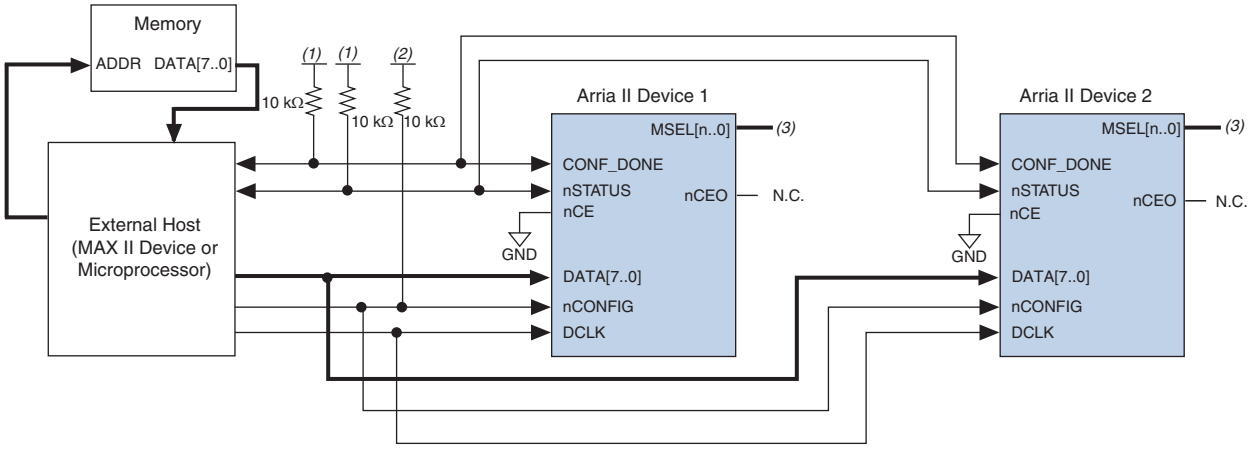
After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the nCE pin of the second device, which prompts the second device to begin configuration. The second device in the chain begins configuration in one clock cycle; therefore, the transfer 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 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 the Arria II device. For Arria II GX devices, use the V_{CCIO} pin. For Arria II GZ devices, use the V_{CCPGM} pin. V_{CCIO}/V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on both the device and the external host. Altera recommends powering up the configuration system I/Os with V_{CCIO}/V_{CCPGM}.

(2) A pull-up resistor to V_{CCIO}/V_{CCPGM} or a pull-down resistor keeps the nCONFIG line in a known state when the external host is not driving the line.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

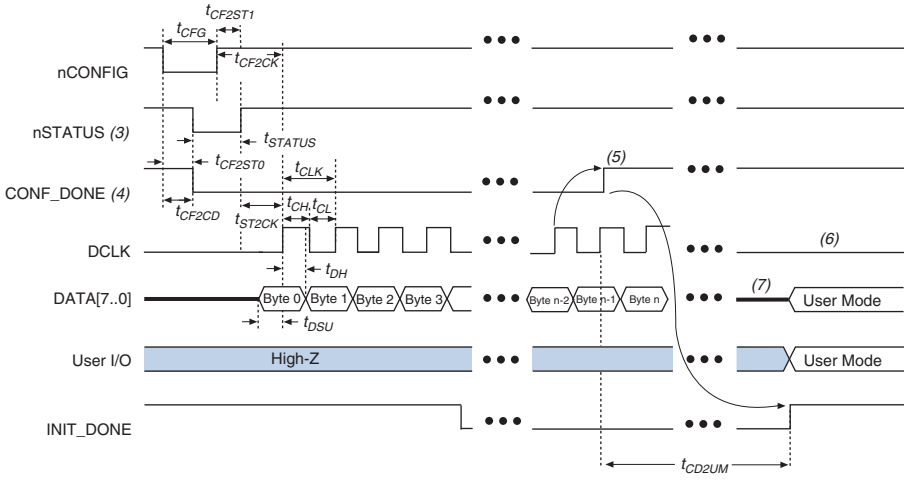
You can use a single configuration chain to configure Arria II devices with other Altera devices that support FPP configuration. To ensure that all devices in the chain complete configuration at the same time, or that an error flagged by one device initiates reconfiguration in all devices, tie all of the 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 an 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 you do not use the decompression and design security features.

(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) After power-up, the Arria II device holds nSTATUS low for the time of the POR delay.

(4) After power-up, before and during configuration, CONF_DONE is low.

(5) Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device.

(6) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(7) DATA[7..1] are available as user I/O pins after configuration. The state of these pins depends on the dual-purpose pin settings. For Arria II GX devices, DATA[0] is a dedicated pin that is used for both the PS and AS configuration modes and is not available as a user I/O pin after configuration. For Arria II GZ devices, DATA[0] is available as a user I/O pin after configuration.

Table 9–9 lists the timing parameters for Arria II devices for an FPP configuration when you do not enable the decompression and design security features.

Table 9–9. FPP Timing Parameters for Arria II Devices with Decompression and Design Security not Enabled *(Note 1)*

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} *(2)*	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 *(4)*	—	ns
t_{CH}	DCLK high time	3.2 *(4)*	—	ns
t_{CL}	DCLK low time	3.2 *(4)*	—	ns
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 *(5)*	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 9–9:

(1) Use these timing parameters when you do not enable the decompression and design security features.

(2) This value is applicable if you do not delay configuration by externally holding the nSTATUS low.

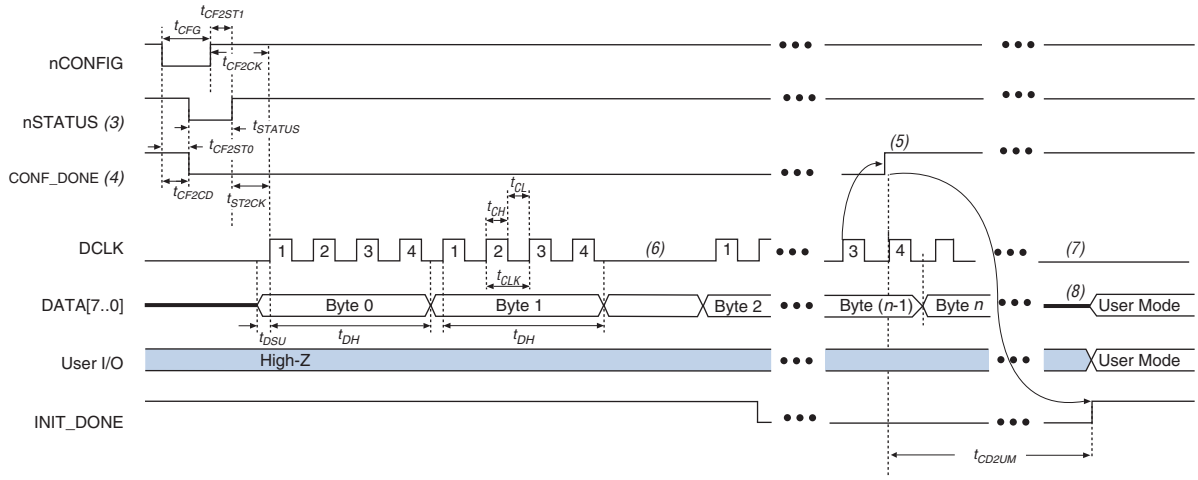
(3) You can obtain this value if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(4) The values listed for t_{DH}, t_{CH}, and t_{CL} are applicable only for Arria II GX devices. For Arria II GZ devices, t_{DH} = 1 ns, t_{CH} = 3.6 ns, and t_{CL} = 3.6 ns, respectively.

(5) The minimum and maximum numbers apply only if you chose the internal oscillator as the clock source for initializing the device.

Figure 9–5 shows the timing waveform for an FPP configuration when using a MAX II device or microprocessor as an external host. This waveform shows timing when you enable the decompression, the design security features, or both.

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 you use the decompression and/or design security features.

(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) After power-up, the Arria II GX device holds nSTATUS low for the time of the POR delay.

(4) After power-up, before and during configuration, CONF_DONE is low.

(5) Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device.

(6) If required, 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.

(7) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(8) DATA[7..1] are available as user I/O pins after configuration. The state of these pins depends on the dual-purpose pin settings. For Arria II GX devices, DATA[0] is a dedicated pin that is used for both the PS and AS configuration modes and is not available as a user I/O pin after configuration. For Arria II GZ devices, DATA[0] is available as a user I/O pin after configuration.

Table 9–10 lists the timing parameters for Arria II devices for an FPP configuration when you enable the decompression, the design security features, or both.

Table 9–10. FPP Timing Parameters for Arria II GX Devices with the Decompression or Design Security Features Enabled *(Note 1)*

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} *(2)*	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 *(4)*	—	ns
t_{CH}	DCLK high time	3.2 *(4)*	—	ns
t_{CL}	DCLK low time	3.2 *(4)*	—	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 *(5)*	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 9–10:

(1) Use these timing parameters when you enable the decompression and design security features.

(2) This value is applicable if you do not delay configuration by externally holding the nSTATUS low.

(3) You can obtain this value if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(4) The values listed for t_{DH}, t_{CH}, and t_{CL} are applicable only for Arria II GX devices. For Arria II GZ devices, t_{DH} = 3/(DCLK frequency) + 1, t_{CH} = 3.6 ns, and t_{CL} = 3.6 ns, respectively.

(5) The minimum and maximum numbers apply only if you choose the internal oscillator as the clock source for initializing 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*.

AS and Fast AS Configuration (Serial Configuration Devices)

Arria II GX and GZ devices are configured using a serial configuration device in the AS configuration scheme and the fast AS configuration scheme, respectively. 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.

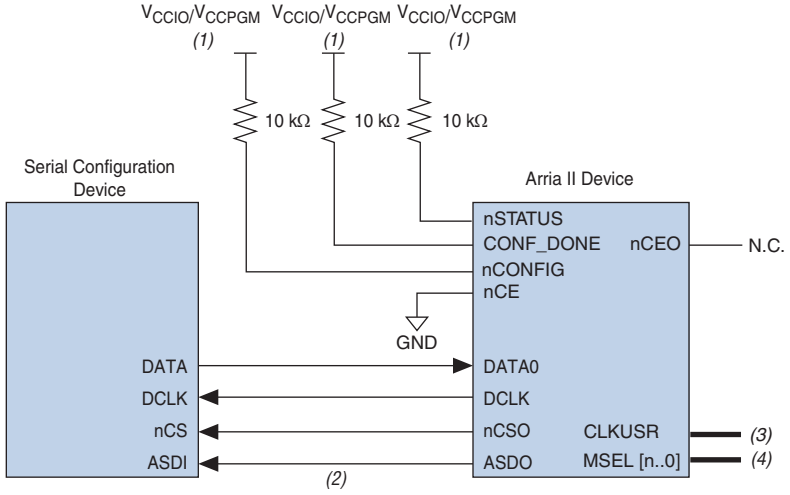
For more information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

Serial configuration devices provide a serial interface to access configuration data. During device configuration, Arria II 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 device controls the configuration interface. This scheme contrasts with the PS configuration scheme, where the configuration device controls the interface.

The Arria II decompression and design security features are available when configuring your Arria II GX device using AS mode and when configuring your Arria II GZ device using fast AS mode.

Serial configuration devices have a four-pin interface—serial clock input (DCLK), serial data output (DATA), AS data input (ASDI), and an active-low chip select (nCS). This four-pin interface connects to the Arria II device pins, as shown in Figure 9–6.

Figure 9–6. Single Device AS Configuration



Notes to Figure 9–6:

(1) Connect the pull-up resistors to the V_{CCIO} power supply of bank 3C for Arria II GX devices and to V_{CCPGM} at a 3.0-V power supply for Arria II GZ devices.

(2) Arria II devices use the ASDO-to-ASDI path to control the configuration device.

(3) Arria II devices have an option to select **CLKUSR** (40 MHz maximum) as the external clock source for DCLK.

(4) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

The serial clock (DCLK) generated by the Arria II device controls the entire configuration cycle and provides timing for the serial interface. During the configuration, Arria II devices use an internal oscillator or an external clock source to generate DCLK. At the initial stage of the configuration cycle, the Arria II device generates a default DCLK (40 MHz maximum) from the internal oscillator to read the header information of the programming data stored in the EPCS. After the header information is read from the EPCS, depending on the clock source being selected, the configuration cycle continues with a slow clock (20 MHz maximum) or a fast clock (40 MHz maximum) from the internal oscillator or an external clock from **CLKUSR** (40 MHz maximum). You can change the clock source option in the Quartus II software from the **Configuration** tab of the **Device and Pin Options** dialog box.



Arria II GZ devices only support fast AS configuration (40 MHz maximum) and do not support a slow clock.

In AS and fast AS configuration schemes, Arria II 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 device on the following falling edge of DCLK.

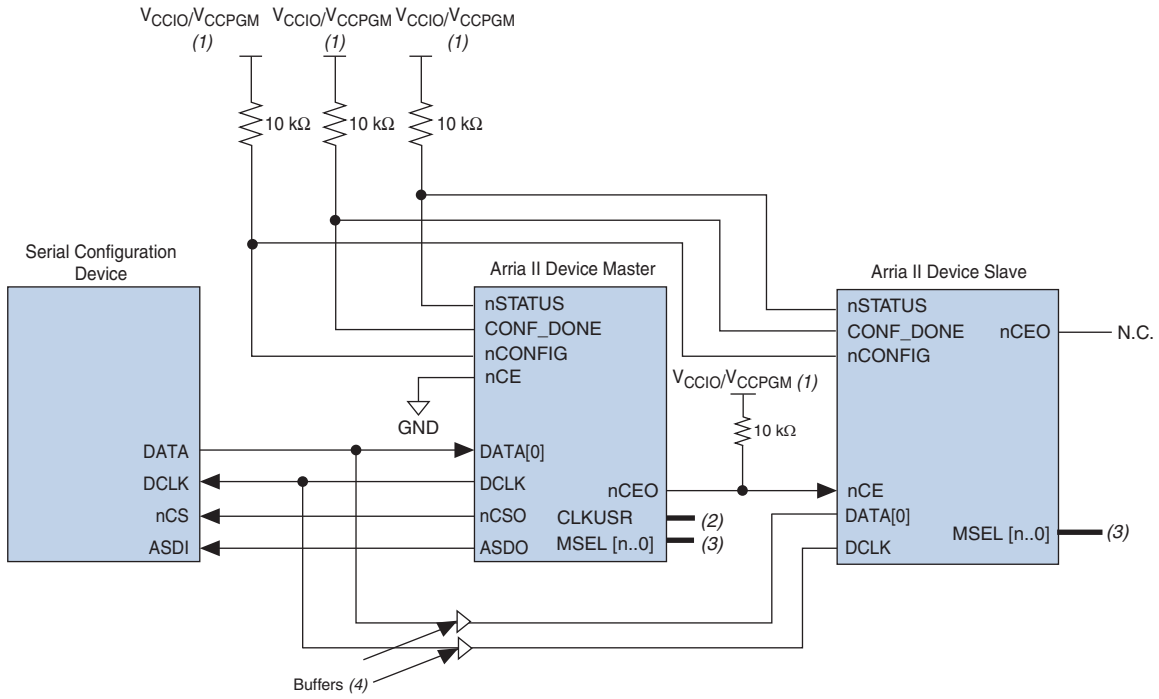
In configuration mode, Arria II devices enable the serial configuration device by driving the nCSO output pin low, which connects to the chip select (nCS) pin of the configuration device. The Arria II device uses the serial clock (DCLK) and serial data output (ASDO) pins to send operation commands, read address signals, or both, 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 devices.

You can configure multiple Arria II devices using a single serial configuration device. Cascade multiple Arria II 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 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 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} power supply of the I/O bank 3C for Arria II GX devices and to V_{CCPGM} at a 3.0-V power supply for Arria II GZ devices.

(2) Arria II devices have an option to select **CLKUSR** (40 MHz maximum) as the external clock source for DCLK.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

(4) Connect the repeater buffers between the Arria II master and slave devices for DATA[0] and DCLK. This is to prevent any potential signal integrity and clock skew problems.

The timing parameters for AS mode are not listed here because the t_{CF2CD}, t_{CF2ST0}, t_{CFG}, t_{STATUS}, t_{CF2ST1}, and t_{CD2UM} timing parameters are identical to the timing parameters for PS mode listed in Table 9–12 on page 9–29.

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 devices, you cannot cascade or chain together serial configuration devices.

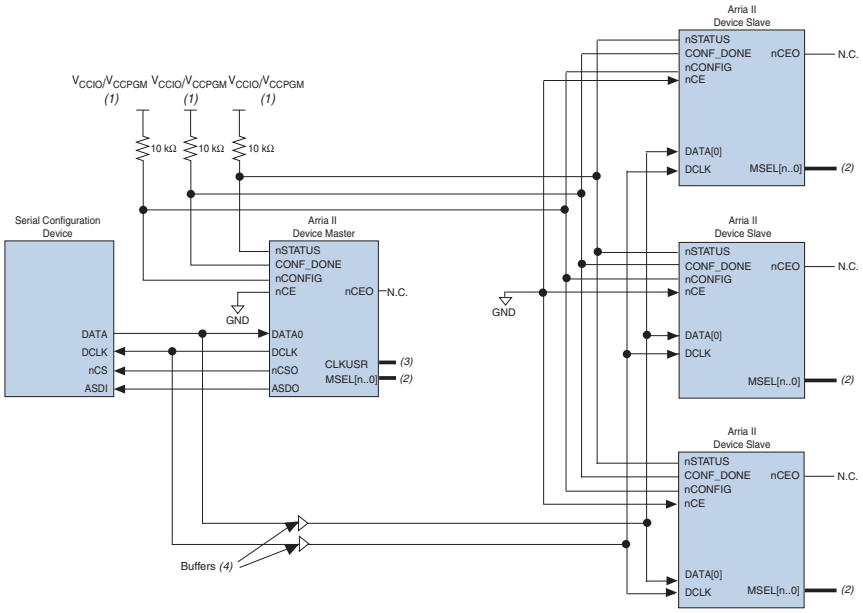
If the configuration bitstream size exceeds the capacity of a serial configuration device, you must select a larger configuration device, enable the compression feature, or both. When configuring multiple devices, the size of the bitstream is the sum of the configuration bitstreams of the individual devices.

A system may have multiple devices that contain the same configuration data. In AS 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** configures the master Arria II device and all remaining slave devices concurrently. All Arria II devices must be the same density and package.

To configure four identical Arria II devices with the same **.sof**, 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 must be set to select AS configuration. The other three slave devices are set up for concurrent configuration and their MSEL pins must 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**.

Figure 9–8. Multi-Device AS Configuration When the Devices Receive the Same Data Using a Single .sof



Notes to Figure 9–8:

(1) Connect the pull-up resistors to the V_{CCIO} power supply of I/O bank 3C for Arria II GX devices and to V_{CCPGM} at a 3.0-V power supply for Arria II GZ devices.

(2) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

(3) Arria II devices have an option to select **CLKUSR** (40 MHz maximum) as the external clock source for DCLK.

(4) Connect the repeater buffers between the Arria II master and slave devices for DATA[0] and DCLK. This is to prevent any potential signal integrity and clock skew problems.

Guidelines for Connecting Serial Configuration Device to Arria II Devices on an AS Interface

For single- and multi-device AS configurations, the board trace length and loading between the supported serial configuration device and the Arria II devices must follow the recommendations listed in Table 9–11.

Table 9–11. Maximum Trace Length and Loading for the AS Configuration in Arria II Devices

Arria II Device AS Pins	Maximum Board Trace Length from the Arria II Device to the Serial Configuration Device (Inches)	Maximum Board Load (pF)
DCLK	10	15
DATA[0]	10	30
nCSO	10	30
ASDO	10	30

Estimating the AS Configuration Time

AS configuration time is dominated by the time it takes to transfer data from the serial configuration device to the Arria II device. This serial interface is clocked by the Arria II DCLK output (generated from an internal oscillator or an option to select **CLKUSR** as external clock source). Arria II devices support DCLK up to 40 MHz (25 ns).

Therefore, you can estimate the minimum configuration time 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 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, EthernetBlaster II, or ByteBlaster™ II download cables. Alternatively, you can program them using a microprocessor with the SRunner software driver.



To gain control of the serial configuration device pins, hold the nCONFIG pin low and pull the nCE pin high. This causes the device to reset and tri-state the AS configuration pins.

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 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 logic high.

Altera has developed the serial flash loader (SFL); an in-system programming solution for serial configuration devices using the JTAG interface. This solution requires the Arria II 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*.

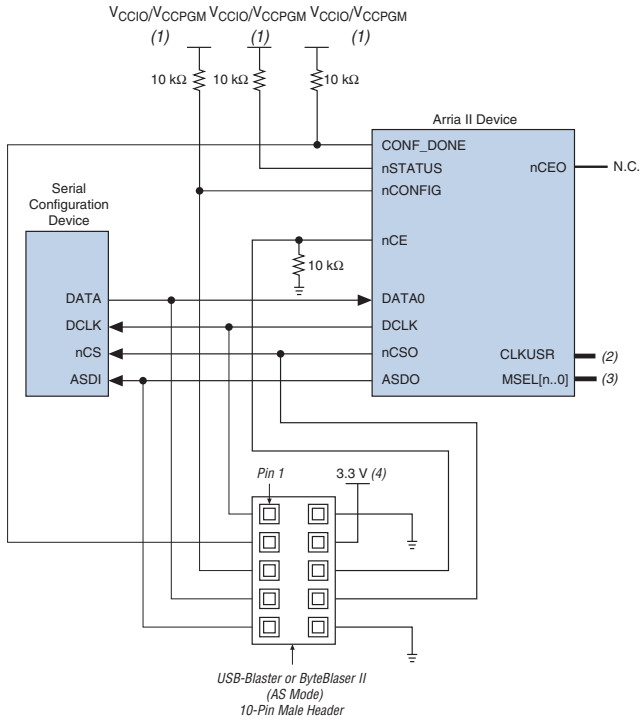


For more information, refer to the following:

- *ByteBlaster II Download Cable User Guide*
- *EthernetBlaster Communications Cable User Guide*
- *EthernetBlaster II Communications Cable User Guide*
- *USB-Blaster Download 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} power supply of the I/O bank 3C for Arria II GX devices and to V_{CCPGM} at a 3.0-V power supply for Arria II GZ devices.

(2) Arria II devices have an option to select **CLKUSR** (40 MHz maximum) as the external clock source for `DCLK`.

(3) The `MSEL` pin settings vary for different configuration voltage standards and POR delay. To connect `MSEL[3..0]` for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect `MSEL[2..0]` for an Arria II GZ device, refer to Table 9–7 on page 9–10.

(4) Power up the USB-ByteBlaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II cable's $V_{CC(TRGT)}$ with V_{CCIO} 3.3 V for Arria II GX device and V_{CCPGM} 3.0 V for Arria II GZ device.

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 SRunner 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 file (.**rpd**) 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.

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*.

PS Configuration

You can program a PS configuration of Arria II 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 device using the DATA0 pin at each rising edge of DCLK.

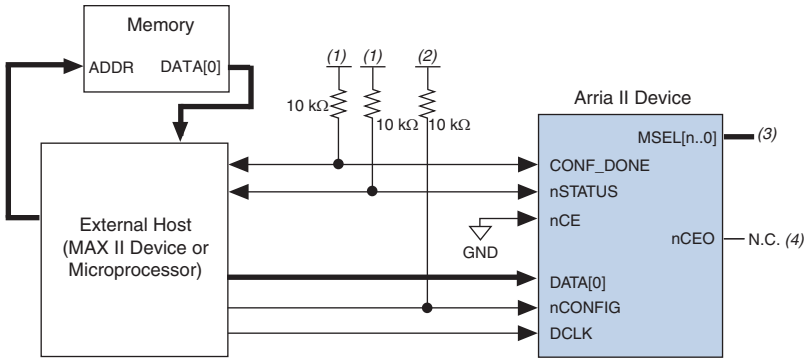
The Arria II decompression and design security features are available when configuring your Arria II device using PS mode.

PS Configuration Using a MAX II Device as 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 device. You can store configuration data in **.rbf**, **.hex**, or **.ttf** format.

Figure 9–10 shows the configuration interface connections between an Arria II device and a MAX II device for single device configuration.

Figure 9–10. Single Device PS Configuration Using an External Host



Notes to Figure 9–10:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the Arria II device. For Arria II GX devices, use the V_{CCIO} pin. For Arria II GZ devices, use the V_{CCPGM} pin. V_{CCIO}/V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on both the device and the external host. Altera recommends powering the configuration system I/Os with V_{CCIO}/V_{CCPGM}.

(2) A pull-up resistor to V_{CCIO}/V_{CCPGM} or a pull-down resistor keeps the nCONFIG line in a known state when the external host is not driving the line.

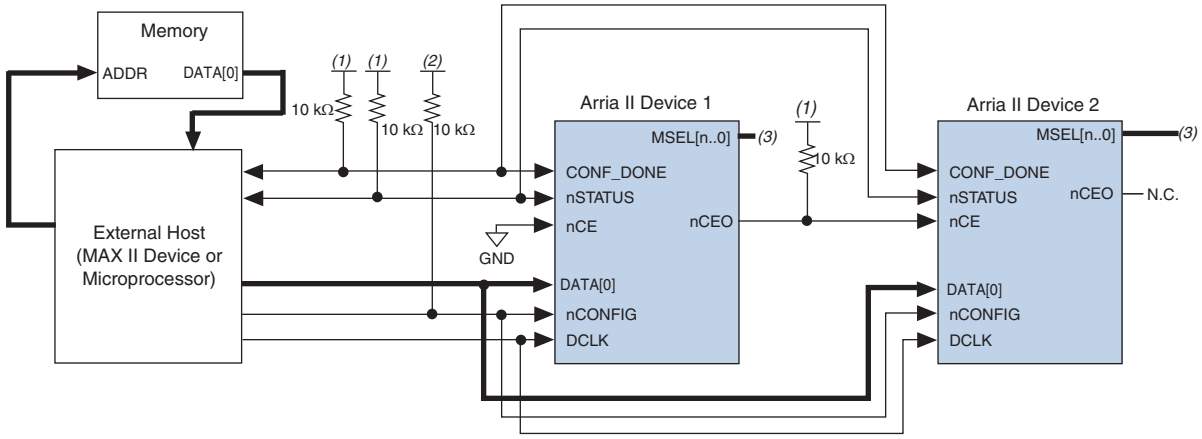
(3) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

(4) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed the nCE pin of the other device.

The Arria II 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** contains the byte sequence 02 1B EE 01 FA, the serial bitstream you must 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 the Arria II 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 the Arria II device. For Arria II GX devices, use the V_{CCIO} pin. For Arria II GZ devices, use the V_{CCPGM} pin. V_{CCIO}/V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on both the device and the external host. Altera recommends powering up the configuration system I/Os with V_{CCIO}/V_{CCPGM}.

(2) A pull-up resistor to V_{CCIO}/V_{CCPGM} or a pull-down resistor keeps the nCONFIG line in a known state when the external host is not driving the line.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

In Arria II 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 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 does not affect the configuration process. Arria II devices support f_{MAX} of 125 MHz.

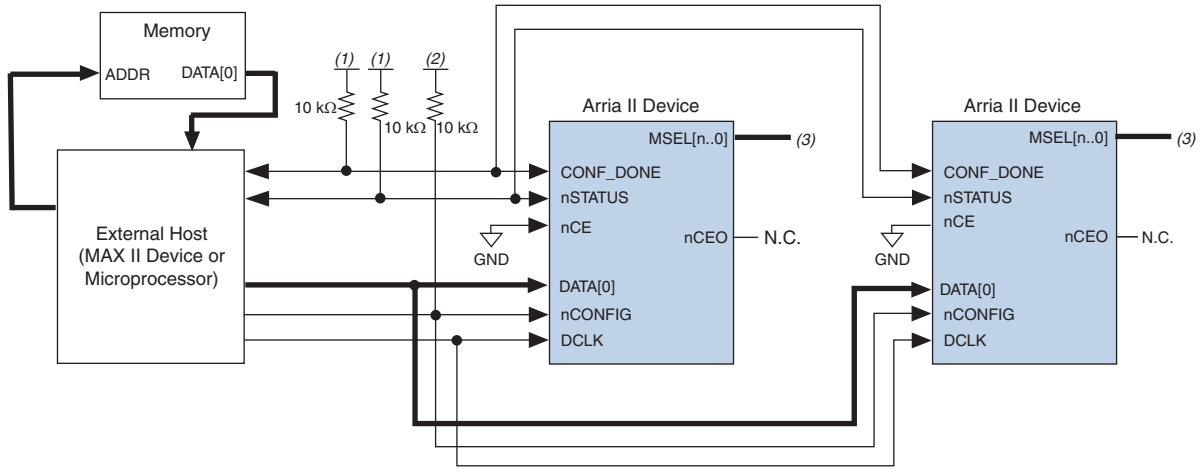
After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration in one clock cycle. Therefore, the transfer 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 the 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 start and complete configuration at the same time.

Figure 9–12 shows a multi-device PS configuration when both Arria II 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 the Arria II device. For Arria II GX devices, use the V_{CCIO} pin. For Arria II GZ devices, use the V_{CCPGM} pin. V_{CCIO}/V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on both the device and the external host. Altera recommends powering up the configuration system I/Os with V_{CCIO}/V_{CCPGM}.

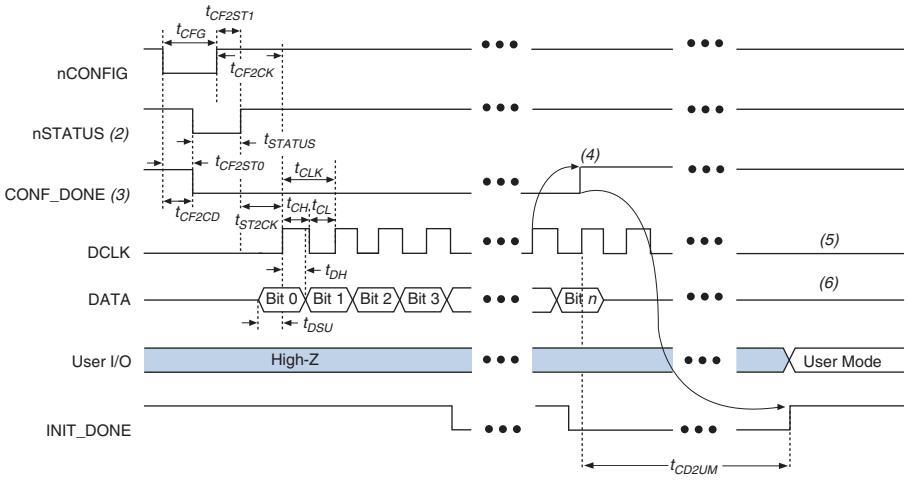
(2) A pull-up resistor to V_{CCIO}/V_{CCPGM} or a pull-down resistor keeps the nCONFIG line in a known state when the external host is not driving the line.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

PS Configuration Timing

Figure 9–13 shows the timing waveform for a 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) After power-up, the Arria II device holds nSTATUS low for the time of the POR delay.

(3) After power-up, before and during configuration, CONF_DONE is low.

(4) Two DCLK falling edges are required after CONF_DONE goes high to begin initialization of the device.

(5) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(6) For Arria II GX devices, DATA[0] is a dedicated pin that is used for both PS and AS configuration modes and is not available as a user I/O pin after configuration. For Arria II GZ devices, DATA[0] is available as a user I/O pin after configuration.

Table 9–12 lists the timing parameters for Arria II devices for PS configuration.

Table 9–12. PS Timing Parameters for Arria II Devices (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} *(1)*	nCONFIG high to nSTATUS high	—	500 *(2)*	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

Table 9–12. PS Timing Parameters for Arria II Devices (Part 2 of 2)

Symbol	Parameter	Minimum	Maximum	Units
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 *(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 9–12:

(1) This value is applicable if you do not delay configuration by externally holding the nSTATUS low.

(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 initializing 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 USB port download cable, ByteBlaster II parallel port download cable, EthernetBlaster download cable, and EthernetBlaster II download cable.

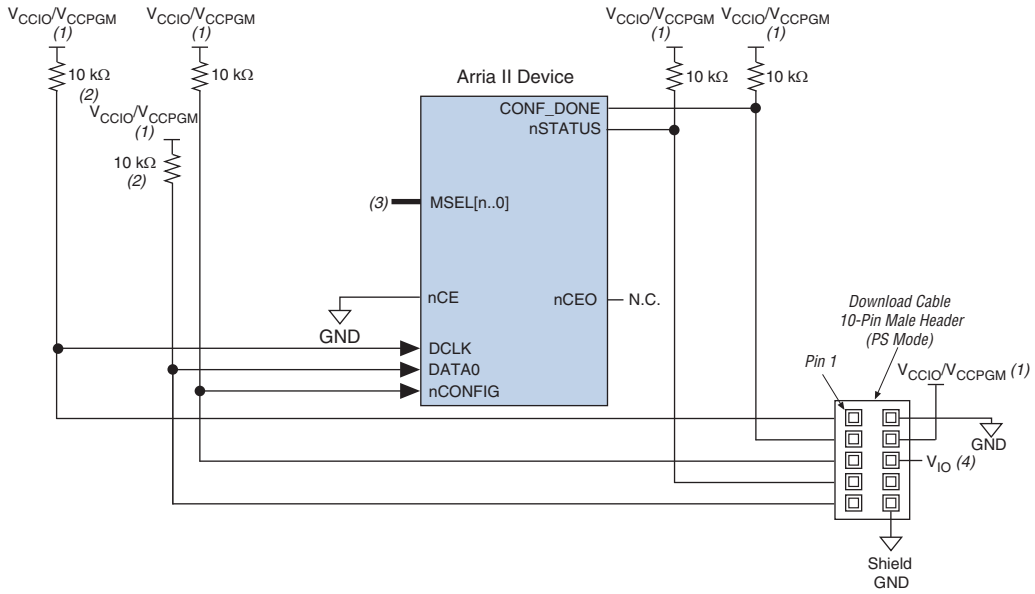
In a PS configuration with a download cable, an intelligent host (such as a PC) transfers data from a storage device to the Arria II device using the download cable.

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** when programming the device using the Quartus II programmer and download cable. Therefore, if you turn on the **CLKUSR** option, you are not required to provide a clock on the CLKUSR pin when you are configuring the device with the Quartus II programmer and a download cable.

Figure 9–14 shows a PS configuration for Arria II devices using a USB-Blaster, EthernetBlaster, EthernetBlaster II, or ByteBlaster II cable.

Figure 9–14. PS Configuration Using a USB-Blaster, EthernetBlaster, EthernetBlaster II, or ByteBlaster II Cable



Notes to Figure 9–14:

(1) Connect the pull-up resistor to the same supply voltage, V_{CCIO} for Arria II GX devices or V_{CCPGM} for Arria II GZ devices as the USB-Blaster, EthernetBlaster, EthernetBlaster II, or ByteBlaster II 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) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

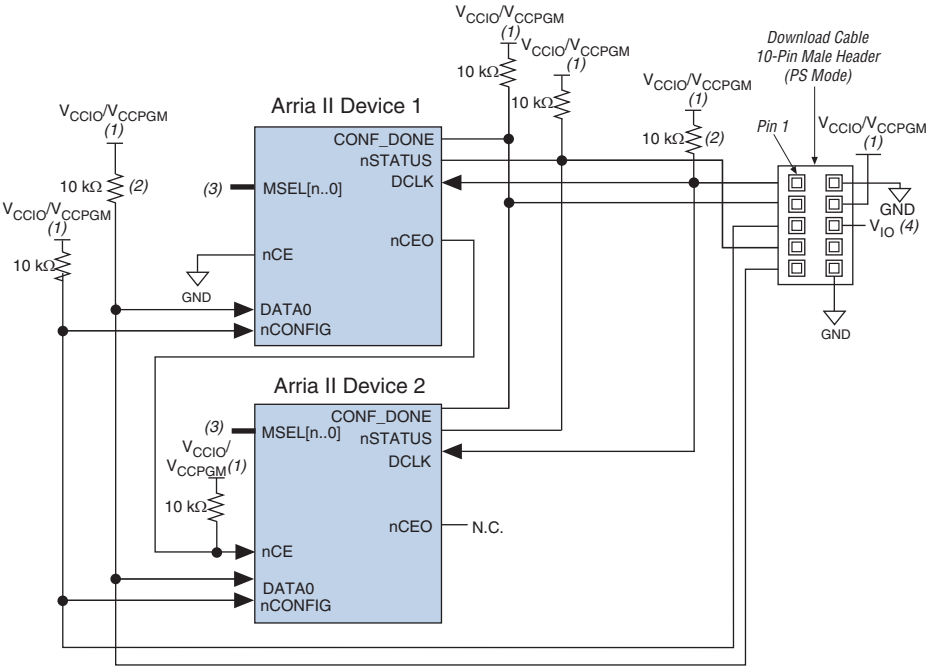
(4) In the USB-Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for AS programming; otherwise, it is a no connect.

You can use a download cable to configure multiple Arria II devices by connecting the nCEO pin of each device to the nCE pin of the subsequent device. The nCE pin of the first device is connected to GND, while its nCEO pin is connected to the nCE of the next device in the chain. The nCE input of the last device 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 devices with a download cable.

Figure 9–15. Multi-Device PS Configuration Using a USB-Blaster, EthernetBlaster, EthernetBlaster II, or ByteBlaster II Cable



Notes to Figure 9–15:

(1) Connect the pull-up resistor to the same supply voltage, V_{CCIO} for Arria II GX devices or V_{CCPGM} for Arria II GZ devices as the USB-Blaster, EthernetBlaster, EthernetBlaster II, or ByteBlaster II 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) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

(4) In the USB-Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for AS programming; otherwise, it is a no connect.

For more information about how to use the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II cables, refer to the following user guides:

- *ByteBlaster II Download Cable User Guide*
- *EthernetBlaster Communications Cable User Guide*
- *EthernetBlaster II Communications Cable User Guide*
- *USB-Blaster Download Cable User Guide*

JTAG Configuration

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**s 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 devices, refer to the following documents:

- *JTAG Boundary-Scan Testing in Arria II Devices* chapter
- *Programming Support for Jam STAPL Language*

Arria II 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 devices during PS configuration, PS configuration is terminated and JTAG configuration begins.

You cannot use the Arria II decompression or design security features if you are configuring your Arria II device using JTAG-based configuration.

A device operating in JTAG mode uses four required pins, TDI, TDO, TMS, and TCK, and one optional pin, TRST. The TCK pin has an internal weak pull-down resistor, while the TDI, TMS, and TRST pins have weak internal pull-up resistors (typically 25 kΩ). All the JTAG pins are powered by the V_{CCIO} power supply of I/O bank 8C for Arria II GX devices and 2.5-V/3.0-V V_{CCPD} power supply for Arria II GZ devices. All the JTAG pins support only the LVTTL I/O standard.

All user I/O pins are tri-stated during JTAG configuration. Table 9–13 lists the function of each JTAG pin.

For more information about how to connect a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary-Scan Testing in Arria II Devices* chapter.

Table 9–13. JTAG Pins Signals (Part 1 of 2)

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 your 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 your board, you can disable the JTAG circuitry by leaving this pin unconnected.

Table 9–13. JTAG Pins Signals (Part 2 of 2)

Pin Name	Pin Type	Description
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 within 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 your 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 your board, you can disable the JTAG circuitry by connecting TCK to GND.
TRST *(1)*	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to the IEEE Std. 1149.1 standard. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting the TRST pin to GND. One kΩ pull-up resistor to V_{CCPD} if you do not use the TRST pin.

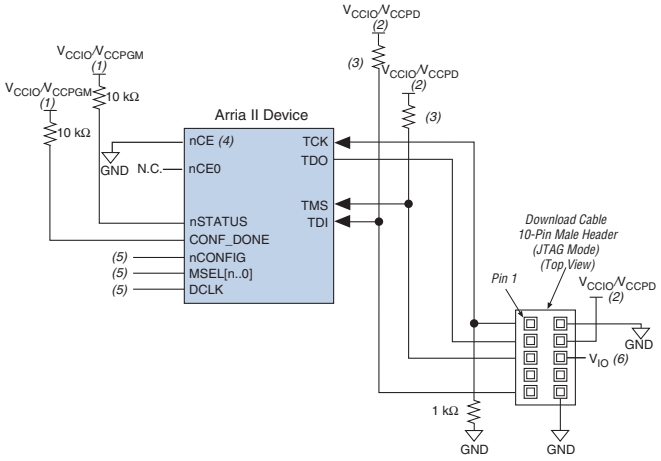
Note to Table 9–13:

(1) The TRST pin is only available for Arria II GZ devices.

During JTAG configuration, you can download data to the device on the PCB through the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II download cable.

Figure 9–16 shows the JTAG configuration of a single Arria II 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} power supply of I/O bank 3C for Arria II GX devices and to V_{CCPGM} (1.8-V, 2.5-V or 3.0-V) power supply for Arria II GZ devices.

(2) Connect the pull-up resistor to the same supply voltage, V_{CCIO} for Arria II GX devices or V_{CCPD} for Arria II GZ devices as the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II cable.

(3) The resistor value can vary from 1 KΩ to 10 KΩ.

(4) You must connect nCE to GND or drive it low for successful JTAG configuration.

(5) Connect the nCONFIG and MSEL pins to support a non-JTAG configuration scheme. If you only use the JTAG configuration, connect nCONFIG to V_{CCIO} for Arria II GX device, V_{CCPGM} for Arria II GZ device, and MSEL to GND. Pull DCLK either high or low, whichever is convenient on your board.

(6) In the USB-Blaster and ByteBlaster II cables, this pin is a no connect.

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 after 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 devices have dedicated JTAG pins that always function as JTAG pins. Not only can you perform JTAG testing on Arria II devices before and after, but also during configuration. While other device families do not support JTAG testing during configuration, Arria II 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 device or waiting for a configuration device to complete configuration. After 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 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 devices, consider the dedicated configuration pins. Table 9–14 lists how these pins are connected during JTAG configuration.

Table 9–14. Dedicated Configuration Pin Connections During JTAG Configuration (Part 1 of 2)

Signal	Description
nCE	On all Arria II devices in the chain, nCE must be driven low by connecting it to GND 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 devices in the chain, you can leave nCEO floating or connected to nCE of the next device.
MSEL	Do not leave these pins 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 V_{CCIO} or V_{CCPGM}, pulling up using a resistor, or driven high by some control circuitry.

Table 9–14. Dedicated Configuration Pin Connections During JTAG Configuration (Part 2 of 2)

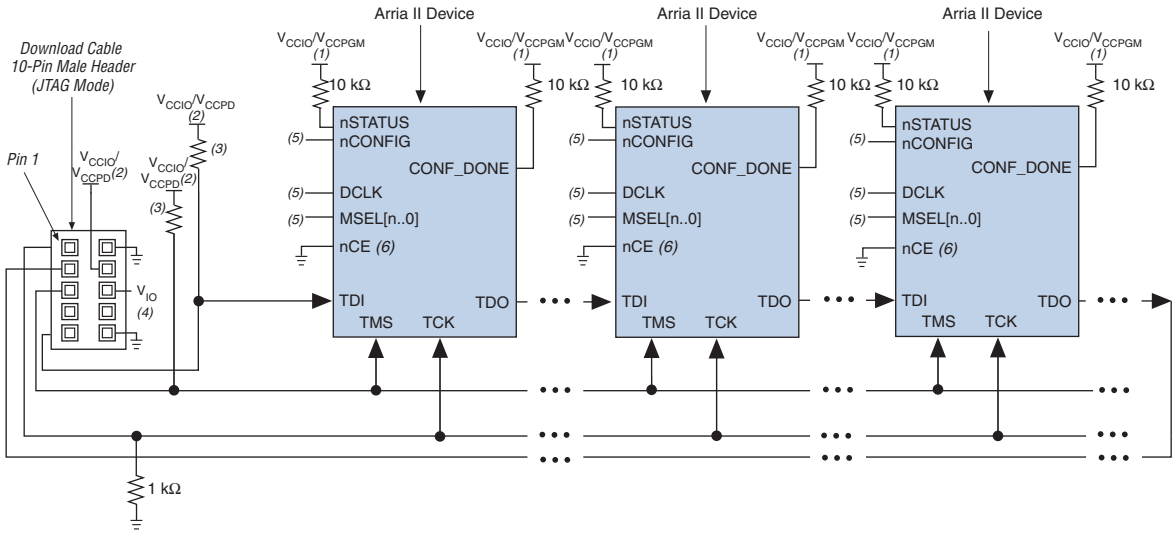
Signal	Description
nSTATUS	Pull to V_{CCIO} or V_{CCPGM} using a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin must be pulled up to V_{CCIO} or V_{CCPGM} individually.
CONF_DONE	Pull to V_{CCIO} or V_{CCPGM} using a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin must be pulled up to V_{CCIO} or V_{CCPGM} individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Do not leave DCLK 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 when using a download cable.

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} power supply of I/O bank 3C for Arria II GX devices and to V_{CCPGM} (1.8-V, 2.5-V or 3.0-V) power supply for Arria II GZ devices.

(2) You must connect the pull-up resistor to the same supply voltage, V_{CCIO} for Arria II GX devices or V_{CCPD} for Arria II GZ devices as the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II cable.

(3) The resistor value can vary from 1 KΩ to 10 KΩ.

(4) In the USB-Blaster and ByteBlaster II cables, pin 6 is a no connect.

(5) You must connect the nCONFIG and MSEL pins to support a non-JTAG configuration scheme. If you only use JTAG configuration, connect nCONFIG to the V_{CCIO} for Arria II GX device, V_{CCPGM} for Arria II GZ device, and MSEL to GND. Pull DCLK either high or low, whichever is convenient on your board.

(6) You must connect nCE to GND or drive it low for successful JTAG configuration.

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 nCE pin of the first device is connected to GND, while its nCEO pin is connected to nCE of the next device in the chain. The nCE input of the last device 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 connecting 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 nCE pin of the second device, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, ensure 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 devices to be cascaded in a JTAG chain.

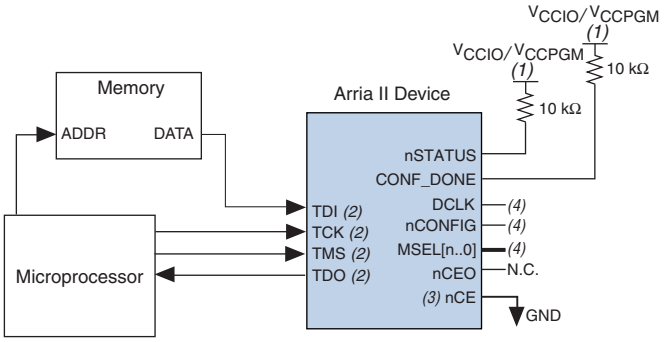
For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera Device Chains* chapter in volume 2 of the *Configuration Handbook*.

You can configure Arria II devices using multiple configuration schemes on the same board. Combining JTAG configuration with a PS or AS 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 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 devices in the chain. The V_{CCIO} power supply for Arria II GX devices or the V_{CCPGM} power supply for Arria II GZ devices must be high enough to meet the V_{IH} specification of the I/O on the device.

(2) To drive the JTAG pins, the microprocessor must use the same I/O standard as V_{CCIO} for Arria II GX devices or V_{CCPD} for Arria II GZ devices.

(3) You must connect `nCE` to GND or drive it low for successful JTAG configuration.

(4) Connect the `nCONFIG` and `MSEL` pins to support a non-JTAG configuration scheme. If you use only the JTAG configuration, connect `nCONFIG` to the V_{CCIO} for Arria II GX device, V_{CCPGM} for Arria II GZ device, and `MSEL` to GND. Pull `DCLK` either high or low, whichever is convenient on your board. Arria II GX devices use `MSEL[3..0]` pins while Arria II GZ devices use `MSEL[2..0]` pins.

Jam STAPL

Jam standard test and programming language (STAPL), JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

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 *AN 425: Using Command-Line Jam STAPL Solution for Device Programming*. To download the Jam Player, visit the Altera website.

Device Configuration Pins

Table 9–15 through Table 9–18 list the connections and functionality of all the configuration-related pins on the Arria II devices.

Table 9–15 lists the Arria II configuration pins and their power supply.

Table 9–15. Configuration Pin Summary for Arria II Devices

Description	Input/Output	Dedicated	Powered By *(1)*	Configuration Mode
TDI	Input	Yes	V_{CCPD}/V_{CCIO}	JTAG
TMS	Input	Yes	V_{CCPD}/V_{CCIO}	JTAG
TCK	Input	Yes	V_{CCPD}/V_{CCIO}	JTAG
TRST	Input	Yes	V_{CCPD}/V_{CCIO}	JTAG
TDO	Output	Yes	V_{CCPD}/V_{CCIO}	JTAG
CRC_ERROR	Output	—	Pull-up	Optional, all modes
DATA0	Input	—	V_{CCPGM}/V_{CCIO} *(2)*	All modes except JTAG
DATA[7..1]	Input	—	V_{CCPGM}/V_{CCIO} *(2)*	FPP
INIT_DONE	Output	—	Pull-up	Optional, all modes
CLKUSR	Input	—	V_{CCPGM}/V_{CCIO} *(2)*	Optional
nSTATUS	Bidirectional	Yes	V_{CCPGM}/Pull-up	All modes
nCE	Input	Yes	V_{CCPGM}/V_{CCIO}	All modes
CONF_DONE	Bidirectional	Yes	V_{CCPGM}/Pull-up	All modes
nCONFIG	Input	Yes	V_{CCPGM}/V_{CCIO}	All modes
ASDO	Output	Yes	V_{CCPGM}/V_{CCIO}	AS
nCSO	Output	Yes	V_{CCPGM}/V_{CCIO}	AS
DCLK	Input	Yes	V_{CCPGM}/V_{CCIO}	PS, FPP
	Output	Yes	V_{CCPGM}/V_{CCIO}	AS
nIO_PULLUP	Input	Yes	V_{CC} *(3)*	All modes
nCEO	Output	—	V_{CCPGM}/Pull-up	All modes
MSEL[2..0] *(4)*	Input	Yes	V_{CCIO} *(6)*	All modes
MSEL[3..0] *(5)*	Input	Yes	V_{CCPD}	All modes

Notes to Table 9–15:

(1) Arria II GX devices use V_{CCIO} while Arria II GZ devices use V_{CCPD}.

(2) For Arria II GZ devices, these pins are powered up by V_{CCPGM} during configuration and V_{CCIO} if they are used as a regular I/O in user mode.

(3) Although the nIO_PULLUP is powered up by V_{CC}, Altera recommends connecting this pin to V_{CCPGM} for Arria II GZ devices, V_{CCIO} for Arria II GX devices, or GND directly without using a pull-up or pull-down resistor.

(4) Arria II GZ devices use a MSEL[2..0] configuration scheme.

(5) Arria II GX devices use a MSEL[3..0] configuration scheme.

(6) Although MSEL[2..0], PORSEL, and nIO_PULLUP are powered by V_{CC}, Altera recommends connecting these pins to V_{CCPGM} or GND directly without using a pull-up or pull-down resistor.

Table 9–16 lists 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 9–16. Dedicated Configuration Pins on the Arria II Device (Part 1 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
VCCPD	N/A	All	Power *(1)*	Dedicated power pin. Use this pin to power the I/O pre-drivers, the HSTL/SSTL input buffers, and the MSEL[3..0] pins. 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, 3.0, 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 device is not successfully configured.
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 (DATA[7..0], CLKUSR, INIT_DONE, DEV_OE, and DEV_CLRn) are on or off before and during configuration. A logic high turns off the weak internal pull-up resistors; 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_{CCPGM} power supply for Arria II GZ devices and the V_{CCIO} power supply for Arria II GX devices, or GND.
MSEL[2..0]	N/A	All	Input	Three-bit configuration input that sets the Arria II GZ device configuration scheme. For more information about the appropriate connections, refer to Table 9–7 on page 9–10. You must hardwire these pins to V_{CCPGM} or GND. The MSEL[2..0] pins have internal 5-kΩ pull-down resistors that are always active.
MSEL[3..0]	N/A	All	Input	Four-bit configuration input that sets the Arria II GX device configuration scheme. For more information about the appropriate connections, refer to Table 9–6 on page 9–9. You must hardwire these pins to 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 9–16. Dedicated Configuration Pins on the Arria II Device (Part 2 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS	N/A	All	Bidirectional open-drain	The device drives nSTATUS low immediately after power-up and releases it after the POR time. 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 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 V_{CCIO} for Arria II GX devices or V_{CCPGM} for Arria II GZ devices are not fully powered up, the following could occur: ■ V_{CCIO}/V_{CCPGM} is powered high enough for the nSTATUS buffer to function properly and nSTATUS is driven low. When V_{CCIO}/V_{CCPGM} is ramped up, POR trips and nSTATUS is released after POR expires. ■ V_{CCIO}/V_{CCPGM} 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 fails 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}/V_{CCPGM} 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	Bidirectional open-drain	Status output. The target device drives the CONF_DONE pin low before and during configuration. After 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 9–16. Dedicated Configuration Pins on the Arria II 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 open-drain	Output that drives low when device configuration is complete. In a single-device configuration, this pin is left floating. In a multi-device configuration, this pin feeds the next device's nCE pin and is pulled high by an external 10-kΩ resistor. The nCEO of the last device in the chain is left floating. The nCEO pin is powered by V_{CCIO} for Arria II GX devices and V_{CCPGM} for Arria II GZ devices. After configuration, nCEO is available as user I/O pins. The state of the nCEO pin depends on the **Dual-Purpose Pin** settings.
ASDO *(2)*	N/A	AS	Output	Control signal from the Arria II 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 *(2)*	N/A	AS	Output	Output control signal from the Arria II 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 *(2)*	N/A	Synchronous configuration schemes (PS, FPP, AS)	Input (PS, FPP) Output (AS)	In PS and FPP configurations, 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 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 by default is driven into an inactive state. 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 9–16. Dedicated Configuration Pins on the Arria II Device (Part 4 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DATA0 *(2)*	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. For Arria II GX devices, DATA0 is a dedicated pin that is used for both PS and AS configuration modes and is not available as a user I/O pin after configuration. For Arria II GZ devices, after PS or FPP configuration, DATA0 is available as a user I/O pin. The state of this pin depends on the **Dual-Purpose Pin** settings.
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. The state of these pin depends on the **Dual-Purpose Pin** settings.

Notes to Table 9–16:

(1) Arria II GZ devices do not support the 3.3-V I/O standard.

(2) To tri-state the AS configuration pins in user mode, turn on the **Enable input tri-state on active configuration pins in user mode** option from the **Device and Pin Options** dialog box in the **Configuration** tab. This tri-states the DCLK, DATA0, nCSO, and ASDO pins.

Table 9–17 lists 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 9–17. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input synchronizes the initialization of one or more devices. 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 a 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. After 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. Enable this pin by turning on the **Enable INIT_DONE output** option in the Quartus II software.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows 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. Enable this pin by turning on the **Enable device-wide reset** (**DEV_CLRn**) option in the Quartus II software.

Table 9–18 lists the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. The TDI, TMS, and TRST pins have weak internal pull-up resistors; the TCK pin 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 device to a JTAG header on your board.

Table 9–18. Dedicated JTAG Pins

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Test data 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} power supply for Arria II GX devices and the V_{CCPD} power supply for Arria II GZ devices. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting this pin to logic high.
TDO	N/A	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. The TDO pin is powered up by the V_{CCPD}/V_{CCIO} power supply. For more information about connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary-Scan Testing in Arria II Devices* chapter. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by leaving this pin unconnected.
TMS	N/A	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. The TMS pin is powered by the V_{CCPD}/V_{CCIO} power supply. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting this pin to logic high.
TCK	N/A	Test clock input	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_{CCPD}/V_{CCIO} power supply. It is expected that the clock input waveform have a nominal 50% duty cycle. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting TCK to GND.
TRST *(1)*	N/A	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to the IEEE Std. 1149.1 standard. The TRST pin is powered by the V_{CCPD} power supply for Arria II GZ devices. Hold TMS at one or keep TCK static while TRST is changed from 0 to 1. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting the TRST pin to GND. You need one kΩ pull-up resistor to V_{CCPD} if you do not use the TRST pin.

Note to Table 9–18:

(1) The TRST pin is only available for Arria II GZ devices.

Configuration Data Decompression

Arria II devices support configuration data decompression, which saves configuration memory space and time. This feature allows you to store compressed configuration data in configuration devices or other memory and transmit this compressed bitstream to Arria II devices. During configuration, the Arria II device decompresses the bitstream in real time and programs its SRAM cells.



Data indicates that compression typically reduces the configuration bitstream size by 35 to 55% based on the designs used.

Arria II devices support decompression in the FPP (when using a MAX II device or microprocessor + flash), AS or fast AS, and PS configuration schemes. The Arria II device decompression feature is not available in the JTAG configuration scheme.



When using FPP mode, the intelligent host must provide a DCLK that is x4 the data rate. Therefore, the configuration data must be valid for four DCLK cycles.

In PS mode, use the Arria II decompression feature because sending compressed configuration data reduces configuration time.

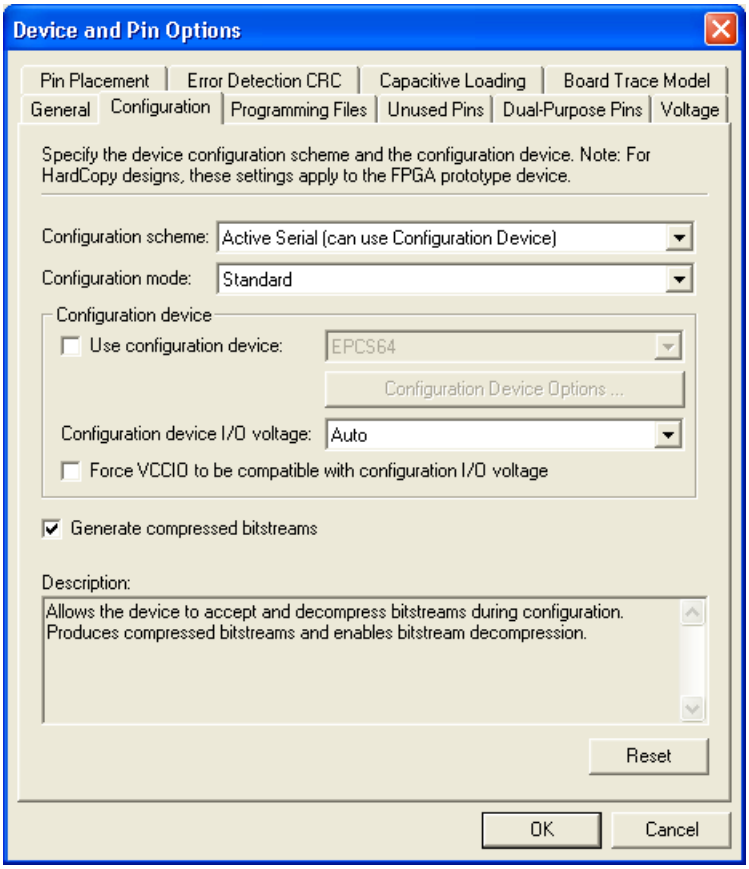
When you enable compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash memory and decreases the time needed to transmit the bitstream to the Arria II device. The time required by an Arria II 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 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, follow these steps:

1. On the Assignments menu, click **Device**. The **Settings** dialog box appears.
2. After selecting your Arria II device, open the **Device and Pin Options** window.
3. In the **Configuration** settings tab, turn on **Generate compressed bitstreams** (as shown in Figure 9–19).

Figure 9–19. Enabling Compression for Arria II Bitstreams in Compiler Settings



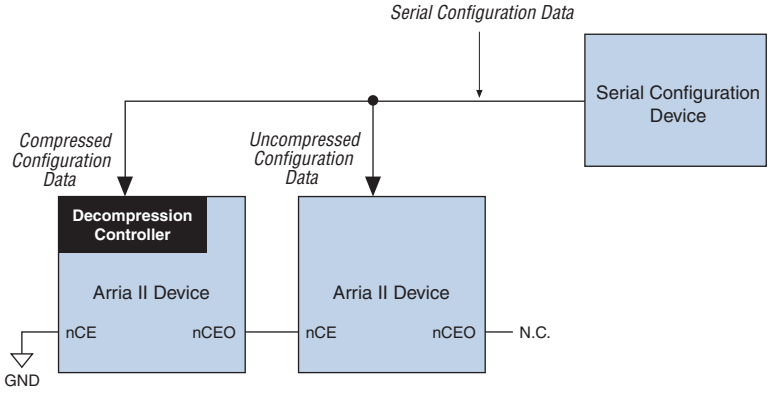
You can also enable compression when creating programming files from the **Convert Programming Files** window. To do this, follow these steps:

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 an Arria II device **.sof**.
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 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 shows a chain of two Arria II devices. The first Arria II device has compression enabled; therefore, receives a compressed bitstream from the configuration device. The second Arria II 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 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



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 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 devices feature dedicated remote system upgrade circuitry. Soft logic (either the Nios® II embedded processor or user logic) implemented in an Arria II 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 AS configuration schemes for Arria II GX devices and in fast AS configuration schemes for Arria II GZ devices. You can also implement remote system upgrades in conjunction with advanced Arria II 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 megabits (Mb) of configuration bitstream.



Arria II devices only support remote system upgrade in the single device fast AS configuration scheme. Because the largest serial configuration device currently supports 128 Mb of configuration bitstream, the remote system upgrade feature is not supported in EP2AGZ300, EP2AGZ350, and larger devices.



The remote system upgrade feature is not supported in a multi-device chain.

Functional Description

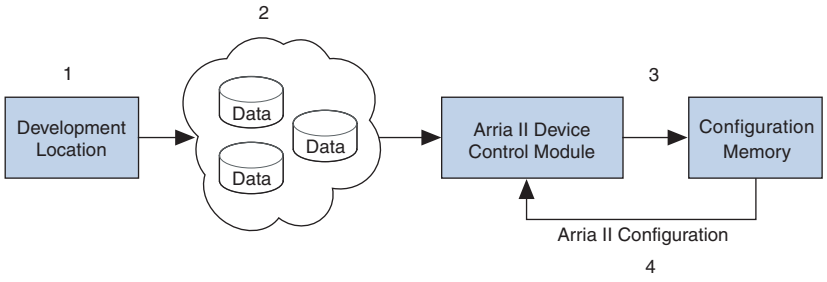
The dedicated remote system upgrade circuitry in Arria II devices manages remote configuration and provides error detection, recovery, and status information. User logic or a Nios II processor implemented in the Arria II device logic array provides access to the remote configuration data source and an interface to the system's configuration memory.

Arria II devices have remote system upgrade processes that involve the following steps:

1. A Nios II processor (or user logic) implemented in the Arria II device logic array receives new configuration data from a remote location. The connection to the remote source uses a communication protocol such as TCP/IP, PCI, user datagram protocol (UDP), 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 error(s) 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.)

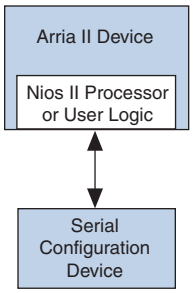
Figure 9–21. Functional Diagram of Arria II Remote System Upgrade



Arria II 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 configuration scheme.

Figure 9–22. Remote System Upgrade Block Diagram for Arria II Configuration Scheme *(Note 1)*



Note to Figure 9–22:

(1) Arria II GX devices use the AS configuration scheme while Arria II GZ devices use the fast AS configuration scheme.

You must set the mode select `MSEL[3..0]` pins to **AS mode** to use the remote system upgrade feature for Arria II GX devices and the `MSEL[2..0]` pins to **fast AS mode** to use the remote system upgrade feature for Arria II GZ devices.

The `MSEL` pin settings vary for different configuration voltage standards and POR delays. To connect `MSEL[3..0]` for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect `MSEL[2..0]` for an Arria II GZ device, refer to Table 9–7 on page 9–10.

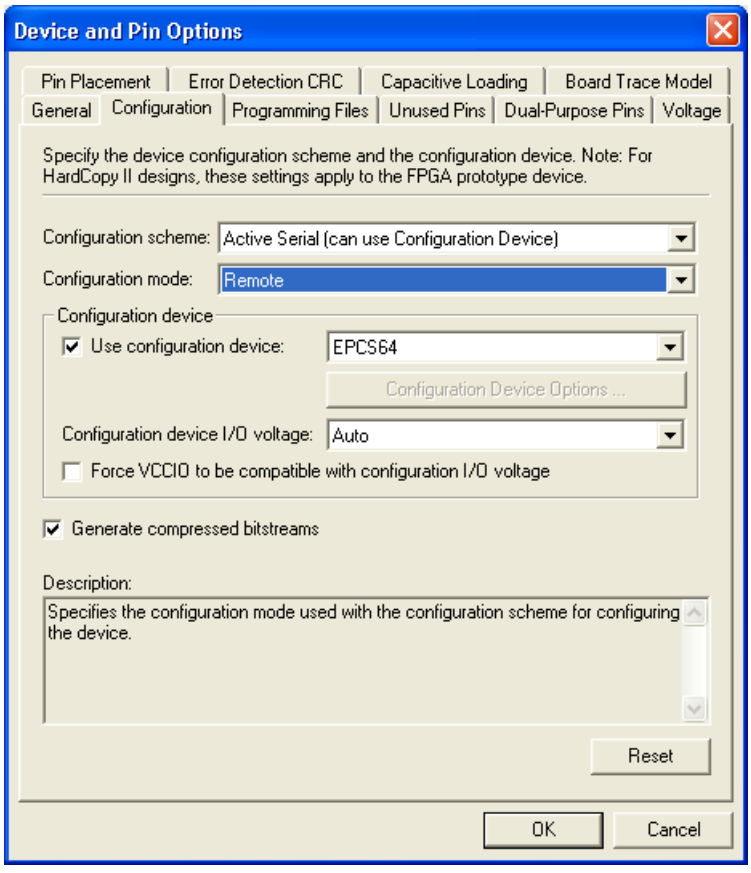
When using AS mode, you must select **remote update mode** in the Quartus II software and insert the ALTREMOTE_UPDATE megafunction to access the circuitry. For more information, refer to "ALTREMOTE_UPDATE Megafunction" on page 9–60.

Enabling Remote Update

You can enable remote update for Arria II 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, follow these steps:

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 in the Compiler Settings Menu for Arria II Devices



Configuration Image Types

When performing a remote system upgrade, Arria II 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 device that performs certain user-defined functions.

Each Arria II 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 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 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 must 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 after power-up and offers several features.

Remote Update Mode

In remote update mode, Arria II devices load the factory configuration image after 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 128 sectors in the EPCS64 device and 32 sectors in the EPCS16 device. Altera recommends not using the same page in the serial configuration devices for two images. Additionally, remote update mode features a user watchdog timer that determines the validity of an application configuration.

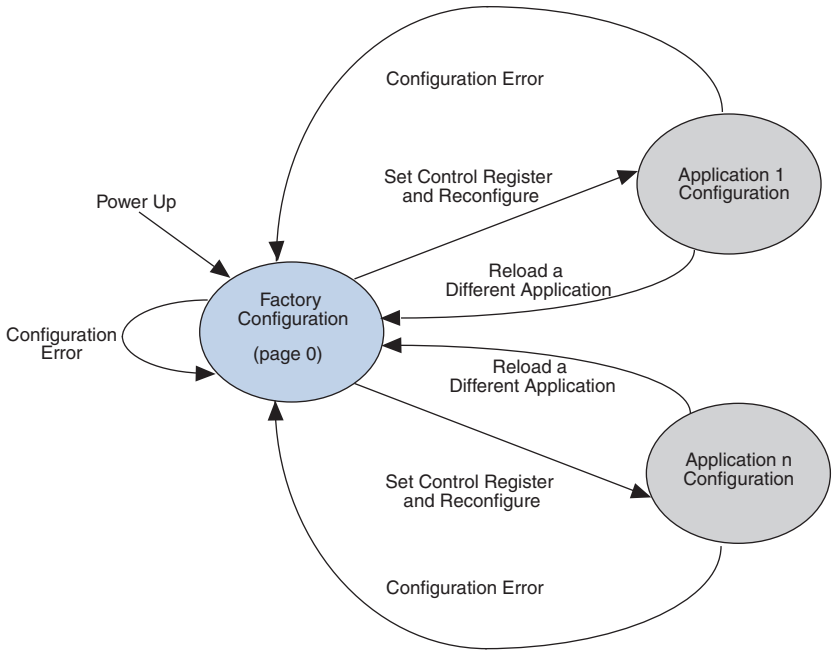
When an Arria II device is first powered up in remote update mode, it loads the factory configuration located at page zero (page registers `PGM[23..0] = 24'b0`). Always store the factory configuration image for your system at page address zero. This corresponds to the start address location 0×000000 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 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 the application configurations in remote update mode.

Figure 9–24. Transitions between Configurations in Remote Update Mode



After power up or a configuration error, the factory configuration logic is loaded automatically. The factory configuration also specifies whether to enable the user watchdog timer for the application configuration and if enabled, to include the timer setting information.

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 must 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 9–59.

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 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 the 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 devices successfully load the application configuration, they enter into user mode. In user mode, the soft logic (a Nios II processor or state machine and the remote communication interface) assists the Arria II 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.

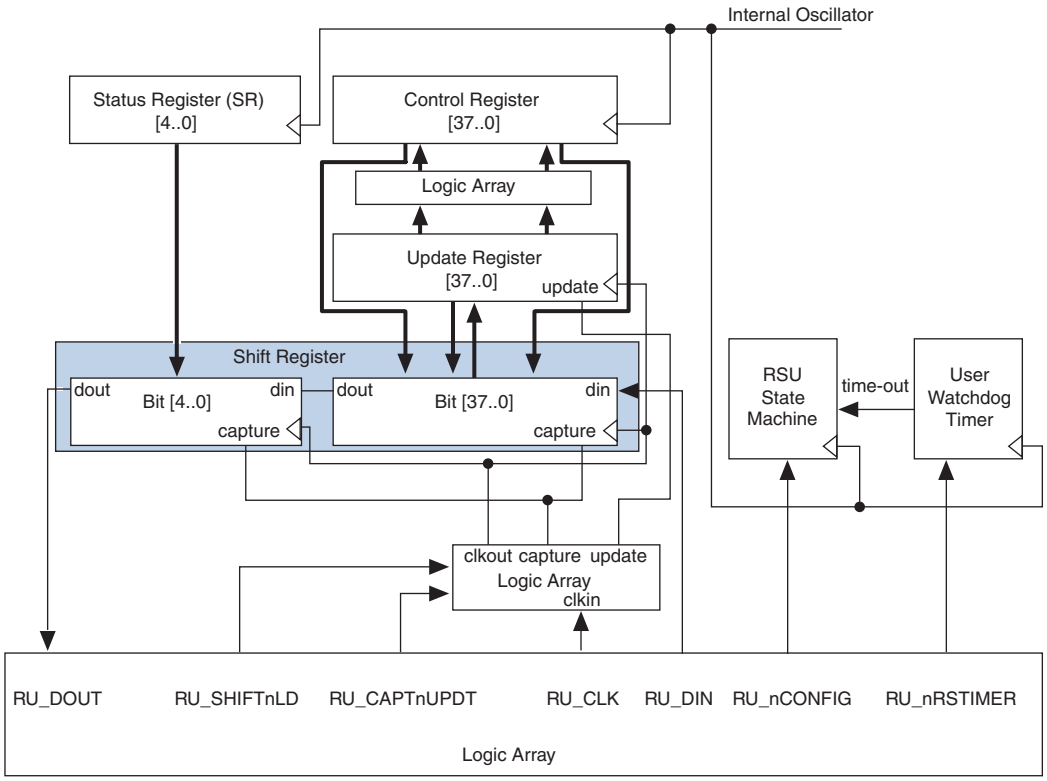
Dedicated Remote System Upgrade Circuitry

This section describes the implementation of the Arria II remote system upgrade dedicated 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 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 datapath of the remote system upgrade block.

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. Table 9–19 lists these registers.

Table 9–19. Remote System Upgrade Registers

Register	Description
Shift	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	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.
Update	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	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.

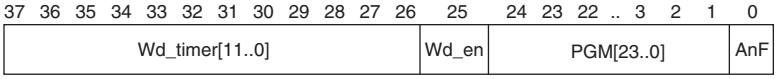
The remote system upgrade control and status registers are clocked by the 10-MHz internal oscillator (the same oscillator that controls the user watchdog timer). However, the remote system upgrade shift and update registers are clocked by the user clock input (`RU_CLK`).

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 listed in Table 9–20. In the figure, the numbers show the bit position of a setting within 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 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 9–20 lists the remote system upgrade control register contents.

Table 9–20. Remote System Upgrade Control Register Contents

Control Register Bit	Remote System Upgrade Mode	Value *(1)*	Definition
AnF *(2)*	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 9–20:

(1) This is the default value of the control register bit.

(2) In remote update mode, the remote configuration block does not update the AnF bit automatically (you can update it manually).

Remote System Upgrade Status Register

The remote system upgrade status register specifies the reconfiguration trigger condition. The various trigger and error conditions include:

- Cyclic redundancy check (CRC) error during application configuration
- nSTATUS assertion by an external device due to an error
- Arria II 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

The contents of the status register are shown in Figure 9–27 and listed in Table 9–21. The numbers in the figure show the bit positions within a 5-bit register.

Figure 9–27. Remote System Upgrade Status Register

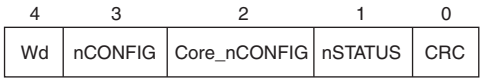


Table 9–21 lists the status register contents for remote system upgrade.

Table 9–21. Remote System Upgrade Status Register Contents

Status Register Bit	Definition	POR Reset Value
CRC (from the 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'
Wd	Watchdog timer caused reconfiguration	1 bit '0'

Note to Table 9–21:

(1) Logic array reconfiguration forces the system to load the application configuration data into the Arria II 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 9–56). 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), the page address, and the 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.



To ensure successful reconfiguration between the pages, assert the RU_nCONFIG signal for a minimum of 250 ns. This is equivalent to strobing the reconfig input of the ALTREMOTE_UPDATE megafunction high for a minimum of 250 ns.

In the event of 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 9–22 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 9–22. Control Register Contents after an Error or Reconfiguration Trigger Condition (Part 1 of 2)

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

Table 9–22. Control Register Contents after an Error or Reconfiguration Trigger Condition (Part 2 of 2)

Reconfiguration Error/Trigger	Control Register Setting Remote Update
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 device.



To allow the remote system upgrade dedicated circuitry to reset the watchdog timer, you must assert the RU_nRSTIMER signal active for a minimum of 250 ns. This is equivalent to strobing the reset_timer input of the ALTREMOTE_UPDATE megafunction high for a minimum of 250 ns.

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^{17} cycles. The cycle time is based on the frequency of the 10-MHz internal oscillator. Table 9–23 lists the operating range of the 10-MHz internal oscillator.

Table 9–23. 10-MHz Internal Oscillator Specifications

Minimum	Typical	Maximum	Units
4.3	5.3	10	MHz

The user watchdog timer begins counting after 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.

During the configuration cycle of the device, the user watchdog timer is not enabled. 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 device logic array and the dedicated circuitry, generates configuration files for production, and allows remote programming of the system configuration memory.

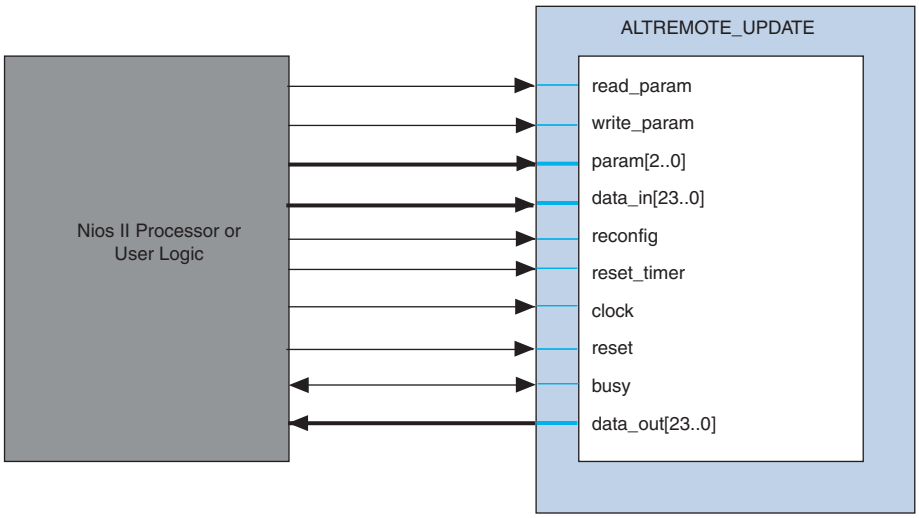
Use the ALTREMOTE_UPDATE megafunction option in the Quartus II software as the interface between the remote system upgrade circuitry and the device logic array interface. Using the megafunction block instead of creating your own logic saves design time and offers more efficient logic synthesis and device 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 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 *Remote Update Circuitry (ALTREMOTE_UPDATE) Megafunction User Guide*.

Design Security

This section provides an overview of the design security features and their implementation on Arria II devices using AES. It also covers the new security modes available in Arria II devices.

As Arria II devices continue 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 devices address these concerns with both volatile and non-volatile security feature support. Arria II devices have the ability to decrypt configuration bitstreams using the AES algorithm, an industry-standard encryption algorithm that is FIPS-197 certified. Arria II devices have a design security feature which uses a 256-bit security key.

Arria II 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 design security feature, the security key is stored in the Arria II device. Depending on the security mode, you can configure the Arria II 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 devices using FPP configuration mode with an external host (such as a MAX II device or microprocessor), or when using AS, fast AS, or PS configuration schemes. The design security feature is also available in remote update mode with AS and fast AS configuration mode.



The design security feature is not available when you are configuring your Arria II device using JTAG-based configuration. For more information, refer to "Supported Configuration Schemes" on page 9–66.



When using a serial configuration scheme such as AS, fast AS, or PS, configuration time is the same whether or not you enable the design security feature. If you use the FPP scheme with the design security or decompression feature, a x4 DCLK is required. This results in a slower configuration time when compared with the configuration time of an Arria II device that has neither the design security nor the decompression feature enabled.

Arria II Security Protection

Arria II 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 device and cannot be read out through any interface. In addition, as configuration file read-back is not supported in Arria II 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 configuration file formats are proprietary and the file contains millions of bits which require specific decryption. Reverse engineering the Arria II device is just as difficult because the device is manufactured on the most advanced 40-nm process technology.

Security Against Tampering

After the Tamper Protection bit is set in the key programming file generated by the Quartus II software, the Arria II 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 devices support two types of security key programming—volatile and non-volatile keys. Table 9–24 lists the differences between volatile keys and non-volatile keys.

Table 9–24. 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. *(2)*	Secure against copying and reverse engineering. Tamper resistant if tamper protection bit is set.

Notes to Table 9–24:

(1) Key programming is carried out using the JTAG interface.

(2) Arria II GZ devices do not support this feature.

You can program the non-volatile key to the Arria II device without an external battery. Also, there are no additional requirements of any of the Arria II 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.

For Arria II GX devices, after power up, wait 100 ms (standard POR delay) or 4 ms (fast POR delay) before beginning key programming to ensure that V_{CCBAT} is at full rail. For Arria II GZ devices, after power up, wait 300 ms (PORSEL = 0) or 12 ms (PORSEL = 1) before beginning key programming to ensure that V_{CCBAT} is at full rail.

For more information about how to calculate the key retention time of the battery used for volatile key storage, refer to the *Arria II GX PowerPlay Early Power Estimator*.

For more information about battery specifications, refer to the *Device Datasheet for Arria II Devices* chapter.

For more information about the VCCBAT pin connection recommendations, refer to *Arria II Device Family Pin Connection Guidelines*.

Arria II Design Security Solution

Arria II devices are SRAM-based devices. To provide design security, Arria II devices require a 256-bit security key for configuration bitstream encryption.

To carry out secure configuration, follow these steps (refer to Figure 9–29):

1. Program the security key into the Arria II device.

   Program the user-defined 256-bit AES keys to the Arria II device through the JTAG interface.

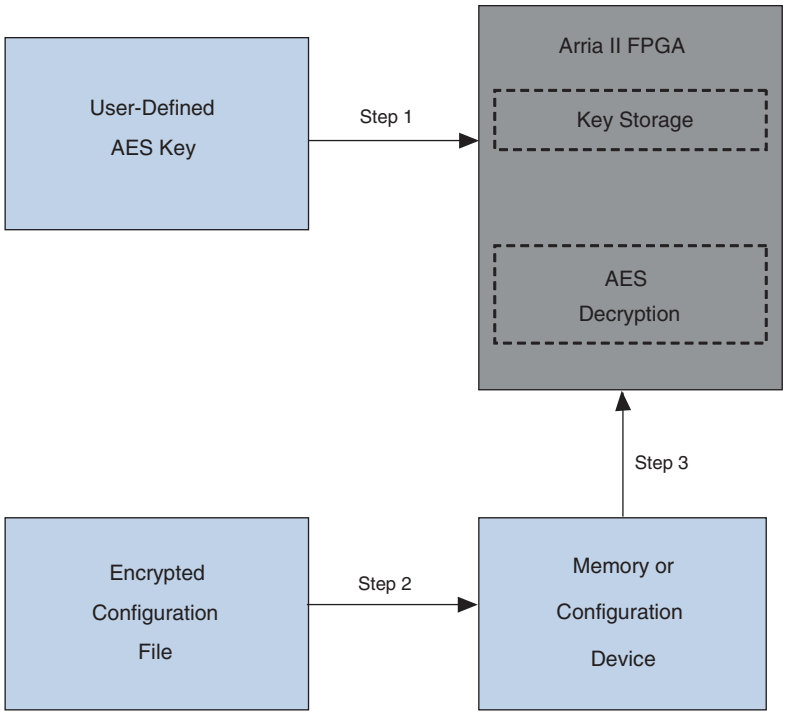
2. Encrypt the configuration file and store it in the external memory.

   Encrypt the configuration file with the same 256-bit keys used to program the Arria II 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 device.

At system power-up, the external memory device sends the encrypted configuration file to the Arria II 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 described in "Design Security" on page 9–61.

Security Modes Available

The following security modes are available on the Arria II device:

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

☞ Arria II GZ devices do not support this feature.

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 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 devices. This process is irreversible and prevents Altera from carry-out failure analysis. Contact Altera Technical Support to enable the tamper protection bit.

No Key Operation

Only unencrypted configuration bitstreams are allowed to configure the device.

Table 9–25 lists the different security modes and configuration bitstream supported for each mode.

Table 9–25. 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 *(2)*	Secure (tamper resistant) *(3)*	Encrypted
Non-volatile key with tamper protection bit set	Secure (tamper resistant) *(3)*	Encrypted

Notes to Table 9–25:

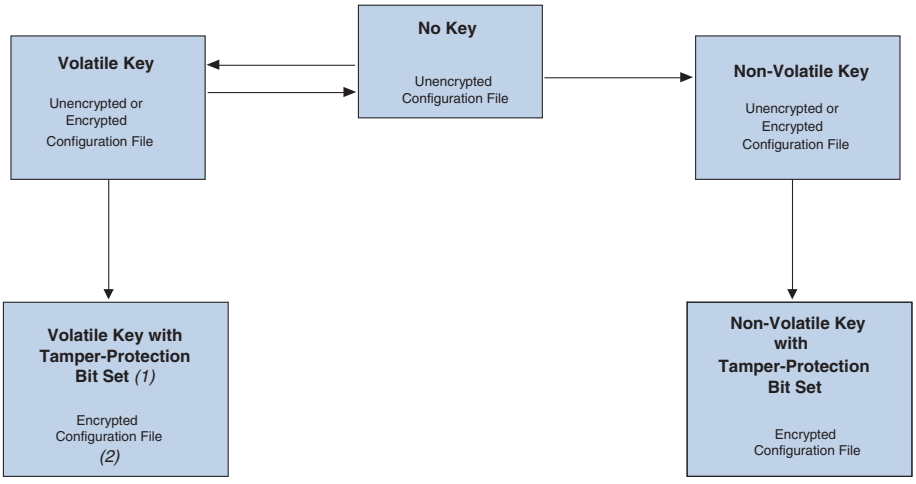
(1) In No key operation, only the unencrypted configuration file is supported.
(2) Arria II GZ devices do not support this feature.
(3) The tamper protection bit setting does not prevent the device from being reconfigured.

Supported Configuration Schemes

The Arria II device supports only selected configuration schemes, depending on the security mode you select when you encrypt the Arria II device.

Figure 9–30 shows the restrictions of each security mode when encrypting Arria II devices.

Figure 9–30. Arria II Security Modes—Sequence and Restrictions



Notes to Figure 9–30:

(1) Arria II GZ devices do not support this feature.
(2) Arria II devices do not accept encrypted configuration files if the volatile key is erased. If the volatile key is erased, you must use the volatile key without the tamper-protection bit set to reprogram the key.

Table 9–26 lists the configuration modes allowed in each of the security modes.

Table 9–26. 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	■ PS with AES (and/or with decompression) ■ FPP with AES (and/or with decompression) ■ Remote update AS or fast AS with AES (and/or with decompression) ■ AS or fast 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	■ PS with AES (and/or with decompression) ■ FPP with AES (and/or with decompression) ■ Remote update AS or fast AS with AES (and/or with decompression) ■ AS or fast 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	■ PS with AES (and/or with decompression) ■ FPP with AES (and/or with decompression) ■ Remote update AS or fast AS with AES (and/or with decompression) ■ AS or fast AS (and/or with decompression)

Note to Table 9–26:

(1) There is no impact to the configuration time required when compared with unencrypted configuration modes except when using FPP with AES (and/or decompression), which requires DCLK that is x4 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 AS, fast AS, and PS configuration schemes.

Table 9–27 lists the configuration schemes that support the design security feature both for volatile key and non-volatile key programming.

Table 9–27. Design Security Configuration Schemes Availability

Configuration Scheme	Configuration Method	Design Security
FPP	MAX II device or microprocessor and flash memory	✓ *(1)*
AS	Serial configuration device	✓
Fast AS	Serial configuration device	✓
PS	MAX II device or microprocessor and flash memory	✓
	Download cable	✓
JTAG *(2)*	MAX II device or microprocessor and flash memory	—
	Download cable	—

Notes to Table 9–27:

(1) In this mode, the host system must send a DCLK that is x4 the data rate.

(2) JTAG configuration supports only unencrypted configuration file.

You can use the design security feature with other configuration features, such as the 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 device first decrypts and then decompresses the configuration file.

Document Revision History

Table 9–28 lists the revision history for this chapter.

Table 9–28. Document Revision History

Date	Version	Changes Made
July 2012	4.3	■ Updated "FPP Configuration Using a MAX II Device as an External Host" section. ■ Added pull-up resistor to nCONFIG in Figure 9–1, Figure 9–2, Figure 9–3, Figure 9–10, Figure 9–11, and Figure 9–12.
December 2011	4.2	■ Updated Table 9–8, Table 9–9, Table 9–10, and Table 9–12. ■ Updated Figure 9–16 and Figure 9–17. ■ Updated "Configuration" and "FPP Configuration Using a MAX II Device as an External Host" sections. ■ Minor text edits.
June 2011	4.1	■ Updated Table 9–9, Table 9–10, Table 9–12, Table 9–18, and Table 9–23. ■ Updated the "Programming Serial Configuration Devices" and "Configuration Data Decompression" sections. ■ Removed references to the "ByteBlaster MV" and "MasterBlaster" cables as they are discontinued. ■ Minor text edits.
December 2010	4.0	■ Updated for the Quartus II software version 10.1 release. ■ Added Arria II GZ devices information. ■ Minor text edits.
July 2010	3.0	Updated for Arria II GX v10.0 release: ■ Updated Table 9–9 and Table 9–17. ■ Updated Figure 9–4, Figure 9–5, Figure 9–13, Figure 9–16, Figure 9–17, Figure 9–21, and Figure 9–30. ■ Updated "AS and Fast AS Configuration (Serial Configuration Devices)" and "Flexible Security Key Storage" sections. ■ Added "Guidelines for Connecting Serial Configuration Device to Arria II Devices on an Active Serial Interface" section. ■ Minor text edits.
November 2009	2.0	Updated for Arria II GX v9.1 release: ■ Updated Table 9–3, Table 9–10, Table 9–11, Table 9–13. ■ Updated Figure 9–2, Figure 9–3, and Figure 9–6. ■ Updated "VCCPD Pins", "JTAG Configuration", "Remote System Upgrade Mode", "Remote System Upgrade State Machine", "User Watchdog Timer" sections. ■ Minor text edits.
June 2009	1.1	■ Updated Table 9–2, Table 9–3, Table 9–9, Table 9–10, Table 9–19, and Table 9–21. ■ Updated Figure 9–6, Figure 9–11, and Figure 9–16. ■ Updated "VCCIO Pins for I/O Banks 3C and 8C", "FPP Configuration Using an External Host", and "Programming Serial Configuration Devices" sections. ■ Removed "Volatite or Non-Volatile Key with JTAG Anti-Tamper Protection Bit Set" section.
February 2009	1.0	Initial release.

10. SEU Mitigation in Arria II Devices

AIIGX51010-4.3

This chapter describes how to activate and use the error detection cyclic redundancy check (CRC) feature when your Arria® II device is in user mode and how to recover from configuration errors caused by CRC errors.

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 device is correct.
- Alert the system to the occurrence of a configuration error.

The error detection CRC feature is provided in the Quartus® II software starting with version 9.1 for Arria II GX devices and version 10.1 for Arria II GZ devices.

Using the error detection CRC feature on Arria II devices has no impact on fitting or performance.

For more information about the CRC feature, refer to *AN 539: Test Methodology of Error Detection and Recovery using CRC in Altera FPGA Devices*.

This chapter contains the following sections:

- "Error Detection Fundamentals"
- "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 10–7
- "Software Support" on page 10–9
- "Recovering From CRC Errors" on page 10–10

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 function to calculate 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 devices are successfully configured and in user mode, the error detection CRC feature ensures the integrity of the configuration data.

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

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 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 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 RAM. The configuration RAM chain used for storing CRC check bits is 16 bits wide and its length is equal to the number of frames in the device.

User Mode Error Detection

Arria II devices have built-in error detection circuitry to detect data corruption by soft errors in the configuration RAM cells. This feature allows all configuration RAM contents to be read and verified to match a configuration-computed CRC value. Soft errors are changes in a configuration RAM's bit state due to an ionizing particle.

The error detection capability continuously calculates the CRC of the configured configuration RAM bits and compares it with the pre-calculated CRC. If the CRCs match, there is no error in the current configuration RAM bits. The process of error detection continues until the device is reset by setting `nCONFIG` low.

To enable the error detection process when the device transitions into user mode, turn on the **Enable Error Detection CRC** option on the **Error Detection CRC** page of the **Device and Pin Options** dialog box in the Quartus II software.

A single 16-bit error detection CRC calculation is done on a per-frame basis. After the error detection circuitry has finished the CRC calculation for a frame, the resulting 16-bit signature is `hex 0000`. If the error detection circuitry detects no configuration RAM bit errors in a frame, the output signal `CRC_ERROR` is 0. If the circuitry detects a configuration RAM bit error in a frame in the device, the resulting signature is non-zero and the error detection circuitry starts searching for the error bit location.

The error detection circuitry in Arria II 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 triple-bit errors. The probability of more than three configuration RAM bits being flipped by a single event upset (SEU) is very low. In general, the probability of detection for all error patterns is 99.998%.

The error detection circuitry 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 the reporting of bit location is not guaranteed.

You can also read the error bit location through the JTAG and the core interface. Before the error detection circuitry detects the next error in another frame, you must shift erroneous bits out from the error message register (EMR) with either the JTAG instruction, `SHIFT_EDERROR_REG`, or the core interface. The CRC circuitry continues to run, and if an error is detected, you must decide whether to complete the reconfiguration or to ignore the CRC error.

For more information about the timing requirement to shift out error information from the EMR, refer to "Error Detection Timing" on page 10–7.

The error detection circuitry 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 must monitor the CRC_ERROR signal and take the appropriate actions if a CRC error occurs.

The error detection circuitry in Arria II 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 configuration RAM. The total storage size is 16 (number of bits per frame) × the number of frames.

The CRC_ERROR signal is asserted if the error detection circuitry verification does not match with the configuration-computed CRC value. However, the Arria II device error detection CRC feature does not check the memory blocks and I/O buffers. Therefore, the CRC_ERROR signal may stay solid high or low, depending on the error status of the previously checked configuration RAM 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 when compared with configuration RAM cells. MLAB and M9K memory blocks support parity bits that are used to check the contents of the memory blocks for any error in Arria II GX devices. In addition to MLAB and M9K memory blocks, M144K memory blocks are used to check the contents of the memory blocks for any error in Arria II GZ devices.

For more information about error detection in Arria II memory blocks, refer to the *Memory Blocks in Arria II Devices* chapter.

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 devices, thereby enabling the 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 lists the EDERROR_INJECT JTAG instruction for Arria II devices.

Table 10–1. EDERROR_INJECT JTAG Instruction for Arria II Devices

JTAG Instruction	Instruction Code	Description
EDERROR_INJECT	00 0001 0101	This instruction controls the 21-bit JTAG fault injection register 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.

For more information about **.jam**, refer to *AN 539: Test Methodology of Error Detection and Recovery using CRC in Altera FPGA Devices*.

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 the EDERROR_INJECT JTAG instruction to flip the readback bits. The Arria II device is then forced into error test mode. Altera recommends reconfiguring the device after the test completes.



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 10–6.

Table 10–2 lists how the fault injection register is implemented and describes error injection for Arria II devices.

Table 10–2. Fault Injection Register for Arria II Devices

Description	Bit[20..19]			Bit[18..8]	Bit[7..0]
	Error Type *(1)*		Error Injection Type	Byte Location of the Injected Error	Error Byte Value
	Bit[20]	Bit[19]			
Content	0	1	Single error injection	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.
	1	0	Double-adjacent 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 devices offer on-chip circuitry for automated SEU detection. Some applications 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 devices, eliminating the need for external logic. The CRC_ERROR pin reports a CRC error when configuration RAM data is corrupted; you must decide whether to reconfigure the device or to ignore the error.

Error Detection Pin Description

Table 10–3 lists the CRC_ERROR pin description for Arria II devices.

Table 10–3. CRC_ERROR Pin Description for Arria II Devices

Pin Name	Pin Type	Description
CRC_ERROR	I/O or output open-drain	Active high signal indicating that the error detection circuit has detected errors in the configuration RAM bits. This is an optional pin and is used when you enable the error detection CRC circuit. When you disable the error detection CRC circuit, it is a user I/O pin. When using the WYSIWYG function, the CRC error output is a dedicated path to the CRC_ERROR pin. To use the CRC_ERROR pin, you can tie this pin to V_{CCIO} 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.

Error Detection Block

The error detection block contains the logic necessary to calculate the 16-bit error detection CRC signature for the configuration RAM bits in the Arria II device.

The CRC circuit continues running even if an error occurs. When a CRC error occurs, the device sets the CRC_ERROR pin high. Table 10–4 lists the two types of CRC detection that check the configuration bits for Arria II devices.

Table 10–4. Two Types of CRC Detection for Arria II Devices

User Mode CRC Detection	Configuration CRC Detection
■ This is the configuration RAM error checking ability (16-bit error detection CRC) during user mode for use by the CRC_ERROR pin. ■ For each frame of data, the pre-calculated 16-bit error detection 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 error detection CRC result for the first frame and also shifts the 16-bit error detection CRC bits to the 16-bit error detection CRC storage registers for test purposes. ■ You can deliberately introduce single error, double errors, or double-adjacent errors to the configuration memory for testing and design verification.	■ This is the 16-bit configuration CRC that is embedded in every configuration data frame. ■ During configuration, after a frame of data is loaded into the Arria II device, the pre-computed configuration CRC is shifted into the CRC circuitry. ■ At the same time, the configuration CRC value for the data frame shifted-in is calculated. If the pre-computed configuration CRC and calculated configuration CRC values do not match, nSTATUS is set low. Every data frame has a 16-bit configuration CRC; therefore, there are many 16-bit configuration CRC values for the whole configuration bitstream as there are many data frames. 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 for Arria II devices.

Figure 10–1. Error Detection Circuitry, Syndrome Registers, and Error Injection Block for Arria II Devices

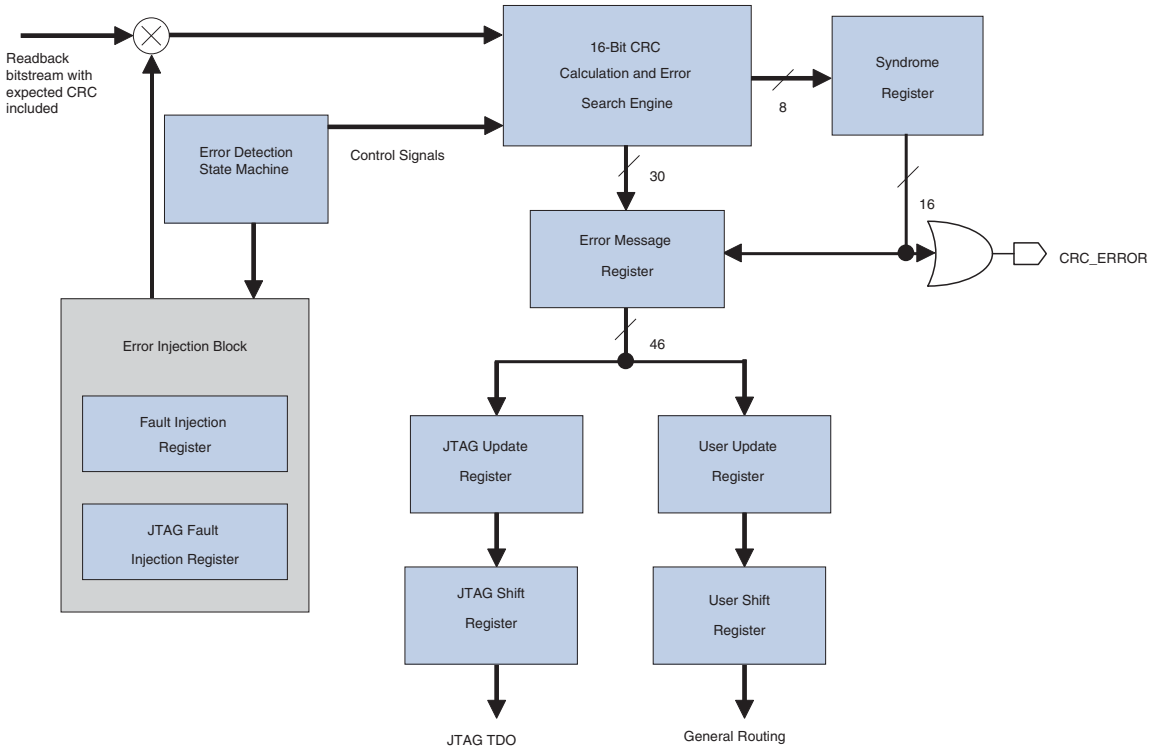


Table 10–5 lists the registers shown in Figure 10–1.

Table 10–5. Error Detection Registers for Arria II Devices

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 EMR. The content of the register is shifted out through the SHIFT_EDERROR_REG JTAG instruction or to the core through the core interface.
JTAG Update Register	This register is automatically updated with the contents of the EMR 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 EMR at 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 EMR 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 EMR at 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 SHIFT_EDERROR_REG JTAG instruction.
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 EDERROR_INJECT JTAG instruction. 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 updated.

Error Detection Timing

When you enable the error detection CRC feature through the Quartus II software, the device automatically activates the CRC error detection process after 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 EMR 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 EMR 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 lists the minimum and maximum error detection frequencies for Arria II devices.

Table 10–6. Minimum and Maximum Error Detection Frequencies for Arria II Devices

Device Type	Error Detection Frequency	Maximum Error Detection Frequency	Minimum Error Detection Frequency	Valid Divisors (n)
Arria II	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 10–9). The divisor is a power of two (2), where *n* is between 1 and 8. The divisor ranges from 2 through 256 (refer to Equation 10–1).

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 devices is done on a per-frame basis.

The EMR is updated whenever an error occurs. If the error location and message are not shifted out before the next error location is found, the previous error location and message are 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 EMR 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 lists the estimated minimum interval time between each update for the EMR in Arria II devices.

Table 10–7. Minimum Update Interval for Error Message Register in Arria II Devices

Device	Timing Interval (µs)
EP2AGX45	11.04
EP2AGX65	11.04
EP2AGX95	14.88
EP2AGX125	14.88
EP2AGX190	19.64
EP2AGX260	19.64
EP2AGZ225	19.8
EP2AGZ300	21.8
EP2AGZ350	21.8

The 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 lists the minimum and maximum estimated clock frequency time for each CRC calculation for Arria II devices. The minimum CRC calculation time is calculated using the maximum error detection frequency with a divisor factor 1. The maximum CRC calculation time is calculated using the minimum error detection frequency with a divisor factor 8.

Table 10–8. CRC Calculation Time for Arria II Devices

Device	Minimum Time (ms)	Maximum Time (s)
EP2AGX45	73.80	20.40
EP2AGX65	73.80	20.40
EP2AGX95	125.80	34.80
EP2AGX125	125.80	34.80
EP2AGX190	216.00	59.90
EP2AGX260	216.00	59.90
EP2AGZ225	225	62.44
EP2AGZ300	296	82.05
EP2AGZ350	296	82.05

Software Support

The Quartus II software, starting with version 9.1 supports the error detection CRC feature for Arria II GX devices and starting with version 10.1 supports the error detection CRC feature for Arria II GZ 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.

To enable the error detection feature using the CRC, follow these steps:

1. Open the Quartus II software and load a project using an Arria II device.
2. On the Assignments menu, click **Device**. The **Device** dialog box appears.
3. Click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
4. In the **Category** list, select **Error Detection CRC** tab.
5. Turn on **Enable Error Detection CRC**.
6. In the **Divide error check frequency by** pull-down list, enter a valid divisor as listed in Table 10–6 on page 10–7.
7. Click **OK**.

Recovering From CRC Errors

The system that the Arria II 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 for the system to reconfigure the device.

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 require a design to account for these errors.

Document Revision History

Table 10–9 lists the revision history for this chapter.

Table 10–9. Document Revision History

Date	Version	Changes
February 2014	4.3	Updated the minimum CRC calculation time for EP2AGX45, EP2AGX65, EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260 in Table 10–8.
July 2012	4.2	Removed repeated paragraph in the “User Mode Error Detection” section.
June 2011	4.1	■ Updated “User Mode Error Detection” section. ■ Minor text edits.
December 2010	4.0	■ Updated for the Quartus II software version 10.1 release. ■ Added Arria II GZ devices information. ■ Minor text edits.
July 2010	3.0	Updated for Arria II GX v10.0 release: ■ Updated Table 10–3, Table 10–6, Table 10–7, and Table 10–8.
November 2009	2.0	Updated for Arria II GX v9.1 release: ■ Updated Table 10–7 and Table 10–8. ■ Minor text edits.
February 2009	1.0	Initial release.

11. JTAG Boundary-Scan Testing in Arria II Devices

AIIGX51011-4.1

This chapter describes the boundary-scan test (BST) features that are supported in Arria® II devices and how to use the IEEE Std. 1149.1 and Std. 1149.6 BST circuitries in Arria II devices. The features are similar to Arria GX devices, unless stated in this chapter.

This chapter includes the following sections:

- "BST Architecture for Arria II Devices" on page 11–1
- "BST Operation Control" on page 11–3
- "I/O Voltage Support in a JTAG Chain" on page 11–5
- "Disabling IEEE Std. 1149.1 BST Circuitry" on page 11–6
- "Boundary-Scan Description Language Support" on page 11–7

Arria II GX devices support IEEE Std. 1149.1 and IEEE Std. 1149.6, while Arria II GZ devices support IEEE Std. 1149.1 only. The IEEE Std. 1149.6 is only supported on the high-speed serial interface (HSSI) transceivers in Arria II GX devices. The IEEE Std. 1149.6 enables board-level connectivity checking between transmitters and receivers that are AC coupled (connected with a capacitor in series between the source and destination).

BST Architecture for Arria II Devices

For Arria II GX devices, the TDO output pin and all JTAG input pins are powered by the V_{CCIO} power supply of I/O Bank 8C, while for Arria II GZ devices, the TDO output pin and all the JTAG input pins are powered by 2.5-V/3.0-V V_{CCPD} supply of I/O Bank 1A. All user I/O pins are tri-stated during JTAG configuration.

For more information about the IEEE Std. 1149.1 BST architecture, BST circuitry, and boundary-scan register for Arria II devices, 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. 1149.6 Boundary-Scan Register for Arria II GX Devices

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.

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Figure 11–1 shows the Arria II GX HSSI transmitter boundary-scan cell.

Figure 11–1. HSSI Transmitter BSC with IEEE Std. 1149.6 BST Circuitry for Arria II GX Devices

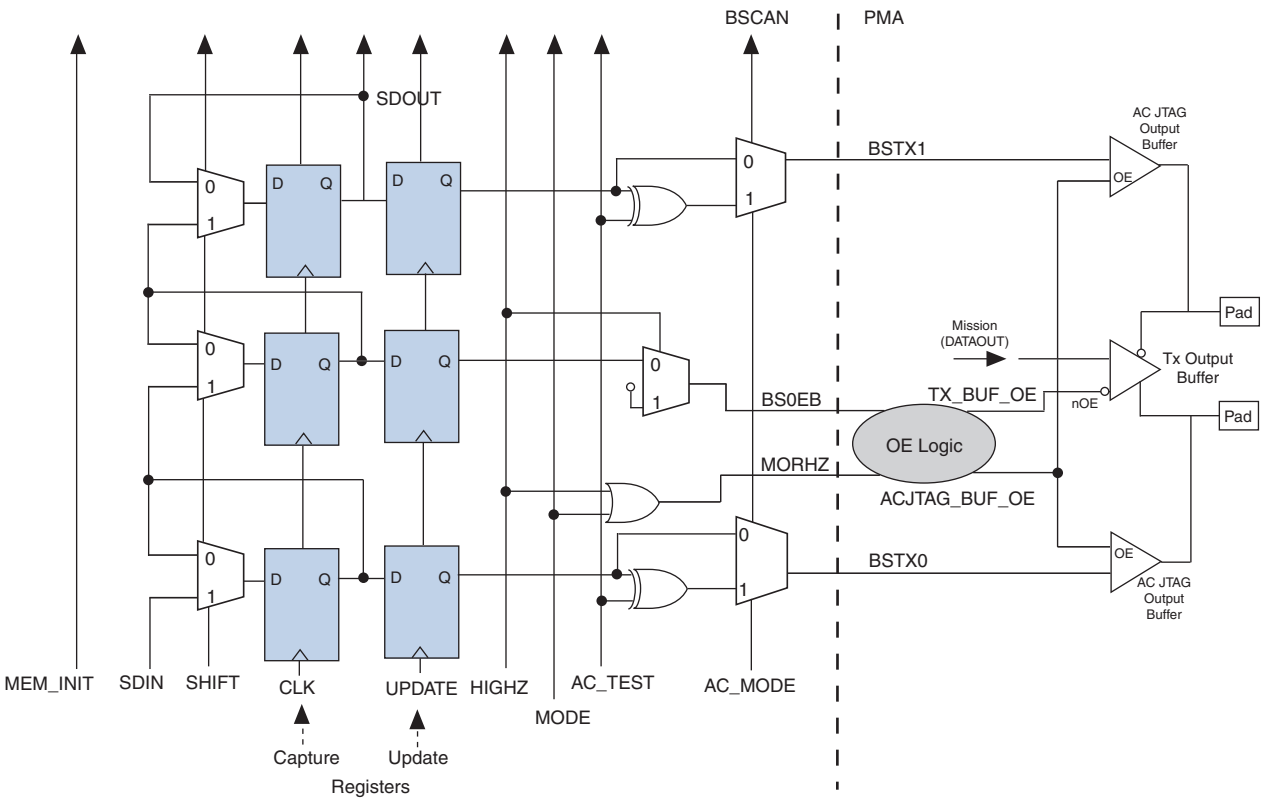
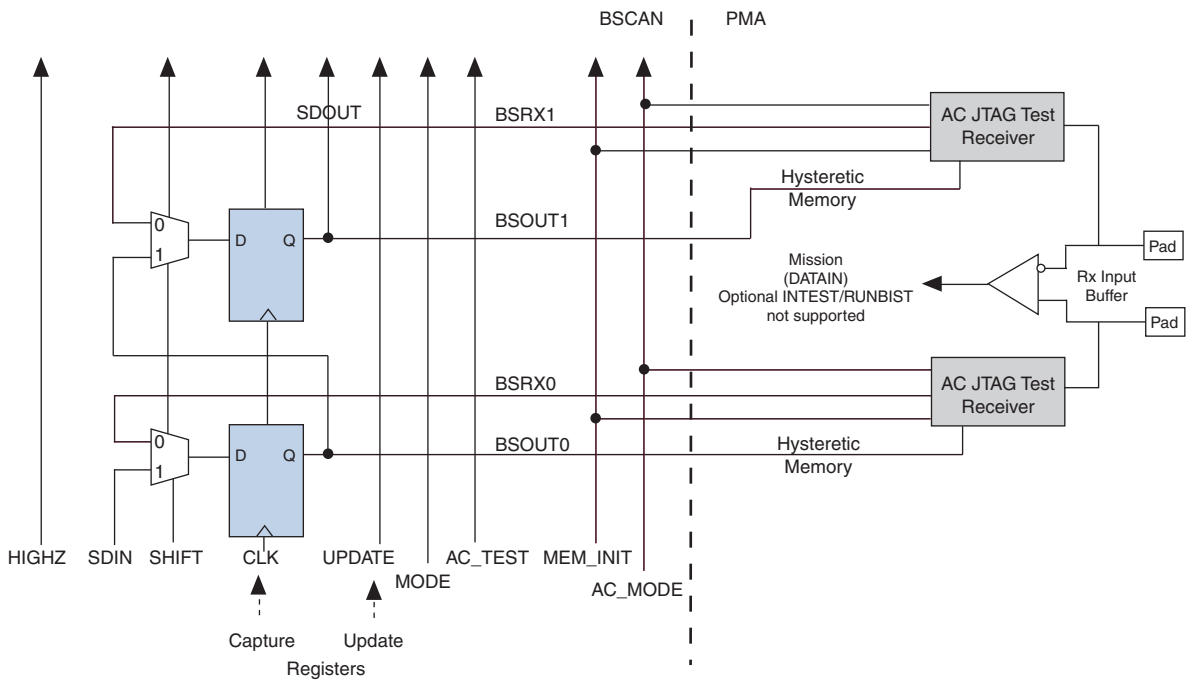


Figure 11–2 shows the Arria II GX HSSI receiver/input clock buffer BSC.

Figure 11–2. HSSI Receiver/Input Clock Buffer BSC with IEEE Std. 1149.6 BST Circuitry for Arria II GX Devices



BST Operation Control

Table 11–1 lists the boundary-scan register length for Arria II devices.

Table 11–1. Boundary-Scan Register Length for Arria II Devices

Device	Boundary-Scan Register Length
EP2AGX45	1,227
EP2AGX65	1,227
EP2AGX95	1,467
EP2AGX125	1,467
EP2AGX190	1,971
EP2AGX260	1,971
EP2AGZ225	2,274
EP2AGZ300	2,682
EP2AGZ350	2,682

Table 11–2 lists the IDCODE information for Arria II devices.

Table 11–2. 32-Bit IDCODE for Arria II Devices

Device	IDCODE (32 Bits) *(1)*			
	Version (4 Bits)	Part Number (16 Bits)	Manufacturer Identity (11 Bits)	LSB (1 Bit) *(2)*
EP2AGX45	0000	0010 0101 0001 0010	000 0110 1110	1
EP2AGX65	0000	0010 0101 0000 0010	000 0110 1110	1
EP2AGX95	0000	0010 0101 0001 0011	000 0110 1110	1
EP2AGX125	0000	0010 0101 0000 0011	000 0110 1110	1
EP2AGX190	0000	0010 0101 0001 0100	000 0110 1110	1
EP2AGX260	0000	0010 0101 0000 0100	000 0110 1110	1
EP2AGZ225	0000	0010 0100 1000 0001	000 0110 1110	1
EP2AGZ300	0000	0010 0100 0000 1010	000 0110 1110	1
EP2AGZ350	0000	0010 0100 1000 0010	000 0110 1110	1

Notes to Table 11–2:

(1) The MSB is on the left.

(2) The IDCODE LSB is always 1.

If the device is in the RESET state, when the nCONFIG or nSTATUS signal is low, the device IDCODE might not be read correctly. To read the device IDCODE correctly, you must issue the IDCODE JTAG instruction only when the nSTATUS signal is high.

For information about JTAG instructions, TAP controller state machine, timing requirements, and how to select the instruction mode, refer to "IEEE Std. 1149.1 BST Operation Control" in the *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

For Arria II GX devices, 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 HSSI pins. These instructions implement new test behaviors for HSSI pins and simultaneously behave identically to the IEEE Std. 1149.1 EXTEST instruction for non-HSSI pins.

EXTEST_PULSE Instruction Mode

The instruction code for EXTEST_PULSE is 0010001111. The EXTEST_PULSE instruction generates three output transitions:

- Driver drives the data on the falling edge of TCK in UPDATE_IR/DR.
- Driver drives the inverted data on the falling edge of TCK after entering the RUN_TEST/IDLE state.
- Driver drives the data on the falling edge of TCK after leaving the RUN_TEST/IDLE state.

If you use DC-coupling on the HSSI signals, you must execute the EXTEST instruction. If you use AC-coupling on the HSSI signals, you must execute the EXTEST_PULSE instruction. AC-coupled and DC-coupled HSSI are only supported in post-configuration mode.

EXTEST_TRAIN Instruction Mode

The instruction code for EXTEST_TRAIN is 0001001111. The EXTEST_TRAIN instruction behaves like 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.

These two instruction codes are only supported in post-configuration mode for Arria II GX devices.

You must not use the following private instructions as invoking such instructions potentially damage the device, rendering the device useless:

- 1100010000
- 0011100101
- 0011001001
- 1100010011
- 0011100110
- 0000101010

You must take precaution not to invoke such instructions at any instance. Altera recommends that you avoid toggling the JTAG pins when the device is not in used.

I/O Voltage Support in a JTAG Chain

The JTAG chain can support several different devices. However, use caution if the chain contains devices that have different V_{CCIO} levels. The output voltage level of the TDO pin must meet the specification of the TDI pin it drives.

Table 11–3 and Table 11–4 show board design recommendations to ensure proper JTAG chain operation.

Table 11–3. Supported TDO/TDI Voltage Combinations for Arria II GX Devices (Part 1 of 2)

Device	TDI Input Buffer Power	Arria II GX TDO V_{CCIO} Voltage Level in I/O Bank 8C				
		V_{CCIO} = 3.3 V *(1)*	V_{CCIO} = 3.0 V *(1)*	V_{CCIO} = 2.5 V *(2)*	V_{CCIO} = 1.8 V	V_{CCIO} = 1.5 V
Arria II GX	V_{CCIO} = 3.3 V	✓	✓	✓	✓ *(3)*	Level shifter required
	V_{CCIO} = 3.0 V	✓	✓	✓	✓ *(3)*	Level shifter required
	V_{CCIO} = 2.5 V	✓	✓	✓	✓ *(3)*	Level shifter required
	V_{CCIO} = 1.8 V	✓	✓	✓	✓ *(3)*	Level shifter required
	V_{CCIO} = 1.5 V	✓	✓	✓	✓ *(3)*	✓

Table 11–3. Supported TDO/TDI Voltage Combinations for Arria II GX Devices (Part 2 of 2)

Device	TDI Input Buffer Power	Arria II GX TDO V_{CCIO} Voltage Level in I/O Bank 8C				
		V_{CCIO} = 3.3 V *(1)*	V_{CCIO} = 3.0 V *(1)*	V_{CCIO} = 2.5 V *(2)*	V_{CCIO} = 1.8 V	V_{CCIO} = 1.5 V
Non-Arria II GX	V_{CC} = 3.3 V	✓	✓	✓	✓ *(3)*	Level shifter required
	V_{CC} = 2.5 V	✓ *(4)*	✓ *(4)*	✓	✓ *(3)*	Level shifter required
	V_{CC} = 1.8 V	✓ *(4)*	✓ *(4)*	✓ *(5)*	✓	Level shifter required
	V_{CC} = 1.5 V	✓ *(4)*	✓ *(4)*	✓ *(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.

Table 11–4. Supported TDO/TDI Voltage Combinations for Arria II GZ Devices

Device	TDI Input Buffer Power	Arria II GZ TDO V_{CCPD} Voltage Level in I/O Bank 1A	
		V_{CCPD} = 3.0 V *(1)*	V_{CCPD} = 2.5 V *(2)*
Arria II GZ	V_{CCPD} = 3.0 V	✓	✓
	V_{CCPD} = 2.5 V	✓	✓
Non-Arria II GZ	V_{CC} = 3.3 V	✓	✓
	V_{CC} = 2.5 V	✓ *(3)*	✓
	V_{CC} = 1.8 V	✓ *(3)*	✓ *(4)*
	V_{CC} = 1.5 V	✓ *(3)*	✓ *(4)*

Notes to Table 11–4:

(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) The input buffer must be 3.0-V tolerant.
(4) The input buffer must be 2.5-V tolerant.

For more information about I/O voltage support in the JTAG chain, refer to the "I/O Voltage Support in JTAG Chain" in the *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

Disabling IEEE Std. 1149.1 BST Circuitry

The IEEE Std. 1149.1 BST circuitry for Arria II devices is enabled after device power up. Because the IEEE Std. 1149.1 BST circuitry is used for BST or in-circuit reconfiguration, you must enable the circuitry only at specific times as mentioned in "IEEE Std. 1149.1 BST Circuitry" in the *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

If you do not use the IEEE Std. 1149.1 circuitry in Arria II devices, permanently disable the circuitry to ensure that you do not inadvertently enable it when it is not required.

Table 11–5 lists the pin connections necessary for disabling the IEEE Std. 1149.1 circuitry in Arria II devices.

Table 11–5. Pin Connections Necessary for Disabling IEEE Std. 1149.1 Circuitry for Arria II Devices

JTAG Pins	Connection for Disabling	
	Arria II GX Devices	Arria II GZ Devices
TMS	V_{CC} supply of Bank 8C	V_{CCPD} supply of Bank 1A
TCK	GND	
TDI	V_{CC} supply of Bank 8C	V_{CCPD} supply of Bank 1A
TDO	Leave Open	
TRST	Not available	GND

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, refer to the IEEE 1149.6 BSDL Files page on the Altera® website.

For more information about BSDL files for IEEE Std. 1149.1-compliant Arria II GZ devices, refer to the IEEE 1149.1 BSDL Files page on the Altera website.

You can also generate BSDL files (pre-configuration and post-configuration) for Arria II devices with the Quartus® II software version 9.1 and later. For the procedure to generate BSDL files using the Quartus II software, refer to Generating BSDL Files in Quartus II.

Document Revision History

Table 11–6 lists the revision history for this document.

Table 11–6. Document Revision History

Date	Version	Changes
December 2013	4.1	Updated the "EXTEST_PULSE Instruction Mode" section.
December 2010	4.0	■ Updated for the Quartus II software version 10.1 release. ■ Added Arria II GZ devices information. ■ Added "BST Architecture for Arria II Devices" and "Disabling IEEE Std. 1149.1 BST Circuitry" sections. ■ Added Table 11–3 and Table 11–5. ■ Updated Table 11–1 and Table 11–2. ■ Minor text edits.
July 2010	3.0	Updated for Arria II GX v10.0 release: ■ Updated "BST Operation Control" section. ■ Minor text edits.
November 2009	2.0	Updated for Arria II GX v9.1 release: ■ Updated Table 11–1 and Table 11–2. ■ Updated "I/O Voltage Support in a JTAG Chain" section. ■ Minor text edits.
February 2009	1.0	Initial release.

12. Power Management in Arria II Devices

AIIGX51012-3.1

This chapter describes the static and dynamic power of Arria® II devices. 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 PowerPlay Power Analyzer in the Quartus® II software optimizes all designs with Arria II 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.

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*.

This chapter includes the following sections:

- "External Power Supply Requirements" on page 12–1
- "Power-On Reset Circuitry" on page 12–1
- "Hot Socketing" on page 12–2

External Power Supply Requirements

For more information about the Arria II external power supply requirements and the power supply pin connections, refer to the following:

- For more information about Altera-recommended power supply operating conditions, refer to the *Device Datasheet for Arria II Devices* chapter.
- For more information about power supply pin connection guidelines and power regulator sharing, refer to the *Arria II Device Family Pin Connection Guidelines*.

Power-On Reset Circuitry

The Arria II power-on reset (POR) circuitry generates a POR signal to keep the device in the reset state until the power supply's voltage levels have stabilized during power-up. The POR circuitry monitors V_{CC}, V_{CCA_PLL}, V_{CCCB}, V_{CCPD}, and V_{CCIO} supplies for I/O banks 3C and 8C in Arria II GX devices, 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 the reset state.

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

The POR circuitry monitors V_{CC}, V_{CCAUX}, V_{CCCB}, V_{CCPGM}, and V_{CCPD} supplies in Arria II GZ devices. The POR circuitry keeps the Arria II GZ devices in reset state until the power supply outputs are within operating range (provided that the V_{CC} powers up fully before V_{CCAUX}).

POR circuitry is important to ensure that all the circuits in the Arria II 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 Datasheet for Arria II Devices* chapter.

For more information about MSEL pin settings, refer to the *Configuration, Design Security, and Remote System Upgrades in Arria II Devices* chapter.

Hot Socketing

Arria II I/O pins are hot-socketing compliant without the need for external components or special design requirements. Hot-socketing support in Arria II devices has the following advantages:

- You can drive the device before power up without damaging the device.
- I/O pins remain tri-stated during power up. The device does not drive out before or during power-up. Therefore, it does not affect other buses in operation.
- You can insert or remove an Arria II 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 Arria II I/O pins and transceiver before or during power up or power down without damaging the device. Arria II devices support any power-up or power-down sequence to simplify the system-level designs.

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 output buffers are turned off during system power up or power down. Also, the Arria II device does not drive out until the device is configured and working within recommended operating conditions.

Insertion or Removal of an Arria II 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 device may be inserted into or removed from a powered up system board without damaging or interfering with system-board operation.

For Arria II GX devices, 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. For Arria II GZ devices, you can power up or power down the V_{CC}, V_{CCIO}, V_{CCPD}, and V_{CCPGM} supplies in any sequence (provided that the V_{CC} powers up fully before V_{CCAUX}).

For more information about the hot-socketing specification, refer to the *Device Datasheet for Arria II Devices* chapter and the *Hot-Socketing and Power-Sequencing Feature and Testing for Altera Devices* white paper.

Hot-Socketing Feature Implementation

Arria II devices are immune to latch-up when using the hot-socketing feature. 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 for Arria II GX devices. Hot-socketing circuitry generates an internal `HOTSCKT` signal when the V_{CC}, V_{CCIO}, or V_{CCPD} power supplies for Arria II GX devices are below the threshold voltage. To support the startup current as reported by the PowerPlay Early Power Estimator (EPE) for Arria II GX devices, fully power V_{CC} before V_{CCCB} begins to ramp.

The hot-socketing feature turns off the output buffer during power up and power down of the V_{CC}, V_{CCIO}, V_{CCPD}, and V_{CCPGM} power supplies for Arria II GZ devices. To support the power-up sequence for all Arria II GZ devices, fully power V_{CC} before V_{CCAUX} begins to ramp.

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 the expected behavior for these pins to drive out during power-up and power-down sequences.

Altera uses GND as reference for the hot-socketing operation and I/O buffer designs. To ensure proper operation, Altera recommends connecting the GND between boards before connecting to the power supplies. This prevents the GND on your board from being pulled up inadvertently by a path to power through other components on your board. A pulled up GND can otherwise cause an out-of-specification I/O voltage or current condition with the Altera® device.

Document Revision History

Table 12–1 lists the revision history for this chapter.

Table 12–1. Document Revision History

Date	Version	Changes
June 2011	3.1	■ Removed Table 1-2. ■ Updated "Insertion or Removal of an Arria II Device from a Powered-Up System" and "Hot-Socketing Feature Implementation" sections. ■ Minor text edits.
December 2010	3.0	■ Updated for the Quartus II software version 10.1 release. ■ Added Arria II GZ devices information. ■ Minor text edits.
July 2010	2.0	Updated "Power-On Reset Circuitry" section for the Arria II GX v10.0 release.
June 2009	1.1	—
February 2009	1.0	Initial release.

This chapter provides additional information about the document and Altera.

About this Handbook

This handbook provides comprehensive information about the Arria® II devices.

How to Contact Altera

To locate the most up-to-date information about Altera products, refer to the following table.

Contact *(1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note to Table:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, **\qdesigns** directory, **D:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicate document titles. For example, *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	Indicate keyboard keys and menu names. For example, the 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`. The suffix `n` denotes an active-low signal. 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`).
↵	An angled arrow instructs you to press the Enter key.
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.
	A question mark directs you to a software help system with related information.
	The feet direct you to another document or website with related information.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The envelope links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.

Arria II Device Handbook

Volume 2: Transceivers

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

AIIGX5V2-3.3

Document publication date: July 2012

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

QUALITY
ISO 9001:2008
NSAI Certified

Contents

Chapter Revision Dates vii

Section 1. Transceiver Architecture for Arria II Devices

Revision History 1–1

Chapter 1. Transceiver Architecture in Arria II Devices

Transceiver Block Overview 1–3
Clock Multiplier Units (CMU) 1–6
CMU PLL 1–9
CMU0 Clock Divider 1–10
Transmitter Channel Local Clock Divider Block 1–10
Transceiver Channel Architecture 1–11
Transmitter Channel Datapath 1–12
Transmitter PCS 1–12
TX Phase Compensation FIFO 1–12
Byte Serializer 1–14
8B/10B Encoder 1–15
Transmitter PMA 1–18
Serializer 1–19
Transmitter Output Buffer 1–20
Receiver Channel Datapath 1–21
Receiver PMA 1–21
Receiver Input Buffer 1–22
CDR 1–26
Deserializer 1–29
Receiver PCS 1–30
Word Aligner 1–30
Deskew FIFO 1–41
Rate-Match FIFO 1–41
8B/10B Decoder 1–42
Byte Deserializer 1–43
Byte Ordering 1–44
RX Phase Compensation FIFO 1–46
Calibration Block 1–47
PCIe Hard IP Block 1–50
Functional Modes 1–50
Basic Mode 1–51
Deterministic Latency 1–56
GbE 1–57
Idle Ordered-Set Generation in GbE Mode 1–59
GbE Mode Reset Condition 1–60
Word Aligner in GbE Mode 1–60
Rate Match FIFO in GbE Mode 1–61
PCIe Mode 1–62
Power State Management 1–65
Transmitter Buffer Electrical Idle 1–66
Receiver Detection 1–67
Compliance Pattern Transmission Support 1–68

Receiver Status . . . 1–69
Fast Recovery Mode . . . 1–70
Electrical Idle Inference . . . 1–70
PCIe Cold Reset Requirements . . . 1–71
SDI . . . 1–72
SRIO . . . 1–74
SONET/ SDH . . . 1–76
Word Aligner in SONET/SDH Mode . . . 1–78
OC-48 and OC-96 Byte Serializer and Deserializer . . . 1–78
Byte Ordering in SONET/SDH OC-48 Mode . . . 1–79
XAUI . . . 1–79
Word Aligner in XAUI Mode . . . 1–82
Deskew FIFO in XAUI Mode . . . 1–82
Rate Match FIFO in XAUI Mode . . . 1–83
Test Modes . . . 1–85
Serial Loopback . . . 1–85
Reverse Serial Loopback . . . 1–86
Reverse Serial Pre-CDR Loopback . . . 1–87
PCIe (Reverse Parallel Loopback) . . . 1–88
Built-In Self Test (BIST) and Pseudo Random Binary Sequence (PRBS) . . . 1–88
Dynamic Reconfiguration . . . 1–91
Transceiver Port List . . . 1–94
Document Revision History . . . 1–103

Chapter 2. Transceiver Clocking in Arria II Devices

CMU PLL and Receiver CDR Input Reference Clocking . . . 2–1
refclk0 and refclk1 Pins . . . 2–3
Inter-Transceiver Block Clock Lines . . . 2–5
Dedicated CLK Input Pins on the FPGA Global Clock Network . . . 2–6
Clock Output from Left and Right PLLs in the FPGA Fabric . . . 2–6
Transceiver Channel Datapath Clocking . . . 2–6
Transmitter Channel Datapath Clocking . . . 2–6
Non-Bonded Channel Configurations . . . 2–7
Bonded Channel Configurations . . . 2–10
Receiver Channel Datapath Clocking . . . 2–18
Non-Bonded Channel Configurations . . . 2–18
Bonded Channel Configurations . . . 2–22
FPGA Fabric-Transceiver Interface Clocking . . . 2–28
Input Reference Clocks . . . 2–28
Phase Compensation FIFO Clocks . . . 2–29
Other Transceiver Clocks . . . 2–29
FPGA Fabric-Transmitter Interface Clocking . . . 2–30
Quartus II Software-Selected Transmitter Phase Compensation FIFO Write Clock . . . 2–30
User-Selected Transmitter Phase Compensation FIFO Write Clock . . . 2–37
FPGA Fabric-Receiver Interface Clocking . . . 2–42
Quartus II Software-Selected Receiver Phase Compensation FIFO Read Clock . . . 2–42
User-Selected Receiver Phase Compensation FIFO Read Clock . . . 2–51
FPGA Fabric PLL-Transceiver PLL Cascading . . . 2–56
Dedicated Left PLL Cascade Lines Network . . . 2–56
Dedicated Left and Right PLL Cascade Network in Arria II GZ Devices . . . 2–61
FPGA Fabric PLLs-Transceiver PLLs Cascading in the 780-Pin Package . . . 2–61
FPGA Fabric PLLs-Transceiver PLLs Cascading in the 1152-Pin Package . . . 2–61
FPGA Fabric PLLs-Transceiver PLLs Cascading in the 1517-Pin Package . . . 2–62
FPGA Fabric PLL-Transceiver PLL Cascading Rules . . . 2–63

Using the CMU PLL for Clocking User Logic in the FPGA Fabric 2–66
Document Revision History 2–67

Chapter 3. Configuring Multiple Protocols and Data Rates in Arria II Devices

Transceiver PLL Configurations 3–1
Creating Transceiver Channel Instances 3–2
General Requirements to Combine Channels 3–2
Control Signals 3–2
Calibration Clock and Power Down 3–2
Sharing CMU PLLs 3–3
Multiple Channels Sharing a CMU PLL 3–3
Example 1 3–3
Multiple Channels Sharing Two CMU PLLs 3–6
Example 2 3–6
Combining Receiver Only Channels 3–8
Combining Transmitter Channel and Receiver Channel Instances 3–9
Multiple Transmitter Channel and Receiver Channel Instances 3–9
Example 3 3–9
Combining Channels Configured in Protocol Functional Modes 3–10
Basic ×4 Mode 3–10
Combining Channels Using the PCIe hard IP Block with Other Channels 3–12
Combining Transceiver Instances Using PLL Cascade Clocks 3–12
Combining Transceiver Instances in Multiple Transceiver Blocks 3–13
Example 4 3–13
Summary 3–15
Document Revision History 3–16

Chapter 4. Reset Control and Power Down in Arria II Devices

User Reset and Power-Down Signals 4–1
Blocks Affected by Reset and Power-Down Signals 4–3
Transceiver Reset Sequences 4–4
All Supported Functional Modes Except PCIe Functional Mode 4–6
Bonded Channel Configuration 4–6
Non-Bonded Channel Configuration 4–10
PCIe Functional Mode 4–15
PCIe Reset Sequence 4–15
Dynamic Reconfiguration Reset Sequences 4–17
Reset Sequence with Data Rate Division in the TX Option 4–17
Reset Sequence with the Channel and TX PLL Select/Reconfig Option 4–18
Hot Socketing Reset Sequence 4–19
Power Down 4–20
Simulation Requirements 4–21
Document Revision History 4–22

Additional Information

How to Contact Altera Info–1
Typographic Conventions Info–1

Chapter Revision Dates

The chapters in this document, Arria II Device Handbook Volume 2: Transceivers, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1. Transceiver Architecture in Arria II Devices
Revised: *October 2013*
Part Number: *AIIGX52001-4.4*

Chapter 2. Transceiver Clocking in Arria II Devices
Revised: *June 2011*
Part Number: *AIIGX52002-3.1*

Chapter 3. Configuring Multiple Protocols and Data Rates in Arria II Devices
Revised: *December 2010*
Part Number: *AIIGX52003-3.0*

Chapter 4. Reset Control and Power Down in Arria II Devices
Revised: *July 2013*
Part Number: *AIIGX52004-3.2*

Section 1. Transceiver Architecture for Arria II Devices

This section provides information about Arria® II device family transceiver architecture and clocking. It also describes configuring multiple protocols, data rates, and reset control and power down in the Arria II device family. This section includes the following chapters:

- Chapter 1, Transceiver Architecture in Arria II Devices
- Chapter 2, Transceiver Clocking in Arria II Devices
- Chapter 3, Configuring Multiple Protocols and Data Rates in Arria II Devices
- Chapter 4, Reset Control and Power Down in Arria II Devices

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. Transceiver Architecture in Arria II Devices

AIIGX52001-4.4

This chapter describes all available modules in the Arria® II GX and GZ transceiver architecture and describes how these modules are used in the protocols shown in Table 1–1. In addition, this chapter lists the available test modes, dynamic reconfiguration, and ALTGX port names.

Arria II GX and GZ devices provide up to 24 full-duplex clock data recovery-based (CDR) transceivers with physical coding sublayer (PCS) and physical medium attachment (PMA), and support the serial protocols listed in Table 1–1 and Table 1–2.

Table 1–1 lists the serial protocols for Arria II GX devices.

Table 1–1. Serial Protocols for Arria II GX Devices

Protocol	Description
PCI Express® (PIPE) (PCIe®)	Gen1, 2.5 Gbps
Serial RapidIO® (SRIO)	1.25 Gbps, 2.5 Gbps, and 3.125 Gbps
Serial ATA (SATA)/ Serial Attached SCSI (SAS)	■ SATA I, 1.5 Gbps ■ SATA II, 3.0 Gbps ■ SATA III, 6.0 Gbps ■ SAS, 1.5 Gbps and 3.0 Gbps
Serial Digital Interface (SDI)	■ HD-SDI, 1.485 Gbps and 1.4835 Gbps ■ 3G-SDI, 2.97 Gbps and 2.967 Gbps
ASI	270 Mbps
Common Public Radio Interface (CPRI)	614.4 Mbps, 1228.8 Mbps, 2457.6 Mbps, 3072 Mbps, 4915.2 Mbps, and 6144 Mbps
OBSAI	768 Mbps, 1536 Mbps, 3072 Mbps, and 6144 Mbps
Gigabit Ethernet (GbE)	1.25 Gbps
XAUI	3.125 Gbps to 3.75 Gbps for HiGig/HiGig+ support
SONET/SDH	■ OC-12 (622 Mbps) ■ OC-48 (2.488 Gbps)
GPON	1.244 uplink and 2.488 downlink
SerialLite II	0.6 Gbps to 3.75 Gbps
Interlaken	—
CEI	—

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Table 1–2 lists the serial protocols for Arria II GZ devices.

Table 1–2. Serial Protocols for Arria II GZ Devices

Protocol	Description
PCIe	Gen2, 5.0 Gbps
SRIO	1.25 Gbps, 2.5 Gbps, and 3.125 Gbps
Serial ATA (SATA)/ Serial Attached SCSI (SAS)	■ SATA I, 1.5 Gbps ■ SATA II, 3.0 Gbps ■ SATA III, 6.0 Gbps ■ SAS, 1.5 Gbps and 3.0 Gbps
Serial Digital Interface (SDI)	■ HD-SDI, 1.485 Gbps and 1.4835 Gbps ■ 3G-SDI, 2.97 Gbps and 2.967 Gbps
ASI	270 Mbps
Common Public Radio Interface (CPRI)	614.4 Mbps, 1228.8 Mbps, 2457.6 Mbps, 3072 Mbps, 4915.2 Mbps, and 6144 Mbps
OBSAI	768 Mbps, 1536 Mbps, 3072 Mbps, and 6144 Mbps
Gigabit Ethernet (GbE)	1.25 Gbps
XAUI	3.125 Gbps to 3.75 Gbps for HiGig/HiGig+ support
SONET/SDH	■ OC-12 (622 Mbps) ■ OC-48 (2.488 Gbps) ■ OC-96 (4.976 Gbps)
GPON	1.244 uplink and 2.488 downlink
Interlaken	40G with 10 channels at 6.375 Gbps
CEI	6.375 Gbps

You can implement these protocols through the ALTGX MegaWizard™ Plug-In Manager, which also offers the highly flexible Basic functional mode to implement proprietary serial protocols.

Transceiver Block Overview

Arria II GX devices offer two to four transceiver blocks per device while Arria II GZ devices offer up to six transceiver blocks. Each block consists of four fully-duplex (transmitter and receiver) channels, located on the left side of the device (in a die-top view).

Figure 1–1 shows the die-top view of the transceiver block locations in Arria II GX devices.

Figure 1–1. Transceiver Channels for Arria II GX Devices

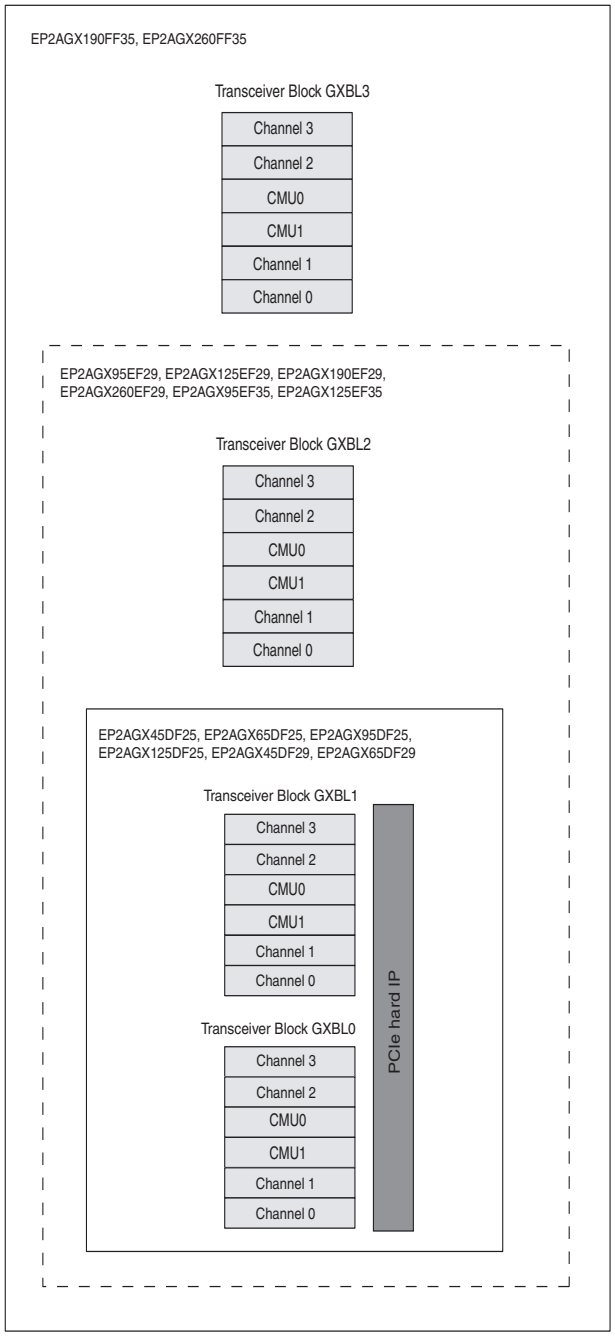


Figure 1–2 shows the die top view of the transceiver block locations in Arria II GZ devices.

Figure 1–2. Transceiver Channels for Arria II GZ Devices

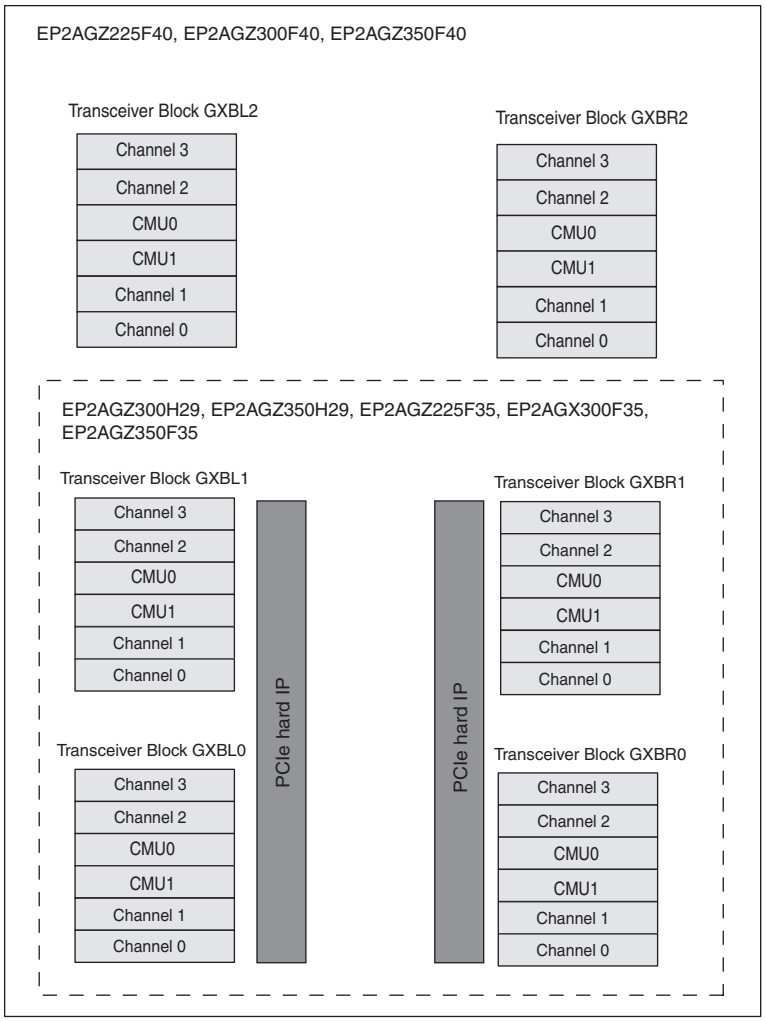
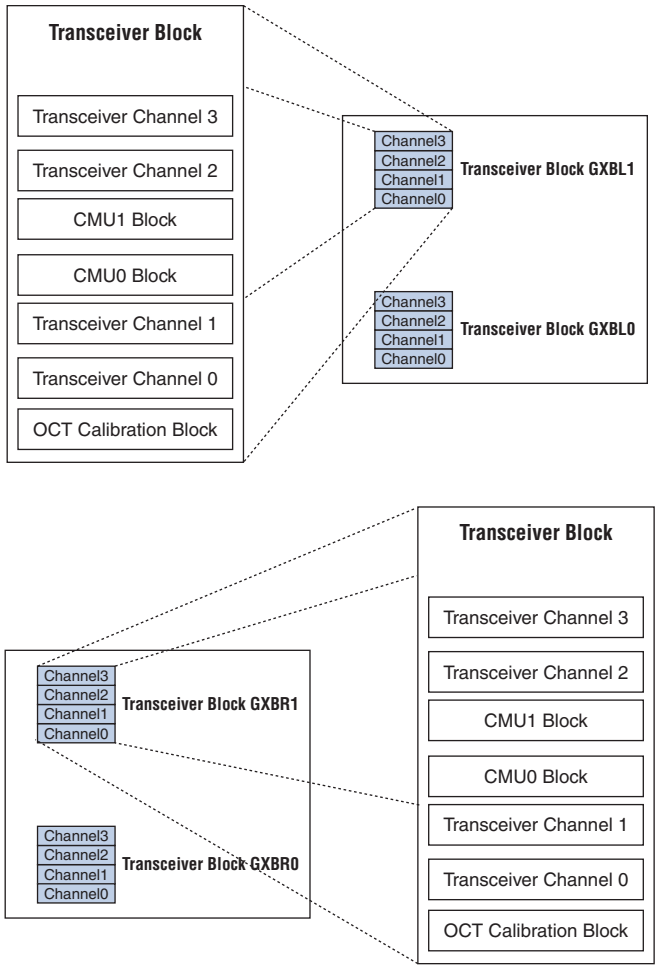


Figure 1–3 shows the block diagram of the transceiver block architecture for Arria II GX and GZ devices.

Figure 1–3. Top-Level View of a Transceiver Block for Arria II GX and GZ Devices



The following sections describe all the modules of the transceiver blocks. The input and output ports of these modules are described in the module sections, and are listed in the "Transceiver Port List" on page 1–94.

Clock Multiplier Units (CMU)

Each transceiver block contains two CMU blocks, which contain a CMU phase-locked loop (PLL) that provides clocks to all the transmitter channels in the same transceiver block. These two CMU blocks can provide two independent high-speed clocks per transceiver block.

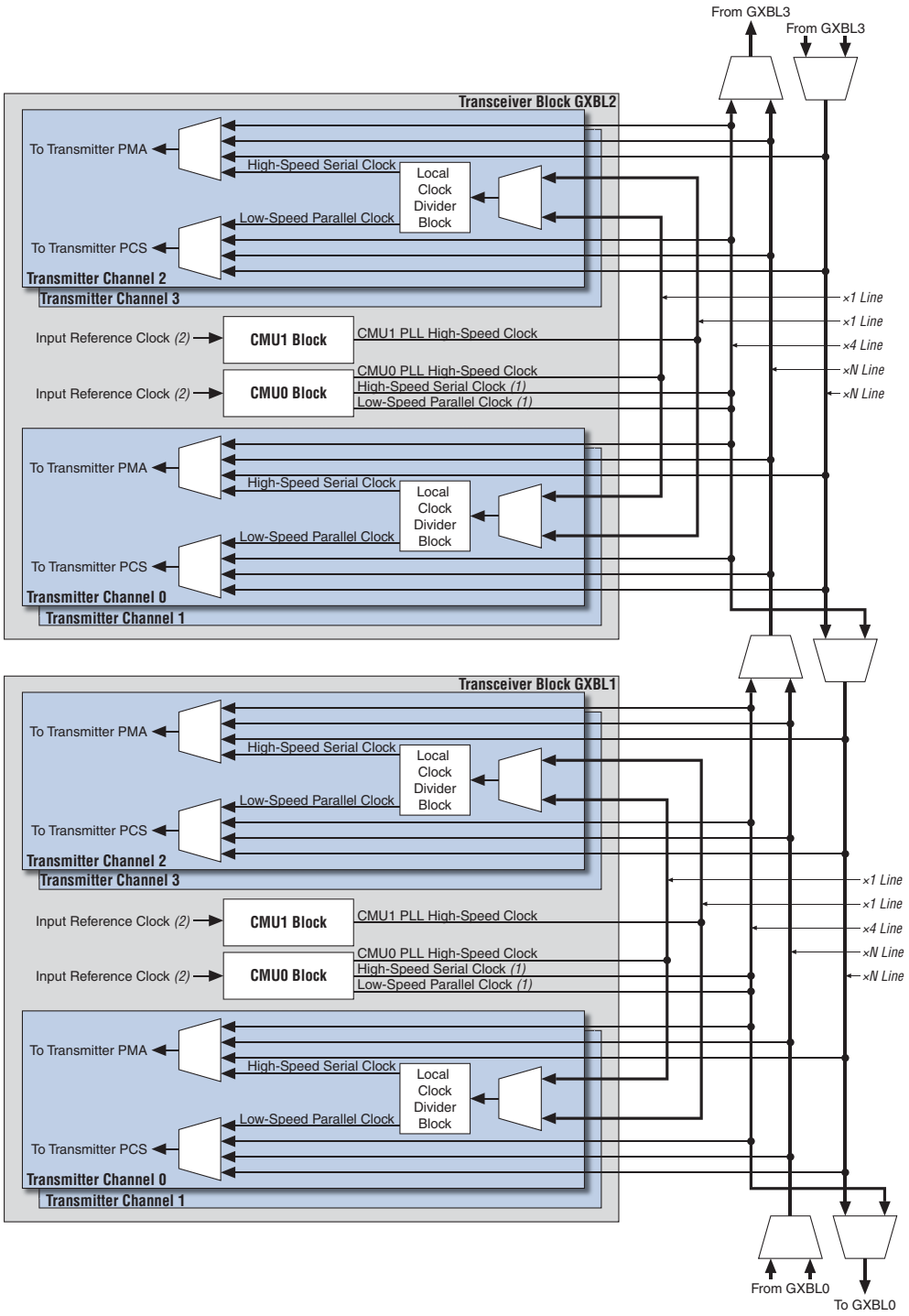


The CMU PLL is also known as the TX PLL.

The CMU PLLs in `CMU0` and `CMU1` are identical and each transmitter channel in the transceiver block can receive a high-speed clock from either of the two CMU PLLs. However, the `CMU0` block has an additional clock divider after the `CMU0` PLL to support bonded functional modes where multiple channels share a common clock to reduce skew between the channels. With the ALTGX MegaWizard Plug-In Manager, you can select the bonded functional modes used in ×4 Basic, PCIe, and XAUI.

Figure 1–4 shows a top-level block diagram of the connections between the CMU blocks and the transceiver channels.

Figure 1–4. Top-Level Diagram of CMU Block Connections in a Transceiver Block



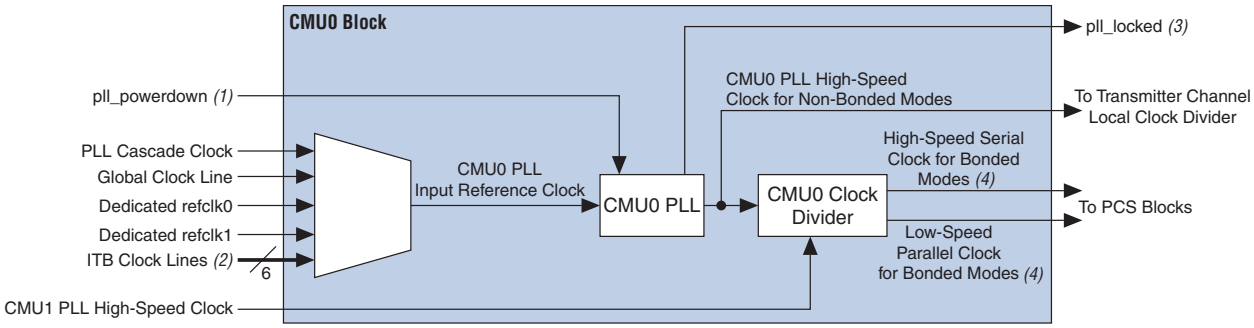
Notes to Figure 1–4:

(1) Clocks provided to support bonded channel functional mode.

(2) For more information, refer to the *Transceiver Clocking for Arria II Devices* chapter.

Figure 1–5 and Figure 1–6 show the top-level block diagram of CMU0 and CMU1 blocks, respectively.

Figure 1–5. CMU0 Block Diagram



Notes to Figure 1–5:

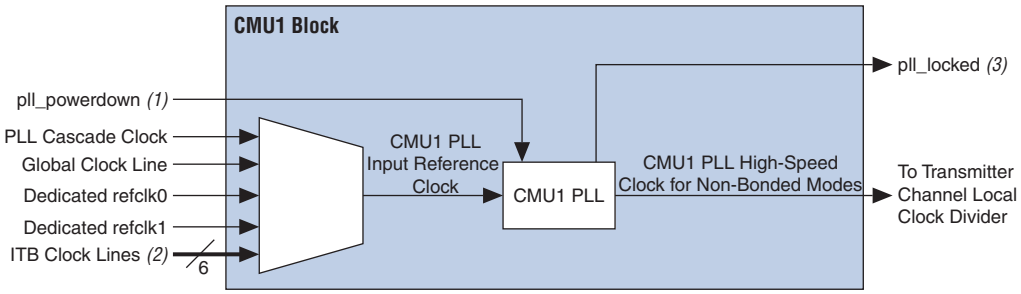
(1) Although each CMU PLL has its own pll_powerdown port, the ALTGX MegaWizard Plug-In Manager instantiation provides only one port per transceiver block. This port power downs one or both CMU PLLs (if used).

(2) The inter-transceiver block (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.

(3) There is one pll_locked signal per CMU PLL.

(4) Used in ×4, ×8, and XAUI functional modes. In ×8 functional mode, only the CMU0 channel of the master transceiver block provides clock output to all eight transceiver channels configured in PCIe functional mode.

Figure 1–6. CMU1 Block Diagram



Notes to Figure 1–6:

(1) Although each CMU PLL has its own pll_powerdown port, the ALTGX MegaWizard Plug-In Manager instantiation provides only one port per transceiver block. This port power downs one or both CMU PLLs (if used).

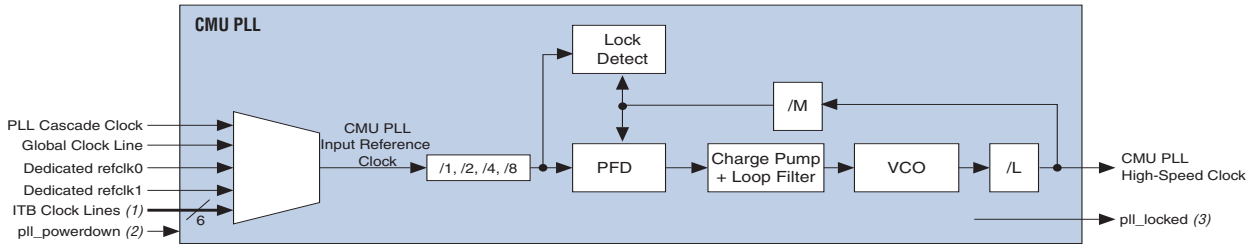
(2) The ITB clock lines shown are the maximum value. The actual number of ITB lines in your device depends on the number of transceiver blocks on one side of the device.

(3) There is one pll_locked signal per CMU PLL.

CMU PLL

Figure 1–7 shows the block diagram of the CMU PLL.

Figure 1–7. Diagram of the CMU PLL



Notes to Figure 1–7:

(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.

(2) Although each CMU PLL has its own `pll_powerdown` port, the ALTGX MegaWizard Plug-In Manager instantiation provides only one port per transceiver block. This port power downs one or both CMU PLLs (if used).

(3) There is one `pll_locked` signal per CMU PLL.

For more information about input reference clocks, refer to the "CMU PLL and Receiver CDR Input Reference Clocks" section of the *Transceiver Clocking in Arria II Devices* chapter.

The phase frequency detector (PFD) in the CMU PLL tracks the voltage-controlled oscillator (VCO) output with the input reference clock. This VCO runs at half the serial data rate. The CMU PLL generates the high-speed clock from the input reference clock through the two divider blocks (/M and /L) in the feedback path. Table 1–3 lists the available /M and /L settings, which are set automatically in the Quartus® II software, based on the input reference clock frequency and serial data rate.

Table 1–3. Multiplier Block Heading to Clock Divider for Arria II Devices

Multiplier Block	Available Values
/M	1, 4, 5, 8, 10, 16, 20, 25
/L	1, 2, 4

You can set the PLL bandwidth in the ALTGX megafunction.

The high-speed clock output from the CMU PLL is forwarded to the `CMU0` clock divider block in bonded functional modes and the transmitter channel local clock divider block in non-bonded functional modes. The output of either clock divider block provides clocks for the PCS and PMA blocks.

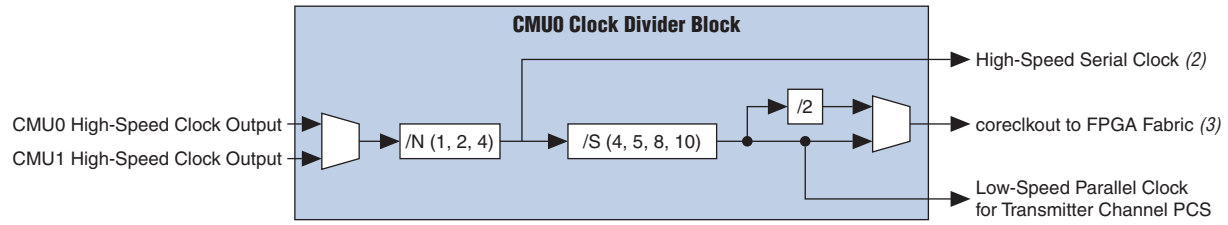
For more information about using two CMU PLLs to configure multiple transmitter channels, refer to the *Configuring Multiple Protocols and Data Rates in Arria II Devices* chapter.

CMU0 Clock Divider

The clock divider is only available only in the CMU0 block and is used in bonded functional modes.

Figure 1–8 shows a diagram of the CMU0 clock divider block.

Figure 1–8. CMU0 Clock Divider Block ***(Note 1)***



Notes to Figure 1–8:

(1) The Quartus II software automatically selects all the divider settings based on the input clock frequency, data rate, deserialization width, and channel width settings.

(2) The high-speed serial clock is available to all the transmitter channels in the transceiver block. In a ×8 configuration, only the CMU0 clock divider of the master transceiver block provides the high-speed serial clock to all eight channels.

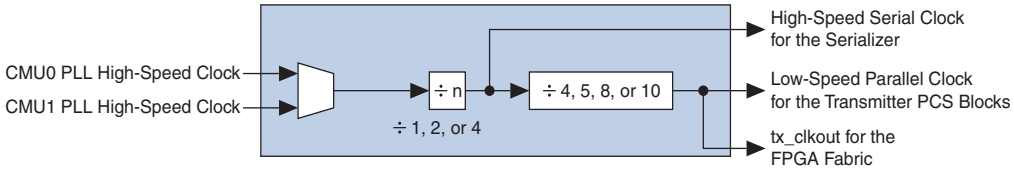
(3) If the byte serializer block is enabled in bonded channel modes, the coreclkout clock output is half the frequency of the low-speed parallel clock. Otherwise, the coreclkout clock output is the same frequency as the low-speed parallel clock.

Transmitter Channel Local Clock Divider Block

Each transmitter channel contains a local clock divider block used automatically by the Quartus II software for non-bonded functional modes (for example, ×1 PCIe, Gbps Ethernet (GbE), SONET/SDH, and SDI mode). This block allows each transmitter channel to run at /1, /2, or /4 of the CMU PLL output data rate.

Figure 1–9 shows the transmitter local clock divider block.

Figure 1–9. Transmitter Local Clock Divider Block

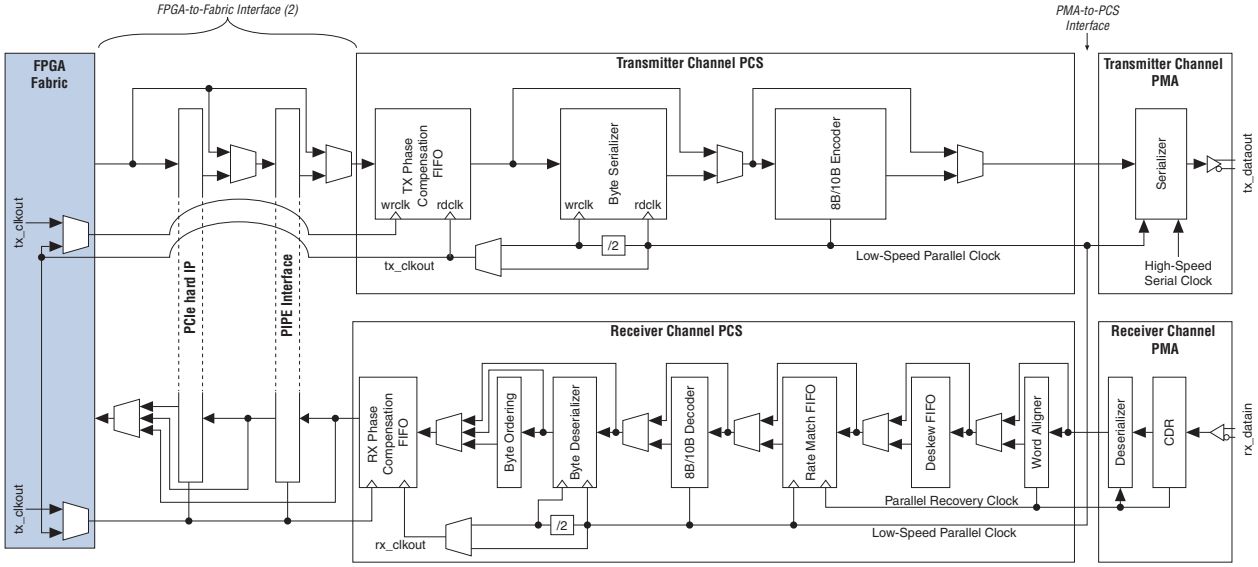


For more information about transceiver channel local clock divider block clocking, refer to the "Transceiver Channel Datapath Clocking" section in the *Transceiver Clocking in Arria II Devices* chapter.

Transceiver Channel Architecture

Each transceiver channel consists of a transmitter channel and a receiver channel. Each transmitter or receiver channel comprises the channel PCS and channel PMA blocks. Figure 1–10 shows the Arria II GX and GZ transceiver channel architecture.

Figure 1–10. Transceiver Channel Architecture for Arria II GX and GZ Devices *(Note 1)*



Notes to Figure 1–10:

(1) Shaded boxes are in the FPGA; unshaded boxes are in the I/O periphery.

(2) The PCIe hard IP block and PIPE interface are used only when the FPGA design includes the PCIe megafunction. For more information about the use of these two blocks, refer to the *PCI Express Compiler User Guide*.

The FPGA fabric-to-transceiver interface and the PMA-to-PCS interface can support an 8, 10, 16, or 20 bit-width data bus.

The transceiver channel is available in two modes:

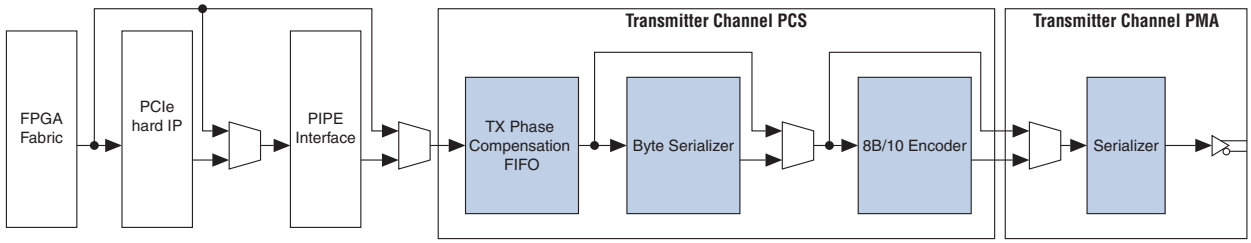
- Single-width mode—In this mode, the PMA-to-PCS interface uses an 8- or 10-bit wide data bus. The FPGA fabric-to-transceiver interface supports an 8- or 10-bit wide data bus, with the byte serializer/deserializer disabled. When the byte serializer/deserializer is enabled, the FPGA fabric-to-transceiver interface supports a 16 or 20 bit-width data bus.
- Double-width mode—In this mode, both the PMA-to-PCS interface and the FPGA fabric-to-transceiver uses an 16- and 20-bit wide data bus. The byte serializer/deserializer is supported in Arria II GZ devices, but not in Arria II GX devices. This mode is only supported for BASIC or Deterministic Latency protocol, used for CPRI and OBSAI interfaces.

Transmitter Channel Datapath

This section describes the Arria II GX and GZ transmitter channel datapath architecture. The sub-blocks in the transmitter datapath are described in order from the transmitter (TX) phase compensation FIFO buffer at the FPGA fabric-to-transceiver interface to the transmitter input buffer.

Figure 1–11 shows the transmitter channel datapath.

Figure 1–11. Transmitter Channel Datapath



Transmitter PCS

This section describes the transmitter PCS modules, which consists of the TX phase compensation FIFO, byte serializer, and 8B/10B encoder.

The `tx_digitalreset` signal resets all modules in the transmitter PCS block.

For more information about the `tx_digitalreset` signal, refer to the *Reset Control and Power Down in Arria II Devices* chapter.

TX Phase Compensation FIFO

This FIFO compensates for the phase difference between the low-speed parallel clock and the FPGA fabric interface clock. Table 1–4 lists the available modes for the TX phase compensation FIFO.

Table 1–4. Transmitter Phase Compensation FIFO Modes for Arria II Devices

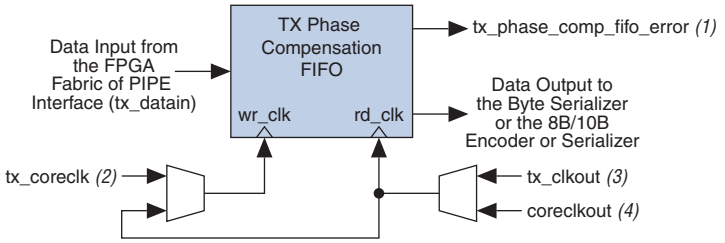
Mode	FIFO Depth	Latency Through FIFO	Applicable Functional Modes *(1)*
Low Latency	4-words deep	2-to-3 parallel clock cycles	All functional modes except PCIe and Deterministic Latency
High Latency	8-words deep	4-to-5 parallel clock cycles	PCIe
Register	—	1	Deterministic Latency

Note to Table 1–4:

(1) Automatically set when you select a protocol in the ALTGX MegaWizard Plug-In Manager.

Figure 1–12 shows the datapath and clocking of the TX phase compensation FIFO.

Figure 1–12. TX Phase Compensation FIFO



Notes to Figure 1–12:

(1) The `tx_phase_comp_fifo_error` is optional and available in all functional modes. This signal is asserted high to indicate an overflow or underflow condition.

(2) Use this optional clock for the FIFO write clock if you instantiate the `tx_coreclk` port in the ALTGX MegaWizard Plug-In Manager, regardless of the channel configurations. Otherwise, the same clock used for the read clock is also used for the write clock. Ensure that there is 0 parts per million (PPM) frequency difference between the `tx_coreclk` clock and the read clock of the FIFO.

(3) The `tx_clkout` low-speed parallel clock is from the local clock divider from the associated transmitter channel and is used in non-bonded configurations.

(4) The `coreclkout` clock is from the `CMU0` block of the associated transceiver block or the master transceiver block for ×4 bonded or ×8 bonded channel configurations, respectively.

For more information about TX phase compensation FIFO clocking, refer to the "Limitation of the Quartus II Software-Selected Transmitter Phase Compensation FIFO Write (or Read) Clocks" section in the *Transceiver Clocking in Arria II Devices* chapter.

An optional `tx_phase_comp_fifo_error` port is available in all functional modes and is asserted high in an overflow or underflow condition. If this signal is asserted, ensure that there is 0 PPM difference between the TX phase compensation FIFO read and write clocks.

The output of this block can go to any of the following blocks:

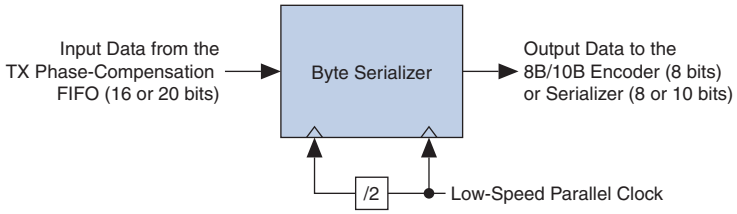
- Byte serializer—If you enable this block.
- 8B/10B encoder—If you disable the byte serializer, but enable the 8B/10B encoder and your channel width is either 8 or 16 bits.
- Serializer—If you disable both the byte serializer and the 8B/10B encoder, or if you use low-latency PCS bypass mode.

Byte Serializer

In Arria II GX devices, you cannot enable the byte serializer in double-width mode. However, in Arria II GZ devices, you can enable both double-width and the byte serializer to achieve a 32- or 40-bit PCS-FPGA interface.

Figure 1–13 shows the byte serializer datapath for Arria II GX devices.

Figure 1–13. Byte Serializer Datapath for Arria II GX Devices



The byte serializer divides the input datapath width by two. This allows you to run the transceiver channel at higher data rates while keeping the FPGA fabric frequency within the maximum limit. This module is required in configurations that exceed the FPGA fabric-to-transceiver interface clock upper frequency limit. It is optional in configurations that do not exceed the FPGA fabric-to-transceiver interface clock upper frequency limit.

For example, if you want to run the transceiver channel at 3.125 Gbps, without the byte serializer, the FPGA fabric interface clock frequency must be 312.5 MHz (3.125 Gbps/10), which violates the FPGA fabric interface frequency limit. When you use the byte serializer, the FPGA fabric interface frequency is 156.25 MHz (3.125 Gbps/20).

For more information about the maximum frequency limit for the FPGA fabric-to-transceiver interface, refer to the *Device Datasheet for Arria II Devices*.

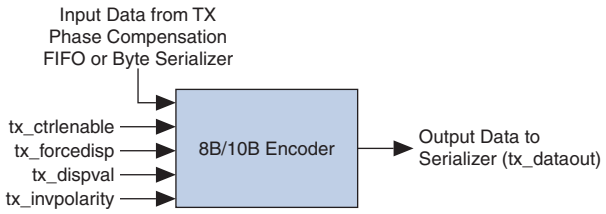
The byte serializer forwards the data from the TX phase compensation FIFO LSByte first. For example, assuming a channel width of 20 bits, the byte serializer sends out the least significant word `datain[9:0]` of the parallel data from the FPGA fabric, followed by `datain[19:10]`.

The data from the byte serializer is forwarded to the 8B/10B encoder if the module is enabled and the input data width is 16 bits. Otherwise, the output is forwarded to the serializer module in the transceiver PMA block.

8B/10B Encoder

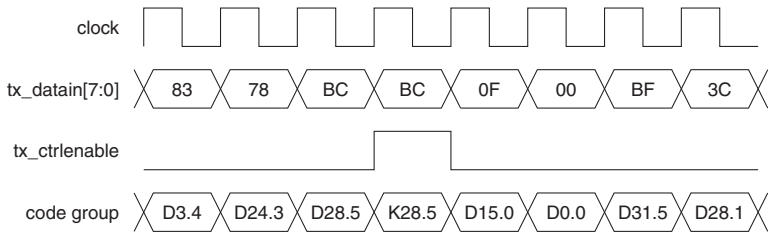
Figure 1–14 shows the inputs and outputs of the 8B/10B encoder.

Figure 1–14. 8B/10B Encoder



The 8B/10B encoder generates 10-bit code groups from the 8-bit data and 1-bit control identifier. If the `tx_ctrlenable` input is high, the 8B/10B encoder translates the 8-bit input data to a 10-bit control word (Kx.y). Otherwise, the 8B/10B encoder translates the 8-bit input data to a 10-bit data word (Dx.y). Figure 1–15 shows an example of how the second 8'hBC data is encoded as a control word, while the reset of the data are encoded as a data word.

Figure 1–15. Control Word and Data Word Transmission

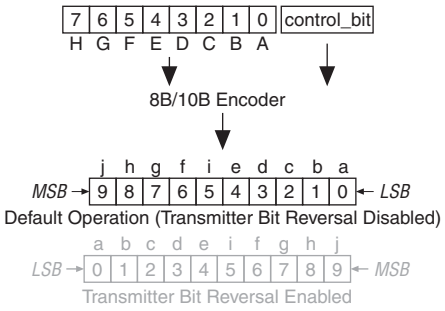




The IEEE 802.3 8B/10B encoder specification identifies only a set of 8-bit characters for which `tx_ctrlenable` should be asserted. If you assert `tx_ctrlenable` for any other set of characters, the 8B/10B encoder might encode the output 10-bit code as an invalid code (it does not map to a valid Dx.y 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. Altera recommends not asserting `tx_ctrlenable` for unsupported 8-bit characters.

Figure 1–16 shows the conversion format. The LSB is transmitted first by default. You can, however, enable the **Transmitter Bit Reversal** option in the ALTGX MegaWizard Plug-In Manager to allow reversing the transmit bit order (MSB first) before it is forwarded to the serializer.

Figure 1–16. 8B/10B Conversion Format



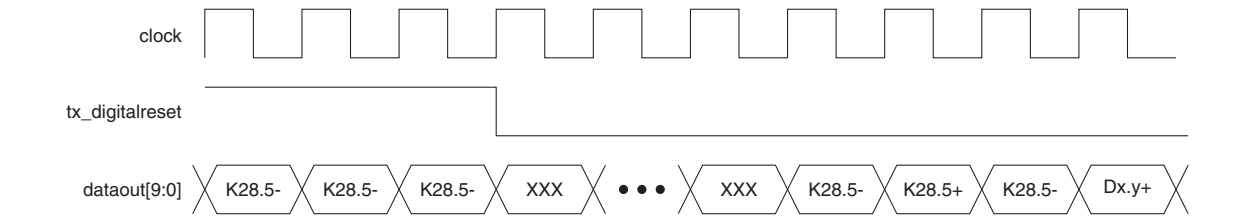
During reset, the running disparity and data registers are cleared. Also, the 8B/10B encoder outputs a K28.5 pattern from the RD- column continuously until `tx_digitalreset` is de-asserted. The input data and control code from the FPGA fabric is ignored during the reset state. After power up or reset, 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 1–17 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 1–17. 8B/10B Encoder Output during tx_digitalreset Assertion



In Basic functional mode, you can use the `tx_forcedisp` and `tx_dispval` ports to control the running disparity of the output from the 8B/10B encoder. 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 match. If the forced disparity flips the current running disparity, the downstream 8B/10B decoder might detect a disparity error.

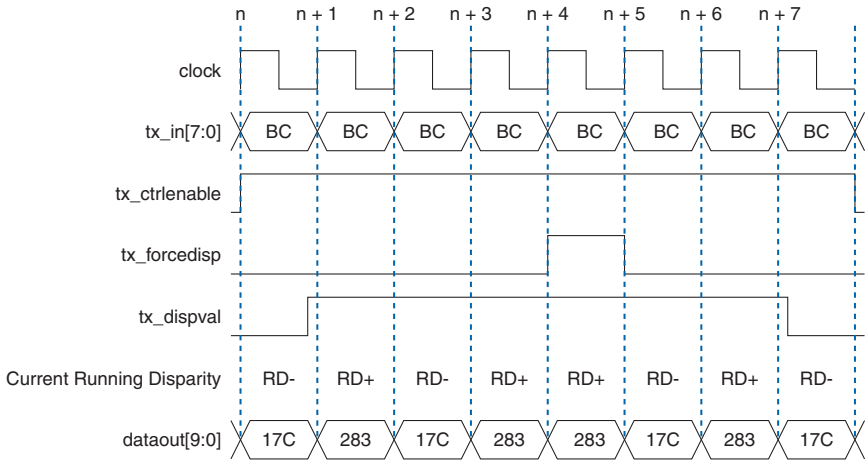
Table 1–5 lists the tx_forcedisp and tx_dispval port values and the effects they have on the data.

Table 1–5. tx_forcedisp and tx_dispval Port Values for Arria II Devices

tx_forcedisp	tx_dispval	Description
0	X	Current running disparity has no change.
1	0	Encoded data has positive disparity.
1	1	Encoded data has negative disparity.

Figure 1–18 shows an example of tx_forcedisp and tx_dispval port use, where data is shown in hexadecimal radix.

Figure 1–18. 8B/10B Encoder Force Running Disparity Operations



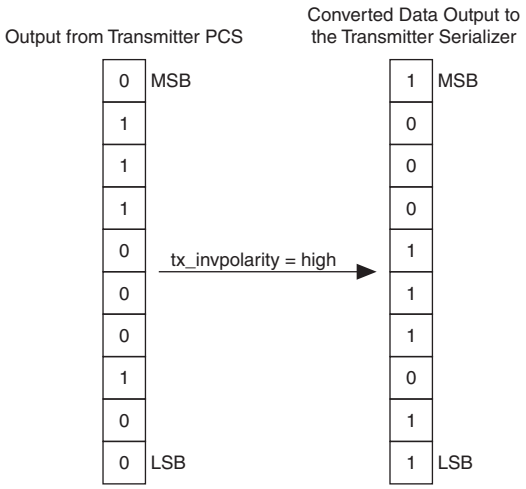
In this example, a series of K28.5 code groups are continuously sent. The stream alternates between a positive running disparity K28.5 (RD+) and a negative running disparity K28.5 (RD-) to maintain a neutral overall disparity. The current running disparity at time n + 3 indicates that the K28.5 in time n + 4 must 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.

The optional tx_invpolarity port is available in all functional modes to dynamically enable the transmitter polarity inversion feature as a workaround to board re-spin or a major update to the FPGA fabric design when the positive and negative signals of a serial differential link are accidentally swapped during board layout.

A high value on the tx_invpolarity port inverts the polarity of every bit of the input data word to the serializer in the transmitter datapath. Correct data is seen by the receiver, because inverting the polarity of each bit has the same effect as swapping the positive and negative signals of the differential link. 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 1–19 shows an example result with the tx_invpolarity feature in a 10-bit wide datapath configuration.

Figure 1–19. Transmitter Polarity Inversion



Transmitter PMA

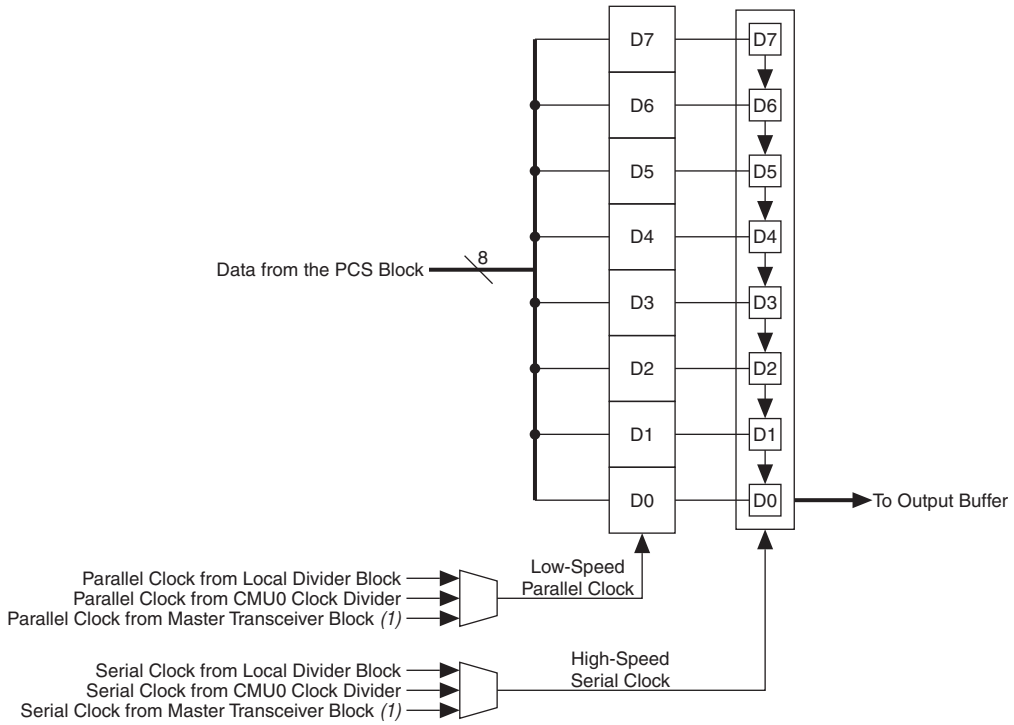
This section describes the transmitter PMA modules that consist of the serializer and the transmitter output buffer.

The tx_analogreset signal resets all modules in the transmitter PMA block. For more information about this signal, refer to the *Reset Control and Power Down in Arria II Devices* chapter.

Serializer

The serializer converts the incoming low-speed parallel signal from the transceiver PCS to the high-speed serial data and sends its LSB first to the transmitter output buffer. Figure 1–20 shows the serializer block diagram in an 8-bit PCS-to-PMA interface.

Figure 1–20. Serializer Block in an 8-Bit PCS-PMA Interface

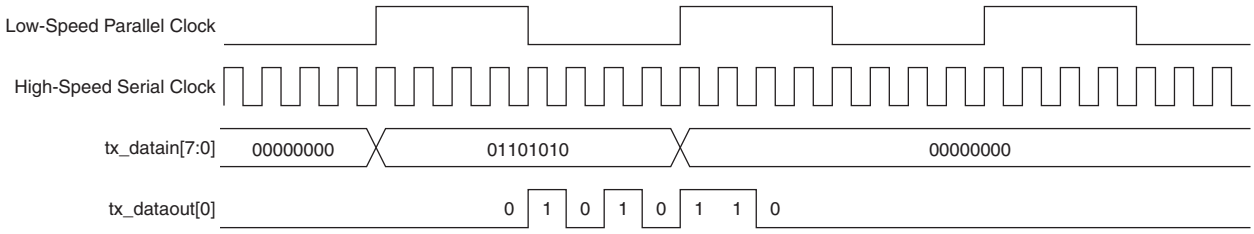


Note to Figure 1–20:

(1) This clock is provided by the CMU0 clock divider of the master transceiver block and is only used in ×8 mode.

Figure 1–21 shows an example of serialized data with a 8'b01101010 value.

Figure 1–21. Serializer Bit Order ***(Note 1)***



Note to Figure 1–21:

(1) 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 and GZ transmitter output buffers support the 1.5-V pseudo current mode logic (PCML) I/O standard and can drive 40 inches of FR4 trace (with 50-Ω impedance) across two connectors.

For data rates > 3.75 Gbps, Altera recommends limiting the FR4 trace length to 15 inches.

The transmitter output buffer power supply (V_{CCH}) only provides voltage to the transmitter output buffers in the transceiver channels. This is set to **1.5 V** in the ALTGX MegaWizard Plug-In Manager. The common mode voltage (V_{CM}) for the Arria II GX and GZ transmitter output buffers is 650 mV.

To improve signal integrity, the transmitter output buffer has the following additional circuitry, which you can set in the ALTGX MegaWizard Plug-In Manager:

- Programmable differential output voltage (V_{OD})—This feature allows you to customize the V_{OD} to handle different trace lengths, various backplanes, and various receiver requirements.
- Programmable pre-emphasis—Pre-emphasis boosts high frequencies in the transmit data signal, which might be attenuated in the transmission media because of data-dependent jitter and other intersymbol interference (ISI) effects. It equalizes the frequency response at the receiver so the differences between the low-frequency and high-frequency components are reduced, minimizing the ISI effects from the transmission medium.

  Pre-emphasis requirements increase as data rates through legacy backplanes increase. Using pre-emphasis can maximize the data eye opening at the far-end receiver.
- Programmable differential on-chip termination (OCT)—The Arria II GX transmitter buffer includes a differential OCT of 100 Ω . The Arria II GZ transmitter buffer includes a differential OCT of 85, 100, 120, and 150 Ω . The resistance is adjusted in the calibration block to compensate for temperature, voltage, and process changes (for more information, refer to "Calibration Block" on page 1–47).

  You can set the transmitter termination setting in the ALTGX MegaWizard Plug-In Manager or through the Quartus II Assignment Editor by setting the assignment output termination to **85, 100, 120, or 150** for Arria II GZ devices or **100** for Arria II GX devices on the transmitter output buffer.

  You can disable OCT and use external termination. In this case, the transmitter common mode is tri-stated.

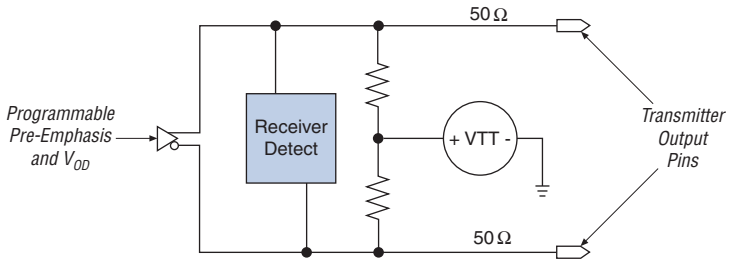
  The Arria II GX and GZ transmitter output buffers in the transceiver block are current-mode drivers. The resulting V_{OD} is a function of the transmitter termination value.

- Receiver-detect capability to support PCIe functional mode—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. For more information, refer to "PCIe Mode" on page 1–62.
- Tristate-able transmitter buffer to support PCIe electrical idle—This feature is only active in PCIe mode to work hand-in-hand with the receiver-detect capability. For more information, refer to "PCIe Mode" on page 1–62.

For more information about the available settings in each feature, refer to the *Device Datasheet for Arria II Devices.*

Figure 1–22 shows the transmitter output buffer block diagram.

Figure 1–22. Transmitter Output Buffer

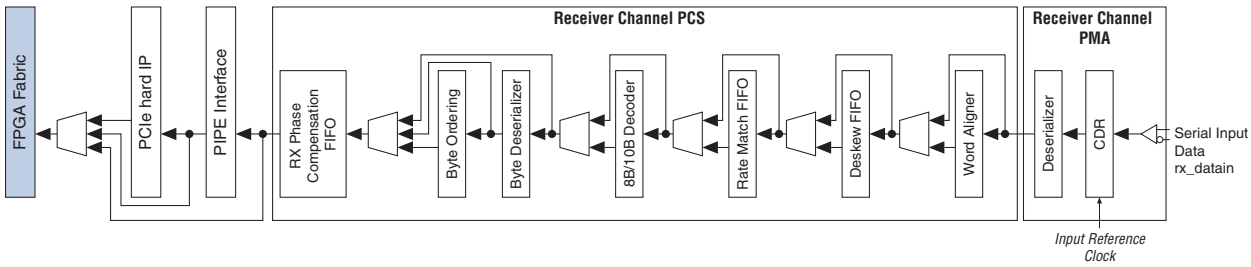


Receiver Channel Datapath

This section describes the Arria II GX and GZ receiver channel datapath architecture. The sub-blocks in the receiver datapath are described in order from the receiver input buffer to the receiver (RX) phase compensation FIFO buffer at the FPGA fabric-to-transceiver interface.

Figure 1–23 shows the receiver channel datapath in Arria II GX and GZ devices.

Figure 1–23. Receiver Channel Datapath



Receiver PMA

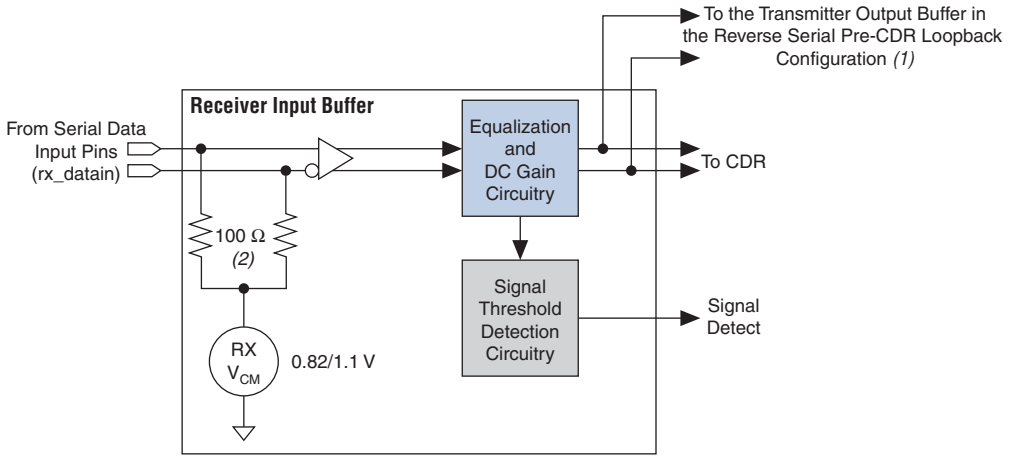
This section describes the receiver PMA modules, which consists of the receiver input buffer, CDR, and deserializer.

The `rx_analogreset` signal resets all modules in the receiver PMA block. For more information about this signal, refer to the *Reset Control and Power Down in Arria II Devices* chapter.

Receiver Input Buffer

The receiver input buffer receives serial data from the rx_datain port and feeds it to the CDR unit. Figure 1–24 shows the receiver input buffer.

Figure 1–24. Receiver Input Buffer for Arria II GX Devices



Notes to Figure 1–24:

(1) For more information about reverse serial pre-CDR loopback mode, refer to "Test Modes" on page 1–85.

(2) For Arria II GZ devices, this is 85, 100, 120, and 150 Ω.

Table 1–6 lists the electrical features supported by the receiver input buffer.

Table 1–6. Electrical Features Supported by the Receiver Input Buffer for Arria II Devices *(Note 1)*

I/O Standard	Programmable Common Mode Voltage (V)	Coupling
1.5 V PCML	0.82	AC, DC
2.5 V PCML	0.82	AC
LVPECL	0.82	AC
LVDS	1.1	AC, DC

Note to Table 1–6:

(1) The differential OCT setting for Arria II GX and GZ transmitters and receivers is 85, 100, 120, and 150 Ω for Arria II GZ devices or 100 Ω for Arria II GX devices.

The following sections describe the features supported in the Arria II GX and GZ receiver input buffers.

Programmable Differential OCT

The Arria II GX transmitter buffer includes a differential OCT of 100 Ω. The Arria II GZ transmitter buffer includes a differential OCT of 85, 100, 120, and 150 Ω. The resistance is adjusted in the calibration block to compensate for temperature, voltage, and process changes (for more information, refer to "Calibration Block" on page 1–47). You can set this option in the Quartus II Assignment Editor by setting the assignment input termination to **OCT 100** Ω for Arria II GX devices and **OCT 85, 100, 120, and 150** Ω for Arria II GZ devices on the receiver input buffer.

Programmable Common Mode Voltage

The Arria II GX and GZ receivers have on-chip biasing circuitry to establish the required common mode voltage at the receiver input that supports two common mode voltage settings of 0.82 V and 1.1 V. You can select the voltage in the ALTGX MegaWizard Plug-In Manager. For the I/O standards supported by each common mode voltage setting, refer to Table 1–6.

This feature is effective only if you use programmable OCT for the receiver input buffers as well. If you use external termination, you must implement off-chip biasing circuitry to establish the common mode voltage at the receiver input buffer.

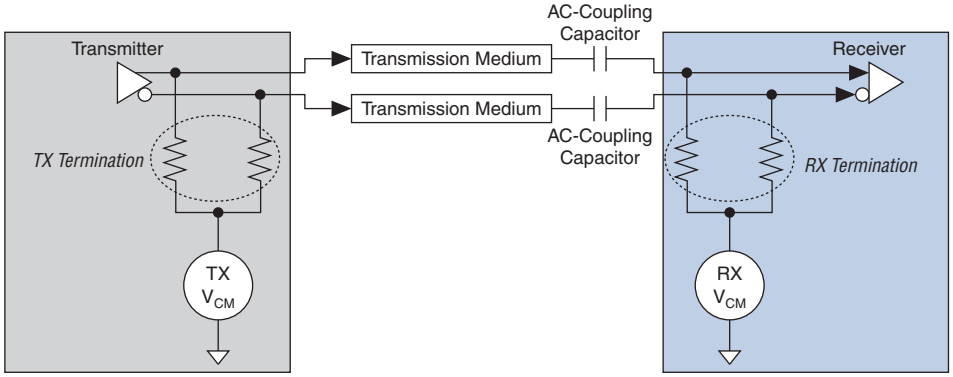
AC and DC Coupling

A high-speed serial link can either be AC-coupled or DC-coupled, depending on the serial protocol implementation. Most of the serial protocols require links to be AC-coupled, protocols similar to SONET optionally allow DC coupling.

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. AC-coupled links are required in GbE, PCIe, SRIO, SDI, and XAUI protocols.

Figure 1–25 shows an AC-coupled link.

Figure 1–25. AC-Coupled Link



In a DC-coupled link, the transmitter DC common mode voltage is seen unblocked at the receiver input 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 1–26 shows a DC-coupled link.

Figure 1–26. DC-Coupled Link

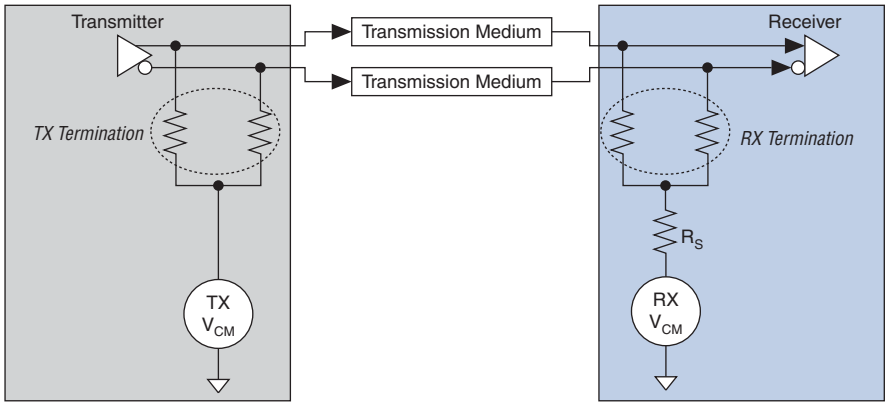


Figure 1–27 shows the DC-coupled link connection from an LVDS transmitter to an Arria II GX and GZ receiver.

Figure 1–27. LVDS Transmitter to Arria II GX and GZ Receiver (PCML) DC-Coupled Link

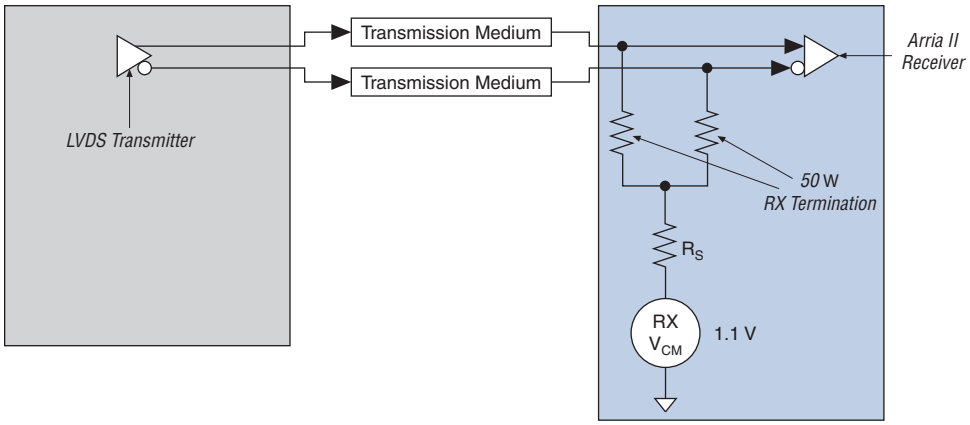


Table 1–7 lists the settings for DC-coupled links between Altera devices. You must comply with the data rates supported by the Arria II GX and GZ receivers.

Table 1–7. DC-Coupled Settings for Arria II Devices *(Note 1)* (Part 1 of 2)

Link	Transceiver Settings	Receiver Settings
	TX VCM (V)	RX VCM (V)
Arria II PCML transmitter to Arria II PCML receiver	0.65	0.82
Stratix II GX PCML transmitter to Arria II PCML receiver	0.6, 0.7	0.82
Arria II PCML transmitter to Stratix II GX PCML receiver	0.65	0.85
Arria II PCML transmitter to Stratix IV GX PCML receiver	0.65	0.82
Stratix IV GX PCML transmitter to Arria II PCML receiver	0.65	0.85

Table 1–7. DC-Coupled Settings for Arria II Devices *(Note 1)* (Part 2 of 2)

Link	Transceiver Settings	Receiver Settings
	TX VCM (V)	RX VCM (V)
LVDS transmitter to Arria II GX and GZ receiver	—	1.1

Note to Table 1–7:

(1) The differential OCT setting for Arria II GX and GZ transmitters and receivers is 85, 100, 120, and 150 Ω for Arria II GZ devices or 100 Ω for Arria II GX devices except for the LVDS transmitter settings, which do not have OCT set on the transmitter (as shown in Figure 1–27).

Programmable Equalization, DC Gain, and Offset Cancellation

Each Arria II GX and GZ receiver input 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 and GZ equalization circuitry supports equalization settings that provide up to 7 dB (Arria II GX) and 16 dB (Arria II GZ) of high-frequency boost.

The Arria II GX and GZ receiver input buffer also supports programmable DC gain circuitry. Unlike equalization circuitry, DC gain circuitry provides equal boost to the incoming signal across the frequency spectrum.

You can select the proper equalization and DC gain settings in the ALTGX MegaWizard Plug-In Manager. The receiver buffer supports DC gain settings of 0 dB, 3 dB, and 6 dB for Arria II GX devices and up to 12 dB for Arria II GZ devices.

This offset cancellation block cancels offset voltages between the positive and negative differential signals within the equalizer stages in order to reduce the minimum V_{ID} requirement. The receiver input buffer and receiver CDR require offset cancellation.

The offset cancellation for the receiver channels option is automatically enabled in both the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers for **Receiver**, **Transmitter**, and **Receiver only** configurations. When offset cancellation is automatically enabled, you must instantiate the dynamic reconfiguration controller to connect the reconfiguration ports created by the ALTGX MegaWizard Plug-In Manager.

For more information about offset cancellation, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II Devices*. For the transceiver reset sequence with the offset cancellation feature, refer to the *Reset Control and Power Down in Arria II Devices* chapter.

Signal Threshold Detection Circuitry

Signal threshold detection circuitry has a hysteresis response that filters out any high-frequency ringing caused by ISI effects 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 `rx_signaldetect` signal is held low.

In PCIe mode, you can enable the optional signal threshold detection circuitry by leaving the **Force signal detection** option unchecked in the ALTGX MegaWizard Plug-In Manager.

The appropriate signal detect threshold level that complies with the PCIe compliance parameter VRX-IDLE-DETDIFFp-p is pending characterization.

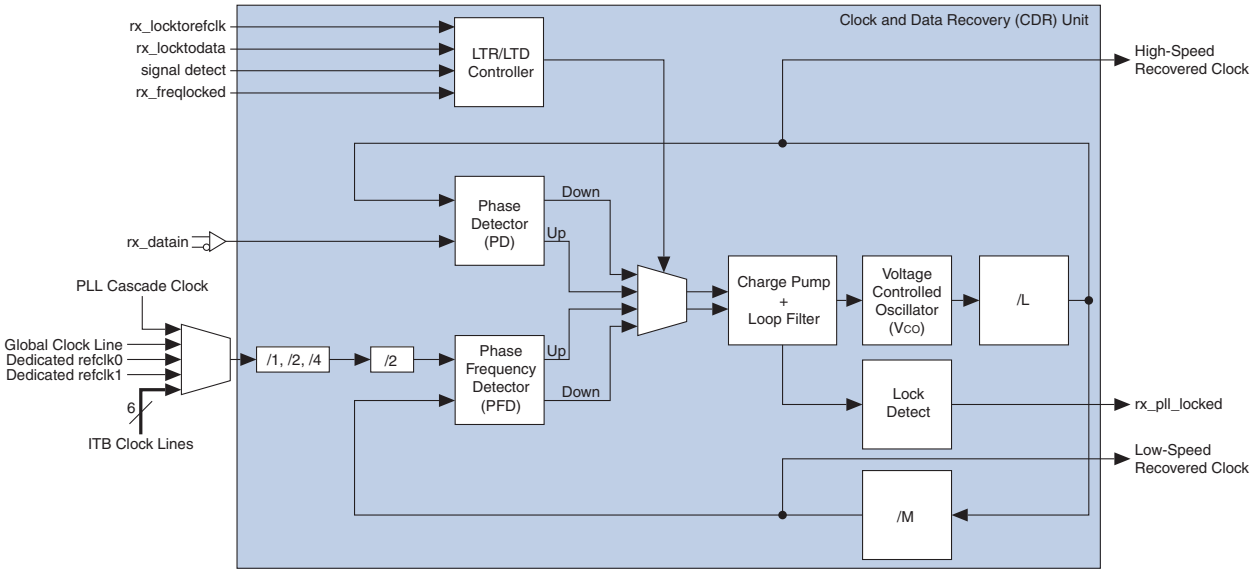
If you enable the **Force signal detection** option in the ALTGX MegaWizard Plug-In Manager, the `rx_signaldetect` signal is always asserted high, irrespective of the signal level on the receiver input buffer. When enabled, this option senses whether the signal level present at the receiver input buffer 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 `rx_signaldetect` signal is also used by the 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 lock-to-data (LTD) to lock-to-reference (LTR) lock mode.

CDR

Each Arria II GX and GZ receiver channel has an independent CDR unit to recover the clock from the incoming serial data stream. High-speed and low-speed recovered clocks are used to clock the receiver PMA and PCS blocks. Figure 1–28 shows the CDR block.

Figure 1–28. CDR Block



The CDR operates in two modes:

- LTR mode—The PFD in the CDR tracks the receiver input reference clock (`rx_cruclk`) and controls the charge pump that tunes the VCO in the CDR. An active high `rx_pll_locked` status signal is asserted to indicate that the CDR has locked to phase and frequency of the receiver input reference clock. In this mode, the phase detector is inactive.

  Depending on the data rate and the selected input reference clock frequency, the Quartus II software automatically selects the appropriate divider values such that the CDR output clock frequency is half the data rate. This includes the pre-divider before the PFD.

- LTD mode—The phase detector in the CDR tracks the incoming serial data at the receiver input buffer to keep the recovered clock phase-matched to the data. Depending on the phase difference between the incoming data and the CDR output clock, the phase detector controls the CDR charge pump that tunes the VCO.

  In this mode, the PFD and the /M divider block are inactive. In addition, the `rx_pll_locked` signal toggles randomly and has no significance in LTD mode.

  The CDR must be in LTD mode to recover the clock from the incoming serial data during normal operation. The actual LTD 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 asserts the `rx_freqlocked` signal and produces a stable recovered clock.

  For more information about receiver reset recommendations, refer to the *Reset Control and Power Down* chapter.

The CDR must be kept in LTR mode until it locks to the input reference clock after the power-up and reset cycle. When locked to the input reference clock, the CDR output clock is trained to the configured data rate and can switch to LTD mode to recover the clock from the incoming data. You can use the optional input ports (`rx_locktorefclk` and `rx_locktodata`) to control the LTR or LTD mode manually or let the lock happen automatically.

Table 1–8 lists the relationship between the optional input ports and the LTR/LTD controller lock mode.

Table 1–8. Optional Input Ports and LTR/LTD Controller Lock Mode for Arria II Devices *(Note 1)*

rx_locktorefclk	rx_locktodata	LTR/LTD Controller Lock Mode
1	0	Manual – LTR Mode
X	1	Manual – LTD Mode
0	0	Automatic Lock Mode

Note to Table 1–8:

(1) If you do not instantiate the optional `rx_locktorefclk` and `rx_locktodata` signals in the ALTGX megafunction, 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 relies on the PPM detector and the phase relationship detector to set the CDR in LTR or LTD mode. Initially, the CDR is set to LTR mode. After the CDR locks to the input reference clock, the LTR/LTD controller automatically sets it to LTD mode and asserts the `rx_freqlocked` signal 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 is locked)
- CDR output clock and input reference clock are phase matched within approximately 0.08 UI (phase is locked)

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. The LTR/LTD controller switches the CDR from LTD to LTR mode and de-asserts the `rx_freqlocked` signal 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

Manual Lock Mode

You may want to use manual lock mode if your application requires faster CDR lock time. 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, as shown in Table 1–8 on page 1–27.

When the `rx_locktorefclk` signal is asserted high, the `rx_freqlocked` signal does not have 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.



You must comply with the different transceiver reset sequences depending on the CDR lock mode.

Deserializer

The deserializer block latches the serial input data from the receiver input buffer with the high-speed serial recovery clock, deserializes it using the low-speed parallel recovery clock, and drives the deserialized data to the receiver PCS channel.

The deserializer supports 8-, 10-, 16-, and 20-bit deserialization factors. Figure 1–29 shows the deserializer operation with a 10-bit deserialization factor.

Figure 1–29. 10-Bit Deserializer Operation

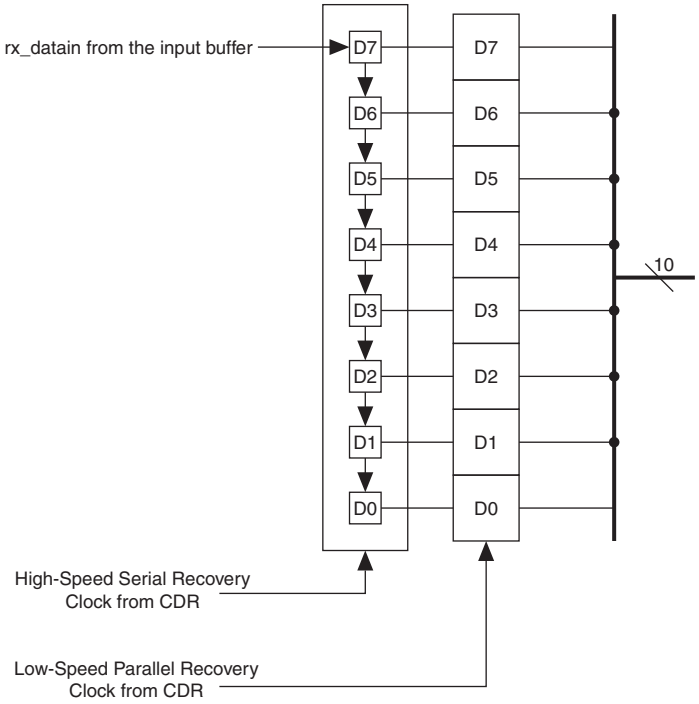
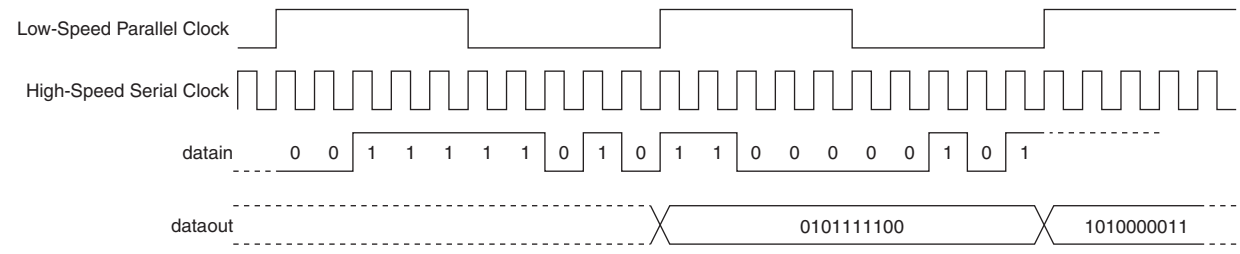


Figure 1–30 shows the serial bit order of the deserializer block input and the parallel data output of the deserializer block with a 10-bit deserialization factor. The serial stream (10'b0101111100) is deserialized to a value 10'h17C. The serial data is assumed to have received the LSB first.

Figure 1–30. 10-Bit Deserializer Bit Order



Receiver PCS

This section describes the receiver PCS modules, which consist of the word aligner, deskew FIFO, rate-match FIFO, 8B/10B decoder, byte deserializer, byte ordering, and RX phase compensation FIFO.

The `rx_digitalreset` signal resets all modules in the receiver PCS block.

For more information about this signal, refer to the *Reset Control and Power Down in Arria II Devices* chapter.

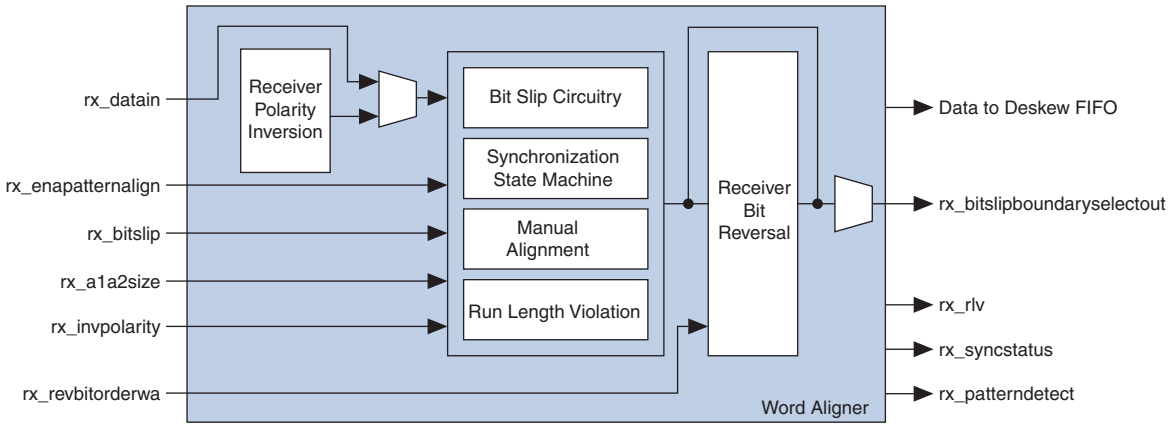
Word Aligner

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 such as GbE, PCIe, SRIO, SONET/SDH, and XAUI specify a standard word alignment pattern. The Arria II GX and GZ transceiver architecture allows you to select a custom word alignment pattern specific to your implementation if you use proprietary protocols.

Figure 1–31 shows the word aligner block diagram.

Figure 1–31. Word Aligner



In addition to restoring word boundaries, the word aligner also implements the following features:

- Programmable run length violation detection—This feature is available in all functional modes. It detects consecutive 1s or 0s in the data stream. If a preset maximum number of consecutive 1s or 0s is detected, the run length violation status signal (`rx_rlv`) is asserted. This signal has lower latency when compared with the parallel data on the `rx_dataout` port.

  The `rx_rlv` signal in each channel is clocked by its parallel recovered clock and is asserted for a minimum of two recovered clock cycles to ensure that the FPGA fabric clock can latch the `rx_rlv` signal reliably because the FPGA fabric clock might have phase differences, PPM differences (in asynchronous systems), or both, with the recovered clock.

Table 1–9 lists the detection capabilities of the run-length violation circuit.

Table 1–9. Detection Capabilities of the Run-Length Violation Circuit for Arria II Devices

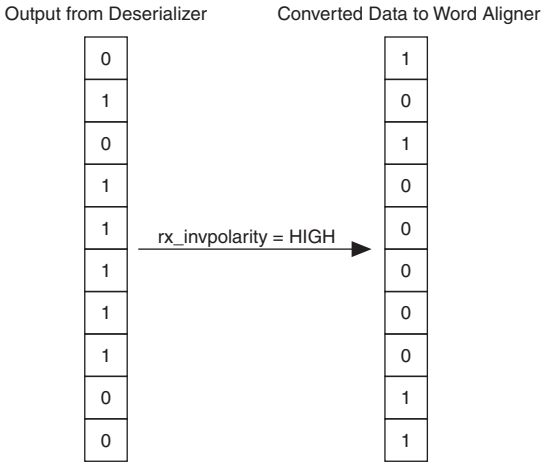
PMA-PCS Interface Width	Run Length Violation Detector Range	
	Minimum	Maximum
8 bit	4	128
10 bit	5	160
16 bit	8	512
20 bit	10	640

- Receiver polarity inversion—This feature is available in all functional modes except PCIe. It offers an optional `rx_invpolarity` port to dynamically enable the receiver polarity inversion feature as a workaround to board re-spin or a major update to the FPGA fabric design when the positive and negative signals of a serial differential link are accidentally swapped during board layout.

  A high value on the `rx_invpolarity` port inverts the polarity of every bit of the input data word to the word aligner in the receiver 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 `rx_invpolarity` signal is dynamic and might cause initial disparity errors in an 8B/10B encoded link. The downstream system must be able to tolerate these disparity errors.

  Figure 1–32 shows an example result with the `rx_invpolarity` feature in a 10-bit wide datapath configuration.

Figure 1–32. 10-Bit Receiver Polarity Inversion



This generic receiver polarity inversion feature is different from the PCIe 8B/10B polarity inversion feature because it inverts the polarity of the data bits at the input of the word aligner, whereas the PCIe 8B/10B polarity inversion feature inverts the polarity of the data bits at the input of the 8B/10B decoder.

- Receiver bit reversal—This feature is only available in Basic mode. By default, the Arria II GX and GZ receiver assumes LSB-to-MSB transmission. If the transmission order is MSB-to-LSB, 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 by flipping the parallel data so that the `rx_dataout` port contains the correct bit-ordered data.

  This feature is available through the `rx_revbitordwa` port in Basic mode only with the word aligner configured in bit-slip mode. When you drive the `rx_revbitordwa` signal high in this configuration, 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.

  Figure 1–33 shows the receiver bit reversal feature in Basic mode with 10-bit wide datapath configurations.

Figure 1–33. 10-Bit Receiver Bit Reversal in Basic Mode with Word Aligner in Bit-Slip Mode

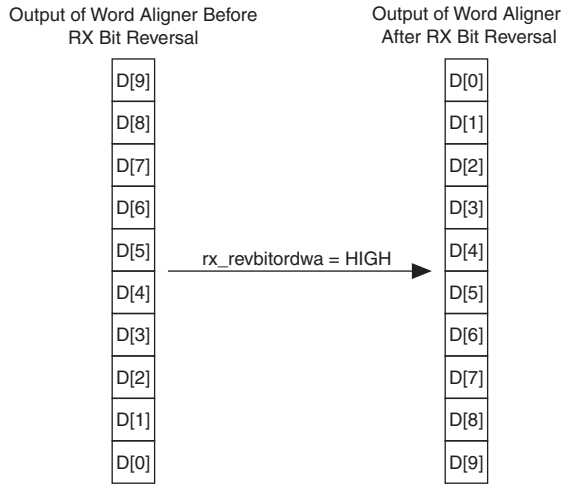


Table 1–10 lists the three modes of the word aligner and their supported data width, functional mode, and allowed alignment pattern length for Arria II devices.

Table 1–10. Word Aligner Modes for Arria II Devices

Word Aligner Mode	Data Width Supported (bits)	Functional Mode Supported	Allowed Word Alignment Pattern Length
Manual Alignment	8	Basic, OC-12, and OC-48	16 bits
	10	Basic and Deterministic Latency	7 or 10 bits
	16	Basic and Deterministic Latency	8, 16, or 32 bits
	20	Basic and Deterministic Latency	7, 10, or 20 bits
Bit-Slip	8	Basic	16 bits
	10	Basic and SDI	7 or 10 bits for Basic N/A for SDI
	16	Basic and Deterministic Latency	8, 16, or 32 bits
	20	Basic and Deterministic Latency	7, 10, or 20 bits
Automatic Synchronization State Machine	10	Basic, GbE, PCIe, Serial RapidIO, and XAUI	10 bits for all functional modes 7 bits or 10 bits for Basic

Manual Alignment Mode

This mode is automatically used in SONET/SDH functional mode. In Basic mode, you can configure the word aligner in manual alignment mode by selecting the **Use manual word alignment mode** option in the word aligner tab of the ALTGX MegaWizard Plug-In Manager.

In manual alignment mode, the input signal (`rx_enapatternalign`) controls the word aligner. The 8-bit word aligner is edge-sensitive to the `rx_enapatternalign` signal; the 10-bit word aligner is level-sensitive to this signal.



If the word alignment pattern is 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.

With 8-bit width data, a rising edge on the `rx_enapatternalign` signal after de-assertion of the `rx_digitalreset` 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 mode, the word aligner looks for the 16-bit word alignment pattern programmed in the ALTGX MegaWizard Plug-In Manager.

With 10-bit width data, the word aligner looks for the programmed 7-bit or 10-bit word alignment pattern in the received data stream, if the `rx_enapatternalign` signal is held high. 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.

The `rx_syncstatus` and `rx_patterndetect` status signals have the same latency as the datapath and are forwarded to the FPGA fabric to indicate word aligner status. On receiving the first word alignment pattern after the assertion of the `rx_enapatternalign` signal, both the `rx_syncstatus` and `rx_patterndetect` signals are driven high for one parallel clock cycle synchronous to the most significant byte (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 the word aligner to re-align to the new word boundary only if there is a rising edge in the `rx_enapatternalign` signal (in the 8-bit word aligner) or if the `rx_enapatternalign` signal is held high (in 10-bit word aligner). The word aligner asserts the `rx_syncstatus` and `rx_patterndetect` signals for one parallel clock cycle whenever it re-aligns to the new word boundary.

Figure 1–34 shows word aligner behavior in SONET/SDH OC-12 functional mode. The least significant byte (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 the initial word alignment, the 16-bit word alignment pattern (16'h28F6) is again received across the word boundary in clock cycles m, m + 1, and m + 2. The word aligner does not re-align to the new word boundary for lack of a preceding rising edge on the rx_enapatternalign signal. If there is a rising edge on the rx_enapatternalign signal before the word alignment pattern occurs across these clock cycles, the word aligner re-aligns to the new word boundary, causing both the rx_syncstatus and rx_patterndetect signals to go high for one parallel clock cycle.

Figure 1–34. Manual Alignment Mode in 8-Bit PMA-PCS Interface Mode

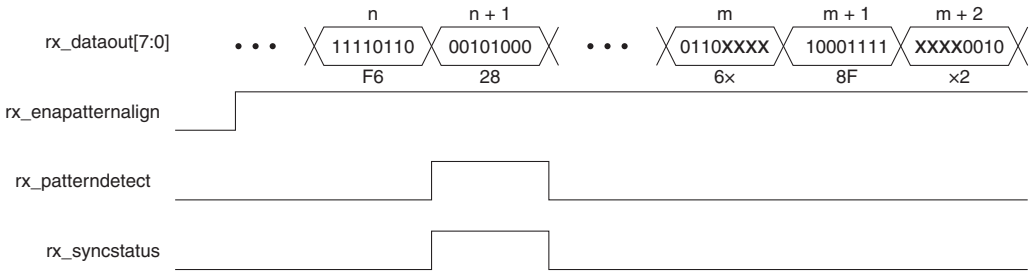
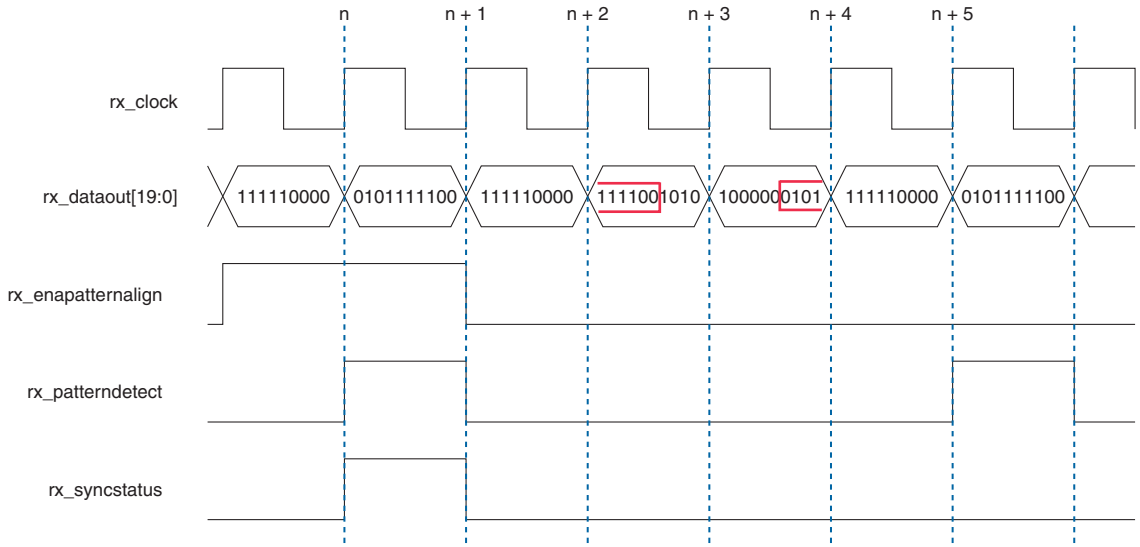


Figure 1–35 shows the manual alignment mode word aligner operation in 10-bit PMA-PCS interface mode. In this example, a /K28.5/ (10'b0101111100) is specified as the word alignment pattern. The word aligner aligns to the /K28.5/ alignment pattern in cycle n because the rx_enapatternalign signal is asserted 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 word aligner does not align to this new word boundary because the rx_enapatternalign signal is held low. The /K28.5/ word alignment pattern is detected again in the current word boundary during cycle n + 5, causing the rx_patterndetect signal to go high for one parallel clock cycle.

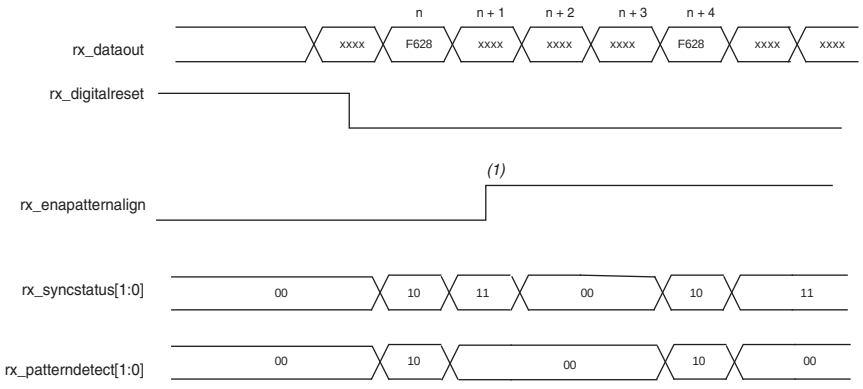
Figure 1–35. Word Aligner in 10-Bit PMA-PCS Manual Alignment Mode



With a 16- or 20-bit width data, the word aligner starts looking for the programmed 8-bit, 16-bit, or 32-bit word alignment pattern in the received data stream as soon as `rx_digitalreset` is de-asserted low. It aligns to the first word alignment pattern received regardless of the logic level driven on the `rx_enapatternalign` signal. Any word alignment pattern received thereafter in a different word boundary does not cause the word aligner to re-align to this new word boundary. After the initial word alignment, following de-assertion of the `rx_digitalreset` signal, if a word re-alignment is required, you must use the `rx_enapatternalign` signal.

Figure 1–36 shows the manual alignment mode word aligner operation in 16-bit PMA-PCS interface mode.

Figure 1–36. Manual Alignment Mode Word Aligner in 16-Bit PMA-PCS Interface Modes



Note to Figure 1–36:

(1) The `rx_syncstatus` de-assertion latency after the assertion of `rx_enapatternalign` depends on your RX PCS implementation.

Bit-Slip Mode

In Basic, deterministic latency, and SDI functional modes, 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.

Bit slip in the 10-bit wide word aligner allows 7-bit and 10-bit word alignment patterns, whereas bit-slip in the 8-bit wide word aligner allows only a 16-bit word alignment pattern. Other than this, the bit-slip operation is the same between the 8-bit and 10-bit word aligner.

The `rx_bitslip` signal controls the word aligner operation 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. The `rx_patterndetect` signal 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.

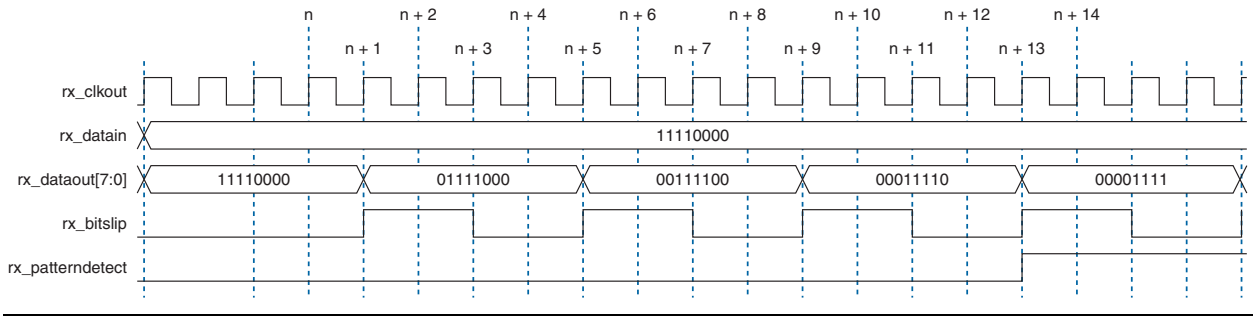


You can implement a bit-slip controller in the FPGA fabric that monitors either the `rx_dataout` signal and/or the `rx_patterndetect` signal and controls the `rx_bitslip` signal to achieve word alignment.

Figure 1–37 shows an example of the word aligner configured in bit-slip mode, which has the following events:

- 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 + 1 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 `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 1–37. Example of Word Aligner Configured in Bit-Slip Mode



Bit-Slip Mode Word Aligner with 16-Bit PMA-PCS Interface Modes

In some Basic double-width configurations with 16-bit PMA-PCS interface, 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 double-width with 16-bit PMA-PCS interface is similar to the word aligner operation in Basic single-width mode with 8-bit PMA-PCS interface. The only difference is that the bit-slip word aligner in 16-bit PMA-PCS interface modes allows 8-bit and 16-bit word alignment patterns, whereas the bit-slip word aligner in 8-bit PMA-PCS interface modes allows only a 16-bit word alignment pattern.

Word Aligner in Double-Width Mode with 20-Bit PMA-PCS Interface Modes

A 20-bit PMA-PCS interface is supported only in Basic double-width mode.

Table 1–11 shows the word aligner configurations allowed in functional modes with a 20-bit PMA-PCS interface.

Table 1–11. Word Aligner in 20-Bit PMA-PCS Interface Modes for Arria II Devices

Functional Mode	Allowed Word Aligner Configurations	Allowed Word Alignment Pattern Length
Basic double-width	Manual alignment, Bit-slip	7 bits, 10 bits, 20 bits

Manual Alignment Mode Word Aligner with 20-Bit PMA-PCS Interface Modes

The word aligner operation in Basic double-width mode with 20-bit PMA-PCS interface is similar to the word aligner operation in Basic double-width mode with a 16-bit PMA-PCS interface. The only difference is that the manual alignment mode word aligner in 20-bit PMA-PCS interface modes allows 7-, 10-, and 20-bit word alignment patterns, whereas the manual alignment mode word aligner in 16-bit PMA-PCS interface modes allows only 8-, 16-, and 32-bit word alignment patterns.

Bit-Slip Mode Word Aligner with 20-Bit PMA-PCS Interface Modes

In some Basic single-width configurations with a 20-bit PMA-PCS interface, 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 double-width with 20-bit PMA-PCS interface is similar to the word aligner operation in Basic single-width mode with an 8-bit PMA-PCS interface. The difference is that the bit-slip word aligner in 20-bit PMA-PCS interface modes allows only 7-, 10-, and 20-bit word alignment patterns, whereas the bit-slip word aligner in 8-bit PMA-PCS interface modes allows only a 16-bit word alignment pattern.

Table 1–12 summarizes the word aligner options available in Basic single-width and double-width modes.

Table 1–12. Word Aligner Options Available in Basic Single-Width and Double-Width Modes for Arria II Devices *(Note 1)* (Part 1 of 2)

Functional Mode	PMA-PCS Interface Width	Word Alignment Mode	Word Alignment Pattern Length	rx_enapatternalign Sensitivity	rx_syncstatus Behavior	rx_patterndetect Behavior
Basic Single-Width	8-bit	Manual Alignment	16-bit	Rising Edge Sensitive	Asserted high for one parallel clock cycle when the word aligner aligns to a new word boundary.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Bit-Slip	16-bit	—	—	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
	10-bit	Manual Alignment	7- and 10-bit	Level Sensitive	Asserted high for one parallel clock cycle when the word aligner aligns to a new word boundary.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Bit-Slip	7- and 10-bit	—	—	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Automatic Synchronization State Machine	7- and 10-bit	—	Stays high as long as the synchronization conditions are satisfied.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.

Table 1–12. Word Aligner Options Available in Basic Single-Width and Double-Width Modes for Arria II Devices *(Note 1)* (Part 2 of 2)

Functional Mode	PMA-PCS Interface Width	Word Alignment Mode	Word Alignment Pattern Length	rx_enapatternalign Sensitivity	rx_syncstatus Behavior	rx_patterndetect Behavior
Basic Double-Width	16-bit	Manual Alignment	8-, 16-, and 32-bit	Rising Edge Sensitive	Stays high after the word aligner aligns to the word alignment pattern. Goes low on receiving a rising edge on `rx_enapatternalign` until a new word alignment pattern is received.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Bit-Slip	8-, 16-, and 32-bit	—	—	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
	20-bit	Manual Alignment	7-, 10-, and 20-bit	Rising Edge Sensitive	Stays high after the word aligner aligns to the word alignment pattern. Goes low on receiving a rising edge on `rx_enapatternalign` until a new word alignment pattern is received.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Bit-Slip	7-, 10-, and 20-bit	—	—	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.

Note to Table 1–12:

(1) For more information about word aligner operation, refer to "Word Aligner" on page 1–30.

Automatic Synchronization State Machine Mode

You must use this mode with 8B/10B encoded data if the input data to the word aligner is 10 bits.

Protocols such as PCIe, XAUI, Gigabit Ethernet, and SRIO 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 to acquire synchronization and a specific number of erroneous code groups that it must receive to fall out of synchronization.

The Quartus II software configures the word aligner in automatic synchronization state machine mode for PCIe, XAUI, Gigabit Ethernet, and SRIO functional modes. It automatically selects the word alignment pattern length and the word alignment pattern as specified by each protocol. In each of these functional modes, the protocol-compliant synchronization state machine is implemented in the word aligner.

By using Basic functional mode with the 10-bit PMA-PCS interface, you can configure the word aligner in automatic synchronization state machine mode by selecting the **Use the automatic synchronization state machine** option in the ALTGX MegaWizard Plug-In Manager. Basic mode also allows you to program a custom 7-bit or 10-bit word alignment pattern that the word aligner uses for synchronization.

Table 1–13 lists the synchronization state machine parameters that the Quartus II software allows in supported functional modes. The synchronization state machine parameters are fixed for PCIe, XAUI, GbE, and SRIO modes as specified by the respective protocol. You can program these parameters as suited to your proprietary protocol implementation for Basic mode.

Table 1–13. Synchronization State Machine Parameters for Arria II Devices

Parameter	PCIe	XAUI	GbE	SRIO	Basic 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

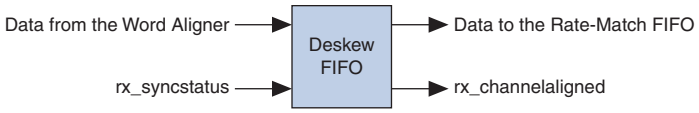
After de-assertion of the `rx_digitalreset` signal in automatic synchronization state machine mode, the word aligner starts looking for the word alignment pattern or synchronization code groups in the received data stream. When the programmed number of valid synchronization code groups or ordered sets 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 the `rx_syncstatus` signal is driven low. The word aligner indicates loss of synchronization (`rx_syncstatus` signal remains low) until the programmed number of valid synchronization code groups are received again.

Deskew FIFO

Use this module, only available in XAUI mode, to align all four channels to meet the XAUI maximum skew of 40 UI (12.8 ns) as seen at the receiver of the four lanes. The deskew operation in XAUI functional mode is compliant to the PCS deskew state machine diagram specified in 8 of the IEEE P802.3ae.

The deskew circuitry consists of a 16-word deep deskew FIFO in each of the four channels and control logic in the `CMU0` channel of the transceiver block that controls the deskew FIFO write and read operations in each channel. Figure 1–38 shows the deskew FIFO block diagram.

Figure 1–38. Deskew FIFO ***(Note 1)***



Note to Figure 1–38:

(1) For more information about the deskew FIFO operation, refer to "XAUI" on page 1–79.

Rate-Match FIFO

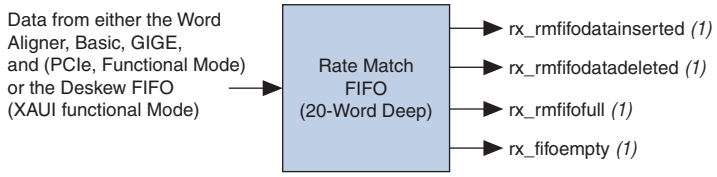
In asynchronous systems, you can clock the upstream transmitter and local receiver with independent reference clocks. Frequency differences in the order of a few hundred PPM can corrupt the data when latching from the recovered clock domain (the same clock domain as the upstream transmitter reference 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.

The 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. This module is optional in Basic functional mode, but is mandatory and cannot be bypassed in GbE, PCIe, and XAUI functional modes.

Figure 1–39 shows the rate-match FIFO block diagram.

Figure 1–39. Rate Match FIFO



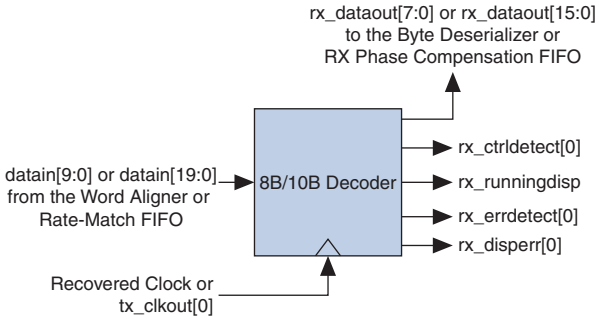
Note to Figure 1–39:

(1) These signals are not available in PCIe functional mode because the rate match FIFO status is encoded in the `pipestatus[2:0]` signal.

8B/10B Decoder

Figure 1–40 shows the block diagram of the 8B/10B decoder.

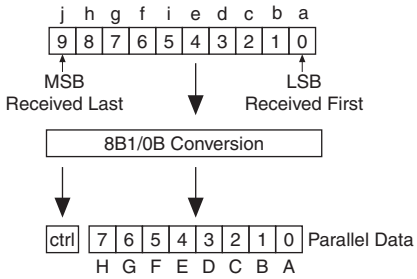
Figure 1–40. 8B/10B Decoder



The 8B/10B decoder is compliant to Clause 36 in the IEEE802.3 specification, decoding the 10-bit data input into an 8-bit data and a 1-bit control identifier. This module is required in GbE, PCIe, SRIO, and XAUI functional modes, but is optional in Basic functional mode.

Figure 1–41 shows a 10-bit code group decoded into an 8-bit data and a 1-bit control identifier by the 8B/10B decoder.

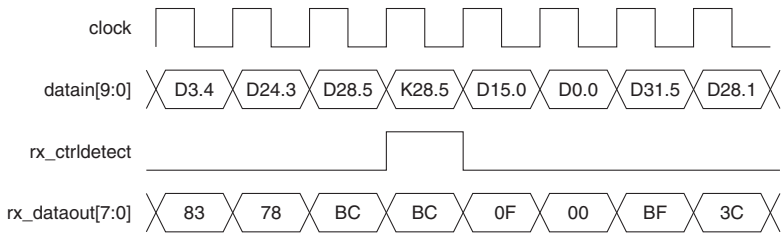
Figure 1–41. 8B/10B Decoder Operation



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 1–42 shows 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. The `rx_ctrldetect` signal 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 1–42. 8B/10B Decoder in Control Code Group Detection



Byte Deserializer

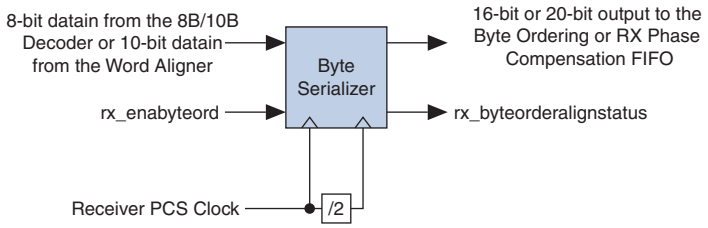
Some serial data rates violate the maximum frequency for the FPGA fabric-to-transceiver interface. In such configurations, the byte deserializer is required to reduce the FPGA fabric-to-transceiver interface frequency to half while doubling the parallel data width. This module is optional in configurations that do not exceed the FPGA fabric-to-transceiver interface clock upper frequency limit.

For example, at a 3.2 Gbps data rate with a deserialization factor of 10, the receiver PCS datapath runs at 320 MHz. The byte serializer converts the 10-bit parallel received data at 320 MHz into 20-bit parallel data at 160 MHz before forwarding it to the FPGA fabric.

In Arria II GX devices, you cannot enable the byte deserializer in double-width mode. However, in Arria II GZ devices, you can enable both double-width mode and the byte deserializer to achieve a 32- or 40-bit PCS-FPGA interface.

Figure 1–43 shows the block diagram of the byte deserializer with 8-bit or 10-bit PMA-to-PCS interface.

Figure 1–43. Byte Deserializer

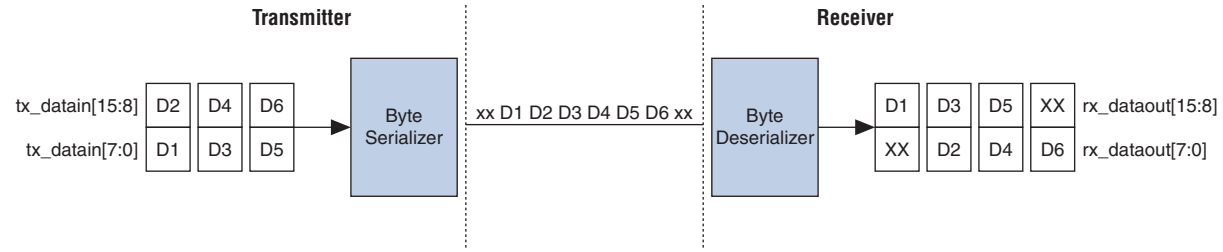


Byte Ordering

Depending on when the receiver PCS logic comes out of reset, byte ordering at the output of the byte deserializer 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 1–44 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 1–44. MSByte and LSByte of the Two-Byte Transmitter Data Straddled Across Two Word Boundaries



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(s) position of the parallel transmitter data. If the byte ordering block finds the programmed byte ordering pattern in the MSbyte(s) 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(s) position, thereby restoring proper byte ordering.

The byte ordering block is available in the following functional modes:

- SONET/SDH OC-48—The Quartus II software automatically configures the byte ordering pattern and byte ordering PAD pattern in this mode.
- SONET/SDH OC-96—For Arria II GZ devices only.
- Basic mode with 16-bit FPGA fabric-to-transceiver interface, no 8B/10B decoder (8-bit PMA-PCS interface) and word aligner in manual alignment mode—You can program a custom 8-bit byte ordering pattern and 8-bit byte ordering PAD pattern in the ALTGX MegaWizard Plug-In Manager.
- Basic mode with 16-bit FPGA fabric-to-transceiver interface, 8B/10B decoder, and word aligner in automatic synchronization state machine mode—You can program a custom 9-bit byte ordering pattern and 9-bit byte ordering PAD pattern in the ALTGX MegaWizard Plug-In Manager. 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 modules have two different modes of operation when you select the rx_syncstatus 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:

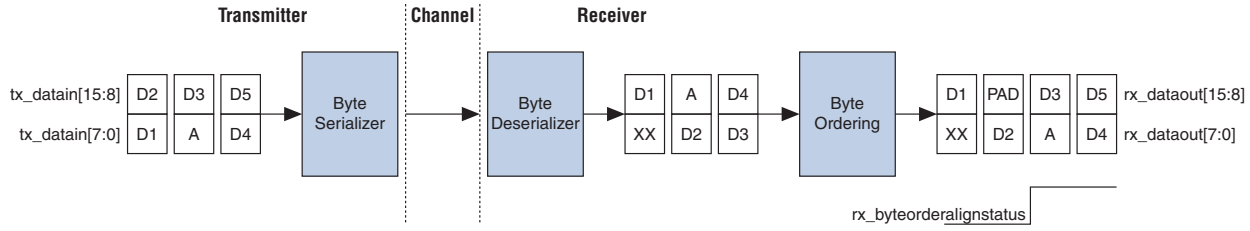
- Word-Alignment-Based Byte Ordering—In this mode, the byte ordering block starts looking for the byte ordering pattern in the byte-deserialized data every time it sees a rising edge on the rx_syncstatus signal.

  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 after a rising edge on the rx_syncstatus signal.

  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 ordered and does not insert any PAD pattern. In either case, the byte ordering block asserts the rx_byteorderalignstatus signal.

  Figure 1–45 shows an example of the byte ordering operation where 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 1–45. Example of Byte Ordering Operation



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.

- 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 as a trigger to the byte ordering operation. A rising edge on the rx_enabyteord port triggers the byte ordering block. 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 after a rising edge on the rx_enabyteord signal. 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.

RX Phase Compensation FIFO

The RX phase compensation FIFO in each channel ensures reliable transfer of data and status signals between the receiver channel and the FPGA fabric. The RX 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).

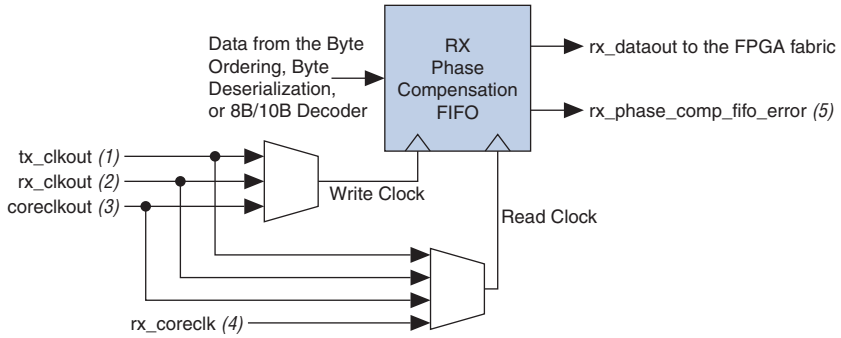
Table 1–14 lists the RX phase compensation FIFO modes. The Quartus II software automatically sets the mode that applies to a particular functional mode.

Table 1–14. RX Phase Compensation FIFO Modes for Arria II Devices

Mode	FIFO Depth	Latency Through FIFO	Applicable Functional Modes
Low Latency	4-words deep	2-to-3 parallel clock cycles	All functional modes except PCIe and Deterministic Latency
High-Latency	8-words deep	4-to-5 parallel clock cycles	PCIe
Register	—	1	Deterministic Latency

Figure 1–46 shows the RX phase compensation FIFO block diagram. The write clock and read clock to the FIFO are at half-rate if the byte serializer is used in the configuration.

Figure 1–46. RX Phase Compensation FIFO Block Diagram



Notes to Figure 1–46:

(1) The tx_clkout clock is from the transmitter channel clock divider and is used in non-bonded configurations with the rate match FIFO.

(2) The rx_clkout clock is from the receiver channel CDR and is used in non-bonded configurations that does not use the rate match FIFO.

(3) The coreclkout clock is from the CMU0 block of the associated transceiver block or the master transceiver block for ×4 bonded or ×8 bonded channel configurations, respectively.

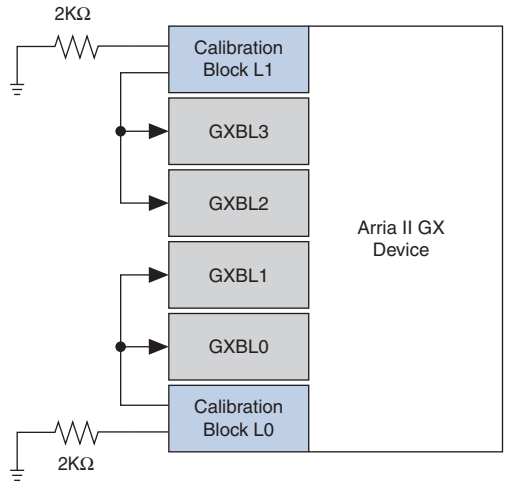
(4) Use this clock for the FIFO read clock if you instantiate the rx_coreclk port in the ALTGX MegaWizard Plug-In Manager, regardless of the channel configurations. Otherwise, use the same clock used for the write clock for the read clock.

(5) The rx_phase_comp_fifo_error is optional and available in all functional modes. This signal is asserted high to indicate an overflow or underflow condition.

Calibration Block

This block calibrates the OCT resistors and the analog portions of the transceiver blocks to ensure that the functionality is independent of process, voltage, or temperature (PVT) variations. Figure 1–47 shows the location of the calibration block and how it is connected to the transceiver blocks in Arria II GX devices.

Figure 1–47. Arria II GX Transceiver Calibration Blocks Location and Connection *(Note 1)*, *(2)*, *(3)*



Notes to Figure 1–47:

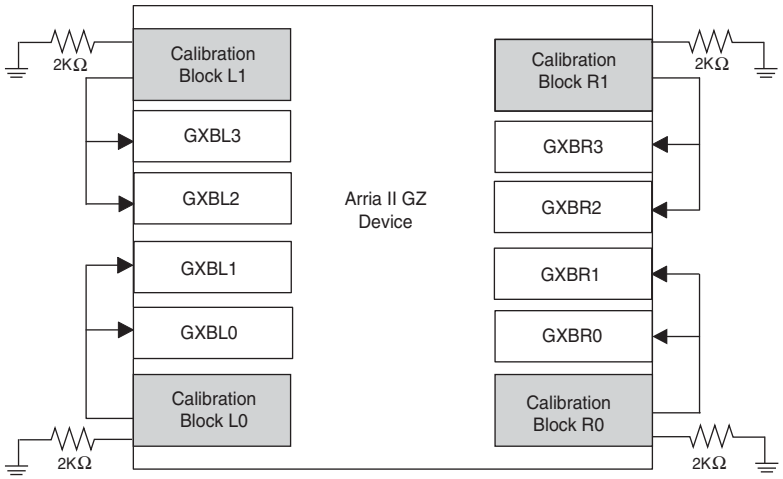
(1) You must connect a separate 2 kΩ (tolerance max ± 1%) external resistor on each RREF pin in the Arria II device to GND. To ensure proper operation of the calibration block, the RREF resistor connection in the board must be free from external noise.

(2) The Quartus II software automatically selects the appropriate calibration block based on the pin assignments of the transceiver tx_dataout and rx_datain pins.

(3) For RREF pin information, refer to the *Device Pin Connection Guidelines*.

Figure 1–48 shows the location of the calibration block and how it is connected to the transceiver blocks in Arria II GZ devices.

Figure 1–48. Arria II GZ Transceiver Calibration Blocks Location and Connection *(Note 1)*, *(2)*, *(3)*



Notes to Figure 1–48:

(1) You must connect a separate 2 kΩ (tolerance max ± 1%) external resistor on each RREF pin in the Arria II device to GND. To ensure proper operation of the calibration block, the RREF resistor connection in the board must be free from external noise.

(2) The Quartus II software automatically selects the appropriate calibration block based on the pin assignments of the transceiver tx_dataout and rx_datain pins.

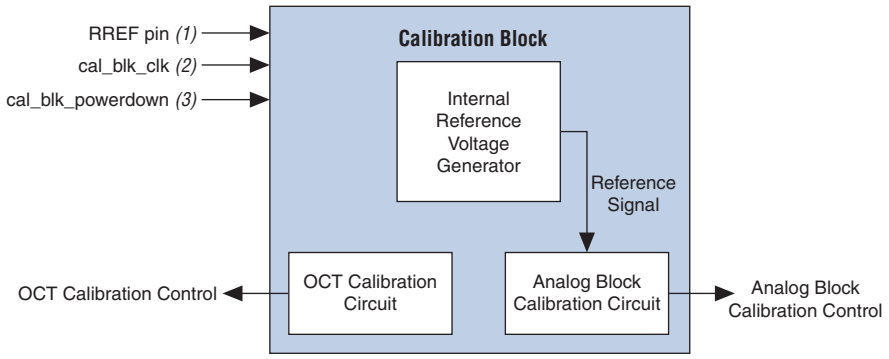
(3) For RREF pin information, refer to the *Device Pin Connection Guidelines*.

The calibration block internally generates a constant internal reference voltage, independent of PVT variations and uses this voltage and the external reference resistor on 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.

Figure 1–49 shows the calibration block diagram. You can instantiate the cal_blk_clk and cal_blk_powerdown ports in the ALTGX MegaWizard Plug-In Manager to control the calibration block. 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.

Figure 1–49. Input Signals to the Calibration Blocks



Notes to Figure 1–49:

(1) You must connect a separate 2 kΩ (tolerance max ± 1%) external resistor on each RREF pin in the Arria II GX and GZ device to GND. To ensure proper operation of the calibration block, the RREF resistor connection in the board must be free from external noise.

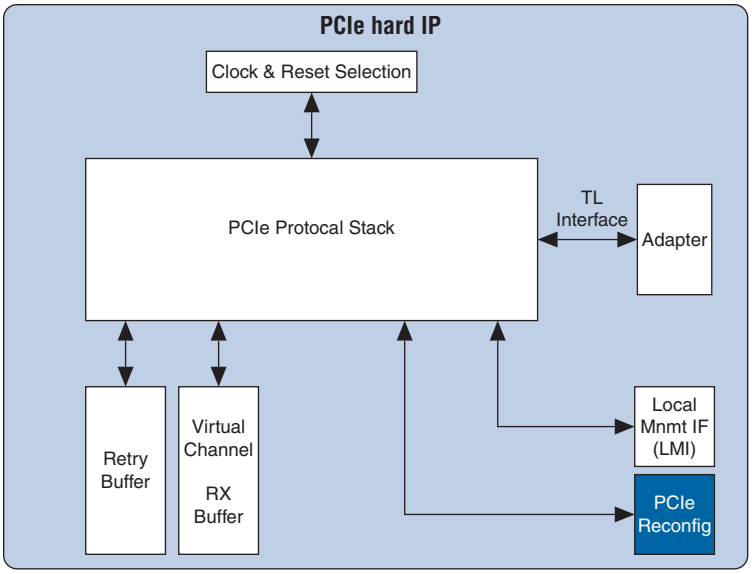
(2) This is the input clock to the calibration block. You can use dedicated clock routes such as the global or regional clock; however, if you do not have a suitable input reference clock or dedicated clock routing resources available, use divide-down logic from the FPGA fabric to generate a slow clock, local clocking routing, or both.

(3) Drive this signal of all ALTGX instances that are associated with the same calibration block from the same input or logic. During calibration, the link may experience higher bit error rates due to changes in the signaling condition. The transceiver on-chip termination calibration process takes approximately 33,000 cal_blk_clk cycles from the de-assertion of the cal_blk_powerdown signal.

PCIe Hard IP Block

Figure 1–50 shows the block diagram of the PCIe hard IP block used to implement the PHY MAC, Data Link Layer, and Transaction Layer for PCIe interfaces. When you use this block, the PIPE interface is used as the interface between the FPGA fabric and the transceiver.

Figure 1–50. PCIe Hard IP High-Level Block Diagram



This block is enabled only when you use the PCIe MegaCore function. For more information about using this block, refer to the *PCI Express Compiler User Guide*.

You can configure the root port or end point in x1, x4, and x8 modes. You can also include instances of both the soft and hard IP PCIe MegaCore function in a single device.

The PCIe hard IP block ignores autonomous and speed change bits set in TS1 and TS2 ordered sets. Altera is fully compatible to the PCIe Base Specification v2.0.

Functional Modes

You can configure Arria II GX and GZ transceivers in one of the following functional modes with the ALTGX MegaWizard Plug-In Manager:

- Basic at 600 Mbps to 6.375 Gbps
- Deterministic latency, used for CPRI and OBSAI protocols
- GbE (1.25 Gbps)
- PCIe (Gen1 at 2.5 Gbps) (Gen2 at 5.0 Gbps for Arria II GZ devices only)
- SDI (HD at 1.485 and 1.4835 Gbps, 3G at 2.97 and 2.967 Gbps)
- SRIO(1.25 Gbps, 2.5 Gbps, and 3.125 Gbps)
- SONET/SDH (OC-12 and OC-48) (OC-96 for Arria II GZ devices only)

- XAUI (3.125 Gbps up to HiGig and HiGig+ at 3.75 Gbps)

The following sections describe the functional modes available in the ALTGX MegaWizard Plug-In Manager that you can set through the **Which protocol will you be using?** option.

Table 1–15 lists the PCS latency for the different functional modes of Arria II devices.

Table 1–15. Functional Modes PCS Latency for Arria II Devices ***(Note 1)***

Functional Mode	PCS Latency (FPGA Fabric-Transceiver Interface Clock Cycles)					
	TX PCS			RX PCS		
Basic	*(2)*			*(2)*		
Deterministic latency	4			6 (byte SERDES enabled)	8 (byte SERDES disabled)	
GbE	5–6			20–24		
PCIe	4–5.5 (byte SERDES enabled)	5–6 (byte SERDES disabled)		11.5–14.5 (byte SERDES enabled)	20–24 (byte SERDES disabled)	
SDI	4–5.5 (byte SERDES enabled)	5–6 (byte SERDES disabled)		6–8 (byte SERDES enabled)	9–11 (byte SERDES disabled)	
SRIO	4–5			6–7 (Rate Match FIFO disable)	19–20 (Rate Match FIFO enable)	
SONET/SDH	5–6 (OC-12)	4–5.5 (OC-48)	4–5.5 (OC-96)	11–13 (OC-12)	7–9 (OC-48)	6.5–8.5 (OC-96)
XAUI	4.5–6			14.5–18		

Notes to Table 1–15:

(1) Not all modes are support by both Arria II GX and Arria II GZ devices. Refer to the respective functional mode sections for the supported modes in each device family.

(2) For basic mode latency values, refer to Figure 1–51, Figure 1–52, Figure 1–53, and Figure 1–54.

Basic Mode

The Arria II GX and GZ transceiver datapath is highly flexible in Basic functional mode. It allows 8-bit and 10-bit PMA-to-PCS interface, which is determined by whether you bypass or use the 8B/10B encoder/decoder.

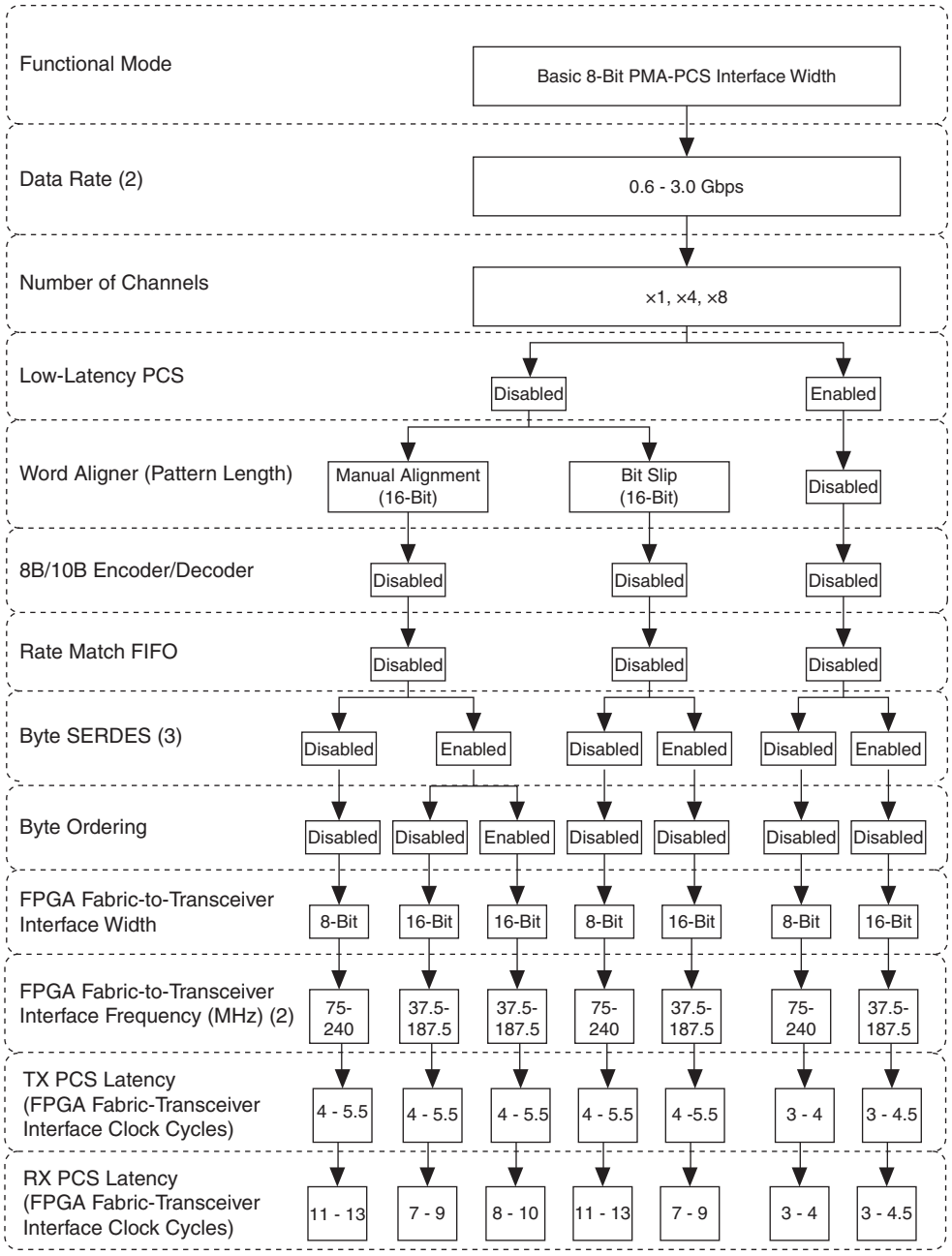
Depending on the targeted data rate, you can optionally bypass the byte serializer and deserializer blocks in Basic mode but the transmitter and RX phase compensation FIFOs are always enabled. The word aligner is always enabled in regular Basic mode, but bypassed in low latency PCS mode, which can be enabled through the **Enable low latency PCS mode** option in the ALTGX MegaWizard Plug-In Manager.

The low latency PCS mode creates a Basic functional mode configuration that bypasses the following transmitter and receiver channel PCS blocks to form a low latency PCS datapath:

- 8B/10B encoder and decoder
- Word aligner
- Deskew FIFO
- Rate match (clock rate compensation) FIFO
- Byte ordering

Figure 1–51 shows the Arria II GX and GZ transceiver configurations allowed in Basic functional mode with an 8-bit wide PMA-PCS interface.

Figure 1–51. Transceiver Configurations in Basic Mode with an 8-Bit Wide PMA-to-PCS Interface *(Note 1)*



Notes to Figure 1–51:

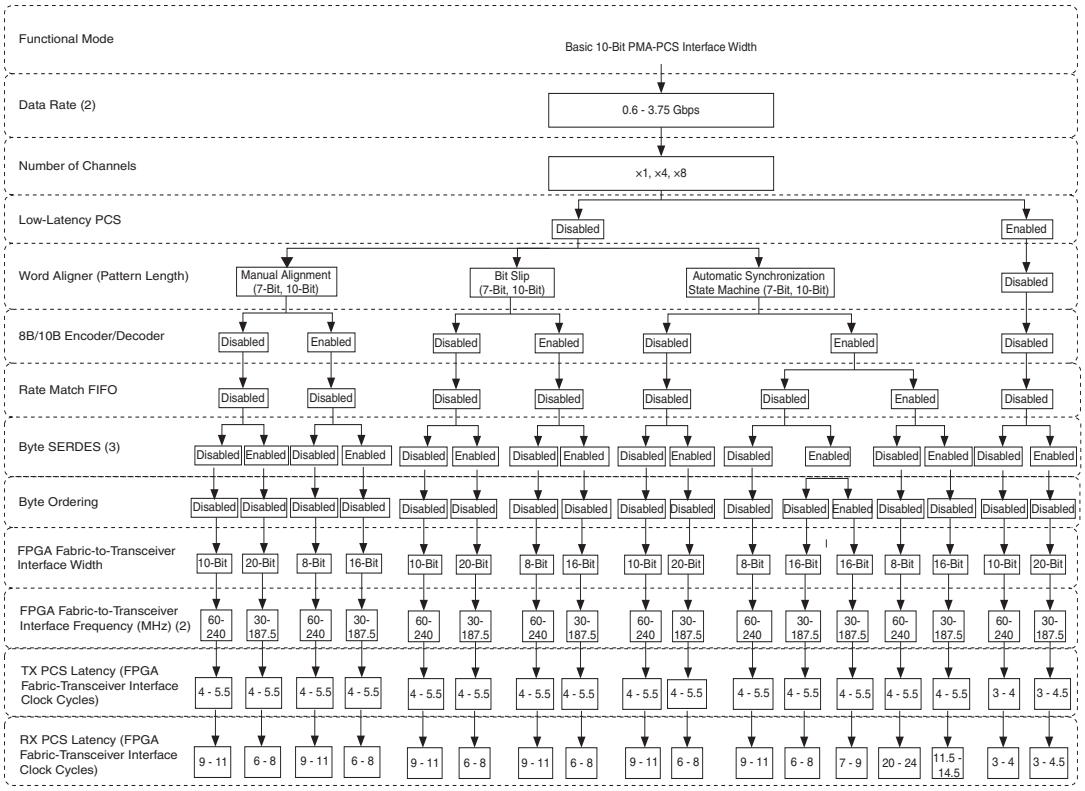
(1) The 10-bit configuration is listed in Figure 1–52.

(2) The maximum data rate specification shown in Figure 1–51 is valid only for the -3 speed grade devices. For data rate specifications for other speed grades offered, refer to the *Device Datasheet for Arria II Devices* chapter.

(3) When you enable byte SERDES, the maximum data is 3G; otherwise, it is 1.92G.

Figure 1–52 shows Arria II GX and GZ transceiver configurations allowed in Basic functional mode with a 10-bit wide PMA-to-PCS interface.

Figure 1–52. Transceiver Configurations in Basic Mode with a 10-Bit Wide PMA-to-PCS Interface ***(Note 1)***



Notes to Figure 1–52:

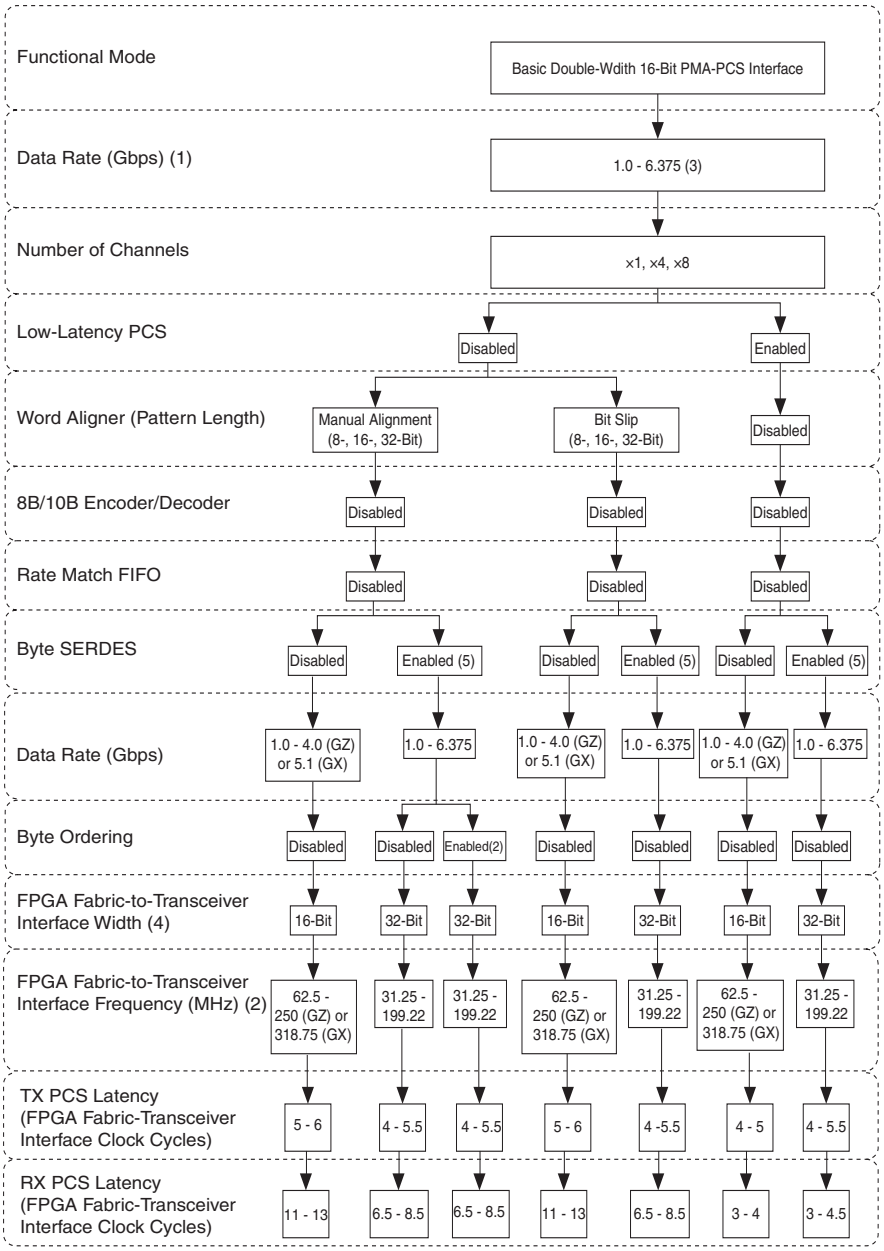
(1) The 8-bit configuration is listed in Figure 1–51.

(2) The maximum data rate specification shown in Figure 1–52 is valid only for the -3 speed grade devices with byte SERDES enabled. For data rate specifications for other speed grades offered, refer to the *Device Datasheet for Arria II Devices* chapter.

(3) When you enable byte SERDES, the maximum data is 3G; otherwise, it is 1.92G.

Figure 1–53 shows Arria II transceiver configurations allowed in Basic double-width functional mode with a 16-bit PMA-PCS interface.

Figure 1–53. Transceiver Configurations in Basic Double-Width Mode with a 16-Bit PMA-PCS Interface for Arria II Devices



Notes to Figure 1–53:

(1) The maximum data rate specification shown in Figure 1–53 is valid only for the -3 (fastest) speed grade devices.

(2) The byte ordering block is available only if you select the word alignment pattern length of 16 or 32 bits.

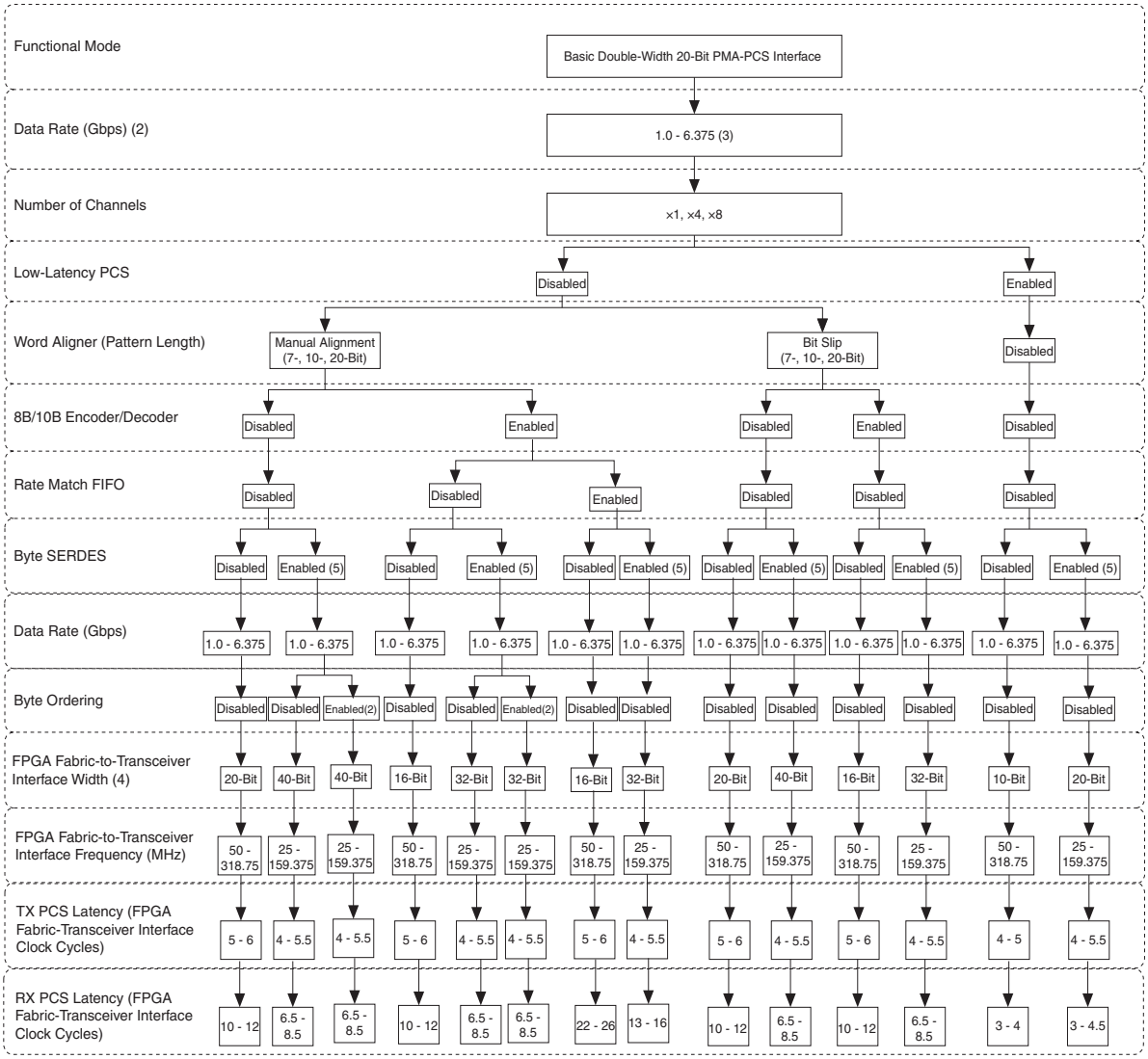
(3) Arria II GX I3 devices can support up to 6.375 Gbps. For more information, refer to the *Device Datasheet for Arria II Devices*.

(4) The FPGA fabric-to-transceiver interface width of 32-bits applies to Arria II GZ devices only.

(5) Byte SERDES is only supported for Arria II GZ devices in double-wide mode.

Figure 1–54 shows Arria II transceiver configurations allowed in Basic double-width functional mode with a 20-bit PMA-PCS interface.

Figure 1–54. Transceiver Configurations in Basic Double-Width Mode with a 20-Bit PMA-PCS Interface for Arria II Devices



Notes to Figure 1–54:

(1) The maximum data rate specification shown in Figure 1–54 is valid only for the -3 (fastest) speed grade devices.

(2) The byte ordering block is available only if you select the word alignment pattern length of 20 bits.

(3) Arria II GX I3 devices can support up to 6.375 Gbps. For more information, refer to the *Device Datasheet for Arria II Devices*.

(4) The FPGA fabric-to-transceiver interface width of 40-bits applies to Arria II GZ devices only.

(5) Byte SERDES is only supported for Arria II GZ devices in double-wide mode.

Deterministic Latency

This mode is typically used to create a CPRI or Open Base Station Architecture Initiative Reference Point 3 (OBSAI RP3) interface to connect radio frequency (RF) processing remote radio heads located at the top of cell phone towers with the base band processing equipment typically found at the bottom of cell phone towers.

CPRI and OBSAI protocols have a requirement for the accuracy of the round trip delay measurement for single-hop and multi-hop connections to be within ± 16.276 ns. For single hops, the round trip delay may only vary within ± 16.276 ns. For multi-hop connections, the round trip variation is equal to ± 16.276 ns divided by the number of hops.



Deterministic latency is the only functional mode that allows 16-bit and 20-bit data on the PCS-to-PMA interface. This is to allow data rates of 2457.6, 3072, 4915.2, and 6144 Mbps for the CPRI protocol and to allow 3072 and 6144 Mbps data rates for the OBSAI protocol.

When you choose the deterministic latency protocol in the ALTGX MegaWizard Plug-In Manager, the bit-slip circuitry in the transmitter channel is automatically enabled and the RX phase compensation FIFO is automatically set to register mode. In addition, two extra ports are created—the `rx_bitslipboundaryselectout` output port from the receiver's word aligner and the `tx_bitslipboundaryselect` input port for the transceiver bit-slip circuitry. You can also set the TX phase compensation FIFO in register mode.



In register mode, the phase compensation FIFO acts as a register and removes the uncertainty in latency. To ensure that the phase relationship between the low-speed parallel clock and the CMU PLL input reference clock is deterministic, you can enable the CMU PLL feedback path, which is only available in this mode. When the feedback path is enabled, you must provide an input reference clock to the CMU PLL that has the same frequency as the low-speed parallel clock.

The information on the `rx_bitslipboundaryselectout[4:0]` output port helps calculate the latency through the receiver datapath. Connect `rx_bitslipboundaryselectout[4:0]` to `tx_bitslipboundaryselect[4:0]` to cancel out the latency uncertainty.

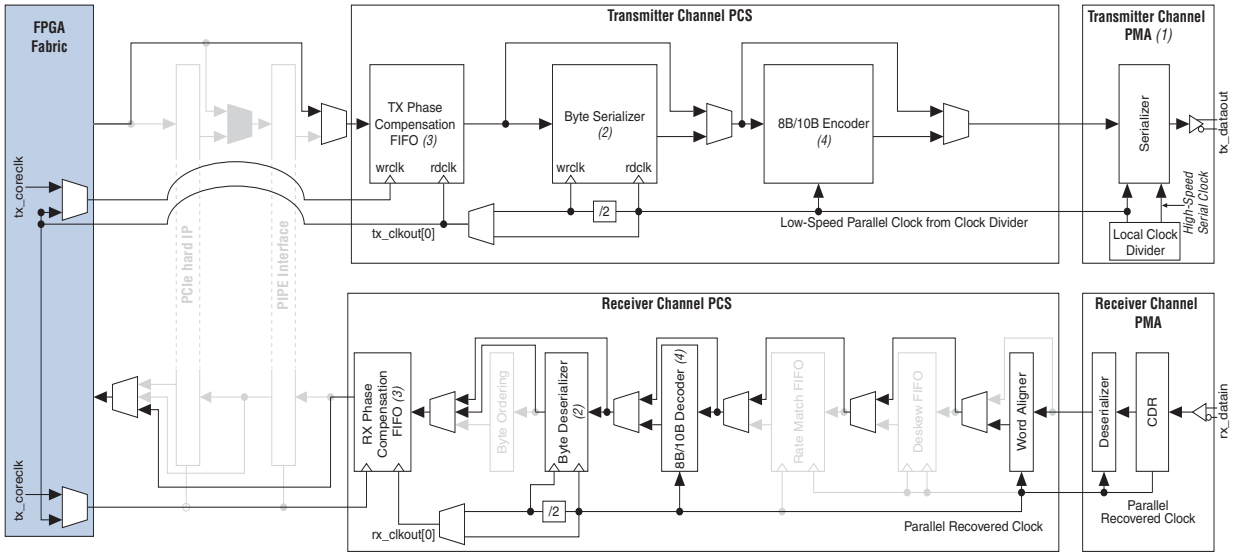
The number of bits slipped in the receiver's word aligner is shown on the `rx_bitslipboundaryselectout[4:0]` output port. In 8- or 10-bit channel width, the number of bits slipped in the receiver path is given out sequentially on this output. For example, if zero bits are slipped, the output on `rx_bitslipboundaryselectout[4:0]` shows a value of 0(5'b00000); if two bits are slipped, the output on `rx_bitslipboundaryselectout[4:0]` shows a value of 2 (5'b00010). In 16- or 20-bit channel width, the output is 19 minus the number of bits slipped. For example, if zero bits are slipped, the output on `rx_bitslipboundaryselectout[4:0]` shows a value of 19 (5'b10011); if two bits are slipped, the output on `rx_bitslipboundaryselectout[4:0]` shows a value of 17 (5'b10001).



You can slip zero to nine bits with 8- or 10-bit channel width and you can slip zero to 19 bits with 16- or 20-bit channel width.

Figure 1–55 shows the block diagram for deterministic latency.

Figure 1–55. Deterministic Latency Functional Mode



Notes to Figure 1–55:

(1) The transmitter is in bit-slip mode.

(2) This block is optional in this mode.

(3) The RX phase compensation FIFO is automatically set in register mode. However, you have the option to set the TX phase compensation FIFO in register mode, which is not set by default.

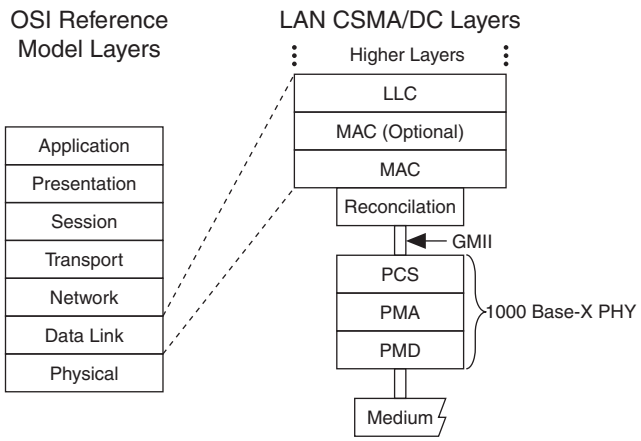
(4) Typically, the 8B/10B encoder and decoder are used when you use deterministic latency to implement CPRI or OBSAI protocols. However, you have the option to disable this module.

GbE

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 GbE system. It shields the MAC layer from the specific nature of the underlying medium. The 1000 Base-X PHY, which has a physical interface data rate of 1.25 Gbps, is divided into three sublayers—the physical coding sublayer (PCS), physical media attachment (PMA), and physical medium dependent (PMD). These sublayers interface with the MAC through the gigabit medium independent interface (GMII).

Figure 1–56 shows the 1000 Base-X PHY position in a GbE OSI reference model.

Figure 1–56. 1000 Base-X PHY in a GbE OSI Reference Model



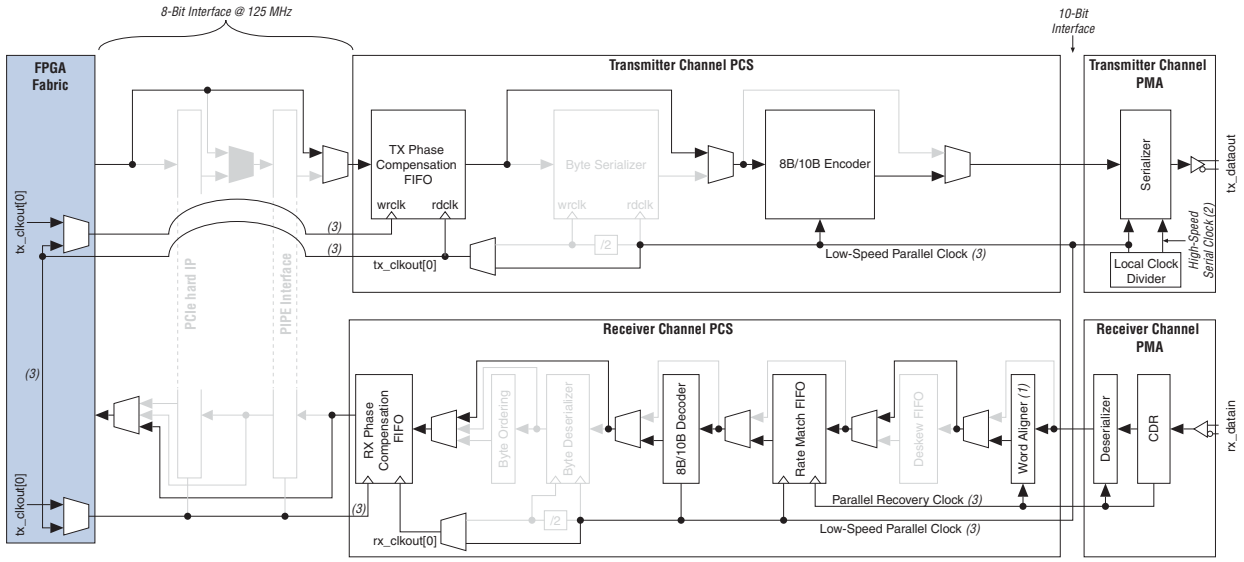
In GbE functional mode, Arria II GX and GZ devices have built-in circuitry to support the following PCS and PMA functions defined in the IEEE 802.3 specification:

- 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
- Optional `rx_recovclkout` port enables the recovered clock at the pin level (use with the voltage-controlled crystal oscillator [VCXO])
- Serialization and deserialization

Arria II GX and GZ transceivers do not have built-in support for some 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 1–57 shows the transceiver datapath when configured in GbE functional mode.

Figure 1–57. GbE Functional Mode



Notes to Figure 1–57:

(1) The word aligner uses automatic synchronization state machine mode (7-bit comma, 10-Bit /K28.5/).
(2) High-speed serial clock is running at 625 MHz.
(3) These clocks are running at 125 MHz.

Idle Ordered-Set Generation in GbE Mode

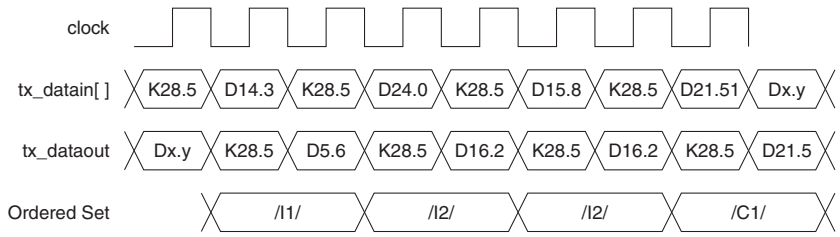
The IEEE 802.3 specification requires the GbE PHY to transmit idle ordered sets (/I/) continuously and repetitively whenever the GMII is idle. This ensures that the receiver maintains bit and word synchronization whenever there is no active data to be transmitted.

In GbE 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, an /I2/ ordered set is generated. The disparity at the end of an /I1/ is the opposite of that at the beginning of the /I1/. The disparity at the end of an /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.

/D14.3/, /D24.0/, and /D15.8/ are replaced by /D5.6/ or /D16.2/ (for /I1/ and /I2/ ordered sets). /D21.5/ (part of the /C1/ order set) is not replaced.

Figure 1–58 shows the automatic idle ordered set generation.

Figure 1–58. Example of Automatic Ordered Set Generation

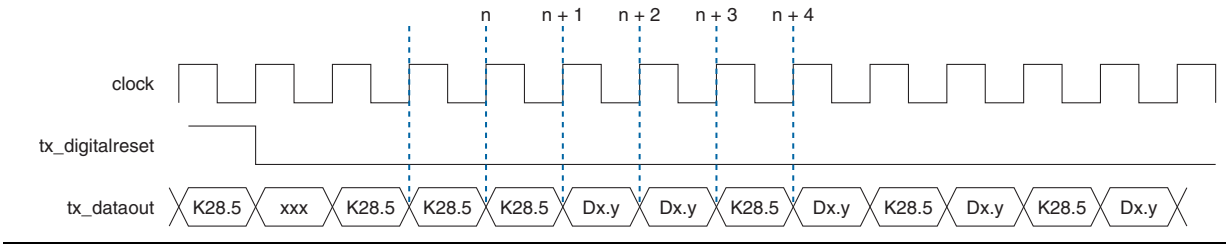


GbE Mode Reset Condition

After de-assertion of the `tx_digitalreset` signal, the GbE 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. An IEEE802.3-compliant GbE synchronization state machine treats this as an error condition and goes into the Loss-of-Sync state.

Figure 1–59 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 are discounted and three additional ordered sets are required for successful synchronization.

Figure 1–59. Example of Reset Condition in GbE Mode



Word Aligner in GbE Mode

The word aligner in GbE functional mode is configured in automatic synchronization state machine mode, which complies with the IEEE P802.3ae standard. 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 sets.

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. Each invalid code group increases the error count. The error count can be reduced by 1 if the state machine sees four continuous valid code groups. The receiver loses synchronization when it detects four invalid code groups separated by less than three valid code groups, or when it is reset.

Rate Match FIFO in GbE Mode

In GbE mode, the rate match FIFO is capable of compensating up to ±100 PPM (200 PPM total) difference between the upstream transmitter and the local receiver reference clock. The GbE 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 P802.3ae specification.

The rate match operation begins after the synchronization state machine in the word aligner indicates synchronization has been acquired by driving the rx_syncstatus signal high. The rate match FIFO deletes or inserts both symbols of the /I2/ ordered sets (which consist of /K28.5/ and /D16.2/) to prevent the rate match FIFO from overflowing or underflowing. It can insert or delete as many /I2/ ordered sets as necessary to perform the rate match operation.

If you have the auto negotiation state machine in the FPGA fabric, 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 underrunning during the auto negotiation phase.

The status flags rx_rmfifodatadeleted and rx_rmfifodatainserted, indicating rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric. These two flags are asserted for two clock cycles for each deleted and inserted /I2/ ordered set, respectively.

Figure 1–60 shows an example of rate match FIFO deletion where three symbols must be deleted. Because the rate match FIFO can only delete /I2/ ordered sets, it deletes two /I2/ ordered sets (four symbols deleted).

Figure 1–60. Example of Rate Match Deletion in GbE Mode

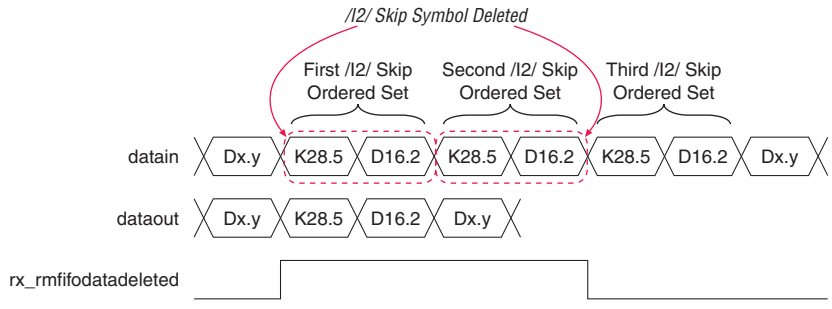
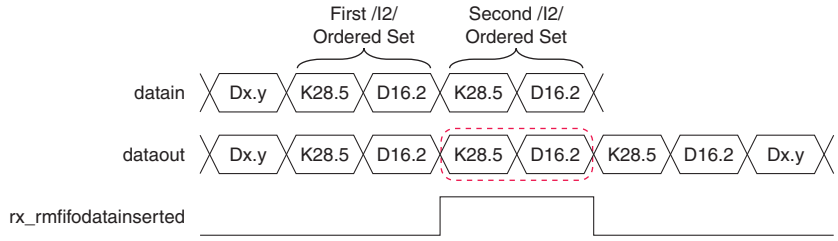


Figure 1–61 shows an example of rate match FIFO insertion in the case where one symbol must be inserted. Because the rate match FIFO can only insert /I2/ ordered sets, it inserts one /I2/ ordered set (two symbols inserted).

Figure 1–61. Example of Rate Match Insertion in GbE Mode



The rate match FIFO does not insert or delete code groups automatically to overcome FIFO empty or full conditions. In this case, the rate match FIFO 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. You must then assert the `rx_digitalreset` signal to reset the receiver PCS blocks.

PCIe Mode

Intel Corporation has developed a PHY interface for the PCIe architecture specification to enable implementation of a PCIe-compliant physical layer device. This specification also defines a standard interface between the physical layer device and the media access control layer (MAC). Version 2.00 of the specification provides implementation details for a PCIe-compliant physical layer device at both Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rates.

Arria II GX and GZ transceivers support ×1, ×4, and ×8 lane configurations in PCIe functional mode at Gen1 (2.5 Gbps) data rates. Arria II GZ devices also support ×1 and ×4 lane configurations in PCIe functional mode at Gen2 (5.0 Gbps). In PCIe ×1 configuration, the PCS and PMA blocks of each channel are clocked and reset independently. PCIe ×4 and ×8 configurations support channel bonding for four-lane and eight-lane PCIe links, where the PCS and PMA blocks of all bonded channels share common clock and reset signals.

You can configure Arria II GX and GZ transceivers to implement a Version 2.00 PCIe-compliant PHY using one of the following methods:

- PCIe Compiler—This method allows you to use the Arria II GX and GZ devices built-in PCIe hard IP blocks to implement the PHY-MAC layer, Data Link layer, and Transaction layer of the PCIe protocol stack. In this mode, each Arria II GX and GZ transceiver channel uses 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 used only in this mode and cannot be bypassed.
- ALTGX MegaWizard Plug-In Manager—This method requires implementing the PHY-MAC layer, Data Link layer, and Transaction layer in the FPGA fabric using a soft IP. Use this method if you do not use the PCIe hard IP block. (You cannot access the PCIe hard IP block if you use this method.)

For descriptions of PCIe hard IP architecture and PCIe mode configurations allowed when using the PCIe hard IP block, refer to the *PCI Express Compiler User Guide*.

For more information about transceiver datapath clocking in different PCIe configurations, refer to the *Transceiver Clocking in Arria II Devices* chapter.

The transmitter datapath in PCIe mode consists of the:

- PIPE interface
- TX 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 PCIe mode consists of the:

- Receiver input buffer with signal detect circuitry
- 1:10 deserializer
- Word aligner that implements PCIe-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)
- RX phase compensation FIFO
- PIPE interface

Table 1–16 lists features supported in PCIe functional mode for Gen1 (2.5 Gbps) and Gen2 (5.0 Gbps) data rate configurations.

Table 1–16. Supported Features in PCIe Mode for Arria II Devices (Part 1 of 2)

Feature	2.5 Gbps (Gen1)	5.0 Gbps (Gen2) *(1)*
×1, ×4, ×8 link configurations	✓	Only ×1 and ×4 are supported
PCIe-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	✓	✓

Table 1–16. Supported Features in PCIe Mode for Arria II Devices (Part 2 of 2)

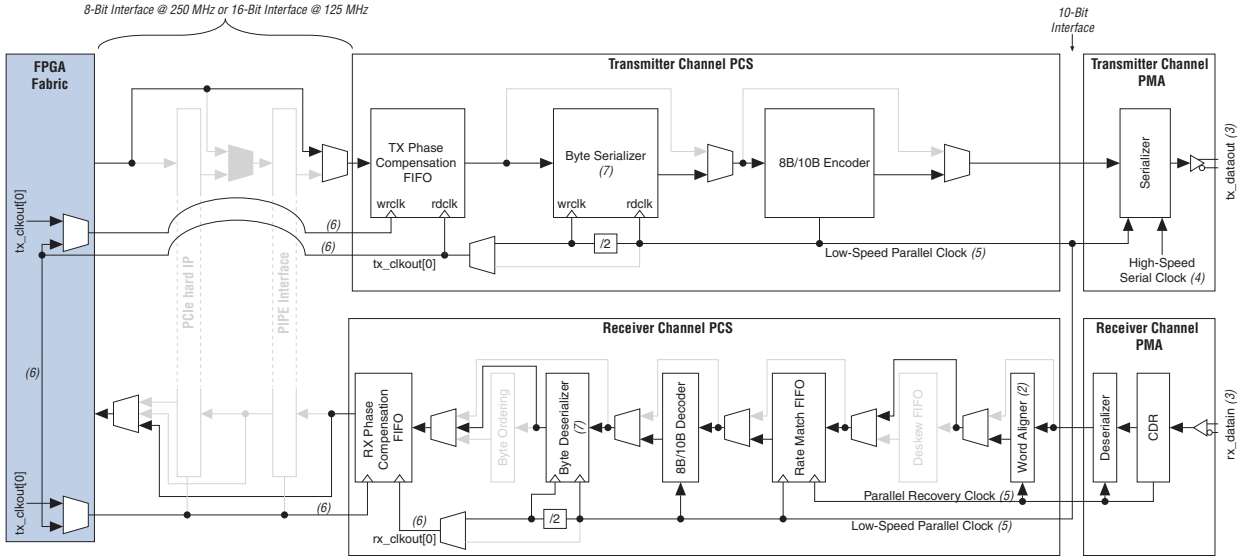
Feature	2.5 Gbps (Gen1)	5.0 Gbps (Gen2) *(1)*
Dynamic switch between 2.5 Gbps and 5 Gbps signaling rate	—	✓
Dynamically selectable transmitter margining for differential output voltage control	—	✓
Dynamically selectable transmitter buffer de-emphasis of -3.5 db and -6 db	—	✓

Note to Table 1–16:

(1) For Arria II GZ devices only.

Figure 1–62 shows the Arria II GX and GZ transceiver datapath when configured in PCIe functional mode.

Figure 1–62. Arria II GX and GZ Transceiver Datapath in PCIe Mode *(Note 1)*



Notes to Figure 1–62:

(1) The transceiver datapath clock varies between non-bonded (×1) and bonded (×4 and ×8) configurations in PCIe mode, described in the *Transceiver Clocking for Arria II Devices* chapter.

(2) The word aligner uses automatic synchronization state machine mode (10-Bit /K28.5/).

(3) This can be ×1, ×4, or ×8 at 2.5 Gbps or ×1 or ×4 at 5.0 Gbps (for Arria II GZ devices only).

(4) The high-speed serial clock is running at 1.25 GHz.

(5) The parallel clocks are running at 250 MHz.

(6) This clock is running at 125 MHz if the byte serializer and deserializer are used. Otherwise, this clock is running at 250 MHz.

(7) If you use the PCIe hard IP, you can enable the byte serializer and deserializer with an 8-bit FPGA fabric-to-transceiver interface running at 250 MHz or disabled with a 16-bit FPGA fabric-to-transceiver interface running at 125 MHz. Otherwise, these blocks are always disabled and your 16-bit FPGA fabric-to-transceiver interface is running at 125 MHz.

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:

- Manages the PIPE power states
- Forces the transmitter buffer in the electrical idle state
- Initiates the receiver detect sequence
- Controls 8B/10B encoder disparity when transmitting compliance pattern
- Indicates completion of various PHY functions, such as receiver detection and power state transitions on the `pipephydonestatus` signal
- Encodes receiver status and error conditions on the `pipestatus[2:0]` signal as specified in the PIPE specification

The following subsections describe each Arria II GX and GZ transceiver function.

Power State Management

The PCIe specification defines four power states 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 PCIe 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 PCIe specification provides the mapping of these power states to the long-term sample storage module (LTSSM) states specified in the PCIe Base Specification 2.0. The PHY-MAC layer is responsible for implementing the mapping logic between the LTSSM states and the four power states in the PCIe-compliant PHY.

The PIPE interface in Arria II GX and GZ transceivers provide an input port (`powerdn[1:0]`) to set the transceivers in one of the four power states, as shown in Table 1–17.

Table 1–17. powerdn [1:0] Port Power State Functions and Descriptions for Arria II Devices

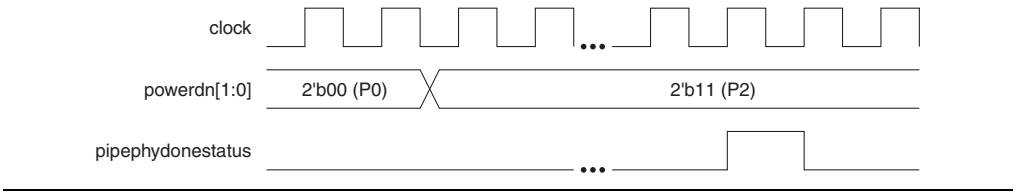
powerdn [1:0] Values	Power State	Description	Function
2'b00	P0	Normal operation mode	Transmits normal data, transmits electrical idle, or enters into loopback mode
2'b01	P0s	Low recovery time saving state	Only transmits electrical idle
2'b10	P1	High recovery time power saving state	Transmitter buffer is powered down and can perform a receiver detect while in this state
2'b11	P2	Lowest power saving state	Transmits electrical idle or a beacon to wake up the downstream receiver

When the device transitions from the P0 power state to lower power states (P0s, P1, and P2), the PCIe specification requires the physical layer device to implement power saving measures. Arria II GX and GZ 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 a successful power state transition by asserting the pipephydonestatus signal for one parallel clock cycle as specified in the PCIe 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 1–63 shows an example waveform for a transition from the P0 to the P2 power state.

Figure 1–63. Example of Power State Transition from P0 to P2



The PCIe specification allows the PIPE interface to perform protocol functions such as 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 1–18 lists the logic levels that the PHY-MAC layer must drive on the tx_detectrxloopback and tx_forceelecidle signals in each power state.

Table 1–18. Logic Levels for the PHY-MAC Layer for Arria II Devices

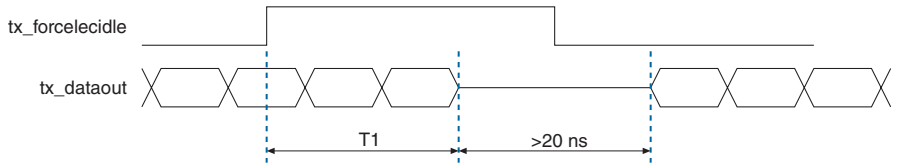
Power State	tx_detectrxloopback Value	tx_forceelecidle Value
P0	0: normal mode 1: loopback mode	0: must be de-asserted 1: illegal mode
P0s	Don't care	0: illegal mode 1: must be asserted in this state
P1	0: electrical state 1: receiver detect	0: illegal mode 1: must be asserted in this state
P2	Don't care	De-asserted in this state for sending beacon. Otherwise asserted.

Transmitter Buffer Electrical Idle

The PCIe specification requires the transmitter buffer to be in electrical idle in the P1 power state, as shown in Table 1–18. During electrical idle, the transmitter buffer differential and common mode output voltage levels are compliant to the PCIe Base Specification 2.0 for both PCIe Gen1 and Gen2 data rates.

In Arria II GX and GZ transceivers, asserting the input signal tx_forceelecidle puts the transmitter buffer in that channel in the electrical idle state. Figure 1–64 shows the relationship between asserting 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 a minimum of 8 ns. When in the electrical idle state, the PCIe protocol requires the transmitter buffer to stay in electrical idle for a minimum of 20 ns for both Gen1 and Gen2 data rates.

Figure 1–64. Transmitter Buffer Electrical Idle State



Receiver Detection

During the detect substate of the LTSSM state machine, the PCIe 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 PCIe specification requires that a receiver detect operation be performed during the P1 power state where the transmitter output buffer is in electrical idle (tri-stated).

This feature requires the transmitter output buffer to be tri-stated (in electrical idle mode), have OCT utilization, and run at 125 MHz on the `fixedclk` signal. You can enable this feature in PCIe functional mode by setting the `tx_forceelecidle` and `tx_detectrxloopback` ports to 1'b1.

When the `tx_detectrxloopback` signal 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. On receiving this command signal, the receiver detect circuitry creates a step voltage at the transmitter output buffer common mode voltage. If an active receiver (that complies with the PCIe 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.

If a far-end receiver is successfully detected, the PIPE interface block asserts the `pipephydonestatus` signal 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 the `pipephydonestatus` signal for one clock cycle and synchronously drives the `pipestatus[2:0]` signal to 3'b000.

There is some latency after asserting the `tx_detectrxloopback` signal before receiver detection is indicated on the `pipephydonestatus` port. In addition, 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.

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 PCIe Base Specification 2.0. Receiver detect circuitry communicates the status of the receiver detect operation to the PIPE interface block.

Figure 1–65 and Figure 1–66 show the receiver detect operation where a receiver was successfully detected and where a receiver was not detected, respectively.

Figure 1–65. Receiver Detect Successful Operation

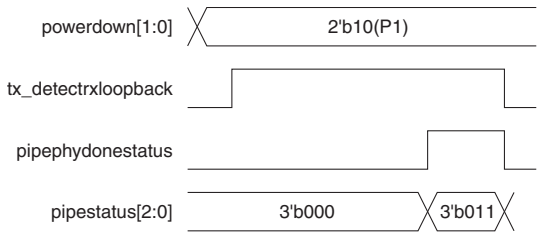
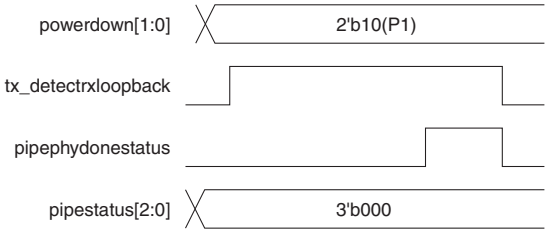


Figure 1–66. Receiver Detect Unsuccessful Operation



Compliance Pattern Transmission Support

The LTSSM state machine can enter the polling.compliance substate where the transmitter must transmit a compliance pattern as specified in the PCIe Base Specification 2.0. The polling.compliance substate is intended to assess if the transmitter is electrically compliant with the PCIe 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/

The PCIe 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 the `tx_forcedispcompliance` signal forces the associated parallel transmitter data on the `tx_datain` port to transmit with a negative current running disparity.



For 8-bit transceiver channel width configurations, you must drive the `tx_forcedispcompliance` signal 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 the `tx_forcedispcompliance[1:0]` signal high in the same parallel clock cycle as /K28.5/D21.5/ of the compliance pattern on the `tx_datain` port.

Figure 1–67 and Figure 1–68 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 1–67. Compliance Pattern Transmission Support, 8-Bit Wide Channel Configuration

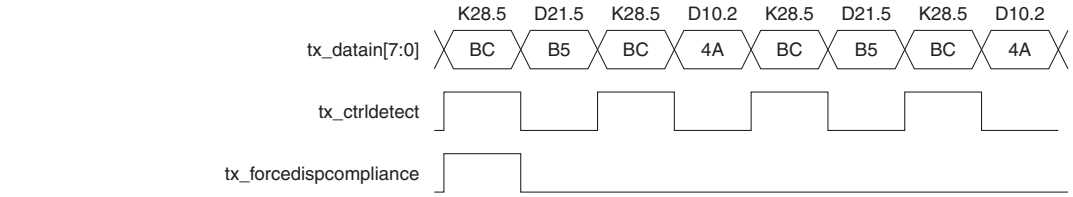
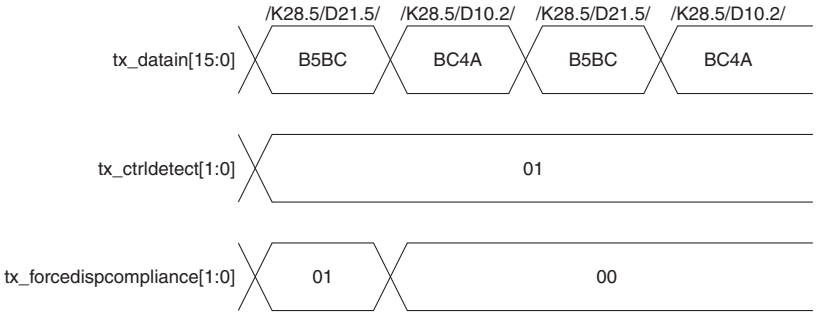


Figure 1–68. Compliance Pattern Transmission Support, 16-Bit Wide Channel Configuration



Receiver Status

The PCIe specification requires the PHY to encode the receiver status on a 3-bit `RxStatus[2:0]` signal. The PIPE interface block receives status signals from the transceiver channel PCS and PMA blocks and drives the status on the 3-bit output signal (`pipestatus[2:0]`) to the FPGA fabric. The encoding of the status signals on the `pipestatus[2:0]` port is compliant with the PCIe specification and is listed in Table 1–19.

Table 1–19. Encoding of the Status Signals on the pipestatus[2:0] Port for Arria II Devices

pipestatus[2:0]	Description	Error Condition Priority *(1)*
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

Note to Table 1–19:

(1) The PIPE interface follows the priority listed in Table 1–19 while encoding the receiver status on the `pipestatus[2:0]` port. Two or more error conditions; for example, 8B/10B decode error (code group violation), rate match FIFO overflow or underflow, or receiver disparity error, can occur simultaneously. When this happens, PIPE drives 3'b100 on the `pipestatus[2:0]` signal.

Fast Recovery Mode

The PCIe 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 PCIe specification requires the physical layer device to acquire bit and byte synchronization after you transition from L0s to L0 state within 16 ns to 4 µs.

If the Arria II GX and GZ receiver CDR is configured in Automatic Lock mode, the receiver cannot meet the PCIe 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 and GZ transceiver has built-in fast recovery circuitry that you can optionally enable in the ALTGX MegaWizard Plug-In Manager with the **Enable fast recovery mode** option.

Fast recovery circuitry controls the receiver CDR `rx_locktorefclk` and `rx_locktodata` signals to force the receiver CDR in LTR or LTD modes, by relying 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. It is self-operational and does not require user inputs.



When you enable fast recovery mode, the `rx_locktorefclk` and `rx_locktodata` ports are not available in the ALTGX MegaWizard Plug-In Manager.

Electrical Idle Inference

The PCIe protocol allows inferring the electrical idle condition at the receiver instead of detecting the electrical idle condition using analog circuitry. PCIe Base Specification 2.0, section .2.4.3, specifies conditions to infer electrical idle at the receiver in various sub-states of the LTSSM state machine.

In all PCIe 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 PCIe Base Specification 2.0.

You can enable the electrical idle inference module by selecting the **Enable electrical idle inference functionality** option in the ALTGX MegaWizard Plug-In Manager. This feature infers electrical idle depending on the logic level driven on the `rx_elecidleinfersel[2:0]` input signal. The electrical idle inference module drives the `pipeelecidle` signal high in each receiver channel when an electrical idle condition is inferred. 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 1–20.

Table 1–20. Electrical Idle Inference Conditions for Arria II Devices (Part 1 of 2)

rx_elecidleinfersel[2:0]	LTSSM State	Description
3'b100	L0	Absence of update FC or alternatively skip ordered set in 128 µs window
3'b101	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

Table 1–20. Electrical Idle Inference Conditions for Arria II Devices (Part 2 of 2)

rx_elecidleinfersel[2:0]	LTSSM State	Description
3'b110	Recovery.Speed when successful speed negotiation = 1'b0	Absence of an exit from electrical idle in 2000 UI interval
3'b111	Loopback.Active (as slave)	Absence of an exit from electrical idle in 128 µs window

The electrical idle inference module cannot detect an electrical idle exit condition based on the reception of the electrical idle exit ordered set (EIEOS), as specified in the PCIe 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. Otherwise, the electrical idle inference module is disabled. In this case, the `rx_signaldetect` signal from the signal detect circuitry in the receiver input buffer is inverted and driven as the `pipeelecidle` signal.

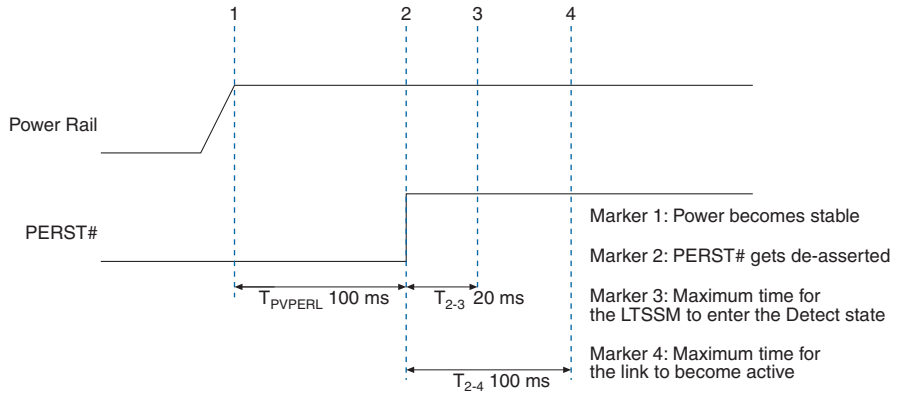
PCIe Cold Reset Requirements

The PCIe Base Specification 2.0 defines the following three types of conventional resets to the PCIe 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

Figure 1–69 shows the PCIe cold reset timing requirements.

Figure 1–69. PCIe Cold Reset Requirements



The following is the time taken by a PCIe port, implemented in an Arria II GX or GZ device, to go from the power-up to the link-active state:

- Power-on reset—begins after power rails become stable, which typically takes 12 ms
- FPGA configuration and 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

To meet the PCIe specification of 200 ms from the power-on to the link-active state, the Arria II GX and GZ device configuration time must be less than 148 ms (200 ms to 12 ms for power on reset, 40 ms for the link to become active after PERST# de-assertion).

For the typical Arria II GX and GZ configuration times using the Fast Passive Parallel (FPP) configuration scheme at 125 MHz, refer to the *Device Datasheet for Arria II Devices*.

For more information about the FPP configuration scheme, refer to the *Configuration, Design Security, Remote System Upgrades in Arria II Devices* chapter.

Most flash memories available in the market can run up to 100 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 and GZ devices at 125 MHz.

SDI

The Society of Motion Picture and Television Engineers (SMPTE) defines various SDI standards for the 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 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 2970 Mbps or 2967 Mbps.

Table 1–21 lists the data rates, refclk frequencies, and interface widths supported by Arria II GX and GZ transceivers in SDI mode.

Table 1–21. SDI Mode Data Rates, refclk Frequencies, and Interface Widths in Arria II Devices (Part 1 of 2)

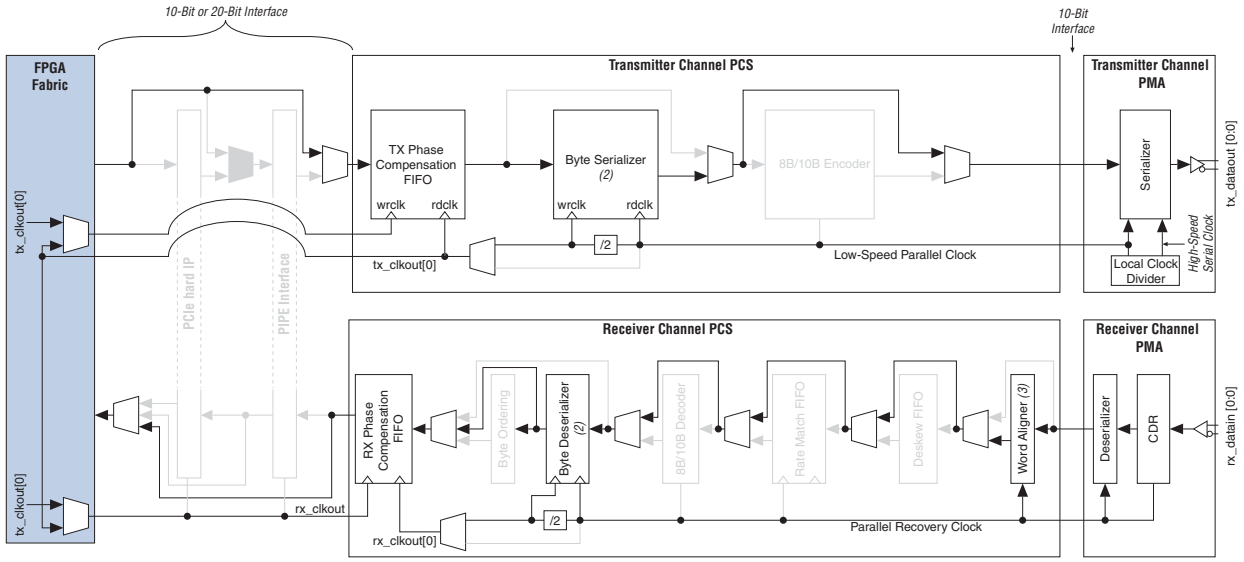
Configuration	Data Rate (Mbps)	Support refclk Frequencies (MHz)	FPGA Fabric-to-Transceiver Width	Byte Serializer/Deserializer Usage
HD	1483.5	74.175	20-bit	Used
		148.35	10-bit	Not used
	1485	74.25	20-bit	Used
		148.5	10-bit	Not used

Table 1–21. SDI Mode Data Rates, refclk Frequencies, and Interface Widths in Arria II Devices (Part 2 of 2)

Configuration	Data Rate (Mbps)	Support refclk Frequencies (MHz)	FPGA Fabric-to-Transceiver Width	Byte Serializer/Deserializer Usage
3G	2967	148.35	20-bit	Used
		296.7		
	2970	148.5		
		297		

Figure 1–70 shows the transceiver datapath when configured in SDI mode.

Figure 1–70. SDI Mode Datapath ***(Note 1)***



Notes to Figure 1–70:

(1) For the frequency, data rate, and interface width supported, refer to Table 1–21 on page 1–72.

(2) This allows the fabric-to-transceiver interface to run below the maximum interface frequency. For more information, refer to Table 1–21 on page 1–72.

(3) The word aligner uses bit-slip mode. However, this block is not useful because word alignment and framing occurs after de-scrambling. Altera recommends driving the ALTGXB megafunction `rx_bitslip` signal low to prevent the word aligner from inserting bits in the received data stream.

In HD-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. Similarly, SDI receiver functions, such as de-scrambling, framing, and CRC checker, must be implemented in the FPGA logic array.

SRIO

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.

The SRIO physical layer specification defines three line rates—1.25 Gbps, 2.5 Gbps, and 3.125 Gbps. It also defines two link widths—single-lane (×1) and bonded four-lane (×4) at each line rate. Arria II GX and GZ transceivers support only single-lane (×1) configuration at all three line rates. You can instantiate four ×1 channels configured in SRIO mode to achieve one non-bonded ×4 SRIO link. The four receiver channels in this ×4 SRIO link do not have lane alignment or deskew capability.

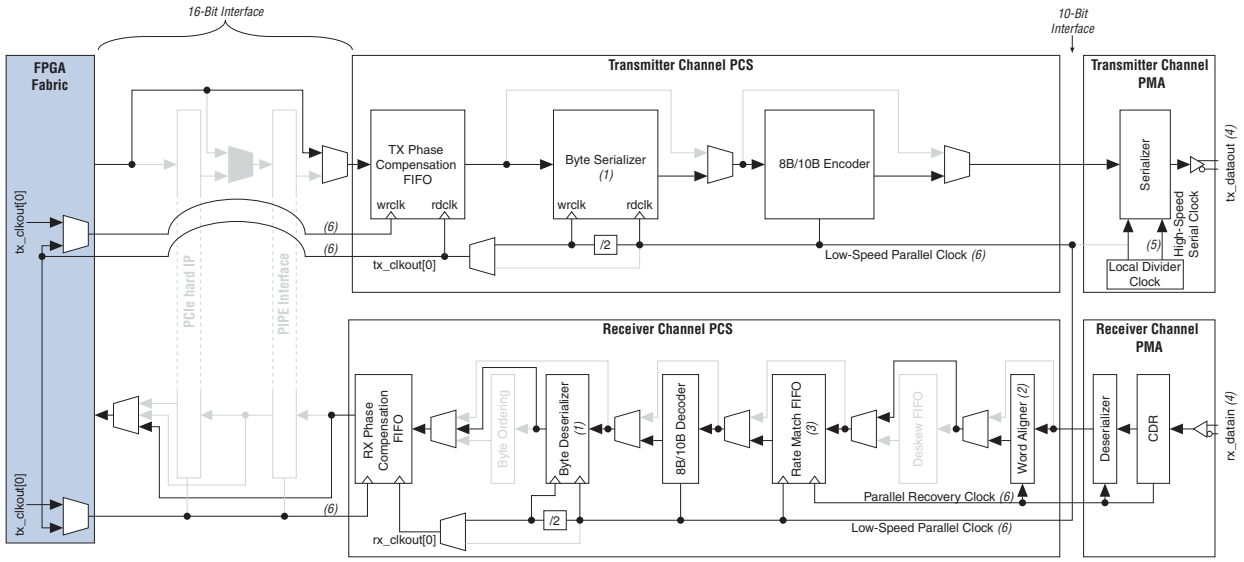
Arria II GX and GZ transceivers, when configured in SRIO functional mode, provide the following PCS and PMA functions:

- 8B/10B encoding and decoding
- Word alignment
- Lane synchronization state machine
- Clock recovery from the encoded data
- Serialization and deserialization

Arria II GX and GZ transceivers do not have built-in support for some PCS functions, such as 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.

Figure 1–71 shows the ALTGX transceiver datapath when configured in SRIO mode.

Figure 1–71. SRIO Mode Datapath



Notes to Figure 1–71:

(1) This allows the fabric-to-transceiver interface to run below the maximum interface frequency and is always enabled for SRIO functional mode.
(2) The word aligner uses the automatic synchronization state machine (10 bit /K28.5/) and is compliant with the SRIO protocol.
(3) This module is optional.
(4) This can run at 1.25, 2.5, or 3.125 Gbps.
(5) This is running at half the rate of the data rate.
(6) This is running at 62.5 MHz for 1.25 Gbps data rate, 125 MHz for 2.5 Gbps data rate, or 156.25 MHz for 3.125 Gbps data rate.

In SRIO mode, the ALTGX MegaWizard Plug-In Manager automatically defaults the synchronization state machine to indicate synchronization (a high logic level on the `rx_syncstatus` port) when the receiver acquires 127 K28.5 (10'b0101111100 or 10'b1010000011) synchronization code groups without receiving an intermediate invalid code group. When synchronized, the state machine indicates loss of synchronization (a low logic level on the `rx_syncstatus` port) when it detects three invalid code groups separated by less than 255 valid code groups, or when it is reset.

SRIO only allows one insertion or deletion of the skip character /R/from the /K/, /R/, /R/, /R/ clock compensation sequence. However, the Arria II GX and GZ rate match FIFO may perform multiple insertions or deletions if the PPM difference is more than the ±200 PPM range. Ensure that the PPM difference in your system is less than ±200 ppm.

SONET/ SDH

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.

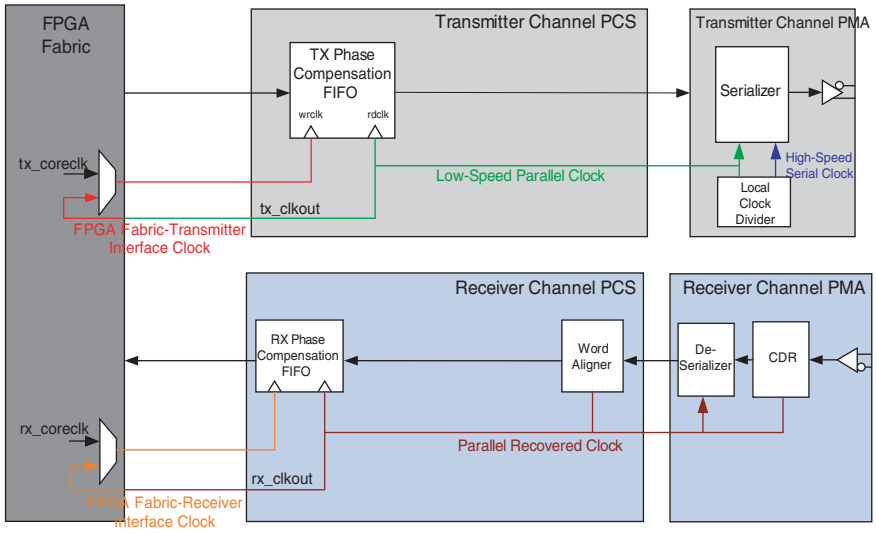
You can use Arria II GX and GZ transceivers as physical layer devices in a SONET/SDH system. These transceivers provide support for SONET/SDH protocol-specific functions and electrical features; for example, alignment to an A1A2 or A1A1A2A2 pattern.

In SONET/SDH systems, A1 is defined as 8'hF6 and A2 is defined as 8'h28. 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.
OC-96 systems are for Arria II GZ devices only and have 96 A 1 bytes followed by 96 A2 bytes.

Arria II GX transceivers are designed to support the protocols OC-12 at 622 Mbps with 8-bit channel width and OC-48 at 2488.32 Mbps with 16-bit channel width. Arria II GZ transceivers are designed to support the protocol OC-96 at 4,967 Mbps.

Figure 1–72 shows the transceiver datapath when configured in SONET/SDH OC-12 mode.

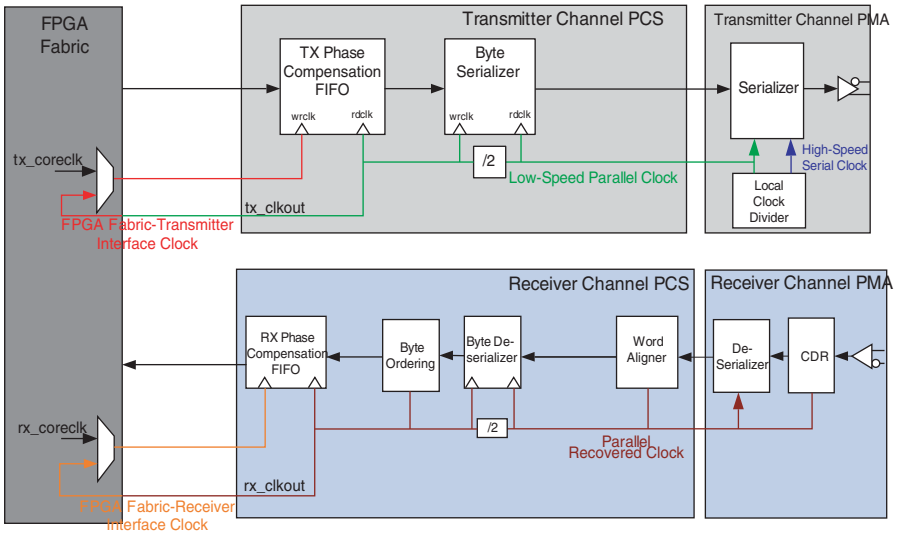
Figure 1–72. SONET/SDH OC-12 Datapath



SONET/SDH OC-48 Datapath

Figure 1–73 shows the transceiver datapath when configured in SONET/SDH OC-48 mode.

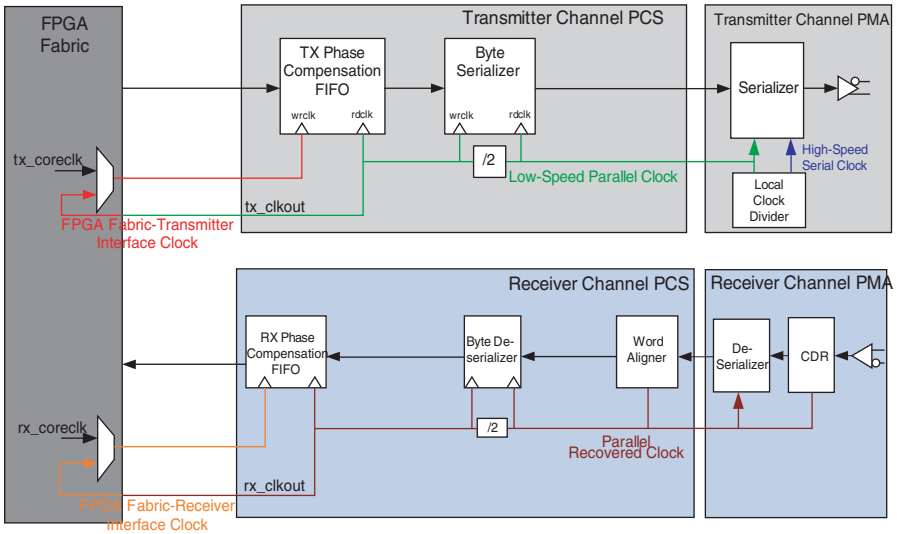
Figure 1–73. SONET/SDH OC-48 Datapath



SONET/SDH OC-96 Datapath

Figure 1–74 shows the transceiver datapath when configured in SONET/SDH OC-96 mode.

Figure 1–74. SONET/SDH OC-96 Datapath



Unlike Ethernet, where the LSB of the parallel data byte is transferred first, SONET/SDH requires the MSB to be transferred first. To facilitate the MSB-to-LSB transfer, you must enable the **Flip Transmitter input data bits** and **Flip Receiver output data bits** options in the ALTGX MegaWizard Plug-In Manager.

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.

Word Aligner in SONET/SDH Mode

The word aligner in SONET/SDH functional mode is configured in manual alignment mode. You can configure the word aligner to either align to a 16-bit A1A2 pattern or a 32-bit A1A1A2A2 pattern, 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 configures the word aligner to align to a 32-bit A1A1A2A2 pattern.

In OC-96 configurations, the word aligner is only allowed to align to an A1A1A2A2 pattern, so the input port `rx_a1a2size` is unavailable. Barring this difference, the OC-96 word alignment operation is similar to that of the OC-12 and OC-48 configurations.

You can also configure the word aligner to flip the word alignment pattern bits programmed in the ALTGX MegaWizard Plug-In Manager and compare them with the incoming data for alignment. This feature offers flexibility to the SONET backplane system for either a MSB-to-LSB or LSB-to-MSB data transfer. Table 1–22 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 1–22. Word Aligner Settings for Arria II Devices

Serial Bit Transmission Order	Flip the Word Alignment Pattern Bits	Word Alignment Pattern
MSB-to-LSB	On	16'hF628
MSB-to-LSB	Off	16'h146F
LSB-to-MSB	Off	16'h28F6

OC-48 and OC-96 Byte Serializer and Deserializer

The OC-48 and OC-96 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 1–14 and "Byte Deserializer" on page 1–43, 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.

The OC-96 byte serializer converts 32-bit data words from the FPGA fabric and translates them into two 16-bit data words at twice the rate. The OC-96 byte deserializer takes in two consecutive 16-bit data words and translates them into a 32-bit data word to the FPGA fabric at half the rate.

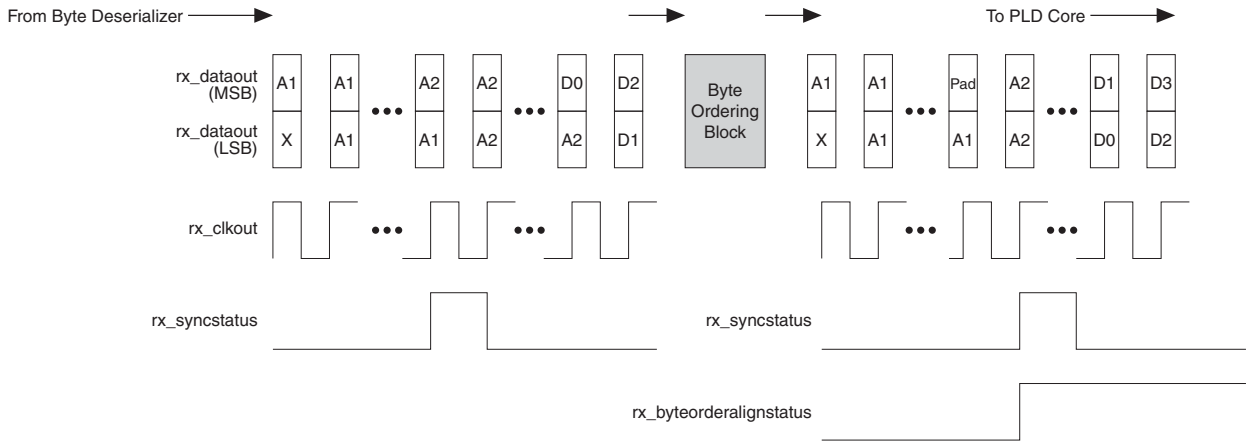
Byte Ordering in SONET/SDH OC-48 Mode

Because of byte deserialization, the MSByte of a word might appear at the `rx_dataout` port along with the LSByte of the next word. In a SONET/SDH OC-48 configuration, you can use the byte ordering block that is built into the datapath to perform byte ordering. Byte ordering in a SONET/SDH OC-48 configuration is in word alignment-based mode, where the byte ordering block is triggered by the rising edge of the `rx_syncstatus` signal.

At the rising edge of the `rx_syncstatus` signal, the byte ordering block compares the LSByte coming out of the byte deserializer with the A2 byte of the A1A2 alignment pattern. If the LSByte 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 shown in Figure 1–75. Inserting 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 1–75. SONET/SDH OC-48 Byte Ordering Example



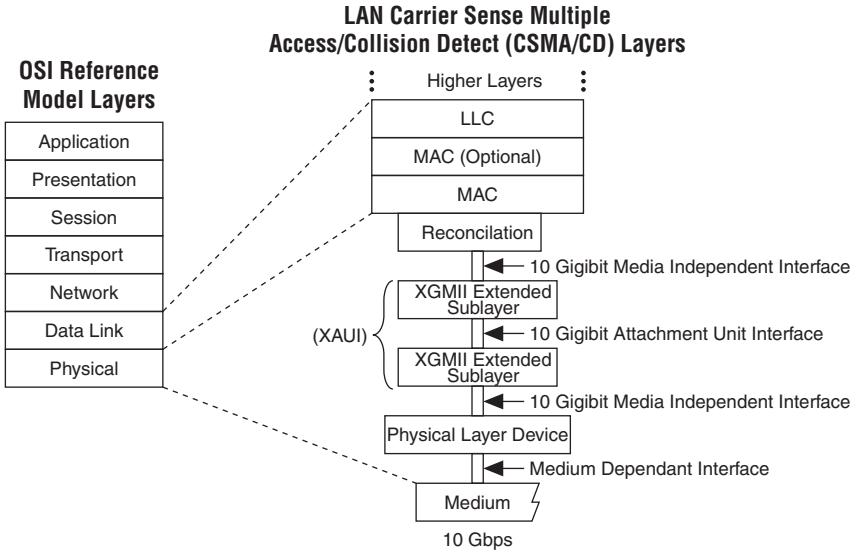
XAUI

Use this functional mode for XAUI, HiGig, or HiGig+ protocols.

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.

Figure 1–76 shows the relationships between the XGMII and XAUI layers.

Figure 1–76. XAUI and XGMII Layers



The XGMII interface consists of four 8-bit lanes. 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.

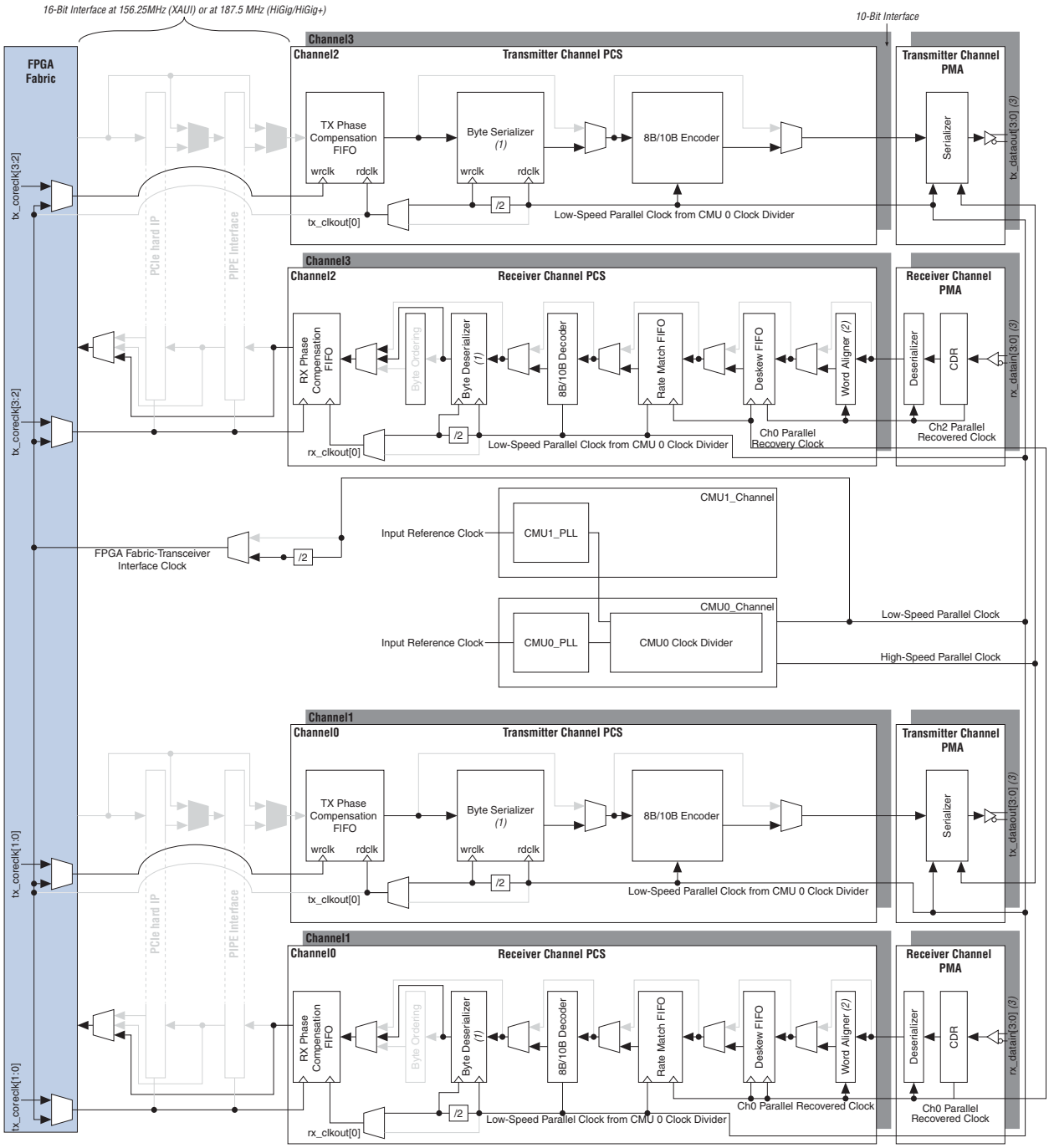
XAUI functions as a self-managed interface because code group synchronization, channel deskew, and clock domain decoupling are handled with no upper layer support requirements. This functionality is based on the PCS code groups that are used during the IPG time and idle periods.

Arria II GX and GZ transceivers configured in XAUI mode provide the following protocol features:

- XGMII-to-PCS code conversion at the transmitter—The 8B/10B encoder in the Arria II GX and GZ 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.
- PCS-to-XGMII code conversion at the receiver—The 8B/10B decoder in the Arria II GX and GZ receiver datapath is controlled by a XAUI receiver state machine that converts received PCS code groups into specific 8-bit XGMII codes.
- 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 1–77 shows the ALTGX megafunction transceiver datapath when configured in XAUI mode.

Figure 1–77. Transceiver Datapath in XAUI Mode



Notes to Figure 1–77:

(1) This allows the fabric-to-transceiver interface to run below the maximum interface frequency.
(2) The word aligner uses the automatic synchronization state machine (10-bit /K28.5/).
(3) This is running at half the rate of the data rate.

Word Aligner in XAUI Mode

The word aligner in XAUI functional mode is configured in automatic synchronization state machine mode that is compliant to the PCS synchronization state diagram specified in section 8 of the IEEE P802.3ae specification. 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.

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. 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 in XAUI Mode

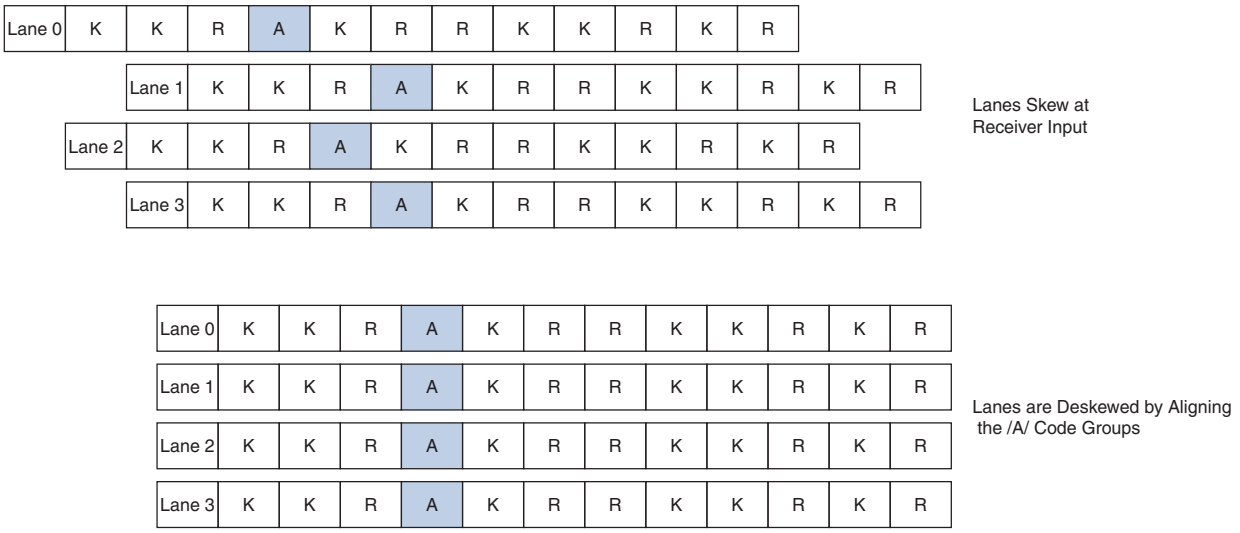
The XAUI protocol requires the physical layer device to implement deskew circuitry to align all four channels. The skew introduced in the physical medium and the receiver channels can cause the /A/ code groups to be received misaligned with respect to each other. To enable the deskew circuitry at the receiver to align the four channels, the transmitter sends an /A/ (/K28.3/) code group simultaneously on all four channels during an inter-packet gap (IPG). 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.



The deskew FIFO operation in XAUI functional mode is compliant to the PCS deskew state machine diagram specified in 8 of the IEEE P802.3ae specification.

Figure 1–78 shows lane skew at the receiver input and how the deskew FIFO uses the /A/ code group to align the channels.

Figure 1–78. Deskew FIFO–Lane Skew at the Receiver Input



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.

Rate Match FIFO in XAUI Mode

In XAUI mode, the rate match FIFO is capable of compensating up to ±100 PPM (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 the ||R|| column) during inter-packet gaps, adhering to rules listed in the IEEE P802.3ae specification.

The rate match operation begins after `rx_syncstatus` and `rx_channelaligned` are asserted. The `rx_syncstatus` signal is from the word aligner, indicating that synchronization is acquired on all four channels; the `rx_channelaligned` signal is from the deskew FIFO, indicating channel alignment.

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 underrunning. The rate match FIFO can insert or delete as many ||R|| columns as necessary to perform the rate match operation.

The `rx_rmfifodatadeleted` and `rx_rmfifodatainserted` flags indicate rate match FIFO deletion and insertion events, respectively, and 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 1–79 shows an example of rate match deletion in the case where three ||R|| columns must be deleted.

Figure 1–79. Example of Rate Match Deletion in XAUI Mode

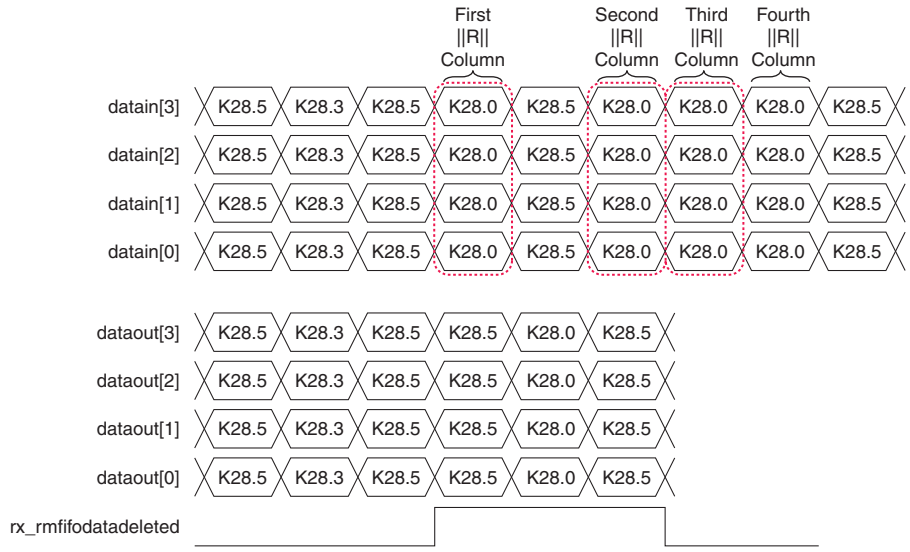
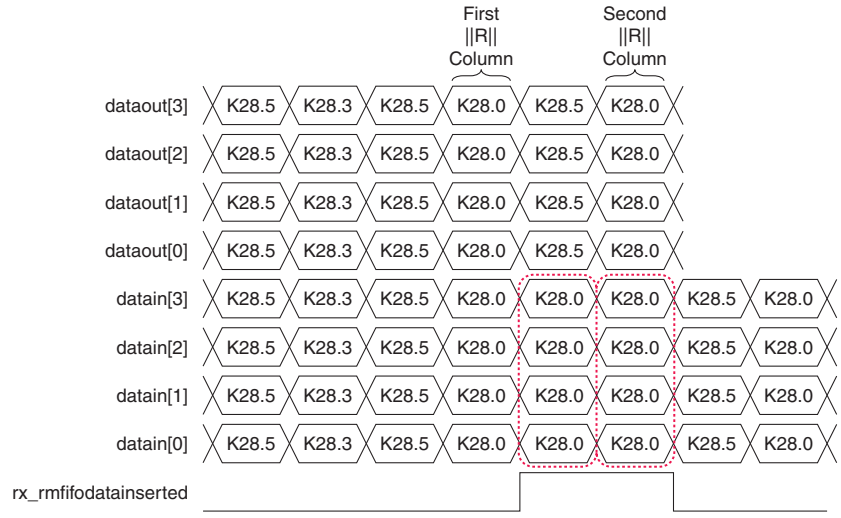


Figure 1–80 shows an example of rate match insertion in the case where two ||R|| columns must be inserted.

Figure 1–80. Example of Rate Match Insertion in XAUI Mode



CAUTION

The rate match FIFO does not insert or delete code groups automatically to overcome FIFO empty or full conditions. In this case, the rate match FIFO 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. You must then assert the `rx_digitalreset` signal to reset the receiver PCS blocks.

Test Modes

Arria II GX and GZ devices provide various loopback options, pattern generators, and verifiers that allow you to ensure the working of different functional blocks in the transceiver channel. These modes include:

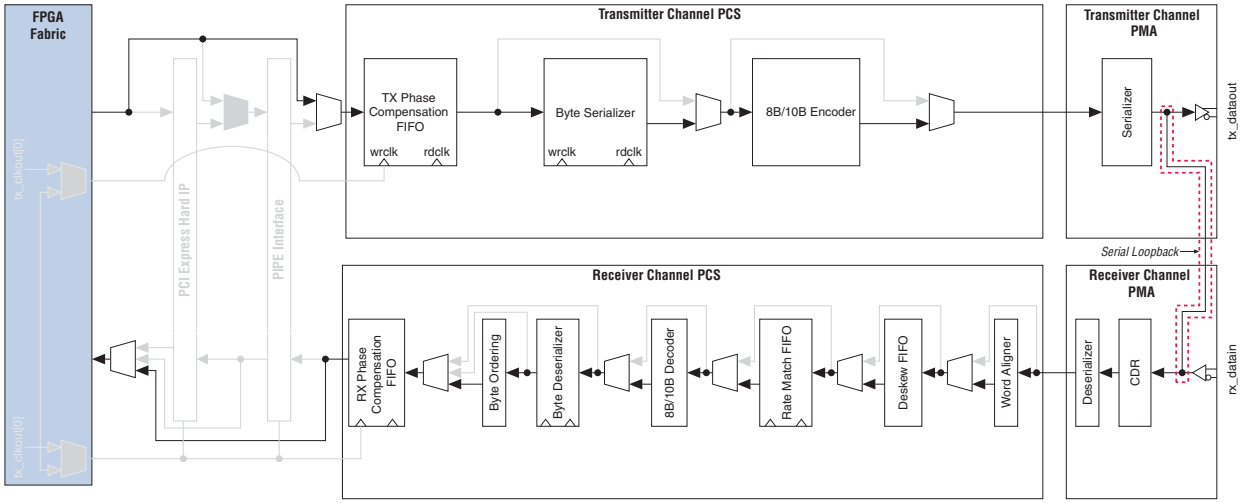
- Serial loopback
- Reverse serial loopback
- Reverse serial pre-CDR loopback
- PCIe reverse parallel loopback
- BIST and PRBS Modes

If you generate a **Transmitter-only** or **Receiver-only** configuration and enable loopback mode, you will have an extra port that you must connect to the transceiver's counterpart port. These ports are described in Table 1–31 on page 1–98 (the PMA port list).

Serial Loopback

This option is available for all functional modes except PCIe mode. Figure 1–81 shows the datapath for serial loopback.

Figure 1–81. Serial Loopback Datapath



The data from the FPGA fabric passes through the transmitter channel and loops back to the receiver channel, bypassing the receiver input buffer. The received data is available to the FPGA logic for verification. Using this option, you can check the operation 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 when the signal is asserted high.

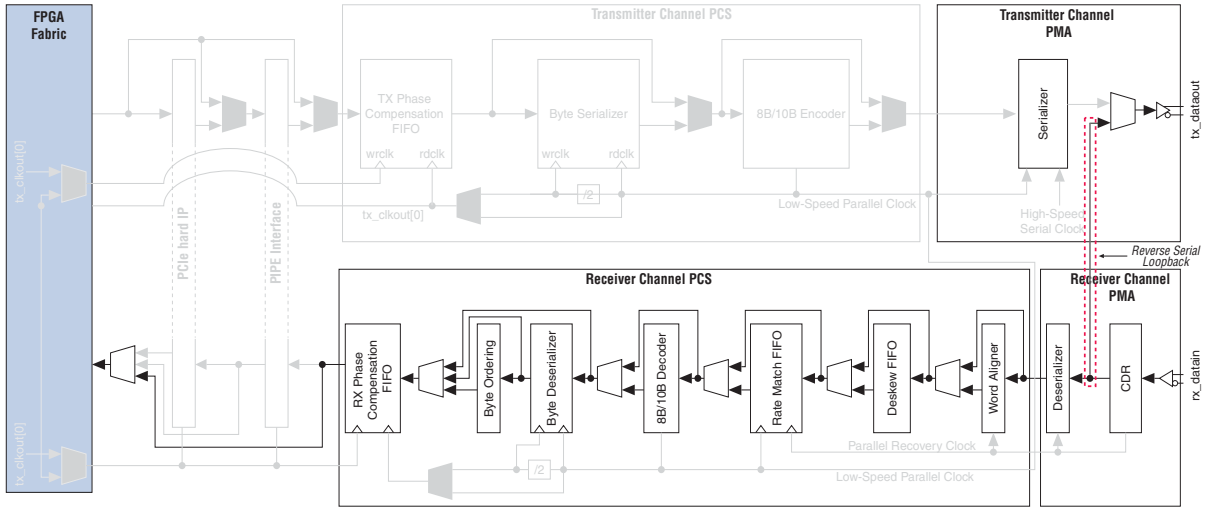
When you enable serial loopback, the transmitter channel sends the data to both the tx_dataout output port and the receiver channel. The differential output voltage on the tx_dataout ports is based on the selected V_{OD} settings. The looped back data is received by the receiver CDR and then timed again through different clock domains. You must provide an alignment pattern for the word aligner to enable the receiver channel to retrieve the byte boundary.

Reverse Serial Loopback

Reverse serial loopback is available in Basic functional mode only and is often implemented when using a bit error rate tester (BERT) on the upstream transmitter. In this mode, the data is received through the rx_datain port, timed again through the receiver CDR, and sent out to the tx_dataout port. The received data is also available to the FPGA logic. You can enable the reverse serial loopback option using the ALTGX MegaWizard Plug-In Manager. Unlike other loopback modes, there is no dynamic pin control to enable or disable reverse serial loopback.

Figure 1–82 shows the transceiver channel datapath for reverse serial loopback mode.

Figure 1–82. Reverse Serial Loopback Datapath ***(Note 1)***



Note to Figure 1–82:

(1) The only active block of the transmitter channel is the transmitter buffer.

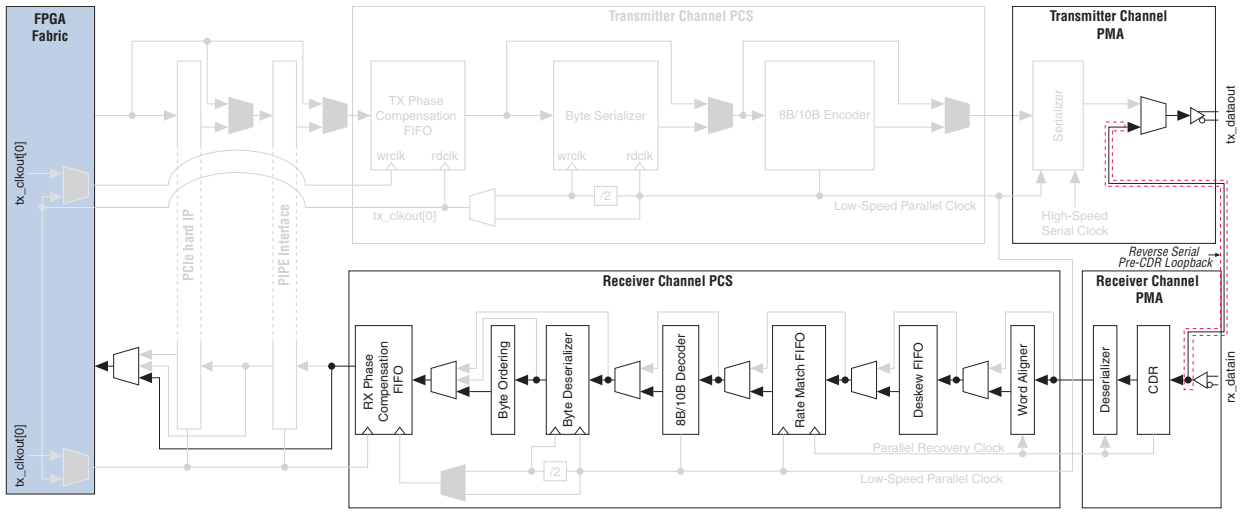
You can change the output differential voltage on the transmitter buffer through the ALTGX MegaWizard Plug-In Manager. However, you cannot alter the pre-emphasis settings for the transmitter buffer.

Reverse Serial Pre-CDR Loopback

Reverse serial pre-CDR loopback is available in Basic functional mode only. In this mode, 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. You can enable the reverse serial pre-CDR loopback option using the ALTGX MegaWizard Plug-In Manager. Unlike other loopback modes, there is no dynamic pin control to enable or disable reverse serial pre-CDR loopback.

Figure 1–83 shows the transceiver channel datapath for reverse serial pre-CDR loopback mode.

Figure 1–83. Reverse Serial Pre-CDR Loopback Datapath ***(Note 1)***



Note to Figure 1–83:

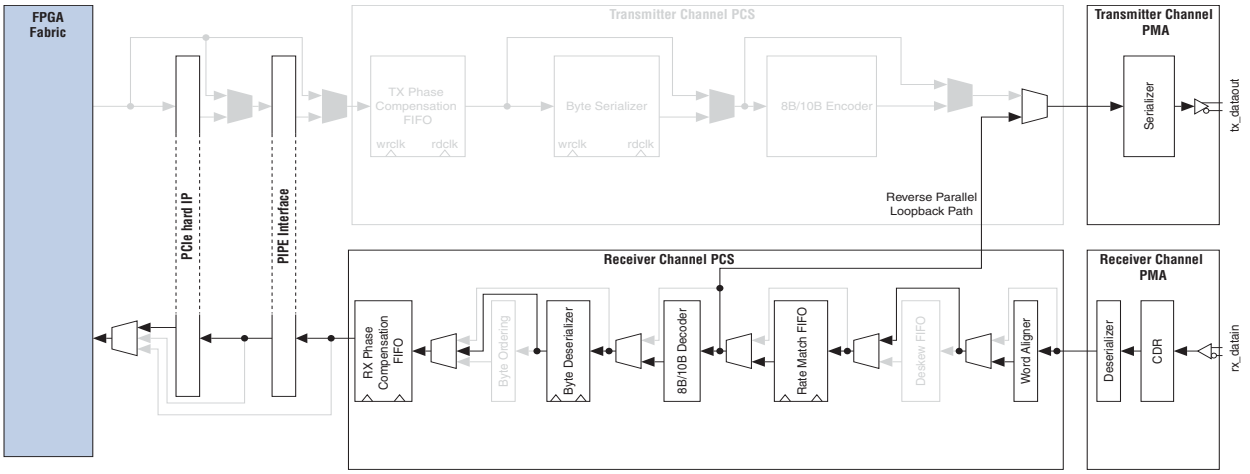
(1) The only active block of the transmitter channel is the transmitter buffer.

You can change the V_{OD} on the transmitter buffer through the ALTGX MegaWizard Plug-In Manager. However, you cannot change the pre-emphasis settings for the transmitter buffer.

PCIe (Reverse Parallel Loopback)

PCIe reverse parallel loopback is only available in PCIe functional mode for the Gen1 and Gen2 data rates. As shown in Figure 1–84, the received serial data passes through the receiver CDR, deserializer, word aligner, and rate match FIFO buffer. The data 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 PCIe Base Specification 2.0. To enable PCIe reverse parallel loopback mode, assert the `tx_detectrxloopback` port.

Figure 1–84. PCIe Reverse Parallel Loopback Mode Datapath

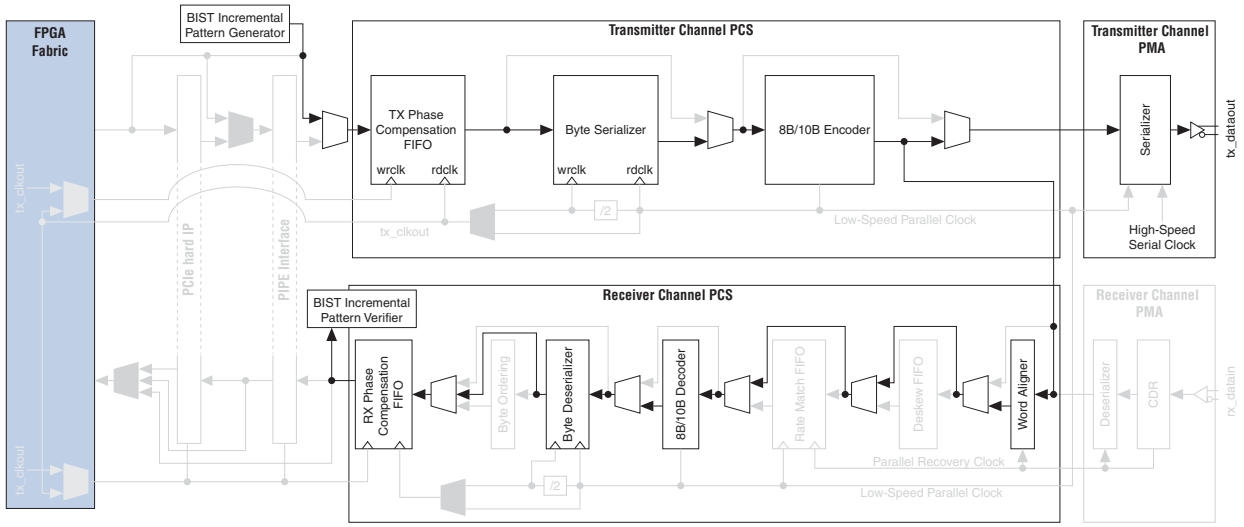


Built-In Self Test (BIST) and Pseudo Random Binary Sequence (PRBS)

Each transceiver channel in Arria II GX and GZ devices contain a pattern generator and a pattern verifier circuit. Using these patterns, you can verify the functionality of the functional blocks in the transceiver channel without requiring user logic. The functionality is provided as an optional mechanism for debugging transceiver channels. To use the Arria II GX and GZ pattern generator and verifier, use the pattern BIST and PRBS sub-protocols under Basic functional mode.

Figure 1–85 shows the datapath for the BIST mode.

Figure 1–85. Enabled PCS Functional Blocks in Parallel Loopback

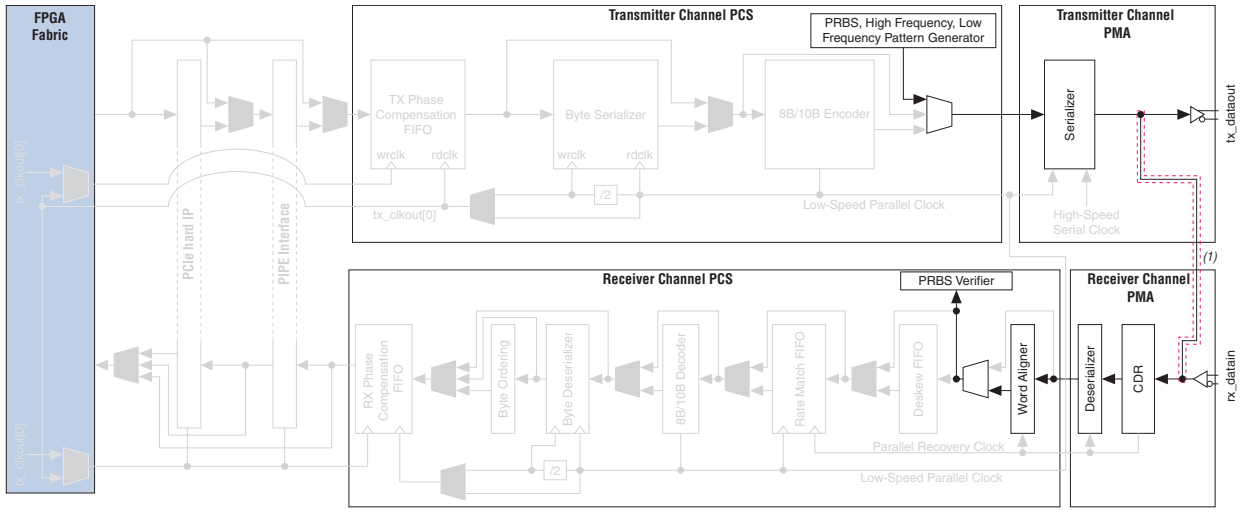


BIST mode allows you to verify the complete PCS blocks for both the transmitter and receiver channel. This mode is available only with a built-in 16-bit incremental pattern generator and verifier; therefore, you must set the channel width to **16 bits** in this mode. The incremental pattern 00-FF is looped back to the receiver channel at the PCS functional block boundary before the PMA and is sent out to the `tx_dataout` port.

The received data is verified by the verifier, but is not available in the FPGA fabric. The V_{OD} of the transmitted serial data on the `tx_dataout` port is based on the selected V_{OD} settings.

Figure 1–86 shows the datapath for the PRBS patterns. The generated pattern is sent to the serializer. The verifier checks the data from the word aligner.

Figure 1–86. Datapath for the PRBS Mode



Note to Figure 1–86:

(1) Serial loopback can be dynamically enabled through the `rx_seriallpbken` port.

PRBS mode has two pattern generator options, selectable in the **BIST** tab of the MegaWizard Plug-In Manager when you choose PRBS as a sub protocol under Basic functional mode.

- PRBS7, PRBS8, PRBS10, and PRBS23 generator and verifier—This is the generator and verifier interface with the serializer and deserializer in the PMA blocks. The advantage of using a PRBS data stream is that the randomness yields an environment that stresses the transmission medium. In the data stream, you can observe both random jitter and deterministic jitter using a time interval analyzer, bit error rate tester, or oscilloscope.

  The PRBS repeats after completing an iteration. The number of bits the PRBSx pattern sends before repeating the pattern is (2^×-1) bits. This mode is available as a sub protocol under Basic functional mode.

- High-frequency and low-frequency pattern generator—The high-frequency patterns generate alternate ones and zeros and the low-frequency patterns generate five ones and five zeroes in single width mode, and ten ones and ten zeroes in double width mode. These patterns do not have a corresponding verifier.

Table 1–23 lists various PRBS patterns and corresponding word alignment patterns.

Table 1–23. Patterns in PRBS Mode for Arria II Devices (Part 1 of 2)

Patterns	Polynomial	Channel Width of 8-Bit *(1)*	Word Alignment Pattern	Maximum Data Rate
PRBS 7	$X^7 + X^6 + 1$	8 bit	16'h3040	2.5
PRBS 8	$X^8 + X^7 + 1$	8 bit	16'hFF5A	2.5
PRBS 10	$X^{10} + X^7 + 1$	10 bit	10'h3FF	3.125
PRBS 23	$X^{23} + X^{18} + 1$	8 bit	16'hFFFF	2.5

Table 1–23. Patterns in PRBS Mode for Arria II Devices (Part 2 of 2)

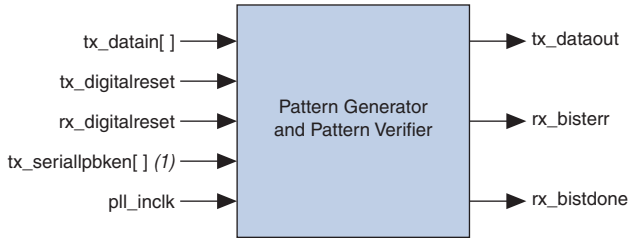
Patterns	Polynomial	Channel Width of 8-Bit *(1)*	Word Alignment Pattern	Maximum Data Rate
High frequency *(1)*	1010101010	8 or 10 bit	NA	2.5 for 8-bit pattern and 3.125 for 10-bit pattern
Low frequency *(1)*	0000011111	10 bit	NA	3.125

Note to Table 1–23:

(1) A verifier is not available for the specified patterns.

Figure 1–87 shows the enabled input and output ports of the pattern generator and pattern verifier.

Figure 1–87. Input and Output Ports for the BIST and PRBS Modes



Note to Figure 1–87:

(1) `rx_serialpbken` is optional.

You can reset the PRBS pattern generator and verifier by asserting the `tx_digitalreset` and `rx_digitalreset` signals, respectively.

`rx_digitalreset` does not reset the BIST output signals when the following conditions are true:

- `pll_powerdown` is high in BIST mode
- `pll_powerdown` or `rx_analogreset` is high in PRBS mode

Dynamic Reconfiguration

Dynamic reconfiguration allows you to reconfigure the transceiver block without reconfiguring the FPGA. For hot-plug 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.

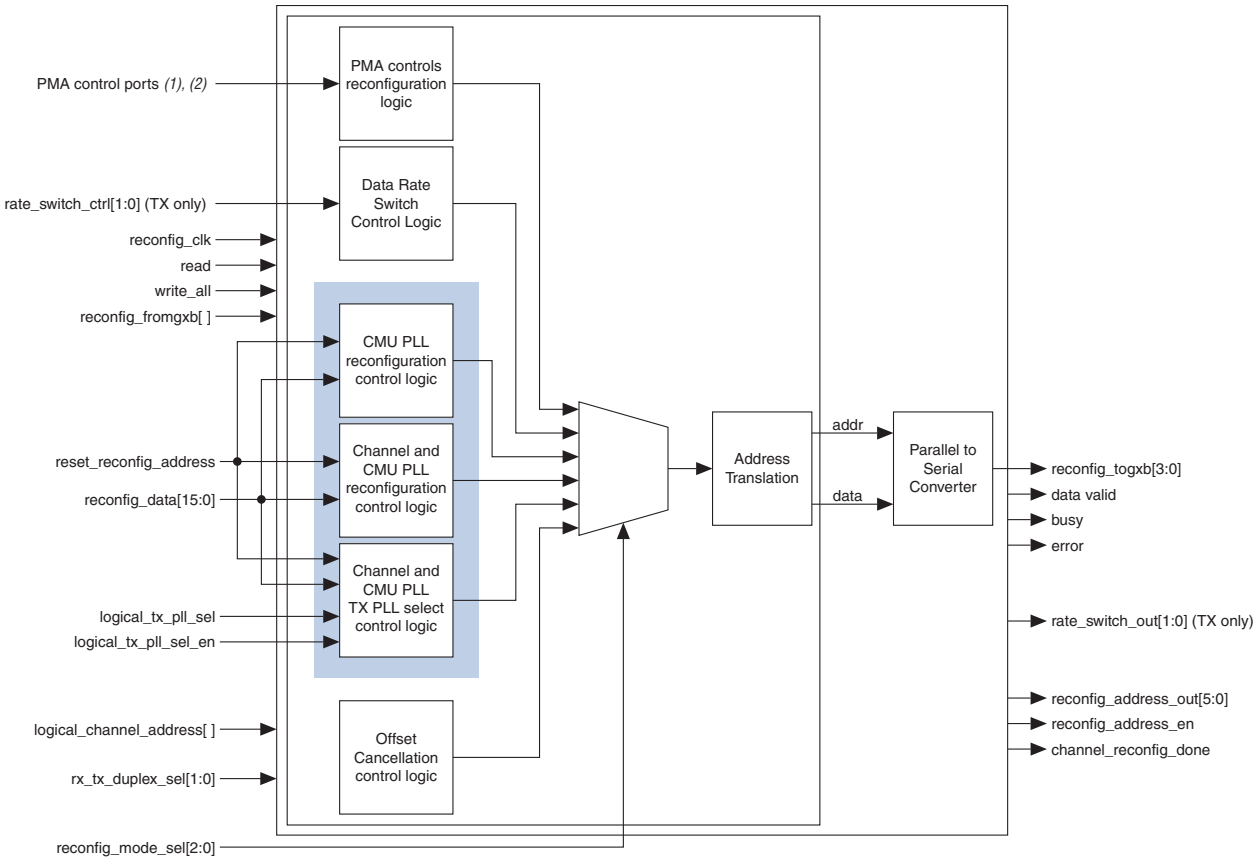
For more information, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II Devices*.

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 the following portions of each transceiver channel dynamically (one channel at a time) without powering down the 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
- Channel and clock multiplier unit PLL
- CMU PLL only

The dynamic reconfiguration controller is a soft IP that uses FPGA-fabric resources. You can use only one dynamic reconfiguration controller per transceiver block. You cannot use the dynamic reconfiguration controller to control multiple Arria II GX and GZ devices or off-chip interfaces. Figure 1–88 shows the conceptual view of the dynamic reconfiguration controller architecture.

Figure 1–88. Block Diagram of the Dynamic Reconfiguration Controller



Notes to Figure 1–88:

(1) The PMA control ports consist of the V_{OD} controls, pre-emphasis controls, DC gain controls, and manual equalization controls.

(2) Only PMA reconfiguration mode supports manual equalization controls.

The dynamic reconfiguration controller requires input from one of the following:

- Its input ports through user logic where they are translated to the address and data bus inside the controller. The address and data bus are then converted into serial data and forwarded to the selected transceiver channel
- A Memory Initialization File (**.mif**) where the controller receives 16-bit words from the **.mif** that you generate and sends this information to the transceiver channel selected

The different modes of dynamic reconfiguration are:

- PMA settings reconfiguration, available for the following PMA settings:
  - Pre-emphasis settings
  - Equalization settings (channel reconfiguration mode does not support equalization settings)
  - DC gain settings
  - V_{OD} settings
- Receiver offset cancellation

  Process variations create offsets in analog circuit voltages, pushing them outside the expected range. The Arria II GX and GZ devices provide 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 input buffer and receiver CDR require offset calibration. Offset cancellation is automatically executed whenever the device is powered on. The control logic for offset cancellation is integrated into the dynamic reconfiguration controller.

  The offset cancellation for the receiver channels option is automatically enabled in both the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers for **Receiver**, **Transmitter**, and **Receiver only** configurations. It is not available for **Transmitter only** configurations.

  
  When offset cancellation is automatically enabled, you must instantiate the dynamic reconfiguration controller to connect the reconfiguration ports created by the ALTGX MegaWizard Plug-In Manager.

  
  For more information about implementing the ALTGX_RECONFIG MegaWizard Plug-In Manager, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II Devices*.

  You must always connect the ALTGX_RECONFIG instance to the ALTGX (with receiver channels) instance in your design. Connect the `reconfig_fromgxb`, `reconfig_togxb`, and necessary clock signals to both the ALTGX_RECONFIG and ALTGX (with receiver channels) instances. For transmitter-only configuration, the ALTGX_RECONFIG must also be connected to the ALTGX either by selecting a dynamic reconfiguration mode or by instantiating a dummy receiver-only ALTGX instance in each side of the device.

  
  The offset cancellation process changes the transceiver reset sequence. For more information, refer to the *Reset Control and Power Down* chapter.

- Transceiver channel reconfiguration—for transceiver channels, dynamic reconfiguration involves the reconfiguration of the following:
  - Data Rate Reconfiguration—achievable by switching between two TX PLLs set to different data rates or reconfiguring the RX PLLS or reconfiguring the local dividers in the transmit side

    Ensure that the functional mode of the transceiver channel supports the reconfigured data rate.

  - CMU PLL Reconfiguration
  - Functional Mode Reconfiguration

    Ensure that the various clocks involved support the transition.

Transceiver Port List

You instantiate the Arria II GX and GZ transceivers with 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.

These signals are available if you enable the block associated with them.

Table 1–24 lists the CMU port names and descriptions for the ALTGX megafunction.

Table 1–24. ALTGX Megafunction CMU Ports for Arria II Devices

Port Name	Input/Output	Description
pll_inclk	Input	Input reference clock for the CMU PLL.
pll_powerdown	Input	Asynchronous active-high signal to power down both CMU PLLs. The minimum pulse width for this signal is specified in the *Device Datasheet for Arria II Devices* chapter. Note: Asserting the pll_powerdown signal does not power down the refclk buffers. Note: While each CMU PLL has its own pll_powerdown port, the ALTGX MegaWizard Plug-In Manager instantiation provides only one port per transceiver block. This port power downs one or both CMU PLLs (if used).
coreclkout	Output	A low-speed parallel clock generated by the CMU0 clock divider for bonded channel configurations. This signal is generated by the CMU0 clock divider in the master transceiver block in ×8 bonded channel configurations and is not available in non-bonded channel configurations. If the byte serializer block is enabled in bonded channel modes, the coreclkout clock output is half the frequency of the low-speed parallel clock. Otherwise, the coreclkout clock output is the same frequency as the low-speed parallel clock. You can also use this clock on the write and read clock ports of the TX phase compensation FIFOs in all bonded channels if tx_coreclk is not enabled in the ALTGX MegaWizard Plug-In Manager.
pll_locked	Output	Asynchronous active-high signal to indicate whether the CMU PLL is locked.

Table 1–25 lists the word aligner port names and descriptions for the ALTGX megafunction.

Table 1–25. ALTGX Megafunction Word Aligner Ports for Arria II Devices (Part 1 of 2)

Port Name	Input/Output	Description
rx_ala2size	Input	Available only in SONET OC-12 and OC-48 modes to select between one of the following two word alignment options: Logic Level — Word Alignment Pattern 0 — 16-bit A1A2 1 — 32'bit A1A1A2A2
rx_bitslip	Input	Asynchronous bit-slip control when the word aligner is configured in bit-slip mode. At every rising edge of this signal, the word aligner slips one bit into the received data stream, effectively shifting the word boundary by one bit. The minimum pulse-width is two recovered clock cycles.
rx_enapatternalign	Input	Asynchronous manual word alignment enable control. This signal is edge-sensitive with 8-bit width data and level sensitive with 10-bit width data. The minimum pulse-width is two recovered clock cycles.
rx_invpolarity	Input	Asynchronous receiver polarity inversion control. When asserted high, the polarity of every bit of the 8-bit or 10-bit input data word to the word aligner is inverted.
rx_revbitorderwa	Input	Asynchronous receiver bit reversal control. Available only in Basic mode with the word aligner configured in bit-slip mode. When asserted high in Basic mode, the 8-bit or 10-bit data D[7:0] or D[9:0] at the output of the word aligner is rewired to D[0:7] or D[0:9], respectively.
rx_bitslipboundary selectout	Output	Asynchronous signal indicating the number of bits slipped in the word aligner when the word aligner is configured in manual mode.
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 width 8/10 — 1 16/20 — 2

Table 1–25. ALTGX Megafunction Word Aligner Ports for Arria II Devices (Part 2 of 2)

Port Name	Input/Output	Description
rx_rlv	Output	Asynchronous 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. This signal is driven for a minimum of two recovered clock cycles in configurations without byte serializer and a minimum of three recovered clock cycles in configurations with byte serializer.
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. For word aligner in manual alignment mode, this signal is driven high for one parallel clock cycle synchronous to the MSByte of the word alignment pattern. 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 width 8/10 — 1 16/20 — 2

Table 1–26 lists the deskew FIFO port name and description for the ALTGX megafunction.

Table 1–26. ALTGX Megafunction Deskew FIFO Port for Arria II Devices

Port Name	Input/Output	Description
rx_channelaligned	Output	Indicates whether all the channels are aligned. This signal is only available in XAUI mode. A high level indicates that the XAUI deskew state machine is either in a 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 a 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.

Table 1–27 lists the rate match (clock rate compensation) FIFO port names and descriptions for the ALTGX megafunction.

Table 1–27. ALTGX Megafunction Rate Match (Clock Rate Compensation) FIFO Ports for Arria II Devices (Part 1 of 2)

Port Name	Input/Output	Description
rx_rmfifodatadeleted	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.
rx_rmfifodatainserted	Output	Rate match FIFO insertion status indicator. A high level indicates that the rate match pattern byte was inserted to compensate for the PPM difference in reference clock frequencies between the upstream transmitter and the local receiver.

Table 1–27. ALTGX Megafunction Rate Match (Clock Rate Compensation) FIFO Ports for Arria II Devices (Part 2 of 2)

Port Name	Input/Output	Description
rx_rmfifoempty	Output	Asynchronous rate match FIFO empty status indicator. A high level indicates that the rate match FIFO is empty. This signal is driven for a minimum of two recovered clock cycles in configurations without byte serializer and a minimum of three recovered clock cycles in configurations with byte serializer. You must then assert the `rx_digitalreset` signal to reset this signal.
rx_rmfifofull	Output	Asynchronous rate match FIFO full status indicator. A high level indicates that the rate match FIFO is full. This signal is driven for a minimum of two recovered clock cycles in configurations without byte serializer and a minimum of three recovered clock cycles in configurations with byte serializer. You must then assert the `rx_digitalreset` signal to reset this signal.

Table 1–28 lists the 8B/10B decoder port names and descriptions for the ALTGX megafunction. These ports are 1-bit wide with 8-bit channel width and 2-bit wide with 16-bit channel width.

Table 1–28. ALTGX Megafunction 8B/10B Decoder Ports for Arria II Devices

Port Name	Input/Output	Description
`rx_ctrldetect`	Output	Receiver control code indicator. 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.
rx_disperr	Output	8B/10B disparity error indicator port. A high level indicates that a disparity error was detected on the associated received code group.
rx_errdetect	Output	8B/10B code group violation or disparity error indicator. 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
rx_runningdisp	Output	8B/10B running disparity indicator. 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.

Table 1–29 lists the byte ordering block port names and descriptions for the ALTGX megafunction.

Table 1–29. ALTGX Megafunction Byte Ordering Block Ports for Arria II GX and GZ Devices

Port Name	Input/Output	Description
rx_enabyteord	Input	Asynchronous enable byte ordering control. 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.
rx_byteorderalignstatus	Output	Byte ordering status indicator. 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.

Table 1–30 lists the RX phase compensation FIFO port names and descriptions for the ALTGX megafunction.

Table 1–30. ALTGX Megafunction RX Phase Compensation FIFO Ports for Arria II Devices

Port Name	Input/Output	Description
coreclkout	Input	Clock from the CMU0 block of the associated transceiver block or the master transceiver block for ×4 bonded or ×8 bonded channel configurations, respectively. This is the default read and write clocks for those configurations.
rx_coreclk	Input	Optional read clock port for the RX phase compensation FIFO. If not enabled, the Quartus II software automatically selects rx_clkout/tx_clkout/coreclkout as the read clock for the RX phase compensation FIFO. If selected, you must drive this port with a clock that has 0 PPM difference with respect to the FIFO write clock.
rx_clkout	Input	Recovered clock from the receiver channel. This is the default read and write clocks for the RX phase compensation FIFO in non-bonded configurations without the rate-match FIFO.
tx_clkout	Input	Clock from the transmitter channel local clock divider. This is the default read and write clocks for the RX phase compensation FIFO in non-bonded configurations with the rate-match 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_phase_comp_ fifo_error	Output	RX phase compensation FIFO full or empty indicator. A high level indicates that the RX phase compensation FIFO is either full or empty.

Table 1–31 lists the receiver physical medium attachment (PMA) port names and descriptions for the ALTGX megafunction.

Table 1–31. ALTGX Megafunction Receiver PMA Ports for Arria II Devices (Part 1 of 2)

Port Name	Input/Output	Description
rx_cruclk	Input	Input reference clock for the receiver CDR.
rx_datain	Input	Receiver serial data input port.
rx_locktodata	Input	Asynchronous receiver CDR LTD mode control signal. When asserted high, the receiver CDR is forced to LTD mode. When de-asserted low, the receiver CDR lock mode depends on the rx_locktorefclk signal level.

Table 1–31. ALTGX Megafunction Receiver PMA Ports for Arria II Devices (Part 2 of 2)

Port Name	Input/Output	Description
rx_locktorefclk	Input	Asynchronous receiver CDR LTR mode control signal. The rx_locktorefclk and rx_locktodata signals control whether the receiver CDR is in LTR or LTD 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
rx_seriallpbken	Input	Active-high serial loopback control port.
rx_seriallpbkin	Input	Input on a **Receiver-only** configuration when you enable serial loopback. You must connect this port to the tx_seriallpbkout port.
tx_revseriallpbkin	Input	Input on a **Transmitter-only** configuration when you enable reverse serial loopback. You must connect this port to the rx_revseriallpbkout port.
rx_freqlocked	Output	Asynchronous receiver CDR lock mode indicator. A high level indicates that the receiver CDR is in LTD mode. A low level indicates that the receiver CDR is in LTR mode.
rx_pll_locked	Output	Asynchronous active high receiver CDR LTR indicator. The receiver CDR is locked to the input reference clock.
rx_revseriallpbkout	Output	Output on a **Receiver-only** configuration when you enable reverse serial loopback. You must connect this port to the tx_revseriallpbkin port.
tx_seriallpbkout	Output	Output on a **Transmitter-only** configuration when you enable serial loopback. You must connect this port to the tx_seriallpbkout port.
rx_signaldetect	Output	Asynchronous signal threshold detect indicator for PCIe mode only. 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 PCIe mode, the rx_signaldetect signal is inverted and driven on the pipeelecidle port.

Table 1–32 lists the TX phase compensation FIFO port names and descriptions for the ALTGX megafunction.

Table 1–32. ALTGX Megafunction TX Phase Compensation FIFO Ports for Arria II Devices

Port Name	Input/Output	Description
tx_coreclk	Input	Optional write clock port for the TX phase compensation FIFO. If enabled, you must drive this port with a clock that is frequency locked to tx_clkout/coreclkout (with 0 PPM frequency difference).
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.
tx_clkout	Input	FPGA fabric-transceiver interface clock. Each channel has a tx_clkout signal in non-bonded channel configurations.
tx_phase_comp_fifo_error	Output	TX phase compensation FIFO full or empty indicator. A high level indicates that the TX phase compensation FIFO is either full or empty.
coreclkout	Input	Clock from the CMU0 block of the associated transceiver block or of the master transceiver block for ×4 bonded or ×8 bonded channel configurations, respectively. This is the default read and write clocks for those configurations.

Table 1–33 lists the 8B/10B encoder port names and descriptions for the ALTGX megafunction.

Table 1–33. ALTGX Megafunction 8B/10B Encoder Ports for Arria II Devices

Port Name	Input/Output	Description
tx_bitslipboundaryselect	Input	Indicates the number of bits to slip at the transmitter for word alignment at the receiver.
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
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
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
tx_invpolarity	Input	Asynchronous 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 is inverted.

Table 1–34 lists the transmitter PMA port names and descriptions for the ALTGX megafunction.

Table 1–34. ALTGX Megafunction Transmitter PMA Ports for Arria II Devices

Port Name	Input/Output	Description
fixedclk	Input	125-MHz clock for receiver detect and offset cancellation in PCIe mode.
tx_dataout	Output	Transmitter serial data output port.

Table 1–35 lists the reconfiguration block port names and descriptions for the ALTGX megafunction.

For more information about these ports, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II Devices*.

Table 1–35. ALTGX Megafunction Reconfiguration Block Ports for Arria II Devices

Port Name	Input/Output	Description
reconfig_clk	Input	Dynamic reconfiguration clock. This clock is also used for offset cancellation in all modes except PCIe mode.
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.
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.

Table 1–36 lists the PIPE interface port names and descriptions for the ALTGX megafunction.

Table 1–36. ALTGX Megafunction PIPE Interface Ports for Arria II Devices (Available only in PCIe functional mode) (Part 1 of 2)

Port Name	Input/Output	Description
pipe8b10binvpolarity	Input	PCIe polarity inversion control. Functionally equivalent to the RxPolarity signal defined in PIPE specification revision 2.00. Available only in PCIe mode. When asserted high, the polarity of every bit of the 10-bit input data to the 8B/10B decoder is inverted.
powerdn	Input	PCIe 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
tx_detectrxloopback	Input	Receiver detect or PCIe loopback control. Functionally equivalent to the TxDetectRx/Loopback signal defined in PIPE specification revision 2.00. When asserted high in the P1 power state with the tx_forceelecidle signal asserted, the transmitter buffer begins the receiver detection operation. When the receiver detect completion is indicated on the pipephydonestatus port, this signal must be de-asserted. When asserted high in the P0 power state with the tx_forceelecidle signal de-asserted, the transceiver datapath is dynamically configured to support parallel loopback, as described in "PCIe (Reverse Parallel Loopback)" on page 1–88.
tx_forcedispcompliance	Input	Forces the 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 PCIe protocol.

Table 1–36. ALTGX Megafunction PIPE Interface Ports for Arria II Devices (Available only in PCIe functional mode) (Part 2 of 2)

Port Name	Input/Output	Description
tx_forceelecidle	Input	Force transmitter buffer to PCIe electrical idle signal levels. Functionally equivalent to the `TxElecIdle` signal defined in PIPE specification revision 2.00.
pipeelecidle	Output	Asynchronous signal to indicate whether electrical idle is detected or inferred at the receiver. Functionally equivalent to the `RxElecIdle` signal defined in PIPE specification revision 2.00. If you enable the electrical idle inference block, it drives this signal high when it infers an electrical idle condition, as described in "Electrical Idle Inference" on page 1–70. 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 input buffer is inverted and driven on this port.
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.
pipestatus	Output	PCIe 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
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.

Table 1–37 lists the reset and power down port names and descriptions for the ALTGX megafunction.

For more information, refer to the *Reset Control and Power Down in Arria II Devices* chapter.

Table 1–37. ALTGX Megafunction Reset and Power Down Ports for Arria II Devices

Port Name	Input/Output	Description
`gxb_powerdown`	Input	Asynchronous transceiver block power down signal. When asserted high, all digital and analog circuitry in the PCS, PMA, CMU of the transceiver block is powered down, except for the refclk buffers.
rx_analogreset	Input	Active-high receiver PMA reset.
rx_digitalreset	Input	Active-high receiver PCS reset.
tx_digitalreset	Input	Active-high transmitter PCS reset.

Table 1–38 lists the calibration block port names and descriptions for the ALTGX megafunction.

Table 1–38. ALTGX Megafunction Calibration Block Ports for Arria II Devices

Port Name	Input/Output	Description
cal_blk_clk	Input	Clock for transceiver calibration blocks.
cal_blk_powerdown	Input	Calibration block power down control.

Table 1–39 lists the verifier port names and descriptions for the ALTGX megafunction.

Table 1–39. ALTGX Megafunction Verifier Ports for Arria II Devices

Port Name	Input/Output	Description
rx_bisterr	output	For BIST mode, the rx_bisterr signal asserts and stays high when the verifier detects an error. For PRBS mode, the rx_bisterr signal assets and stays high for a minimum of three rx_clkout clock cycles when the verifier detects an error and de-asserts if the following PRBS sequence has no error.
rx_bistdone	output	For BIST mode, the rx_bistdone port asserts and stays high when the verifier either receives one full cycle of incremental pattern or detects an error in the receiver data. For PRBS mode, the rx_bistdone port asserts high and stays high when the verifier a full cycle of PRBS pattern.

Document Revision History

Table 1–40 lists the revision history for this chapter.

Table 1–40. Document Revision History (Part 1 of 2)

Date	Version	Changes
October 2013	4.4	■ Updated the "Dynamic Reconfiguration" section.
July 2012	4.3	■ Removed Fiber Channel in Table 1–2. ■ Finalized information in Table 1–4, Table 1–14, and Table 1–15. ■ Removed OC-3 (155 Mbps) from Table 1–1 and Table 1–2.
December 2011	4.2	■ Updated Table 1–1. ■ Added Table 1–15. ■ Updated Figure 1–55. ■ Minor text edits.
June 2011	4.1	■ Added Table 1–38. ■ Updated Figure 1–1, Figure 1–2, Figure 1–3, Figure 1–24, Figure 1–35, Figure 1–36, Figure 1–49, Figure 1–53, Figure 1–54, Figure 1–87 and Figure 1–88. ■ Updated Table 1–2, Table 1–6, and Table 1–7. ■ Updated the "Transmitter Output Buffer", "Programmable Differential OCT", "Dynamic Reconfiguration", "PCIe Hard IP Block", and "GIGE" sections. ■ Minor text edits.

Table 1–40. Document Revision History (Part 2 of 2)

Date	Version	Changes
December 2010	4.0	■ Updated to add Arria II GZ information. ■ Updated Figure 1–7. ■ Updated Table 1–20. ■ Updated the "Programmable Equalization, DC Gain, and Offset Cancellation" section. ■ Minor text edits.
July 2010	3.0	■ Updated Figure 1–1, Figure 1–4, Figure 1–45, Figure 1–74, Figure 1–76, Figure 1–48, Figure 1–49, and Figure 1–50. ■ Updated the "Transceiver Block Overview", "Programmable Equalization, DC gain, and Offset Cancellation", "TX Phase Compensation FIFO", "Transmitter Output Buffer", "Functional Modes", "PCIe Mode" "Reverse Serial Loopback", "Reverse Serial Pre-CDR Loopback", and "Dynamic Reconfiguration" sections. ■ Moved Table 1-17 to the Arria II Device Family Datasheet. ■ Converted protocol information to Table 1–1. ■ Minor text edits. For example, change "PCI Express (PIPE)" to "PCIe" and "8B10B" to "8B/10B".
November 2009	2.1	■ Updated figures. ■ Updated Base Specification references to 2.0. ■ Removed table 1-4 from 2.0 version and referenced Arria II GX Device Data Sheet
June 2009	2.0	■ Reorganized. ■ Added "Deterministic Latency" on page 1–44 and "Built-In Self Test (BIST) and Pseudo Random Binary Sequence (PRBS)" on page 1–77. ■ Updated all figures. ■ Port list tables were updated.
March 2009	1.1	Updated: ■ Dynamic Reconfiguration Controller Architecture ■ All AN 558: Implementing dynamic Reconfiguration in Arria II GX Devices ■ Transceiver Channel Reconfiguration ■ Table 1.2 and Table 1.4 ■ Figure 1.69 and Figure 1.70 Added: ■ Offset Cancellation in the Receiver Buffer and Receiver CDR ■ Basic Double-Width Mode Configurations
February 2009	1.0	Initial release.

2. Transceiver Clocking in Arria II Devices

AIIGX52002-3.1

This chapter describes the Arria® II GX and GZ transceiver clocking architecture, including the input reference clocking, transceiver channel datapath clocking, FPGA fabric-transceiver interface clocking, and FPGA fabric phase-locked loop (PLL)-transceiver PLL cascading.

This chapter includes the following sections:

- "CMU PLL and Receiver CDR Input Reference Clocking"
- "Transceiver Channel Datapath Clocking" on page 2–6
- "FPGA Fabric-Transceiver Interface Clocking" on page 2–28
- "FPGA Fabric PLL-Transceiver PLL Cascading" on page 2–56
- "Using the CMU PLL for Clocking User Logic in the FPGA Fabric" on page 2–66

CMU PLL and Receiver CDR Input Reference Clocking

Each transceiver block in the Arria II GX and GZ device contains the following:

- Two clock multiplier unit (CMU) PLLs (`CMU0 PLL` and `CMU1 PLL`)
- Four clock data recovery (CDR) units, one in each receiver channel

The CMU PLLs and receiver CDRs require an input reference clock to operate. The CMU PLL synthesizes the input reference clock to generate the high-speed serial clock used in the transmitter physical media attachment (PMA). The receiver CDR uses the input reference clock as a training clock when it is in lock-to-reference (LTR) mode.

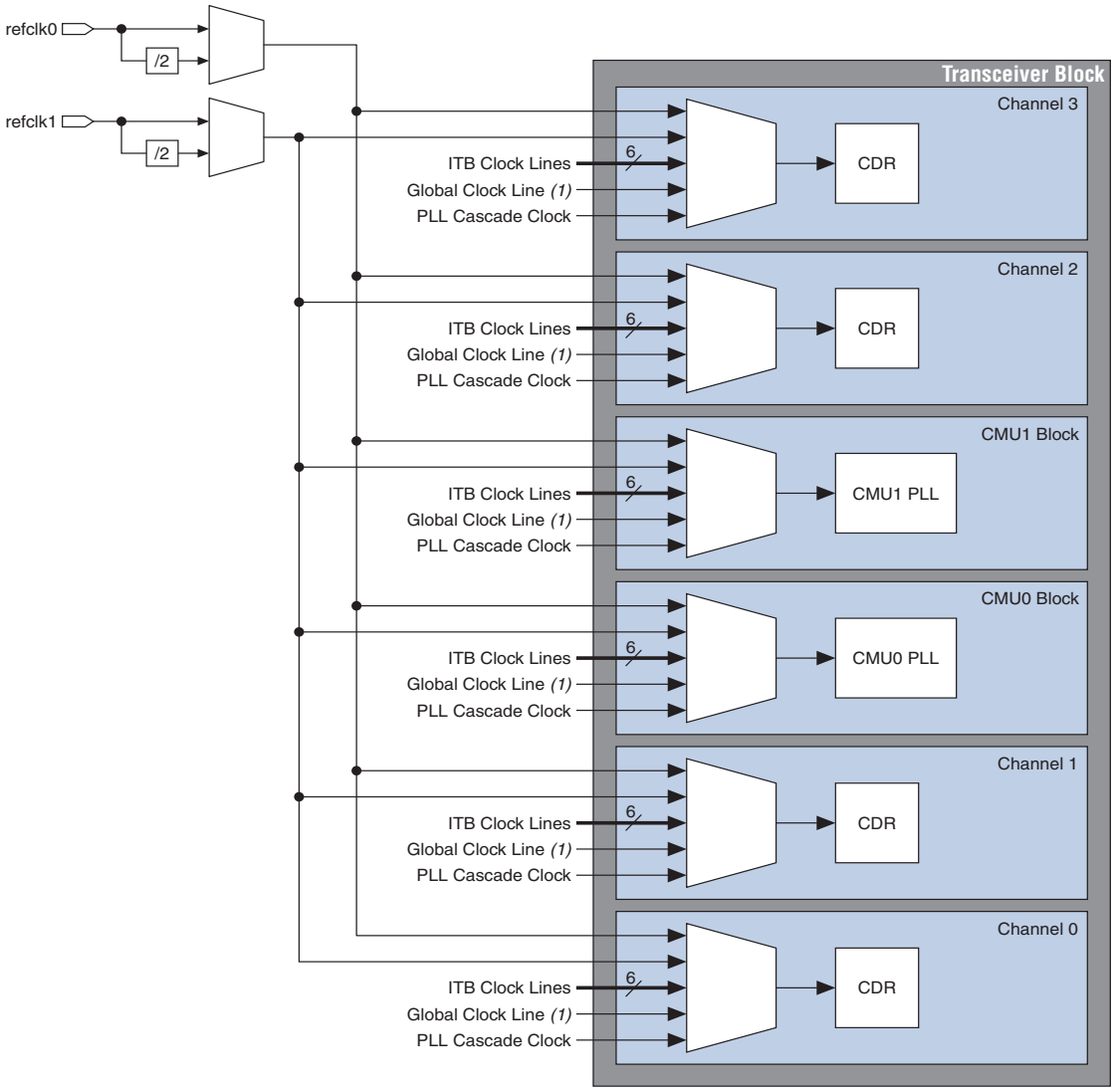
The CMU PLLs and receiver CDRs in each transceiver block can derive input reference from one of the following sources:

- `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 side and right side PLLs in the FPGA fabric

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

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

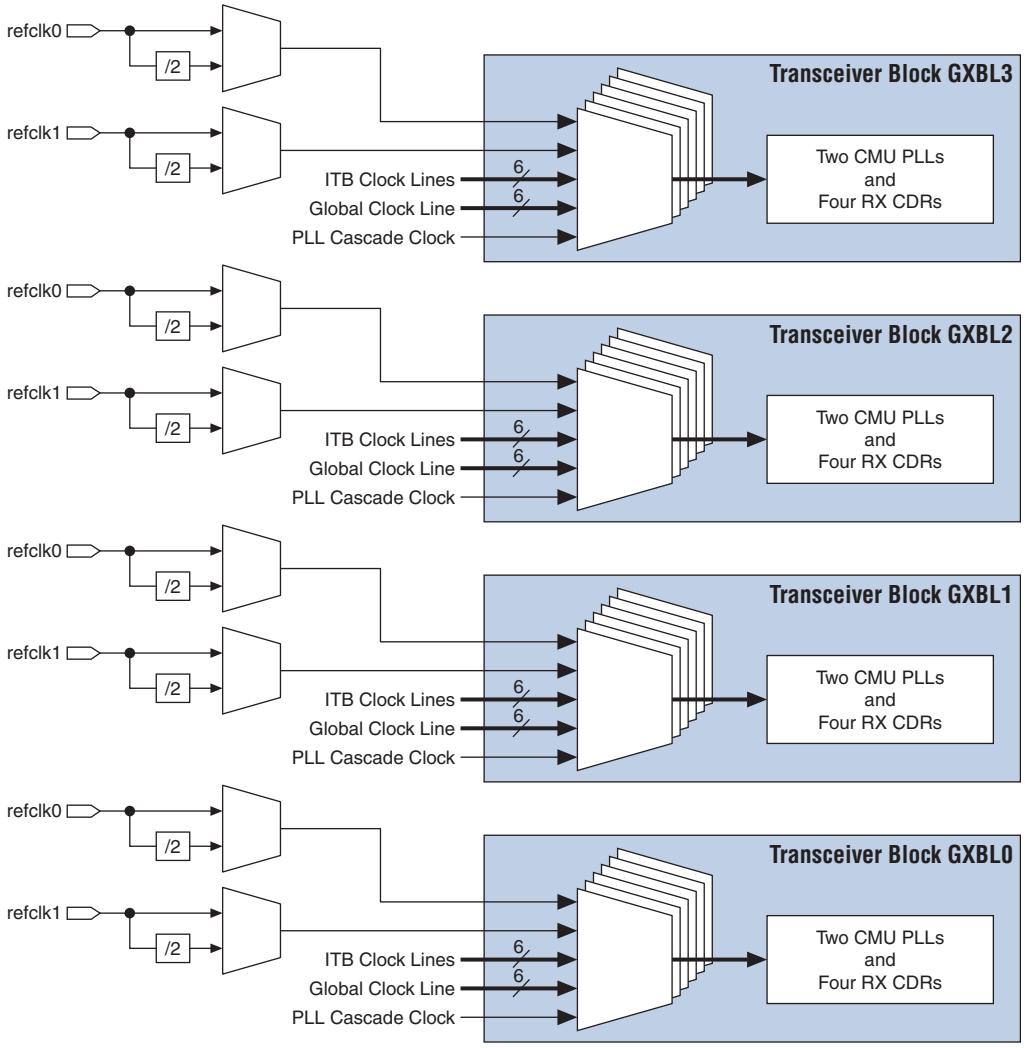


Note to Figure 2–1:

(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 clock, 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 2–1 lists the electrical specifications for the input reference clock signal driven on the refclk pins.

Table 2–1. Electrical Specifications for the Input Reference Clock for Arria II Devices

Protocol	I/O Standard	Coupling	Termination
■ Gigabit Ethernet (GbE) ■ XAUI ■ Serial RapidIO® (SRIO) ■ 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

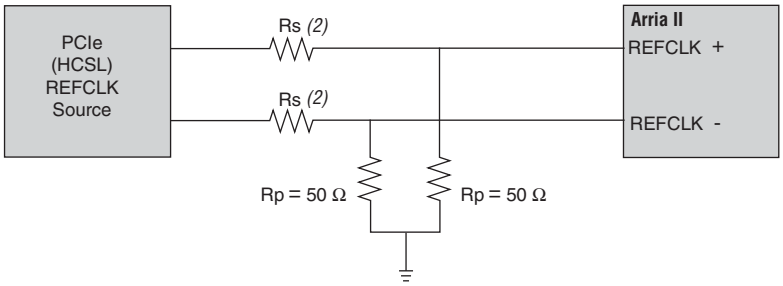
Notes to Table 2–1:

(1) In PCI Express® (PIPE) (PCIe) mode, you have the option of selecting the HCSL standard for the reference clock if compliance to the PCIe protocol is required. The Quartus® II software automatically selects DC coupling with external termination for the refclk pins signal if configured as HCSL.

(2) For an example termination scheme, refer to Figure 2–3.

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)*



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 PCIe specification.

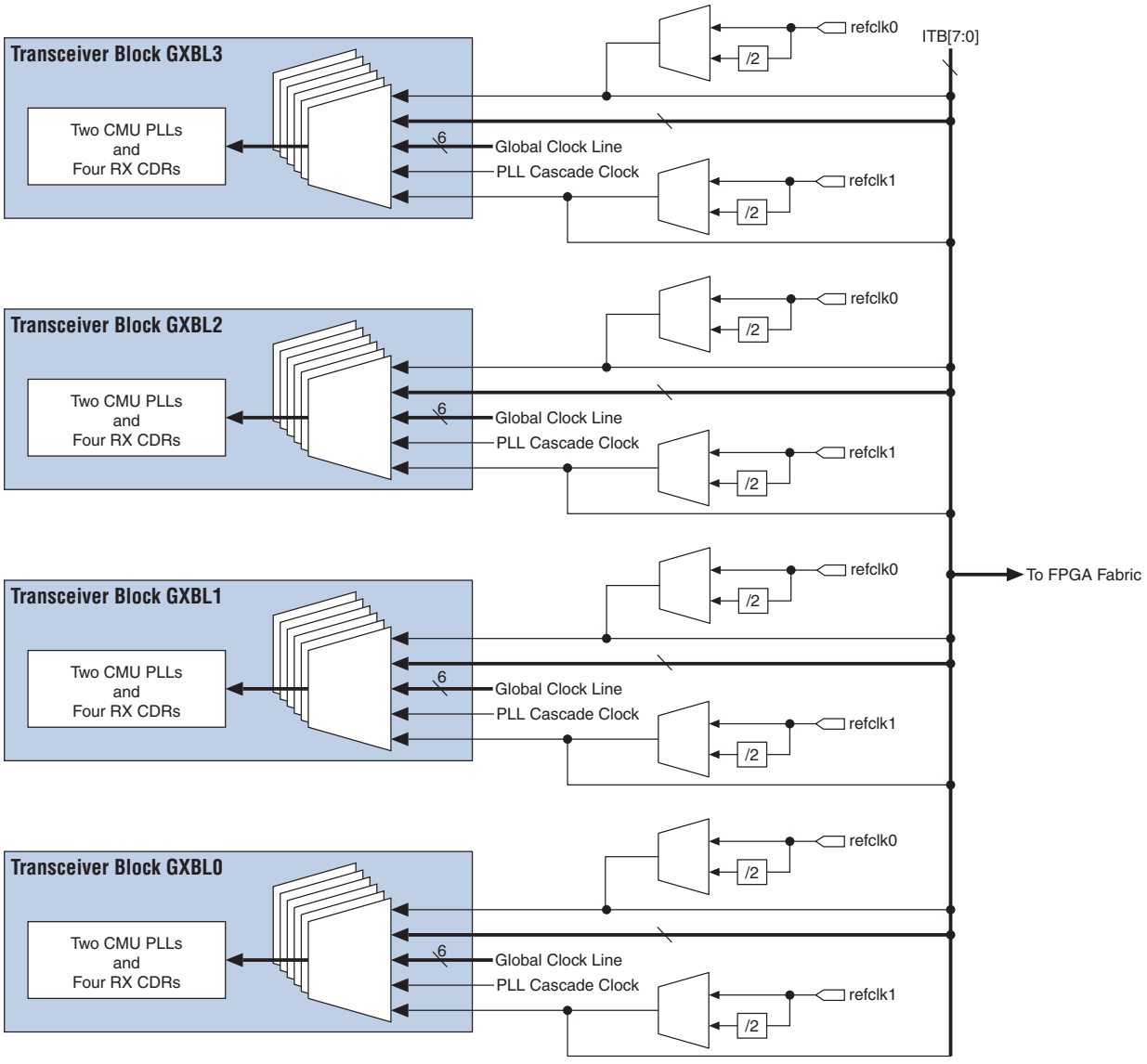
(2) Select resistor values as recommended by the PCIe clock source vendor.

Inter-Transceiver Block 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 the 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 or GZ 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 and GZ 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 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 Devices* chapter.

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 and Right PLLs in the FPGA Fabric

You can use the synthesized clock output from one of the left or right PLLs to provide the input reference clock to the CMU PLLs and receiver CDRs. Arria II GX devices provide a dedicated clock path from the left PLLs (PLL_L1, PLL_L2, PLL_L3, and PLL_L4) in the FPGA fabric to the PLL cascade network located on the left side of the device.

Arria II GZ devices also provide a dedicated clock path from the right PLLs (PLL_R1, PLL_R2, PLL_R3, and PLL_R4) in the FPGA fabric to the PLL cascade network located on the right side of the device. The additional clock multiplication factors available in the left and right PLLs allow more options for on-board crystal oscillator frequencies. For more information, refer to "FPGA Fabric PLL-Transceiver PLL 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 and 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 provide low channel-to-channel skew when compared with 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, resulting 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 (refer to 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 provide lower channel-to-channel skew in bonded channel configurations.

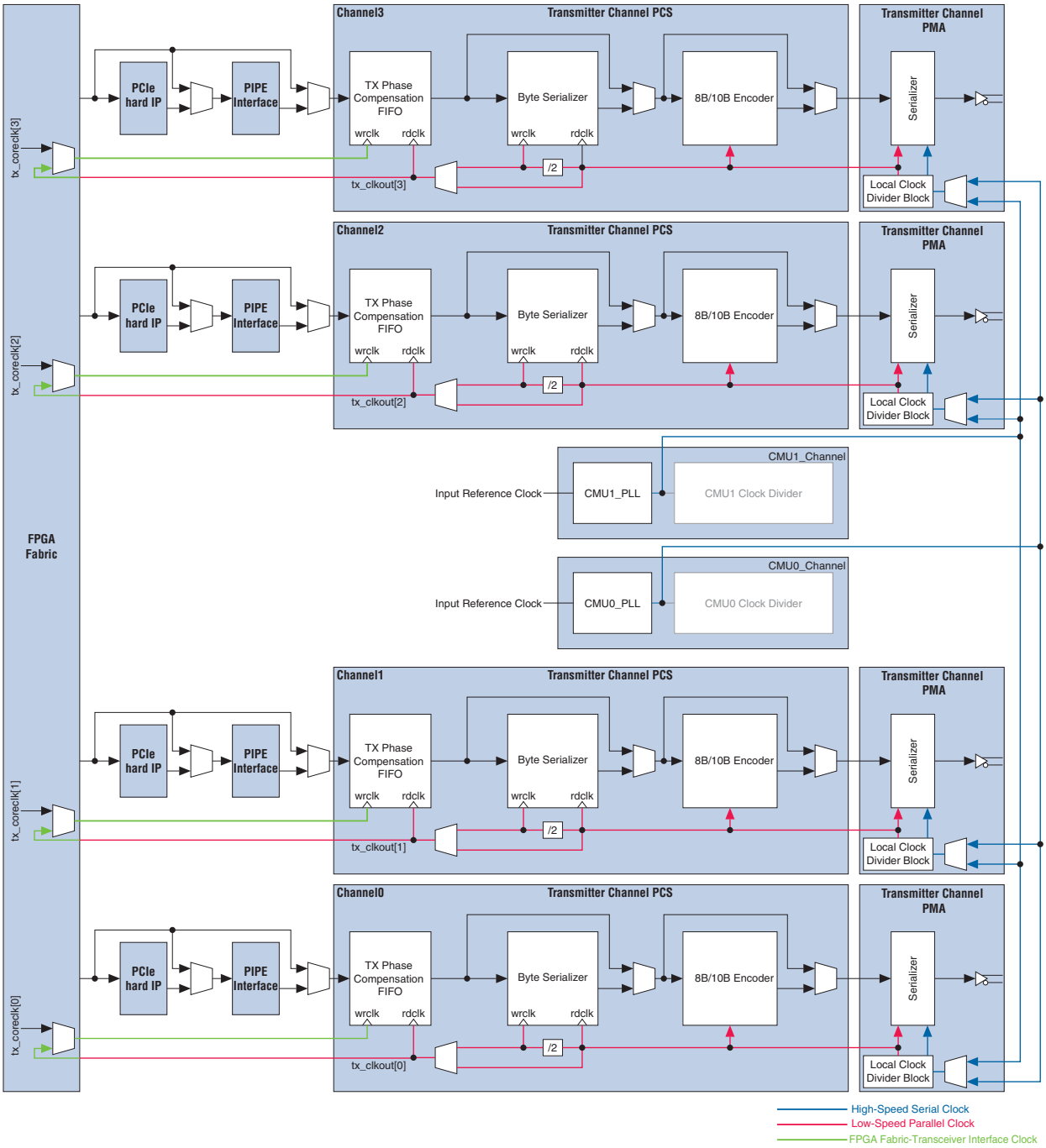
Non-Bonded Channel Configurations

The following functional modes support non-bonded transmitter channel configuration:

- PCIe x1—Gen1 and Gen2 (Gen2 for Arria II GZ only)
- GbE
- SRIO
- SONET/SDH
- SDI
- Common Public Radio Interface (CPRI)/OBSAI
- Basic (except Basic x4 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 a 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 a 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 clock transmitter data and control logic in the FPGA fabric.

Table 2–2 lists 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 for Arria II Devices

Functional Mode *(2)*	Data Rate	High-Speed Serial Clock Frequency	Low-Speed Parallel Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer (MHz)	With Byte Serializer (MHz)
PCIe x1 (Gen 1)	2.5 Gbps	1.25 GHz	250	250 *(1)*	125
PCIe x1 (Gen 2)	5 Gbps	2.5 GHz	500	—	250
GbE	1.25 Gbps	625 MHz	125	125	—
SRIO	1.25 Gbps	625 MHz	125	—	62.5
	2.5 Gbps	1.25 GHz	250	—	125
	3.125 Gbps	1.5625 GHz	312.5	—	156.25
SONET/SDH OC12	622 Mbps	311 MHz	77.75	77.75	—
SONET/SDH OC48	2.488 Gbps	1.244 GHz	311	—	155.5
HD-SDI	1.485 Gbps	742.5 MHz	148.5	148.5	74.25
	1.4835 Gbps	741.75 MHz	148.35	148.35	74.175
3G-SDI	2.97 Gbps	1.485 GHz	297	—	148.5
	2.967 Gbps	1.4835 GHz	296.7	—	148.35

Notes to Table 2–2:

(1) 250 MHz when you enable the PCIe hard IP.

(2) Altera also supports CPRI and OBSAI. For more information, refer to *AN 610: Implementing CPRI and OBSAI Protocols in Altera Devices*.

Bonded Channel Configurations

Arria II GX and GZ devices support x4 PCS and PMA channel bonding that allows bonding of four channels within the same transceiver block. These devices also support x8 channel bonding in PCIe and Basic modes that allows bonding of eight PCS and PMA channels across two transceiver blocks on the same side of the device. Arria II GX and GZ devices with at least two transceiver blocks support x8 bonding.

Bonding is not supported on the receive side for Basic x4 and Basic x8 functional modes. If you use rate matcher, the clocking scheme for Basic x4 and Basic x8 functional modes, the clocking is similar to PCIe x4 mode, as shown in Figure 2–6 on page 2–11 and PCIe x8 mode, as shown in Figure 2–7 on page 2–14.

×4 Bonded Channel Configurations

The following functional modes support x4 bonded transmitter channel configuration:

- PCIe x4—Gen1 and Gen2 (Gen2 for Arria II GZ only)
- XAUI
- Basic x4

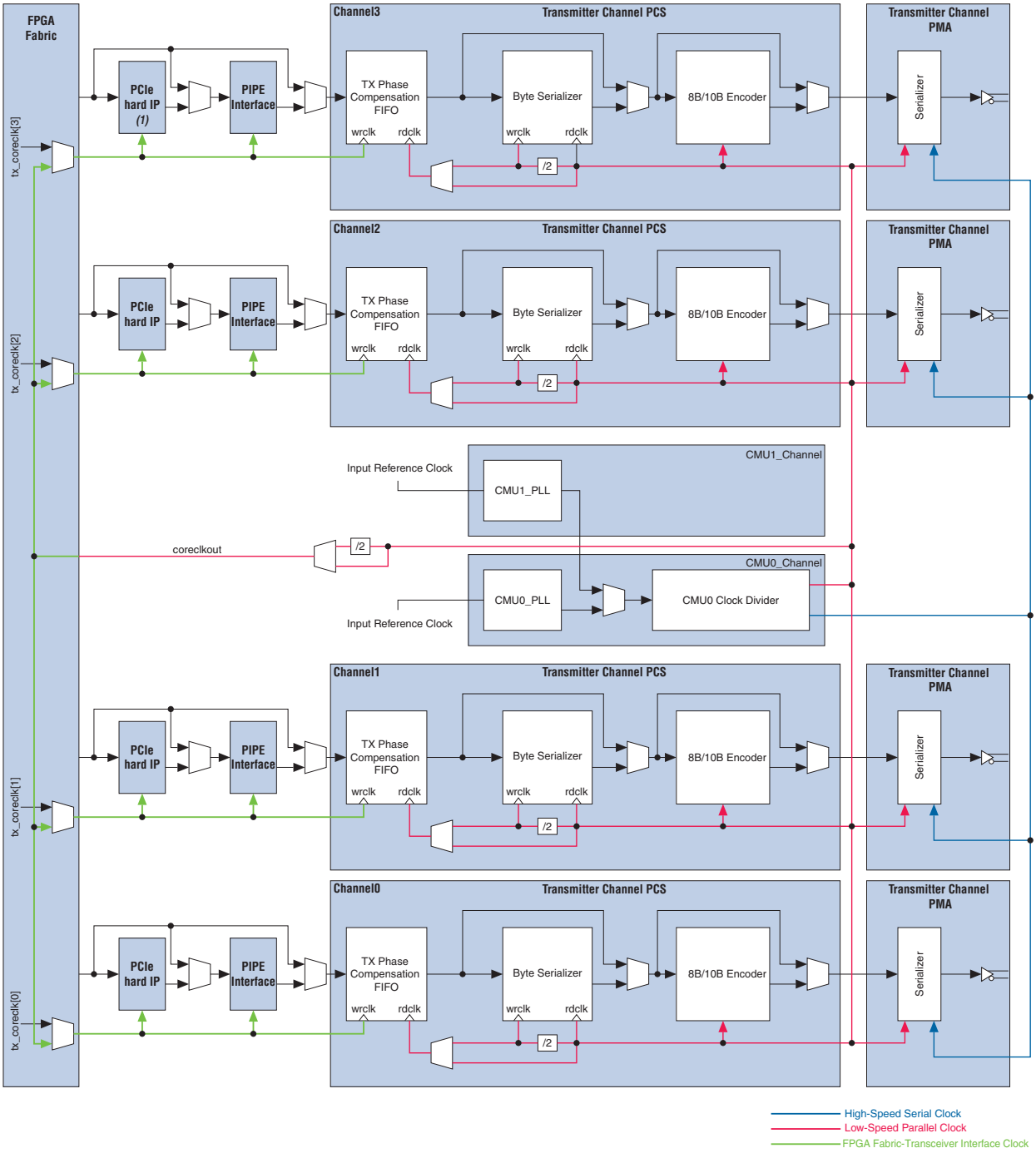
In x4 bonded channel configurations, the receiver datapath clocking varies depending on whether the configured functional mode uses the deskew FIFO or not.

Figure 2–6 shows the transmitter channel datapath clocking in x4 channel bonding configurations.

The Quartus II Compiler generates an error if you do not make the following assignments:

- `tx_dataout[0]` of the x4 bonded link (XAUI or PCIe x4) 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 x4 Bonded Configurations



Note to Figure 2–6:

(1) In Arria II GX and GZ devices, there is only one dedicated PCIe hard IP, which supports PCIe Gen 1 x1, x4, and x8; and PCIe Gen 2 x1 and x4.

In x4 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 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 a 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 a 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 x4 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, ensuring equal transmitter phase compensation FIFO latency across all four bonded channels, resulting in low transmitter channel-to-channel skew.

Table 2–3 lists the transmitter datapath clock frequencies in x4 bonded functional modes that have a fixed data rate.

Table 2–3. Transmitter Datapath Clock Frequencies in x4 Bonded Functional Modes for Arria II Devices

Functional Mode *(1)*	Data Rate (Gbps)	High-Speed Serial Clock Frequency (GHz)	Low-Speed Parallel Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer	With Byte Serializer (MHz)
PCIe x4 (Gen 1)	2.5	1.25	250	— *(2)*	125
PCIe x4 (Gen 2)	5	2.5	500	—	250
XAUI	3.125	1.5625	312.5	—	156.25

Notes to Table 2–3:

(1) Altera also supports CPRI x4 and OBSAI x4. For more information, refer to *AN 610: Implementing CPRI and OBSAI Protocols in Altera Devices*.

(2) 250 MHz when you enable the PCIe hard IP.

x8 Bonded Channel Configuration

The PCIe x8 and Basic x8 functional modes support x8 bonded channel configuration in Arria II GX and GZ devices with 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 of the master transceiver block for their parallel-in, serial-out operation. The low-speed parallel clock from the CMU0 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.

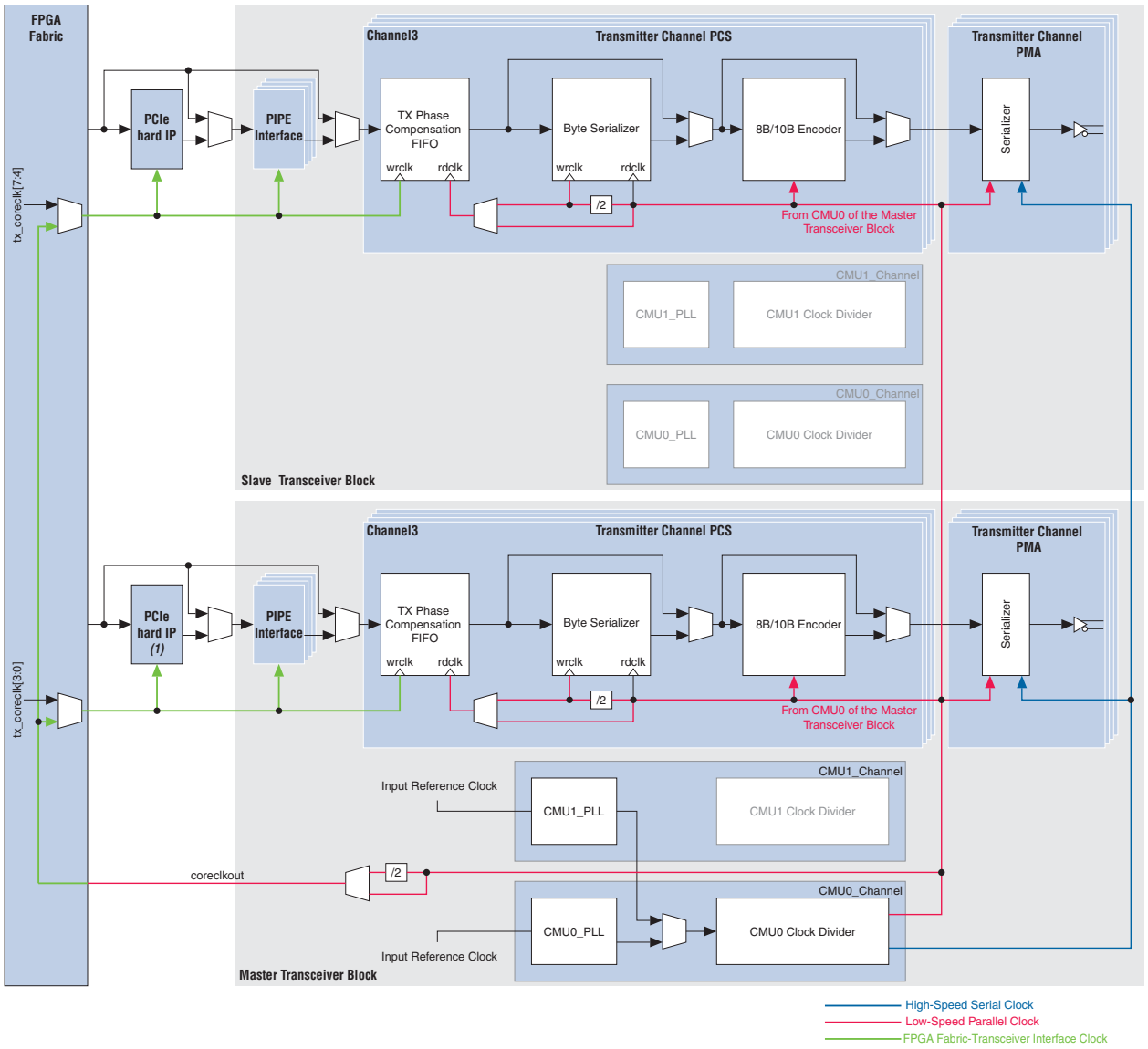
For an 8-bit FPGA fabric-transceiver channel interface that does not use the byte serializer, the low-speed parallel clock from the CMU0 clock divider block in the master transceiver block clocks the read port of the transmitter phase compensation FIFO in all eight bonded channels. This low-speed parallel 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 a 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 x8 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, ensuring 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 PCIe x8 channel bonding configurations.

Figure 2–7. Transmitter Datapath Clocking in a x8 Bonded Configuration



Note to Figure 2–7:

(1) In Arria II GX and GZ devices, there is only one dedicated PCIe hard IP, which supports PCIe Gen 1 x1, x4, and x8; and PCIe Gen 2 (Arria II GZ only) x1 and x4.

Figure 2–8 through Figure 2–10 show allowed master and slave transceiver block locations and PCIe logical lane-to-physical transceiver channel mapping in all Arria II GX and GZ devices.

The Quartus II Compiler generates an error if you do not map the PCIe logical lanes to the physical transceiver channels, as shown in Figure 2–8 through Figure 2–10 on page 2–16.

Figure 2–8 shows the PCIe x8 link in two transceiver block Arria II GX devices.

Figure 2–8. One PCIe x8 Link in Two Transceiver Block Arria II GX Devices

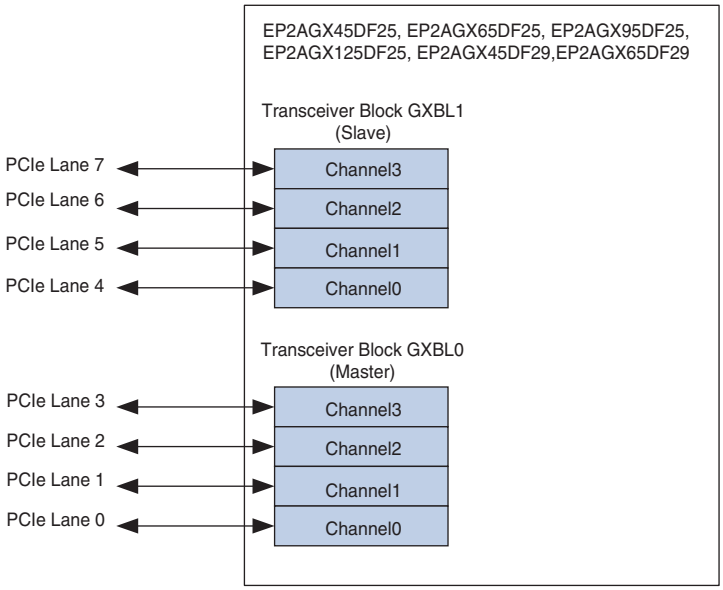
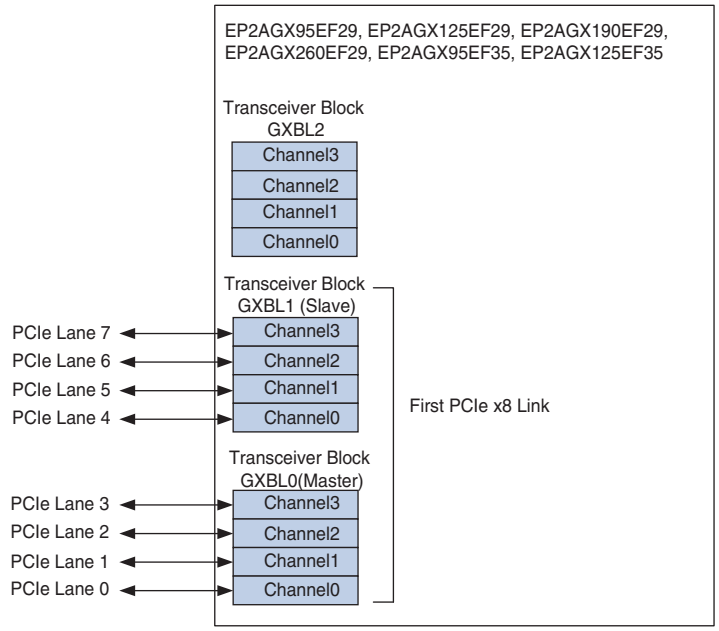


Figure 2–9 shows the PCIe x8 link in three transceiver block devices for Arria II GX devices.

Figure 2–9. One PCIe x8 Link in Three Transceiver Block Arria II GX Devices *(Note 1)*

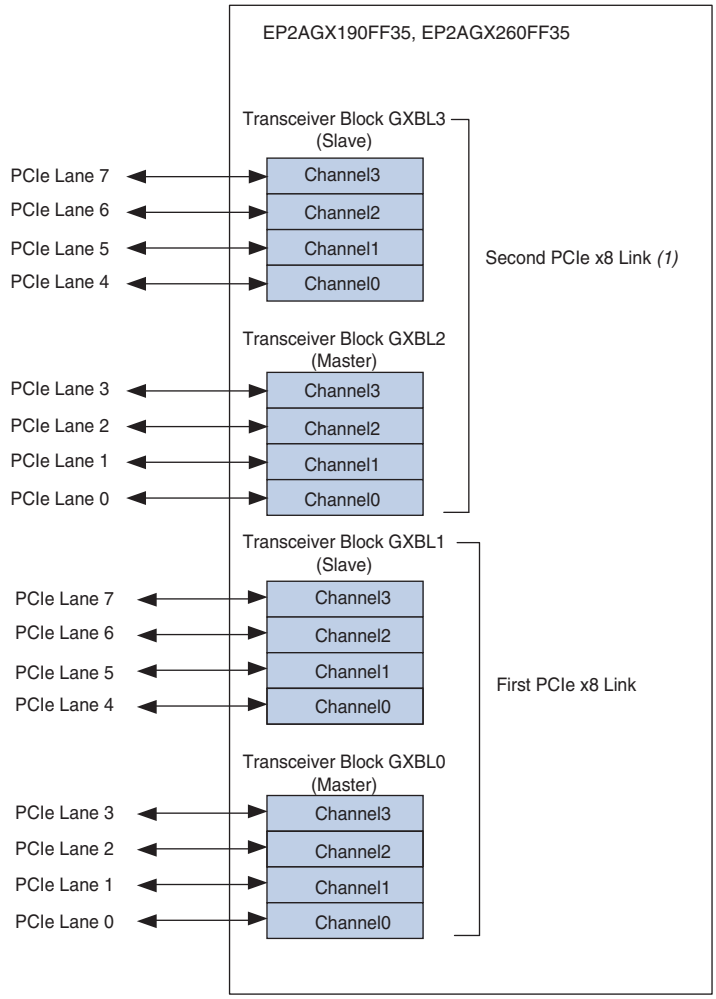


Note to Figure 2–9:

(1) Arria II GX and GZ devices with three transceiver blocks allow a maximum of one PCIe x8 link occupying two transceiver blocks. You can configure the other transceiver block to implement other functional modes.

Figure 2–10 shows the PCIe x8 link in four transceiver block Arria II GX devices.

Figure 2–10. Two PCIe x8 Link in Four Transceiver Block Arria II GX Devices

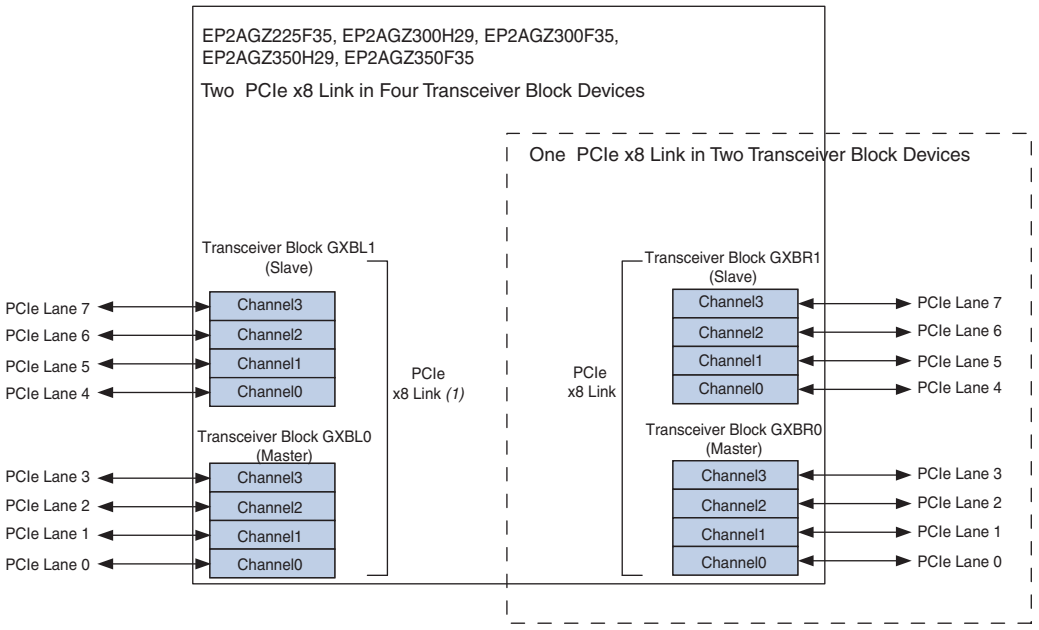


Note to Figure 2–10:

(1) The second x8 link does not have PCIe hard IP support. Use soft IP support for the second x8 link.

Figure 2–11 shows two PCIe x8 links in four transceiver block Arria II GZ devices.

Figure 2–11. One PCIe x8 Link in Two Transceiver Block Devices and Two PCIe x8 Links in Four Transceiver Block Arria II GZ Devices

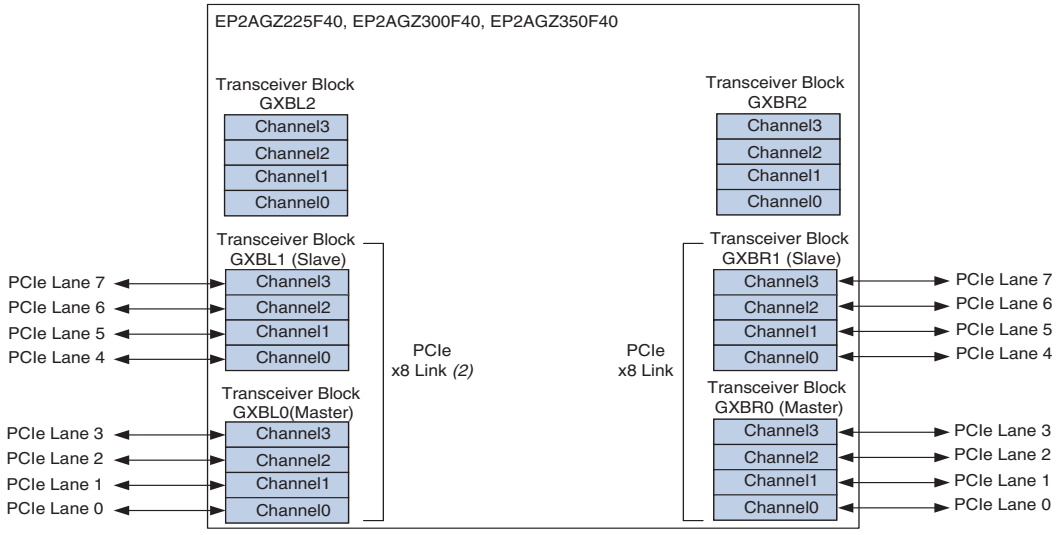


Note to Figure 2–11:

(1) The PCIe hard IP block is only available on the left side of the Arria II device.

Figure 2–12 shows two PCIe x8 links in six transceiver block Arria II GZ devices.

Figure 2–12. Two PCIe x8 Links in Six Transceiver Block Arria II GZ Devices *(Note 1)*



Notes to Figure 2–12:

(1) Arria II GZ devices with six transceiver blocks allow a maximum of two PCIe x8 links occupying four transceiver blocks. You can configure the other two transceiver blocks to implement other functional modes.

(2) The PCIe hard IP block is only available on the left side of the Arria II device.

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, the 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 in Arria II Devices* chapter.

In addition, using the rate matcher block affects PCS clocking in non-bonded channel configurations.

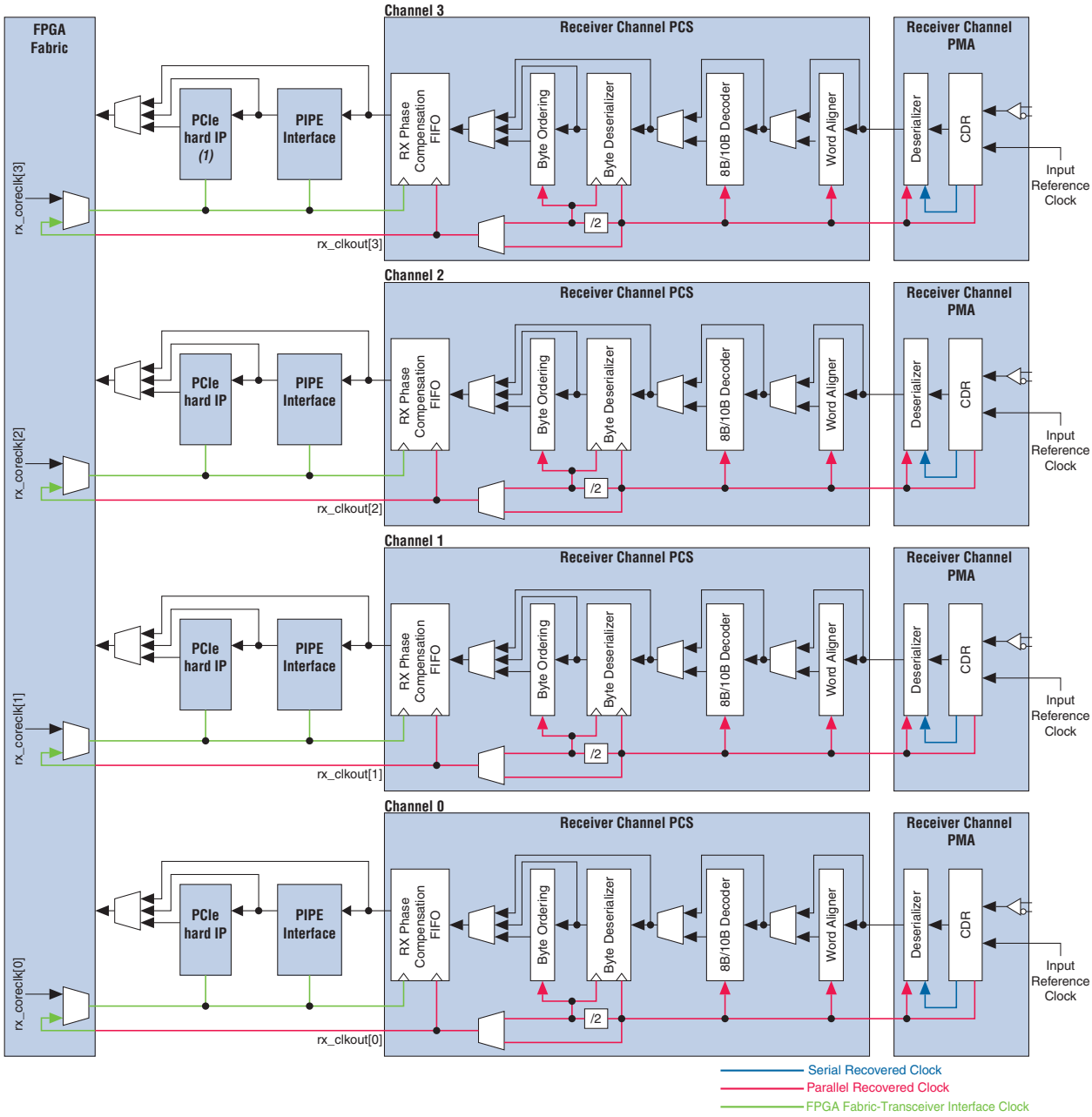
Non-Bonded Receiver Clocking Without Rate Matcher

The following functional modes have non-bonded receiver channel configuration without rate-matcher:

- SRIO
- SONET/SDH
- SDI
- CPRI/OBSAI
- Basic without rate matcher

Figure 2–13 shows receiver datapath clocking in non-bonded channel configurations without rate matcher.

Figure 2–13. Receiver Datapath Clocking in Non-Bonded Configurations without Rate Matcher



Note to Figure 2–13:

(1) In Arria II GX and GZ devices, there is only one dedicated PCIe hard IP, which supports PCIe Gen 1 x1, x4, and x8; and PCIe Gen 2 x1 and x4.

In non-bonded configurations without rate matcher, 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 PMA receiver divides the serial recovered clock 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 lists the 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 for Arria II Devices

Functional Mode *(1)*	Data Rate	High-Speed Serial Clock Frequency	Low-Speed Parallel Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer (MHz)	With Byte Serializer (MHz)
SRIO	1.25 Gbps	625 MHz	125	—	62.5
	2.5 Gbps	1.25 GHz	250	—	125
	3.125 Gbps	1.5625 GHz	312.5	—	156.25
SONET/SDH OC12	622 Mbps	311 MHz	77.75	77.75	—
SONET/SDH OC48	2.488 Gbps	1.244 GHz	311	—	155.5
HD SDI	1.485 Gbps	742.5 MHz	148.5	148.5	74.25
	1.4835 Gbps	741.75 MHz	148.35	148.35	74.175
3G-SDI	2.97 Gbps	1.485 GHz	297	—	148.5
	2.967 Gbps	1.4835 Ghz	296.7	—	148.35

Note to Table 2–4:

(1) Altera also supports CPRI and OBSAI. For more information, refer to *AN 610: Implementing CPRI and OBSAI Protocols in Altera Devices*.

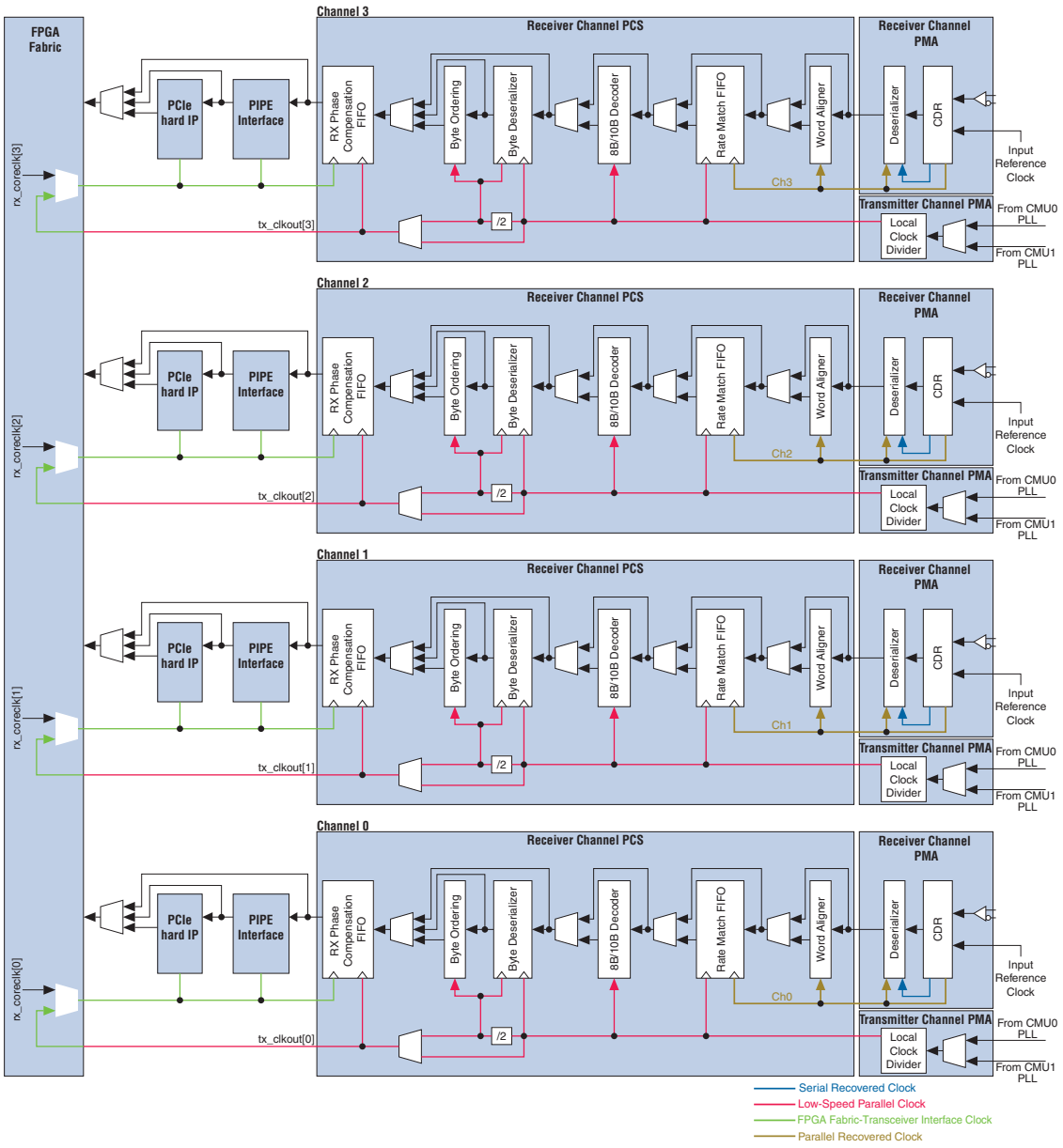
Non-Bonded Receiver Clocking with Rate Matcher

The following functional modes have non-bonded receiver channel configurations with rate-matcher:

- PCIe x1
- GbE
- SRIO
- Basic with rate matcher

Figure 2–14 shows the receiver datapath clocking in non-bonded channel configurations with rate matcher.

Figure 2–14. 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, the 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 lists 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 for Arria II Devices

Functional Mode	Data Rate (Gbps)	Serial Recovered Clock Frequency	Parallel Recovered Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer (MHz)	With Byte Serializer (MHz)
PCIe x1 (Gen 1)	2.5	1.25 GHz	250	250 *(1)*	125
PCIe x1 (Gen 2)	5	2.5 GHz	500	—	250
GbE	1.25	625 MHz	125	125	—
SRIO	1.25	625 MHz	125	—	62.5
	2.5	1.25 GHz	250	—	125
	3.125	1.5625 GHz	312.5	—	156.25

Note to Table 2–5:

(1) 250 MHz when you enable the PCIe hard IP.

Bonded Channel Configurations

Arria II GX and GZ devices support x4 channel bonding that allows bonding of four channels within the same transceiver block. It also supports x8 channel bonding that allows bonding of eight channels across two transceiver blocks in PCIe mode.

x4 Bonded Channel Configuration

The following functional modes support x4 receiver channel bonded configuration:

- PCIe x4
- XAUI
- Basic x4

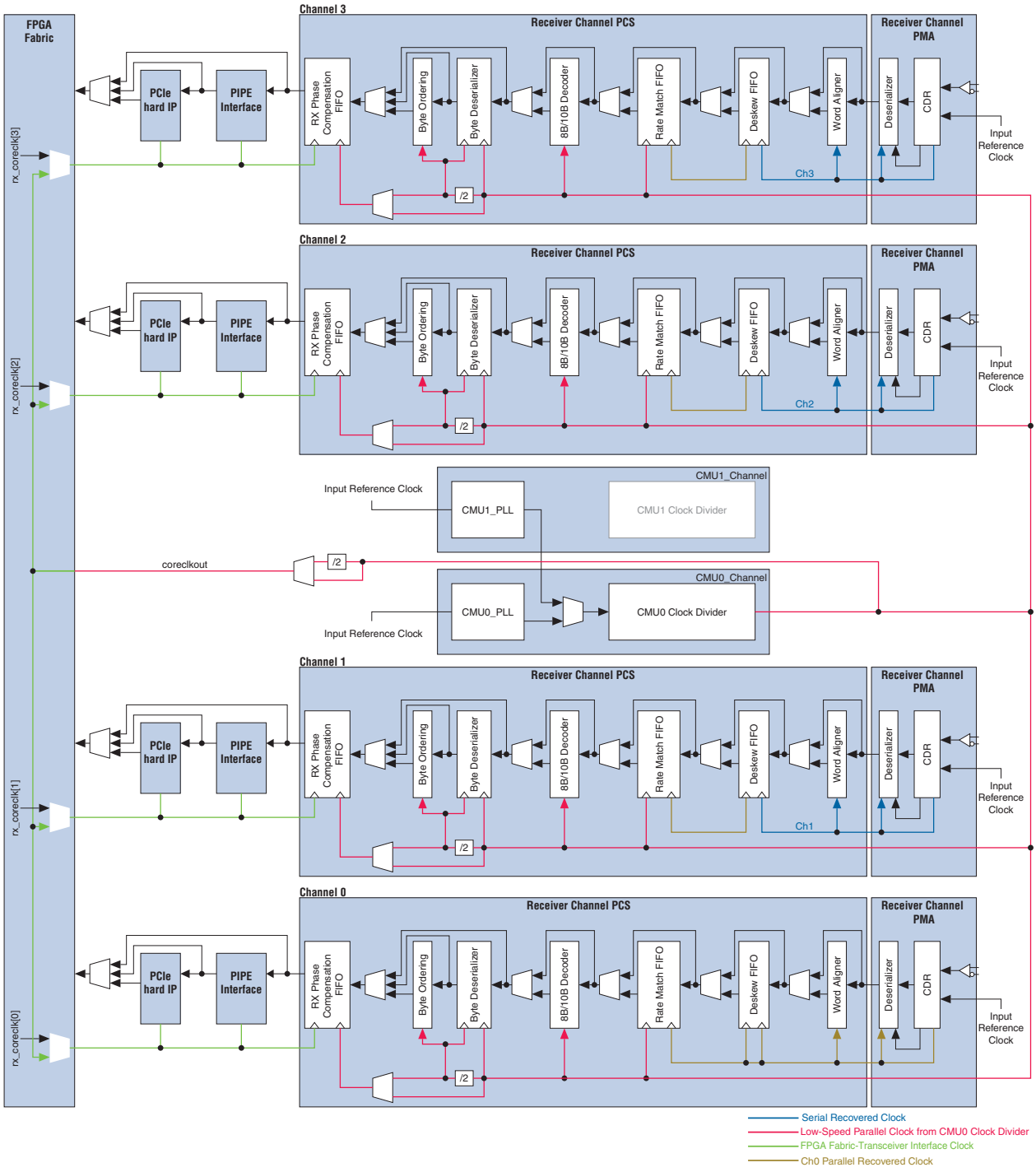
In x4 bonded channel configurations, the receiver datapath clocking varies, depending on whether the configured functional mode uses the deskew FIFO or not.

x4 Bonded Channel Configuration with Deskew FIFO

XAUI functional mode has x4 bonded channel configuration with deskew FIFO.

Figure 2–15 shows receiver datapath clocking in x4 channel bonding configurations with deskew FIFO.

Figure 2–15. Receiver Datapath Clocking in x4 Bonded Channel Configuration with Deskew FIFO



In x4 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 the 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, the 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 x4 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 lists the receiver datapath clock frequencies in x4 bonded functional modes with deskew FIFO.

Table 2–6. Receiver Datapath Clock Frequencies in x4 Bonded Functional Modes with Deskew FIFO for Arria II Devices

Functional Mode	Data Rate (Gbps)	Serial Recovered Clock Frequency (GHz)	Parallel Recovered Clock and Parallel Transmitter PCS Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer	With Byte Serializer (MHz)
PCIe x4 (Gen 1)	2.5	1.25	250	— *(1)*	125
PCIe x4 (Gen 2) *(2)*	5	2.5	500	—	250
XAUI	3.125	1.5625	312.5	—	156.25

Notes to Table 2–6:

(1) 250 MHz when you enable the PCIe hard IP.

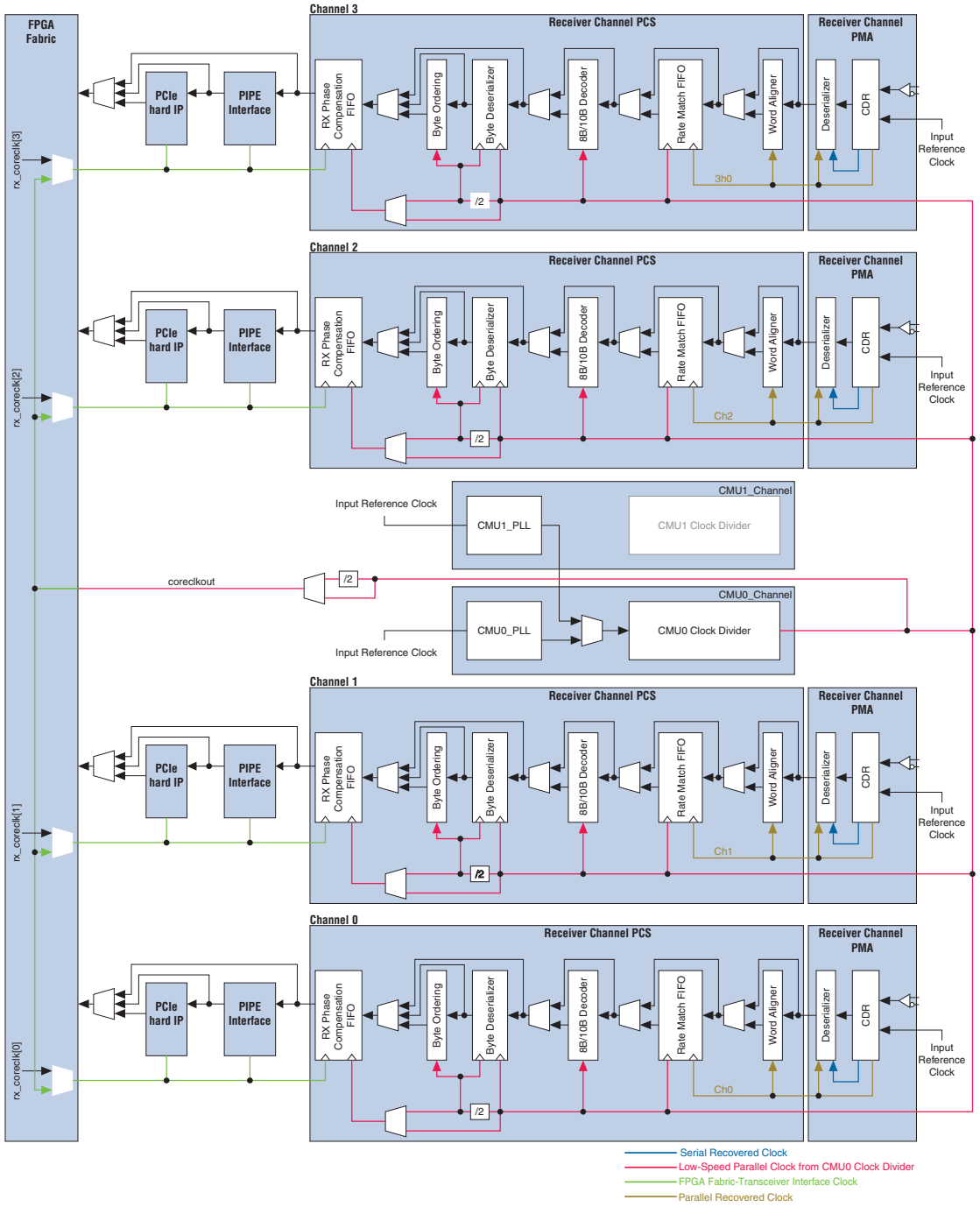
(2) Arria II GZ devices only.

x4 Bonded Channel Configurations Without Deskew FIFO

PCIe x4 functional modes have x4 bonded channel configurations without deskew FIFO.

Figure 2–16 shows receiver datapath clocking in x4 channel bonding configurations without deskew FIFO.

Figure 2–16. Receiver Datapath Clocking in x4 Bonded Channel Configurations Without Deskew FIFO



In x4 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 the `CMU0 Channel` clocks the read port of the rate match FIFO, the 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 x4 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 lists the receiver datapath clock frequencies in x4 bonded functional modes without deskew FIFO.

Table 2–7. Receiver Datapath Clock Frequencies in x4 Bonded Functional Modes Without Deskew FIFO for Arria II Devices

Functional Mode	Data Rate (Gbps)	Serial Recovered Clock Frequency (GHz)	Parallel Recovered Clock and Parallel Transmitter PCS Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer (MHz)	With Byte Serializer (MHz)
PCIe x4 (Gen 1)	2.5	1.25	250	250 *(1)*	125
PCIe x4 (Gen 2) *(2)*	5	2.5	500	—	250

Notes to Table 2–7:

(1) 250 MHz when you enable the PCIe hard IP.

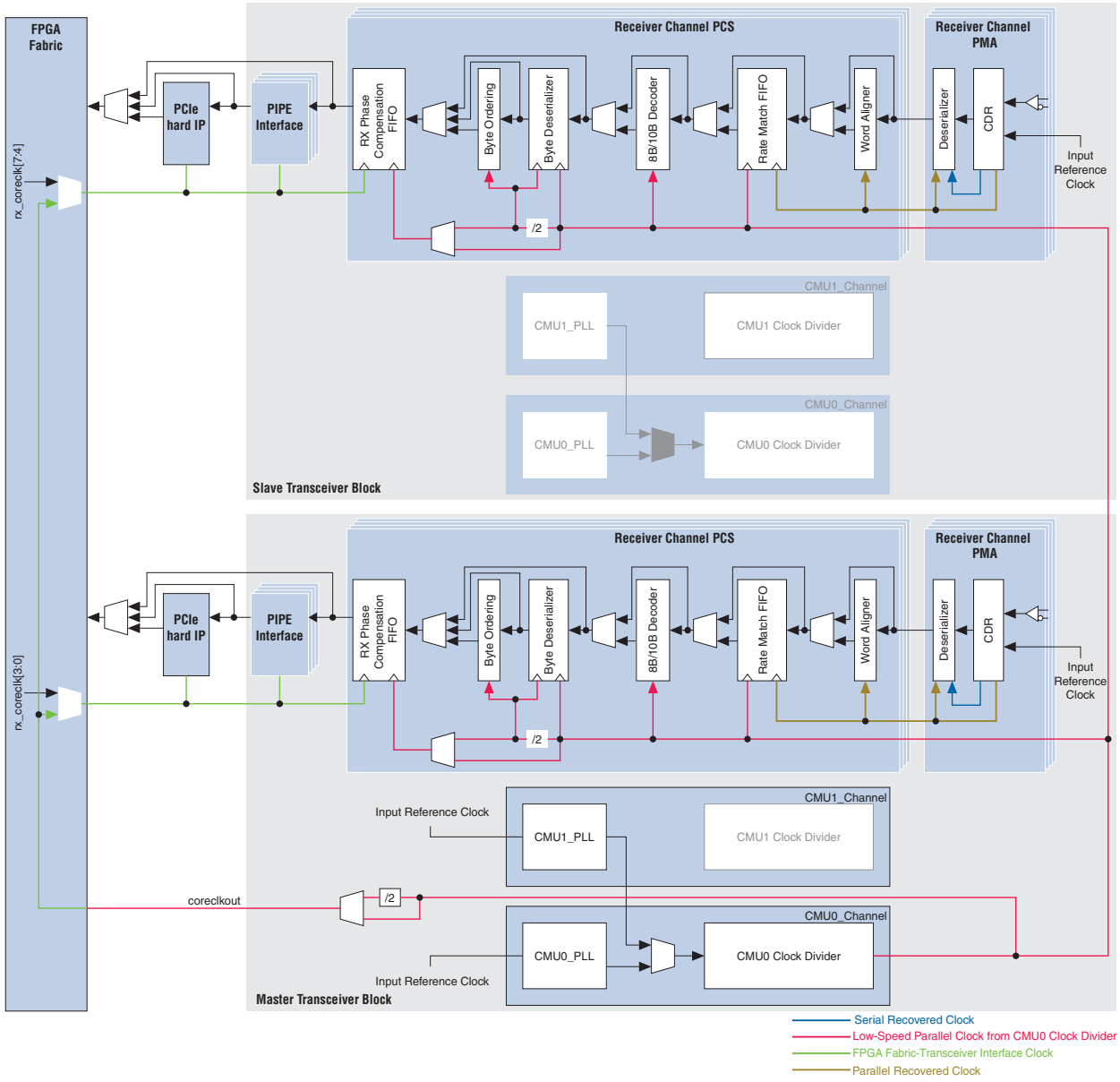
(2) Arria II GZ devices only.

x8 Bonded Channel Configuration

PCIe x8 and Basic x8 functional mode supports x8 receiver channel bonding configurations. 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 2–17 shows receiver datapath clocking in x8 bonded channel configuration.

Figure 2–17. Receiver Datapath Clocking in x8 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, the 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 PCIe x8 mode.

Table 2–8 lists the receiver datapath clock frequencies in PCIe x8 functional mode.

Table 2–8. Receiver Datapath Clock Frequencies in PCIe x8 Functional Modes for Arria II Devices

Functional Mode	Data Rate (Gbps)	Serial Recovered Clock Frequency (GHz)	Parallel Recovered clock and Parallel Transmitter PCS Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer	With Byte Serializer (MHz)
PCIe x8 (Gen 1)	2.5	1.25	250	— *(1)*	125
PCIe x8 (Gen 2) *(2)*	5	2.5	500	—	250

Notes to Table 2–8:

(1) 250 MHz when you enable the PCIe hard IP.

(2) Arria II GZ devices only.

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 divided into the following three categories:

- "Input Reference Clocks"
- "Phase Compensation FIFO Clocks" on page 2–29
- "Other Transceiver Clocks" on page 2–29

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 ITB clock network
- CLK input pins on the FPGA global clock network
- Clock output pins from the left side and right side 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 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 the 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, regional clock resource, or periphery clock 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 PCIe receiver detect circuitry

The Quartus II software automatically routes fixed_clk on the FPGA fabric global clock or regional clock network.

Table 2–9 lists the FPGA fabric-transceiver interface clocks.

Table 2–9. FPGA Fabric-Transceiver Interface Clocks for Arria II Devices (Part 1 of 2)

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	Global clock
rx_cruclk	Receiver CDR input reference clock when driven from an FPGA CLK input pin	FPGA fabric-to-transceiver	Global clock
tx_clkout	Phase compensation FIFO clock	Transceiver-to-FPGA fabric	Global, Regional, Periphery clocks

Table 2–9. FPGA Fabric-Transceiver Interface Clocks for Arria II Devices (Part 2 of 2)

Clock Name	Clock Description	Interface Direction	FPGA Fabric Clock Resource Utilization *(1)*
coreclkout	Phase compensation FIFO clock	Transceiver-to-FPGA fabric	Global, Regional, Periphery clocks
rx_clkout	Phase compensation FIFO clock	Transceiver-to-FPGA fabric	Global, Regional, Periphery clocks
fixed_clk	PCIe receiver detect clock	FPGA fabric-to-transceiver	Global, Regional clocks

Note to Table 2–9:

(1) For more information about global, regional, and periphery clock resources available in each device, refer to the *Clock Networks and PLLs in Arria II Devices* chapter.

"FPGA Fabric-Transmitter Interface Clocking" 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 global, regional, and periphery clock resource usage 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 parts per million (PPM) frequency difference.

Arria II GX and GZ 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.

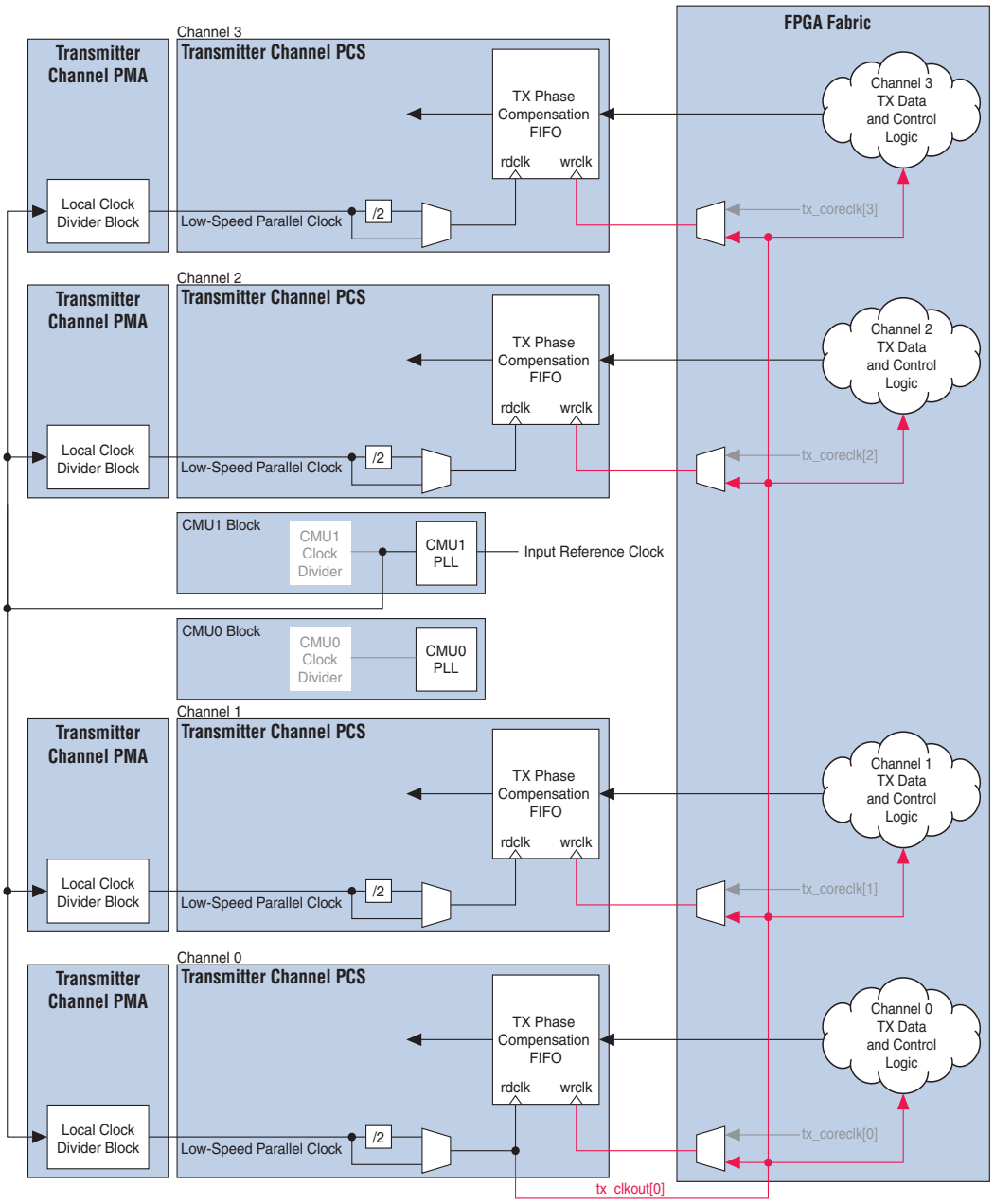
Example 1: Four Identical Channels in a Transceiver Block

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 2–18. 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 global, regional, and periphery clock resource for `tx_clkout[0]`.

Figure 2–18. Four Identical Channels in a Transceiver Block for Example 1



Example 2: Two Groups of Two Identical Channels in a Transceiver Block

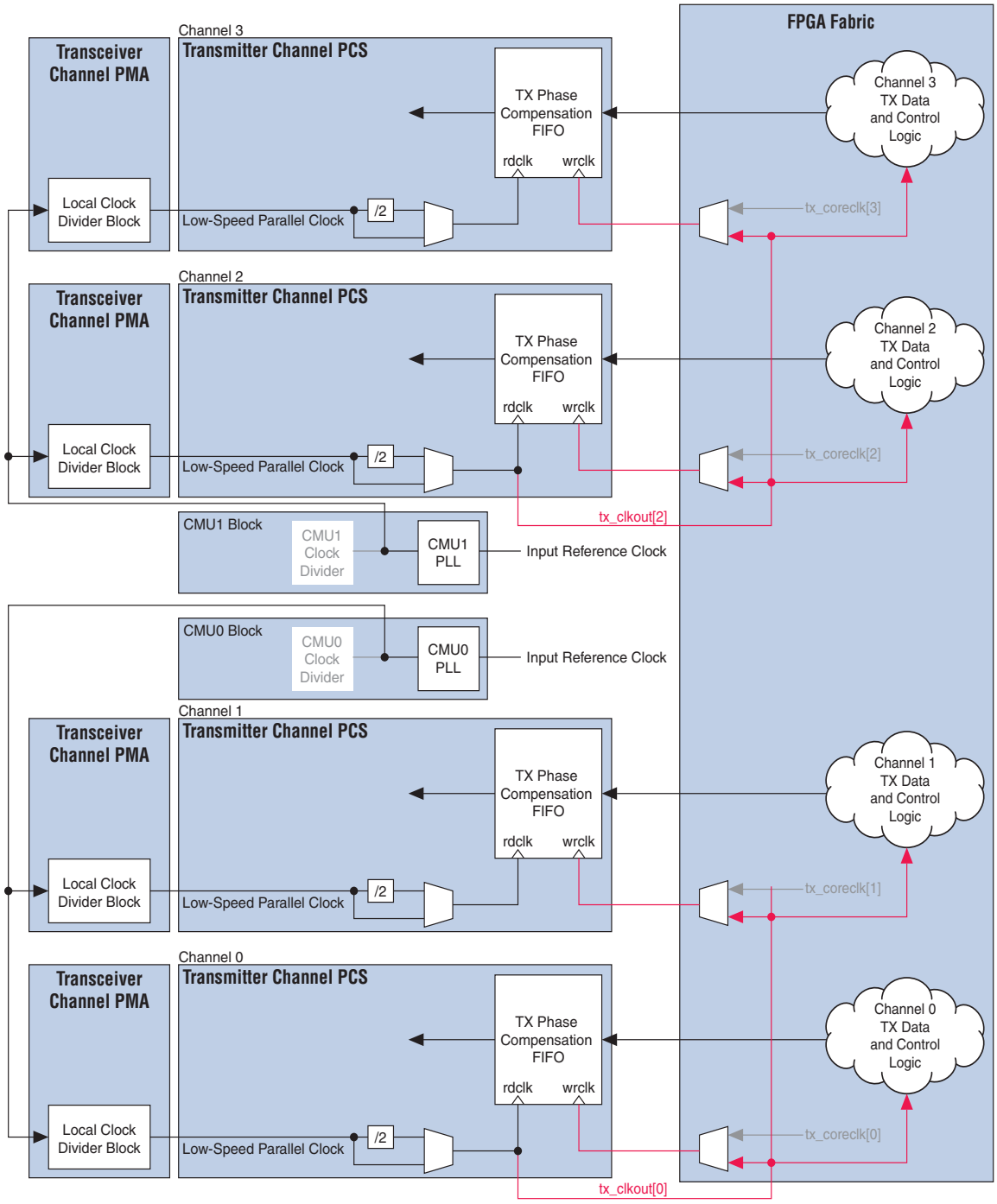
Example 2 assumes channels 0 and 1, driven by `CMU0 PLL` in a transceiver block, are identical. Also, channels 2 and 3, driven by `CMU1 PLL` in the same transceiver block, are identical. In this case, the Quartus II software automatically drives the write port of the transmitter phase compensation FIFO 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 2–19 shows FPGA fabric-transmitter interface clocking for Example 2.

Figure 2–19. FPGA Fabric-Transmitter Interface Clocking for Example 2



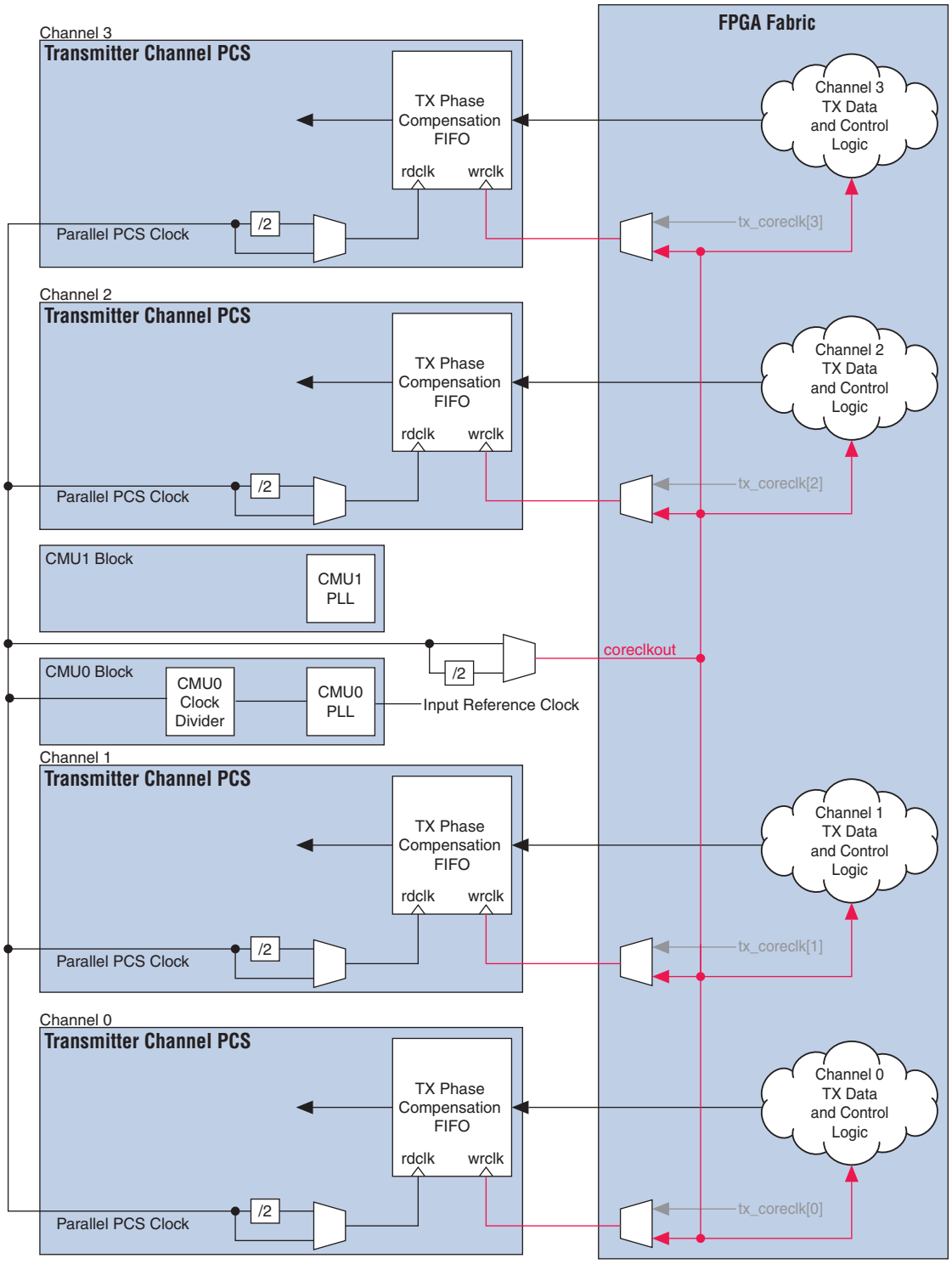
Bonded Channel Configuration

In the x4 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 x8 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 2–20 shows FPGA fabric-transmitter interface clocking in an x4 bonded channel configuration.

Figure 2–20. FPGA Fabric-Transmitter Interface Clocking in an x4 Bonded Channel Configuration



Limitations of the Quartus II Software-Selected Transmitter Phase Compensation FIFO Write Clock

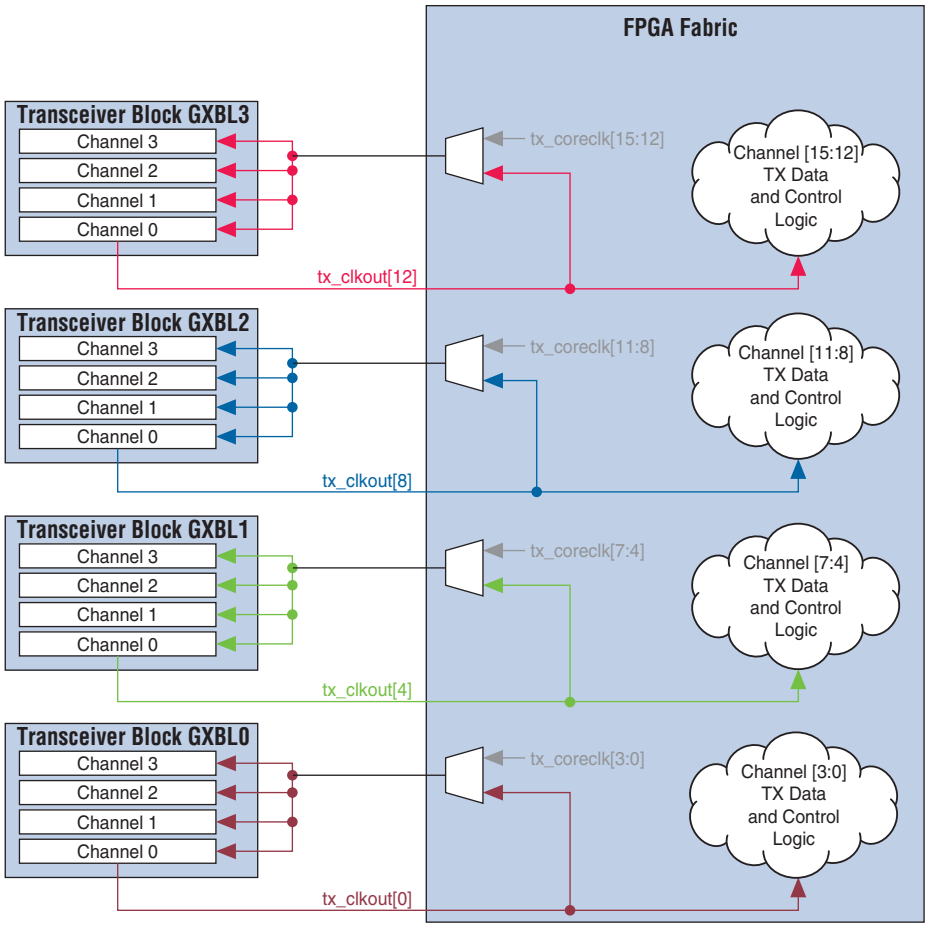
The Quartus II software uses a single tx_clkout signal to clock the transmitter phase compensation FIFO write port of all identical channels within a transceiver block. This usage results in one global or regional clock resource for each group of identical channels within a transceiver block.

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, resulting in higher clock resource usage.

Example 3: Sixteen Identical Channels Across Four Transceiver Blocks

Consider 16 identical transmitter channels located across four transceiver blocks, as shown in Figure 2–21. 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, resulting in four clocks resources (global, regional, or both) being used, one for each transceiver block.

Figure 2–21. 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 implement this clocking scheme, 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".

User-Selected Transmitter Phase Compensation FIFO Write Clock

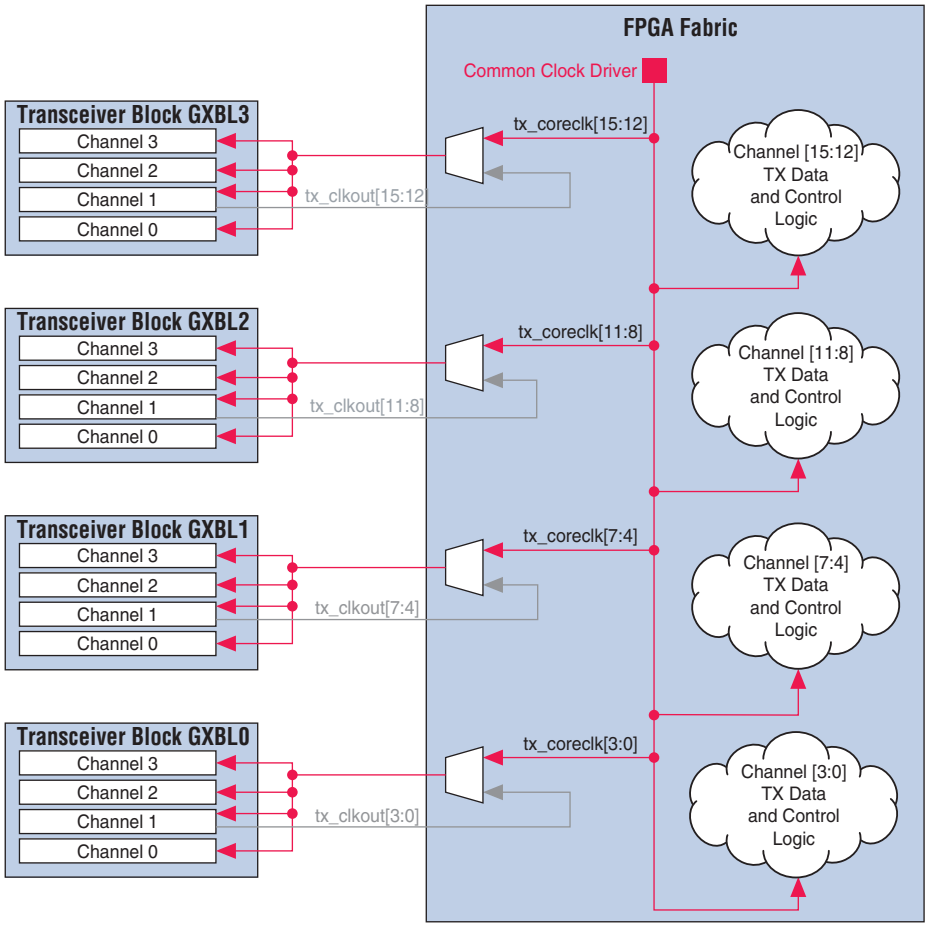
The ALTGX MegaWizard Plug-In Manager provides an optional tx_coreclk port for each instantiated transmitter channel. If you enable this port, the Quartus II software does not automatically select the transmitter phase compensation FIFO write clock source. Instead, the signal that you drive on the tx_coreclk 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 usage (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 uses only one global or regional clock resource for all identical channels in your design.

Example 4: Sixteen Identical Channels Across Four Transceiver Blocks

Figure 2–22 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 with 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 2–22. 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 lists 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 for Arria II Devices

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)
x4 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)
x8 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 "GXB 0 PPM Core Clock Setting" user assignment whenever you use the tx_coreclk port to drive the transmitter phase compensation FIFO write clock.



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 usage. 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 the 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.

Table 2–11 lists the Quartus II assignments that you must make in the assignment editor.

Table 2–11. Quartus II Assignments for Arria II Devices

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 2–23 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 2–23. Sixteen Identical Channels Across Four Transceiver Blocks for Example 5

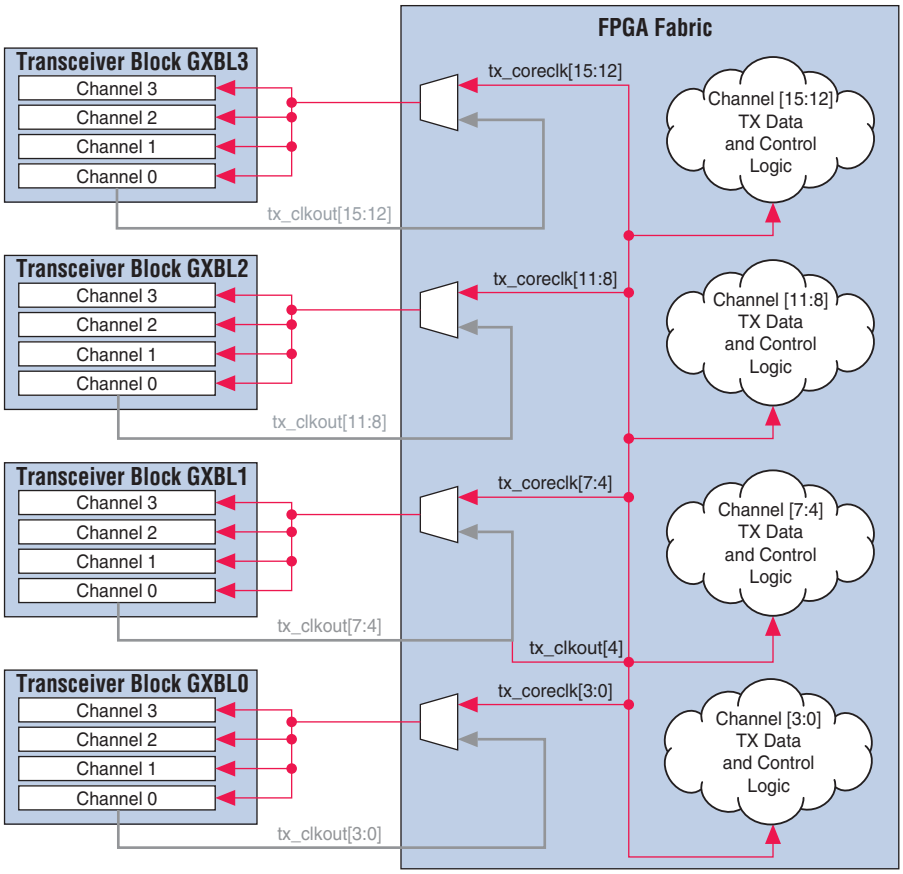


Table 2–12 lists the Quartus II assignments that you must make for the clocking scheme shown in Figure 2–23.

Table 2–12. Quartus II Assignments for Arria II Devices

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 and GZ 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

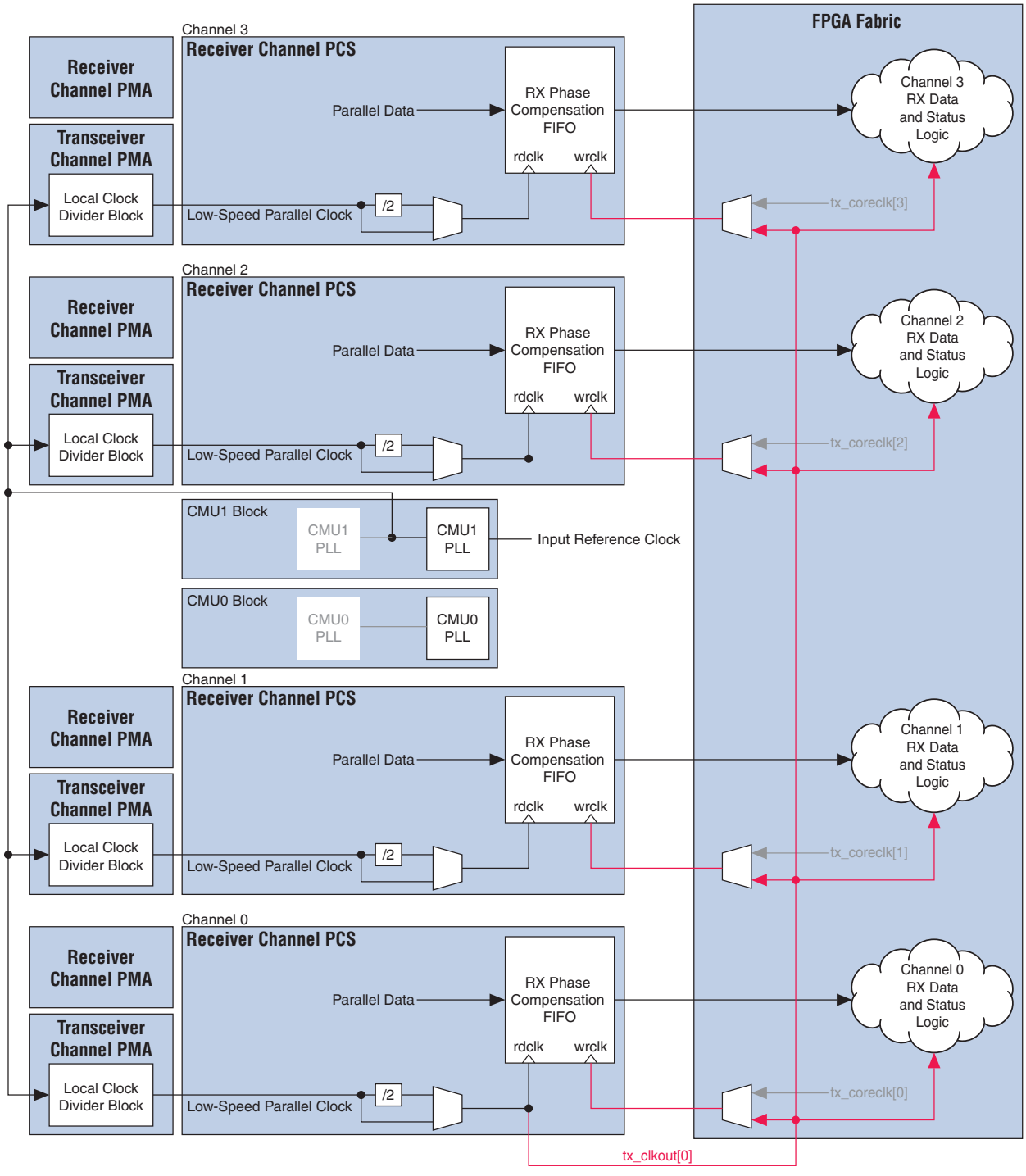
In the non-bonded channel configuration, the transceiver channels may or may not be identical. Identical transceiver channels are defined as channels that have the same CMU PLL and receiver CDR input reference clock source, have exactly the same CMU PLL and receiver CDR configuration, and have exactly 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 2–24. 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 2–24. Four Identical Channels in a Transceiver Block for Example 6



Example 7: Two Groups of Two Identical Channels in a Transceiver Block

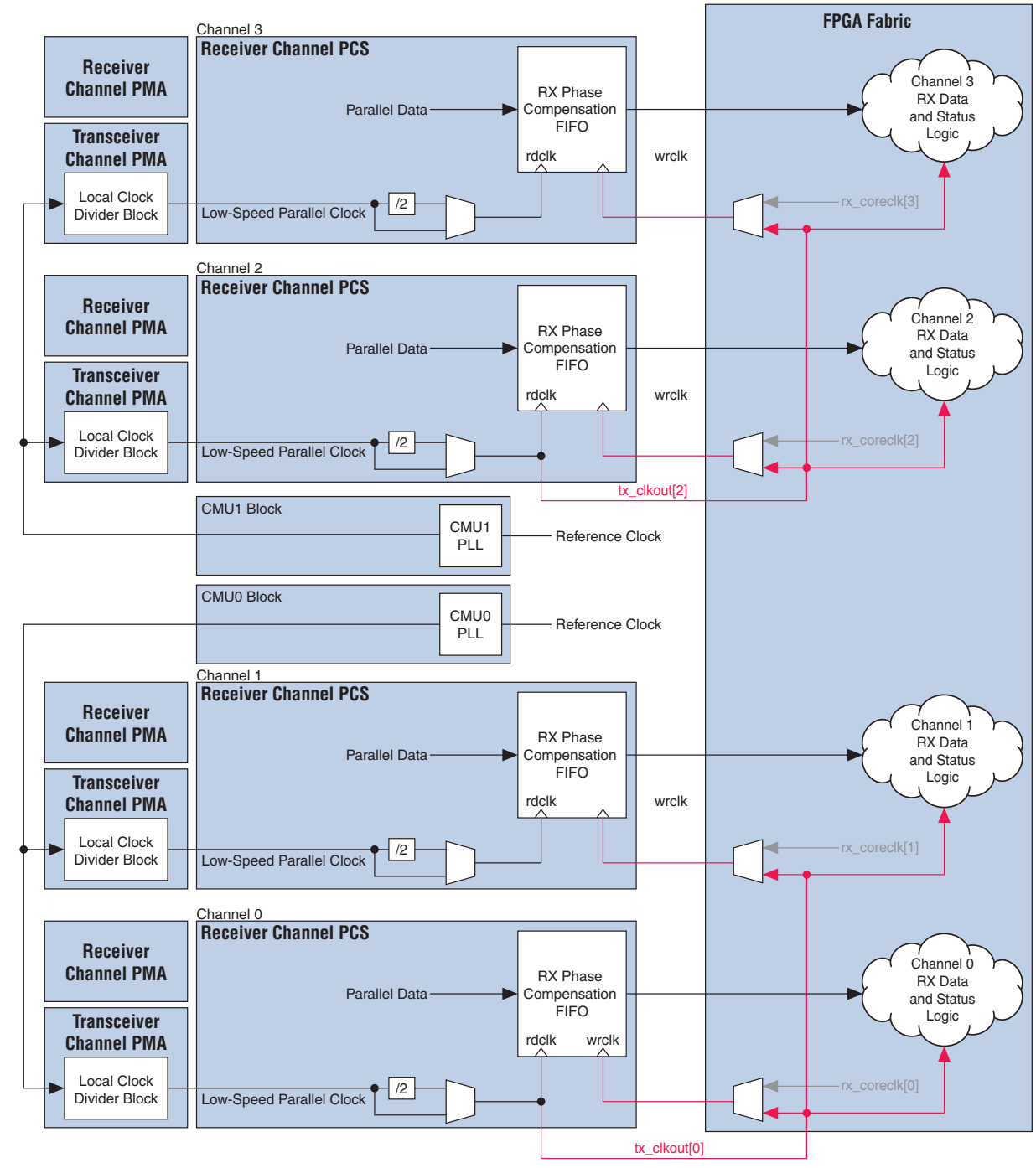
This example assumes channels 0 and 1, driven by CMU0 PLL in a transceiver block, are identical. Also, channels 2 and 3, driven by CMU1 PLL in the same transceiver block, are identical. In this case, the Quartus II software automatically drives the read port of the receiver phase compensation FIFO in 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 2–25 shows FPGA fabric-receiver interface clocking for Example 7.

Figure 2–25. FPGA Fabric-Receiver Interface Clocking for Example 7



Non-Bonded Channel Configuration without Rate Matcher

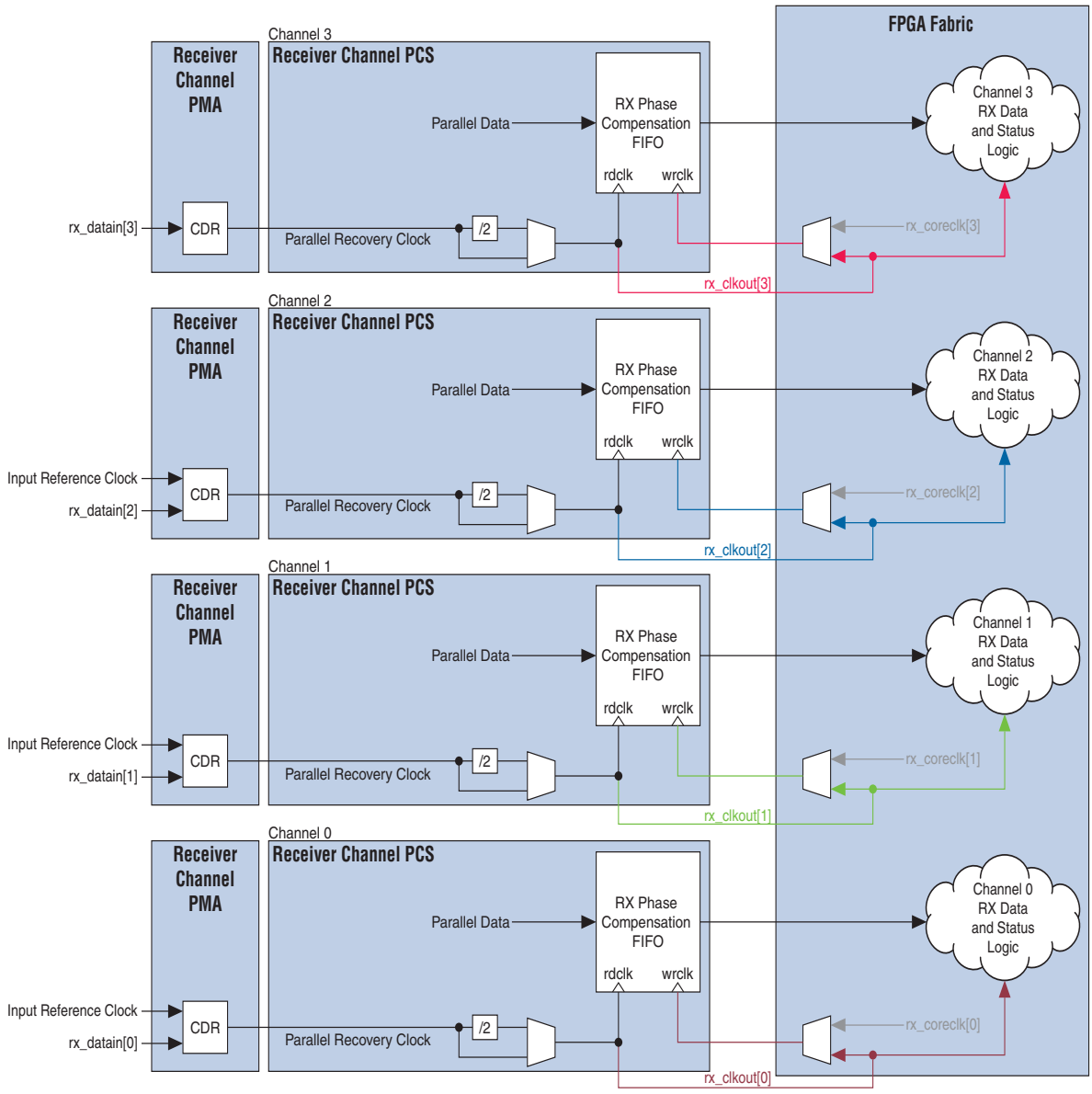
In the non-bonded channel configuration without the rate matcher, the Quartus II software cannot determine if the incoming serial data in all channels have a 0 PPM frequency difference. The Quartus II software automatically drives the read port of the receiver phase compensation FIFO in each channel 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 2–26 shows the FPGA fabric-receiver interface clocking for non-bonded channel configurations without rate matcher.

Figure 2–26. 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 x4 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.

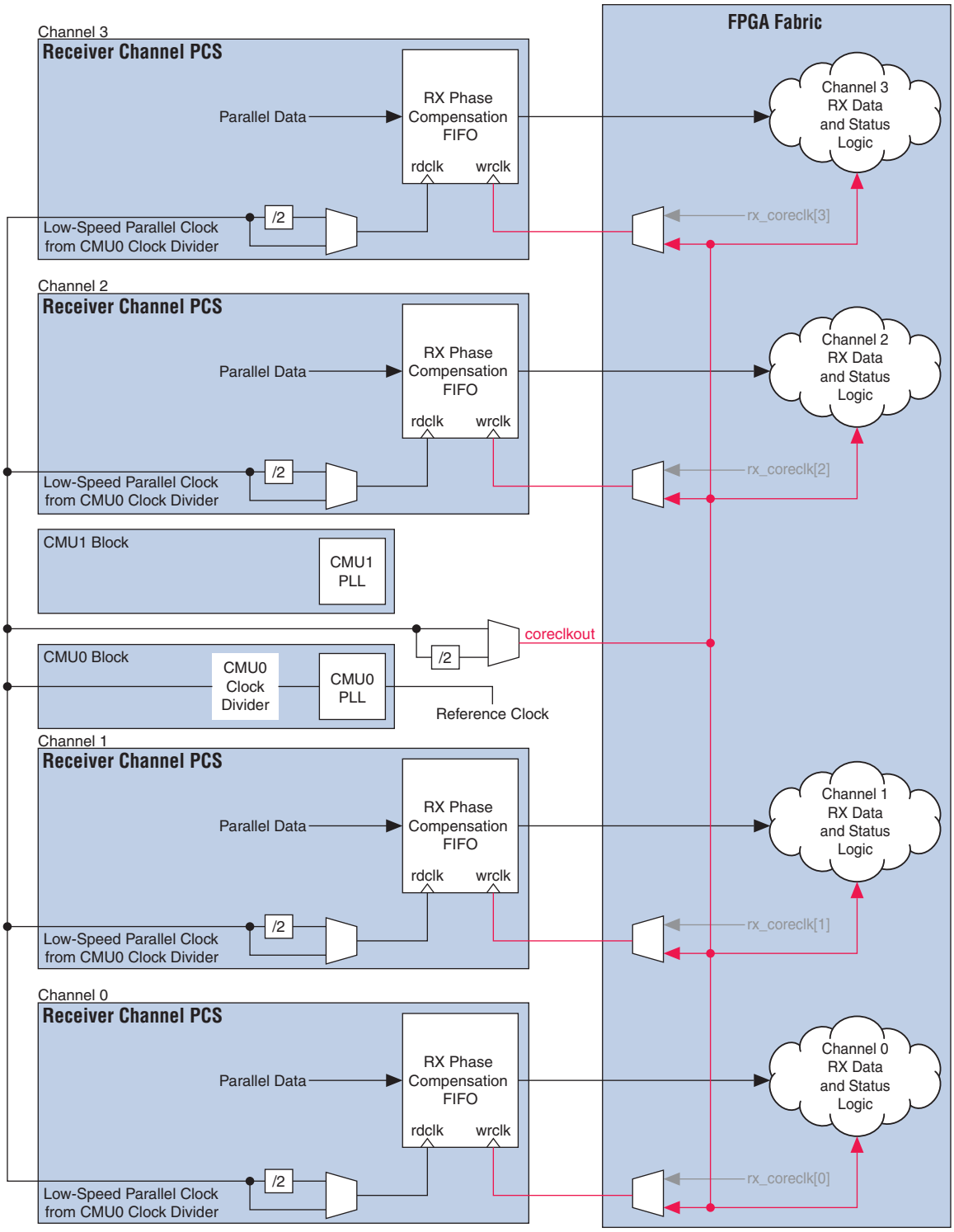
In the x8 bonded channel configurations, the Quartus II software automatically drives the read port of the receiver phase compensation FIFO in all eight channels with the `coreclkout` signal from the master transceiver block. Use the `coreclkout` signal to latch the receiver data and status signals from 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 2–27 shows FPGA fabric-receiver interface clocking in an x4 bonded channel configuration.

Figure 2–27. FGPA Fabric-Receiver Interface Clocking in an x4 Bonded Channel Configuration



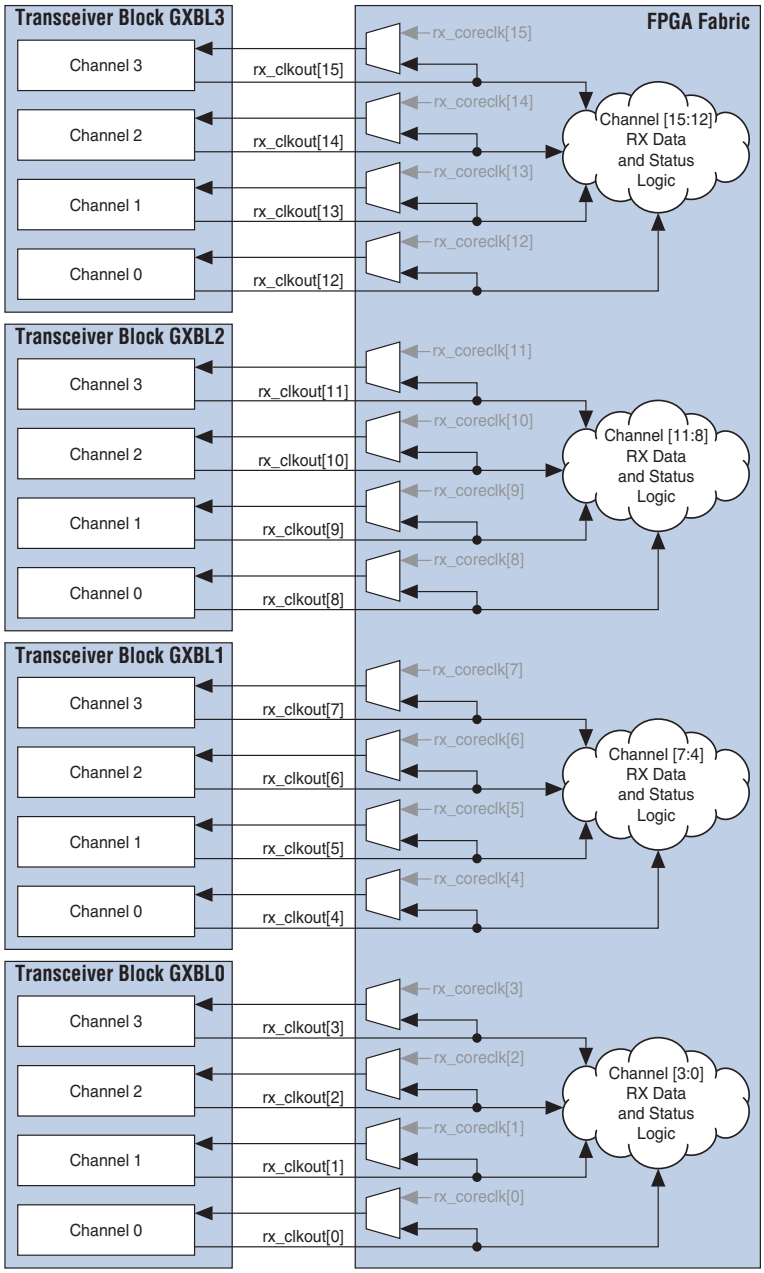
Limitations of Quartus II Software-Selected Receiver Phase Compensation FIFO Read Clock

In the non-bonded channel configurations without a 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, resulting 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 a rate matcher located across four transceiver blocks, as shown in Figure 2–28. 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, resulting in 16 clocks resources (global, regional, or both) being used, one for each channel.

Figure 2–28. Sixteen Non-Bonded Receiver Channels without Rate Matcher for Example 8



Because 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, resulting in only one global or regional clock resource being used instead of 16. To implement this clocking scheme, you must select the receiver phase compensation FIFO read clocks instead of the Quartus II software automatic selection, as described in the "User-Selected Receiver Phase Compensation FIFO Read Clock" section.

User-Selected Receiver Phase Compensation FIFO Read Clock

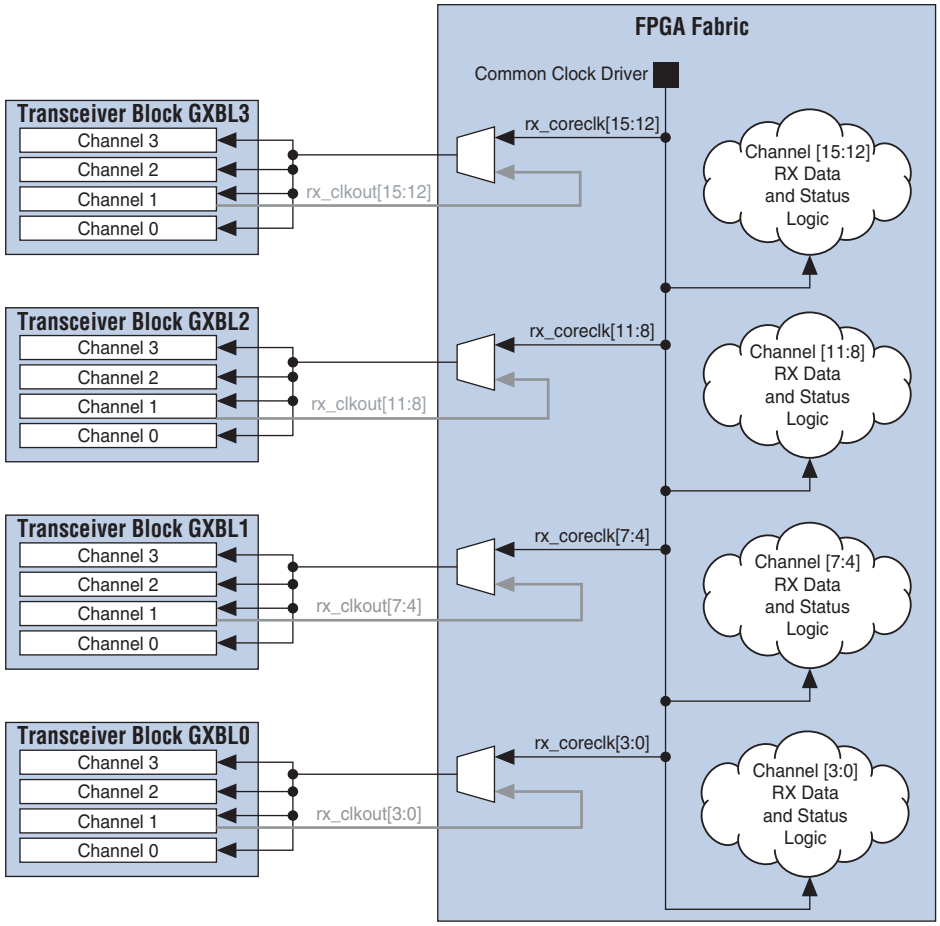
The ALTGX MegaWizard Plug-In Manager provides an optional `rx_coreclk` port for each instantiated receiver channel. If you enable this port, the Quartus II software does not automatically select the receiver phase compensation FIFO read clock source. Instead, the signal that you drive on the `rx_coreclk` 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 usage (global, regional, or both). You can connect the `rx_coreclk` ports of all the 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 uses only one global or regional clock resource for all channels.

Example 9: Sixteen Identical Channels Across Four Transceiver Blocks

Figure 2–29 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 with 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 2–29. 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 lists the receiver phase compensation FIFO write clocks that the Quartus II software selects in various configurations.

Table 2–13. Receiver Phase Compensation FIFO Write Clocks for Arria II Devices

Configuration	Receiver Phase Compensation FIFO Write Clock	
	Without Byte Deserializer	With Byte Deserializer
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)
x4 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)
x8 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 "GXB 0 PPM Core Clock Setting" user assignment whenever you use the rx_coreclk port to drive the receiver phase compensation FIFO read clock.

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 usage. 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.

Because the 0 PPM clock group assignment allows the 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 lists the Quartus II assignments that you must make for the clocking scheme shown in Figure 2–29.

Table 2–14. Quartus II Assignments for Arria II Devices

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 2–30 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 2–30. Sixteen Channels Across Four Transceiver Blocks for Example 10

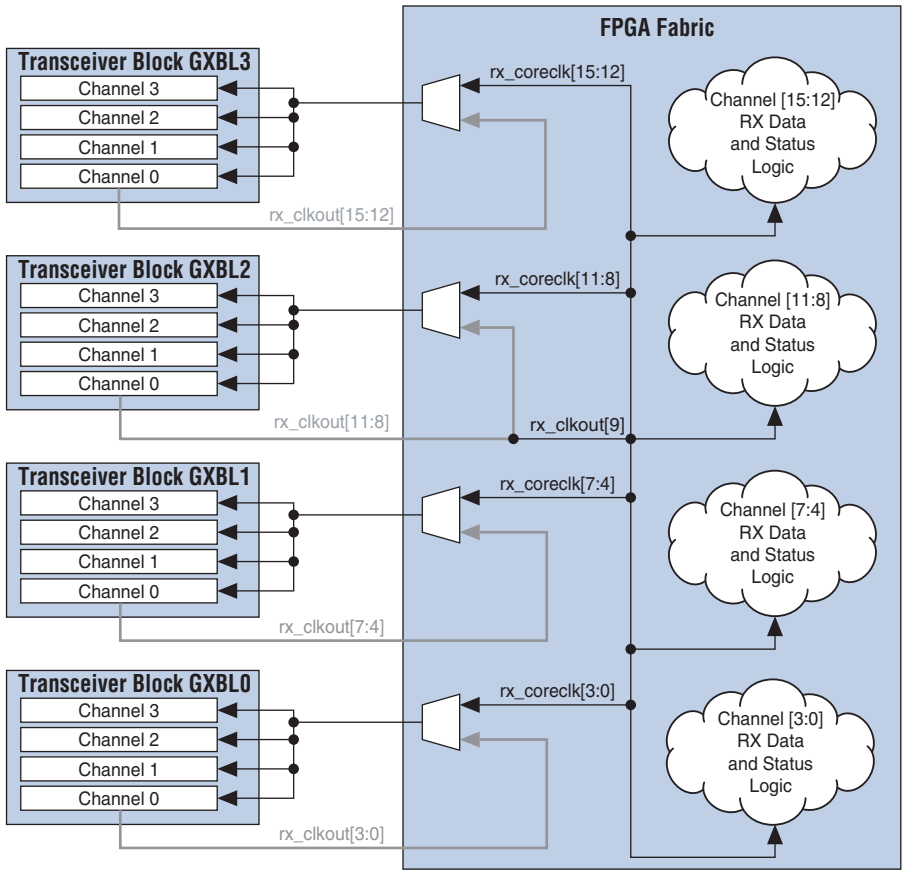


Table 2–15 lists the Quartus II assignments that you must make for the clocking scheme shown in Figure 2–30.

Table 2–15. Quartus II Assignments for Example 10 for Arria II Devices

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 PLL-Transceiver PLL 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 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

For 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 lists the allowed input reference clock frequencies for Example 11.

Table 2–16. Allowed Input Reference Clock Frequencies for Example 11 for Arria II Devices

Multiplication Factor (M)	On-Board Crystal Reference Clock Frequency (MHz)		Allowed
	with /N = 1	With /N = 2	
2	750	1500	No *(1)*
4	375	750	No *(1)*
5	300	600	Yes
8	187.5	375	Yes
10	150	300	Yes
16	93.75	187.5	Yes
20	75	150	Yes
25	60	120	Yes

Note to Table 2–16:

(1) Violates the PFD frequency limit of 325 MHz.

For a 3 Gbps data rate, the Quartus II software 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 and GZ 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.

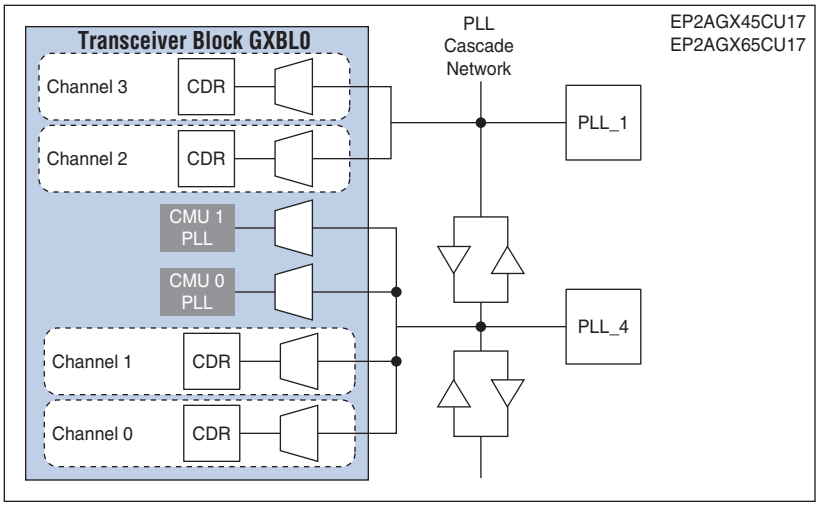
The following sections describe the dedicated PLL cascade networks available in the Arria II GX device family.

The FPGA fabric PLLs-transceiver PLLs cascading option is available in the following Arria II GX devices with four channels:

- EP2AGX45CU17
- EP2AGX65CU17

Figure 2–31 shows the FPGA fabric PLLs-transceiver PLLs cascading option allowed in the EP2AGX45CU17 and EP2AGX65CU17 devices.

Figure 2–31. FPGA Fabric PLLs-Transceiver PLLs Cascading Option Allowed in the EP2AGX45CU17 and EP2AGX65CU17 Devices

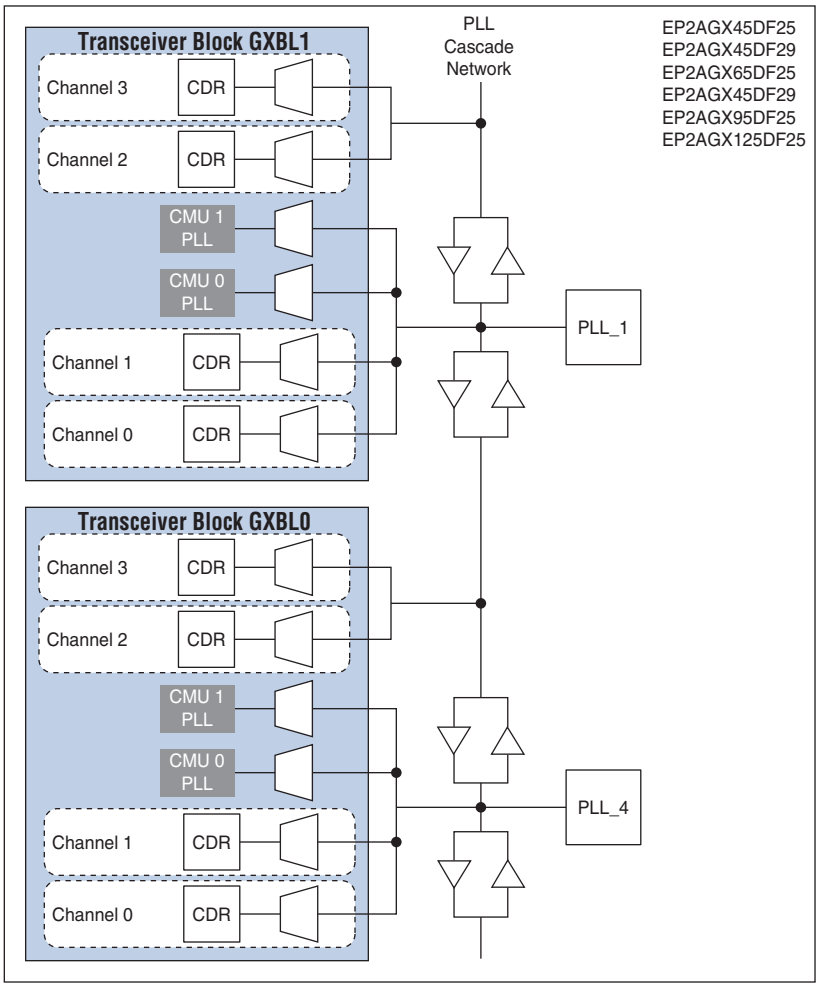


The FPGA fabric PLLs-transceiver PLLs cascading option is available in the following Arria II GX devices with eight channels:

- EP2AGX45DF25
- EP2AGX45DF29
- EP2AGX65DF25
- EP2AGX45DF29
- EP2AGX95DF25
- EP2AGX125DF25

Figure 2–32 shows the FPGA fabric PLLs-transceiver PLLs cascading option allowed in the EP2AGX45DF25, EP2AGX45DF29, EP2AGX65DF25, EP2AGX45DF29, EP2AGX95DF25, and EP2AGX125DF25 devices.

Figure 2–32. FPGA Fabric PLLs-Transceiver PLLs Cascading Option Allowed in the EP2AGX45DF25, EP2AGX45DF29, EP2AGX65DF25, EP2AGX45DF29, EP2AGX95DF25, and EP2AGX125DF25 Devices

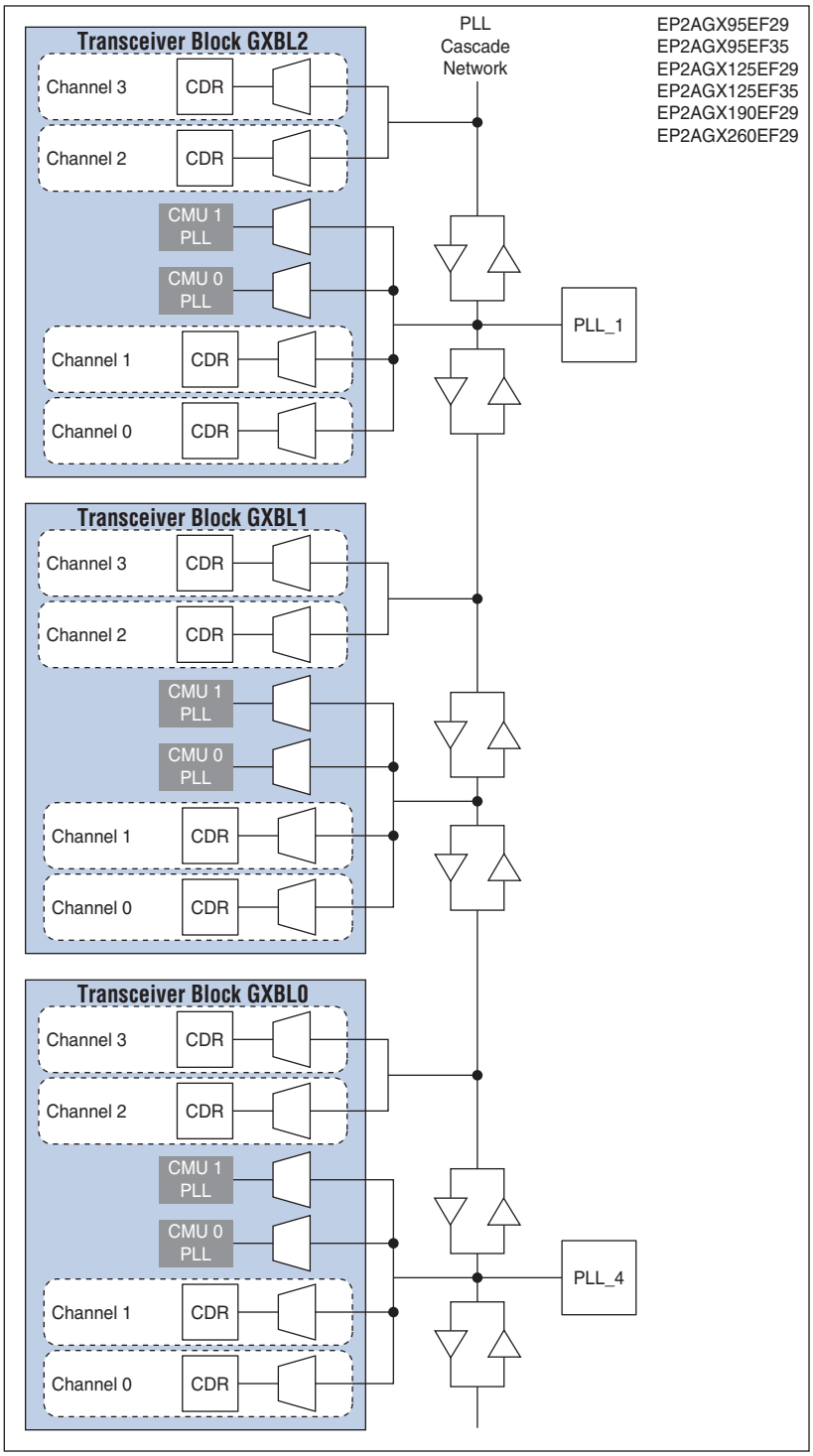


The FPGA fabric PLLs-transceiver PLLs cascading option is available in the following Arria II GX devices with twelve channels:

- EP2AGX95EF29
- EP2AGX95EF35
- EP2AGX125EF29
- EP2AGX125EF35
- EP2AGX190EF29
- EP2AGX260EF29

Figure 2–33 shows the FPGA fabric PLLs-transceiver PLLs cascading option allowed in the EP2AGX95EF29, EP2AGX95EF35, EP2AGX125EF29, EP2AGX125EF35, EP2AGX190EF29, and EP2AGX260EF29 devices.

Figure 2–33. FPGA Fabric PLLs Transceiver PLLs Cascading Option Allowed in the EP2AGX95EF29, EP2AGX95EF35, EP2AGX125EF29, EP2AGX125EF35, EP2AGX190EF29, and EP2AGX260EF29 Devices

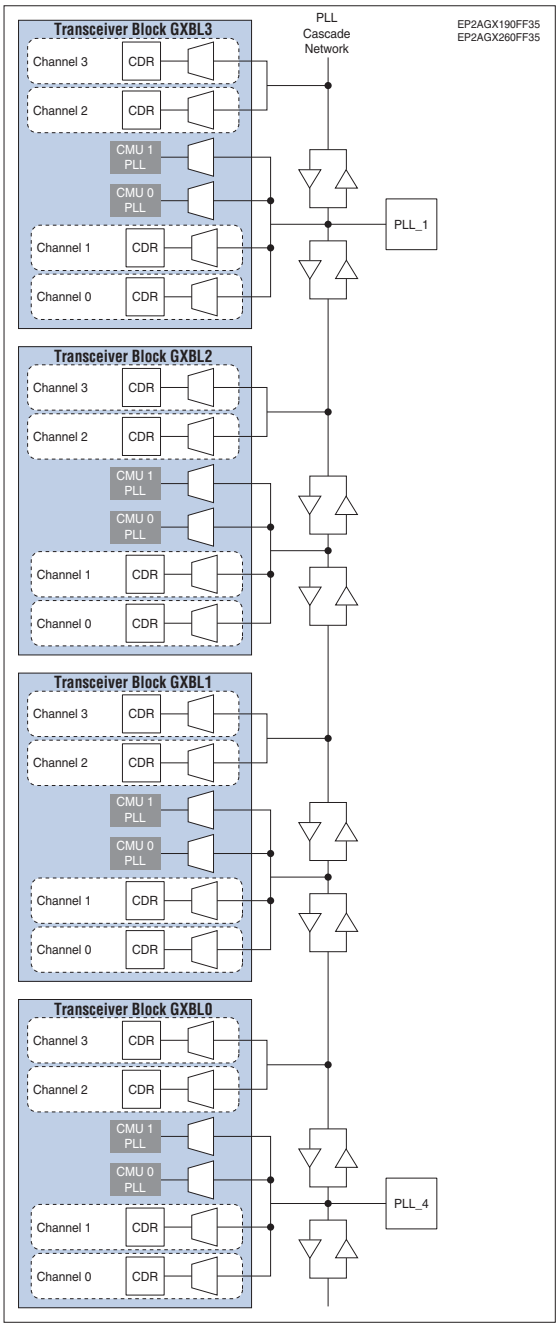


The FPGA fabric PLLs-transceiver PLL's cascading option is available in the following Arria II GX devices with sixteen channels:

- EP2AGX190FF35
- EP2AGX260FF35

Figure 2–34 shows the FPGA fabric PLLs-transceiver PLLs cascading option allowed in the EP2AGX190FF35 and EP2AGX260FF35 devices.

Figure 2–34. FPGA Fabric PLLs-Transceiver PLL's Cascading Option Allowed in the EP2AGX190FF35 and EP2AGX260FF35 Devices



Dedicated Left and Right PLL Cascade Network in Arria II GZ Devices

Arria II GZ devices have a dedicated PLL cascade network on the left and right side of the device that connects to the input reference clock selection multiplexer of the CMU PLLs, and receiver CDRs on the left and right 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 or more left and right PLLs to drive the cascade clock line simultaneously.

Because the number of left and right PLLs and transceiver blocks vary from device to device, the capability of cascading a left and right PLL to the CMU PLLs, and receiver CDRs also varies from device to device.

The following sections describe the Arria II GZ FPGA fabric-transceiver PLL's cascading for the various device packages.

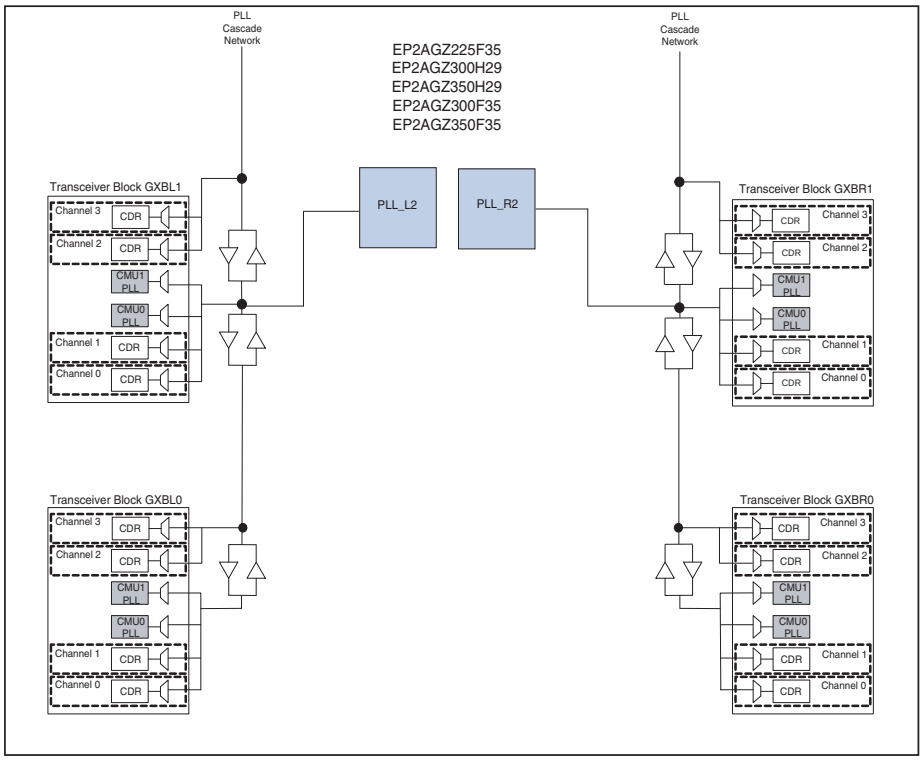
FPGA Fabric PLLs-Transceiver PLLs Cascading in the 780-Pin Package

Arria II GZ devices in 780-pin packages do not support FPGA fabric PLLs-transceiver PLL's cascading.

FPGA Fabric PLLs-Transceiver PLLs Cascading in the 1152-Pin Package

Figure 2–35 shows the FPGA fabric PLLs-transceiver PLL's cascading options allowed in the EP2AGZ300H29, EP2AGZ350H29, EP2AGZ225FF35, EP2AGZ300FF35, and EP2AGZ350FF35 devices.

Figure 2–35. FPGA Fabric PLLs-Transceiver PLLs Cascading Options Allowed for 1152-Pin Package Devices

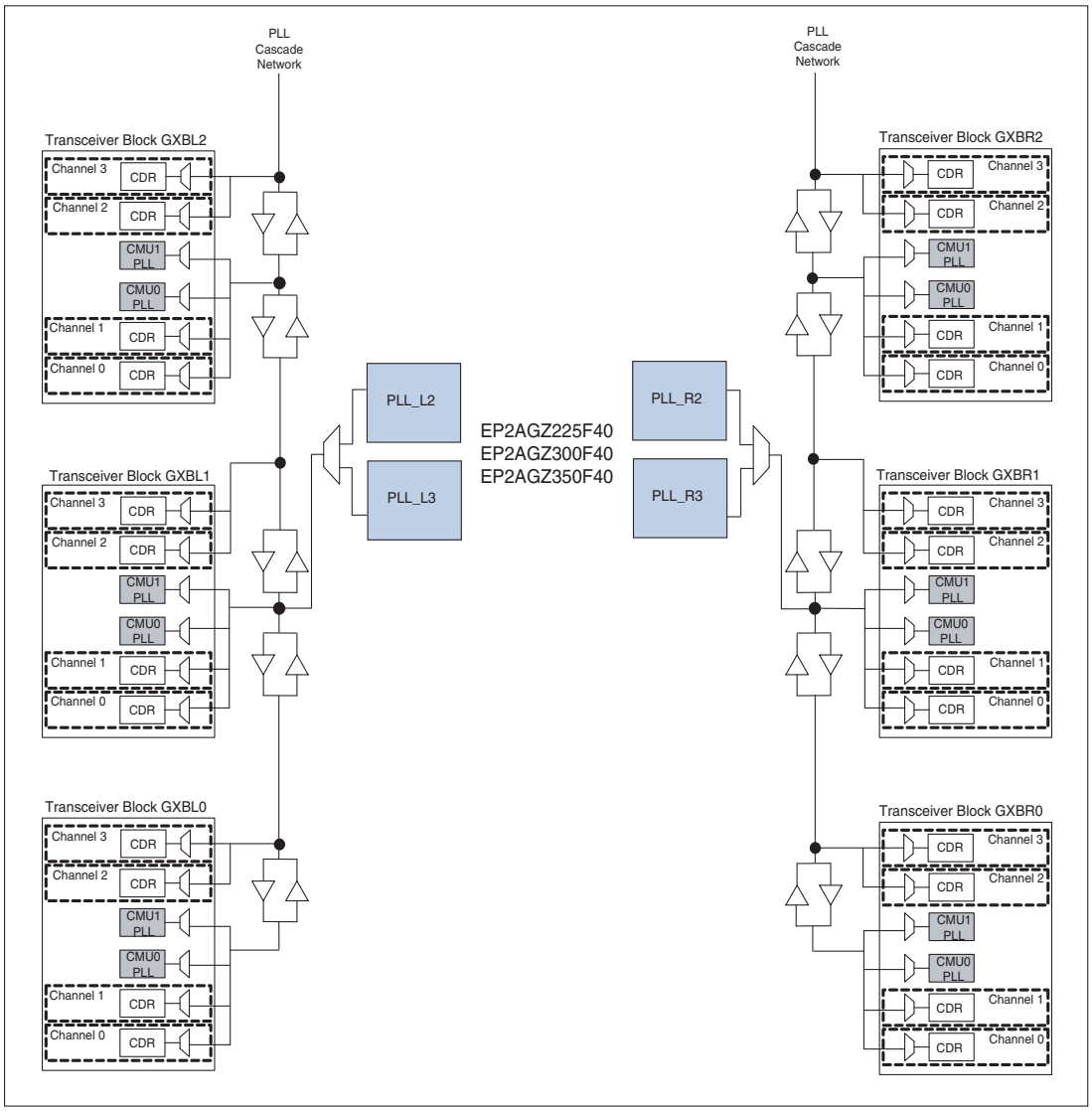


FPGA Fabric PLLs-Transceiver PLLs Cascading in the 1517-Pin Package

Figure 2–36 shows the FPGA fabric PLLs-transceiver PLL's cascading options allowed in the EP2AGZ225F40, EP2AGZ300F40, and EP2AGZ350F40 devices.

For the EP4S40G2KF40, EP4S40G5KF40, EP4S100G2KF40, and EP4S100G5KF40 devices, FPGA fabric PLLs-Transceiver PLLs cascading for the CMU PLLs is the same as the Arria II devices in the 1517-pin package.

Figure 2–36. FPGA Fabric PLLs-Transceiver PLLs Cascading Options Allowed in the 1517-Pin Package Devices



FPGA Fabric PLL-Transceiver PLL 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

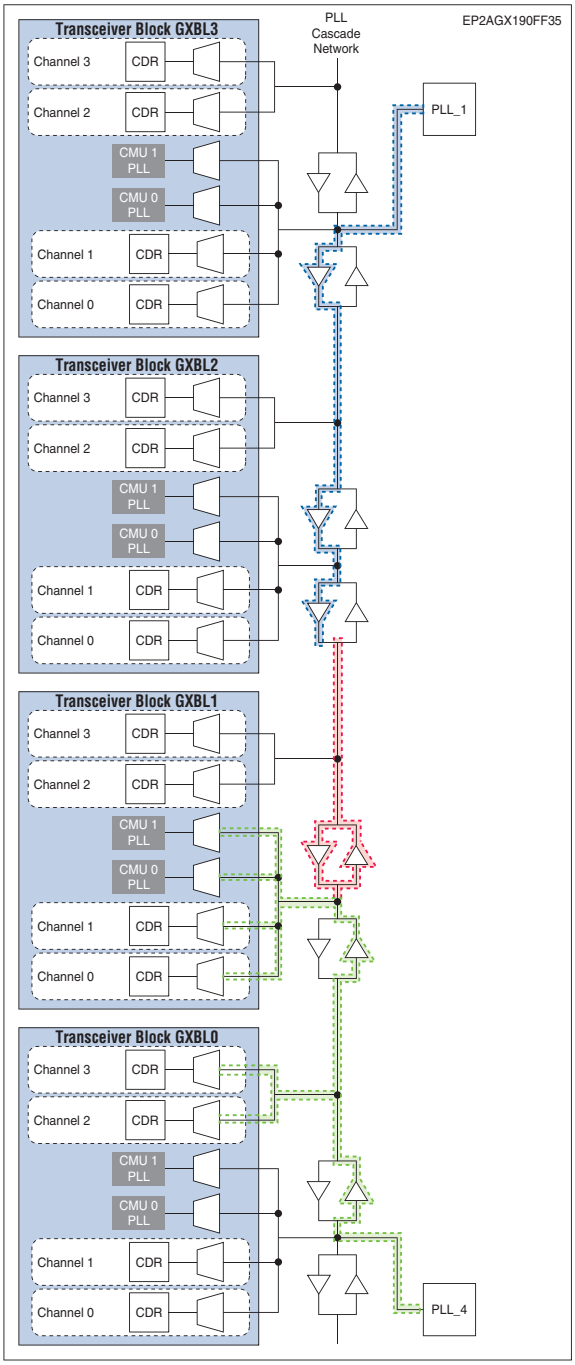
Consider a design targeting the EP2AGX190FF35 device and requiring input reference clocks to the following CMU PLLs and receiver CDRs from two left PLLs in the FPGA fabric:

- `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 2–37 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.

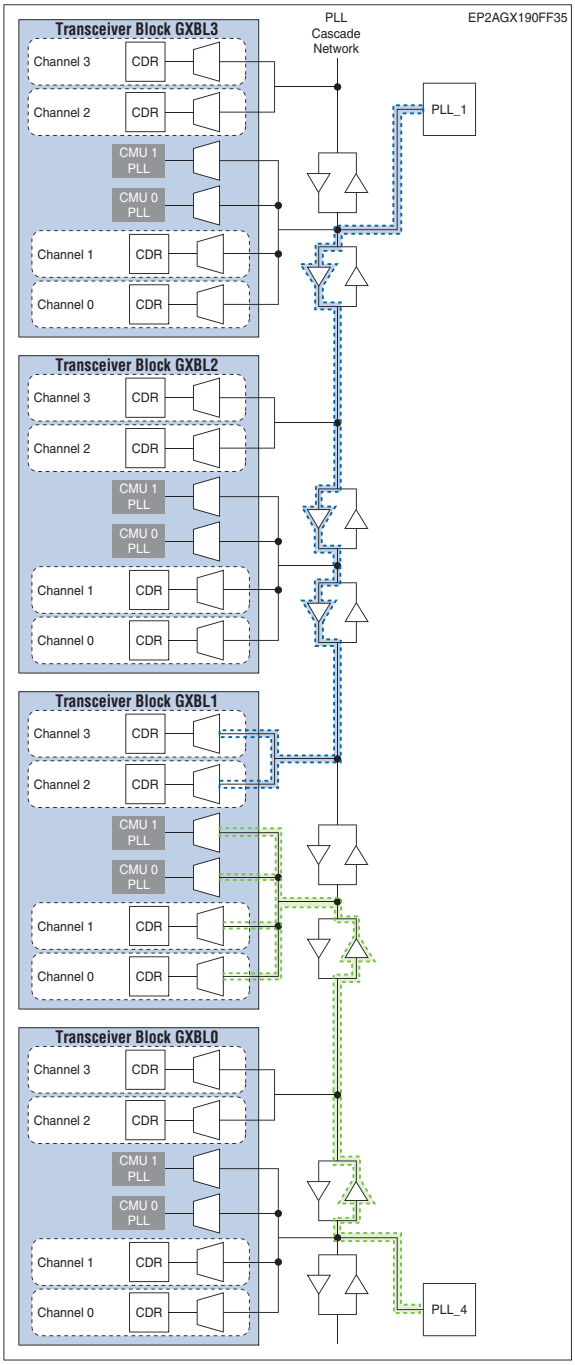
Figure 2–37. Illegal FPGA Fabric-Transceiver PLL Cascading Configuration



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 2–38 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 2–38. Legal FPGA Fabric-Transceiver PLL Cascading Configuration



Using the CMU PLL for Clocking User Logic in the FPGA Fabric

Some designs that use multiple clock domains may run out of PLLs in the FPGA fabric. In such a scenario, if your design has CMU PLLs that are not being used, it may be possible to use them for clocking user logic in the FPGA fabric. However, the CMU PLLs do not have many features that are supported by the PLLs in the FPGA fabric.

The following features are supported on CMU PLLs used as PLLs for clocking user logic in the FPGA fabric:

- Single clock output
- Programmable PLL bandwidth
- PLL PFD power down control
- Lock status signal

To use this feature, you must create an ALTGX instance with a single channel in **Transmitter Only** mode that uses the required CMU PLL. To create the ALTGX instance, complete these steps:

1. Choose **Basic mode** as the protocol.
2. Select **Transmitter Only** operation mode.
3. Select the input clock frequency.
4. Select the appropriate values of data rate and channel width based on the desired output clock frequency. To generate a 250 MHz clock using an input clock frequency of 50 MHz, select a channel width of **10** and a data rate of **2500 Mbps** (Equation 2–1).

Equation 2–1.

$$f_{out} = \frac{\text{data rate}}{\text{channel width}}$$

5. You can select the PLL bandwidth by choosing **Tx PLL bandwidth mode**.
6. You can instantiate the `pll_locked` port to indicate the PLL lock status.
7. You can instantiate `pll_powerdown` or `gxb_powerdown` to enable the PLL PFD power down control.

Use `tx_clkout` of the ALTGX instance as the clock source for clocking user logic in the FPGA fabric.

Document Revision History

Table 2–17 lists the revision history for this chapter.

Table 2–17. Document Revision History

Date	Version	Changes
June 2011	3.1	■ Updated Figure 2–11, Figure 2–12, and Figure 2–20. ■ Updated "Bonded Channel Configurations", "CMU PLL and Receiver CDR Input Reference Clocking" and "Input Reference Clocks" sections. ■ Minor text edits.
December 2010	3.0	■ Updated to add Arria II GZ information. ■ Removed "QL" designations. ■ Minor text edits.
July 2010	2.0	■ Updated Figure 2–29. ■ Updated Table 2–2, Table 2–3, and Table 2–4. ■ Updated the "Non-Bonded Channel Configurations", "Non-Bonded Receiver Clocking without Rate Matcher", and "Dedicated Left PLL Cascade Lines Network" sections. ■ "(QLn)" (where n = 1 or 2) was added to all "GXBn" references. ■ Minor text edits.
November 2009	1.1	■ Updated figures in the document for clarity. ■ Added "Using the CMU PLL for Clocking User Logic in the FPGA Fabric" on page 2–64.
February 2009	1.0	Initial release.

3. Configuring Multiple Protocols and Data Rates in Arria II Devices

AIIGX52003-3.0

This chapter describes the configuration of multiple protocols and data rates for Arria® II GX and GZ devices. Each transceiver channel in an Arria II GX or GZ 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" on page 3–1
- "Creating Transceiver Channel Instances" on page 3–2
- "General Requirements to Combine Channels" on page 3–2
- "Sharing CMU PLLs" on page 3–3
- "Combining Receiver Only Channels" on page 3–8
- "Combining Transmitter Channel and Receiver Channel Instances" on page 3–9
- "Combining Channels Configured in Protocol Functional Modes" on page 3–10
- "Combining Transceiver Instances Using PLL Cascade Clocks" on page 3–12
- "Combining Transceiver Instances in Multiple Transceiver Blocks" on page 3–13
- "Summary" on page 3–15

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 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.

© 2010 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Creating Transceiver Channel Instances

You can instantiate multiple transceiver channels in the **General** screen of the ALTGX MegaWizard Plug-In Manager in the following two different 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 require 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 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 describe 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 you disable the `gxb_powerdown` port, the Quartus II software ties the port to ground.

Calibration Clock and Power Down

Each calibration block in an Arria II GX or GZ 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 *Transceiver Architecture in Arria II Devices* chapter.

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 and GZ 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. Different settings are useful 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 a 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 lists the configuration for Example 1.

Table 3–1. Configuration for Example 1 for Arria II Devices (Part 1 of 2)

User-Created Instance Name	ALTGX MegaWizard Plug-In Manager Settings		
	Number of Channels	Configuration	Effective Data Rate (Gbps)
inst0	1	Receiver and Transmitter	3.75
inst1	1	Receiver and Transmitter	0.9375

Table 3–1. Configuration for Example 1 for Arria II Devices (Part 2 of 2)

User-Created Instance Name	ALTGX MegaWizard Plug-In Manager Settings		
	Number of Channels	Configuration	Effective Data Rate (Gbps)
inst2	1	Receiver and Transmitter	1.875
inst3	1	Receiver and Transmitter	3.75

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 describe 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



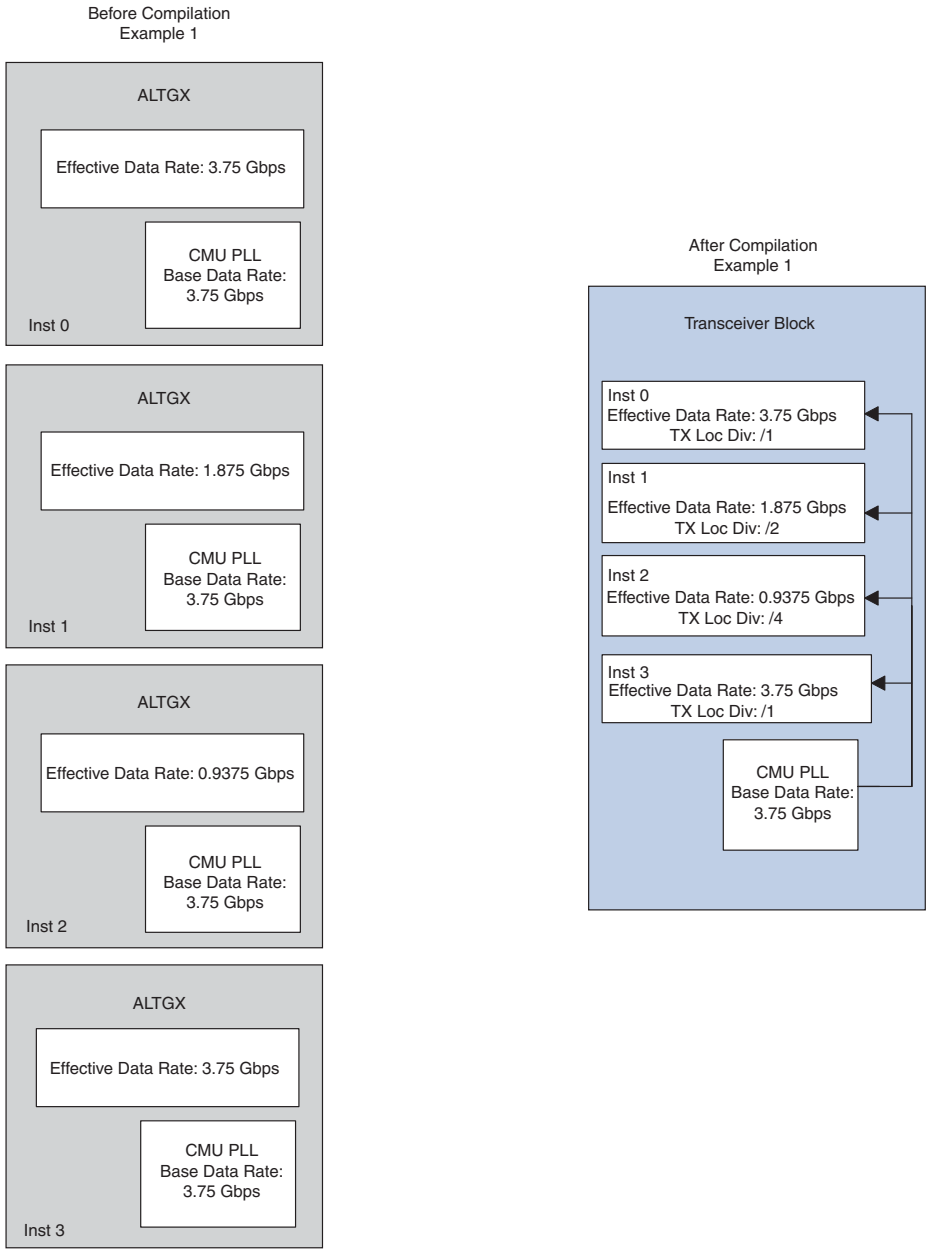
The **Specify base data rate** option is 3.75 Gbps for all four instances. Because the CMU PLL bandwidth setting and input reference clock are the same and the `pll_powerdown` ports are driven from the same logic or pin, the Quartus II software shares a single CMU PLL that runs at 3.75 Gbps.

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.

Use a second CMU PLL if you want to combine channels that require different configurations, such as:

- Quartus II software-defined protocols (for example, Basic, Gbps Ethernet (GbE), SONET/synchronous digital hierarchy [SDH], Serial Digital Interface [SDI], or PCI Express® [PIPE] [PCIe] 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 for Arria II Devices

User-Created Instance Name	ALTGX MegaWizard Plug-In Manager Settings		
	Number of Channels	Configuration	Effective Data Rate (Gbps)
inst0	1	Receiver and Transmitter	3.75
inst1	1	Receiver and Transmitter	1.875
inst2	1	Receiver and Transmitter	0.9375
inst3	1	Receiver and Transmitter	2

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. For the ALTGX MegaWizard Plug-In Manager settings that enable the Quartus II software to share the same CMU PLL, refer to "Example 1" on page 3–3. A second CMU PLL is required for instance 3.

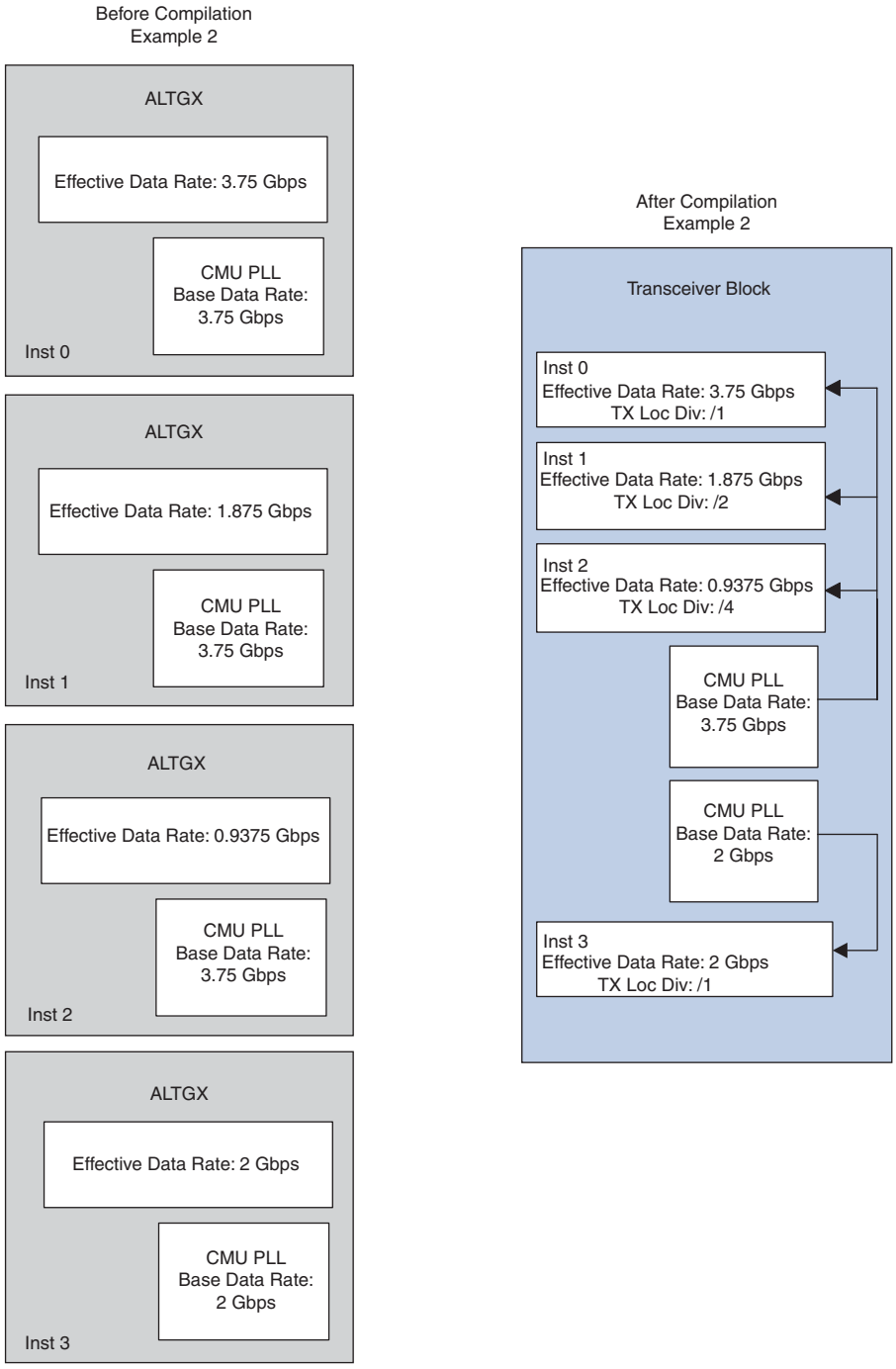
You can force the placement of transceiver channels to a specific transceiver block by assigning pins to the `tx_dataout` and `rx_datain` pins of the four ALTGX instances. Otherwise, the Quartus II software selects a transceiver block.

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 for Arria II Devices

User-Created Instance Name	ALTGX MegaWizard Plug-In Manager Settings			
	Number of Channels	Configuration	Effective Data Rate (Gbps)	Base Data Rate (Gbps)
inst0	1	Receiver and Transmitter	1.5	1.5
inst1	1	Receiver and Transmitter	1.5	3.0
inst2	1	Receiver and Transmitter	1	1

Even though the effective data rate of inst1 and inst0 are 1.5 Gbps (3 Gbps/2 = 1.5 Gbps), 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. Because 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 have 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 for Arria II Devices

Instance Name	Configuration	Effective Data Rate (Gbps)	Input Reference Clock Frequency (MHz)
inst0	Transmitter only	3.125	156.25
inst1	Receiver only	2.5	156.25
inst2	Transmitter only	1.25	125
inst3	Receiver only	2	125

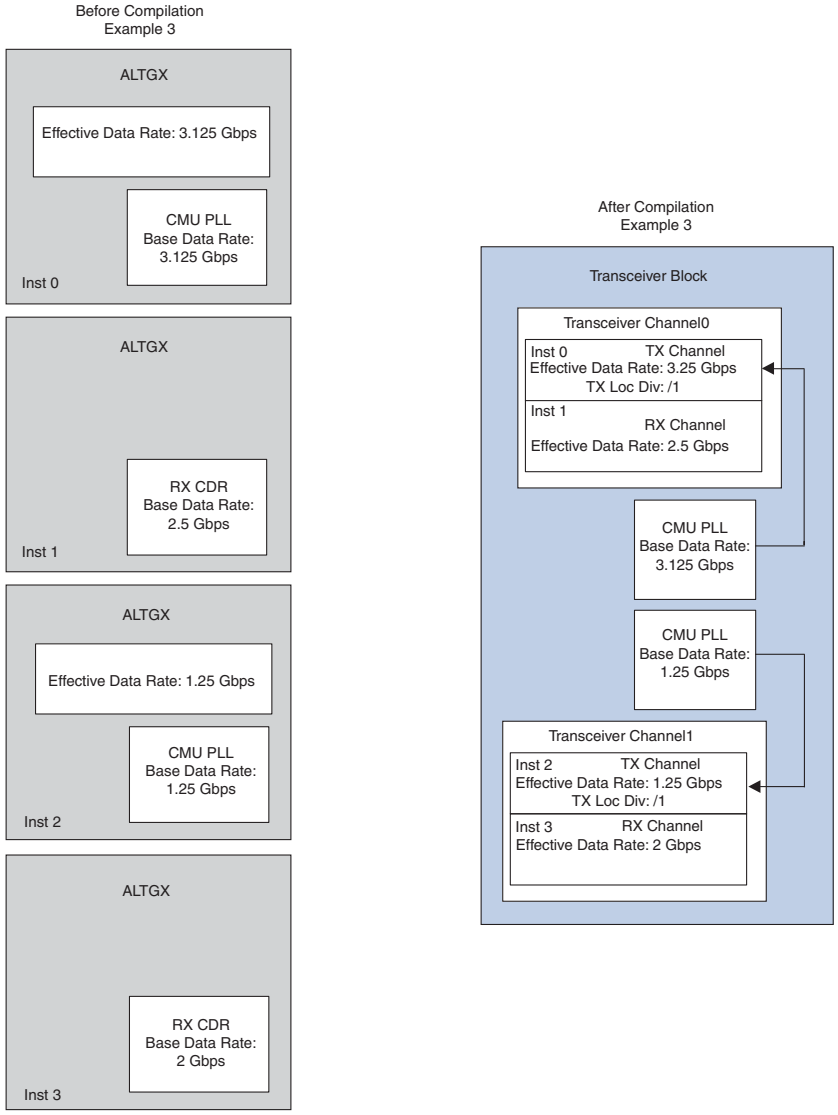
After you create the above instances, if you force the placement of the instances shown in Table 3–5, the Quartus II software combines inst0 and inst1 into physical channel 0 and inst2 and inst3 into physical channel 1.

Table 3–5. Forced Placement of the Instances for Example 3 for Arria II Devices

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 ×4 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 cannot be used within the transceiver block. 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.

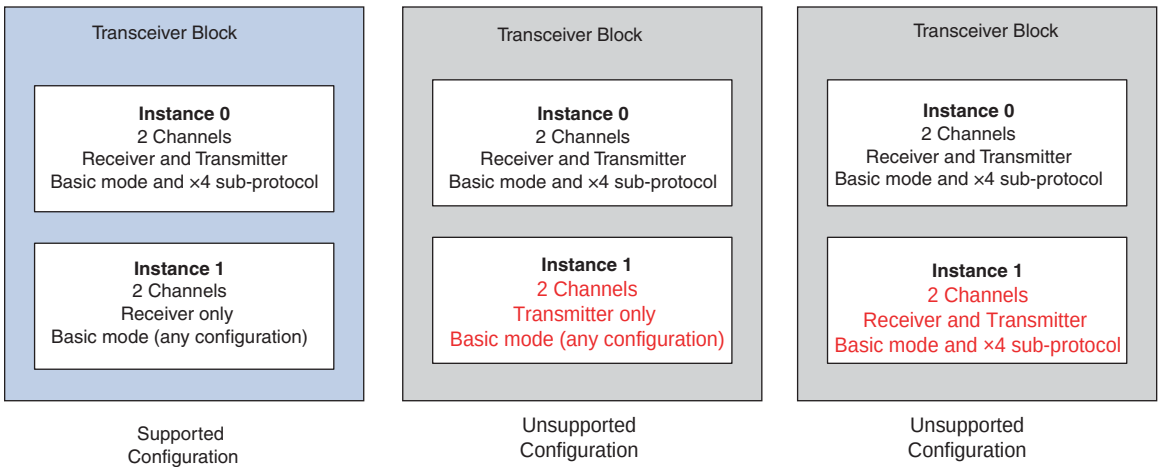
Table 3–6. General Screen Options in Basic ×4 Mode for Arria II Devices

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 PCIe hard IP Block with Other Channels

The Arria II device family contains an embedded PCIe hard IP block that performs the physical, datalink, and transaction layer functionality specified by PCIe base specification 1.1. Each PCIe hard IP block is shared by two transceiver blocks. The PCIe Compiler MegaWizard Plug-In Manager provides you the options to configure the PCIe 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 the PCIe 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 PCIe Compiler MegaWizard Plug-In Manager. Table 3–7 lists the restrictions.

Table 3–7. PCIe Hard IP Block Restrictions When Combining Transceiver Channels with Different Functional and/or Protocol Modes for Arria II Devices *(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 PCIe ×4 or ×8 configuration.

(4) Transceiver block 1—the adjacent transceiver block that shares the same PCIe 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 PCIe functional mode, refer to the "×8 Channel Configuration" section in the *Arria II Transceiver Clocking* chapter.

For more information about the PCIe 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 device family 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 and GZ transceiver can cascade the output of the general purpose PLLs to the CMU PLLs and receiver CDRs. The left side of the Arria II GX and GZ device contains a PLL cascade clock network—a single line network that connects the PLL cascade clock to the transceiver block. Similarly, for Arria II GZ devices, the right side PLLs can only be cascaded with the transceivers on the right side of the device. Each side of the Arria II GZ device contains a PLL cascade clock network. 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 *Transceiver Clocking in Arria II Devices* chapter.

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, as 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 for Arria II Devices

Instance Name	Number of Transceiver Channels	Configuration	Effective Data Rate (Gbps)	Input Reference Clock (Gbps)
inst0	7	Receiver and Transmitter	3.125	156.25
inst1	1	Receiver and Transmitter	3.125	156.25

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

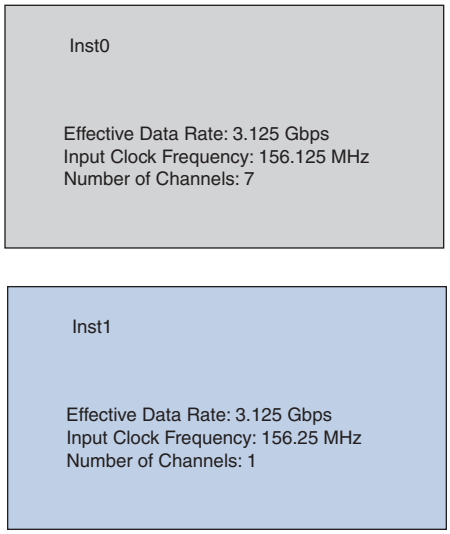
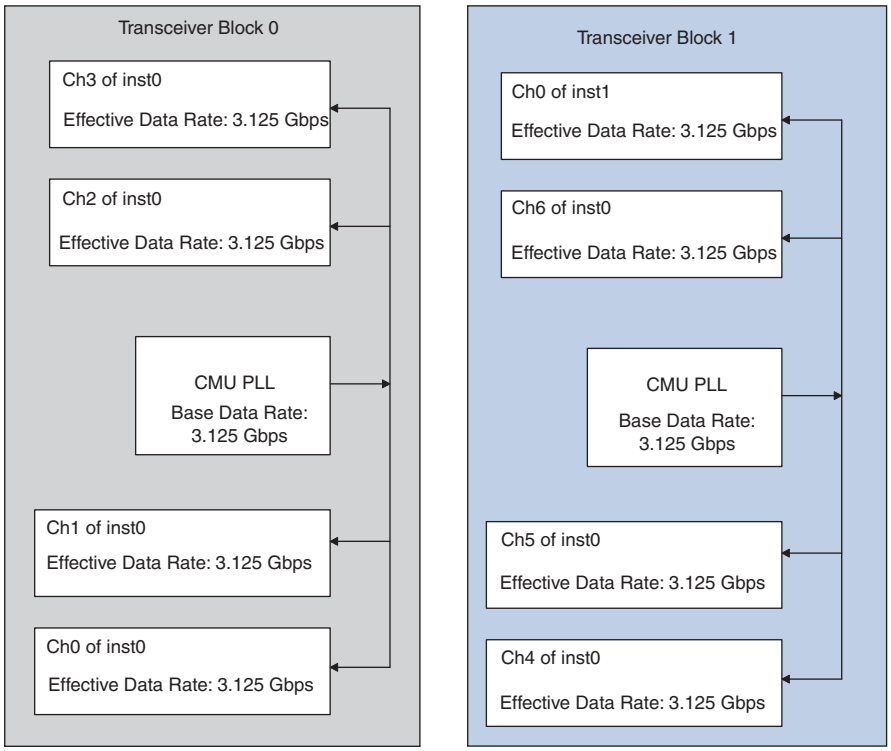


Figure 3–6 shows the transceiver instances after compilation for Example 4.

Figure 3–6. Combined Transceiver Instances after Compilation for Example 4



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 to enable 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 on page 3–14.

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), allowing 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 PCIe hard IP block using the PCIe 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 PCIe hard IP Block with Other Channels" on page 3–12.

Document Revision History

Table 3–9 lists the revision history for this chapter.

Table 3–9. Document Revision History

Date	Version	Changes
December 2010	3.0	■ Updated to add Arria II GZ information. ■ Minor text edits.
July 2010	2.0	■ Updated the "Transceiver PLL Configurations" and "Combining Channels Using the PCIe hard IP Block with Other Channels" sections. ■ Minor text edits.
February 2009	1.0	Initial release.

4. Reset Control and Power Down in Arria II Devices

AIIGX52004-3.2

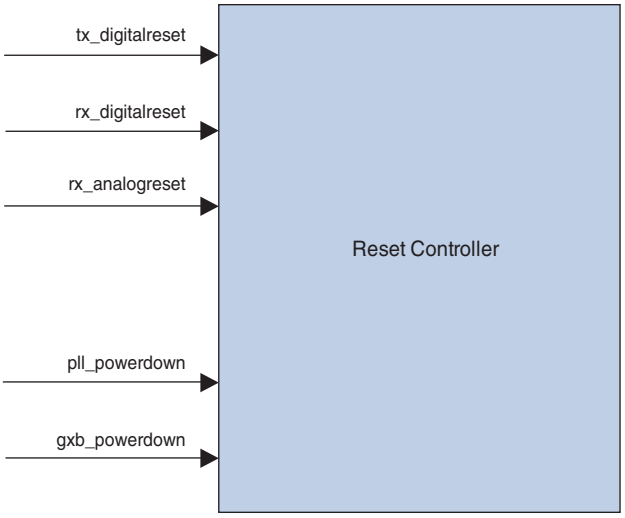
Arria® II GX and GZ devices offer multiple reset signals to control transceiver channels and clock multiplier unit (CMU) phase-locked loops (PLLs) independently. The ALTGX Transceiver MegaWizard™ Plug-In Manager provides individual reset signals for each channel instantiated in the design. It also provides 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
- "Dynamic Reconfiguration Reset Sequences" on page 4–17
- "Hot Socketing Reset Sequence" on page 4–19
- "Power Down" on page 4–20
- "Simulation Requirements" on page 4–21

Figure 4–1 shows the reset control and power-down block for an Arria II GX or GZ device.

Figure 4–1. Reset Control and Power-Down Block



User Reset and Power-Down Signals

Each transceiver channel in the Arria II device family 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.

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

☞ All reset and power-down signals are asynchronous.

Table 4–1 lists the available reset, power-down, and status signals.

Table 4–1. Reset, Power-Down, and Status Signals for Arria II Devices (Part 1 of 2)

Signal	Description
Reset Signals For Each Transceiver Channel:	
tx_digitalreset *(1)*	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 *(1)*	Resets all digital logic in the receiver PCS, including the XAUI and GbE 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.
rx_analogreset	Resets the receiver CDR, receiver deserializer, and signal detect in the receiver buffer. 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 busy signal has precedence over the rx_analogreset assertion. When the busy signal is high, rx_analogreset is ignored.
Power-Down Signal For Each CMU PLL in the Transceiver Block:	
pll_powerdown *(2)*	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. Note: While each CMU PLL has its own pll_powerdown port, the ALTGX MegaWizard Plug-In Manager instantiation provides only one port per transceiver block. This port power downs one or both CMU PLLs (if used).
Power-Down Signal Common to the Transceiver Block:	
gxb_powerdown *(2)*	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
Status Signals:	
pll_locked	Indicates the status of the transmitter PLL. A high level on this signal indicates that the transmitter PLL is locked to the incoming reference clock frequency.
rx_pll_locked	A high level on this signal indicates 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.

Table 4–1. Reset, Power-Down, and Status Signals for Arria II Devices (Part 2 of 2)

Signal	Description
busy	When the busy signal is high during offset cancellation, the signal detect (in the receiver buffer), receiver CDR, and receiver deserializer are enabled regardless of the rx_analogreset signal state. However, the rx_pll_locked and rx_freqlocked signals always deasserted if rx_analogreset is asserted regardless of the busy signal.

Notes to Table 4–1:

(1) 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 the transmitter and receiver phase compensation FIFOs in the PCS.

(2) The refclk (refclk0 or refclk1) buffer is not powered down by this signal.

For more information, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II Devices*.

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 4–2 lists the blocks that are affected by specific reset and power-down signals.

Table 4–2. Blocks Affected by Reset and Power-Down Signals for Arria II Devices (Part 1 of 2)

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		✓ *(1)*			✓
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	✓				✓

Table 4–2. Blocks Affected by Reset and Power-Down Signals for Arria II Devices (Part 2 of 2)

Transceiver Block	rx_digitalreset	rx_analogreset	tx_digitalreset	pll_powerdown	gxb_powerdown
Receiver XAUI State Machine	✓				✓

Note to Table 4–2:

(1) RX signal detect in the RX buffer is disabled by `rx_analogreset`. However, RX signal detect is enabled when the `busy` signal is asserted.

The dynamic reconfiguration controller is always enabled when a receiver exists in a transceiver-based design even if the channel will not be reconfigured. The dynamic reconfiguration controller is used for receiver offset cancellation. Because of this, the `busy` signal from the dynamic reconfiguration controller must always be monitored when instantiating a receiver.

For more information about reset and offset cancellation, refer to the reset waveforms and the "Offset Cancellation" section of *AN 558: Implementing Dynamic Reconfiguration in Arria II Devices*.

Transceiver Reset Sequences

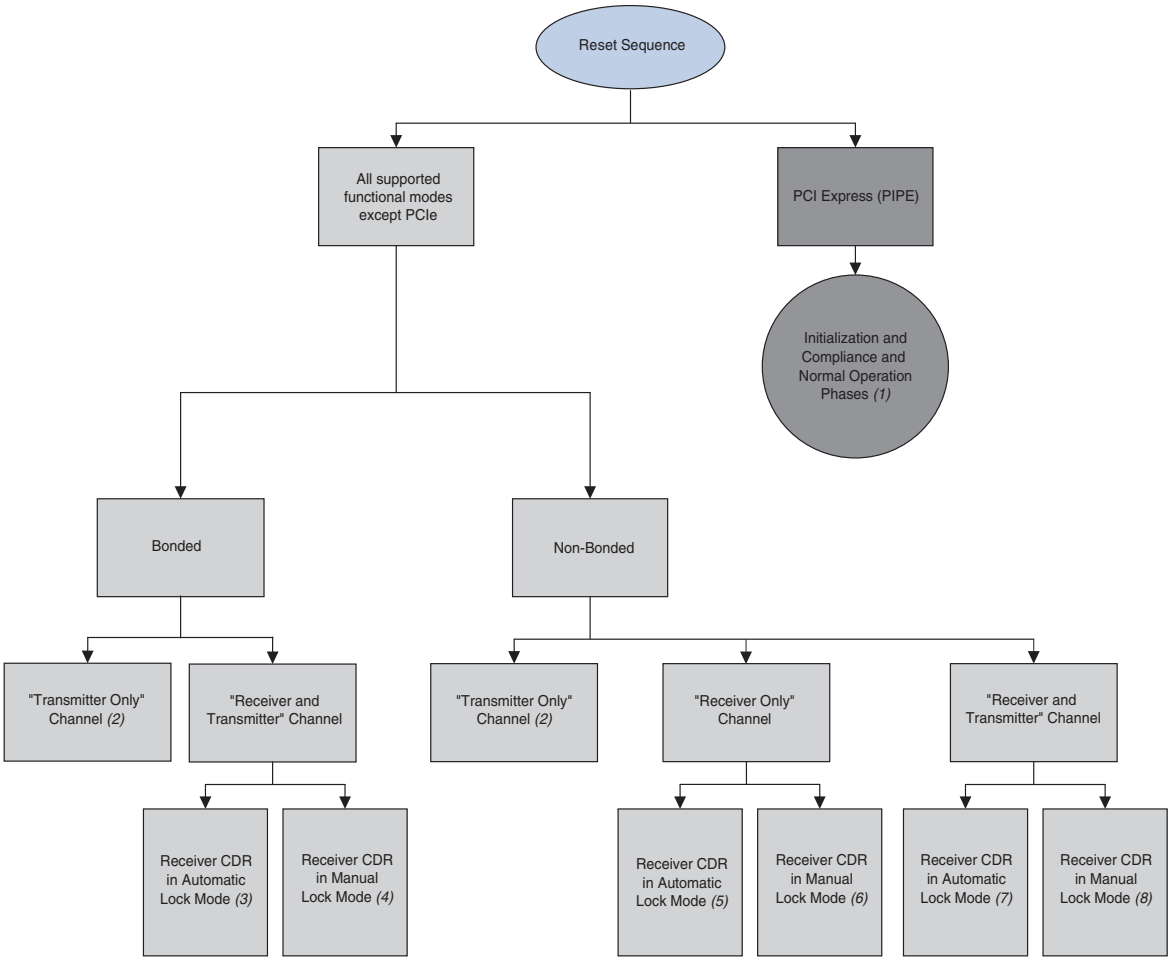
You can configure transceiver channels in Arria II GX and GZ devices in various configurations. In all functional modes except XAUI functional mode, transceiver channels can be either bonded or non-bonded. In XAUI functional mode, transceiver channels must be bonded. In PCI Express® (PIPE) (PCIe®) functional 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 the Arria II device family described in this chapter are:

- "All Supported Functional Modes Except PCIe Functional Mode" on page 4–6 describes the reset sequences in bonded and non-bonded configurations.
- "PCIe Functional Mode" on page 4–15 describes the reset sequence for the initialization/compliance phase and normal operation phase in PCIe functional modes.

The busy signal remains low for the first reconfig_clk clock cycle. It is then 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



Notes to Figure 4–2:

(1) Refer to the Timing Diagram in Figure 4–10 on page 4–15.
(2) Refer to the Timing Diagram in Figure 4–3 on page 4–7.
(3) Refer to the Timing Diagram in Figure 4–4 on page 4–8.
(4) Refer to the Timing Diagram in Figure 4–5 on page 4–9.
(5) Refer to the Timing Diagram in Figure 4–6 on page 4–11.
(6) Refer to the Timing Diagram in Figure 4–7 on page 4–12.
(7) Refer to the Timing Diagram in Figure 4–8 on page 4–13.
(8) Refer to the Timing Diagram in Figure 4–9 on page 4–14.

Altera strongly recommends adhering to these reset sequences for proper operation of the Arria II transceiver.

All Supported Functional Modes Except PCIe 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 of the `rx_locktorefclk` and `rx_locktodata` signals. With the receiver CDR in manual lock mode, you can configure the transceiver channels in the Arria II GX or GZ device either in a non-bonded configuration or a bonded configuration. In a bonded configuration, such as XAUI mode, four channels are bonded together.

Table 4–3 lists the lock-to-reference (LTR) and lock-to-data (LTD) controller lock modes for the `rx_locktorefclk` and `rx_locktodata` signals.

Table 4–3. Lock-To-Reference and Lock-To-Data Modes for Arria II Devices

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, PCIe, and Basic x4 functional modes. In Basic x4 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.

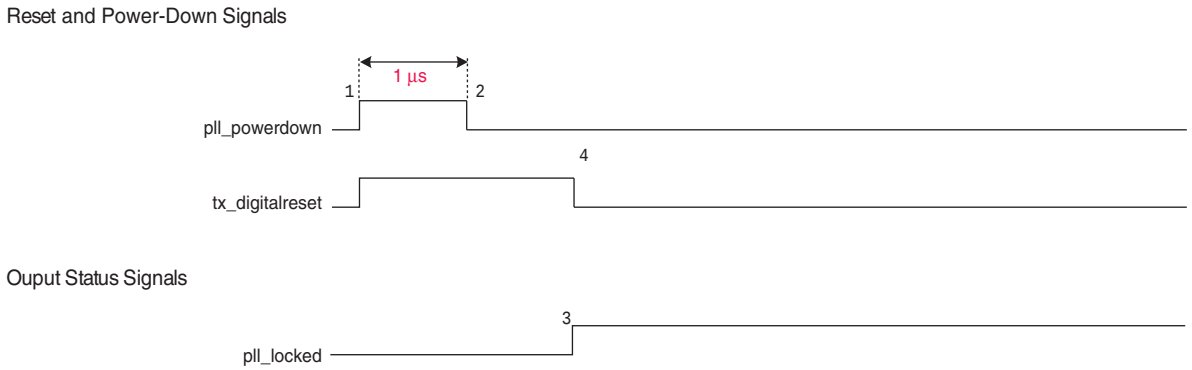
The following timing diagrams describe the reset and power-down sequences for bonded configurations under the following setups:

- **Transmitter Only** channel setup—applicable to Basic x4 functional mode
- **Receiver and Transmitter** channel setup—receiver CDR in automatic lock mode; applicable to XAUI functional mode
- **Receiver and Transmitter** channel setup—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 x4 functional mode, use the reset sequence shown in Figure 4–3.

Figure 4–3. Sample Reset Sequence for Four Transmitter Only Channels



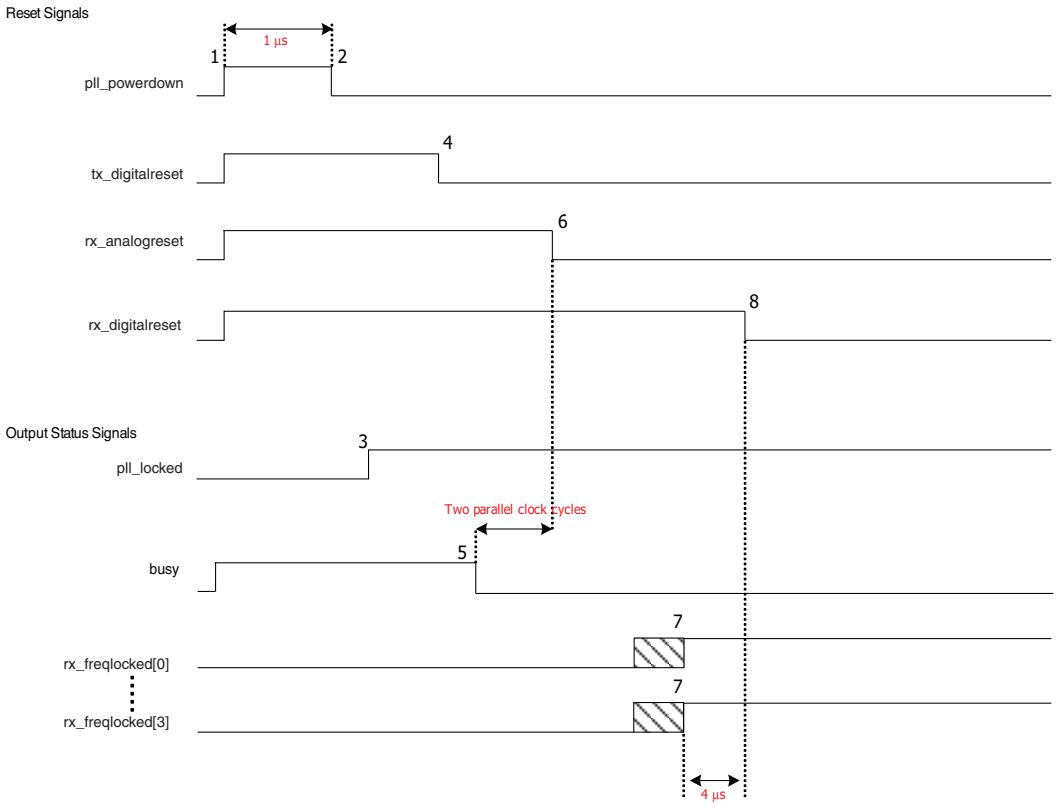
As shown in Figure 4–3, perform the following reset sequence steps for the **Transmitter Only** channel configuration:

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 to transmit 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



As shown in Figure 4–4, perform the following reset sequence steps for the receiver CDR in automatic lock mode configuration:

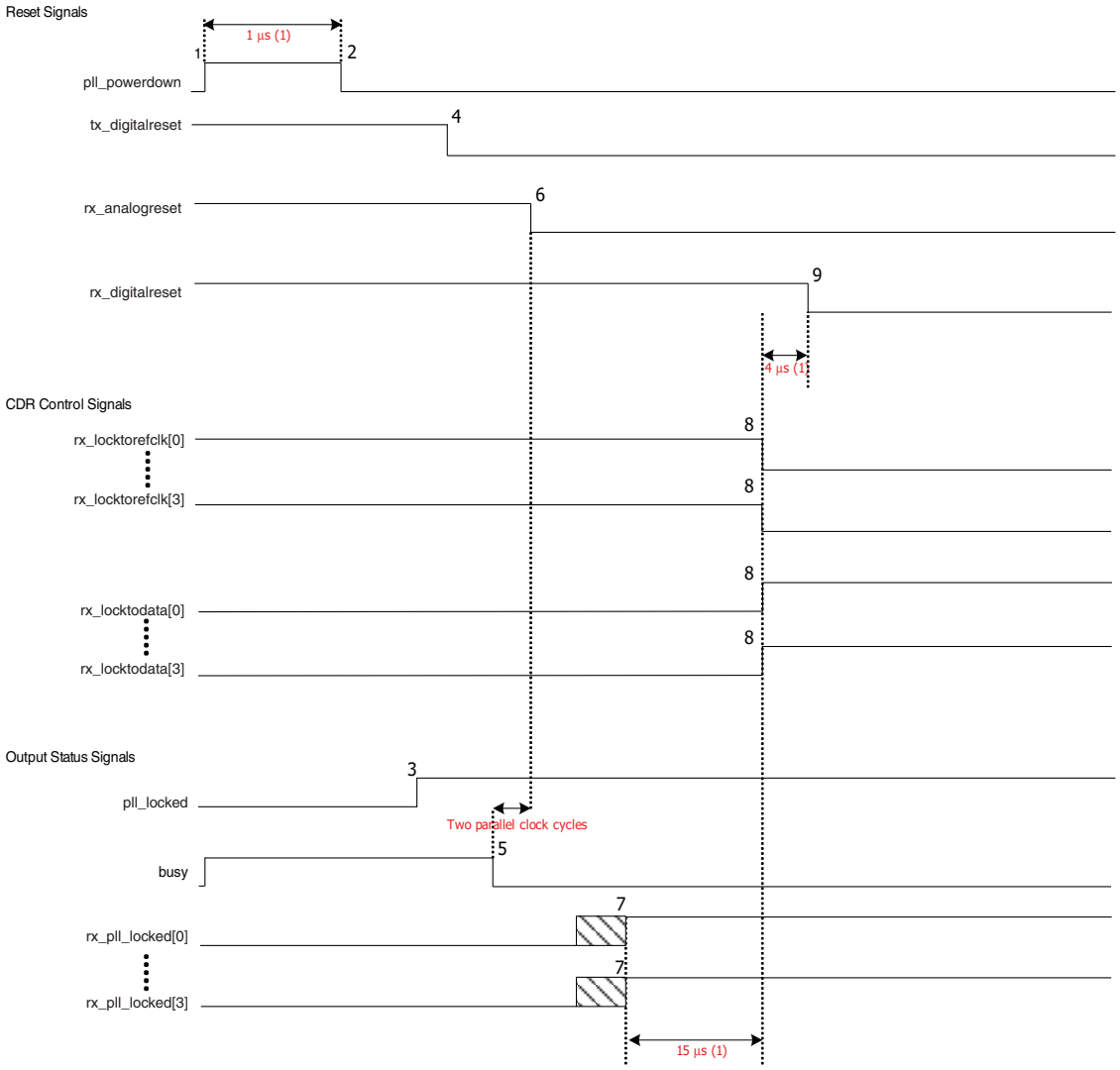
1. After power up, assert `pll_powerdown` for a minimum period of 1 µs (the time between markers 1 and 2).
2. Keep the `tx_digitalreset`, `rx_analogreset`, and `rx_digitalreset` signals asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking 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) For the transceiver block power down duration, refer to the *Device Datasheet for Arria II Devices* chapter.

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 setups:

- **Transmitter Only** channel setup
- **Receiver Only** channel setup—receiver CDR in automatic lock mode
- **Receiver Only** channel setup—receiver CDR in manual lock mode
- **Receiver and Transmitter** channel setup—receiver CDR in automatic lock mode
- **Receiver and Transmitter** channel setup—receiver CDR in manual lock mode



Follow the same reset sequence for all the other channels in the non-bonded configuration.

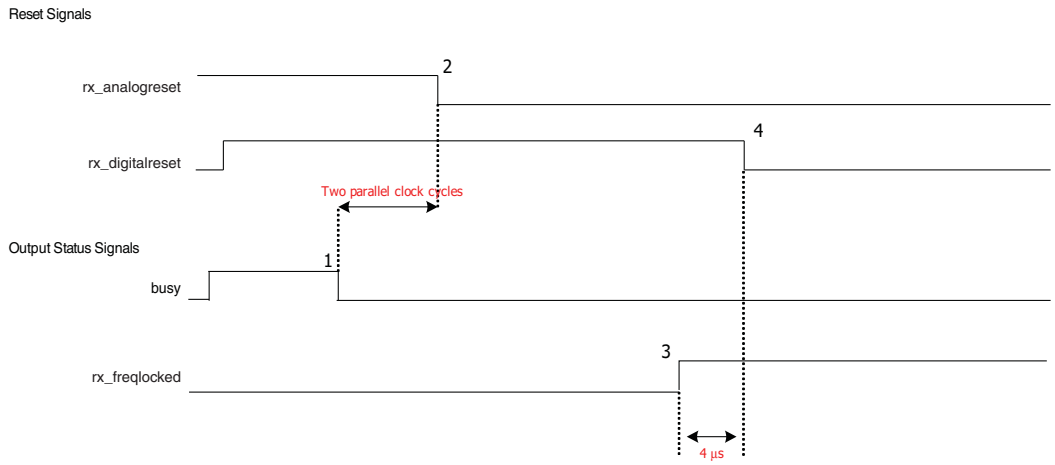
Transmitter Only Channel

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 on page 4–5.

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



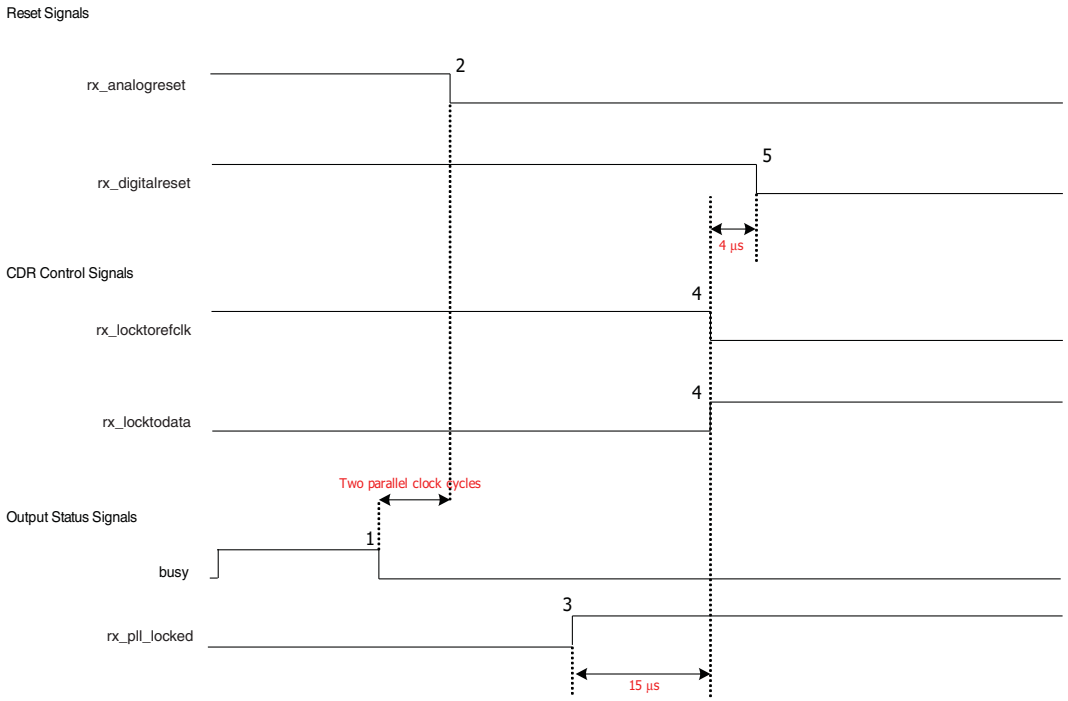
As shown in Figure 4–6, perform the following reset sequence steps for the receiver CDR in automatic lock mode configuration:

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



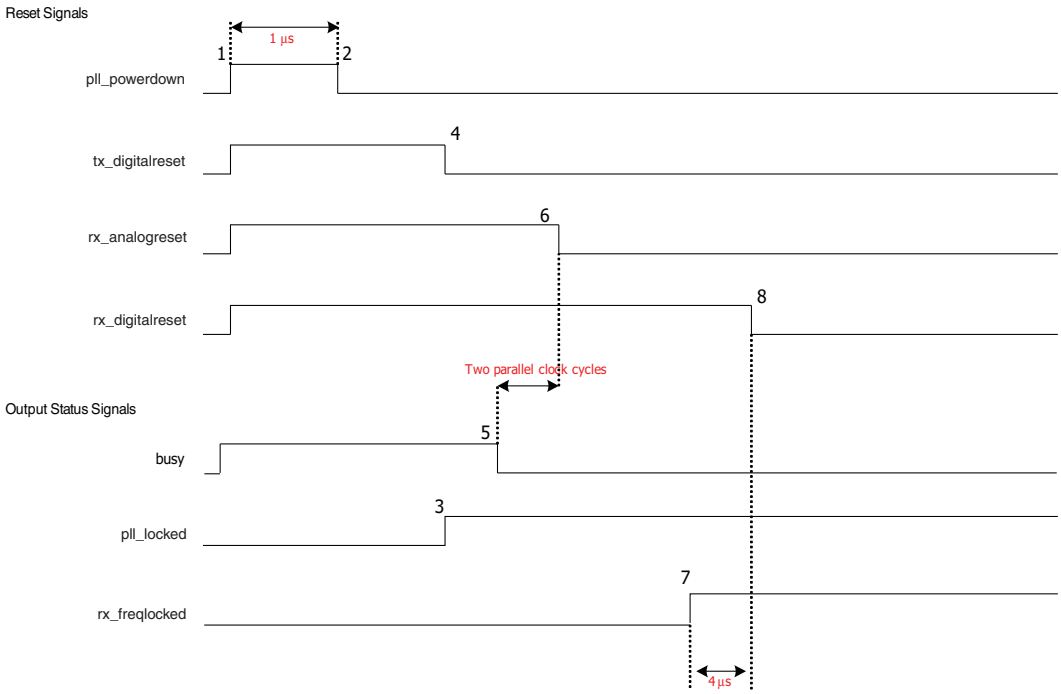
As shown in Figure 4–7, perform the following reset sequence steps for the receiver CDR in manual lock mode:

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



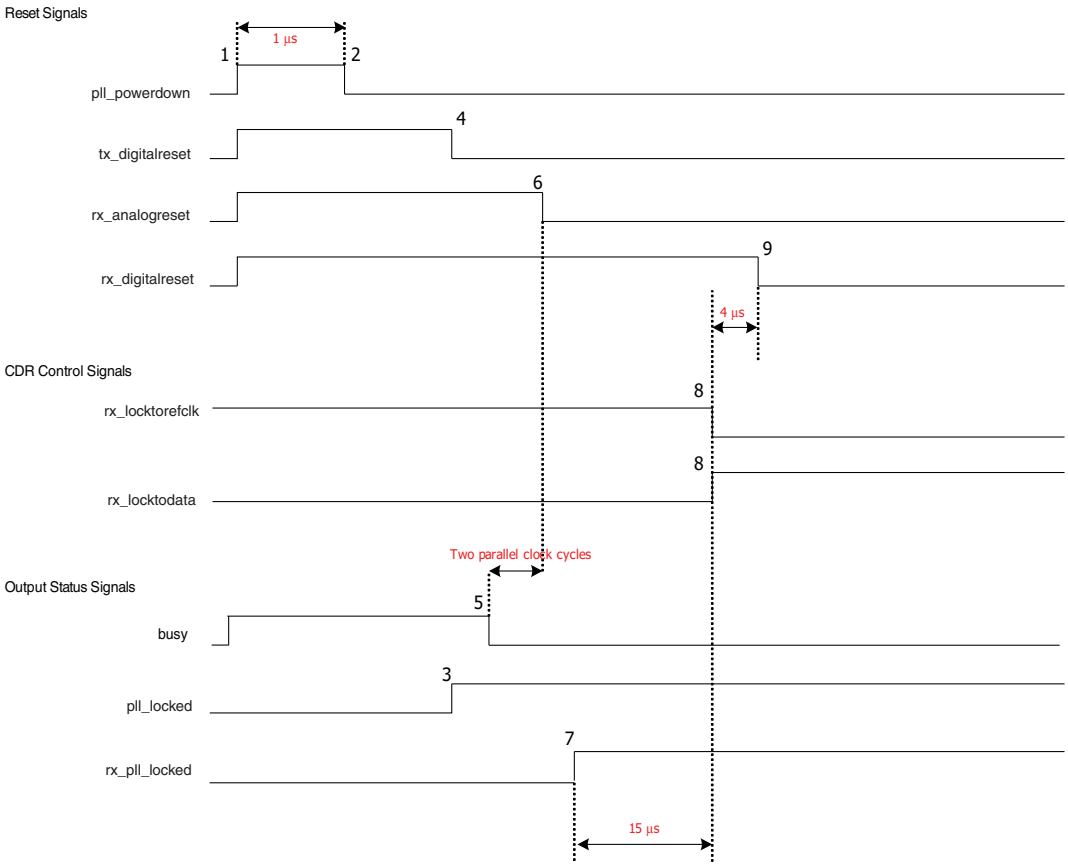
As shown in Figure 4–8, perform the following reset sequence steps for the receiver CDR in automatic lock mode:

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



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.

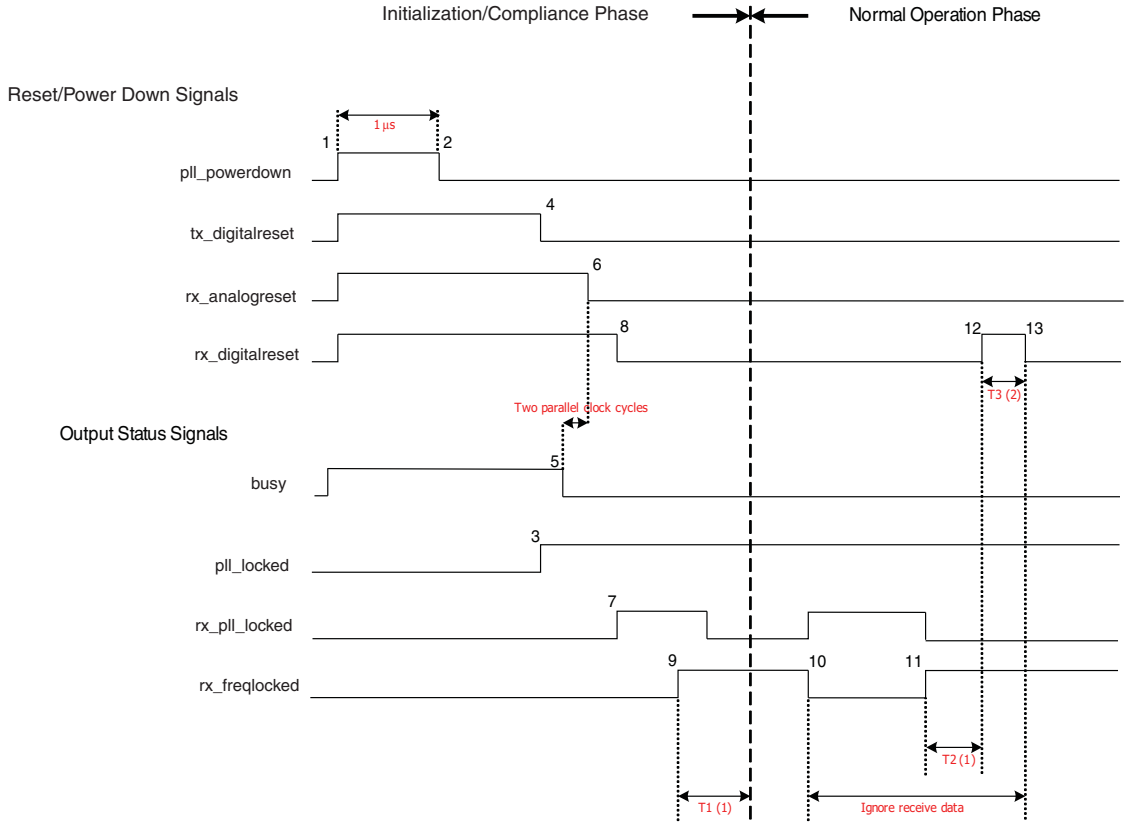
PCIe Functional Mode

You can configure PCIe functional mode with or without the receiver clock rate compensation FIFO in the Arria II device family. The reset sequence remains the same of whether or not you use the receiver clock rate compensation FIFO.

PCIe Reset Sequence

PCIe protocol consists of the initialization/compliance phase and normal operation phase. The reset sequences for these two phases are based on the timing diagram shown in Figure 4–10.

Figure 4–10. Reset Sequence of PCIe Functional Mode



Notes to Figure 4–10:

(1) The minimum T1 and T2 period is 4 µs.
(2) The minimum T3 period is two parallel clock cycles.

PCIe Initialization/Compliance Phase

After the device is powered up, a PCIe-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 as shown in Figure 4–10:

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, 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.

PCIe Normal Phase

For the normal operation phase, perform the following reset sequence, as shown in Figure 4–10:

1. After completion of the Initialization/Compliance phase, when the rx_freqlocked signal is de-asserted, (marker 10), wait for the rx_pll_locked signal assertion signifying the lock-to-reference clock.
2. 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) 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) to the time rx_digitalreset is de-asserted (marker 13). The PLD logic ignores the data during this period (between markers 10 and 13).

You can configure the Arria II GX or GZ device in x1, x2, x4, and x8 PIPE lane configurations. The reset sequence described in "PCIe Reset Sequence" on page 4–15 applies to all these multi-lane configurations.

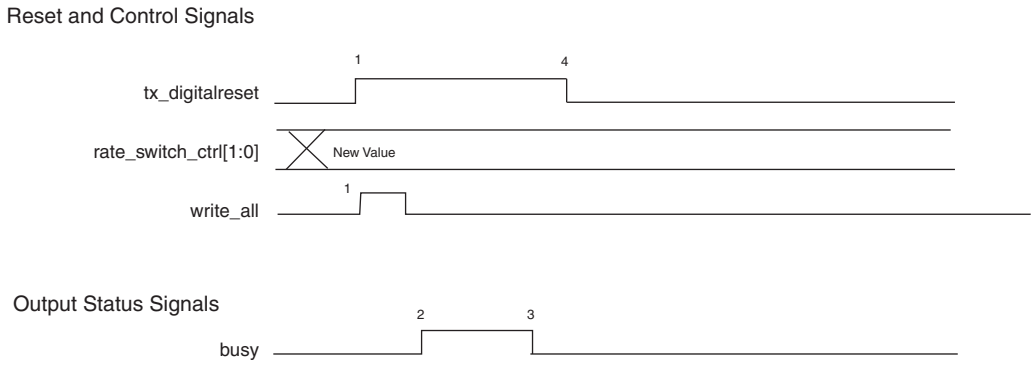
Dynamic Reconfiguration Reset Sequences

When using dynamic reconfiguration in data rate divisions in TX or Channel and TX CMU PLL select/reconfig modes, use the following reset sequences.

Reset Sequence with Data Rate Division in the TX Option

Use the example reset sequence shown in Figure 4–11 when you are using the dynamic reconfiguration controller to change the data rate of the transceiver channel. In this example, dynamic reconfiguration is used to dynamically reconfigure the data rate of the transceiver channel configured in Basic x1 mode with the receiver CDR in automatic lock mode.

Figure 4–11. Reset Sequence in Basic x1 Mode with the Receiver CDR in Automatic Lock Mode (TX Option)



As shown in Figure 4–11, perform the following reset procedure when using the dynamic reconfiguration controller to change the configuration of the transmitter channel:

1. After power up and properly establishing that the transmitter is operating correctly, write the new value for the data rate in the appropriate register (in this example, `rate_switch_ctrl[1:0]`) and subsequently assert the `write_all` signal (marker 1) to initiate the dynamic reconfiguration.

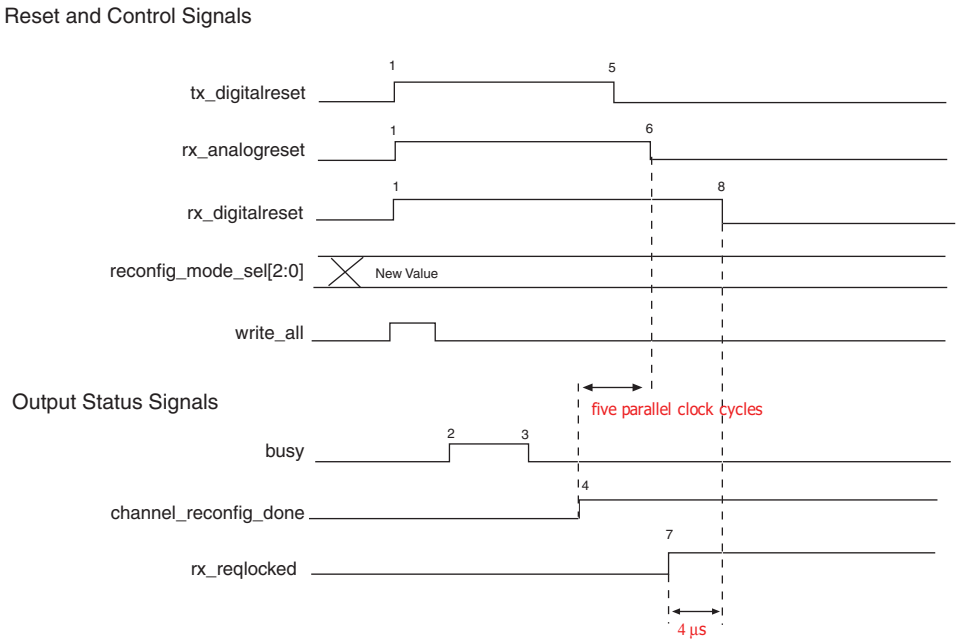
   For more information, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II Devices*.

2. Assert the `tx_digitalreset` signal.
3. As soon as `write_all` is asserted, the dynamic reconfiguration controller starts to execute its operation, as indicated by the assertion of the `busy` signal (marker 2).
4. After the completion of dynamic reconfiguration, the `busy` signal is de-asserted (marker 3).
5. Finally, `tx_digitalreset` can be de-asserted to continue with the transmitter operation (marker 4).

Reset Sequence with the Channel and TX PLL Select/Reconfig Option

Use the example reset sequence shown in Figure 4–12 when you are using the dynamic reconfiguration controller to change the TX PLL settings of the transceiver channel. In this example, the dynamic reconfiguration is used to dynamically reconfigure the data rate of the transceiver channel configured in Basic x1 mode with the receiver CDR in automatic lock mode.

Figure 4–12. Reset Sequence in Basic x1 Mode with Receiver CDR in Automatic Lock Mode (Channel and TX PLL select/reconfig Option)



As shown in Figure 4–12, perform the following reset procedure when using the dynamic reconfiguration controller to change the configuration of the transceiver channel:

1. After power up and establishing that the transceiver is operating correctly, write the new value in the appropriate registers (including `reconfig_mode_sel[2:0]`) and subsequently assert the `write_all` signal (marker 1) to initiate the dynamic reconfiguration.

   For more information, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II Devices*.

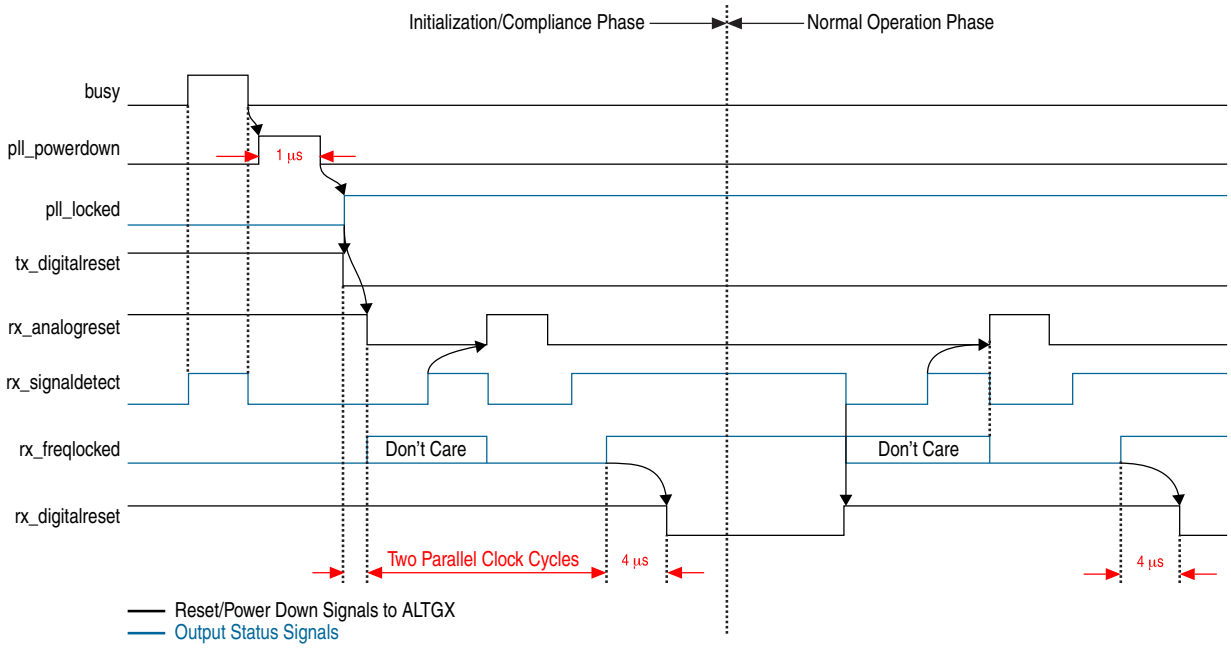
2. Assert the `tx_digitalreset`, `rx_analogreset`, and `rx_digitalreset` signals.
3. As soon as `write_all` is asserted, the dynamic reconfiguration controller starts to execute its operation, as indicated by the assertion of the `busy` signal (marker 2).
4. Wait for the assertion of the `channel_reconfig_done` signal (marker 4), which indicates the completion of dynamic reconfiguration in this mode.

5. After assertion of the `channel_reconfig_done` signal, de-assert `tx_digitalreset` (marker 5) and wait for at least five parallel clock cycles to de-assert the `rx_analogreset` signal (marker 6).
6. Finally, wait for the `rx_freqlocked` signal to go high. After `rx_freqlocked` goes high (marker 7), wait for 4 µs to de-assert the `rx_digitalreset` signal (marker 8). At this point, the receiver is ready for data traffic.

Hot Socketing Reset Sequence

Use the example hot socketing reset sequence shown in Figure 4–13 when you are using hot socketing.

Figure 4–13. Reset Sequence for Hot Socketing



As shown in Figure 4–13, perform the following reset procedure when using hot socketing to change the configuration of the transceiver channel:

1. After power up, assert `pll_powerdown` for a minimum period of 1 µs.
2. When `busy` is asserted, the RX signal detect block is enabled.
3. When `busy` is asserted, `rx_signaldetect` is asserted if the RX input is valid or de-asserted if the RX input is invalid.
4. If `rx_freqlocked` is asserted when `rx_signaldetect` is still low, keep `rx_digitalreset` asserted.
5. Wait until `rx_signaldetect` is asserted, then re-assert `rx_analogreset` for at least two parallel clock cycle.
6. De-assert `rx_digitalreset` when `rx_freqlocked` and `rx_signaldetect` are asserted for ≥ 4 µs.
7. If the RX input is invalid, `rx_signaldetect` is de-asserted.

8. When rx_signaldetect de-asserts, assert rx_digitalreset to avoid RX from receiving corrupted data.
9. Monitor rx_signaldetect until rx_signaldetect is asserted.
10. If the RX input is not floating or the far end TX input is not in electrical idle, rx_signaldetect is asserted when the RX input is valid.
11. When rx_signaldetect is asserted, assert the rx_analogreset for two parallel clock cycles.
12. The rx_freqlocked and rx_signaldetect is de-asserted when rx_analogreset is asserted.



For other functional modes that do not have the rx_signaldetect output port but require hot socketing, you have several options available to determine the link status. You can logically AND together any combinations of these options and use them alternatively as a loss-of-link status indicator.

- Option 1: Create or use the Receiver Monitor Block to observe the rx word alignment status and monitor the received data at the upper layer. Repeat step 4 of the "Hot Socketing Reset Sequence" on page 4–19 if the receiver losses word alignment synchronization or detects an error in the received data.
- Option 2: Implement a Digital Loss of Signal detector by enabling the **Run Length Violation** option in the Word Aligner to detect bit transitions in a predetermined sliding window length.
- Option 3: Implement a parts per million (PPM) detector in the device core to detect loss-of-link status.
- Option 4: Use the Receiver Phase Compensation FIFO overflow/underflow status port to detect loss-of-link status.

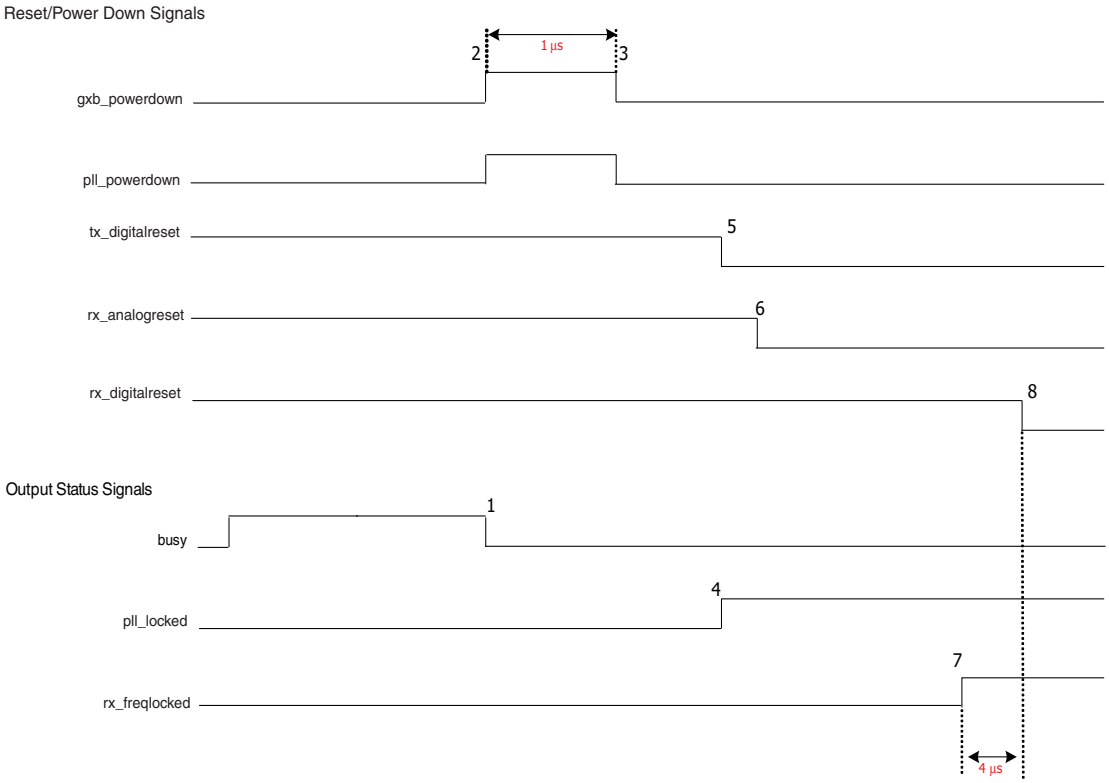
Power Down

The Quartus II software automatically selects the power down channel feature, which takes effect when you configure the Arria II GX or GZ 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 shown in Figure 4–14.

Figure 4–14. Sample Reset Sequence of Four Receiver and Transmitter Channels—Receiver CDR in Automatic Lock Mode with Optional gxb_powerdown Signal



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 approximately 20 parallel reconfig_clk clock cycles in order to reduce the simulation run time. For silicon behavior in the hardware, follow the reset sequences described in this chapter.
- In PCIe 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 4–4 lists the revision history for this chapter.

Table 4–4. Document Revision History

Date	Version	Changes
July 2013	3.2	Updated Figure 4–13.
June 2011	3.1	■ Updated Table 4–1 and Table 4–2. ■ Added "Hot Socketing Reset Sequence" section. ■ Minor text edits.
December 2010	3.0	■ Updated to add Arria II GZ information. ■ Minor text edits.
July 2010	2.0	■ Updated Figure 4–4, Figure 4–5, and Figure 4–12. ■ updated the "Blocks Affected by Reset and Power-Down Signals" section. ■ Minor text edits.
March 2009	1.1	Added the "Dynamic Reconfiguration Reset Sequences" section.
February 2009	1.0	Initial release.

This chapter provides additional information about the *Arria II Device Handbook* and Altera.

About this Handbook

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

How to Contact Altera

To locate the most up-to-date information about Altera products, refer to the following table.

Contact *(1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note to Table:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, **\qdesigns** directory, **D:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicate document titles. For example, *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	Indicate keyboard keys and menu names. For example, the 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`. The suffix `n` denotes an active-low signal. 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`).
↵	An angled arrow instructs you to press the Enter key.
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.
	A question mark directs you to a software help system with related information.
	The feet direct you to another document or website with related information.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The envelope links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.

Arria II Device Handbook

Volume 3: Device Datasheet and Addendum

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

AIIGX5V3-4.4

Document publication date: December 2013

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

QUALITY
ISO 9001:2008
NSAI Certified

Contents

Chapter Revision Dates v

Section I. Device Datasheet and Addendum for Arria II Devices

Revision History 1–1

Chapter 1. Device Datasheet for Arria II Devices

Electrical Characteristics 1–1
Operating Conditions 1–1
Absolute Maximum Ratings 1–1
Maximum Allowed I/O Operating Frequency 1–4
Recommended Operating Conditions 1–5
DC Characteristics 1–7
Schmitt Trigger Input 1–14
I/O Standard Specifications 1–15
Power Consumption for the Arria II Device Family 1–20
Switching Characteristics 1–21
Transceiver Performance Specifications 1–21
Core Performance Specifications for the Arria II Device Family 1–53
Clock Tree Specifications 1–53
PLL Specifications 1–53
DSP Block Specifications 1–57
Embedded Memory Block Specifications 1–58
Configuration 1–60
JTAG Specifications 1–60
Chip-Wide Reset (Dev_CLRn) Specifications 1–60
Periphery Performance 1–61
High-Speed I/O Specification 1–61
External Memory Interface Specifications 1–68
Duty Cycle Distortion (DCD) Specifications 1–71
IOE Programmable Delay 1–72
I/O Timing 1–73
Glossary 1–74
Document Revision History 1–77

Chapter 2. Addendum for the Arria II Device Handbook

Highlights 2–1
High-Speed LVDS I/O with DPA and Soft CDR 2–1
Auto-Calibrating External Memory Interfaces 2–1
Connecting a Serial Configuration Device to an Arria II Device Family on AS Interface 2–1
Document Revision History 2–1

Additional Information

How to Contact Altera Info–1
Typographic Conventions Info–1

Chapter Revision Dates

The chapters in this document, Arria II Device Handbook Volume 3: Device Datasheet and Addendum, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1. Device Datasheet for Arria II Devices
Revised: *December 2013*
Part Number: *AIIGX53001-4.4*

Chapter 2. Addendum for the Arria II Device Handbook
Revised: *December 2010*
Part Number: *AIIGX53002-2.0*

Section I. Device Datasheet and Addendum for Arria II Devices

This section provides information about the Arria® II device family datasheet and addendum. This section includes the following chapters:

- Chapter 1, Device Datasheet for Arria II Devices
- Chapter 2, Addendum for the Arria II Device Handbook

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. Device Datasheet for Arria II Devices

AIIGX53001-4.4

This chapter describes the electrical and switching characteristics of the Arria® II device family. The Arria II device family includes the Arria II GX and GZ devices. Electrical characteristics include operating conditions and power consumption. Switching characteristics include transceiver specifications, core, and periphery performance. This chapter also describes I/O timing, including programmable I/O element (IOE) delay and programmable output buffer delay.

For information regarding the densities and packages of devices in the Arria II device family, refer to *Overview for the Arria II Device Family* chapter.

This chapter contains the following sections:

- "Electrical Characteristics" on page 1–1
- "Transceiver Performance Specifications" on page 1–21
- "Glossary" on page 1–74

Electrical Characteristics

The following sections describe the electrical characteristics.

Operating Conditions

Arria II devices are rated according to a set of defined parameters. To maintain the highest possible performance and reliability of Arria II devices, you must consider the operating requirements described in this chapter.

Arria II devices are offered in both commercial and industrial grades. Arria II GX devices are offered in –4 (fastest), –5, and –6 (slowest) commercial speed grades and –3 and –5 industrial speed grades. Arria II GZ devices are offered in –3 and –4 speed grades for both commercial and industrial grades.

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 indicated as C4, C5, and C6 speed grade, and the industrial devices are indicated as I3 and I5.

Absolute Maximum Ratings

Absolute maximum ratings define the maximum operating conditions for Arria II 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. Table 1–1 lists the absolute maximum ratings for Arria II GX devices. Table 1–2 lists the absolute maximum ratings for Arria II GZ devices.

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

CAUTION

Conditions beyond those listed in Table 1–1 and Table 1–2 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 lists the absolute maximum ratings for Arria II GX devices.

Table 1–1. Absolute Maximum Ratings for Arria II GX Devices

Symbol	Description	Minimum	Maximum	Unit
V_{CC}	Supplies power to the core, periphery, I/O registers, PCI Express® (PIPE) (PCIe) HIP block, and transceiver PCS	–0.5	1.35	V
V_{CCCB}	Supplies power for the configuration RAM bits	–0.5	1.8	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
I_{OUT}	DC output current, per pin	–25	40	mA
V_{CCA}	Supplies power to the transceiver PMA regulator	—	3.75	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	—	1.8	V
T_J	Operating junction temperature	–55	125	°C
T_{STG}	Storage temperature (no bias)	–65	150	°C

Table 1–2 lists the absolute maximum ratings for Arria II GZ devices.

Table 1–2. Absolute Maximum Ratings for Arria II GZ Devices (Part 1 of 2)

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}	Power supply to the configuration RAM bits	-0.5	1.8	V
V_{CCPGM}	Supplies power to the configuration pins	-0.5	3.75	V
V_{CCAUX}	Auxiliary supply	-0.5	3.75	V
V_{CCBAT}	Supplies battery back-up power 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_{CC_CLKIN}	Supplies power to the differential clock input	-0.5	3.75	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
I_{OUT}	DC output current, per pin	-25	40	mA

Table 1–2. Absolute Maximum Ratings for Arria II GZ Devices (Part 2 of 2)

Symbol	Description	Minimum	Maximum	Unit
V_{CCA_L}	Supplies transceiver high voltage power (left side)	-0.5	3.75	V
V_{CCA_R}	Supplies transceiver high voltage power (right side)	-0.5	3.75	V
V_{CCHIP_L}	Supplies transceiver HIP digital power (left side)	-0.5	1.35	V
V_{CCR_L}	Supplies receiver power (left side)	-0.5	1.35	V
V_{CCR_R}	Supplies receiver power (right side)	-0.5	1.35	V
V_{CCT_L}	Supplies transmitter power (left side)	-0.5	1.35	V
V_{CCT_R}	Supplies transmitter power (right side)	-0.5	1.35	V
V_{CCL_GXBLn} *(1)*	Supplies power to the transceiver PMA TX, PMA RX, and clocking (left side)	-0.5	1.35	V
V_{CCL_GXBRn} *(1)*	Supplies power to the transceiver PMA TX, PMA RX, and clocking (right side)	-0.5	1.35	V
V_{CCH_GXBLn} *(1)*	Supplies power to the transceiver PMA output (TX) buffer (left side)	-0.5	1.8	V
V_{CCH_GXBRn} *(1)*	Supplies power to the transceiver PMA output (TX) buffer (right side)	-0.5	1.8	V
T_J	Operating junction temperature	-55	125	°C
T_{STG}	Storage temperature (no bias)	-65	150	°C

Note to Table 1–2:

(1) n = 0, 1, or 2.

Maximum Allowed Overshoot and Undershoot Voltage

During transitions, input signals may overshoot to the voltage shown in Table 1–3 and undershoot to –2.0 V for magnitude of currents less than 100 mA and periods shorter than 20 ns.

Table 1–3 lists the Arria II GX and GZ 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 1–3. Maximum Allowed Overshoot During Transitions for Arria II Devices

Symbol	Description	Condition (V)	Overshoot Duration as % of High Time	Unit
V_I (AC)	AC Input Voltage	4.0	100.000	%
		4.05	79.330	%
		4.1	46.270	%
		4.15	27.030	%
		4.2	15.800	%
		4.25	9.240	%
		4.3	5.410	%
		4.35	3.160	%
		4.4	1.850	%
		4.45	1.080	%
		4.5	0.630	%
		4.55	0.370	%
		4.6	0.220	%

Maximum Allowed I/O Operating Frequency

Table 1–4 lists the maximum allowed I/O operating frequency for Arria II GX I/Os using the specified I/O standards to ensure device reliability.

Table 1–4. Maximum Allowed I/O Operating Frequency for Arria II GX Devices

I/O Standard	I/O Frequency (MHz)
HSTL-18 and HSTL-15	333
SSTL -15	400
SSTL-18	333
2.5-V LVCMOS	260
3.3-V and 3.0-V LVTTL	250
3.3-V, 3.0-V, 1.8-V, and 1.5-V LVCMOS	
PCI and PCI-X	
SSTL-2	
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 and GZ devices. All supplies are required to monotonically reach their full-rail values without plateaus within t_{RAMP}.

Table 1–5 lists the recommended operating conditions for Arria II GX devices.

Table 1–5. Recommended Operating Conditions for Arria II GX Devices *(Note 1)* (Part 1 of 2)

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} *(3)*	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 *(4)*	—	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

Table 1–5. Recommended Operating Conditions for Arria II GX Devices *(Note 1)* (Part 2 of 2)

Symbol	Description	Condition	Minimum	Typical	Maximum	Unit
t_{RAMP}	Power Supply Ramp time	Normal POR	0.05	—	100	ms
		Fast POR	0.05	—	4	ms

Notes to Table 1–5:

(1) For more information about supply pin connections, refer to the *Arria II Device Family Pin Connection Guidelines*.

(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.

(3) V_{CCPD} must be 2.5-V for I/O banks with 2.5-V and lower V_{CCIO}, 3.0-V for 3.0-V V_{CCIO}, and 3.3-V for 3.3-V V_{CCIO}.

(4) 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.

Table 1–6 lists the recommended operating conditions for Arria II GZ devices.

Table 1–6. Recommended Operating Conditions for Arria II GZ Devices *(Note 6)* (Part 1 of 2)

Symbol	Description	Condition	Minimum	Typical	Maximum	Unit
V_{CC}	Core voltage and periphery circuitry power supply	—	0.87	0.90	0.93	V
V_{CCCB}	Supplies power for the configuration RAM bits	—	1.45	1.50	1.55	V
V_{CCAUX}	Auxiliary supply	—	2.375	2.5	2.625	V
V_{CCPD} *(2)*	I/O pre-driver (3.0 V) power supply	—	2.85	3.0	3.15	V
	I/O pre-driver (2.5 V) power supply	—	2.375	2.5	2.625	V
V_{CCIO}	I/O buffers (3.0 V) power supply	—	2.85	3.0	3.15	V
	I/O buffers (2.5 V) power supply	—	2.375	2.5	2.625	V
	I/O buffers (1.8 V) power supply	—	1.71	1.8	1.89	V
	I/O buffers (1.5 V) power supply	—	1.425	1.5	1.575	V
	I/O buffers (1.2 V) power supply	—	1.14	1.2	1.26	V
V_{CCPGM}	Configuration pins (3.0 V) power supply	—	2.85	3.0	3.15	V
	Configuration pins (2.5 V) power supply	—	2.375	2.5	2.625	V
	Configuration pins (1.8 V) power supply	—	1.71	1.8	1.89	V
V_{CCA_PLL}	PLL analog voltage regulator power supply	—	2.375	2.5	2.625	V
V_{CCD_PLL}	PLL digital voltage regulator power supply	—	0.87	0.90	0.93	V
V_{CC_CLKIN}	Differential clock input power supply	—	2.375	2.5	2.625	V
V_{CCBAT} *(1)*	Battery back-up power supply (For design security volatile key register)	—	1.2	—	3.3	V
V_I	DC input voltage	—	–0.5	—	3.6	V
V_O	Output voltage	—	0	—	V_{CCIO}	V
V_{CCA_L}	Transceiver high voltage power (left side)	—	2.85/2.375	3.0/2.5 *(4)*	3.15/2.625	V
V_{CCA_R}	Transceiver high voltage power (right side)	—				
V_{CCHIP_L}	Transceiver HIP digital power (left side)	—	0.87	0.9	0.93	V
V_{CCR_L}	Receiver power (left side)	—	1.05	1.1	1.15	V
V_{CCR_R}	Receiver power (right side)	—	1.05	1.1	1.15	V
V_{CCT_L}	Transmitter power (left side)	—	1.05	1.1	1.15	V
V_{CCT_R}	Transmitter power (right side)	—	1.05	1.1	1.15	V

Table 1–6. Recommended Operating Conditions for Arria II GZ Devices *(Note 6)* (Part 2 of 2)

Symbol	Description	Condition	Minimum	Typical	Maximum	Unit
V_{CCL_GXBLn} *(3)*	Transceiver clock power (left side)	—	1.05	1.1	1.15	V
V_{CCL_GXBRn} *(3)*	Transceiver clock power (right side)	—	1.05	1.1	1.15	V
V_{CCH_GXBLn} *(3)*	Transmitter output buffer power (left side)	—	1.33/1.425	1.4/1.5 *(5)*	1.575	V
V_{CCH_GXBRn} *(3)*	Transmitter output buffer power (right side)	—				
T_J	Operating junction temperature	Commercial	0	—	85	°C
		Industrial	–40	—	100	°C
t_{RAMP}	Power supply ramp time	Normal POR (PORSEL=0)	0.05	—	100	ms
		Fast POR (PORSEL=1)	0.05	—	4	ms

Notes to Table 1–6:

(1) 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.

(2) V_{CCPD} must be 2.5 V when V_{CCIO} is 2.5, 1.8, 1.5, or 1.2 V. V_{CCPD} must be 3.0 V when V_{CCIO} is 3.0 V.

(3) n = 0, 1, or 2.

(4) $V_{CCA_L/R}$ must be connected to a 3.0-V supply if the clock multiplier unit (CMU) phase-locked loop (PLL), receiver clock data recovery (CDR), or both, are configured at a base data rate > 4.25 Gbps. For data rates up to 4.25 Gbps, you can connect $V_{CCA_L/R}$ to either 3.0 V or 2.5 V.

(5) $V_{CCH_GXBL/R}$ must be connected to a 1.4-V supply if the transmitter channel data rate is > 6.5 Gbps. For data rates up to 6.5 Gbps, you can connect $V_{CCH_GXBL/R}$ to either 1.4 V or 1.5 V.

(6) Transceiver power supplies do not have power-on-reset (POR) circuitry. After initial power-up, violating the transceiver power supply operating conditions could lead to unpredictable link behavior.

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. Because these currents vary largely with the resources used, use the Microsoft Excel-based Early Power Estimator (EPE) to get supply current estimates for your design.

For more information about power estimation tools, refer to the *PowerPlay Early Power Estimator User Guide* and the *PowerPlay Power Analysis* chapter.

I/O Pin Leakage Current

Table 1–7 lists the Arria II GX I/O pin leakage current specifications.

Table 1–7. I/O Pin Leakage Current for Arria II GX Devices

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

Table 1–8 lists the Arria II GZ I/O pin leakage current specifications.

Table 1–8. I/O Pin Leakage Current for Arria II GZ Devices

Symbol	Description	Conditions	Min	Typ	Max	Unit
I_I	Input pin	V_I = 0 V to $V_{CCIOMAX}$	–20	—	20	µA
I_{OZ}	Tri-stated I/O pin	V_O = 0 V to $V_{CCIOMAX}$	–20	—	20	µA

Bus Hold

Bus hold retains the last valid logic state after the source driving it either enters the high impedance state or is removed. Each I/O pin has an option to enable bus hold in user mode. Bus hold is always disabled in configuration mode.

Table 1–9 lists bus hold specifications for Arria II GX devices.

Table 1–9. Bus Hold Parameters for Arria II GX Devices *(Note 1)*

Parameter	Symbol	Cond.	V_{CCIO} (V)												Unit
			1.2		1.5		1.8		2.5		3.0		3.3		
			Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
Bus-hold low, sustaining current	I_{SUSL}	$V_{IN} > V_{IL}$ (max.)	8	—	12	—	30	—	50	—	70	—	70	—	µA
Bus-hold high, sustaining current	I_{SUSH}	$V_{IN} < V_{IL}$ (min.)	–8	—	–12	—	–30	—	–50	—	–70	—	–70	—	µA
Bus-hold low, overdrive current	I_{ODL}	0 V < V_{IN} < V_{CCIO}	—	125	—	175	—	200	—	300	—	500	—	500	µA
Bus-hold high, overdrive current	I_{ODH}	0 V < V_{IN} < V_{CCIO}	—	–125	—	–175	—	–200	—	–300	—	–500	—	–500	µA
Bus-hold trip point	V_{TRIP}	—	0.3	0.9	0.375	1.125	0.68	1.07	0.7	1.7	0.8	2	0.8	2	V

Note to Table 1–9:

(1) The bus-hold trip points are based on calculated input voltages from the JEDEC standard.

Table 1–10 lists the bus hold specifications for Arria II GZ devices.

Table 1–10. Bus Hold Parameters for Arria II GZ Devices

Parameter	Symbol	Cond.	V_{CCIO} (V)										Unit
			1.2		1.5		1.8		2.5		3.0		
			Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
Bus-hold Low sustaining current	I_{SUSL}	$V_{IN} > V_{IL}$ (max.)	22.5	—	25.0	—	30.0	—	50.0	—	70.0	—	µA
Bus-hold High sustaining current	I_{SUSH}	$V_{IN} < V_{IH}$ (min.)	-22.5	—	-25.0	—	-30.0	—	-50.0	—	-70.0	—	µA
Bus-hold Low overdrive current	I_{ODL}	$0V < V_{IN} < V_{CCIO}$	—	120	—	160	—	200	—	300	—	500	µA
Bus-hold High overdrive current	I_{ODH}	$0V < V_{IN} < V_{CCIO}$	—	-120	—	-160	—	-200	—	-300	—	-500	µA
Bus-hold trip point	V_{TRIP}	—	0.45	0.95	0.50	1.00	0.68	1.07	0.70	1.70	0.80	2.00	V

OCT Specifications

Table 1–11 lists the Arria II GX device and differential OCT with and without calibration accuracy.

Table 1–11. OCT With and Without Calibration Specification for Arria II GX Device I/Os *(Note 1)* (Part 1 of 2)

Symbol	Description	Conditions (V)	Calibration Accuracy		Unit
			Commercial	Industrial	
25-Ω R_S 3.0, 2.5	25-Ω series OCT without calibration	V_{CCIO} = 3.0, 2.5	± 30	± 40	%
50-Ω R_S 3.0, 2.5	50-Ω series OCT without calibration	V_{CCIO} = 3.0, 2.5	± 30	± 40	%
25-Ω R_S 1.8	25-Ω series OCT without calibration	V_{CCIO} = 1.8	± 40	± 50	%
50-Ω R_S 1.8	50-Ω series OCT without calibration	V_{CCIO} = 1.8	± 40	± 50	%
25-Ω R_S 1.5, 1.2	25-Ω series OCT without calibration	V_{CCIO} = 1.5, 1.2	± 50	± 50	%
50-Ω R_S 1.5, 1.2	50-Ω series OCT without calibration	V_{CCIO} = 1.5, 1.2	± 50	± 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	± 10	± 10	%

Table 1–11. OCT With and Without Calibration Specification for Arria II GX Device I/Os *(Note 1)* (Part 2 of 2)

Symbol	Description	Conditions (V)	Calibration Accuracy		Unit
			Commercial	Industrial	
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	± 10	± 10	%
100-Ω R_D 2.5	100-Ω differential OCT without calibration	V_{CCIO} = 2.5	± 30	± 30	%

Note to Table 1–11:

(1) OCT with calibration accuracy is valid at the time of calibration only.

Table 1–12 lists the OCT termination calibration accuracy specifications for Arria II GZ devices.

Table 1–12. OCT with Calibration Accuracy Specifications for Arria II GZ Devices *(Note 1)*

Symbol	Description	Conditions (V)	Calibration Accuracy			Unit
			C2	C3,I3	C4,I4	
25-Ω R_S 3.0, 2.5, 1.8, 1.5, 1.2 *(2)*	25-Ω series OCT with calibration	V_{CCIO} = 3.0, 2.5, 1.8, 1.5, 1.2	± 8	± 8	± 8	%
50-Ω R_S 3.0, 2.5, 1.8, 1.5, 1.2	50-Ω internal series OCT with calibration	V_{CCIO} = 3.0, 2.5, 1.8, 1.5, 1.2	± 8	± 8	± 8	%
50-Ω R_T 2.5, 1.8, 1.5, 1.2	50-Ω internal parallel OCT with calibration	V_{CCIO} = 2.5, 1.8, 1.5, 1.2	± 10	± 10	± 10	%
20-Ω , 40-Ω , and 60-Ω R_S 3.0, 2.5, 1.8, 1.5, 1.2 *(3)*	20-Ω , 40-Ω and 60-Ω R_S expanded range for internal series OCT with calibration	V_{CCIO} = 3.0, 2.5, 1.8, 1.5, 1.2	± 10	± 10	± 10	%
25-Ω $R_{S_left_shift}$ 3.0, 2.5, 1.8, 1.5, 1.2	25-Ω $R_{S_left_shift}$ internal left shift series OCT with calibration	V_{CCIO} = 3.0, 2.5, 1.8, 1.5, 1.2	± 10	± 10	± 10	%

Notes to Table 1–12:

(1) OCT calibration accuracy is valid at the time of calibration only.

(2) 25-Ω R_S is not supported for 1.5 V and 1.2 V in Row I/O.

(3) 20-Ω R_S is not supported for 1.5 V and 1.2 V in Row I/O.

The calibration accuracy for calibrated series and parallel OCTs are applicable at the moment of calibration. When process, voltage, and temperature (PVT) conditions change after calibration, the tolerance may change.

Table 1–13 lists the Arria II GZ OCT without calibration resistance tolerance to PVT changes.

Table 1–13. OCT Without Calibration Resistance Tolerance Specifications for Arria II GZ Devices

Symbol	Description	Conditions (V)	Resistance Tolerance		Unit
			C3,I3	C4,I4	
25-Ω R_S 3.0 and 2.5	25-Ω internal series OCT without calibration	V_{CCIO} = 3.0, 2.5	± 40	± 40	%
25-Ω R_S 1.8 and 1.5	25-Ω internal series OCT without calibration	V_{CCIO} = 1.8, 1.5	± 40	± 40	%
25-Ω R_S 1.2	25-Ω internal series OCT without calibration	V_{CCIO} = 1.2	± 50	± 50	%
50-Ω R_S 3.0 and 2.5	50-Ω internal series OCT without calibration	V_{CCIO} = 3.0, 2.5	± 40	± 40	%
50-Ω R_S 1.8 and 1.5	50-Ω internal series OCT without calibration	V_{CCIO} = 1.8, 1.5	± 40	± 40	%
50-Ω R_S 1.2	50-Ω internal series OCT without calibration	V_{CCIO} = 1.2	± 50	± 50	%
100-Ω R_D 2.5	100-Ω internal differential OCT	V_{CCIO} = 2.5	± 25	± 25	%

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 and Table 1–14 to determine the OCT variation when voltage and temperature vary after power-up calibration for Arria II GX and GZ devices.

Equation 1–1. OCT Variation ***(Note 1)***

$$R_{OCT} = R_{SCAL}\left(1 + \langle\frac{dR}{dT} \times \Delta T\rangle \pm \langle\frac{dR}{dV} \times \Delta V\rangle\right)$$

Notes to Equation 1–1:

(1) R_{OCT} value calculated from Equation 1–1shows the range of OCT resistance with the variation of temperature and V_{CCIO}.

Use the following with Equation 1–1:

- R_{SCAL} is the OCT resistance value at power up.
- ΔT is the variation of temperature with respect to the temperature at power up.
- ΔV is the variation of voltage with respect to the V_{CCIO} at power up.
- dR/dT is the percentage change of R_{SCAL} with temperature.
- dR/dV is the percentage change of R_{SCAL} with voltage.

Table 1–14 lists the OCT variation with temperature and voltage after power-up calibration for Arria II GX devices.

Table 1–14. OCT Variation after Power-up Calibration for Arria II GX Devices

Nominal Voltage V_{CCIO} (V)	dR/dT (%/°C)	dR/dV (%/mV)
3.0	0.262	0.035
2.5	0.234	0.039
1.8	0.219	0.086
1.5	0.199	0.136
1.2	0.161	0.288

Table 1–15 lists the OCT variation with temperature and voltage after power-up calibration for Arria II GZ devices.

Table 1–15. OCT Variation after Power-Up Calibration for Arria II GZ Devices *(Note 1)*

Nominal Voltage, V_{CCIO} (V)	dR/dT (%/°C)	dR/dV (%/mV)
3.0	0.189	0.0297
2.5	0.208	0.0344
1.8	0.266	0.0499
1.5	0.273	0.0744
1.2	0.317	0.1241

Note to Table 1–15:

(1) Valid for V_{CCIO} range of ±5% and temperature range of 0° to 85°C.

Pin Capacitance

Table 1–16 lists the pin capacitance for Arria II GX devices.

Table 1–16. Pin Capacitance for Arria II GX Devices

Symbol	Description	Typical	Unit
C_{IO}	Input capacitance on I/O pins, dual-purpose pins (differential I/O, clock, R_{up}, R_{dn}), and dedicated clock input pins	7	pF

Table 1–17 lists the pin capacitance for Arria II GZ devices.

Table 1–17. Pin Capacitance for Arria II GZ Devices

Symbol	Description	Typical	Unit
C_{IOTB}	Input capacitance on the top and bottom I/O pins	4	pF
C_{IOLR}	Input capacitance on the left and right I/O pins	4	pF
C_{CLKTB}	Input capacitance on the top and bottom non-dedicated clock input pins	4	pF
C_{CLKLR}	Input capacitance on the left and right non-dedicated clock input pins	4	pF
C_{OUTFB}	Input capacitance on the dual-purpose clock output and feedback pins	5	pF
C_{CLK1}, C_{CLK3}, C_{CLK8}, and C_{CLK10}	Input capacitance for dedicated clock input pins	2	pF

Internal Weak Pull-Up and Weak Pull-Down Resistors

Table 1–18 lists the weak pull-up and pull-down resistor values for Arria II GX devices.

Table 1–18. Internal Weak Pull-up and Weak Pull-Down Resistors for Arria II GX Devices *(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)*	7	25	41	kΩ
		V_{CCIO} = 3.0 V ±5% *(2)*	7	28	47	kΩ
		V_{CCIO} = 2.5 V ±5% *(2)*	8	35	61	kΩ
		V_{CCIO} = 1.8 V ±5% *(2)*	10	57	108	kΩ
		V_{CCIO} = 1.5 V ±5% *(2)*	13	82	163	kΩ
		V_{CCIO} = 1.2 V ±5% *(2)*	19	143	351	kΩ
R_{PD}	Value of TCK pin pull-down resistor	V_{CCIO} = 3.3 V ±5%	6	19	29	kΩ
		V_{CCIO} = 3.0 V ±5%	6	22	32	kΩ
		V_{CCIO} = 2.5 V ±5%	6	25	42	kΩ
		V_{CCIO} = 1.8 V ±5%	7	35	70	kΩ
		V_{CCIO} = 1.5 V ±5%	8	50	112	kΩ

Notes to Table 1–18:

(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}.

Table 1–19 lists the weak pull-up resistor values for Arria II GZ devices.

Table 1–19. Internal Weak Pull-Up Resistor for Arria II GZ Devices *(Note 1)*, *(2)*

Symbol	Description	Conditions	Min	Typ	Max	Unit
R_{PU}	Value of the 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} = 3.0 V ±5% *(3)*	—	25	—	kΩ
		V_{CCIO} = 2.5 V ±5% *(3)*	—	25	—	kΩ
		V_{CCIO} = 1.8 V ±5% *(3)*	—	25	—	kΩ
		V_{CCIO} = 1.5 V ±5% *(3)*	—	25	—	kΩ
		V_{CCIO} = 1.2 V ±5% *(3)*	—	25	—	kΩ

Notes to Table 1–19:

(1) All I/O pins have an option to enable weak pull-up except configuration, test, and JTAG pins.

(2) The internal weak pull-down feature is only available for the JTAG TCK pin. The typical value for this internal weak pull-down resistor is approximately 25 kΩ .

(3) Pin pull-up resistance values may be lower if an external source drives the pin higher than V_{CCIO}.

Hot Socketing

Table 1–20 lists the hot-socketing specification for Arria II GX and GZ devices.

Table 1–20. Hot Socketing Specifications for Arria II Devices

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

Note to Table 1–20:

(1) The I/O ramp rate is 10 ns or more. For ramp rates faster than 10 ns, $|I_{IOPIN}|$ = 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 the TDI, TMS, TCK, nSTATUS, nCONFIG, nCE, CONF_DONE, and DCLK pins. A Schmitt trigger feature introduces hysteresis to the input signal for improved noise immunity, especially for signals with slow edge rates.

Table 1–21 lists the hysteresis specifications across the supported V_{CCIO} range for Schmitt trigger inputs in Arria II GX devices.

Table 1–21. Schmitt Trigger Input Hysteresis Specifications for Arria II GX Devices

Symbol	Description	Condition (V)	Minimum	Unit
$V_{Schmitt}$	Hysteresis for Schmitt trigger input	V_{CCIO} = 3.3	220	mV
		V_{CCIO} = 2.5	180	mV
		V_{CCIO} = 1.8	110	mV
		V_{CCIO} = 1.5	70	mV

I/O Standard Specifications

Table 1–22 through Table 1–35 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 the Arria II device family. They also show the Arria II 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 1–22 through Table 1–35, refer to "Glossary" on page 1–74.

Table 1–22 lists the single-ended I/O standards for Arria II GX devices.

Table 1–22. Single-Ended I/O Standards for Arria II GX Devices

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	3.6	0.45	2.4	4	–4
3.3 V LVCMOS	3.135	3.3	3.465	–0.3	0.8	1.7	3.6	0.2	V_{CCIO} -0.2	2	–2
3.0 V LVTTL	2.85	3	3.15	–0.3	0.8	1.7	V_{CCIO} + 0.3	0.45	2.4	4	–4
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}	V_{CCIO} + 0.3	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}	V_{CCIO} + 0.3	0.1 × V_{CCIO}	0.9 × V_{CCIO}	1.5	–0.5

Table 1–23 lists the single-ended I/O standards for Arria II GZ devices.

Table 1–23. Single-Ended I/O Standards for Arria II GZ Devices (Part 1 of 2)

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		
LVTTL	2.85	3	3.15	-0.3	0.8	1.7	3.6	0.4	2.4	2	-2
LVCMOS	2.85	3	3.15	-0.3	0.8	1.7	3.6	0.2	V_{CCIO} - 0.2	0.1	-0.1
2.5 V	2.375	2.5	2.625	-0.3	0.7	1.7	3.6	0.4	2	1	-1
1.8 V	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	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

Table 1–23. Single-Ended I/O Standards for Arria II GZ Devices (Part 2 of 2)

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		
1.2 V	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 1–24 lists the single-ended SSTL and HSTL I/O reference voltage specifications for Arria II GX devices.

Table 1–24. Single-Ended SSTL and HSTL I/O Reference Voltage Specifications for Arria II GX Devices

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	0.833	0.9	0.969	V_{REF} - 0.04	V_{REF}	V_{REF} + 0.04
SSTL-15 Class I, II	1.425	1.5	1.575	0.47 × V_{CCIO}	0.5 × V_{CCIO}	0.53 × V_{CCIO}	0.47 × V_{CCIO}	0.5 × V_{CCIO}	0.53 × V_{CCIO}
HSTL-18 Class I, II	1.71	1.8	1.89	0.85	0.9	0.95	0.85	0.9	0.95
HSTL-15 Class I, II	1.425	1.5	1.575	0.71	0.75	0.79	0.71	0.75	0.79
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 1–25 lists the single-ended SSTL and HSTL I/O reference voltage specifications for Arria II GZ devices.

Table 1–25. Single-Ended SSTL and HSTL I/O Reference Voltage Specifications for Arria II GZ Devices

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	0.833	0.9	0.969	V_{REF} - 0.04	V_{REF}	V_{REF} + 0.04
SSTL-15 Class I, II	1.425	1.5	1.575	0.47 × V_{CCIO}	0.5 × V_{CCIO}	0.53 × V_{CCIO}	0.47 × V_{CCIO}	V_{REF}	0.53 × V_{CCIO}
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	0.47 × V_{CCIO}	0.5 × V_{CCIO}	0.53 × V_{CCIO}	—	V_{CCIO}/2	—

Table 1–26 lists the single-ended SSTL and HSTL I/O standard signal specifications for Arria II GX devices.

Table 1–26. Single-Ended SSTL and HSTL I/O Standard Signal Specifications for Arria II GX Devices

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} - 0.18	V_{REF} + 0.18	V_{CCIO} + 0.3	V_{REF} - 0.35	V_{REF} + 0.35	V_{TT} - 0.57	V_{TT} + 0.57	8.1	–8.1
SSTL-2 Class II	–0.3	V_{REF} - 0.18	V_{REF} + 0.18	V_{CCIO} + 0.3	V_{REF} - 0.35	V_{REF} + 0.35	V_{TT} - 0.76	V_{TT} + 0.76	16.4	–16.4
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}	14	–14

Table 1–27 lists the single-ended SSTL and HSTL I/O standard signal specifications for Arria II GZ devices.

Table 1–27. Single-Ended SSTL and HSTL I/O Standards Signal Specifications for Arria II GZ Devices (Part 1 of 2)

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	—	V_{REF} - 0.1	V_{REF} + 0.1	—	V_{REF} - 0.175	V_{REF} + 0.175	0.2 × V_{CCIO}	0.8 × V_{CCIO}	8	-8

Table 1–27. Single-Ended SSTL and HSTL I/O Standards Signal Specifications for Arria II GZ Devices (Part 2 of 2)

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-15 Class II	—	V_{REF} - 0.1	V_{REF} + 0.1	—	V_{REF} - 0.175	V_{REF} + 0.175	0.2 × V_{CCIO}	0.8 × V_{CCIO}	16	-16
HSTL-18 Class I	—	V_{REF} - 0.1	V_{REF} + 0.1	—	V_{REF} - 0.2	V_{REF} + 0.2	0.4	V_{CCIO} - 0.4	8	-8
HSTL-18 Class II	—	V_{REF} - 0.1	V_{REF} + 0.1	—	V_{REF} - 0.2	V_{REF} + 0.2	0.4	V_{CCIO} - 0.4	16	-16
HSTL-15 Class I	—	V_{REF} - 0.1	V_{REF} + 0.1	—	V_{REF} - 0.2	V_{REF} + 0.2	0.4	V_{CCIO} - 0.4	8	-8
HSTL-15 Class II	—	V_{REF} - 0.1	V_{REF} + 0.1	—	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 1–28 lists the differential SSTL I/O standards for Arria II GX devices.

Table 1–28. Differential SSTL I/O Standards for Arria II GX Devices

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}	$V_{CCIO}/2$ - 0.15	—	$V_{CCIO}/2$ + 0.15
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}	$V_{CCIO}/2$ - 0.125	—	$V_{CCIO}/2$ + 0.125
SSTL-15 Class I, II	1.425	1.5	1.575	0.2	—	—	$V_{CCIO}/2$	—	0.35	—	—	$V_{CCIO}/2$	—

Table 1–29 lists the differential SSTL I/O standards for Arria II GZ devices

Table 1–29. Differential SSTL I/O Standards for Arria II GZ Devices

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	0.3	V_{CCIO} + 0.6	$V_{CCIO}/2$ - 0.2	—	$V_{CCIO}/2$ + 0.2	0.62	V_{CCIO} + 0.6	$V_{CCIO}/2$ - 0.15	—	$V_{CCIO}/2$ + 0.15
SSTL-18 Class I, II	1.71	1.8	1.89	0.25	V_{CCIO} + 0.6	$V_{CCIO}/2$ - 0.175	—	$V_{CCIO}/2$ + 0.175	0.5	V_{CCIO} + 0.6	$V_{CCIO}/2$ - 0.125	—	$V_{CCIO}/2$ + 0.125
SSTL-15 Class I, II	1.425	1.5	1.575	0.2	—	—	$V_{CCIO}/2$	—	0.35	—	—	$V_{CCIO}/2$	—

Table 1–30 lists the HSTL I/O standards for Arria II GX devices.

Table 1–30. Differential HSTL I/O Standards for Arria II GX Devices

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 1–31 lists the HSTL I/O standards for Arria II GZ devices.

Table 1–31. Differential HSTL I/O Standards for Arria II GZ Devices

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	—	0.78	—	1.12	0.78	—	1.12	0.4	—
HSTL-15 Class I, II	1.425	1.5	1.575	0.2	—	0.68	—	0.9	0.68	—	0.9	0.4	—
HSTL-12 Class I, II	1.14	1.2	1.26	0.16	V_{CCIO} + 0.3	—	0.5 × V_{CCIO}	—	0.4 × V_{CCIO}	0.5 × V_{CCIO}	0.6 × V_{CCIO}	0.3	V_{CCIO} + 0.48

Table 1–32 lists the differential I/O standard specifications for Arria II GX devices.

Table 1–32. Differential I/O Standard Specifications for Arria II GX Devices *(Note 1)*

I/O Standard	V_{CCIO} (V)			V_{ID} (mV)			V_{ICM} (V) *(2)*		V_{OD} (V) *(3)*			V_{OCM} (V)		
	Min	Typ	Max	Min	Cond.	Max	Min	Max	Min	Typ	Max	Min	Typ	Max
2.5 V LVDS	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.05	1.80	0.247	—	0.6	1.125	1.25	1.375
RSDS *(4)*	2.375	2.5	2.625	—	—	—	—	—	0.1	0.2	0.6	0.5	1.2	1.4
Mini-LVDS *(4)*	2.375	2.5	2.625	—	—	—	—	—	0.25	—	0.6	1	1.2	1.4
LVPECL *(5)*	2.375	2.5	2.625	300	—	—	0.6	1.8	—	—	—	—	—	—
BLVDS *(6)*	2.375	2.5	2.625	100	—	—	—	—	—	—	—	—	—	—

Notes to Table 1–32:

(1) The 1.5 V PCML transceiver I/O standard specifications are described in "Transceiver Performance Specifications" on page 1–21.
(2) V_{IN} range: 0 <= V_{IN} <= 1.85 V.
(3) R_L range: 90 <= RL <= 110 Ω.
(4) The RSDS and mini-LVDS I/O standards are only supported for differential outputs.
(5) The LVPECL input standard is supported at the dedicated clock input pins (GCLK) only.
(6) There are no fixed V_{ICM}, V_{OD}, and V_{OCM} specifications for BLVDS. These specifications depend on the system topology.

Table 1–33 lists the differential I/O standard specifications for Arria II GZ devices.

Table 1–33. Differential I/O Standard Specifications for Arria II GZ Devices *(Note 1)*

I/O Standard *(2)*	V_{CCIO} (V)			V_{ID} (mV)			$V_{ICM(DC)}$ (V)		V_{OD} (V) *(3)*			V_{OCM} (V) *(3)*		
	Min	Typ	Max	Min	Cond.	Max	Min	Max	Min	Typ	Max	Min	Typ	Max
2.5 V LVDS (HIO)	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.05	1.8	0.247	—	0.6	1.125	1.25	1.375
2.5 V LVDS (VIO)	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.05	1.8	0.247	—	0.6	1	1.25	1.5
RSDS (HIO)	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.3	1.4	0.1	0.2	0.6	0.5	1.2	1.4
RSDS (VIO)	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.3	1.4	0.1	0.2	0.6	0.5	1.2	1.5
Mini-LVDS (HIO)	2.375	2.5	2.625	200	—	600	0.4	1.325	0.25	—	0.6	1	1.2	1.4
Mini-LVDS (VIO)	2.375	2.5	2.625	200	—	600	0.4	1.325	0.25	—	0.6	1	1.2	1.5
LVPECL	2.375	2.5	2.625	300	—	—	0.6	1.8	—	—	—	—	—	—
BLVDS *(4)*	2.375	2.5	2.625	100	—	—	—	—	—	—	—	—	—	—

Notes to Table 1–33:

(1) 1.4-V/1.5-V PCML transceiver I/O standard specifications are described in "Transceiver Performance Specifications" on page 1–21.
(2) Vertical I/O (VIO) is top and bottom I/Os; horizontal I/O (HIO) is left and right I/Os.
(3) R_L range: $90 \leq RL \leq 110\ \Omega$.
(4) There are no fixed V_{ICM}, V_{OD}, and V_{OCM} specifications for BLVDS. These specifications depend on the system topology.

Power Consumption for the Arria II Device Family

Altera offers two ways to estimate power for a design:

- Using the Microsoft Excel-based Early Power Estimator
- Using the Quartus® II PowerPlay Power Analyzer feature

The interactive Microsoft 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 *PowerPlay Early Power Estimator User Guide* and the *PowerPlay Power Analysis* chapter in volume 3 of the *Quartus II Handbook*.

Switching Characteristics

This section provides performance characteristics of the Arria II GX and GZ core and periphery blocks for commercial grade devices. The following tables are considered final and 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.

Transceiver Performance Specifications

Table 1–34 lists the Arria II GX transceiver specifications.

Table 1–34. Transceiver Specifications for Arria II GX Devices *(Note 1)* (Part 1 of 7)

Symbol/ Description	Condition	I3			C4			C5 and I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Reference Clock														
Supported I/O Standards	1.2-V PCML, 1.5-V PCML, 2.5-V PCML, Differential LVPECL, LVDS, and HCSL													
Input frequency from REFCLK input pins	—	50	—	622.08	50	—	622.08	50	—	622.08	50	—	622.08	MHz
Input frequency from PLD input	—	50	—	200	50	—	200	50	—	200	50	—	200	MHz
Absolute V_{MAX} for a REFCLK pin	—	—	—	2.2	—	—	2.2	—	—	2.2	—	—	2.2	V
Absolute V_{MIN} for a REFCLK pin	—	–0.3	—	—	–0.3	—	—	–0.3	—	—	–0.3	—	—	V
Rise/fall time *(2)*	—	—	—	0.2	—	—	0.2	—	—	0.2	—	—	0.2	UI
Duty cycle	—	45	—	55	45	—	55	45	—	55	45	—	55	%
Peak-to-peak differential input voltage	—	200	—	2000	200	—	2000	200	—	2000	200	—	2000	mV
Spread-spectrum modulating clock frequency	PCIe	30	—	33	30	—	33	30	—	33	30	—	33	kHz

Table 1–34. Transceiver Specifications for Arria II GX Devices *(Note 1)* (Part 2 of 7)

Symbol/ Description	Condition	I3			C4			C5 and I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Spread-spectrum downspread	PCIe	—	0 to –0.5%	—	—	0 to –0.5%	—	—	0 to –0.5%	—	—	0 to –0.5%	—	—
On-chip termination resistors	—	—	100	—	—	100	—	—	100	—	—	100	—	Ω
V_{ICM} (AC coupled)	—	1100 ± 5%			1100 ± 5%			1100 ± 5%			1100 ± 5%			mV
V_{ICM} (DC coupled)	HCSL I/O standard for PCIe reference clock	250	—	550	250	—	550	250	—	550	250	—	550	mV
Transmitter REFCLK Phase Noise	10 Hz	—	—	-50	—	—	-50	—	—	-50	—	—	-50	dBc/Hz
	100 Hz	—	—	-80	—	—	-80	—	—	-80	—	—	-80	dBc/Hz
	1 KHz	—	—	-110	—	—	-110	—	—	-110	—	—	-110	dBc/Hz
	10 KHz	—	—	-120	—	—	-120	—	—	-120	—	—	-120	dBc/Hz
	100 KHz	—	—	-120	—	—	-120	—	—	-120	—	—	-120	dBc/Hz
	≥ 1 MHz	—	—	-130	—	—	-130	—	—	-130	—	—	-130	dBc/Hz
Transmitter REFCLK Phase Jitter (rms) for 100 MHz REFCLK *(3)*	10 KHz to 20 MHz	—	—	3	—	—	3	—	—	3	—	—	3	ps
R_{ref}	—	—	2000 ± 1%	—	—	2000 ± 1%	—	—	2000 ± 1%	—	—	2000 ± 1%	—	Ω
Transceiver Clocks														
Calibration block clock frequency (`cal_blk_clk`)	—	10	—	125	10	—	125	10	—	125	10	—	125	MHz

Table 1–34. Transceiver Specifications for Arria II GX Devices *(Note 1)* (Part 3 of 7)

Symbol/ Description	Condition	I3			C4			C5 and I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
`fixedclk` clock frequency	PCIe Receiver Detect	—	125	—	—	125	—	—	125	—	—	125	—	MHz
`reconfig_clk` clock frequency	Dynamic reconfig. clock frequency	2.5/ 37.5 *(4)*	—	50	2.5/ 37.5 *(4)*	—	50	2.5/ 37.5 *(4)*	—	50	2.5/ 37.5 *(4)*	—	50	MHz
Delta time between `reconfig_clks` *(5)*	—	—	—	2	—	—	2	—	—	2	—	—	2	ms
Transceiver block minimum power-down pulse width	—	—	1	—	—	1	—	—	1	—	—	1	—	µs
Receiver														
Supported I/O Standards	1.4-V PCML, 1.5-V PCML, 2.5-V PCML, 2.5-V PCML, LVPECL, and LVDS													
Data rate *(13)*	—	600	—	6375	600	—	3750	600	—	3750	600	—	3125	Mbps
Absolute V_{MAX} for a receiver pin *(6)*	—	—	—	1.5	—	—	1.5	—	—	1.5	—	—	1.5	V
Absolute V_{MIN} for a receiver pin	—	-0.4	—	—	-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	—	—	2.7	V
	V_{ICM} =1.1 V setting *(7)*	—	—	1.6	—	—	1.6	—	—	1.6	—	—	1.6	V

Table 1–34. Transceiver Specifications for Arria II GX Devices *(Note 1)* (Part 4 of 7)

Symbol/ Description	Condition	I3			C4			C5 and I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Minimum peak-to-peak differential input voltage V_{ID} (diff p-p)	—	100	—	—	100	—	—	100	—	—	100	—	—	mV
V_{ICM}	V_{ICM} = 0.82 V setting	—	820	—	—	820	—	—	820	—	—	820	—	mV
	V_{ICM} =1.1 V setting *(7)*	—	1100	—	—	1100	—	—	1100	—	—	1100	—	mV
Differential on-chip termination resistors	100–Ω setting	—	100	—	—	100	—	—	100	—	—	100	—	Ω
Return loss differential mode	PCIe				50 MHz to 1.25 GHz: –10dB									
	XAUI				100 MHz to 2.5 GHz: –10dB									
Return loss common mode	PCIe				50 MHz to 1.25 GHz: –6dB									
	XAUI				100 MHz to 2.5 GHz: –6dB									
Programmable PPM detector *(8)*	—	± 62.5, 100, 125, 200, 250, 300, 500, 1000												ppm
Run length	—	—	80	—	—	80	—	—	80	—	—	80	—	UI
Programmable equalization	—	—	—	7	—	—	7	—	—	7	—	—	7	dB
Signal detect/loss threshold	PCIe Mode	65	—	175	65	—	175	65	—	175	65	—	175	mV
CDR LTR time *(9)*	—	—	—	75	—	—	75	—	—	75	—	—	75	µs
CDR minimum T1b *(10)*	—	15	—	—	15	—		15	—	—	15	—	—	µs

Table 1–34. Transceiver Specifications for Arria II GX Devices *(Note 1)* (Part 5 of 7)

Symbol/ Description	Condition	I3			C4			C5 and I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
LTD lock time *(11)*	—	0	100	4000	0	100	4000	0	100	4000	0	100	4000	ns
Data lock time from `rx_freqlocked` *(12)*	—	—	—	4000	—	—	4000	—	—	4000	—	—	4000	ns
Programmable DC gain	DC Gain Setting = 0	—	0	—	—	0	—	—	0	—	—	0	—	dB
	DC Gain Setting = 1	—	3	—	—	3	—	—	3	—	—	3	—	dB
	DC Gain Setting = 2	—	6	—	—	6	—	—	6	—	—	6	—	dB
Transmitter														
Supported I/O Standards	1.5-V PCML													
Data rate	—	600	—	6375	600	—	3750	600	—	3750	600	—	3125	Mbps
V_{OCM}	0.65 V setting	—	650	—	—	650	—	—	650	—	—	650	—	mV
Differential on-chip termination resistors	100–Ω setting	—	100	—	—	100	—	—	100	—	—	100	—	Ω
Return loss differential mode	PCIe	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	PCIe	50 MHz to 1.25 GHz: –6dB												
Rise time *(2)*	—	50	—	200	50	—	200	50	—	200	50	—	200	ps
Fall time	—	50	—	200	50	—	200	50	—	200	50	—	200	ps

Table 1–34. Transceiver Specifications for Arria II GX Devices *(Note 1)* (Part 6 of 7)

Symbol/ Description	Condition	I3			C4			C5 and I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Intra-differential pair skew	—	—	—	15	—	—	15	—	—	15	—	—	15	ps
Intra-transceiver block skew	PCIe ×4	—	—	120	—	—	120	—	—	120	—	—	120	ps
Inter-transceiver block skew	PCIe ×8	—	—	300	—	—	300	—	—	300	—	—	300	ps
CMU PLL0 and CMU PLL1														
CMU PLL lock time from `CMUPLL_reset` deassertion	—	—	—	100	—	—	100	—	—	100	—	—	100	µs
PLD-Transceiver Interface														
Interface speed	—	25	—	320	25	—	240	25	—	240	25	—	200	MHz

Table 1–34. Transceiver Specifications for Arria II GX Devices *(Note 1)* (Part 7 of 7)

Symbol/ Description	Condition	I3			C4			C5 and I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Digital reset pulse width	—	Minimum is 2 parallel clock cycles												

Notes to Table 1–34:

(1) For AC-coupled links, the on-chip biasing circuit is switched off before and during configuration. Ensure that input specifications are not violated during this period.

(2) The rise/fall time is specified from 20% to 80%.

(3) To calculate the REFCLK rms phase jitter requirement at reference clock frequencies other than 100 MHz, use the following formula:
REFCLK rms phase jitter at f (MHz) = REFCLK rms phase jitter at 100 MHz * 100/f.

(4) The minimum `reconfig_clk` frequency is 2.5 MHz if the transceiver channel is configured in **Transmitter only** mode. The minimum `reconfig_clk` frequency is 37.5 MHz if the transceiver channel is configured in **Receiver only** or **Receiver and Transmitter** mode. For more information, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II Devices*.

(5) If your design uses more than one dynamic reconfiguration controller instances (`altgx_reconfig`) to control the transceiver channels (`altgx`) physically located on the same side of the device, and if you use different `reconfig_clk` sources for these `altgx_reconfig` instances, the delta time between any two of these `reconfig_clk` sources becoming stable must not exceed the maximum specification listed.

(6) The device cannot tolerate prolonged operation at this absolute maximum.

(7) You must use the 1.1-V RX V_{ICM} setting if the input serial data standard is LVDS and the link is DC-coupled.

(8) The rate matcher supports only up to ±300 parts per million (ppm).

(9) Time taken to `rx_pll_locked` goes high from `rx_analogreset` de-assertion. Refer to Figure 1–1.

(10) The time in 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.

(11) The time taken to recover valid data after the `rx_locktodata` signal is asserted in manual mode. Refer to Figure 1–1.

(12) The time taken to recover valid data after the `rx_freqlocked` signal goes high in automatic mode. Refer to Figure 1–2.

(13) To support data rates lower than the minimum specification through oversampling, use the CDR in LTR mode only.

Table 1–35 lists the transceiver specifications for Arria II GZ devices.

Table 1–35. Transceiver Specifications for Arria II GZ Devices (Part 1 of 5)

Symbol/ Description	Conditions	–C3 and –I3 (1)			–C4 and –I4			Unit
		Min	Typ	Max	Min	Typ	Max	
Reference Clock								
Supported I/O Standards	1.2-V PCML, 1.5-V PCML, 2.5-V PCML, Differential LVPECL, LVDS, and HCSL							
Input frequency from REFCLK input pins	—	50	—	697	50	—	637.5	MHz
Phase frequency detector (CMU PLL and receiver CDR)	—	50	—	325	50	—	325	MHz
Absolute V_{MAX} for a REFCLK pin	—	—	—	1.6	—	—	1.6	V
Operational V_{MAX} for a REFCLK pin	—	—	—	1.5	—	—	1.5	V
Absolute V_{MIN} for a REFCLK pin	—	-0.4	—	—	-0.4	—	—	V
Rise/fall time (2)	—	—	—	0.2	—	—	0.2	UI
Duty cycle	—	45	—	55	45	—	55	%
Peak-to-peak differential input voltage	—	200	—	1600	200	—	1600	mV
Spread-spectrum modulating clock frequency	PCIe	30	—	33	30	—	33	kHz
Spread-spectrum downspread	PCIe	—	0 to -0.5%	—	—	0 to -0.5%	—	—
On-chip termination resistors	—	—	100	—	—	100	—	Ω
V_{ICM} (AC coupled)	—	1100 ± 10%			1100 ± 10%			mV
V_{ICM} (DC coupled)	HCSL I/O standard for PCIe reference clock	250	—	550	250	—	550	mV
Transmitter REFCLK Phase Noise	10 Hz	—	—	-50	—	—	-50	dBc/Hz
	100 Hz	—	—	-80	—	—	-80	dBc/Hz
	1 KHz	—	—	-110	—	—	-110	dBc/Hz
	10 KHz	—	—	-120	—	—	-120	dBc/Hz
	100 KHz	—	—	-120	—	—	-120	dBc/Hz
	≥ 1 MHz	—	—	-130	—	—	-130	dBc/Hz
Transmitter REFCLK Phase Jitter (rms) for 100 MHz REFCLK (3)	10 KHz to 20 MHz	—	—	3	—	—	3	ps
R_{REF}	—	—	2000 ± 1%	—	—	2000 ± 1%	—	Ω

Table 1–35. Transceiver Specifications for Arria II GZ Devices (Part 2 of 5)

Symbol/ Description	Conditions	–C3 and –I3 *(1)*			–C4 and –I4			Unit
		Min	Typ	Max	Min	Typ	Max	
Transceiver Clocks								
Calibration block clock frequency (cal_blk_clk)	—	10	—	125	10	—	125	MHz
fixedclk clock frequency	PCIe Receiver Detect	—	125	—	—	125	—	MHz
reconfig_clk clock frequency	Dynamic reconfiguration clock frequency	2.5/ 37.5 *(4)*	—	50	2.5/ 37.5 *(4)*	—	50	MHz
Delta time between reconfig_clks *(5)*	—	—	—	2	—	—	2	ms
Transceiver block minimum power-down (gxb_powerdown) pulse width	—	1	—	—	1	—	—	µs
Receiver								
Supported I/O Standards	1.4-V PCML, 1.5-V PCML, 2.5-V PCML, LVPECL, and LVDS							
Data rate *(16)*	—	600	—	6375	600	—	3750	Mbps
Absolute V_{MAX} for a receiver pin *(6)*	—	—	—	1.6	—	—	1.6	V
Operational V_{MAX} for a receiver pin	—	—	—	1.5	—	—	1.5	V
Absolute V_{MIN} for a receiver pin	—	-0.4	—	—	-0.4	—	—	V
Maximum peak-to-peak differential input voltage V_{ID} (diff p-p) before device configuration	—	—	—	1.6	—	—	1.6	V
Maximum peak-to-peak differential input voltage V_{ID} (diff p-p) after device configuration	V_{ICM} = 0.82 V setting	—	—	2.7	—	—	2.7	V
	V_{ICM} =1.1 V setting *(7)*	—	—	1.6	—	—	1.6	V
Minimum differential eye opening at receiver serial input pins *(8)*	Data Rate = 600 Mbps to 5 Gbps Equalization = 0 DC gain = 0 dB	100	—	—	165	—	—	mV
	Data Rate > 5 Gbps Equalization = 0 DC gain = 0 dB	165	—	—	165	—	—	mV
V_{ICM}	V_{ICM} = 0.82 V setting	820 ± 10%			820 ± 10%			mV
	V_{ICM} = 1.1 V setting *(7)*	1100 ± 10%			1100 ± 10%			mV

Table 1–35. Transceiver Specifications for Arria II GZ Devices (Part 3 of 5)

Symbol/ Description	Conditions	–C3 and –I3 *(1)*			–C4 and –I4			Unit
		Min	Typ	Max	Min	Typ	Max	
Receiver DC Coupling Support	—	For more information about receiver DC coupling support, refer to the "DC-Coupled Links" section in the *Transceiver Architecture for Arria II Devices* chapter.						
Differential on-chip termination resistors	85–Ω setting	85 ± 20%			85 ± 20%			Ω
	100–Ω setting	100 ± 20%			100 ± 20%			Ω
	120–Ω setting	120 ± 20%			120 ± 20%			Ω
	150-Ω setting	150 ± 20%			150 ± 20%			Ω
Differential and common mode return loss	PCIe (Gen 1 and Gen 2), XAUI, HiGig+, CEI SR/LR, SRIO SR/LR, CPRI LV/HV, OBSAI, SATA	Compliant						—
Programmable PPM detector *(9)*	—	± 62.5, 100, 125, 200, 250, 300, 500, 1,000						ppm
Run length	—	—	—	200	—	—	200	UI
Programmable equalization	—	—	—	16	—	—	16	dB
t_{LTR} *(10)*	—	—	—	75	—	—	75	µs
$t_{LTR_LTD_Manual}$ *(11)*	—	15	—	—	15	—	—	µs
t_{LTD_Manual} *(12)*	—	—	—	4000	—	—	4000	ns
t_{LTD_Auto} *(13)*	—	—	—	4000	—	—	4000	ns
Receiver CDR 3 dB Bandwidth in lock-to-data (LTD) mode	PCIe Gen1	2.0 - 3.5						MHz
	PCIe Gen2	40 - 65						MHz
	(OIF) CEI PHY at 6.375 Gbps	20 - 35						MHz
	XAUI	10 - 18						MHz
	SRIO 1.25 Gbps	10 - 18						MHz
	SRIO 2.5 Gbps	10 - 18						MHz
	SRIO 3.125 Gbps	6 - 10						MHz
	GIGE	6 - 10						MHz
	SONET OC12	3 - 6						MHz
	SONET OC48	14 - 19						MHz
Receiver buffer and CDR offset cancellation time (per channel)	—	—	—	17000	—	—	17000	reconfig_clk cycles
Programmable DC gain	DC Gain Setting = 0	—	0	—	—	0	—	dB
	DC Gain Setting = 1	—	3	—	—	3	—	dB
	DC Gain Setting = 2	—	6	—	—	6	—	dB

Table 1–35. Transceiver Specifications for Arria II GZ Devices (Part 4 of 5)

Symbol/ Description	Conditions	–C3 and –I3 *(1)*			–C4 and –I4			Unit
		Min	Typ	Max	Min	Typ	Max	
Transmitter								
Supported I/O Standards	1.5-V PCML							
Data rate *(14)*	—	600	—	6375	600	—	3750	Mbps
V_{OCM}	0.65 V setting	—	650	—	—	650	—	mV
Differential on-chip termination resistors	85–Ω setting	85 ± 15%			85 ± 15%			Ω
	100–Ω setting	100 ± 15%			100 ± 15%			Ω
	120–Ω setting	120 ± 15%			120 ± 15%			Ω
	150-Ω setting	150 ± 15%			150 ± 15%			Ω
Differential and common mode return loss	PCIe Gen1 and Gen2 (TX V_{OD}=4), XAUI (TX V_{OD}=6), HiGig+ (TX V_{OD}=6), CEI SR/LR (TX V_{OD}=8), SRIO SR (V_{OD}=6), SRIO LR (V_{OD}=8), CPRI LV (V_{OD}=6), CPRI HV (V_{OD}=2), OBSAI (V_{OD}=6), SATA (V_{OD}=4),	Compliant						—
Rise time *(15)*	—	50	—	200	50	—	200	ps
Fall time *(15)*	—	50	—	200	50	—	200	ps
Intra-differential pair skew	—	—	—	15	—	—	15	ps
Intra-transceiver block transmitter channel-to-channel skew	×4 PMA and PCS bonded mode Example: XAUI, PCIe ×4, Basic ×4	—	—	120	—	—	120	ps
Inter-transceiver block transmitter channel-to-channel skew	×8 PMA and PCS bonded mode Example: PCIe ×8, Basic ×8	—	—	500	—	—	500	ps
CMU0 PLL and CMU1 PLL								
Supported Data Range	—	600	—	6375	600	—	3750	Mbps
`pll_powerdown` minimum pulse width (`tpll_powerdown`)	—	1			1			µs
CMU PLL lock time from `pll_powerdown` de-assertion	—	—	—	100	—	—	100	µs

Table 1–35. Transceiver Specifications for Arria II GZ Devices (Part 5 of 5)

Symbol/ Description	Conditions	–C3 and –I3 *(1)*			–C4 and –I4			Unit
		Min	Typ	Max	Min	Typ	Max	
-3 dB Bandwidth	PCIe Gen1	2.5 - 3.5						MHz
	PCIe Gen2	6 - 8						MHz
	(OIF) CEI PHY at 4.976 Gbps	7 - 11						MHz
	(OIF) CEI PHY at 6.375 Gbps	5 - 10						MHz
	XAUI	2 - 4						MHz
	SRIO 1.25 Gbps	3 - 5.5						MHz
	SRIO 2.5 Gbps	3 - 5.5						MHz
	SRIO 3.125 Gbps	2 - 4						MHz
	GIGE	2.5 - 4.5						MHz
	SONET OC12	1.5 - 2.5						MHz
	SONET OC48	3.5 - 6						MHz
Transceiver-FPGA Fabric Interface								
Interface speed	—	25	—	325	25	—	250	MHz
Digital reset pulse width	—	Minimum is two parallel clock cycles						—

Notes to Table 1–35:

(1) The 3x speed grade is the fastest speed grade offered in the following Arria II GZ devices: EP2AGZ225, EP2AGZ300, and EP2AGZ350.

(2) The rise and fall time transition is specified from 20% to 80%.

(3) To calculate the REFCLK rms phase jitter requirement at reference clock frequencies other than 100 MHz, use the following formula: REFCLK rms phase jitter at f (MHz) = REFCLK rms phase jitter at 100 MHz * 100/f.

(4) The minimum `reconfig_clk` frequency is 2.5 MHz if the transceiver channel is configured in **Transmitter only** mode. The minimum `reconfig_clk` frequency is 37.5 MHz if the transceiver channel is configured in **Receiver only** or **Receiver and Transmitter** mode.

(5) If your design uses more than one dynamic reconfiguration controller (`altgx_reconfig`) instances to control the transceiver (`altgx`) channels physically located on the same side of the device AND if you use different `reconfig_clk` sources for these `altgx_reconfig` instances, the delta time between any two of these `reconfig_clk` sources becoming stable must not exceed the maximum specification listed.

(6) The device cannot tolerate prolonged operation at this absolute maximum.

(7) You must use the 1.1-V RX V_{ICM} setting if the input serial data standard is LVDS.

(8) The differential eye opening specification at the receiver input pins assumes that Receiver Equalization is disabled. If you enable Receiver Equalization, the receiver circuitry can tolerate a lower minimum eye opening, depending on the equalization level. Use H-Spice simulation to derive the minimum eye opening requirement with Receiver Equalization enabled.

(9) The rate matcher supports only up to ± 300 ppm.

(10) Time taken to `rx_pll_locked` goes high from `rx_analogreset` de-assertion. Refer to Figure 1–1 on page 1–33.

(11) 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 on page 1–33.

(12) Time taken to recover valid data after the `rx_locktodata` signal is asserted in manual mode. Refer to Figure 1–1 on page 1–33.

(13) Time taken to recover valid data after the `rx_freqlocked` signal goes high in automatic mode. Refer to Figure 1–2 on page 1–33.

(14) A GPLL may be required to meet the PMA-FPGA fabric interface timing above certain data rates. For more information, refer to the *Transceiver Clocking for Arria II Devices* chapter.

(15) The Quartus II software automatically selects the appropriate slew rate depending on the configured data rate or functional mode.

(16) To support data rates lower than the minimum specification through oversampling, use the CDR in LTR mode only.

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

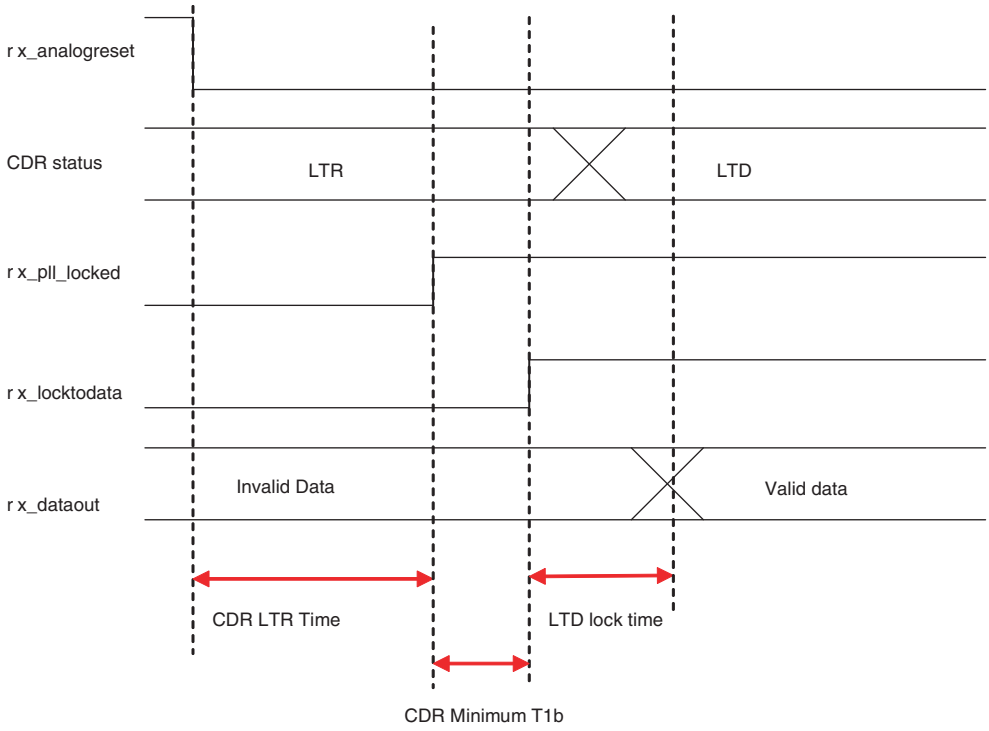


Figure 1–2 shows the lock time parameters in automatic mode.

Figure 1–2. Lock Time Parameters for Automatic Mode

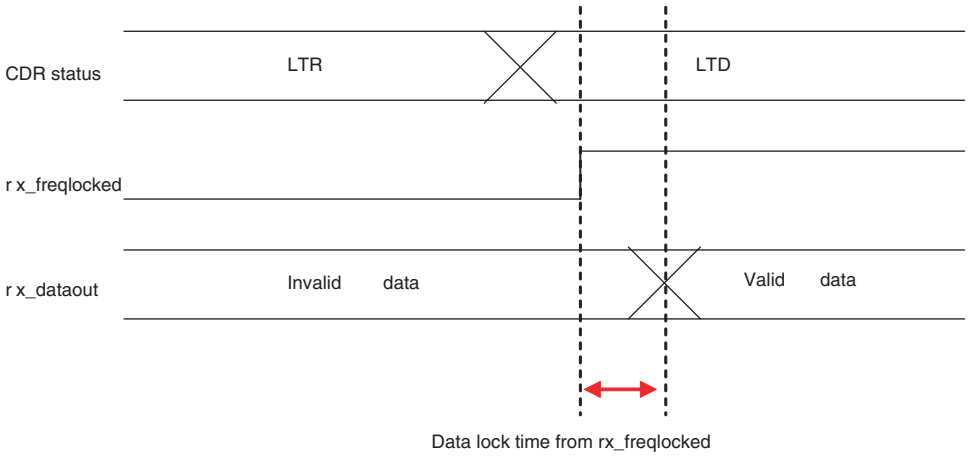


Figure 1–3 shows the differential receiver input waveform.

Figure 1–3. Receiver Input Waveform

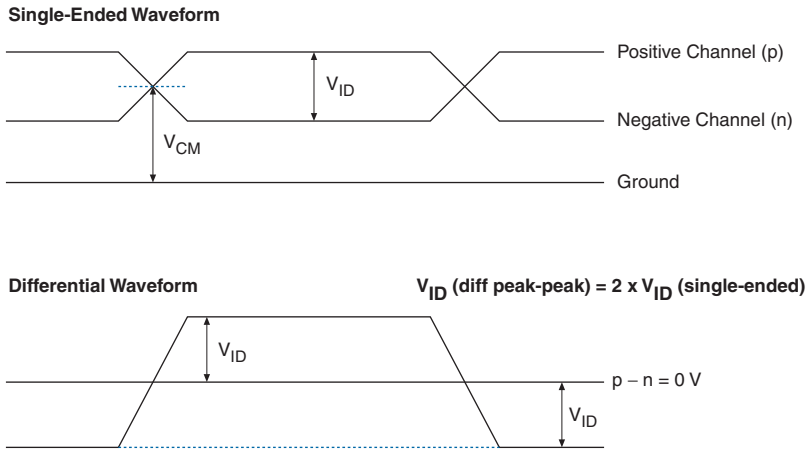


Figure 1–4 shows the transmitter output waveform.

Figure 1–4. Transmitter Output Waveform

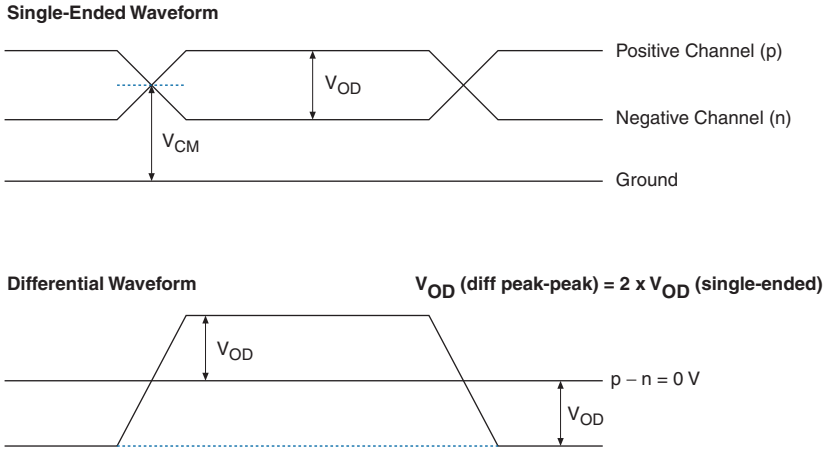


Table 1–36 lists the typical V_{OD} for TX term that equals 85 Ω for Arria II GZ devices.

Table 1–36. Typical V_{OD} Setting, TX Term = 85 Ω for Arria II GZ Devices

Symbol	V_{OD} Setting (mV)							
	0	1	2	3	4	5	6	7
V_{OD} differential peak-to-peak Typical (mV)	170 ± 20%	340 ± 20%	510 ± 20%	595 ± 20%	680 ± 20%	765 ± 20%	850 ± 20%	1020 ± 20%

Table 1–37 lists the typical V_{OD} for TX term that equals 100 Ω for Arria II GX and GZ devices.

Table 1–37. Typical V_{OD} Setting, TX Termination = 100 Ω for Arria II Devices

Quartus II Setting	V_{OD} Setting (mV)
1	400
2	600
3 (Arria II GZ)	700
4	800
5	900
6	1000
7	1200

Table 1–38 lists the typical transmitter pre-emphasis levels in dB for the first post tap under the following conditions: low-frequency data pattern (five 1s and five 0s) at 6.375 Gbps. The levels listed in Table 1–38 are a representation of possible pre-emphasis levels under these specified conditions only, the pre-emphasis levels may change with data pattern and data rate.

To predict the pre-emphasis level for your specific data rate and pattern, run simulations using the Arria II GX HSSI HSPICE models.

Table 1–38. Transmitter Pre-Emphasis Levels for Arria II GX Devices

Arria II GX (Quartus II Software) First Post Tap Setting	Arria II GX (Quartus II Software) VOD Setting						
	1	2	4	5	6	7	Unit
0 (off)	0	0	0	0	0	0	—
1	0.7	0	0	0	0	0	dB
2	2.7	1.2	0.3	0	0	0	dB
3	4.9	2.4	1.2	0.8	0.5	0.2	dB
4	7.5	3.8	2.1	1.6	1.2	0.6	dB
5	—	5.3	3.1	2.4	1.8	1.1	dB
6	—	7	4.3	3.3	2.7	1.7	dB

Table 1–39 lists typical transmitter pre-emphasis levels for Arria II GZ devices (in dB) for the first post tap under the following conditions (low-frequency data pattern [five 1s and five 0s] at 6.25 Gbps). The levels listed in Table 1–39 are a representation of possible pre-emphasis levels under the specified conditions only and that the pre-emphasis levels may change with data pattern and data rate.

To predict the pre-emphasis level for your specific data rate and pattern, run simulations using the Arria II HSSI HSPICE models.

Table 1–39. Transmitter Pre-Emphasis Levels for Arria II GZ Devices (Part 1 of 2)

Pre-Emphasis 1st Post-Tap Setting	V_{OD} Setting							
	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	N/A	0.7	0	0	0	0	0	0
2	N/A	1	0.3	0	0	0	0	0
3	N/A	1.5	0.6	0	0	0	0	0
4	N/A	2	0.7	0.3	0	0	0	0
5	N/A	2.7	1.2	0.5	0.3	0	0	0
6	N/A	3.1	1.3	0.8	0.5	0.2	0	0
7	N/A	3.7	1.8	1.1	0.7	0.4	0.2	0
8	N/A	4.2	2.1	1.3	0.9	0.6	0.3	0
9	N/A	4.9	2.4	1.6	1.2	0.8	0.5	0.2
10	N/A	5.4	2.8	1.9	1.4	1	0.7	0.3
11	N/A	6	3.2	2.2	1.7	1.2	0.9	0.4
12	N/A	6.8	3.5	2.6	1.9	1.4	1.1	0.6
13	N/A	7.5	3.8	2.8	2.1	1.6	1.2	0.6
14	N/A	8.1	4.2	3.1	2.3	1.7	1.3	0.7
15	N/A	8.8	4.5	3.4	2.6	1.9	1.5	0.8
16	N/A	N/A	4.9	3.7	2.9	2.2	1.7	0.9
17	N/A	N/A	5.3	4	3.1	2.4	1.8	1.1
18	N/A	N/A	5.7	4.4	3.4	2.6	2	1.2
19	N/A	N/A	6.1	4.7	3.6	2.8	2.2	1.4
20	N/A	N/A	6.6	5.1	4	3.1	2.4	1.5
21	N/A	N/A	7	5.4	4.3	3.3	2.7	1.7
22	N/A	N/A	8	6.1	4.8	3.8	3	2
23	N/A	N/A	9	6.8	5.4	4.3	3.4	2.3
24	N/A	N/A	10	7.6	6	4.8	3.9	2.6
25	N/A	N/A	11.4	8.4	6.8	5.4	4.4	3
26	N/A	N/A	12.6	9.4	7.4	5.9	4.9	3.3
27	N/A	N/A	N/A	10.3	8.1	6.4	5.3	3.6
28	N/A	N/A	N/A	11.3	8.8	7.1	5.8	4

Table 1–39. Transmitter Pre-Emphasis Levels for Arria II GZ Devices (Part 2 of 2)

Pre-Emphasis 1st Post-Tap Setting	V_{OD} Setting							
	0	1	2	3	4	5	6	7
29	N/A	N/A	N/A	12.5	9.6	7.7	6.3	4.3
30	N/A	N/A	N/A	N/A	11.4	9	7.4	N/A
31	N/A	N/A	N/A	N/A	12.9	10	8.2	N/A

Table 1–40 lists the transceiver jitter specifications for all supported protocols for Arria II GX devices.

Table 1–40. Transceiver Block Jitter Specifications for Arria II GX Devices *(Note 1)* (Part 1 of 10)

Symbol/ Description	Conditions	I3			C4			C5, I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
SONET/SDH Transmit Jitter Generation *(2)*														
Peak-to-peak jitter at 622.08 Mbps	Pattern = PRBS15	—	—	0.1	—	—	0.1	—	—	0.1	—	—	0.1	UI
RMS jitter at 622.08 Mbps	Pattern = PRBS15	—	—	0.01	—	—	0.01	—	—	0.01	—	—	0.01	UI
Peak-to-peak jitter at 2488.32 Mbps	Pattern = PRBS15	—	—	0.1	—	—	0.1	—	—	0.1	—	—	0.1	UI
RMS jitter at 2488.32 Mbps	Pattern = PRBS15	—	—	0.01	—	—	0.01	—	—	0.01	—	—	0.01	UI
SONET/SDH Receiver Jitter Tolerance *(2)*														
Jitter tolerance at 622.08 Mbps	Jitter frequency = 0.03 KHz Pattern = PRBS15	> 15			> 15			> 15			> 15			UI
	Jitter frequency = 25 KHZ Pattern = PRBS15	> 1.5			> 1.5			> 1.5			> 1.5			UI
	Jitter frequency = 250 KHz Pattern = PRBS15	> 0.15			> 0.15			> 0.15			> 0.15			UI

Table 1–40. Transceiver Block Jitter Specifications for Arria II GX Devices *(Note 1)* (Part 2 of 10)

Symbol/ Description	Conditions	I3			C4			C5, I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Jitter tolerance at 2488.32 Mbps	Jitter frequency = 0.06 KHz Pattern = PRBS15		> 15			> 15			> 15			> 15		UI
	Jitter frequency = 100 KHZ Pattern = PRBS15		> 1.5			> 1.5			> 1.5			> 1.5		UI
	Jitter frequency = 1 MHz Pattern = PRBS15		> 0.15			> 0.15			> 0.15			> 0.15		UI
	Jitter frequency = 10 MHz Pattern = PRBS15		> 0.15			> 0.15			> 0.15			> 0.15		UI
XAUI Transmit Jitter Generation *(3)*														
Total jitter at 3.125 Gbps	Pattern = CJPAT	—	—	0.3	—	—	0.3	—	—	0.3	—	—	0.3	UI
Deterministic jitter at 3.125 Gbps	Pattern = CJPAT	—	—	0.17	—	—	0.17	—	—	0.17	—	—	0.17	UI
XAUI Receiver Jitter Tolerance *(3)*														
Total jitter	—		> 0.65			> 0.65			> 0.65			> 0.65		UI
Deterministic jitter	—		> 0.37			> 0.37			> 0.37			> 0.37		UI
Peak-to-peak jitter	Jitter frequency = 22.1 KHz		> 8.5			> 8.5			> 8.5			> 8.5		UI
Peak-to-peak jitter	Jitter frequency = 1.875 MHz		> 0.1			> 0.1			> 0.1			> 0.1		UI
Peak-to-peak jitter	Jitter frequency = 20 MHz		> 0.1			> 0.1			> 0.1			> 0.1		UI
PCIe Transmit Jitter Generation *(4)*														
Total jitter at 2.5 Gbps (Gen1)	Compliance pattern	—	—	0.25	—	—	0.25	—	—	0.25	—	—	0.25	UI

Table 1–40. Transceiver Block Jitter Specifications for Arria II GX Devices *(Note 1)* (Part 3 of 10)

Symbol/ Description	Conditions	I3			C4			C5, I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
PCIe Receiver Jitter Tolerance *(4)*														
Total jitter at 2.5 Gbps (Gen1)	Compliance pattern	> 0.6			> 0.6			> 0.6			> 0.6			UI
PCIe (Gen 1) Electrical Idle Detect Threshold *(9)*														
VRX-IDLE-DETDIFF (p-p)	Compliance pattern	65	—	175	65	—	175	65	—	175	65	—	175	mV
Serial RapidIO® (SRIO) Transmit Jitter Generation *(5)*														
Deterministic jitter (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	—	—	0.17	—	—	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	—	—	0.35	UI
SRIO Receiver Jitter Tolerance *(5)*														
Deterministic jitter tolerance (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.37			> 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			> 0.55			UI
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			> 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			> 0.1			UI
	Jitter frequency = 20 MHz Data rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.1			> 0.1			> 0.1			> 0.1			UI
GIGE Transmit Jitter Generation *(6)*														
Deterministic jitter (peak-to-peak)	Pattern = CRPAT	—	—	0.14	—	—	0.14	—	—	0.14	—	—	0.14	UI

Table 1–40. Transceiver Block Jitter Specifications for Arria II GX Devices *(Note 1)* (Part 4 of 10)

Symbol/ Description	Conditions	I3			C4			C5, I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Total jitter (peak-to-peak)	Pattern = CRPAT	—	—	0.279	—	—	0.279	—	—	0.279	—	—	0.279	UI
GIGE Receiver Jitter Tolerance *(6)*														
Deterministic jitter tolerance (peak-to-peak)	Pattern = CJPAT	> 0.4			> 0.4			> 0.4			> 0.4			UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Pattern = CJPAT	> 0.66			> 0.66			> 0.66			> 0.66			UI
HiGig Transmit Jitter Generation *(7)*														
Deterministic jitter (peak-to-peak)	Data rate = 3.75 Gbps Pattern = CJPAT	—	—	0.17	—	—	0.17	—	—	—	—	—	—	UI
Total jitter (peak-to-peak)	Data rate = 3.75 Gbps Pattern = CJPAT	—	—	0.35	—	—	0.35	—	—	—	—	—	—	UI
HiGig Receiver Jitter Tolerance *(7)*														
Deterministic jitter tolerance (peak-to-peak)	Data rate = 3.75 Gbps Pattern = CJPAT	> 0.37			> 0.37			—	—	—	—	—	—	UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Data rate = 3.75 Gbps Pattern = CJPAT	> 0.65			> 0.65			—	—	—	—	—	—	UI
Sinusoidal jitter tolerance (peak-to-peak)	Jitter frequency = 22.1 KHz Data rate = 3.75 Gbps Pattern = CJPAT	> 8.5			> 8.5			—	—	—	—	—	—	UI
	Jitter frequency = 1.875MHz Data rate = 3.75 Gbps Pattern = CJPAT	> 0.1			> 0.1			—	—	—	—	—	—	UI
	Jitter frequency = 20 MHz Data rate = 3.75 Gbps Pattern = CJPAT	> 0.1			> 0.1			—	—	—	—	—	—	UI

Table 1–40. Transceiver Block Jitter Specifications for Arria II GX Devices *(Note 1)* (Part 5 of 10)

Symbol/ Description	Conditions	I3			C4			C5, I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
SDI Transmitter Jitter Generation *(8)*														
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	—	—	0.2	—	—	UI
	Data rate = 2.97 Gbps (3G) pattern = Color bar Low-frequency Roll-off = 100 KHz	0.3	—	—	0.3	—	—	0.3	—	—	0.3	—	—	UI
SDI Receiver Jitter Tolerance *(8)*														
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			> 2			UI
	Jitter frequency = 100 KHz Data rate = 2.97 Gbps (3G) Pattern = single line scramble color bar	> 0.3			> 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			> 0.3			UI

Table 1–40. Transceiver Block Jitter Specifications for Arria II GX Devices *(Note 1)* (Part 6 of 10)

Symbol/ Description	Conditions	I3			C4			C5, I5			C6			Unit
		Min	Typ	Max	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			> 1		UI
	Jitter frequency = 100 KHz Data rate = 1.485 Gbps (HD) Pattern = 75% color bar		> 0.2			> 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			> 0.2		UI
SATA Transmit Jitter Generation *(10)*														
Total jitter at 1.5 Gbps (G1)	Compliance pattern	—	—	0.55	—	—	0.55	—	—	0.55	—	—	0.55	UI
Deterministic jitter at 1.5 Gbps (G1)	Compliance pattern	—	—	0.35	—	—	0.35	—	—	0.35	—	—	0.35	UI
Total jitter at 3.0 Gbps (G2)	Compliance pattern	—	—	0.55	—	—	0.55	—	—	0.55	—	—	0.55	UI
Deterministic jitter at 3.0 Gbps (G2)	Compliance pattern	—	—	0.35	—	—	0.35	—	—	0.35	—	—	0.35	UI
Total jitter at 6.0 Gbps (G3)	Compliance pattern	—	—	0.52	—	—	—	—	—	—	—	—	—	UI
Random jitter at 6.0 Gbps (G3)	Compliance pattern	—	—	0.18	—	—	—	—	—	—	—	—	—	UI
SATA Receiver Jitter Tolerance *(10)*														
Total jitter tolerance at 1.5 Gbps (G1)	Compliance pattern		> 0.65			> 0.65			> 0.65			> 0.65		UI
Deterministic jitter tolerance at 1.5 Gbps (G1)	Compliance pattern		> 0.35			> 0.35			> 0.35			> 0.35		UI
SSC modulation frequency at 1.5 Gbps (G1)	Compliance pattern		33			33			33			33		kHz

Table 1–40. Transceiver Block Jitter Specifications for Arria II GX Devices *(Note 1)* (Part 7 of 10)

Symbol/ Description	Conditions	I3			C4			C5, I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
SSC modulation deviation at 1.5 Gbps (G1)	Compliance pattern		5700			5700			5700			5700		ppm
RX differential skew at 1.5 Gbps (G1)	Compliance pattern		80			80			80			80		ps
RX AC common mode voltage at 1.5 Gbps (G1)	Compliance pattern		150			150			150			150		mV
Total jitter tolerance at 3.0 Gbps (G2)	Compliance pattern		> 0.65			> 0.65			> 0.65			> 0.65		UI
Deterministic jitter tolerance at 3.0 Gbps (G2)	Compliance pattern		> 0.35			> 0.35			> 0.35			> 0.35		UI
SSC modulation frequency at 3.0 Gbps (G2)	Compliance pattern		33			33			33			33		kHz
SSC modulation deviation at 3.0 Gbps (G2)	Compliance pattern		5700			5700			5700			5700		ppm
RX differential skew at 3.0 Gbps (G2)	Compliance pattern		75			75			75			75		ps
RX AC common mode voltage at 3.0 Gbps (G2)	Compliance pattern		150			150			150			150		mV
Total jitter tolerance at 6.0 Gbps (G3)	Compliance pattern		> 0.60			> 0.60			> 0.60			> 0.60		UI
Random jitter tolerance at 6.0 Gbps (G3)	Compliance pattern		> 0.18			> 0.18			> 0.18			> 0.18		UI
SSC modulation frequency at 6.0 Gbps (G3)	Compliance pattern		33			33			33			33		kHz
SSC modulation deviation at 6.0 Gbps (G3)	Compliance pattern		5700			5700			5700			5700		ppm
RX differential skew at 6.0 Gbps (G3)	Compliance pattern		30			30			30			30		ps
RX AC common mode voltage at 6.0 Gbps (G3)	Compliance pattern		100			100			100			100		mV

Table 1–40. Transceiver Block Jitter Specifications for Arria II GX Devices *(Note 1)* (Part 8 of 10)

Symbol/ Description	Conditions	I3			C4			C5, I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
CPRI Transmit Jitter Generation *(11)*														
Total jitter	E.6.HV, E.12.HV Pattern = CJPAT	—	—	0.279	—	—	0.279	—	—	0.279	—	—	0.279	UI
	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJTPAT	—	—	0.35	—	—	0.35	—	—	0.35	—	—	0.35	UI
Deterministic jitter	E.6.HV, E.12.HV Pattern = CJPAT	—	—	0.14	—	—	0.14	—	—	0.14	—	—	0.14	UI
	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJTPAT	—	—	0.17	—	—	0.17	—	—	0.17	—	—	0.17	UI
CPRI Receiver Jitter Tolerance *(11)*														
Total jitter tolerance	E.6.HV, E.12.HV Pattern = CJPAT	> 0.66			> 0.66			> 0.66			> 0.66			UI
Deterministic jitter tolerance	E.6.HV, E.12.HV Pattern = CJPAT	> 0.4			> 0.4			> 0.4			> 0.4			UI
Total jitter tolerance	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJTPAT	> 0.65			> 0.65			> 0.65			> 0.65			UI
	E.60.LV Pattern = PRBS31	> 0.6			—			—			—			UI
Deterministic jitter tolerance	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJTPAT	> 0.37			> 0.37			> 0.37			> 0.37			UI
	E.60.LV Pattern = PRBS31	> 0.45			—			—			—			UI
Combined deterministic and random jitter tolerance	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJTPAT	> 0.55			> 0.55			> 0.55			> 0.55			UI
OBSAI Transmit Jitter Generation *(12)*														
Total jitter at 768 Mbps, 1536 Mbps, and 3072 Mbps	REFCLK = 153.6 MHz Pattern = CJPAT	—	—	0.35	—	—	0.35	—	—	0.35	—	—	0.35	UI
Deterministic jitter at 768 Mbps, 1536 Mbps, and 3072 Mbps	REFCLK = 153.6 MHz Pattern = CJPAT	—	—	0.17	—	—	0.17	—	—	0.17	—	—	0.17	UI

Table 1–40. Transceiver Block Jitter Specifications for Arria II GX Devices *(Note 1)* (Part 9 of 10)

Symbol/ Description	Conditions	I3			C4			C5, I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
OBSAI Receiver Jitter Tolerance *(12)*														
Deterministic jitter tolerance at 768 Mbps, 1536 Mbps, and 3072 Mbps	Pattern = CJPAT	> 0.37			> 0.37			> 0.37			> 0.37			UI
Combined deterministic and random jitter tolerance at 768 Mbps, 1536 Mbps, and 3072 Mbps	Pattern = CJPAT	> 0.55			> 0.55			> 0.55			> 0.55			UI
Sinusoidal jitter tolerance at 768 Mbps	Jitter frequency = 5.4 KHz Pattern = CJPAT	> 8.5			> 8.5			> 8.5			> 8.5			UI
	Jitter frequency = 460.8 KHz to 20 MHz Pattern = CJPAT	> 0.1			> 0.1			> 0.1			> 0.1			UI
Sinusoidal jitter tolerance at 1536 Mbps	Jitter frequency = 10.9 KHz Pattern = CJPAT	> 8.5			> 8.5			> 8.5			> 8.5			UI
	Jitter frequency = 921.6 KHz to 20 MHz Pattern = CJPAT	> 0.1			> 0.1			> 0.1			> 0.1			UI

Table 1–40. Transceiver Block Jitter Specifications for Arria II GX Devices *(Note 1)* (Part 10 of 10)

Symbol/ Description	Conditions	I3			C4			C5, I5			C6			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Sinusoidal jitter tolerance at 3072 Mbps	Jitter frequency = 21.8 KHz Pattern = CJPAT		> 8.5			> 8.5			> 8.5			> 8.5		UI
	Jitter frequency = 1843.2 KHz to 20 MHz Pattern = CJPAT		> 0.1			> 0.1			> 0.1			> 0.1		UI

Notes to Table 1–40:

(1) Dedicated `refclk` pins are used to drive the input reference clocks. The jitter numbers are valid for the stated conditions only.
(2) The jitter numbers for SONET/SDH are compliant to the GR-253-CORE Issue 3 Specification.
(3) The jitter numbers for XAUI are compliant to the IEEE802.3ae-2002 Specification.
(4) The jitter numbers for PCIe are compliant to the PCIe Base Specification 2.0.
(5) The jitter numbers for SRIO are compliant to the RapidIO Specification 1.3.
(6) The jitter numbers for GIGE are compliant to the IEEE802.3-2002 Specification.
(7) The jitter numbers for HiGig are compliant to the IEEE802.3ae-2002 Specification.
(8) The HD-SDI and 3G-SDI jitter numbers are compliant to the SMPTE292M and SMPTE424M Specifications.
(9) Arria II PCIe receivers are compliant to this specification provided the VTX_CM-DC-ACTIVEIDLE-DELTA of the upstream transmitter is less than 50 mV.
(10) The jitter numbers for Serial Advanced Technology Attachment (SATA) are compliant to the Serial ATA Revision 3.0 Specification.
(11) The jitter numbers for Common Public Radio Interface (CPRI) are compliant to the CPRI Specification V3.0.
(12) The jitter numbers for Open Base Station Architecture Initiative (OBSAI) are compliant to the OBSAI RP3 Specification V4.1.

Table 1–41 lists the transceiver jitter specifications for all supported protocols for Arria II GZ devices.

Table 1–41. Transceiver Block Jitter Specifications for Arria II GZ Devices *(Note 1)*, *(2)* (Part 1 of 7)

Symbol/ Description	Conditions	–C3 and –I3			–C4 and –I4			Unit
		Min	Typ	Max	Min	Typ	Max	
SONET/SDH Transmit Jitter Generation *(3)*								
Peak-to-peak jitter at 622.08 Mbps	Pattern = PRBS15	—	—	0.1	—	—	0.1	UI
RMS jitter at 622.08 Mbps	Pattern = PRBS15	—	—	0.01	—	—	0.01	UI
Peak-to-peak jitter at 2488.32 Mbps	Pattern = PRBS15	—	—	0.1	—	—	0.1	UI
RMS jitter at 2488.32 Mbps	Pattern = PRBS15	—	—	0.01	—	—	0.01	UI
SONET/SDH Receiver Jitter Tolerance *(3)*								
Jitter tolerance at 622.08 Mbps	Jitter frequency = 0.03 KHz Pattern = PRBS15		> 15			> 15		UI
	Jitter frequency = 25 KHZ Pattern = PRBS15		> 1.5			> 1.5		UI
	Jitter frequency = 250 KHz Pattern = PRBS15		> 0.15			> 0.15		UI

Table 1–41. Transceiver Block Jitter Specifications for Arria II GZ Devices *(Note 1)*, *(2)* (Part 2 of 7)

Symbol/ Description	Conditions	–C3 and –I3			–C4 and –I4			Unit
		Min	Typ	Max	Min	Typ	Max	
Jitter tolerance at 2488.32 Mbps	Jitter frequency = 0.06 KHz Pattern = PRBS15		> 15			> 15		UI
	Jitter frequency = 100 KHZ Pattern = PRBS15		> 1.5			> 1.5		UI
	Jitter frequency = 1 MHz Pattern = PRBS15		> 0.15			> 0.15		UI
	Jitter frequency = 10 MHz Pattern = PRBS15		> 0.15			> 0.15		UI
Fibre Channel Transmit Jitter Generation *(4)*, *(5)*								
Total jitter FC-1	Pattern = CRPAT	—	—	0.23	—	—	0.23	UI
Deterministic jitter FC-1	Pattern = CRPAT	—	—	0.11	—	—	0.11	UI
Total jitter FC-2	Pattern = CRPAT	—	—	0.33	—	—	0.33	UI
Deterministic jitter FC-2	Pattern = CRPAT	—	—	0.2	—	—	0.2	UI
Total jitter FC-4	Pattern = CRPAT	—	—	0.52	—	—	0.52	UI
Deterministic jitter FC-4	Pattern = CRPAT	—	—	0.33	—	—	0.33	UI
Fibre Channel Receiver Jitter Tolerance *(4)*, *(6)*								
Deterministic jitter FC-1	Pattern = CJTPAT		> 0.37			> 0.37		UI
Random jitter FC-1	Pattern = CJTPAT		> 0.31			> 0.31		UI
Sinusoidal jitter FC-1	Fc/25000		> 1.5			> 1.5		UI
	Fc/1667		> 0.1			> 0.1		UI
Deterministic jitter FC-2	Pattern = CJTPAT		> 0.33			> 0.33		UI
Random jitter FC-2	Pattern = CJTPAT		> 0.29			> 0.29		UI
Sinusoidal jitter FC-2	Fc/25000		> 1.5			> 1.5		UI
	Fc/1667		> 0.1			> 0.1		UI
Deterministic jitter FC-4	Pattern = CJTPAT		> 0.33			> 0.33		UI
Random jitter FC-4	Pattern = CJTPAT		> 0.29			> 0.29		UI
Sinusoidal jitter FC-4	Fc/25000		> 1.5			> 1.5		UI
	Fc/1667		> 0.1			> 0.1		UI
XAUI Transmit Jitter Generation *(7)*								
Total jitter at 3.125 Gbps	Pattern = CJPAT	—	—	0.3	—	—	0.3	UI
Deterministic jitter at 3.125 Gbps	Pattern = CJPAT	—	—	0.17	—	—	0.17	UI
XAUI Receiver Jitter Tolerance *(7)*								
Total jitter	—		> 0.65			> 0.65		UI
Deterministic jitter	—		> 0.37			> 0.37		UI

Table 1–41. Transceiver Block Jitter Specifications for Arria II GZ Devices *(Note 1)*, *(2)* (Part 3 of 7)

Symbol/ Description	Conditions	–C3 and –I3			–C4 and –I4			Unit
		Min	Typ	Max	Min	Typ	Max	
Peak-to-peak jitter	Jitter frequency = 22.1 KHz		> 8.5			> 8.5		UI
Peak-to-peak jitter	Jitter frequency = 1.875 MHz		> 0.1			> 0.1		UI
Peak-to-peak jitter	Jitter frequency = 20 MHz		> 0.1			> 0.1		UI
PCIe Transmit Jitter Generation *(8)*								
Total jitter at 2.5 Gbps (Gen1)—x1, x4, and x8	Compliance pattern	—	—	0.25	—	—	0.25	UI
Total jitter at 5 Gbps (Gen2)—x1, x4, and x8	Compliance pattern	—	—	0.25	—	—	—	UI
PCIe Receiver Jitter Tolerance *(8)*								
Total jitter at 2.5 Gbps (Gen1)	Compliance pattern		> 0.6			> 0.6		UI
Total jitter at 5 Gbps (Gen2)	Compliance pattern		Not supported			Not supported		UI
PCIe (Gen 1) Electrical Idle Detect Threshold								
$V_{RX-IDLE-DETDIFFp-p}$ *(9)*	Compliance pattern	65	—	175	65	—	175	UI
SRIO Transmit Jitter Generation *(10)*								
Deterministic jitter (peak-to-peak)	Data rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	—	—	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	UI
SRIO Receiver Jitter Tolerance *(10)*								
Deterministic jitter tolerance (peak-to-peak)	Data rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT		> 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		UI
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		UI
	Jitter frequency = 1.875 MHz Data rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT		> 0.1			> 0.1		UI
	Jitter frequency = 20 MHz Data rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT		> 0.1			> 0.1		UI
GIGE Transmit Jitter Generation *(11)*								
Deterministic jitter (peak-to-peak)	Pattern = CRPAT	—	—	0.14	—	—	0.14	UI
Total jitter (peak-to-peak)	Pattern = CRPAT	—	—	0.279	—	—	0.279	UI

Table 1–41. Transceiver Block Jitter Specifications for Arria II GZ Devices *(Note 1)*, *(2)* (Part 4 of 7)

Symbol/ Description	Conditions	–C3 and –I3			–C4 and –I4			Unit
		Min	Typ	Max	Min	Typ	Max	
GIGE Receiver Jitter Tolerance *(11)*								
Deterministic jitter tolerance (peak-to-peak)	Pattern = CJPAT		> 0.4			> 0.4		UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Pattern = CJPAT		> 0.66			> 0.66		UI
HiGig Transmit Jitter Generation								
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								
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
Sinusoidal jitter tolerance (peak-to-peak)	Jitter frequency = 22.1 KHz Data rate = 3.75 Gbps Pattern = CJPAT		> 8.5		—	—	—	UI
	Jitter frequency = 22.1 KHz Data rate = 3.75 Gbps Pattern = CJPAT		> 0.1		—	—	—	UI
	Jitter frequency = 22.1 KHz Data rate = 3.75 Gbps Pattern = CJPAT		> 0.1		—	—	—	UI
(OIF) CEI Transmitter Jitter Generation								
Total jitter (peak-to-peak)	Data rate = 6.375 Gbps Pattern = PRBS15 BER = 10^{-12}	—	—	0.3	—	—	0.3	UI
(OIF) CEI Receiver Jitter Tolerance								
Deterministic jitter tolerance (peak-to-peak)	Data rate = 6.375 Gbps Pattern = PRBS31 BER = 10^{-12}		> 0.675		—	—	—	UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Data rate = 6.375 Gbps Pattern = PRBS31 BER = 10^{-12}		> 0.988		—	—	—	UI

Table 1–41. Transceiver Block Jitter Specifications for Arria II GZ Devices *(Note 1)*, *(2)* (Part 5 of 7)

Symbol/ Description	Conditions	–C3 and –I3			–C4 and –I4			Unit
		Min	Typ	Max	Min	Typ	Max	
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 38.2 KHz Data rate = 6.375 Gbps Pattern = PRBS31 BER = 10^{-12}	> 0.5			—	—	—	UI
	Jitter Frequency = 3.82 MHz Data rate = 6.375 Gbps Pattern = PRBS31 BER = 10^{-12}	> 0.05			—	—	—	UI
	Jitter Frequency = 20 MHz Data rate = 6.375 Gbps Pattern = PRBS31 BER = 10^{-12}	> 0.05			—	—	—	UI
SDI Transmitter Jitter Generation *(12)*								
Alignment jitter (peak-to-peak)	Data rate = 1.485 Gbps (HD) Pattern = color bar Low-frequency roll-off = 100 KHz	0.2	—	—	0.2	—	—	UI
	Data rate = 2.97 Gbps (3G) Pattern = color bar Low-frequency roll-off = 100 KHz	0.3	—	—	0.3	—	—	UI
SDI Receiver Jitter Tolerance *(12)*								
Sinusoidal jitter tolerance (peak-to-peak)	Jitter frequency = 15 KHz Data rate = 2.97 Gbps (3G) Pattern = single line scramble color bar	> 2			> 2			UI
	Jitter frequency = 100 KHz Data rate = 2.97 Gbps (3G) Pattern = single line scramble color bar	> 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			UI
Sinusoidal jitter tolerance (peak-to-peak)	Jitter frequency = 20 KHz Data rate = 1.485 Gbps (HD) pattern = 75% color bar	> 1			> 1			UI
	Jitter frequency = 100 KHz Data rate = 1.485 Gbps (HD) Pattern = 75% color bar	> 0.2			> 0.2			UI
	Jitter frequency = 148.5 MHz Data rate = 1.485 Gbps (HD) Pattern = 75% color bar	> 0.2			> 0.2			UI
SAS Transmit Jitter Generation *(13)*								
Total jitter at 1.5 Gbps (G1)	Pattern = CJPAT	—	—	0.55	—	—	0.55	UI
Deterministic jitter at 1.5 Gbps (G1)	Pattern = CJPAT	—	—	0.35	—	—	0.35	UI
Total jitter at 3.0 Gbps (G2)	Pattern = CJPAT	—	—	0.55	—	—	0.55	UI

Table 1–41. Transceiver Block Jitter Specifications for Arria II GZ Devices *(Note 1)*, *(2)* (Part 6 of 7)

Symbol/ Description	Conditions	–C3 and –I3			–C4 and –I4			Unit
		Min	Typ	Max	Min	Typ	Max	
Deterministic jitter at 3.0 Gbps (G2)	Pattern = CJPAT	—	—	0.35	—	—	0.35	UI
Total jitter at 6.0 Gbps (G3)	Pattern = CJPAT	—	—	0.25	—	—	0.25	UI
Random jitter at 6.0 Gbps (G3)	Pattern = CJPAT	—	—	0.15	—	—	0.15	UI
SAS Receiver Jitter Tolerance *(13)*								
Total jitter tolerance at 1.5 Gbps (G1)	Pattern = CJPAT	—	—	0.65	—	—	0.65	UI
Deterministic jitter tolerance at 1.5 Gbps (G1)	Pattern = CJPAT	—	—	0.35	—	—	0.35	UI
Sinusoidal jitter tolerance at 1.5 Gbps (G1)	Jitter frequency = 900 KHz to 5 MHz Pattern = CJTPAT BER = 1E-12	> 0.1			> 0.1			UI
CPRI Transmit Jitter Generation *(14)*								
Total jitter	E.6.HV, E.12.HV Pattern = CJPAT	—	—	0.279	—	—	0.279	UI
	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJPAT	—	—	0.35	—	—	0.35	UI
Deterministic jitter	E.6.HV, E.12.HV Pattern = CJPAT	—	—	0.14	—	—	0.14	UI
	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJPAT	—	—	0.17	—	—	0.17	UI
CPRI Receiver Jitter Tolerance *(14)*								
Total jitter tolerance	E.6.HV, E.12.HV Pattern = CJPAT	> 0.66			> 0.66			UI
Deterministic jitter tolerance	E.6.HV, E.12.HV Pattern = CJPAT	> 0.4			> 0.4			UI
Total jitter tolerance	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJPAT	> 0.65			> 0.65			UI
Deterministic jitter tolerance	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJPAT	> 0.37			> 0.37			UI
Combined deterministic and random jitter tolerance	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJPAT	> 0.55			> 0.55			UI
OBSAI Transmit Jitter Generation *(15)*								
Total jitter at 768 Mbps, 1536 Mbps, and 3072 Mbps	REFCLK = 153.6 MHz Pattern CJPAT	—	—	0.35	—	—	0.35	UI
Deterministic jitter at 768 MBps, 1536 Mbps, and 3072 Mbps	REFCLK = 153.6 MHz Pattern CJPAT	—	—	0.17	—	—	0.17	UI

Table 1–41. Transceiver Block Jitter Specifications for Arria II GZ Devices *(Note 1)*, *(2)* (Part 7 of 7)

Symbol/ Description	Conditions	–C3 and –I3			–C4 and –I4			Unit
		Min	Typ	Max	Min	Typ	Max	
OBSAI Receiver Jitter Tolerance *(15)*								
Deterministic jitter tolerance at 768 Mbps, 1536 Mbps, and 3072 Mbps	Pattern = CJPAT		> 0.37			> 0.37		UI
Combined deterministic and random jitter tolerance at 768 Mbps, 1536 Mbps, and 3072 Mbps	Pattern = CJPAT		> 0.55			> 0.55		UI
Sinusoidal jitter tolerance at 768 Mbps	Jitter frequency = 5.4 KHz Pattern = CJPAT		> 8.5			> 8.5		UI
	Jitter frequency = 460 MHz to 20 MHz Pattern = CJPAT		> 0.1			> 0.1		UI
Sinusoidal jitter tolerance at 1536 Mbps	Jitter frequency = 10.9 KHz Pattern = CJPAT		> 8.5			> 8.5		UI
	Jitter frequency = 921.6 MHz to 20 MHz Pattern = CJPAT		> 0.1			> 0.1		UI
Sinusoidal jitter tolerance at 3072 Mbps	Jitter frequency = 21.8 KHz Pattern = CJPAT		> 8.5			> 8.5		UI
	Jitter frequency = 1843.2 MHz to 20 MHz Pattern = CJPAT		> 0.1			> 0.1		UI

Notes to Table 1–41:

(1) Dedicated `refclk` pins were used to drive the input reference clocks.
(2) The jitter numbers are valid for the stated conditions only.
(3) The jitter numbers for SONET/SDH are compliant to the GR-253-CORE Issue 3 Specification.
(4) The jitter numbers for Fibre Channel are compliant to the FC-PI-4 Specification revision 6.10.
(5) The Fibre Channel transmitter jitter generation numbers are compliant to the specification at the δ_T inter operability point.
(6) The Fibre Channel receiver jitter tolerance numbers are compliant to the specification at the δ_R interpretability point.
(7) The jitter numbers for XAUI are compliant to the IEEE802.3ae-2002 Specification.
(8) The jitter numbers for PCIe are compliant to the PCIe Base Specification 2.0.
(9) Arria II GZ PCIe receivers are compliant to this specification provided the $V_{TX-CM-DC-ACTIVEIDLE-DELTA}$ of the upstream transmitter is less than 50 mV.
(10) The jitter numbers for SRIO are compliant to the RapidIO Specification 1.3.
(11) The jitter numbers for GIGE are compliant to the IEEE802.3-2002 Specification.
(12) The HD-SDI and 3G-SDI jitter numbers are compliant to the SMPTE292M and SMPTE424M Specifications.
(13) The jitter numbers for Serial Attached SCSI (SAS) are compliant to the SAS-2.1 Specification.
(14) The jitter numbers for CPRI are compliant to the CPRI Specification V3.0.
(15) The jitter numbers for OBSAI are compliant to the OBSAI RP3 Specification V4.1.

Core Performance Specifications for the Arria II Device Family

This section describes the clock tree, phase-locked loop (PLL), digital signal processing (DSP), embedded memory, configuration, and JTAG specifications for Arria II GX and GZ devices.

Clock Tree Specifications

Table 1–42 lists the clock tree specifications for Arria II GX devices.

Table 1–42. Clock Tree Performance for Arria II GX Devices

Clock Network	Performance			Unit
	I3, C4	C5,I5	C6	
GCLK and RCLK	500	500	400	MHz
PCLK	420	350	280	MHz

Table 1–43 lists the clock tree specifications for Arria II GZ devices.

Table 1–43. Clock Tree Performance for Arria II GZ Devices

Clock Network	Performance		Unit
	–C3 and –I3	–C4 and –I4	
GCLK and RCLK	700	500	MHz
PCLK	500	450	MHz

PLL Specifications

Table 1–44 lists the PLL specifications for Arria II GX devices.

Table 1–44. PLL Specifications for Arria II GX Devices (Part 1 of 3)

Symbol	Description	Min	Typ	Max	Unit
f_{IN}	Input clock frequency (from clock input pins residing in right/top/bottom banks) (–4 Speed Grade)	5	—	670 *(1)*	MHz
	Input clock frequency (from clock input pins residing in right/top/bottom banks) (–5 Speed Grade)	5	—	622 *(1)*	MHz
	Input clock frequency (from clock input pins residing in right/top/bottom banks) (–6 Speed Grade)	5	—	500 *(1)*	MHz
f_{INPFD}	Input frequency to the PFD	5	—	325	MHz
f_{VCO}	PLL VCO operating Range *(2)*	600	—	1,400	MHz
f_{INDUTY}	Input clock duty cycle	40	—	60	%
$f_{EINDUTY}$	External feedback clock input duty cycle	40	—	60	%
t_{INCCJ} *(3)*, *(4)*	Input clock cycle-to-cycle jitter (Frequency ≥ 100 MHz)	—	—	0.15	UI (p–p)
	Input clock cycle-to-cycle jitter (Frequency ≤ 100 MHz)	—	—	±750	ps (p–p)

Table 1–44. PLL Specifications for Arria II GX Devices (Part 2 of 3)

Symbol	Description	Min	Typ	Max	Unit
f_{OUT}	Output frequency for internal global or regional clock (–4 Speed Grade)	—	—	500	MHz
	Output frequency for internal global or regional clock (–5 Speed Grade)	—	—	500	MHz
	Output frequency for internal global or regional clock (–6 Speed Grade)	—	—	400	MHz
f_{OUT_EXT}	Output frequency for external clock output (–4 Speed Grade)	—	—	670 *(5)*	MHz
	Output frequency for external clock output (–5 Speed Grade)	—	—	622 *(5)*	MHz
	Output frequency for external clock output (–6 Speed Grade)	—	—	500 *(5)*	MHz
$t_{OUTDUTY}$	Duty cycle for external clock output (when set to 50%)	45	50	55	%
t_{OUTPJ_DC}	Dedicated clock output period jitter ($f_{OUT} \geq 100$ MHz)	—	—	300	ps (p–p)
	Dedicated clock output period jitter ($f_{OUT} < 100$ MHz)	—	—	30	mUI (p–p)
t_{OUTCCJ_DC}	Dedicated clock output cycle-to-cycle jitter ($f_{OUT} \geq 100$ MHz)	—	—	300	ps (p–p)
	Dedicated clock output cycle-to-cycle jitter ($f_{OUT} < 100$ MHz)	—	—	30	mUI (p–p)
f_{OUTPJ_IO}	Regular I/O clock output period jitter ($f_{OUT} \geq 100$ MHz)	—	—	650	ps (p–p)
	Regular I/O clock output period jitter ($f_{OUT} < 100$ MHz)	—	—	65	mUI (p–p)
f_{OUTCCJ_IO}	Regular I/O clock output cycle-to-cycle jitter ($f_{OUT} \geq 100$ MHz)	—	—	650	ps (p–p)
	Regular I/O clock output cycle-to-cycle jitter ($f_{OUT} < 100$ MHz)	—	—	65	mUI (p–p)
$t_{CONFIGPLL}$	Time required to reconfigure PLL scan chains	—	3.5	—	SCANCLK cycles
$t_{CONFIGPHASE}$	Time required to reconfigure phase shift	—	1	—	SCANCLK cycles
$f_{SCANCLK}$	SCANCLK frequency	—	—	100	MHz
t_{LOCK}	Time required to lock from end of device configuration	—	—	1	ms
t_{DLOCK}	Time required to lock dynamically (after switchover or reconfiguring any non-post-scale counters/delays)	—	—	1	ms
f_{CLBW}	PLL closed-loop low bandwidth	—	0.3	—	MHz
	PLL closed-loop medium bandwidth	—	1.5	—	MHz
	PLL closed-loop high bandwidth	—	4	—	MHz
t_{PLL_PSERR}	Accuracy of PLL phase shift	—	—	±50	ps
t_{ARESET}	Minimum pulse width on `areset` signal	10	—	—	ns

Table 1–44. PLL Specifications for Arria II GX Devices (Part 3 of 3)

Symbol	Description	Min	Typ	Max	Unit
$t_{CASC_OUTJITTER_PERIOD_DEDCLK}$ *(6), (7)*	Period Jitter for dedicated clock output in cascaded PLLs (FOUT ≥ 100 MHz)	—	—	425	ps (p-p)
	Period Jitter for dedicated clock output in cascaded PLLs (FOUT ≤ 100 MHz)	—	—	42.5	mUI (p-p)

Notes to Table 1–44:

(1) f_{IN} is limited by the 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) A high-input jitter directly affects the PLL output jitter. To have low PLL output clock jitter, you must provide a clean-clock source, which is less than 200 ps.

(4) F_{REF} is fIN/N when N = 1.

(5) This specification is limited by the lower of the two: I/O f_{MAX} or f_{OUT} of the PLL.

(6) Peak-to-peak jitter with a probability level of 10^{-12} (14 sigma, 99.99999999974404% confidence level). The output jitter specification applies to the intrinsic jitter of the PLL, when an input jitter of 30 ps is applied. The external memory interface clock output jitter specifications use a different measurement method and are available in Table 1–62 on page 1–70.

(7) The cascaded PLL specification is only applicable with the following condition:
a. Upstream PLL: 0.59 Mhz ≤ Upstream PLL BW < 1 MHz
b. Downstream PLL: Downstream PLL BW > 2 MHz

Table 1–45 lists the PLL specifications for Arria II GZ devices when operating in both the commercial junction temperature range (0° to 85°C) and the industrial junction temperature range (-40° to 100°C).

Table 1–45. PLL Specifications for Arria II GZ Devices (Part 1 of 2)

Symbol	Parameter	Min	Typ	Max	Unit
f_{IN}	Input clock frequency (–3 speed grade)	5	—	717 *(1)*	MHz
	Input clock frequency (–4 speed grade)	5	—	717 *(1)*	MHz
f_{INPFD}	Input frequency to the PFD	5	—	325	MHz
f_{VCO}	PLL VCO operating range (–3 speed grade)	600	—	1,300	MHz
	PLL VCO operating range (–4 speed grade)	600	—	1,300	MHz
$t_{EINDUTY}$	Input clock or external feedback clock input duty cycle	40	—	60	%
f_{OUT}	Output frequency for internal global or regional clock (–3 speed grade)	—	—	700 *(2)*	MHz
	Output frequency for internal global or regional clock (–4 speed grade)	—	—	500 *(2)*	MHz
f_{OUT_EXT}	Output frequency for external clock output (–3 speed grade)	—	—	717 *(2)*	MHz
	Output frequency for external clock output (–4 speed grade)	—	—	717 *(2)*	MHz
$t_{OUTDUTY}$	Duty cycle for external clock output (when set to 50%)	45	50	55	%
t_{FCOMP}	External feedback clock compensation time	—	—	10	ns
$t_{CONFIGPLL}$	Time required to reconfigure scan chain	—	3.5	—	scanclk cycles
$t_{CONFIGPHASE}$	Time required to reconfigure phase shift	—	1	—	scanclk cycles
$f_{SCANCLK}$	scanclk frequency	—	—	100	MHz
t_{LOCK}	Time required to lock from end-of-device configuration or de-assertion of areset	—	—	1	ms

Table 1–45. PLL Specifications for Arria II GZ Devices (Part 2 of 2)

Symbol	Parameter	Min	Typ	Max	Unit
t_{DLOCK}	Time required to lock dynamically (after switchover or reconfiguring any non-post-scale counters/delays)	—	—	1	ms
f_{CLBW}	PLL closed-loop low bandwidth	—	0.3	—	MHz
	PLL closed-loop medium bandwidth	—	1.5	—	MHz
	PLL closed-loop high bandwidth *(7)*	—	4	—	MHz
t_{PLL_PSERR}	Accuracy of PLL phase shift	—	—	±50	ps
t_{ARESET}	Minimum pulse width on the `areset` signal	10	—	—	ns
t_{INCCJ} *(3)*, *(4)*	Input clock cycle to cycle jitter ($F_{REF} \geq 100$ MHz)	—	—	0.15	UI (p-p)
	Input clock cycle to cycle jitter ($F_{REF} < 100$ MHz)	—	—	±750	ps (p-p)
t_{OUTPJ_DC} *(5)*	Period Jitter for dedicated clock output ($F_{OUT} \geq 100$ MHz)	—	—	175	ps (p-p)
	Period Jitter for dedicated clock output ($F_{OUT} < 100$ MHz)	—	—	17.5	mUI (p-p)
t_{OUTCCJ_DC} *(5)*	Cycle to Cycle Jitter for dedicated clock output ($F_{OUT} \geq 100$ MHz)	—	—	175	ps (p-p)
	Cycle to Cycle Jitter for dedicated clock output ($F_{OUT} < 100$ MHz)	—	—	17.5	mUI (p-p)
t_{OUTPJ_IO} *(5)*, *(8)*	Period Jitter for clock output on regular I/O ($F_{OUT} \geq 100$ MHz)	—	—	600	ps (p-p)
	Period Jitter for clock output on regular I/O ($F_{OUT} < 100$ MHz)	—	—	60	mUI (p-p)
t_{OUTCCJ_IO} *(5)*, *(8)*	Cycle to Cycle Jitter for clock output on regular I/O ($F_{OUT} \geq 100$ MHz)	—	—	600	ps (p-p)
	Cycle to Cycle Jitter for clock output on regular I/O ($F_{OUT} < 100$ MHz)	—	—	60	mUI (p-p)
$t_{CASC_OUTPJ_DC}$ *(5)*, *(6)*	Period Jitter for dedicated clock output in cascaded PLLs ($F_{OUT} \geq 100$MHz)	—	—	250	ps (p-p)
	Period Jitter for dedicated clock output in cascaded PLLs ($F_{OUT} < 100$MHz)	—	—	25	mUI (p-p)
f_{DRIFT}	Frequency drift after PFDENA is disabled for duration of 100 us	—	—	±10	%

Notes to Table 1–45:

(1) This specification is limited in the Quartus II software by the I/O maximum frequency. The maximum I/O frequency is different for each I/O standard.

(2) This specification is limited by the lower of the two: I/O F_{MAX} or F_{OUT} of the PLL.

(3) A high input jitter directly affects the PLL output jitter. To have low PLL output clock jitter, you must provide a clean clock source that is less than 120 ps.

(4) F_{REF} is fIN/N when N = 1.

(5) Peak-to-peak jitter with a probability level of 10^{-12} (14 sigma, 99.99999999974404% confidence level). The output jitter specification applies to the intrinsic jitter of the PLL, when an input jitter of 30 ps is applied. The external memory interface clock output jitter specifications use a different measurement method and are available in Table 1–64 on page 1–71.

(6) The cascaded PLL specification is only applicable with the following condition:
a. Upstream PLL: 0.59 Mhz ≤ Upstream PLL BW < 1 MHz
b. Downstream PLL: Downstream PLL BW > 2 MHz

(7) High bandwidth PLL settings are not supported in external feedback mode.

(8) External memory interface clock output jitter specifications use a different measurement method, which is available in Table 1–63 on page 1–71.

DSP Block Specifications

Table 1–46 lists the DSP block performance specifications for Arria II GX devices.

Table 1–46. DSP Block Performance Specifications for Arria II GX Devices *(Note 1)*

Mode	Resources Used	Performance				Unit
	Number of Multipliers	C4	I3	C5,I5	C6	
9 × 9-bit multiplier	1	380	310	300	250	MHz
12 × 12-bit multiplier	1	380	310	300	250	MHz
18 × 18-bit multiplier	1	380	310	300	250	MHz
36 × 36-bit multiplier	1	350	270	270	220	MHz
18 × 36-bit high-precision multiplier adder mode	1	350	270	270	220	MHz
18 × 18-bit multiply accumulator	4	380	310	300	250	MHz
18 × 18-bit multiply adder	4	380	310	300	250	MHz
18 × 18-bit multiply adder-signed full precision	2	380	310	300	250	MHz
18 × 18-bit multiply adder with loopback *(2)*	2	275	220	220	180	MHz
36-bit shift (32-bit data)	1	350	270	270	220	MHz
Double mode	1	350	270	270	220	MHz

Notes to Table 1–46:

(1) Maximum is for a fully-pipelined block with **Round** and **Saturation** disabled.

(2) Maximum is for loopback input registers disabled, **Round** and **Saturation** disabled, pipeline and output registers enabled.

Table 1–47 lists the DSP block performance specifications for Arria II GZ devices.

Table 1–47. DSP Block Performance Specifications for Arria II GZ Devices *(Note 1)* (Part 1 of 2)

Mode	Resources Used	Performance		Unit
	Number of Multipliers	–3	–4	
9 × 9-bit multiplier	1	460	400	MHz
12 × 12-bit multiplier	1	500	440	MHz
18 × 18-bit multiplier	1	550	480	MHz
36 × 36-bit multiplier	1	440	380	MHz
18 × 18-bit multiply accumulator	4	440	380	MHz
18 × 18-bit multiply adder	4	470	410	MHz
18 × 18-bit multiply adder-signed full precision	2	450	390	MHz
18 × 18-bit multiply adder with loopback *(2)*	2	350	310	MHz
36-bit shift (32-bit data)	1	440	380	MHz

Table 1–47. DSP Block Performance Specifications for Arria II GZ Devices *(Note 1)* (Part 2 of 2)

Mode	Resources Used	Performance		Unit
	Number of Multipliers	–3	–4	
Double mode	1	440	380	MHz

Notes to Table 1–47:

(1) Maximum is for fully pipelined block with **Round** and **Saturation** disabled.

(2) Maximum for loopback input registers disabled, **Round** and **Saturation** disabled, and pipeline and output registers enabled.

Embedded Memory Block Specifications

Table 1–48 lists the embedded memory block specifications for Arria II GX devices.

Table 1–48. Embedded Memory Block Performance Specifications for Arria II GX Devices

Memory	Mode	Resources Used		Performance				Unit
		ALUTs	Embedded Memory	I3	C4	C5,I5	C6	
Memory Logic Array Block (MLAB)	Single port 64 × 10	0	1	450	500	450	378	MHz
	Simple dual-port 32 × 20 single clock	0	1	270	500	450	378	MHz
	Simple dual-port 64 × 10 single clock	0	1	428	500	450	378	MHz
M9K Block	Single-port 256 × 36	0	1	360	400	360	310	MHz
	Single-port 256 × 36, with the **read-during-write** option set to **Old Data**	0	1	250	280	250	210	MHz
	Simple dual-port 256 × 36 single CLK	0	1	360	400	360	310	MHz
	Single-port 256 × 36 single CLK, with the **read-during-write** option set to **Old Data**	0	1	250	280	250	210	MHz
	True dual port 512 × 18 single CLK	0	1	360	400	360	310	MHz
	True dual-port 512 × 18 single CLK, with the **read-during-write** option set to **Old Data**	0	1	250	280	250	210	MHz
	Min Pulse Width (clock high time)	—	—	900	850	950	1130	ps
	Min Pulse Width (clock low time)	—	—	730	690	770	920	ps

Table 1–49 lists the embedded memory block specifications for Arria II GZ devices.

Table 1–49. Embedded Memory Block Performance Specifications for Arria II GZ Devices *(Note 1)*

Memory	Mode	Resources Used		Performance				Unit
		ALUTs	TriMatrix Memory	C3	I3	C4	I4	
MLAB *(2)*	Single port 64 × 10	0	1	500	500	450	450	MHz
	Simple dual-port 32 × 20	0	1	500	500	450	450	MHz
	Simple dual-port 64 × 10	0	1	500	500	450	450	MHz
	ROM 64 × 10	0	1	500	500	450	450	MHz
	ROM 32 × 20	0	1	500	500	450	450	MHz
M9K Block *(2)*	Single-port 256 × 36	0	1	540	540	475	475	MHz
	Simple dual-port 256 × 36	0	1	490	490	420	420	MHz
	Simple dual-port 256 × 36, with the read-during-write option set to **Old Data**	0	1	340	340	300	300	MHz
	True dual port 512 × 18	0	1	430	430	370	370	MHz
	True dual-port 512 × 18, with the read-during-write option set to **Old Data**	0	1	335	335	290	290	MHz
	ROM 1 Port	0	1	540	540	475	475	MHz
	ROM 2 Port	0	1	540	540	475	475	MHz
	Min Pulse Width (clock high time)	—	—	800	800	850	850	ps
	Min Pulse Width (clock low time)	—	—	625	625	690	690	ps
M144K Block *(2)*	Single-port 2K × 72	0	1	440	400	380	350	MHz
	Simple dual-port 2K × 72	0	1	435	375	385	325	MHz
	Simple dual-port 2K × 72, with the read-during-write option set to **Old Data**	0	1	240	225	205	200	MHz
	Simple dual-port 2K × 64 (with ECC)	0	1	300	295	255	250	MHz
	True dual-port 4K × 36	0	1	375	350	330	310	MHz
	True dual-port 4K × 36, with the read-during-write option set to **Old Data**	0	1	230	225	205	200	MHz
	ROM 1 Port	0	1	500	450	435	420	MHz
	ROM 2 Port	0	1	465	425	400	400	MHz
	Min Pulse Width (clock high time)	—	—	755	860	860	950	ps
	Min Pulse Width (clock low time)	—	—	625	690	690	690	ps

Notes to Table 1–48:

(1) To achieve the maximum memory block performance, use a memory block clock that comes through global clock routing from an on-chip PLL set to 50% output duty cycle. Use the Quartus II software to report timing for this and other memory block clocking schemes.

(2) When you use the error detection CRC feature, there is no degradation in F_{MAX}.

Configuration

Table 1–50 lists the configuration mode specifications for Arria II GX and GZ devices.

Table 1–50. Configuration Mode Specifications for Arria II Devices

Programming Mode	DCLK Frequency			Unit
	Min	Typ	Max	
Passive serial	—	—	125	MHz
Fast passive parallel	—	—	125	MHz
Fast active serial (fast clock)	17	26	40	MHz
Fast active serial (slow clock)	8.5	13	20	MHz
Remote update only in fast AS mode	—	—	10	MHz

JTAG Specifications

Table 1–51 lists the JTAG timing parameters and values for Arria II GX and GZ devices.

Table 1–51. JTAG Timing Parameters and Values for Arria II Devices

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	ns
t_{JPZX}	JTAG port high impedance to valid output	—	14	ns
t_{JPXZ}	JTAG port valid output to high impedance	—	14	ns

Chip-Wide Reset (Dev_CLRn) Specifications

Table 1–52 lists the specifications for the chip-wide reset (Dev_CLRn) for Arria II GX and GZ devices.

Table 1–52. Chip-Wide Reset (Dev_CLRn) Specifications for Arria II Devices

Description	Min	Typ	Max	Unit
Dev_CLRn	500	—	—	µs

Periphery Performance

This section describes periphery performance, including high-speed I/O, external memory interface, and IOE programmable delay.

I/O performance supports several system interfaces, 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 with typical DDR2 SDRAM memory interface setup. I/O using general purpose I/O (GPIO) 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 1–53 lists the high-speed I/O timing for Arria II GX devices.

Table 1–53. High-Speed I/O Specifications for Arria II GX Devices (Part 1 of 4)

Symbol	Conditions	I3		C4		C5,I5		C6		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
Clock										
f_{HSCLK_IN} (input clock frequency)–Row I/O	Clock boost factor, W = 1 to 40 *(1)*	5	670	5	670	5	622	5	500	MHz
f_{HSCLK_IN} (input clock frequency)–Column I/O	Clock boost factor, W = 1 to 40 *(1)*	5	500	5	500	5	472.5	5	472.5	MHz
f_{HSCLK_OUT} (output clock frequency)–Row I/O	—	5	670	5	670	5	622	5	500	MHz
f_{HSCLK_OUT} (output clock frequency)–Column I/O	—	5	500	5	500	5	472.5	5	472.5	MHz

Table 1–53. High-Speed I/O Specifications for Arria II GX Devices (Part 2 of 4)

Symbol	Conditions	I3		C4		C5,I5		C6		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
Transmitter										
f_{HSDR_TX} (true LVDS output data rate)	SERDES factor, J = 3 to 10 (using dedicated SERDES)	150	1250 *(2)*	150	1250 *(2)*	150	1050 *(2)*	150	840	Mbps
	SERDES factor, J = 4 to 10 (using logic elements as SERDES)	*(3)*	945	*(3)*	945	*(3)*	840	*(3)*	740	Mbps
	SERDES factor, J = 2 (using DDR registers) and J = 1 (using SDR register)	*(3)*	*(3)*	*(3)*	*(3)*	*(3)*	*(3)*	*(3)*	*(3)*	Mbps
$f_{HSDR_TX_E3R}$ (emulated LVDS_E_3R output data rate) *(7)*	SERDES factor, J = 4 to 10	*(3)*	945	*(3)*	945	*(3)*	840	*(3)*	740	Mbps

Table 1–53. High-Speed I/O Specifications for Arria II GX Devices (Part 3 of 4)

Symbol	Conditions	I3		C4		C5,I5		C6		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
t_{TX_JITTER} *(4)*	True LVDS with dedicated SERDES (data rate 600–1,250 Mbps)	—	175	—	175	—	225	—	300	ps
	True LVDS with dedicated SERDES (data rate < 600 Mbps)	—	0.105	—	0.105	—	0.135	—	0.18	UI
	True LVDS and emulated LVDS_E_3R with logic elements as SERDES (data rate 600 – 945 Mbps)	—	260	—	260	—	300	—	350	ps
	True LVDS and emulated LVDS_E_3R with logic elements as SERDES (data rate < 600 Mbps)	—	0.16	—	0.16	—	0.18	—	0.21	UI
t_{TX_DCD}	True LVDS and emulated LVDS_E_3R	45	55	45	55	45	55	45	55	%
t_{RISE} and t_{FALL}	True LVDS and emulated LVDS_E_3R	—	200	—	200	—	225	—	250	ps
TCCS	True LVDS *(5)*	—	150	—	150	—	175	—	200	ps
	Emulated LVDS_E_3R	—	200	—	200	—	250	—	300	ps
Receiver ***(6)***										
True differential I/O standards - $f_{HSDRDPA}$ (data rate)	SERDES factor J = 3 to 10	150	1250	150	1250	150	1050	150	840	Mbps

Table 1–53. High-Speed I/O Specifications for Arria II GX Devices (Part 4 of 4)

Symbol	Conditions	I3		C4		C5,I5		C6		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{HSDR} (data rate)	SERDES factor J = 3 to 10	*(3)*	945 *(7)*	*(3)*	945 *(7)*	*(3)*	740 *(7)*	*(3)*	640 *(7)*	Mbps
	SERDES factor J = 2 (using DDR registers)	*(3)*	*(7)*	*(3)*	*(7)*	*(3)*	*(7)*	*(3)*	*(7)*	Mbps
	SERDES factor J = 1 (using SDR registers)	*(3)*	*(7)*	*(3)*	*(7)*	*(3)*	*(7)*	*(3)*	*(7)*	Mbps
Soft-CDR PPM tolerance	Soft-CDR mode	—	300	—	300	—	300	—	300	±PPM
DPA run length	DPA mode	—	10,000	—	10,000	—	10,000	—	10,000	UI
Sampling window (SW)	Non-DPA mode *(5)*	—	300	—	300	—	350	—	400	ps

Notes to Table 1–53:

(1) $f_{HSCLK_IN} = f_{HSDR} / W$. Use W to determine the supported selection of input reference clock frequencies for the desired data rate.

(2) Applicable for interfacing with DPA receivers only. For interfacing with non-DPA receivers, you must calculate the leftover timing margin in the receiver by performing link timing closure analysis. For Arria II GX transmitter to Arria II GX non-DPA receiver, the maximum supported data rate is 945 Mbps. For data rates above 840 Mbps, perform PCB trace compensation by adjusting the PCB trace length for LVDS channels to improve channel-to-channel skews.

(3) The minimum and maximum specification depends 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) The specification is only applicable under the influence of core noise.

(5) Applicable for true LVDS using dedicated SERDES only.

(6) Dedicated SERDES and DPA features are only available on the right banks.

(7) You must calculate the leftover timing margin in the receiver by performing link timing closure analysis. You must consider the board skew margin, transmitter channel-to-channel skew, and the receiver sampling margin to determine the leftover timing margin.

Table 1–54 lists the high-speed I/O timing for Arria II GZ devices.

Table 1–54. High-Speed I/O Specifications for Arria II GZ Devices *(Note 1), (2), (10)* (Part 1 of 3)

Symbol	Conditions	C3, I3			C4, I4			Unit
		Min	Typ	Max	Min	Typ	Max	
Clock								
f_{HSCLK_in} (input clock frequency) true differential I/O standards	Clock boost factor W = 1 to 40 *(3)*	5	—	717	5	—	717	MHz
f_{HSCLK_in} (input clock frequency) single ended I/O standards *(9)*	Clock boost factor W = 1 to 40 *(3)*	5	—	717	5	—	717	MHz
f_{HSCLK_in} (input clock frequency) single ended I/O standards *(10)*	Clock boost factor W = 1 to 40 *(3)*	5	—	420	5	—	420	MHz

Table 1–54. High-Speed I/O Specifications for Arria II GZ Devices *(Note 1), (2), (10)* (Part 2 of 3)

Symbol	Conditions	C3, I3			C4, I4			Unit
		Min	Typ	Max	Min	Typ	Max	
f_{HSCLK_OUT} (output clock frequency)	—	5	—	717 *(7)*	5	—	717 *(7)*	MHz
Transmitter								
f_{HSDR} (true LVDS output data rate)	SERDES factor, J = 3 to 10 (using dedicated SERDES) *(8)*	*(4)*	—	1250	*(4)*	—	1250	Mbps
	SERDES factor J = 2, (using DDR registers)	*(4)*	—	*(5)*	*(4)*	—	*(5)*	Mbps
	SERDES factor J = 1, (uses an SDR register)	*(4)*	—	*(5)*	*(4)*	—	*(5)*	Mbps
f_{HSDR} (emulated LVDS_E_3R output data rate) *(5)*	SERDES factor J = 4 to 10	*(4)*	—	1152	*(4)*	—	800	Mbps
f_{HSDR} (emulated LVDS_E_1R output data rate)		*(4)*	—	200	*(4)*	—	200	Mbps
$t_{x\ Jitter}$	Total jitter for data rate, 600 Mbps to 1.6 Gbps	—	—	160	—	—	160	ps
	Total jitter for data rate, < 600 Mbps	—	—	0.1	—	—	0.1	UI
$t_{x\ Jitter}$ - emulated differential I/O standards with three external output resistor network	Total jitter for data rate, 600 Mbps to 1.25 Gbps	—	—	300	—	—	325	ps
	Total jitter for data rate < 600 Mbps	—	—	0.2	—	—	0.25	UI
$t_{x\ Jitter}$ - emulated differential I/O standards with one external output resistor network	—	—	—	0.15	—	—	0.15	UI
t_{DUTY}	TX output clock duty cycle for both True and emulated differential I/O standards	45	50	55	45	50	55	%

Table 1–54. High-Speed I/O Specifications for Arria II GZ Devices *(Note 1), (2), (10)* (Part 3 of 3)

Symbol	Conditions	C3, I3			C4, I4			Unit
		Min	Typ	Max	Min	Typ	Max	
t_{RISE} & t_{FALL}	True differential I/O standards	—	—	200	—	—	200	ps
	Emulated differential I/O standards with three external output resistor networks	—	—	250	—	—	300	ps
	Emulated differential I/O standards with one external output resistor	—	—	500	—	—	500	ps
TCCS	True LVDS	—	—	100	—	—	100	ps
	Emulated LVDS_E_3R	—	—	250	—	—	250	ps
Receiver								
True differential I/O standards - $f_{HSDRDPA}$ (data rate)	SERDES factor J = 3 to 10	150	—	1250	150	—	1250	Mbps
f_{HSDR} (data rate)	SERDES factor J = 3 to 10	*(4)*	—	*(6)*	*(4)*	—	*(6)*	Mbps
	SERDES factor J = 2, uses DDR registers	*(4)*	—	*(5)*	*(4)*	—	*(5)*	Mbps
	SERDES factor J = 1, uses an SDR register	*(4)*	—	*(5)*	*(4)*	—	*(5)*	Mbps
DPA run length	DPA mode	—	—	10000	—	—	10000	UI
Soft-CDR PPM tolerance	Soft-CDR mode	—	—	300	—	—	300	± PPM
Sampling Window (SW)	Non-DPA mode	—	—	300	—	—	300	ps

Notes to Table 1–54:

(1) When J = 3 to 10, use the SERDES block.

(2) When J = 1 or 2, bypass the SERDES block.

(3) Clock Boost Factor (W) is the ratio between input data rate to the input clock rate.

(4) The minimum specification depends on the clock source (for example, the PLL and clock pin) and the clock routing resource (global, regional, or local) that you use. The I/O differential buffer and input register do not have a minimum toggle rate.

(5) You must calculate the leftover timing margin in the receiver by performing link timing closure analysis. You must consider the board skew margin, transmitter channel-to-channel skew, and receiver sampling margin to determine leftover timing margin.

(6) You can estimate the achievable maximum data rate for non-DPA mode by performing link timing closure analysis. You must consider the board skew margin, transmitter delay margin, and the receiver sampling margin to determine the maximum data rate supported.

(7) This is achieved by using the LVDS and DPA clock network.

(8) If the receiver with DPA enabled and transmitter are using shared PLLs, the minimum data rate is 150 Mbps.

(9) This only applies to DPA and soft-CDR modes.

(10) This only applies to LVDS source synchronous mode.

Table 1–55 lists DPA lock time specifications for Arria II GX and GZ devices.

Table 1–55. DPA Lock Time Specifications for Arria II Devices *(Note 1)*, *(2)*, *(3)*

Standard	Training Pattern	Number of Data Transitions in One Repetition of the Training Pattern	Number of Repetitions per 256 Data Transitions *(4)*	Maximum
SPI-4	00000000001111111111	2	128	640 data transitions
Parallel Rapid I/O	00001111	2	128	640 data transitions
	10010000	4	64	640 data transitions
Miscellaneous	10101010	8	32	640 data transitions
	01010101	8	32	640 data transitions

Notes to Table 1–55:

(1) The DPA lock time is for one channel.
(2) One data transition is defined as a 0-to-1 or 1-to-0 transition.
(3) The DPA lock time stated in the table applies to both commercial and industrial grade.
(4) This is the number of repetitions for the stated training pattern to achieve the 256 data transitions.

Figure 1–5 shows the LVDS soft-CDR/DPA sinusoidal jitter tolerance specification for Arria II GZ devices at a data rate less than 1.25 Gbps and all the Arria II GX devices.

Figure 1–5. LVDS Soft-CDR/DPA Sinusoidal Jitter Tolerance Specification for All Arria II GX Devices and for Arria II GZ Devices at a Data Rate less than 1.25 Gbps

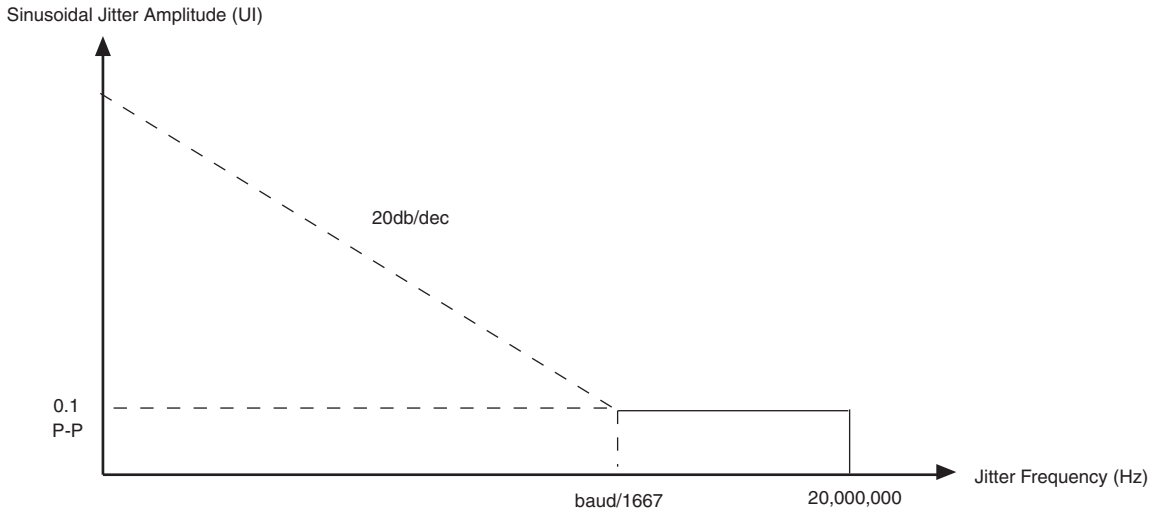


Figure 1–6 shows the LVDS soft-CDR/DPA sinusoidal jitter tolerance specification for Arria II GZ devices at 1.25 Gbps data rate.

Figure 1–6. LVDS Soft-CDR/DPA Sinusoidal Jitter Tolerance Specification for Arria II GZ Devices at a 1.25 Gbps Data Rate

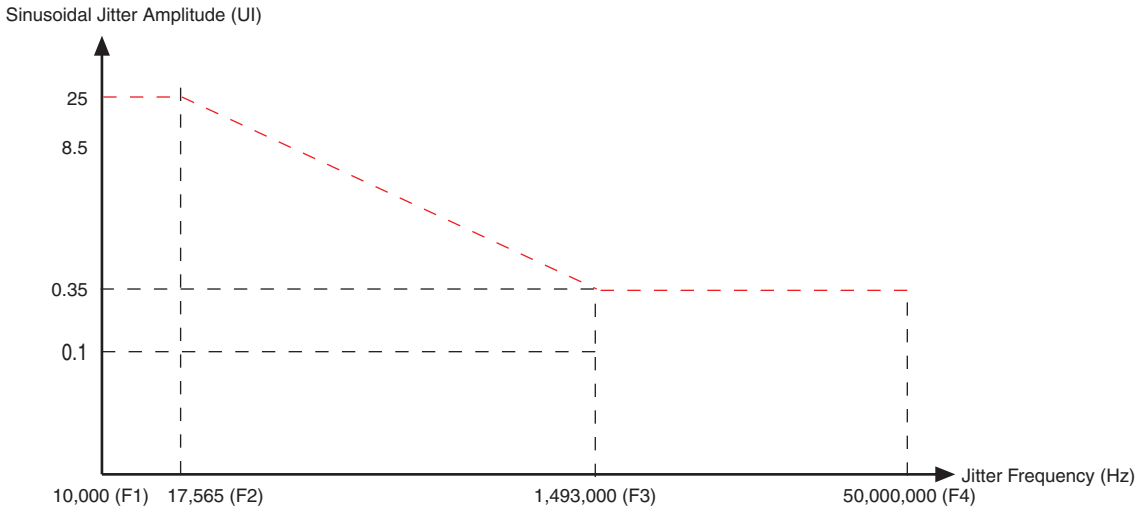


Table 1–56 lists the LVDS soft-CDR/DPA sinusoidal jitter tolerance specification for Arria II GZ devices at 1.25 Gbps data rate.

Table 1–56. LVDS Soft-CDR/DPA Sinusoidal Jitter Mask Values for Arria II GZ Devices at 1.25 Gbps Data Rate

Jitter Frequency (Hz)		Sinusoidal Jitter (UI)
F1	10,000	25.000
F2	17,565	25.000
F3	1,493,000	0.350
F4	50,000,000	0.350

External Memory Interface Specifications

For the maximum clock rate supported for Arria II GX and GZ device family, refer to the External Memory Interface Spec Estimator page on the Altera website.

Table 1–57 lists the external memory interface specifications for Arria II GX devices.

Table 1–57. External Memory Interface Specifications for Arria II GX Devices (Part 1 of 2)

Frequency Mode	Frequency Range (MHz)			Resolution (°)	DQS Delay Buffer Mode *(1)*	Number of Delay Chains
	C4	I3, C5, I5	C6			
0	90-140	90-130	90-110	22.5	Low	16
1	110-180	110-170	110-150	30	Low	12
2	140-220	140-210	140-180	36	Low	10
3	170-270	170-260	170-220	45	Low	8
4	220-340	220-310	220-270	30	High	12

Table 1–57. External Memory Interface Specifications for Arria II GX Devices (Part 2 of 2)

Frequency Mode	Frequency Range (MHz)			Resolution (°)	DQS Delay Buffer Mode *(1)*	Number of Delay Chains
	C4	I3, C5, I5	C6			
5	270-410	270-380	270-320	36	High	10
6	320-450	320-410	320-370	45	High	8

Note to Table 1–57:

(1) Low indicates a 6-bit DQS delay setting; high indicates a 5-bit DQS delay setting.

Table 1–58 lists the DLL frequency range specifications for Arria II GZ devices.

Table 1–58. DLL Frequency Range Specifications for Arria II GZ Devices

Frequency Mode	Frequency Range (MHz)		Available Phase Shift	DQS Delay Buffer Mode *(1)*	Number of Delay Chains
	–3	–4			
0	90-130	90-120	22.5°, 45°, 67.5°, 90°	Low	16
1	120-170	120-160	30°, 60°, 90°, 120°	Low	12
2	150-210	150-200	36°, 72°, 108°, 144°	Low	10
3	180-260	180-240	45°, 90°,135°, 180°	Low	8
4	240-320	240-290	30°, 60°, 90°, 120°	High	12
5	290-380	290-360	36°, 72°, 108°, 144°	High	10
6	360-450	360-450	45°, 90°, 135°, 180°	High	8
7	470-630	470-590	60°, 120°, 180°, 240°	High	6

Note to Table 1–58:

(1) Low indicates a 6-bit DQS delay setting; high indicates a 5-bit DQS delay setting.

Table 1–59 lists the DQS phase offset delay per stage for Arria II GX devices.

Table 1–59. DQS Phase Offset Delay Per Setting for Arria II GX Devices *(Note 1)*, *(2)*, *(3)*

Speed Grade	Min	Max	Unit
C4	7.0	13.0	ps
I3, C5, I5	7.0	15.0	ps
C6	8.5	18.0	ps

Notes to Table 1–59:

(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) The delay settings are linear.

Table 1–60 lists the DQS phase shift error for Arria II GX devices.

Table 1–60. DQS Phase Shift Error Specification for DLL-Delayed Clock (t_{DQS_PSERR}) for Arria II GX Devices *(Note 1)*

Number of DQS Delay Buffer	C4	I3, C5, I5	C6	Unit
1	26	30	36	ps
2	52	60	72	ps
3	78	90	108	ps
4	104	120	144	ps

Note to Table 1–60:

(1) This error specification is the absolute maximum and minimum error. For example, skew on three DQS delay buffers in a C4 speed grade is ± 78 ps or ± 39 ps.

Table 1–61 lists the DQS phase shift error for Arria II GZ devices.

Table 1–61. DQS Phase Shift Error Specification for DLL-Delayed Clock (t_{DQS_PSERR}) for Arria II GZ Devices *(Note 1)*

Number of DQS Delay Buffer	–3	–4	Unit
1	28	30	ps
2	56	60	ps
3	84	90	ps
4	112	120	ps

Note to Table 1–61:

(1) This error specification is the absolute maximum and minimum error. For example, skew on three DQS delay buffers in a 3 speed grade is ± 84 ps or ± 42 ps.

Table 1–62 lists the memory output clock jitter specifications for Arria II GX devices.

Table 1–62. Memory Output Clock Jitter Specification for Arria II GX Devices *(Note 1)*, *(2)*, *(3)*

Parameter	Clock Network	Symbol	–4		–5		–6		Unit
			Min	Max	Min	Max	Min	Max	
Clock period jitter	Global	$t_{JIT(per)}$	-100	100	-125	125	-125	125	ps
Cycle-to-cycle period jitter	Global	$t_{JIT(cc)}$	-200	200	-250	250	-250	250	ps
Duty cycle jitter	Global	$t_{JIT(duty)}$	-100	100	-125	125	-125	125	ps

Notes to Table 1–62:

(1) The memory output clock jitter measurements are for 200 consecutive clock cycles, as specified in the JEDEC DDR2/DDR3 SDRAM standard.

(2) The clock jitter specification applies to memory output clock pins generated using DDIO circuits clocked by a PLL output routed on a global clock network.

(3) The memory output clock jitter stated in Table 1–62 is applicable when an input jitter of 30 ps is applied.

Table 1–63 lists the memory output clock jitter specifications for Arria II GZ devices.

Table 1–63. Memory Output Clock Jitter Specification for Arria II GZ Devices *(Note 1)*, *(2)*, *(3)*

Parameter	Clock Network	Symbol	–3		–4		Unit
			Min	Max	Min	Max	
Clock period jitter	Regional	$t_{JIT(per)}$	-55	55	-55	55	ps
Cycle-to-cycle period jitter	Regional	$t_{JIT(cc)}$	-110	110	-110	110	ps
Duty cycle jitter	Regional	$t_{JIT(duty)}$	-82.5	82.5	-82.5	82.5	ps
Clock period jitter	Global	$t_{JIT(per)}$	-82.5	82.5	-82.5	82.5	ps
Cycle-to-cycle period jitter	Global	$t_{JIT(cc)}$	-165	165	-165	165	ps
Duty cycle jitter	Global	$t_{JIT(duty)}$	-90	90	-90	90	ps

Notes to Table 1–63:

(1) The memory output clock jitter measurements are for 200 consecutive clock cycles, as specified in the JEDEC DDR2/DDR3 SDRAM standard.

(2) The clock jitter specification applies to memory output clock pins generated using differential signal-splitter and DDIO circuits clocked by a PLL output routed on a regional or global clock network as specified. Altera recommends using regional clock networks whenever possible.

(3) The memory output clock jitter stated in Table 1–63 is applicable when an input jitter of 30 ps is applied.

Duty Cycle Distortion (DCD) Specifications

Table 1–64 lists the worst-case DCD specifications for Arria II GX devices.

Table 1–64. Duty Cycle Distortion on I/O Pins for Arria II GX Devices *(Note 1)*

Symbol	C4		I3, C5, I5		C6		Unit
	Min	Max	Min	Max	Min	Max	
Output Duty Cycle	45	55	45	55	45	55	%

Note to Table 1–64:

(1) The DCD specification applies to clock outputs from the PLL, global clock tree, IOE driving dedicated, and general purpose I/O pins.

Table 1–65 lists the worst-case DCD specifications for Arria II GZ devices.

Table 1–65. Duty Cycle Distortion on I/O Pins for Arria II GZ Devices *(Note 1)*

Symbol	C3, I3		C4, I4		Unit
	Min	Max	Min	Max	
Output Duty Cycle	45	55	45	55	%

Note to Table 1–65:

(1) The DCD specification applies to clock outputs from the PLL, global clock tree, IOE driving dedicated, and general purpose I/O pins.

IOE Programmable Delay

Table 1–66 lists the delay associated with each supported IOE programmable delay chain for Arria II GX devices.

Table 1–66. IOE Programmable Delay for Arria II GX Devices

Parameter	Available Settings (1)	Minimum Offset (2)	Maximum Offset								Unit
			Fast Model			Slow Model					
			I3	C4	I5	I3	C4	C5	I5	C6	
Output enable pin delay	7	0	0.413	0.442	0.413	0.814	0.713	0.796	0.801	0.873	ns
Delay from output register to output pin	7	0	0.339	0.362	0.339	0.671	0.585	0.654	0.661	0.722	ns
Input delay from pin to internal cell	52	0	1.494	1.607	1.494	2.895	2.520	2.733	2.775	2.944	ns
Input delay from pin to input register	52	0	1.493	1.607	1.493	2.896	2.503	2.732	2.774	2.944	ns
DQS bus to input register delay	4	0	0.074	0.076	0.074	0.140	0.124	0.147	0.147	0.167	ns

Notes to Table 1–66:

(1) The available setting for every delay chain starts with zero and ends with the specified maximum number of settings.

(2) The minimum offset represented in the table does not include intrinsic delay.

Table 1–67 lists the IOE programmable delay settings for Arria II GZ devices.

Table 1–67. IOE Programmable Delay for Arria II GZ Devices

Parameter	Available Settings (1)	Minimum Offset (2)	Maximum Offset						Unit
			Fast Model		Slow Model				
			Industrial	Commercial	C3	I3	C4	I4	
D1	15	0	0.462	0.505	0.795	0.801	0.857	0.864	ns
D2	7	0	0.234	0.232	0.372	0.371	0.407	0.405	ns
D3	7	0	1.700	1.769	2.927	2.948	3.157	3.178	ns
D4	15	0	0.508	0.554	0.882	0.889	0.952	0.959	ns
D5	15	0	0.472	0.500	0.799	0.817	0.875	0.882	ns
D6	6	0	0.186	0.195	0.319	0.321	0.345	0.347	ns

Notes to Table 1–67:

(1) You can set this value in the Quartus II software by selecting **D1**, **D2**, **D3**, **D4**, **D5**, and **D6** in the **Assignment Name** column.

(2) Minimum offset does not include the intrinsic delay.

I/O Timing

Altera offers two ways to determine I/O timing:

- Using the Microsoft Excel-based I/O Timing.
- Using the Quartus II Timing Analyzer.

The Microsoft Excel-based I/O Timing provides pin timing performance for each device density and speed grade. The data is typically used prior to designing the FPGA to get an estimate of the timing budget as part of the link timing analysis. The Quartus II timing analyzer provides a more accurate and precise I/O timing data based on the specifics of the design after place-and-route is complete.



The Microsoft Excel-based I/O Timing spreadsheet is downloadable from the Literature: Arria II Devices web page.

Glossary

Table 1–68 lists the glossary for this chapter.

Table 1–68. Glossary (Part 1 of 4)

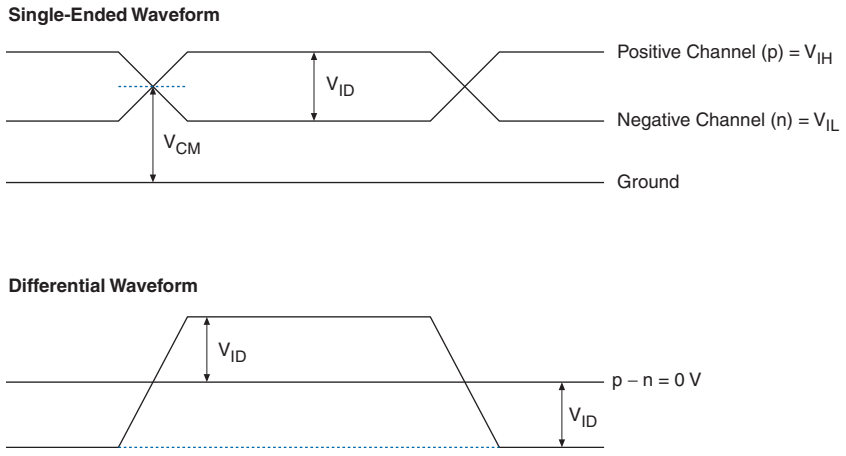
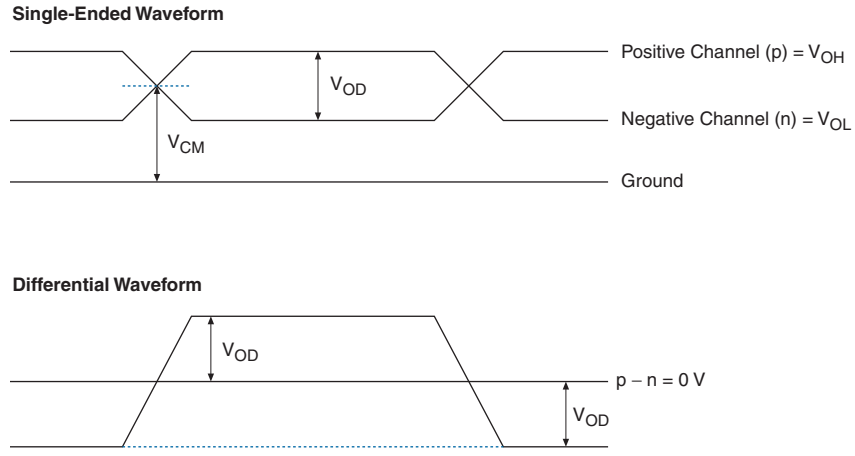
Letter	Subject	Definitions
A, B, C, D	Differential I/O Standards	*Receiver Input Waveforms*  *Transmitter Output Waveforms* 
E, F	f_{HSCLK}	Left/Right PLL input clock frequency.
	f_{HSDR}	High-speed I/O block: Maximum/minimum LVDS data transfer rate (f_{HSDR} = 1/TUI), non-DPA.
	$f_{HSDRDPA}$	High-speed I/O block: Maximum/minimum LVDS data transfer rate ($f_{HSDRDPA}$ = 1/TUI), DPA.

Table 1–68. Glossary (Part 2 of 4)

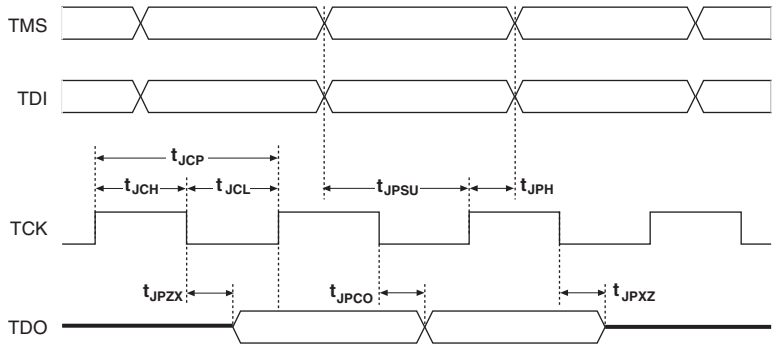
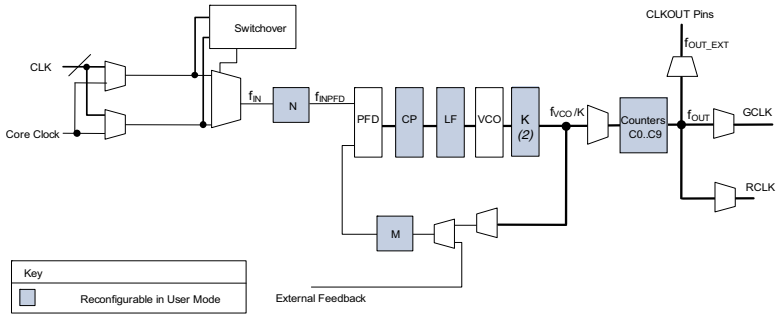
Letter	Subject	Definitions
G, H, I, J	J	High-speed I/O block: Deserialization factor (width of parallel data bus).
	JTAG Timing Specifications	JTAG Timing Specifications: 
K, L, M, N, O, P	PLL Specifications	PLL Specification parameters: **Diagram of PLL Specifications** ***(1)***  **Notes:** (1) CoreClock can only be fed by dedicated clock input pins or PLL outputs. (2) This is the VCO post-scale counter K.
Q, R	R_L	Receiver differential input discrete resistor (external to the Arria II device).

Table 1–68. Glossary (Part 3 of 4)

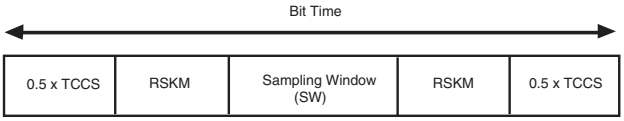
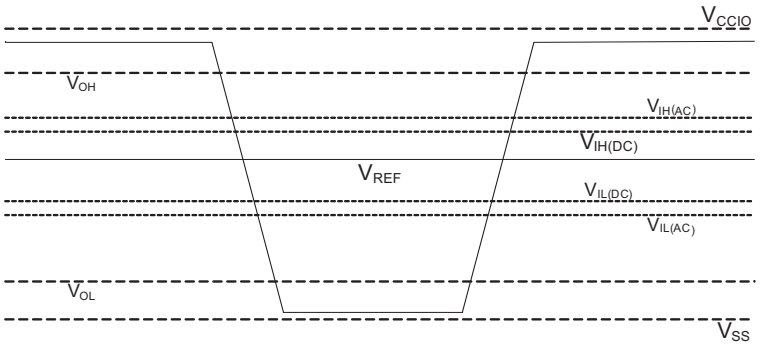
Letter	Subject	Definitions
S	**SW (sampling window)**	The period of time during which the data must be valid in order to capture it correctly. The setup and hold times determine the ideal strobe position within the sampling window: *Timing Diagram* 
	Single-ended Voltage Referenced I/O Standard	The JEDEC 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. After 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* 
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).
	t_{DUTY}	High-speed I/O block: Duty cycle on the 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)
	t_{FALL}	Signal high-to-low transition time (80-20%)
	t_{INCCJ}	Cycle-to-cycle jitter tolerance on the PLL clock input.
	t_{OUTPJ_IO}	Period jitter on the general purpose I/O driven by a PLL.
	t_{OUTPJ_DC}	Period jitter on the dedicated clock output driven by a PLL.
	t_{RISE}	Signal low-to-high transition time (20-80%).

Table 1–68. Glossary (Part 4 of 4)

Letter	Subject	Definitions
U, 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 is 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 is 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, X, Y, Z	W	High-speed I/O block: The clock boost factor.

Document Revision History

Table 1–69 lists the revision history for this chapter.

Table 1–69. Document Revision History (Part 1 of 2)

Date	Version	Changes
December 2013	4.4	Updated Table 1–34 and Table 1–35.
July 2012	4.3	■ Updated the $V_{CCH_GXBL/R}$ operating conditions in Table 1–6. ■ Finalized Arria II GZ information in Table 1–20. ■ Added BLVDS specification in Table 1–32 and Table 1–33. ■ Updated input and output waveforms in Table 1–68.
December 2011	4.2	■ Updated Table 1–32, Table 1–33, Table 1–34, Table 1–35, Table 1–40, Table 1–41, Table 1–54, and Table 1–67. ■ Minor text edits.
June 2011	4.1	■ Added Table 1–60. ■ Updated Table 1–32, Table 1–33, Table 1–38, Table 1–41, and Table 1–61. ■ Updated the “Switching Characteristics” section introduction. ■ Minor text edits.

Table 1–69. Document Revision History (Part 2 of 2)

Date	Version	Changes
December 2010	4.0	■ Added Arria II GZ information. ■ Added Table 1–61 with Arria II GX information. ■ Updated Table 1–1, Table 1–2, Table 1–5, Table 1–6, Table 1–7, Table 1–11, Table 1–35, Table 1–37, Table 1–40, Table 1–42, Table 1–44, Table 1–45, Table 1–57, Table 1–61, and Table 1–63. ■ Updated Figure 1–5. ■ Updated for the Quartus II version 10.0 release. ■ Updated the first paragraph for searchability. ■ Minor text edits.
July 2010	3.0	■ Updated Table 1–1, Table 1–4, Table 1–16, Table 1–19, Table 1–21, Table 1–23, Table 1–25, Table 1–26, Table 1–30, and Table 1–35 ■ Added Table 1–27 and Table 1–29. ■ Added I3 speed grade information to Table 1–19, Table 1–21, Table 1–22, Table 1–24, Table 1–25, Table 1–30, Table 1–32, Table 1–33, Table 1–34, and Table 1–35. ■ Updated the "Operating Conditions" section. ■ Removed "Preliminary" from Table 1–19, Table 1–21, Table 1–22, Table 1–23, Table 1–24, Table 1–25, Table 1–26, Table 1–28, Table 1–30, Table 1–32, Table 1–33, Table 1–34, and Figure 1–4. ■ Minor text edits.
March 2010	2.3	Updated for the Quartus II version 9.1 SP2 release: ■ Updated Table 1–3, Table 1–7, Table 1–19, Table 1–21, Table 1–22, Table 1–24, Table 1–25 and Table 1–33. ■ Updated "Recommended Operating Conditions" section. ■ Minor text edits.
February 2010	2.2	Updated Table 1–19.
February 2010	2.1	Updated for Arria II GX v9.1 SP1 release: ■ Updated Table 1–19, Table 1–23, Table 1–28, Table 1–30, and Table 1–33. ■ Added Figure 1–5. ■ Minor text edits.
November 2009	2.0	Updated for Arria II GX v9.1 release: ■ Updated Table 1–1, Table 1–4, Table 1–13, Table 1–14, Table 1–19, Table 1–15, Table 1–22, Table 1–24, and Table 1–28. ■ Added Table 1–6 and Table 1–33. ■ Added "Bus Hold" on page 1–5. ■ Added "IOE Programmable Delay" section. ■ Minor text edit.
June 2009	1.2	■ Updated Table 1–1, Table 1–3, Table 1–7, Table 1–8, Table 1–18, Table 1–23, Table 1–25, Table 1–26, Table 1–29, Table 1–30, Table 1–31, Table 1–32, and Table 1–33. ■ Added Table 1–32. ■ Updated Equation 1–1.
March 2009	1.1	Added "I/O Timing" section.
February 2009	1.0	Initial release.

2. Addendum for the Arria II Device Handbook

AIIGX53002-2.0

This chapter describes changes to the published version of the *Arria II Device Handbook.* All changes from Revision 1.1 of this chapter are now incorporated in the main handbook chapters.

Highlights

This information is now located in the *Overview for the Arria II Device Family* chapter.

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

This information is now located in the *Overview for the Arria II Device Family* chapter.

Auto-Calibrating External Memory Interfaces

This information is now located in the *Overview for the Arria II Device Family* chapter.

Connecting a Serial Configuration Device to an Arria II Device Family on AS Interface

This information is now located in the *Configuration, Design Security, and Remote System Upgrades in Arria II Devices* chapter.

Document Revision History

Table 2–1 lists the revision history for this chapter.

Table 2–1. Document Revision History

Date	Version	Changes
December 2010	2.0	■ Updated chapter titles.
July 2010	1.2	■ Moved the "Highlights", "High-Speed LVDS I/O with DPA and Soft CDR", and "Auto-Calibrating External Memory Interfaces" sections to the Arria II GX Device Family Overview chapter. ■ Moved the "Guidelines for Connecting Serial Configuration Device to Arria II GX Device Family on AS Interface" section to the Configuration, Design Security, and Remote System Upgrades in Arria II GX Devices chapter
March 2010	1.1	Added "Guidelines for Connecting Serial Configuration Device to Arria II GX Device Family on AS Interface"
February 2010	1.0	Initial release.

© 2010 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

This chapter provides additional information about the Arria II Device Handbook and Altera.

About this Handbook

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

How to Contact Altera

To locate the most up-to-date information about Altera products, refer to the following table.

Contact *(1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note to Table:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, **\qdesigns** directory, **D:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicate document titles. For example, *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	Indicate keyboard keys and menu names. For example, the 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`. The suffix `n` denotes an active-low signal. 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`).
↵	An angled arrow instructs you to press the Enter key.
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.
	A question mark directs you to a software help system with related information.
	The feet direct you to another document or website with related information.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The envelope links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.