# 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

# Contents

## Section II. I/O Interfaces for Arria II Devices

## 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



## Section III. System Integration for Arria II Devices

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

# 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

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.

- 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**

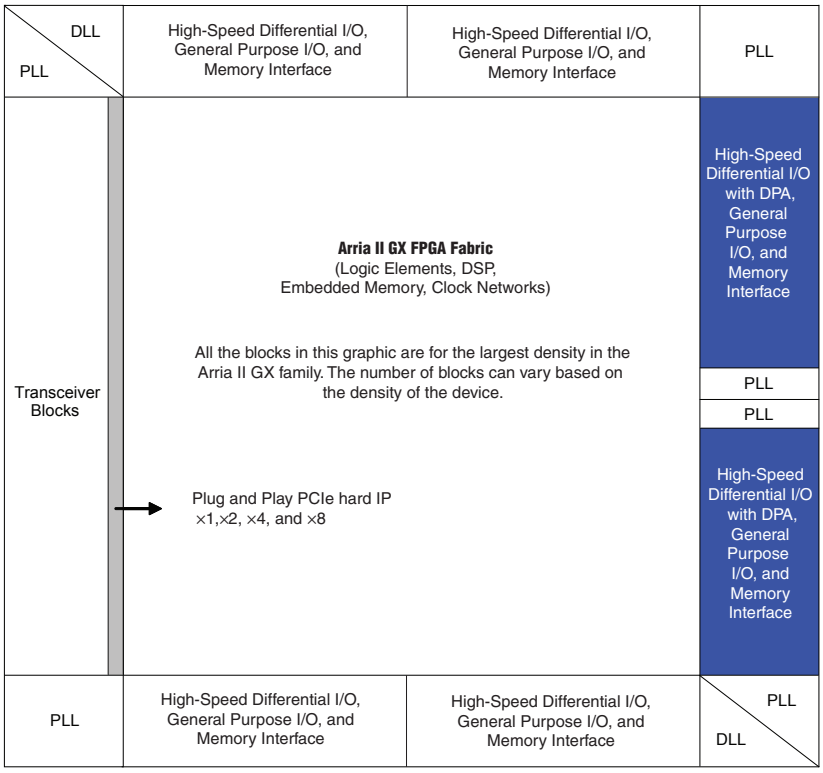
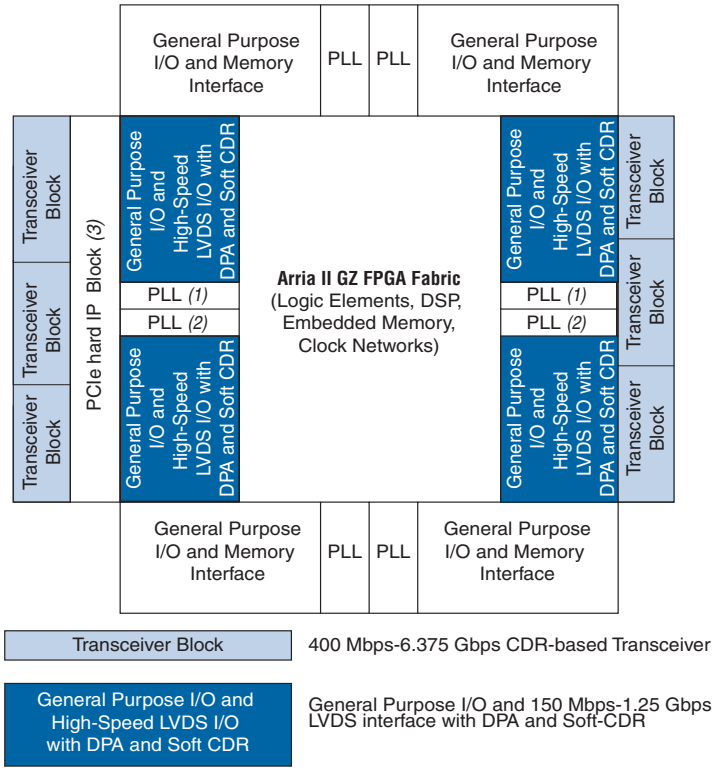


**Figure 1–2. Architecture Overview for Arria II GZ Device**




**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.<br>■ 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<br>■ Built-in circuitry for electrical idle generation and detection, receiver detect, power state transitions, lane reversal, and polarity inversion<br>■ 8B/10B encoder and decoder, receiver synchronization state machine, and ±300 parts per million (PPM) clock compensation circuitry<br>■ Options to use:<br>  ■ Hard IP Data Link Layer and Transaction Layer<br>  ■ Hard IP Data Link Layer and custom Soft IP Transaction Layer |
| XAUI/HiGig/HiGig+ | ■ Compliant to IEEE P802.3ae specification<br>■ 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<br>■ 8B/10B encoder and decoder, receiver synchronization state machine, lane deskew, and ±100 PPM clock compensation circuitry |
| GbE | ■ Compliant to IEEE 802.3 specification<br>■ Automatic idle ordered set (/I1/, /I2/) generation at the transmitter, depending on the current running disparity<br>■ 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<br>■ 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

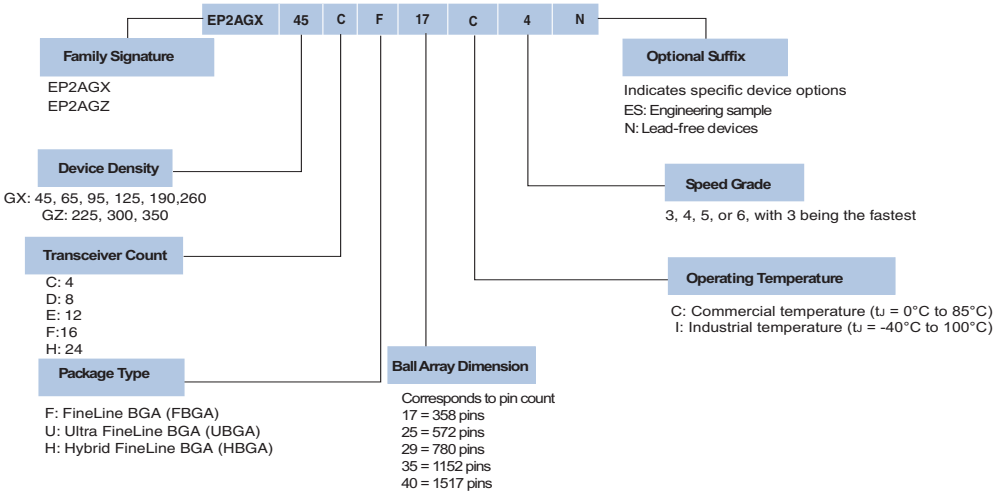
## JTAG Boundary Scan Testing

- 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.<br>■ Updated Table 1–10.<br>■ Updated the “Arria II Device Feature” section.<br>■ Added Table 1–6.<br>■ Minor text edits. |
| December 2010 | 4.0 | ■ Updated for the Quartus II software version 10.0 release<br>■ Added information about Arria II GZ devices<br>■ Updated Table 1–1, Table 1–4, Table 1–5, Table 1–6, Table 1–7, and Table 1–9<br>■ Added Table 1–3<br>■ Added Figure 1–2<br>■ Updated Figure 1–3<br>■ 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:<br>■ Added information about –I3 speed grade<br>■ Updated Table 1–1, Table 1–3, and Table 1–7<br>■ Updated Figure 1–2<br>■ Updated "Highlights" and "High-Speed LVDS I/O and DPA"section<br>■ Minor text edits |
| November 2009 | 2.0 | ■ Updated Table 1–1, Table 1–2, and Table 1–3<br>■ Updated "Configuration Features" section |
| June 2009 | 1.1 | ■ Updated Table 1–2.<br>■ 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:

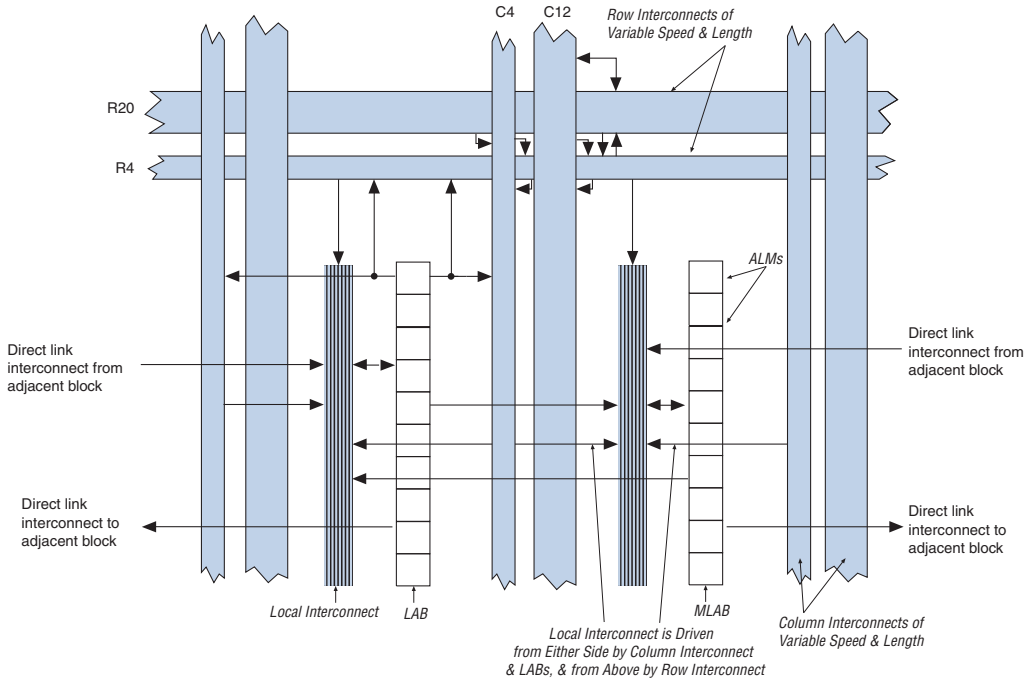


## Logic Array Blocks

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.

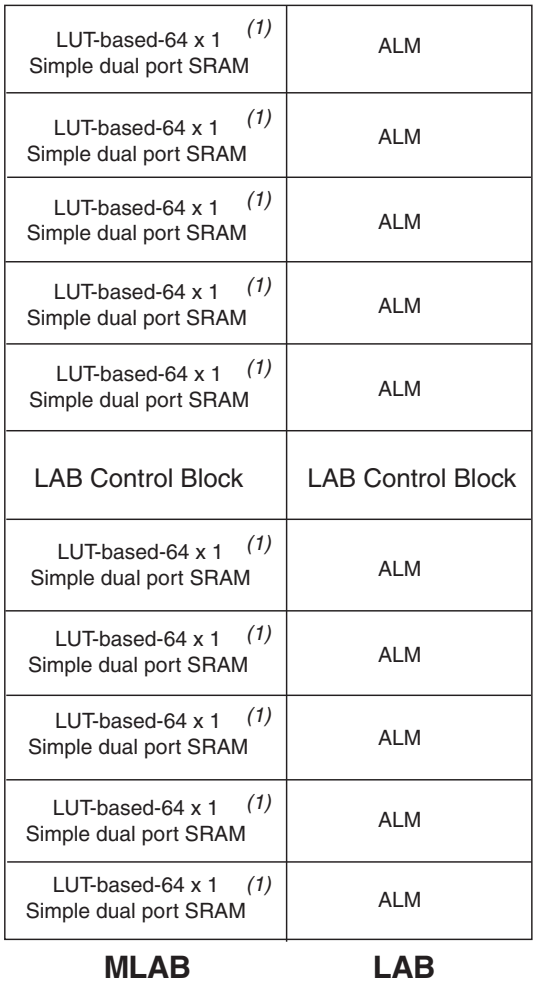
**Figure 2–1. LAB Structure in Arria II Devices**



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:**

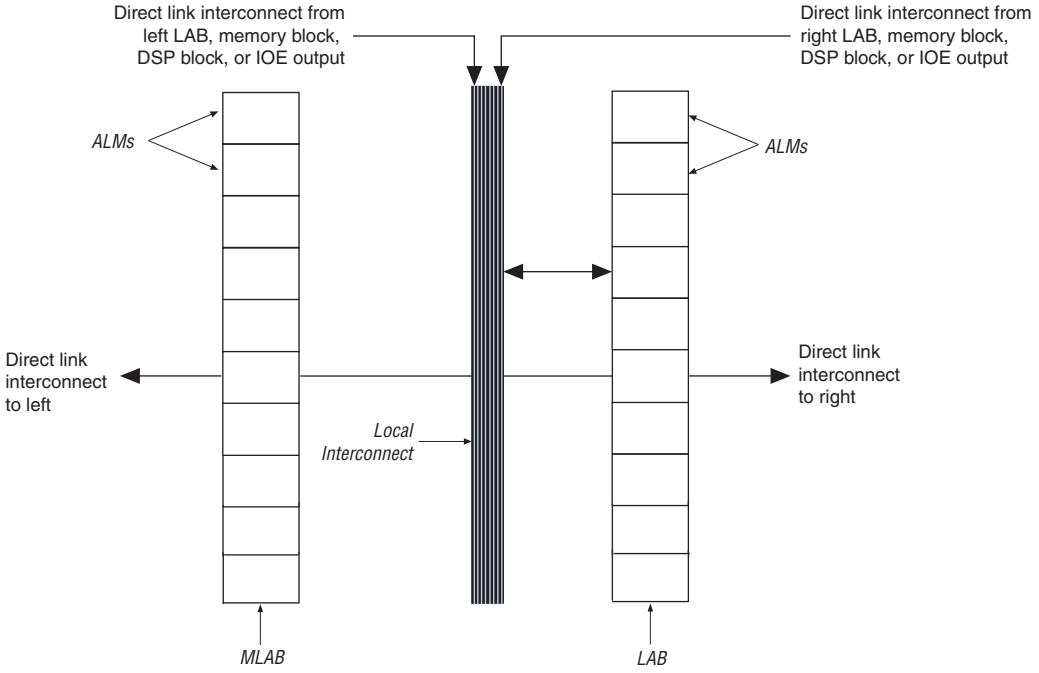
(1) You can use an MLAB ALM as a regular LAB ALM or configure it as a dual-port SRAM.

## 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.

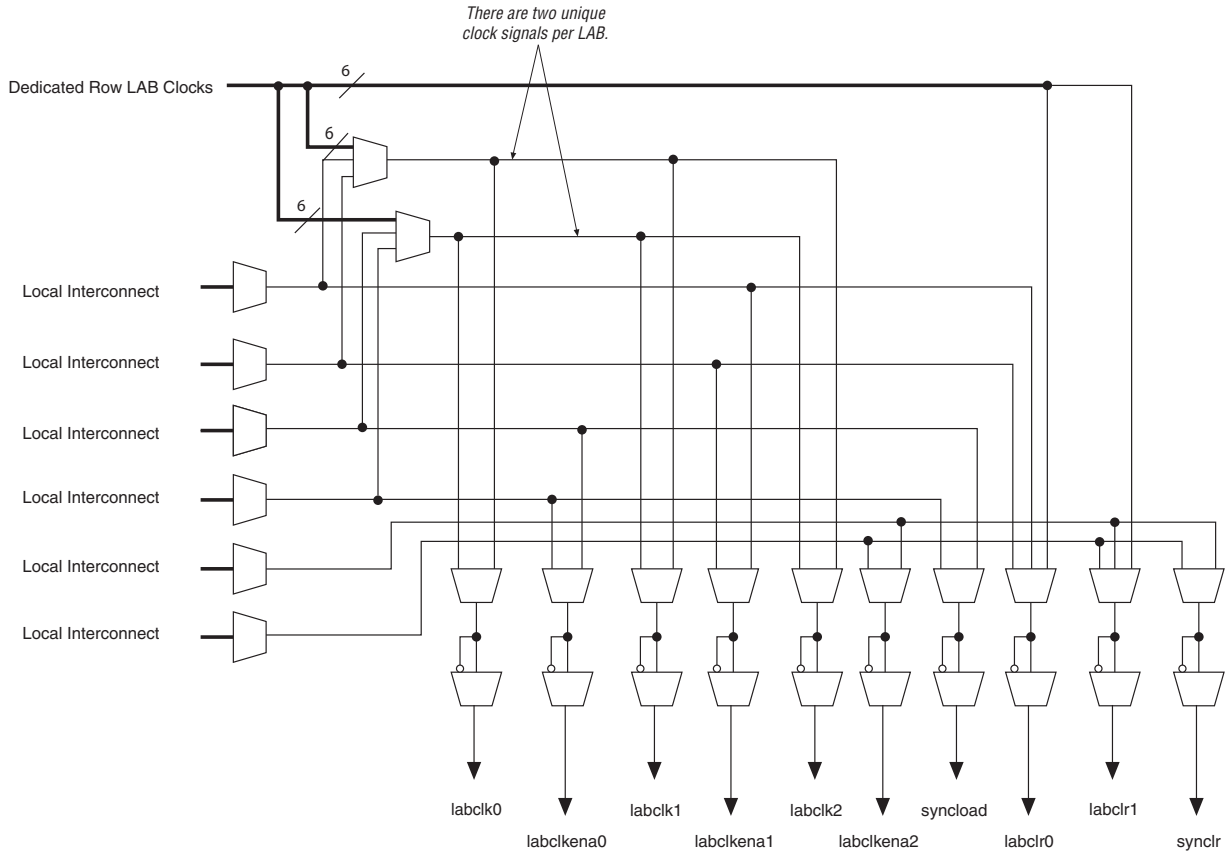
**Figure 2–3. Direct Link Connection**



## 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**

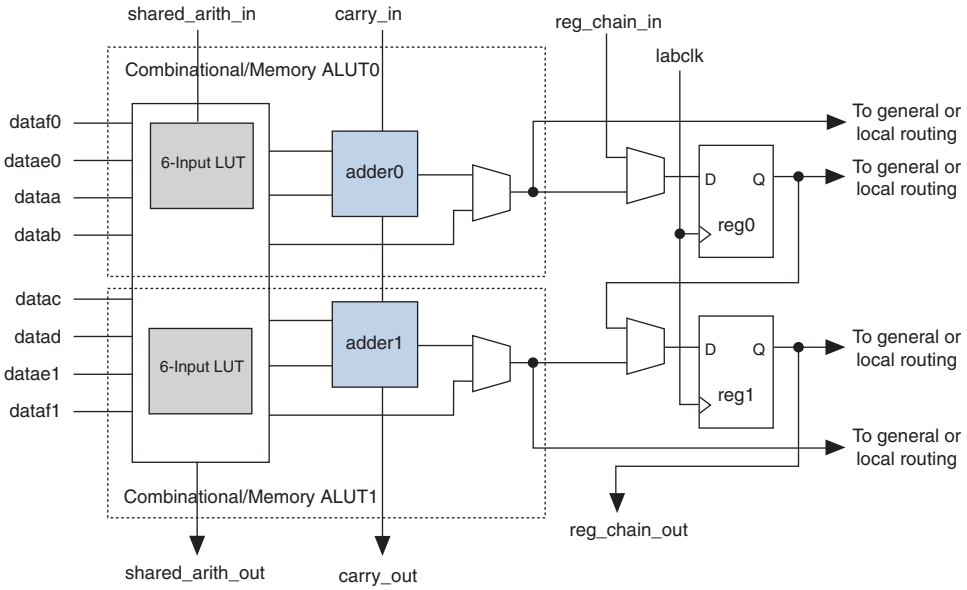
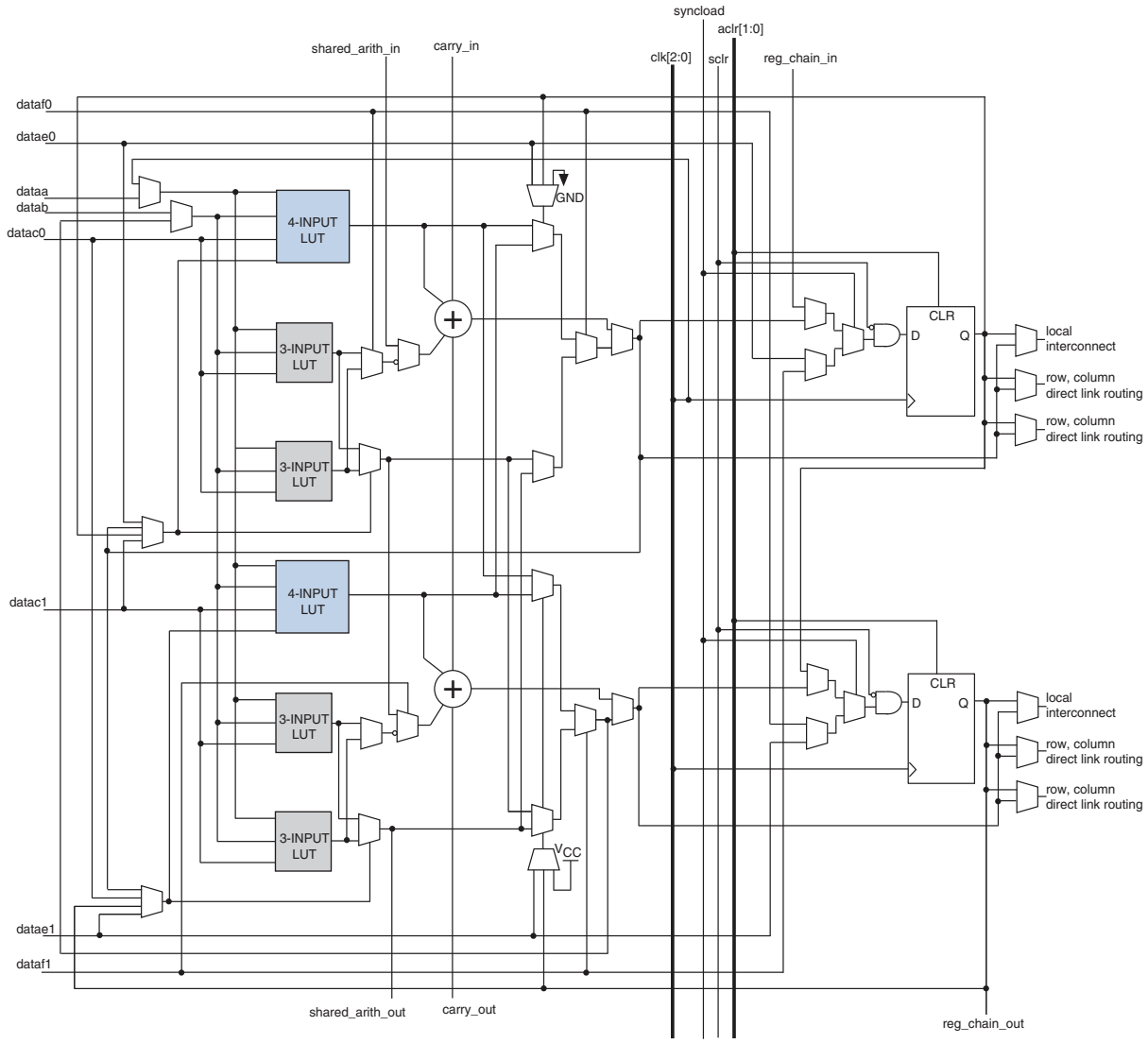


Figure 2–6 shows a detailed view of all the connections in an ALM.

**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

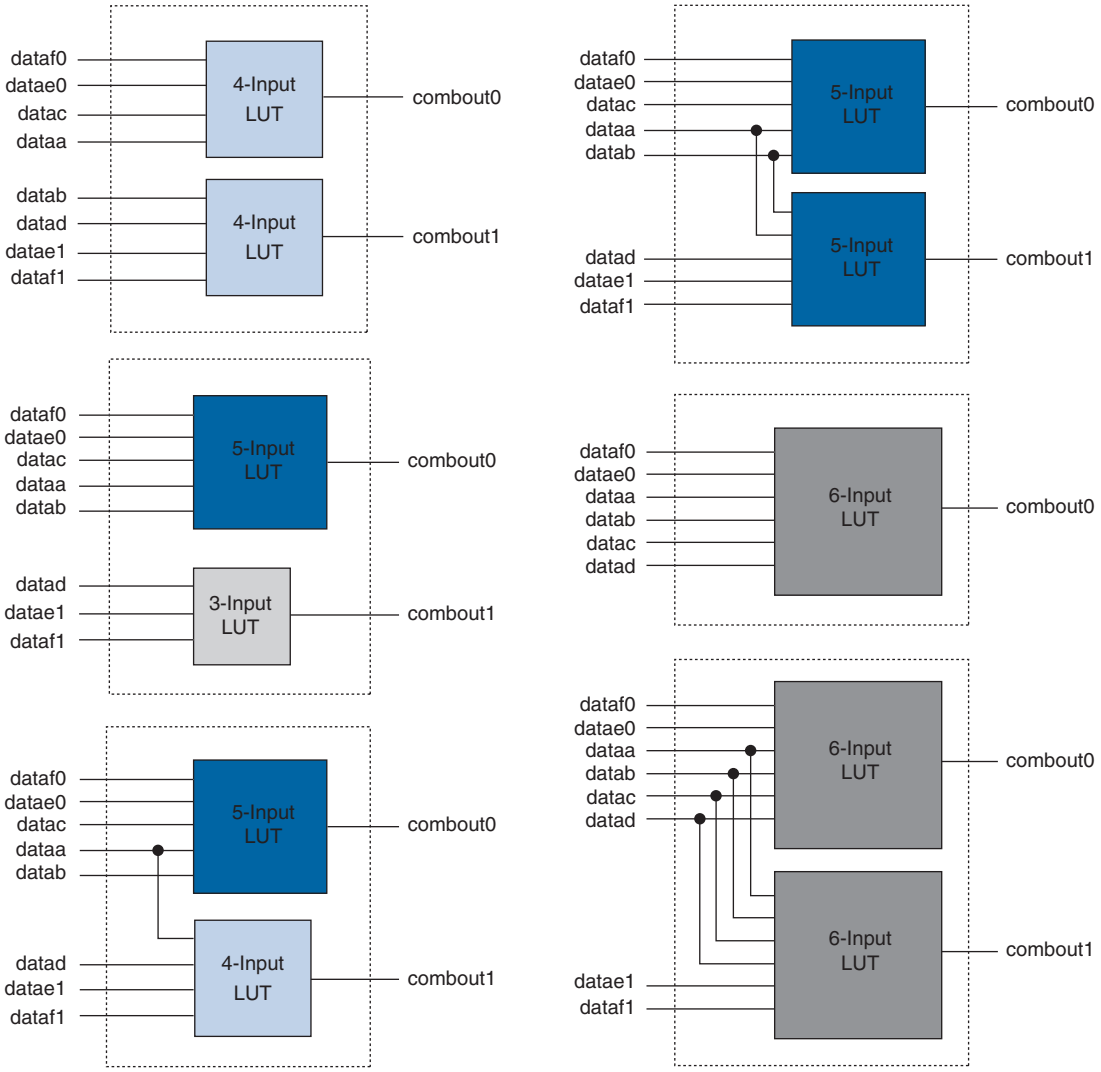
The Quartus II software and other supported third-party synthesis tools, in conjunction with parameterized functions such as the library of parameterized modules (LPM) functions, automatically choose the appropriate mode for common functions such as counters, adders, subtractors, and arithmetic functions.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

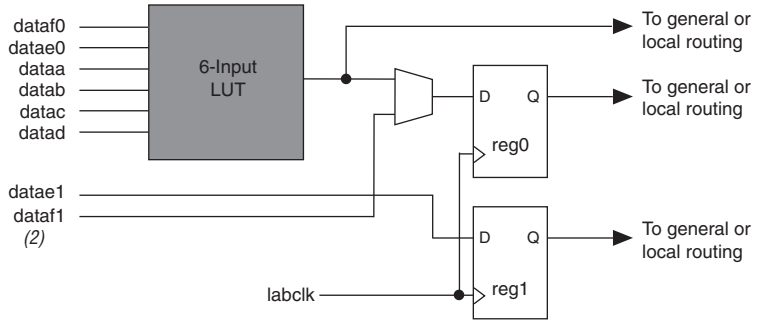
Normal mode provides complete backward-compatibility with 4-input LUT architectures.

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.

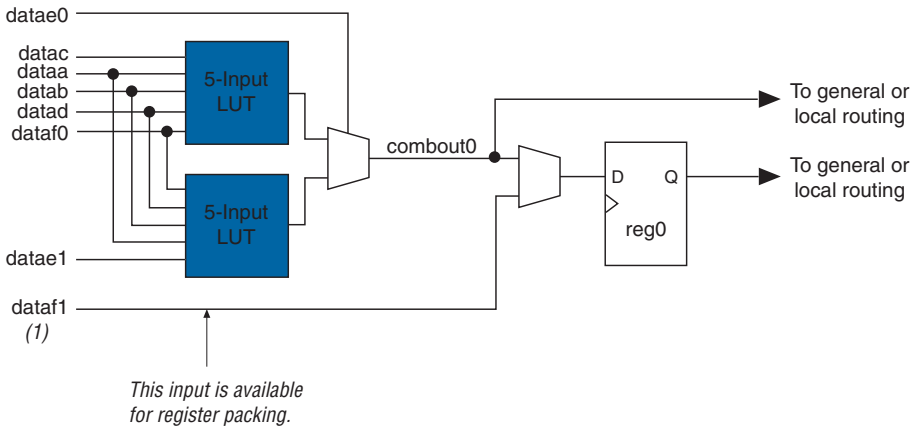
(2) The `dataf1` input is available for register packing only if the 6-input function is unregistered.

## 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**



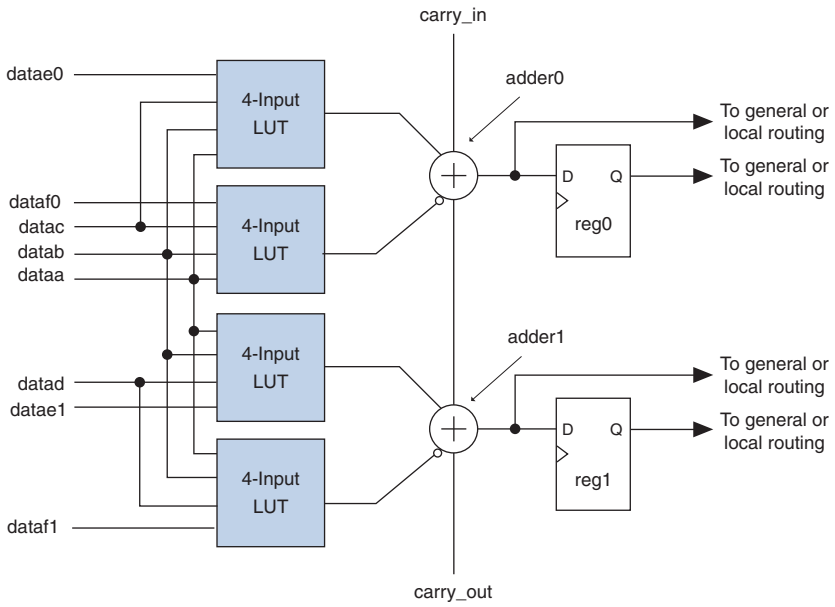

**Note to Figure 2–9:**

(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.

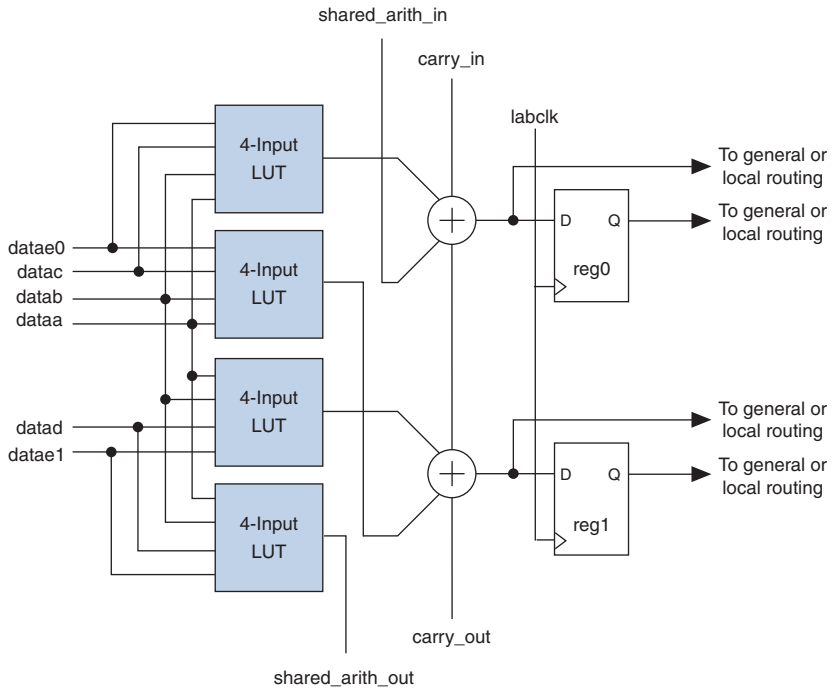
To avoid routing congestion in one small area of the device when a high fan-in arithmetic function is implemented, the LAB can support carry chains that only use either the top half or bottom half of the LAB before connecting to the next LAB. This leaves the other half of the ALMs in the LAB available 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**

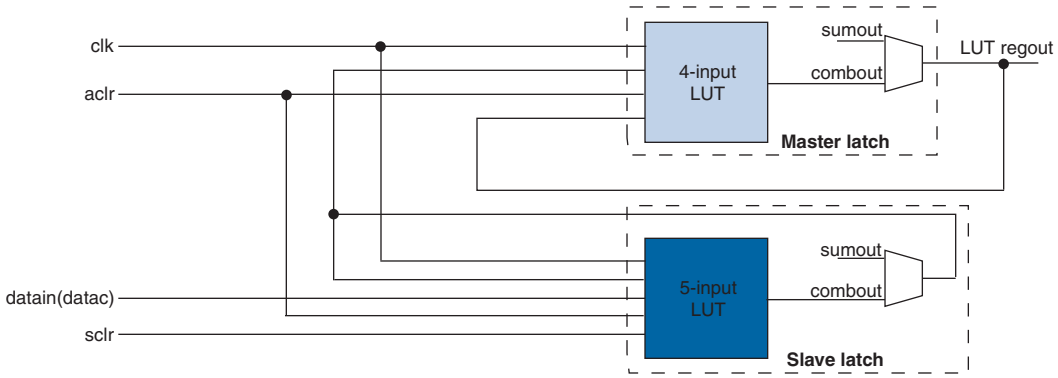
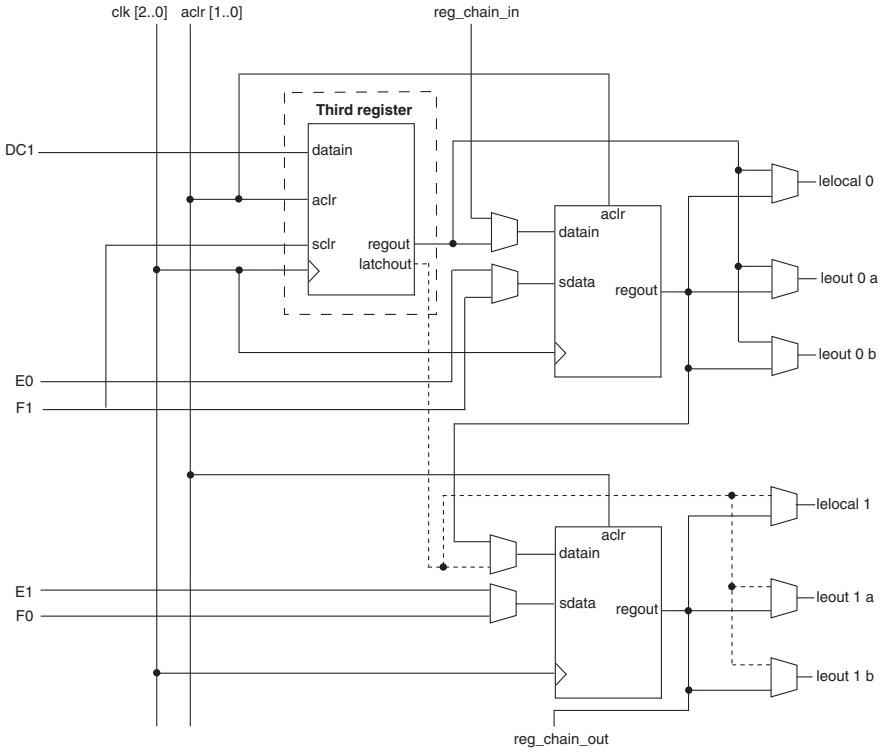



Figure 2–13 shows the ALM in LUT-Register mode.

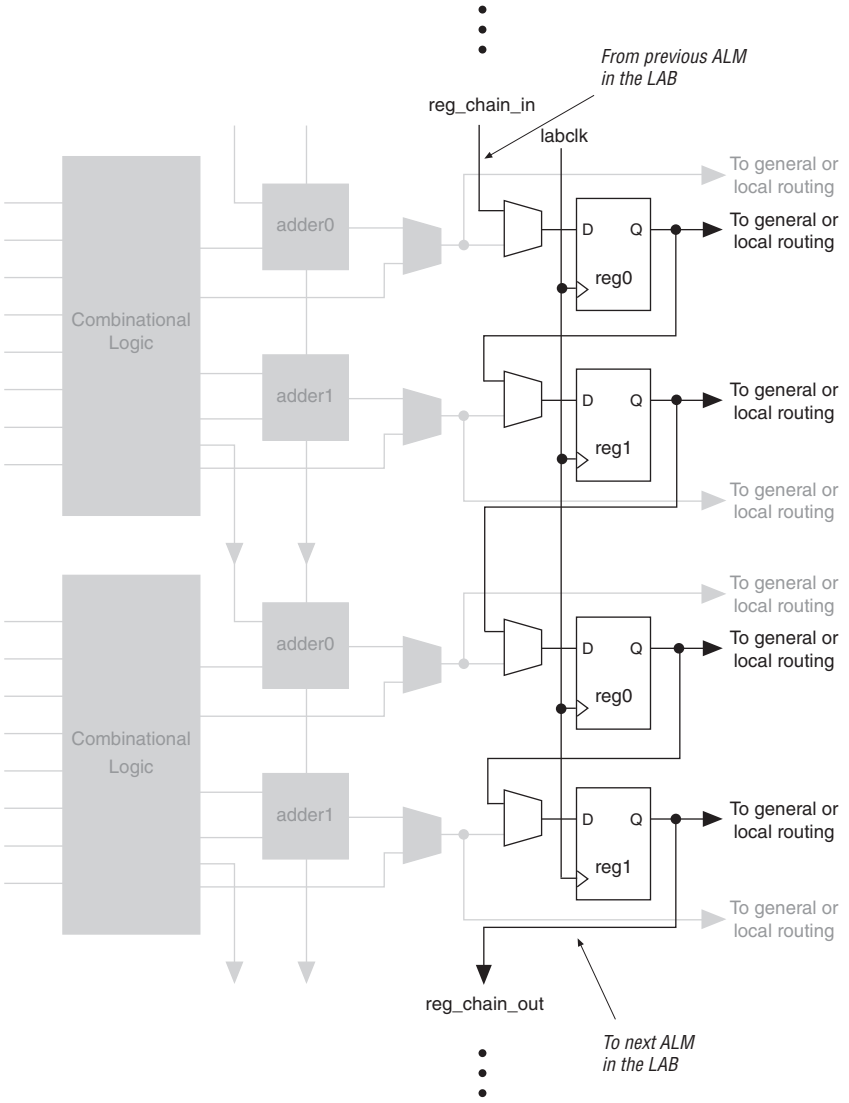
**Figure 2–13. ALM in LUT-Register Mode with 3-Register Capability**



## 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:**

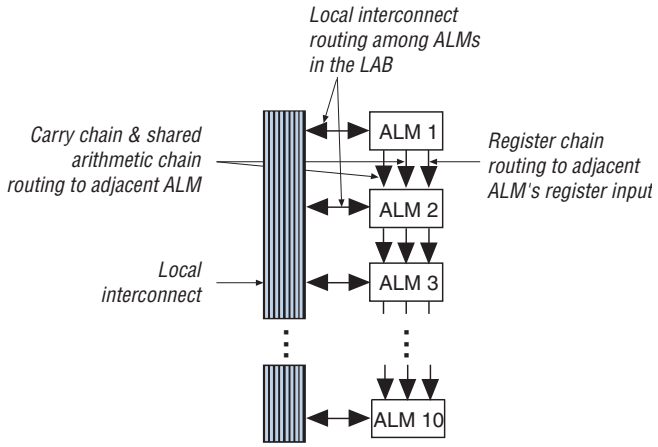
(1) You can use the combinational or adder logic to implement an unrelated, un-registered function.

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:<br>■ Added Arria II GZ device information.<br>■ Updated "Logic Array Blocks", "LAB Interconnects", "LAB Control Signals", "Adaptive Logic Modules", "ALM Operating Modes", "Normal Mode" sections.<br>■ Added Figure 2–7 and Figure 2–8.<br>■ 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

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<br>64 × 9<br>64 × 10<br>32 × 16<br>32 × 18<br>32 × 20 | 64 × 8<br>64 × 9<br>64 × 10<br>32 × 16<br>32 × 18<br>32 × 20 | 8K × 1<br>4K × 2<br>2K × 4<br>1K × 8<br>1K × 9<br>512 × 16<br>512 × 18<br>256 × 32<br>256 × 36 | 8K × 1<br>4K × 2<br>2K × 4<br>1K × 8<br>1K × 9<br>512 × 16<br>512 × 18<br>256 × 32<br>256 × 36 | 16K × 8<br>16K × 9<br>8K × 16<br>8K × 18<br>4K × 32<br>4K × 36<br>2K × 64<br>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.<br>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**

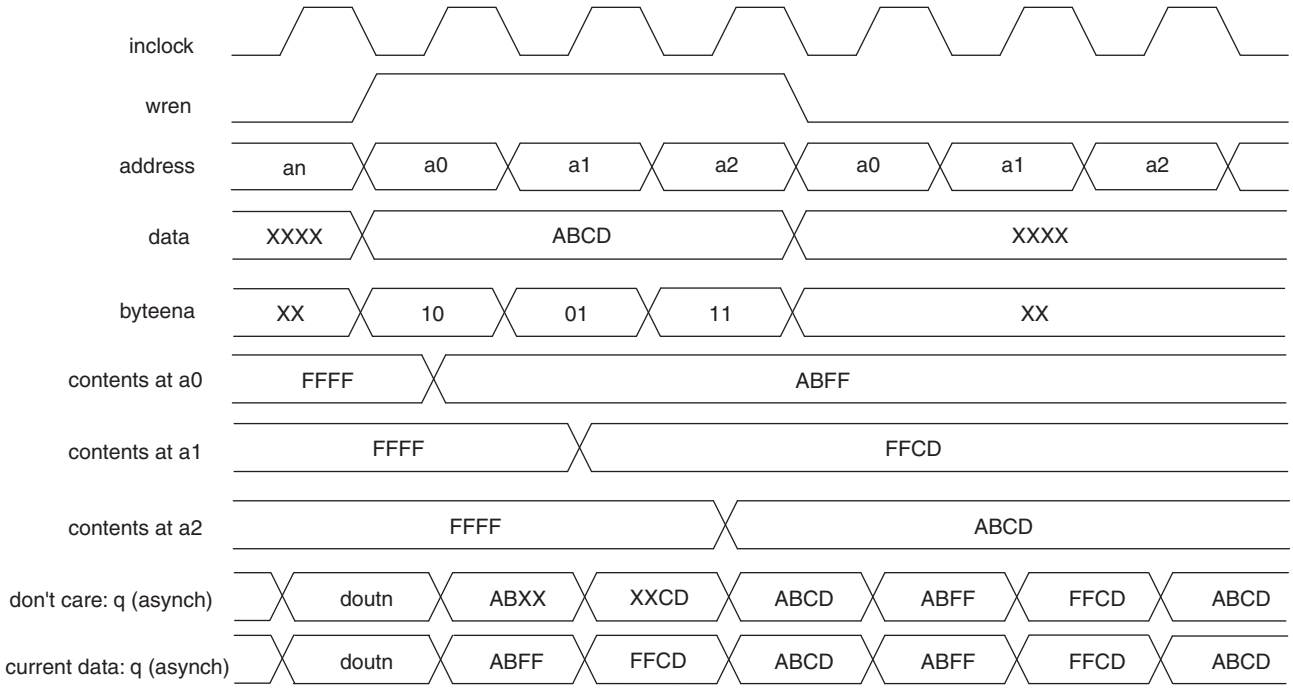
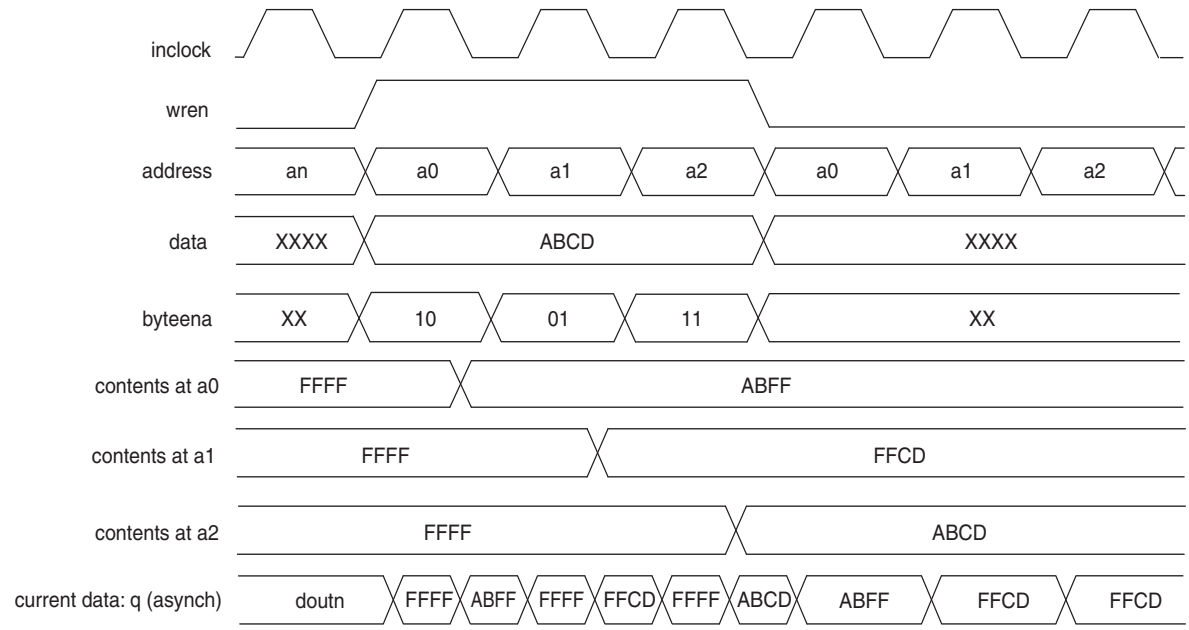


Figure 3–2 shows how the wren and byteena signals control the operations of the MLABs. Falling clock edges triggers the write operation in MLABs.

**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**

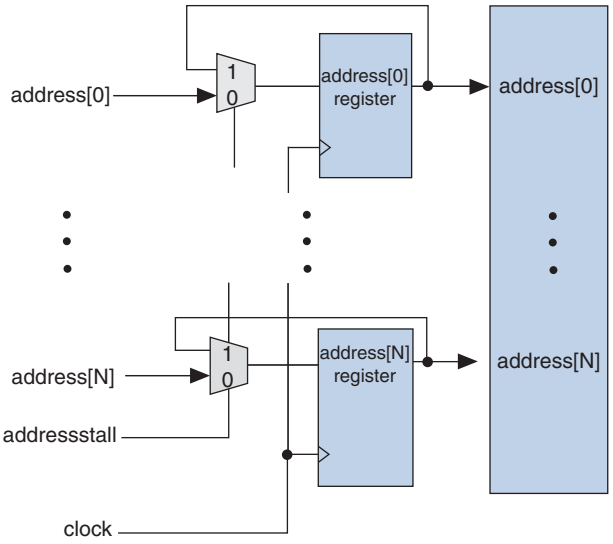



Figure 3–4 shows the address clock enable waveform during the read cycle.

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

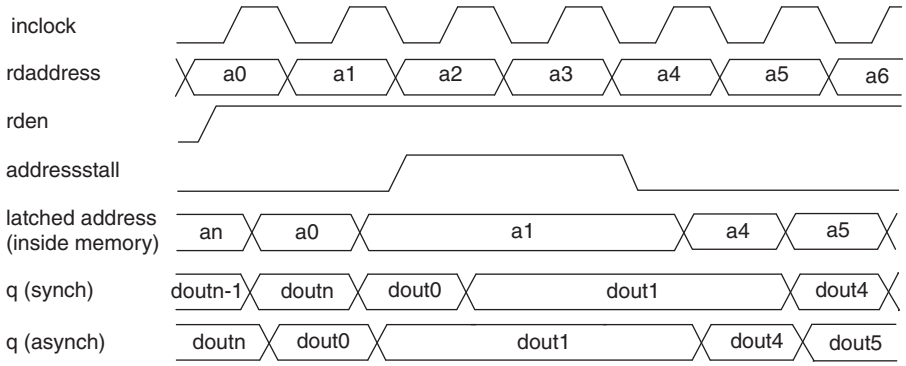


Figure 3–5 shows the address clock enable waveform during write cycle for M9K and M144K blocks.

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

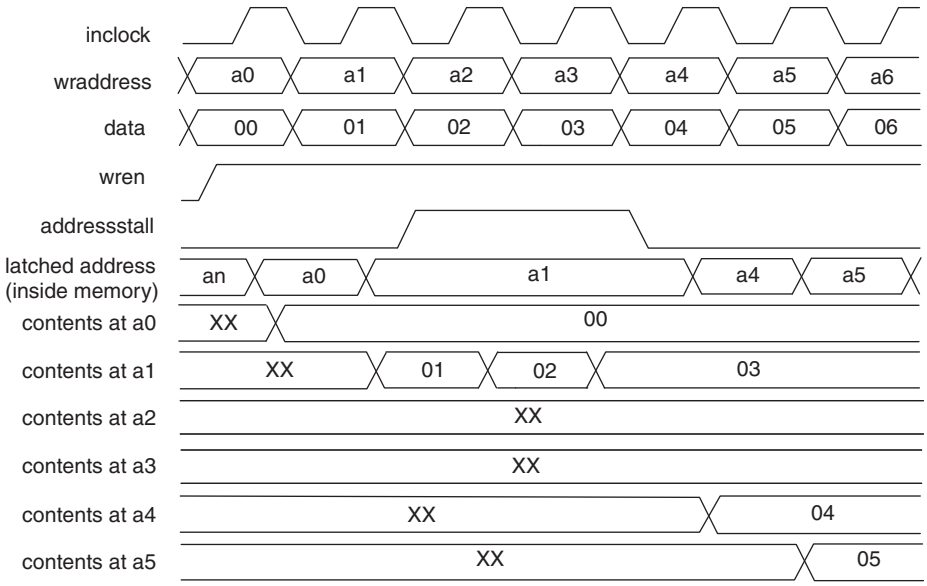
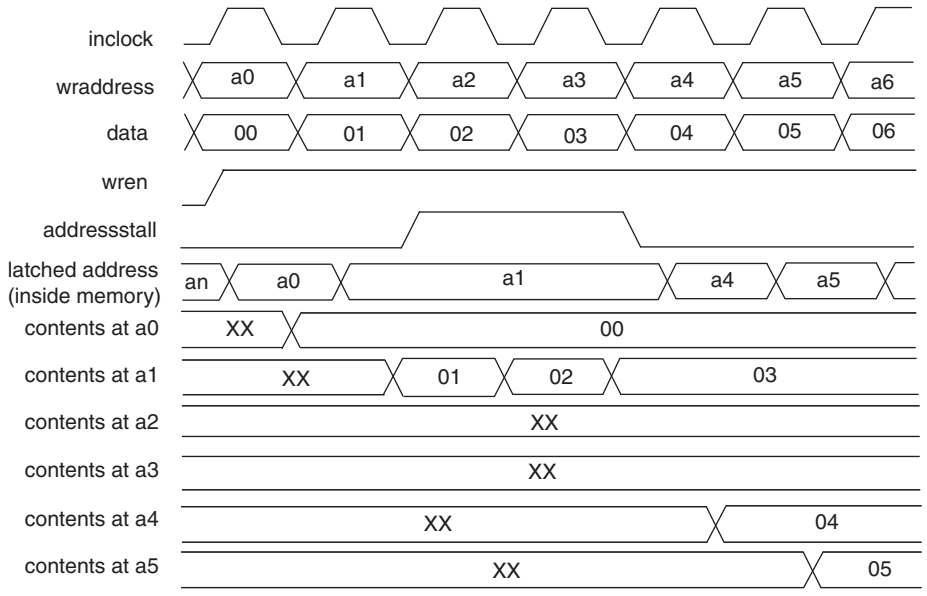



Figure 3–6 shows the address clock enable waveform during the write cycle for MLABs.

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



## Mixed Width Support

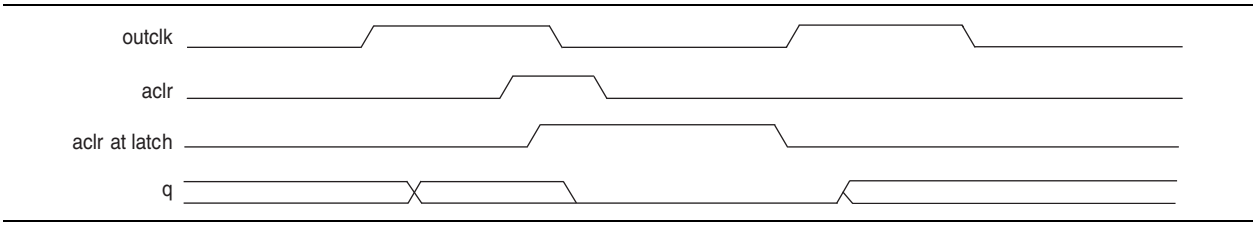
M9K and M144K blocks support mixed data widths inherently. MLABs can support mixed data widths through emulation with the Quartus II software. When using simple dual-port, true dual-port, or FIFO modes, mixed width support allows you to read and write different data widths to a memory block. For more 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.

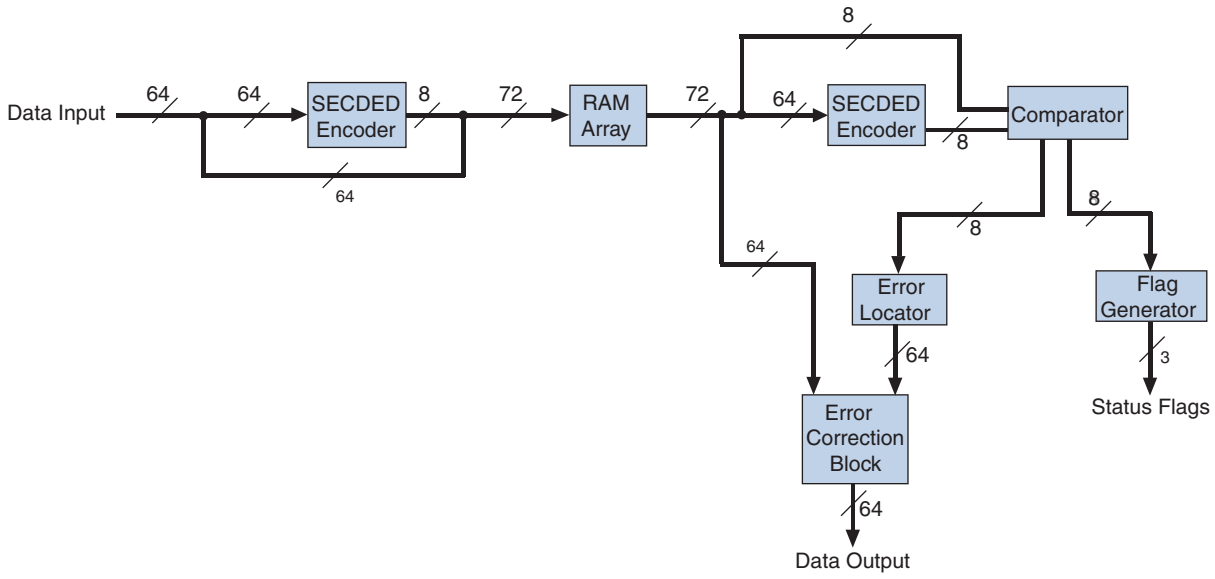
**Table 3–3. Truth Table for ECC Status Flags in Arria II Devices**

| 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

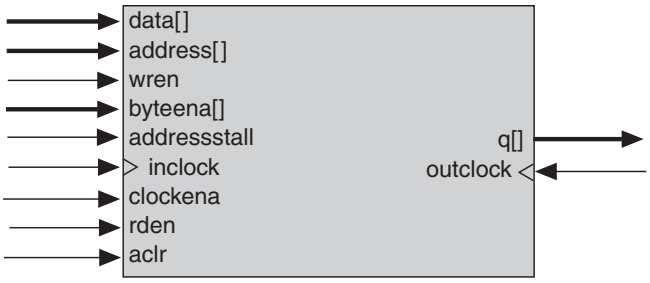
To choose the desired read-during-write behavior, set the read-during-write behavior to either **new data**, **old data**, or **don't care** in the RAM MegaWizard Plug-In Manager in the Quartus II software. For more information about this behavior, refer to "Read-During-Write Behavior" on page 3–21.

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<br>64 × 9<br>64 × 10<br>32 × 16<br>32 × 18<br>32 × 20 | 8K × 1<br>4K × 2<br>2K × 4<br>1K × 8<br>1K × 9<br>512 × 16<br>512 × 18<br>256 × 32<br>256 × 36 | 16K × 8<br>16K × 9<br>8K × 16<br>8K × 18<br>4K × 32<br>4K × 36<br>2K × 64<br>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)**

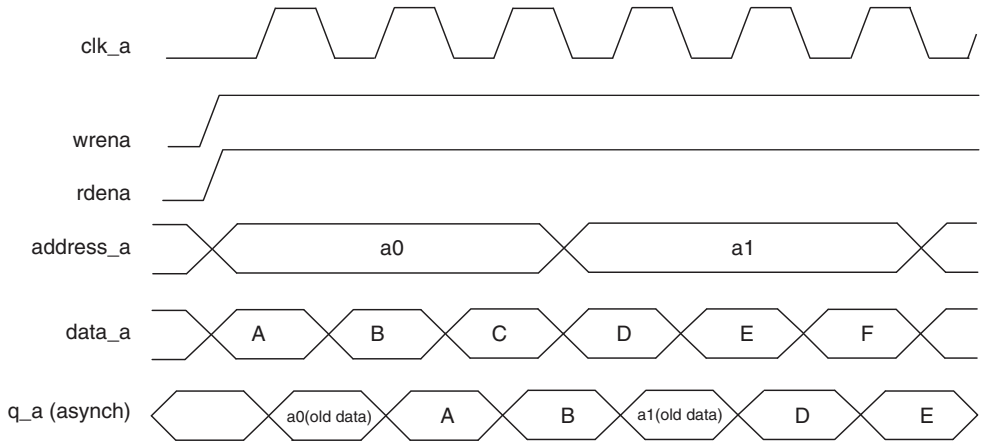
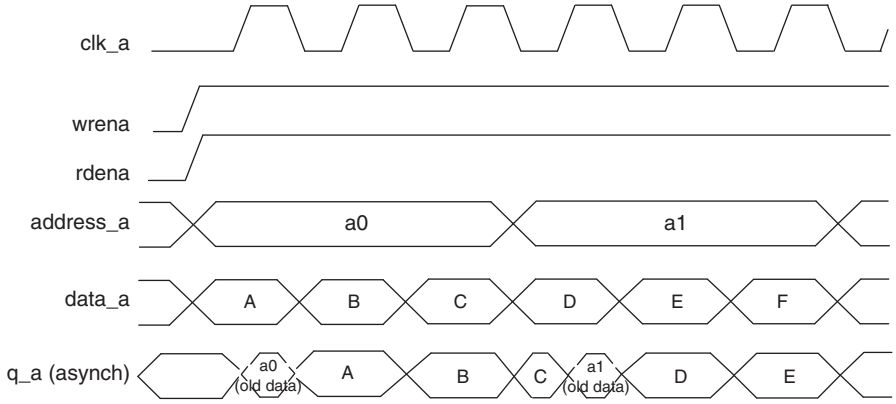


Figure 3–11 shows the timing waveforms for read and write operations in single-port mode with unregistered outputs for the MLAB. The rising clock edges trigger the read operation whereas the falling clock edges triggers the write operation.

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

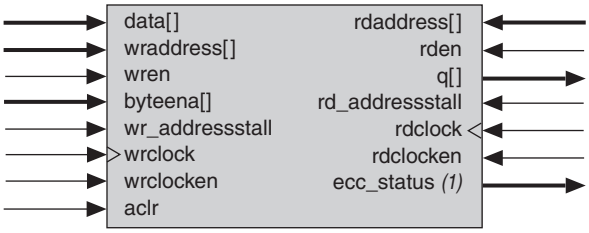



## Simple Dual-Port Mode

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**

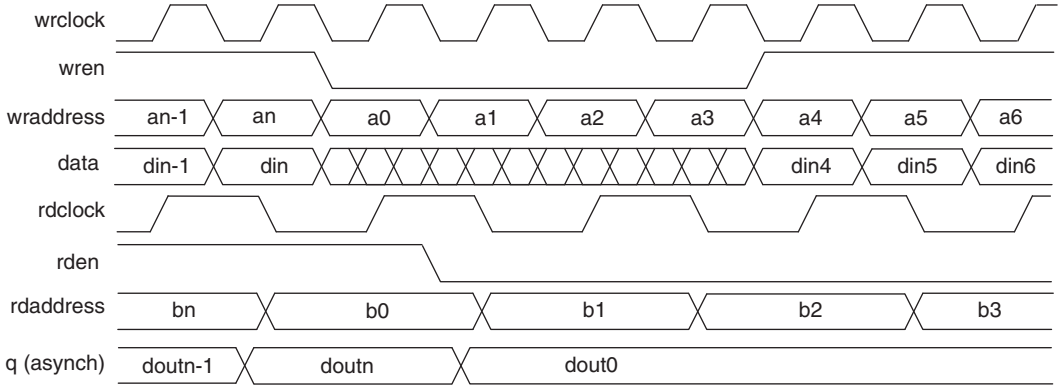



Figure 3–14 shows the timing waveforms for read and write operations in simple dual-port mode with unregistered outputs in the MLAB. The write operation is triggered by the falling clock edges.

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

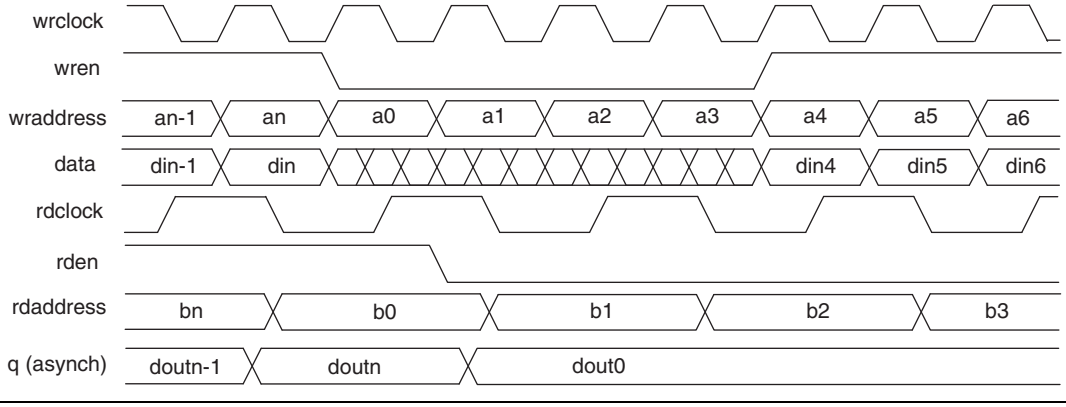
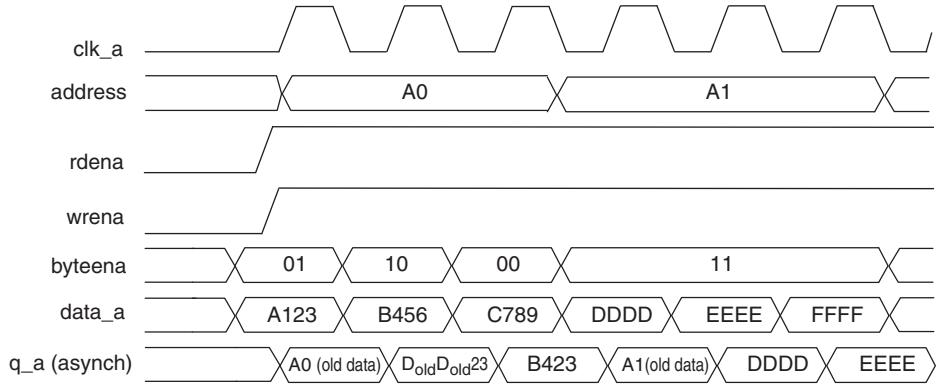


Figure 3–15 shows timing waveforms for read and write operations in mixed-port mode with unregistered outputs.

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

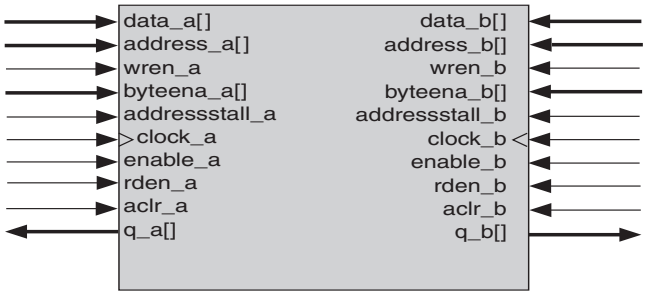



## True Dual-Port Mode

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)**

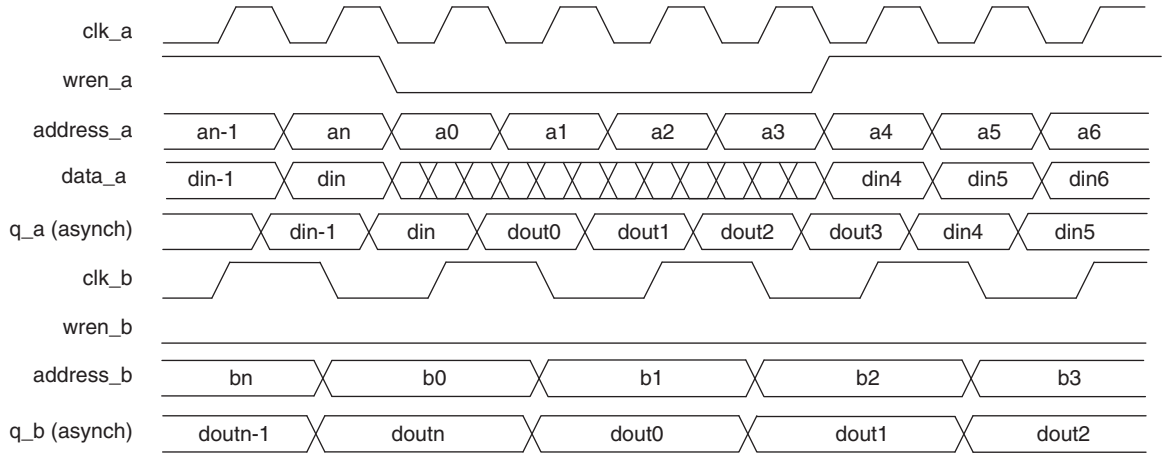
| Read Port | Write Port | | | | | |
|---|---|---|---|---|---|---|
| | 16K × 8 | 8K × 16 | 4K × 32 | 16K × 9 | 8K × 18 | 4K × 36 |
| 16K × 8 | ✓ | ✓ | ✓ | — | — | — |
| 8K × 16 | ✓ | ✓ | ✓ | — | — | — |
| 4K × 32 | ✓ | ✓ | ✓ | — | — | — |
| 16K × 9 | — | — | — | ✓ | ✓ | ✓ |
| 8K × 18 | — | — | — | ✓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



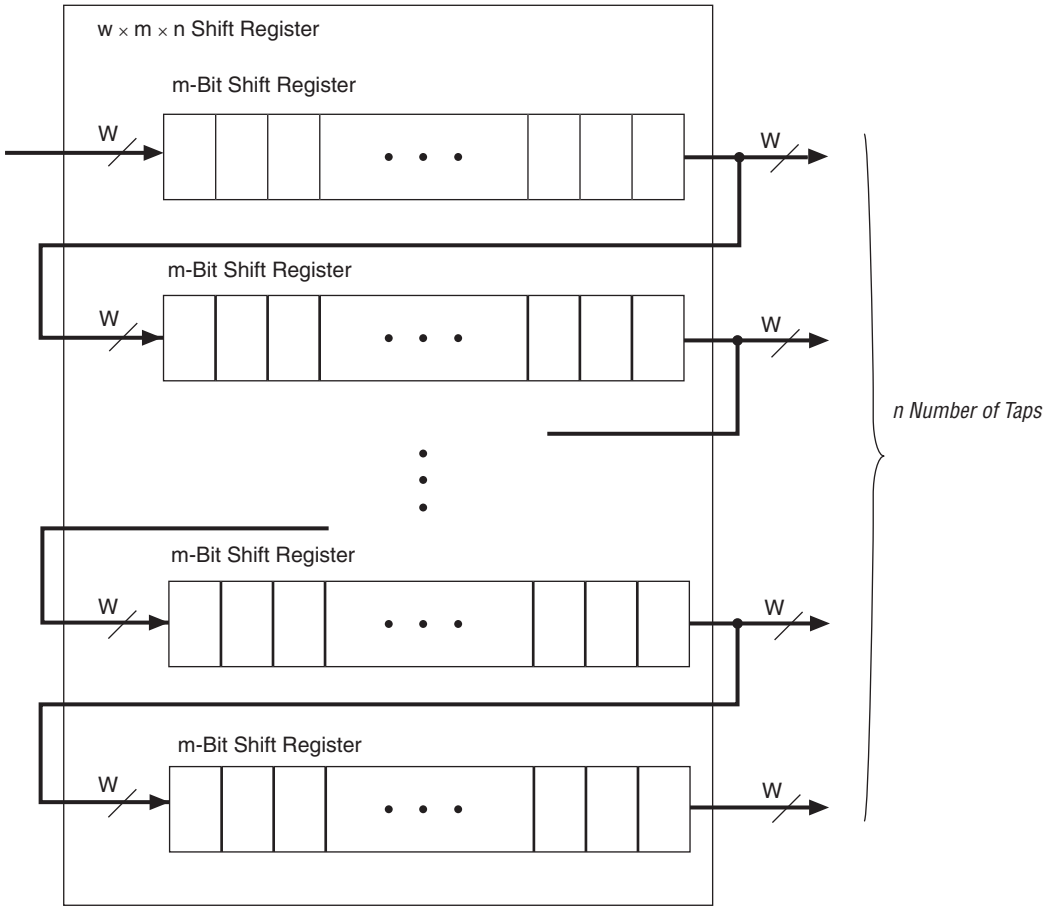

## Shift-Register Mode

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.

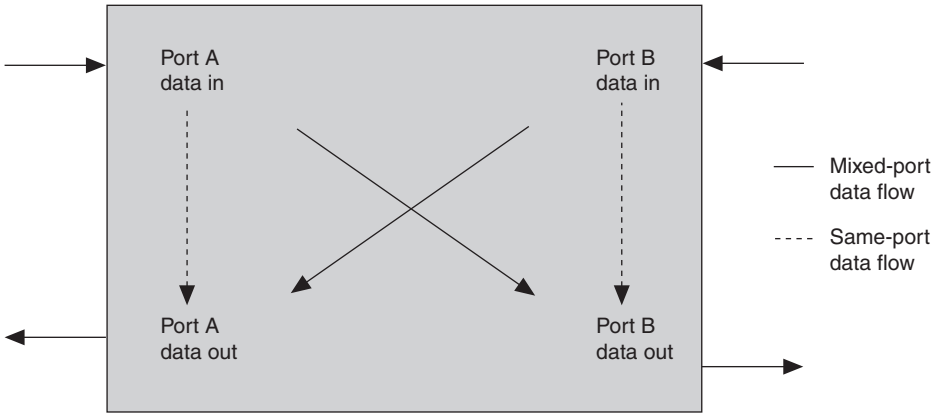
### Conflict Resolution

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**

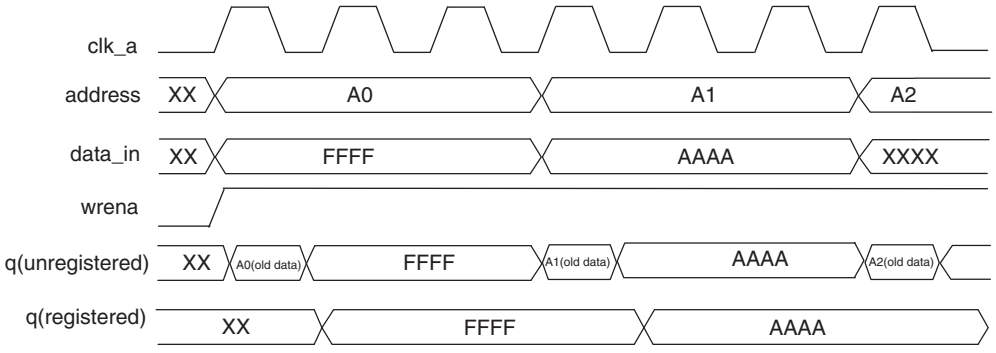


Figure 3–21 shows sample functional waveforms of same-port read-during-write behavior in new data mode.

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

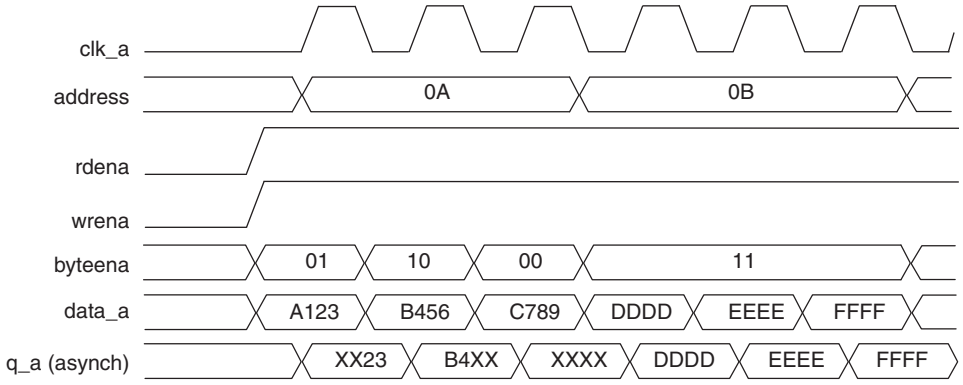
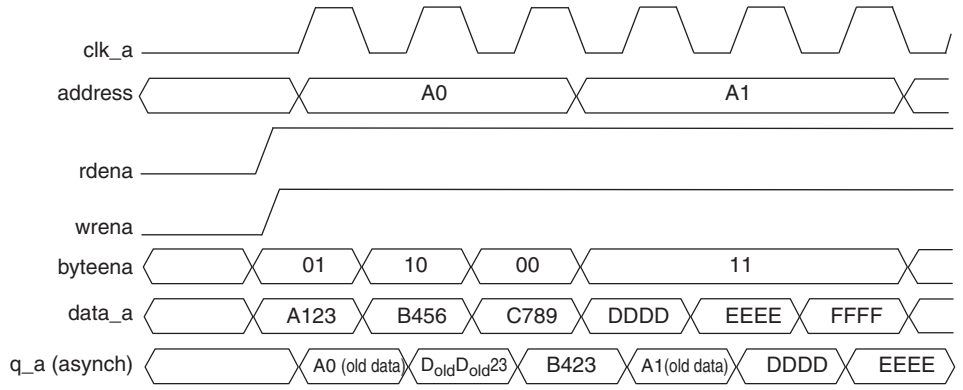



Figure 3–22 shows sample functional waveforms of same-port read-during-write behavior in old data mode.

**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**

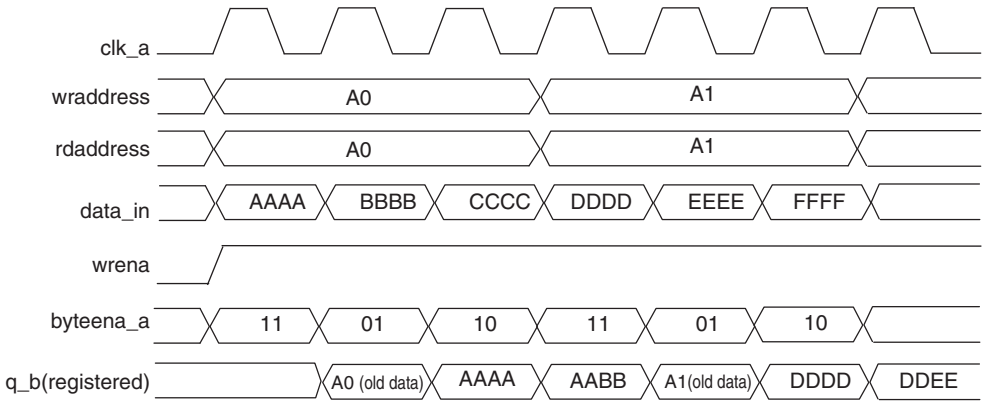


Figure 3–24 shows a sample functional waveform of mixed-port read-during-write behavior for new data mode in MLABs.

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

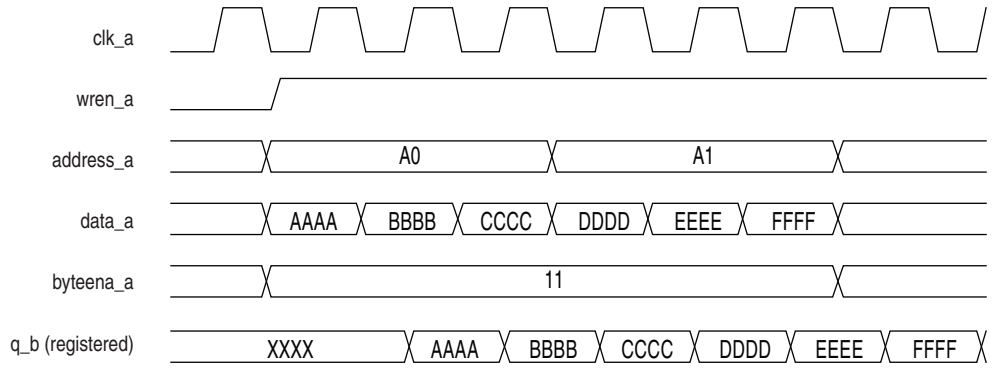



Figure 3–25 shows a sample functional waveform of mixed-port read-during-write behavior for don't care mode in MLABs.

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

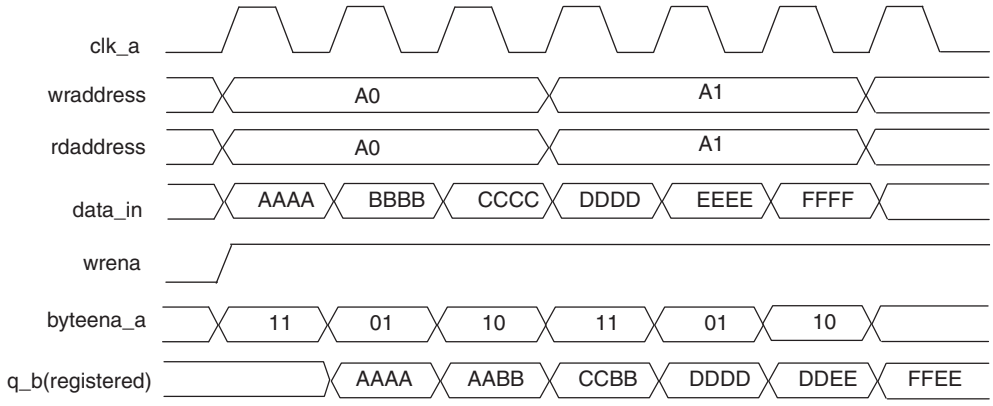


Figure 3–26 shows a sample functional waveform of mixed-port read-during-write behavior in old data mode.

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

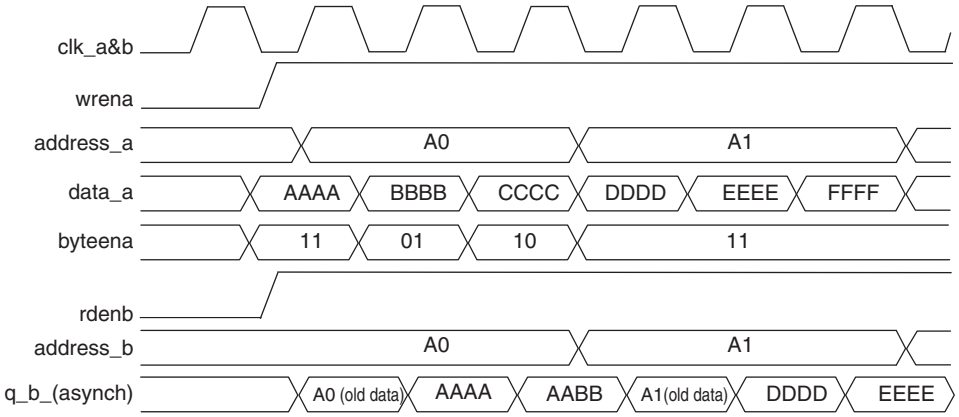
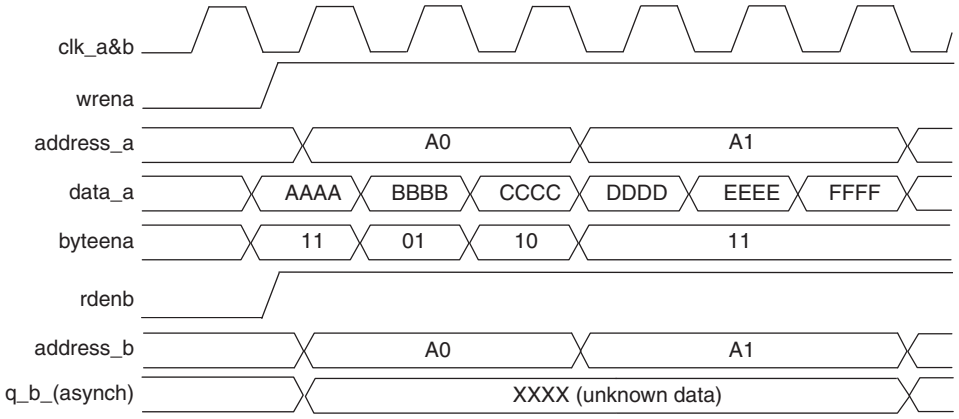



Figure 3–27 shows a sample functional waveform of mixed-port read-during-write behavior for don't care mode in M9K and M144K blocks.

**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.<br>■ Updated "Byte Enable Support" and "Mixed-Port Read-During-Write Mode" sections. |
| June 2011 | 3.1 | ■ Updated Table 3–1.<br>■ Updated the "Mixed-Port Read-During-Write Mode" section.<br>■ Minor text edits. |
| December 2010 | 3.0 | ■ Updated for the Quartus II software version 10.1 release.<br>■ Added Arria II GZ devices information.<br>■ Updated Table 3–1 and Table 3–2.<br>■ Updated Figure 3–10, Figure 3–12, and Figure 3–16.<br>■ Added Table 3–6 and Table 3–8.<br>■ Added Figure 3–10, Figure 3–15, Figure 3–21, Figure 3–23, and Figure 3–24.<br>■ Added "Error Correction Code Support" section.<br>■ Minor text edit. |
| November 2009 | 2.0 | Updated for Arria II GX v9.1 release:<br>■ Updated Table 3–2<br>■ Updated Figure 3–16<br>■ Minor text edit |
| June 2009 | 1.1 | ■ Updated Table 3–1<br>■ Updated "Byte Enable Support", "Simple Dual-Port Mode", and "Read and Write Clock Mode" sections<br>■ 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<br>■ 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

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)***

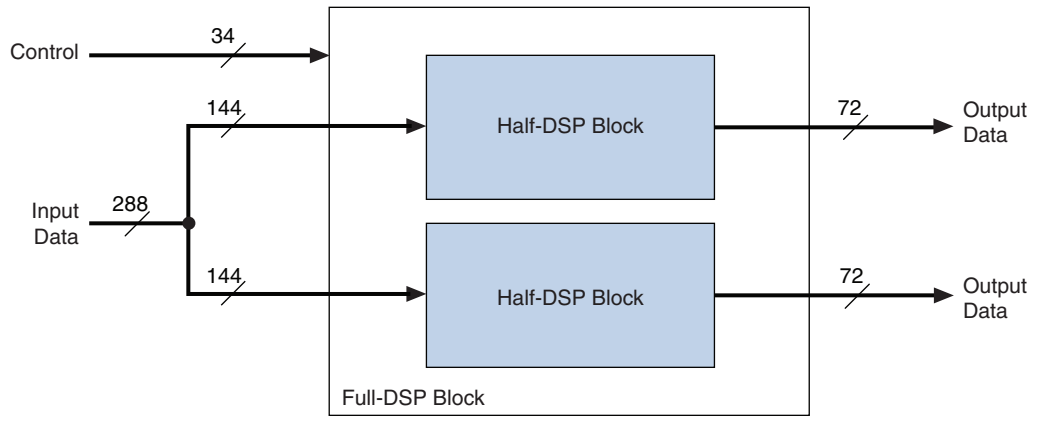
| Family | Device | DSP Blocks | Independent Input and Output Multiplication Operators | | | | | High Precision Multiplier Adder Mode | Four Multiplier Adder Mode |
|---|---|---|---|---|---|---|---|---|---|
| | | | 9 × 9 Multipliers | 12 × 12 Multipliers | 18 × 18 Multipliers | 18 × 18 Complex | 36 × 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**



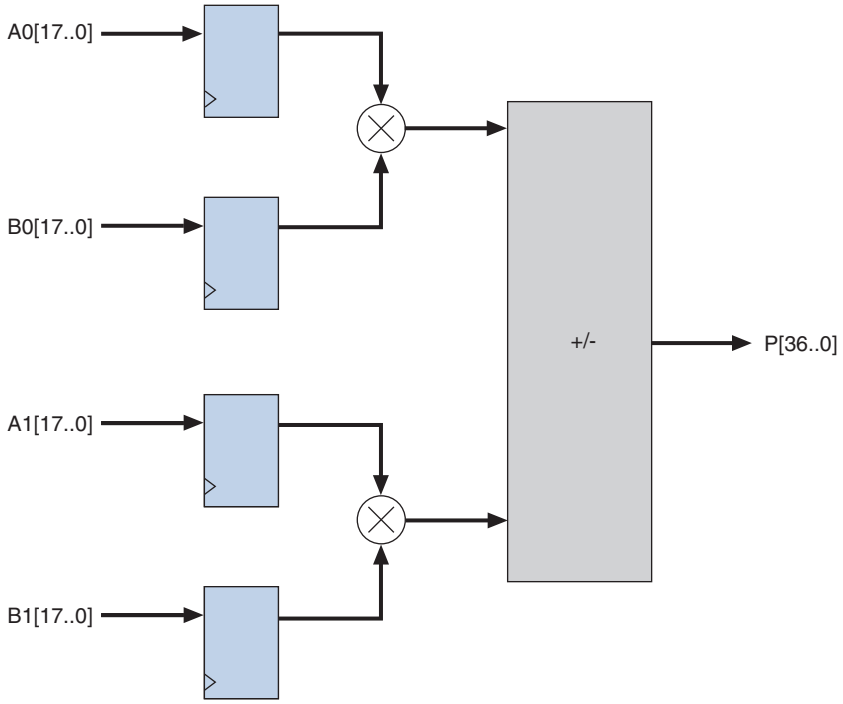
# Simplified DSP Operation

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)**

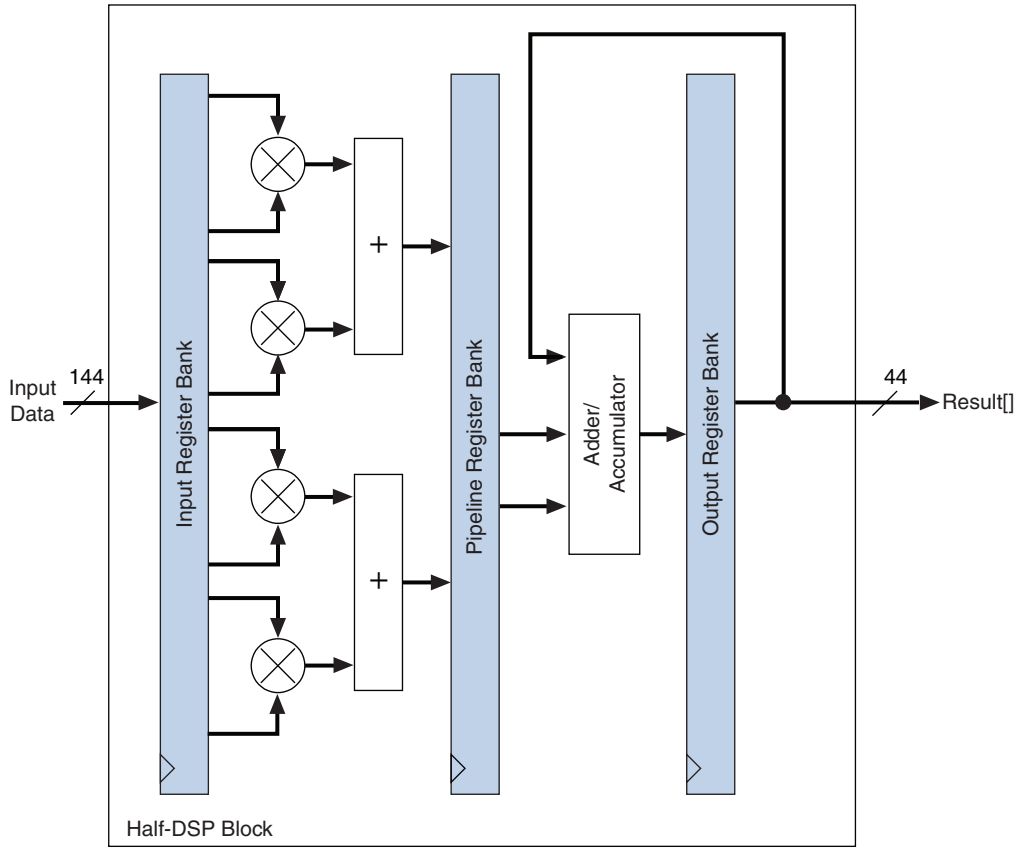
$$W_n[43..0] = W_{n-1}[43..0] \pm Z_n[37..0]$$

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**

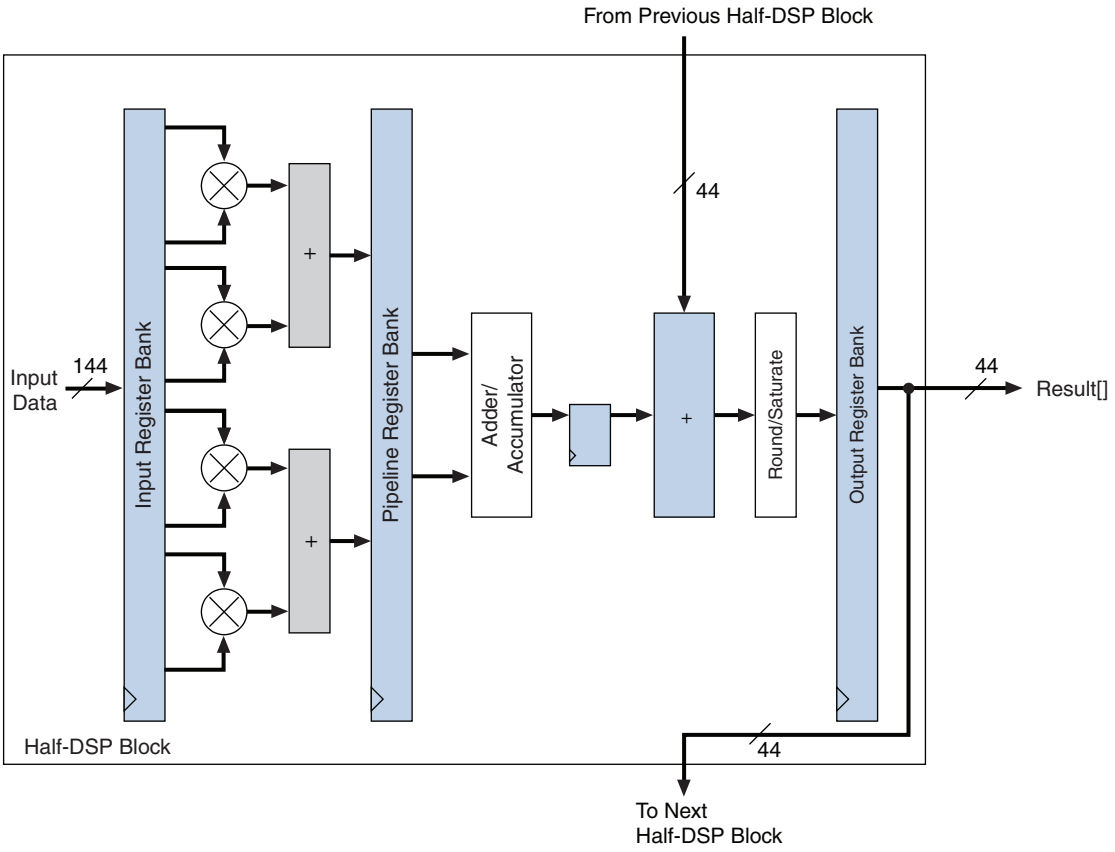



Figure 4–4 shows the optional rounding and saturation unit. This unit provides a set of commonly found arithmetic rounding and saturation functions in signal processing.

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.

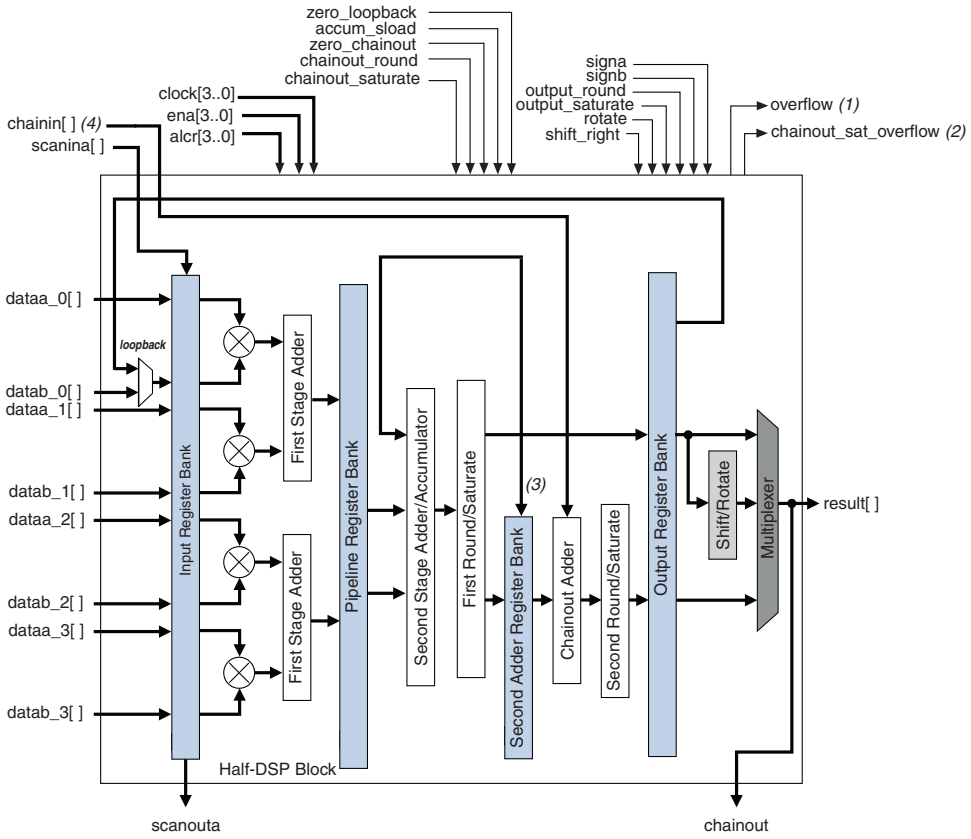
# DSP Block Resource Descriptions

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**



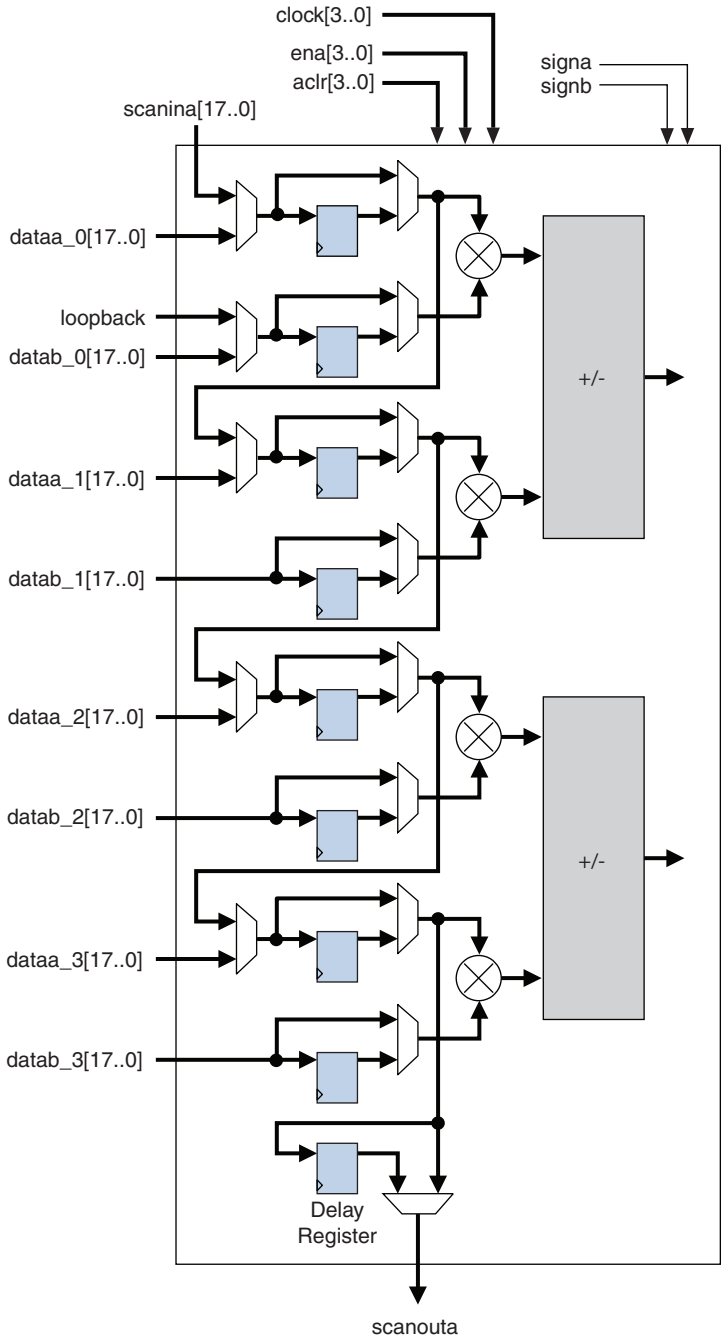

**Notes to Figure 4–5:**

(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

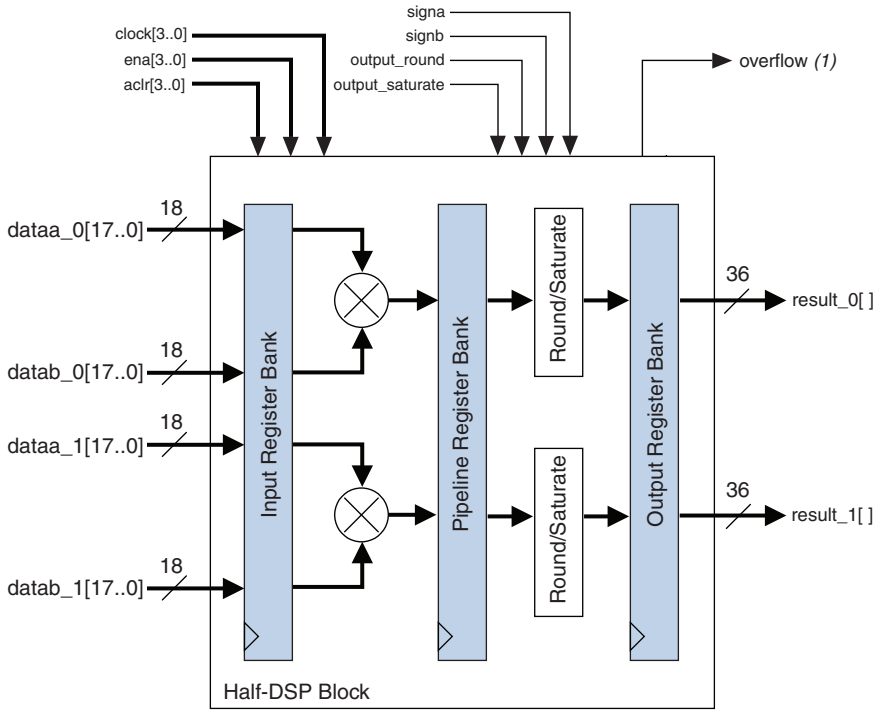
In the independent input and output multiplier mode, the DSP block performs individual multiplication operations for general-purpose multipliers.

### 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**

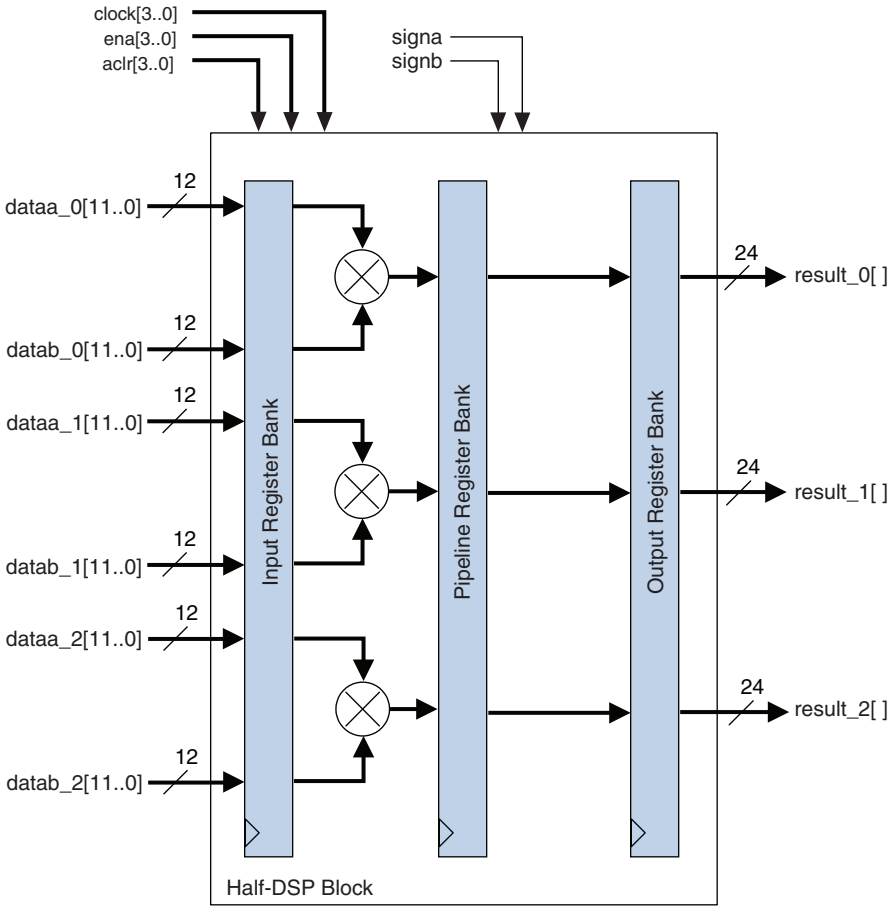
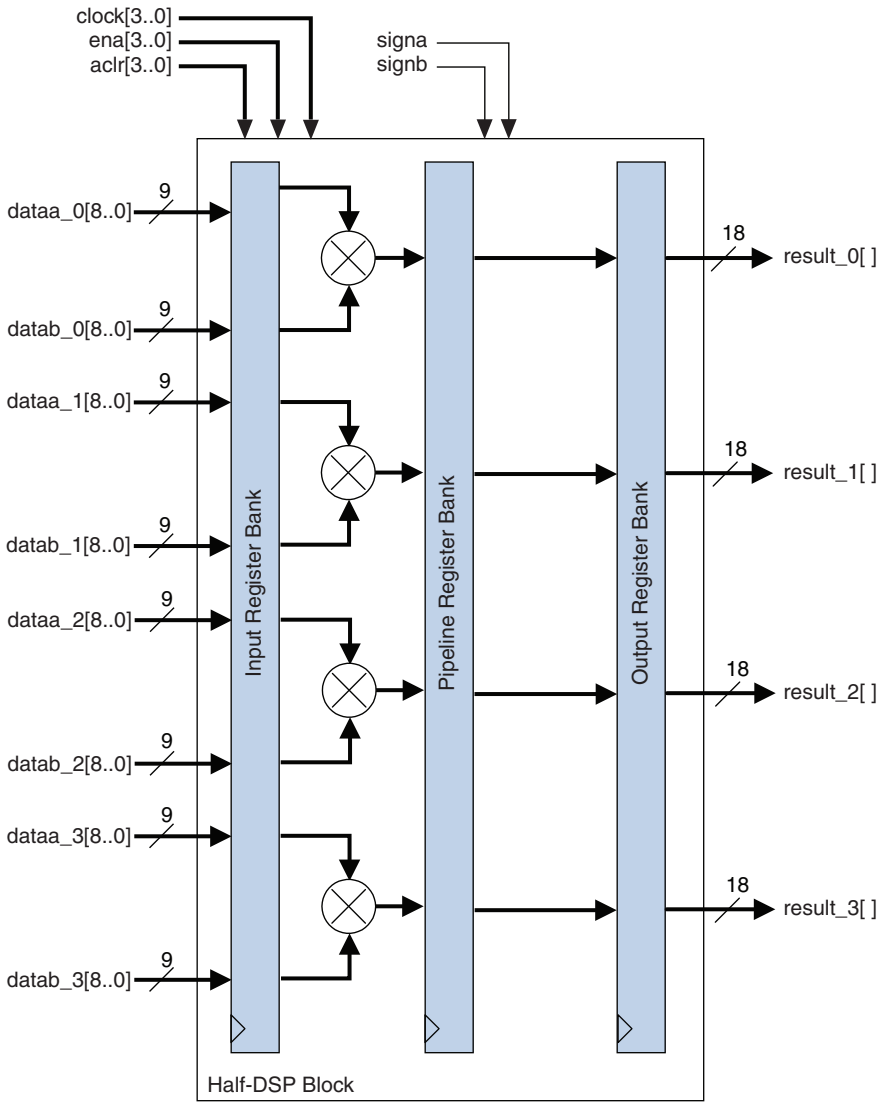


**Figure 4–9. 9-Bit Independent Multiplier Mode Shown for Half-DSP Block**




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.

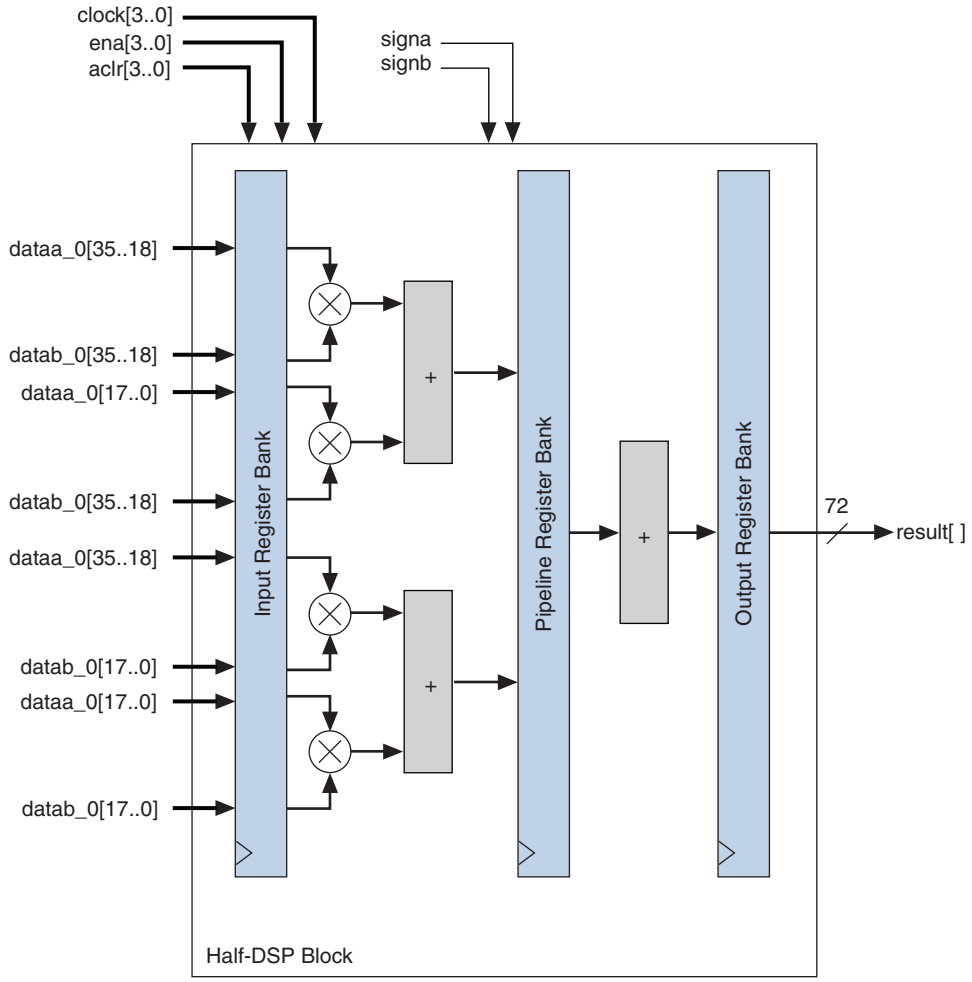
The rounding and saturation logic unit is supported for 18-bit independent multiplier mode only.

## 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**

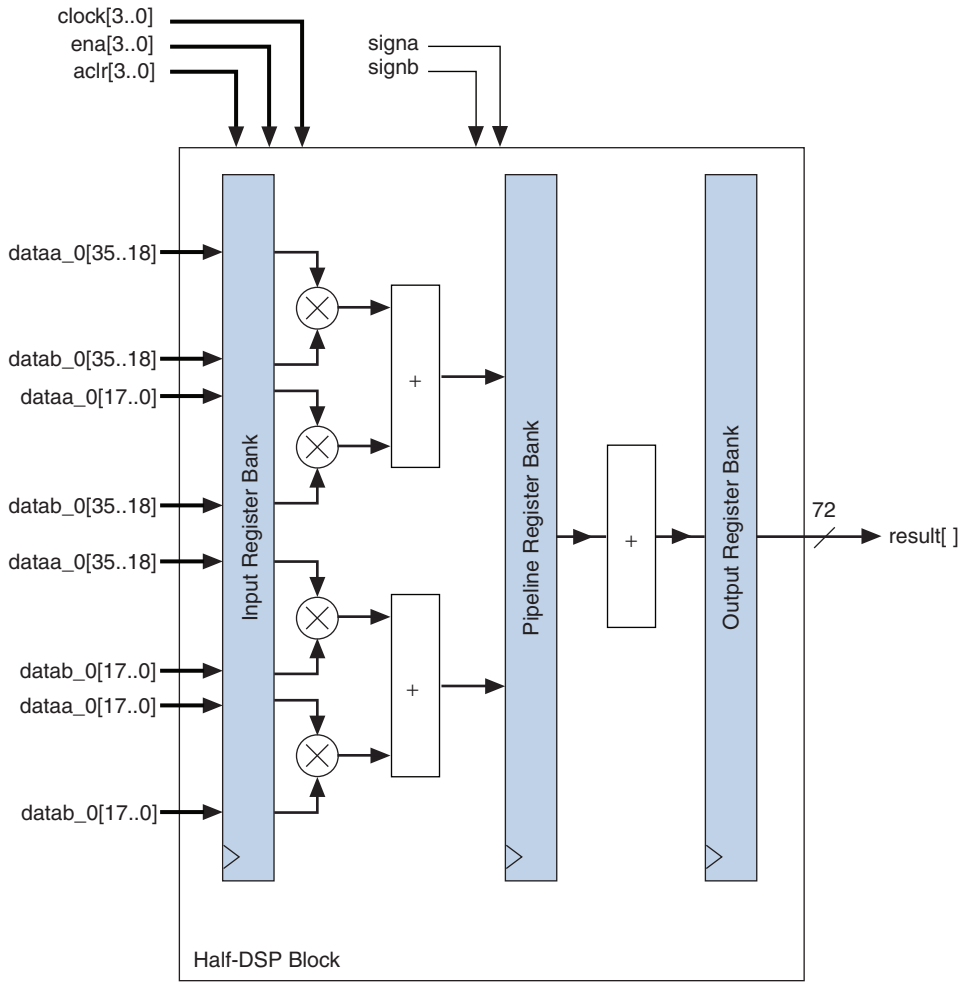
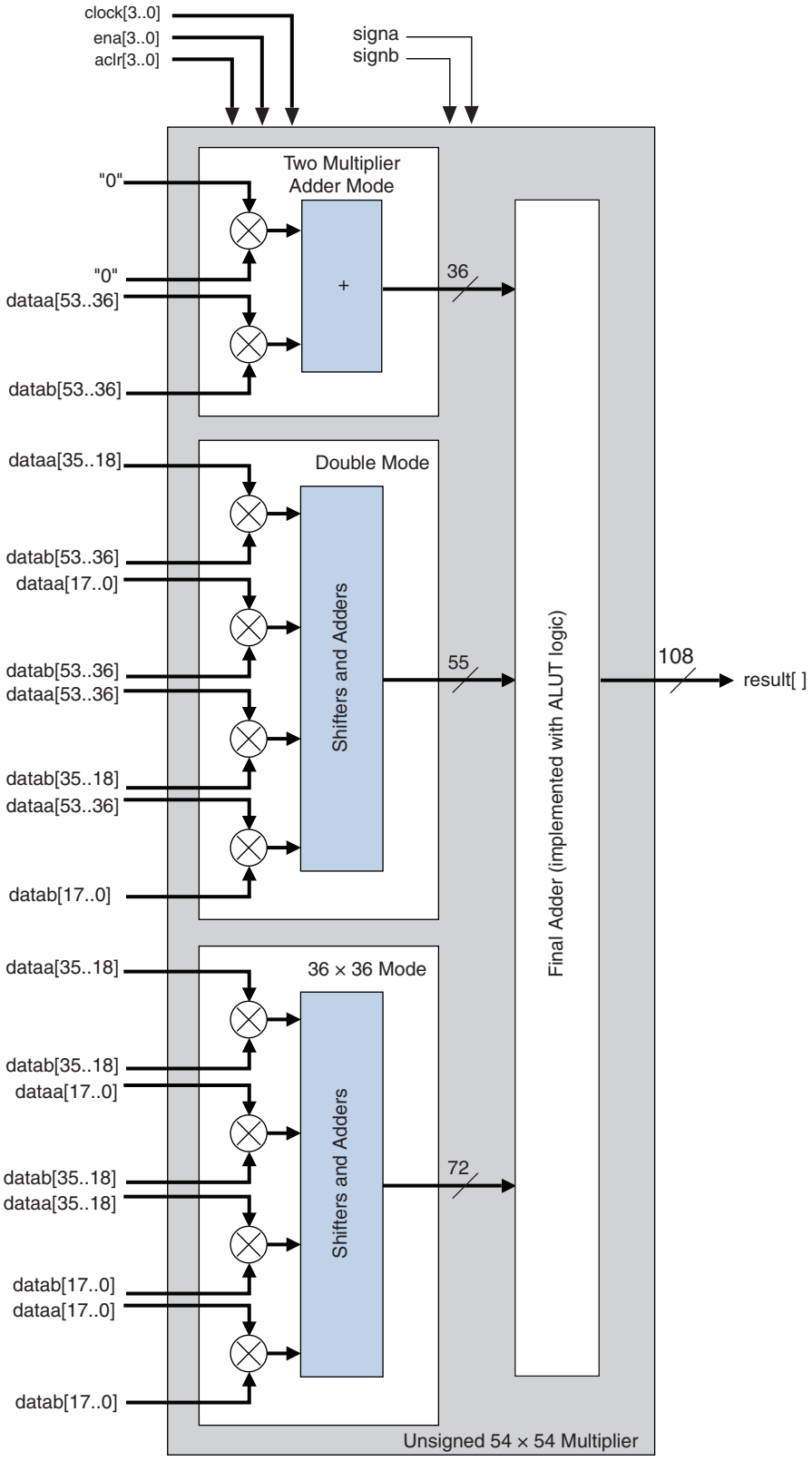


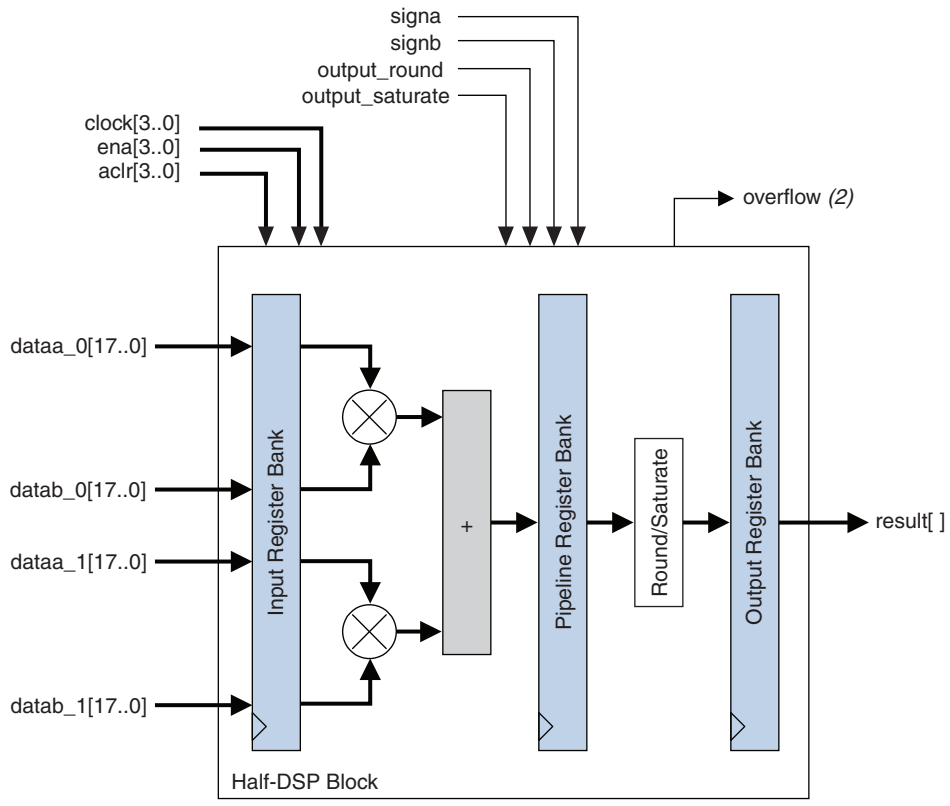
**Figure 4–12. Unsigned 54 × 54-Bit Multiplier**



## 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:**

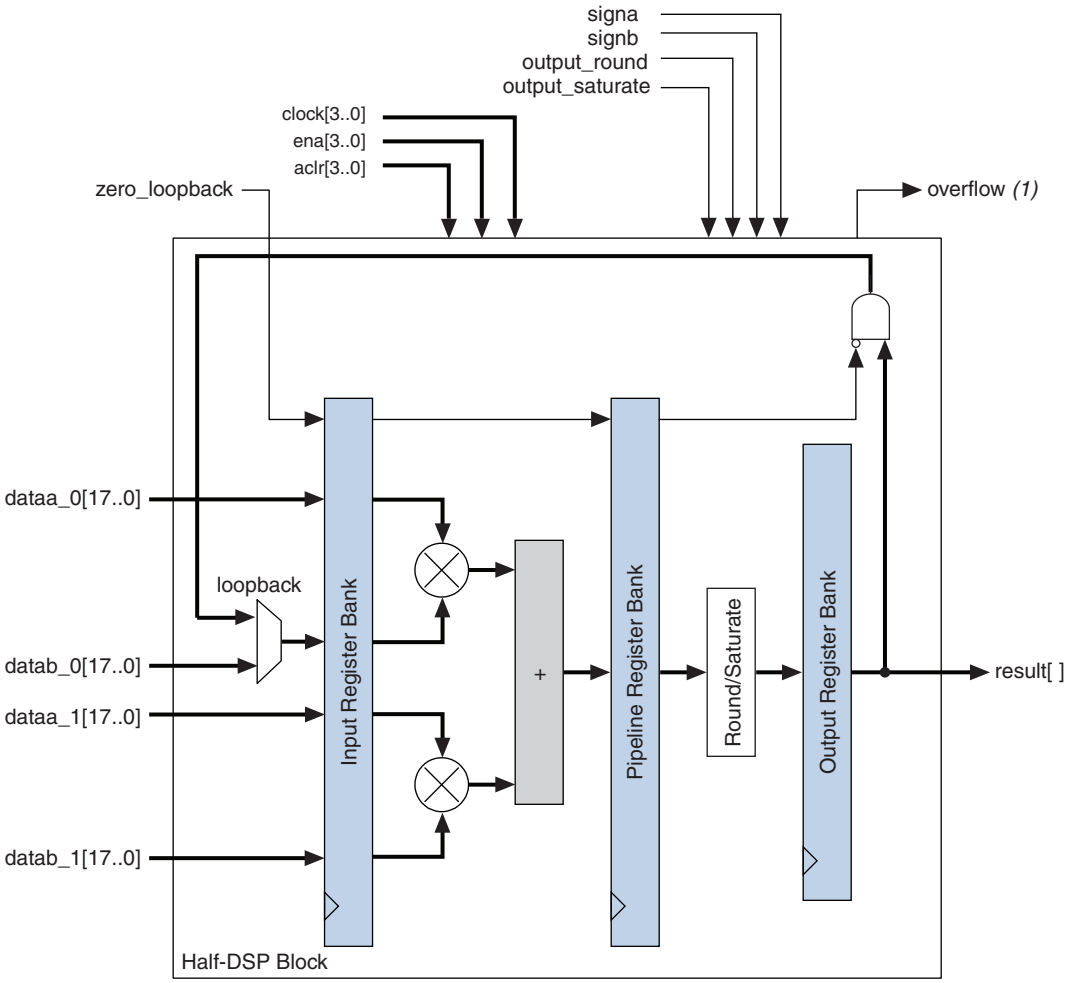
(1) In a half-DSP block, you can implement 2 two-multiplier adders.

(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.

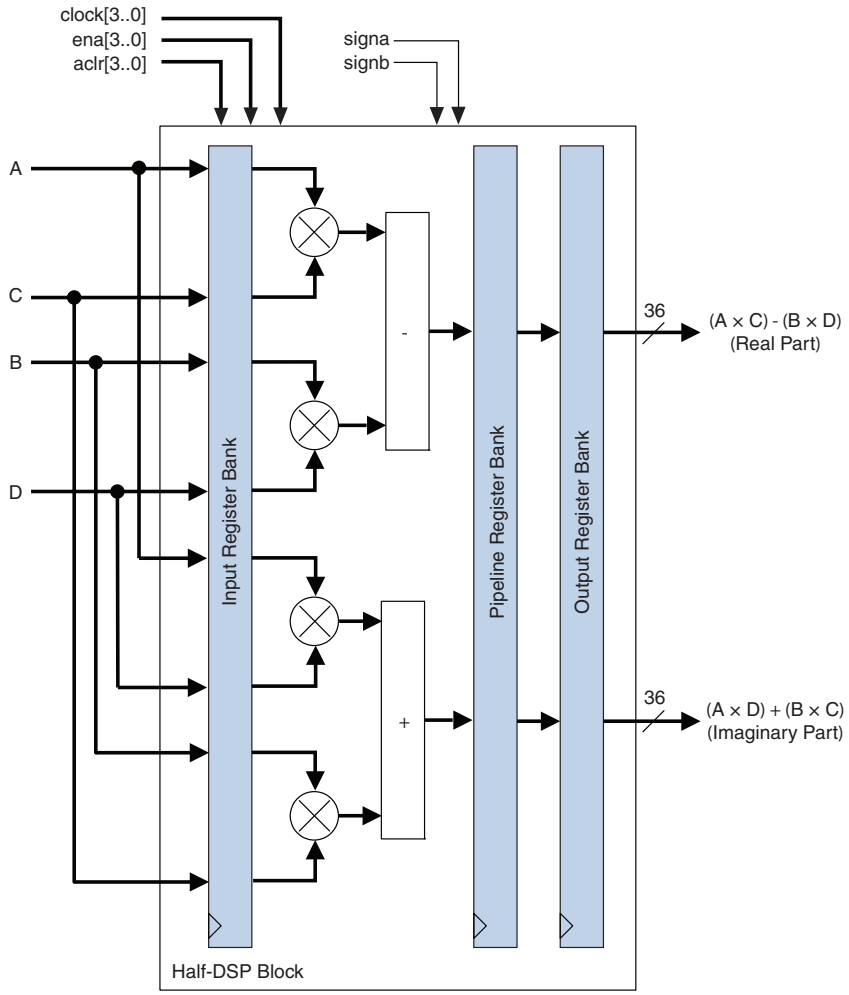
**Equation 4–4. Complex Multiplication Equation**

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

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.

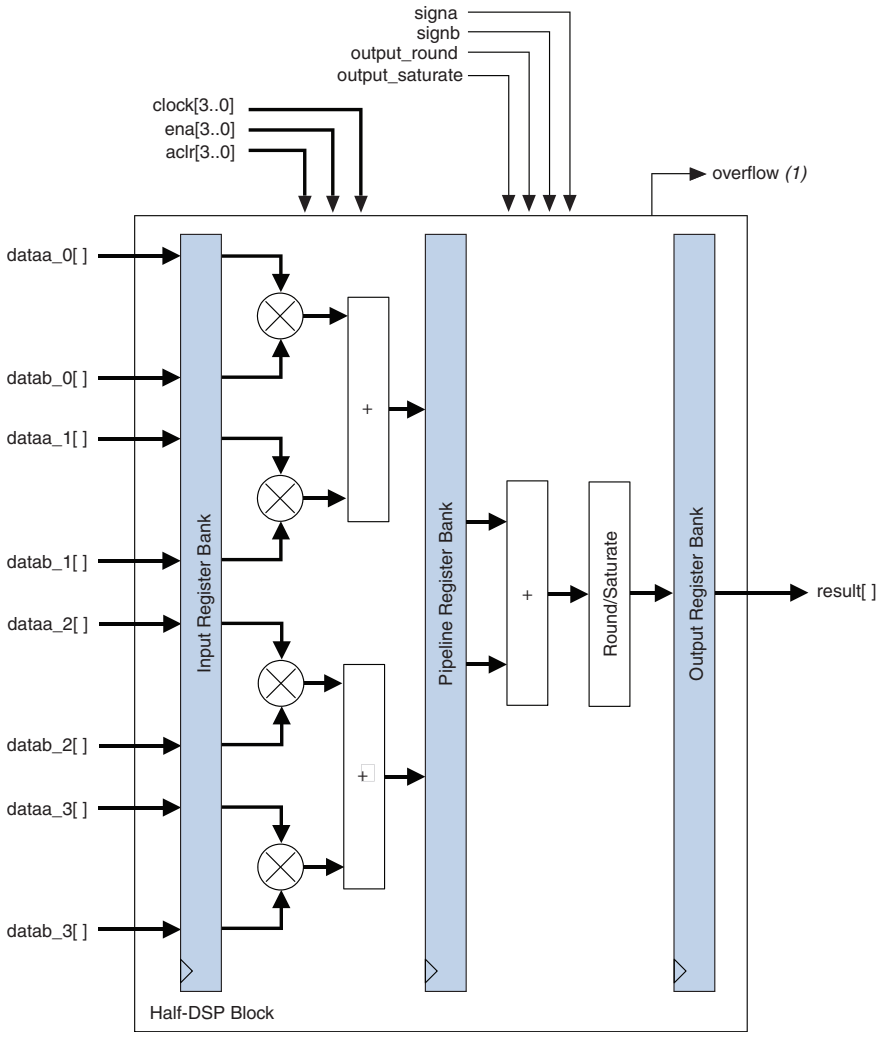
**Figure 4–15. Complex Multiplier Using Two-Multiplier Adder Mode**



## 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

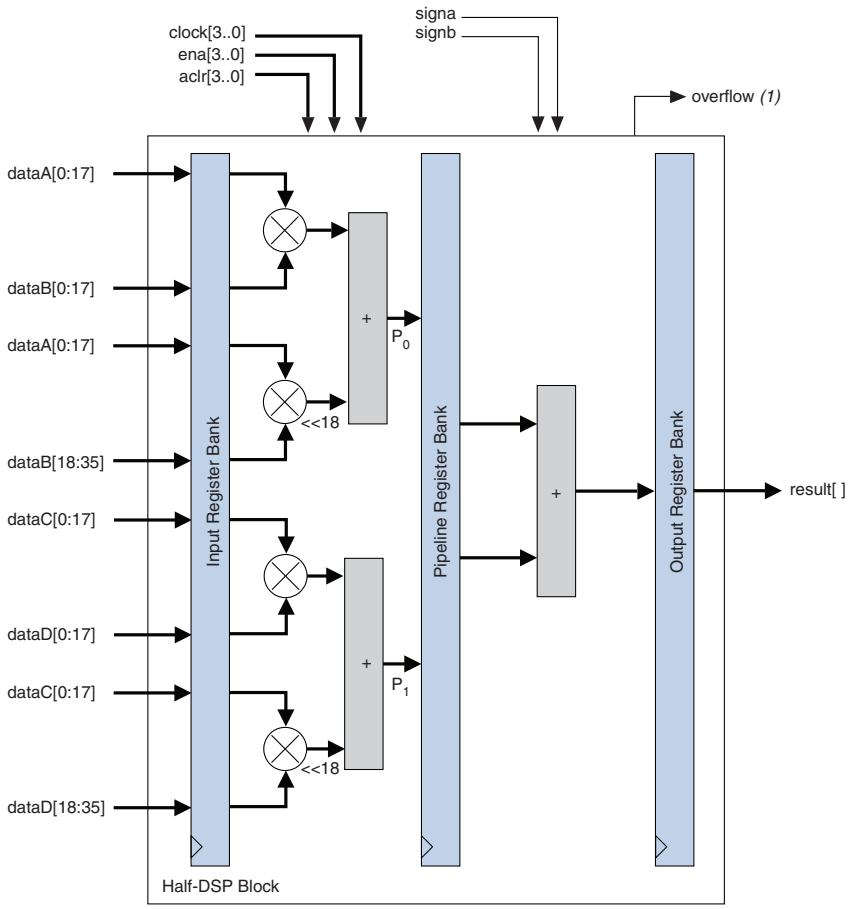
In the high-precision multiplier adder, the DSP block can implement 2 two-multiplier adders, with a multiplier precision of 18 × 36 (one two-multiplier adder per half-DSP block). This mode is useful in filtering or FFT applications where a datapath greater than 18 bits is required, yet 18 bits is sufficient 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:**

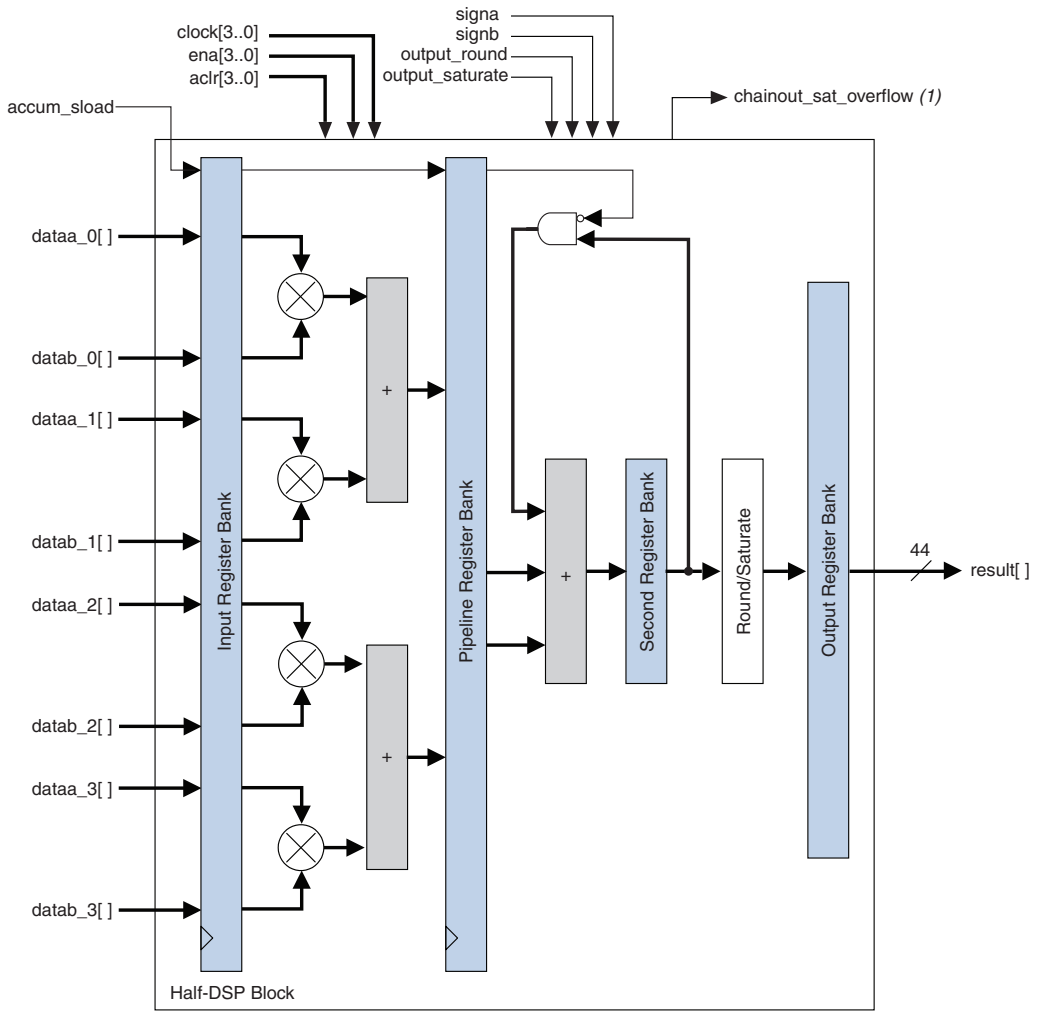
(1) Block output for accumulator overflow and saturate overflow.

## 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**

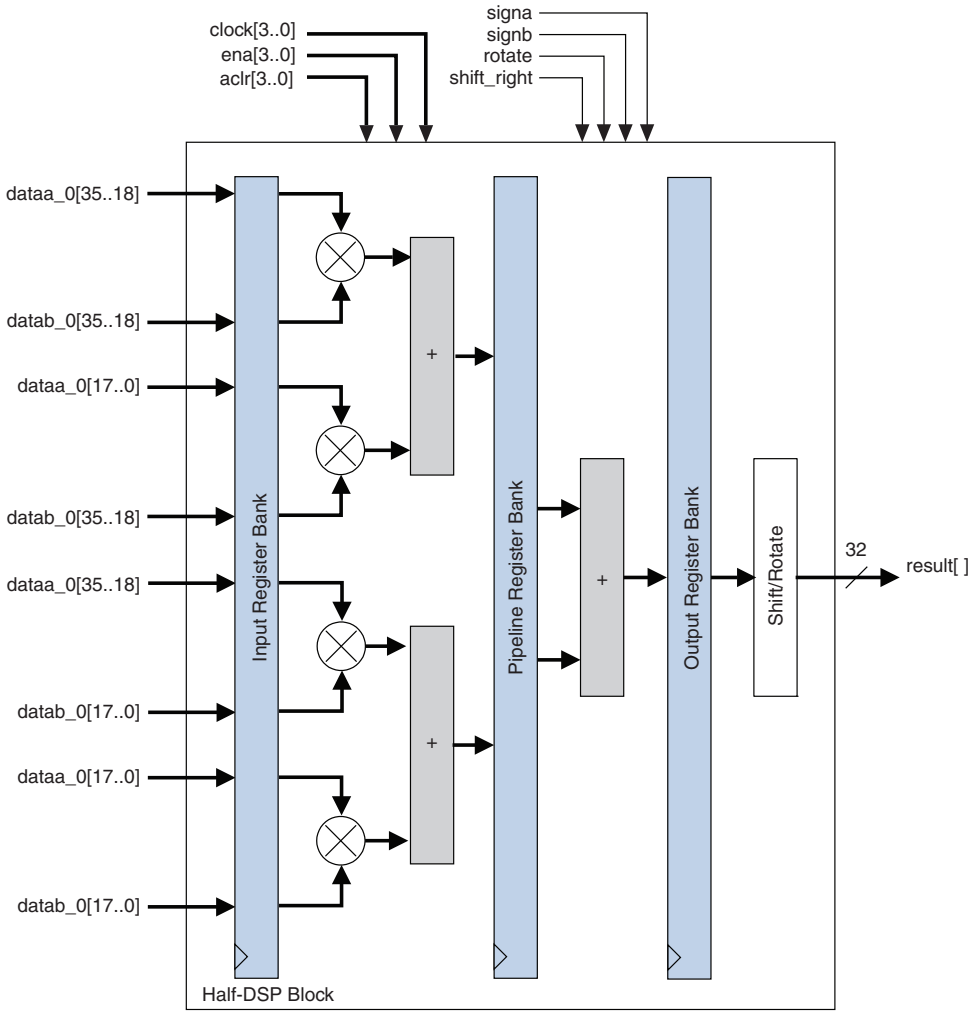



Table 4–5 lists examples of shift operations.

**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

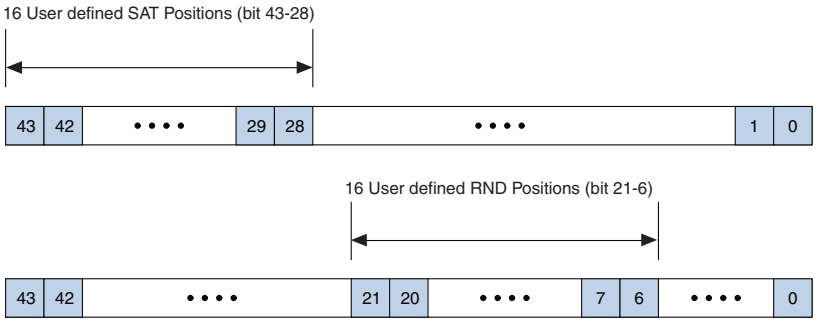
Table 4–8 lists how the saturation works. In this example, a 44-bit input is saturated to 36-bits.

**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<br>signb | Signed/unsigned control for all multipliers and adders.<br>signa for "multiplicand" input bus to dataa[17:0] each multiplier.<br>signb for "multiplier" input bus datab[17:0] to each multiplier.<br>■ signa = 1, signb = 1 for signed-signed multiplication<br>■ signa = 1, signb = 0 for signed-unsigned multiplication<br>■ signa = 0, signb = 1 for unsigned-signed multiplication<br>■ signa = 0, signb = 0 for unsigned-unsigned multiplication | 2 |
| output_round | Round control for first stage round/saturation block.<br>■ output_round = 1 for rounding on multiply output<br>■ output_round = 0 for normal multiply output | 1 |
| chainout_round | Round control for second stage round/saturation block.<br>■ chainout_round = 1 for rounding on multiply output<br>■ 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.<br>■ output_saturate = 1 for saturation support<br>■ 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.<br>■ chainout_saturate = 1 for saturation support<br>■ chainout_saturate = 0 for no saturation support | 1 |
| accum_sload | Dynamically specifies whether the accumulator value is zero.<br>■ accum_sload = 0, accumulation input is from the output registers<br>■ 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<br>clock1<br>clock2<br>clock3 | DSP-block-wide clock signals | 4 |
| ena0<br>ena1<br>ena2<br>ena3 | Input and Pipeline Register enable signals | 4 |
| aclr0<br>aclr1<br>aclr2<br>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.<br>■ Added Arria II GZ devices information.<br>■ Updated "DSP Block Overview", "Operational Modes Overview", and "DSP Block Resource Descriptions" sections.<br>■ Updated Table 4–1<br>■ Added Figure 4–3, Figure 4–7, Figure 4–11, and Figure 4–15<br>■ Minor text edits |
| July 2010 | 3.0 | Updated for the Arria II GX v10.0 release:<br>■ Updated "DSP Block Resource Descriptions" and "Second-Stage Adder" sections<br>■ Minor text edits |
| November 2009 | 2.0 | Updated for Arria II GX v9.1 release:<br>■ Updated Table 4–1 and Table 4–9<br>■ Updated Figure 4–9<br>■ 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

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.

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<br>16 GCLKs + 22 RCLKs |
| GCLKs/RCLKs per device | 64 | 80/104 *(4)* | 16 GCLKs + 48 RCLKs | 16 GCLKs + 64 RCLKs<br>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.

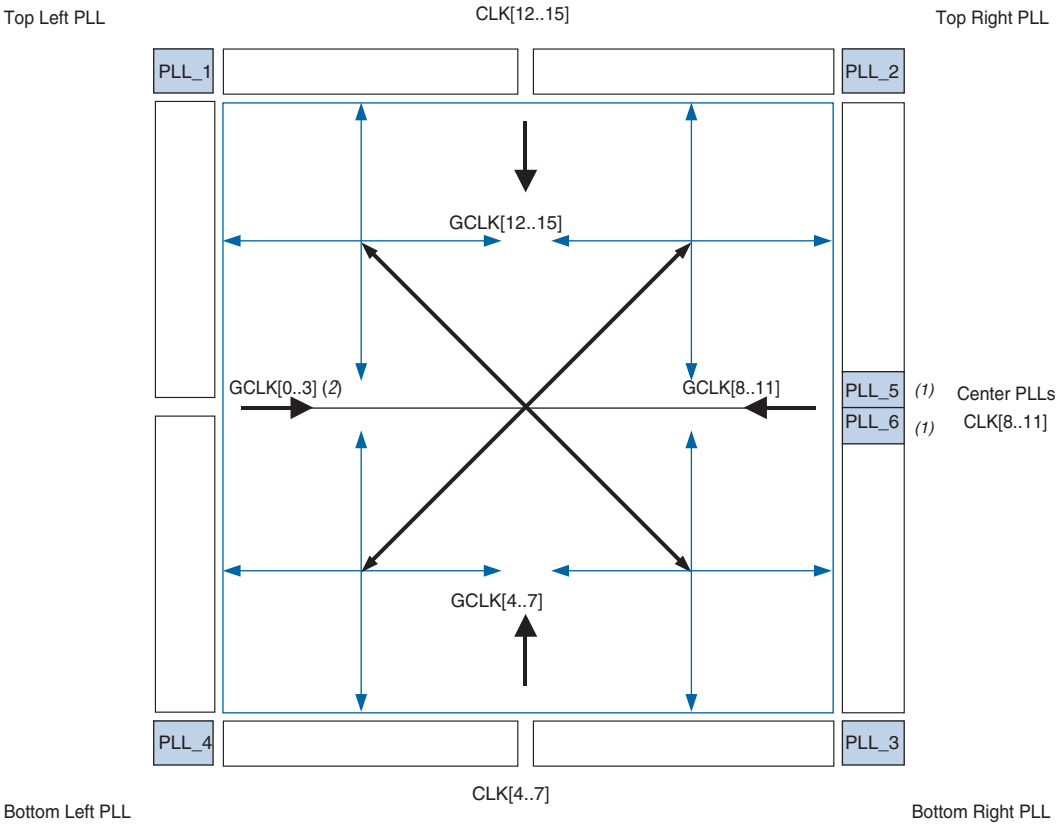
Arria II GX devices have up to 12 dedicated single-ended clock pins or six dedicated differential clock pins (DIFFCLK_[0..5]p and DIFFCLK_[0..5]n) that can drive either the GCLK or RCLK networks. These clock pins are arranged on the three sides (top, bottom, and right sides) of the Arria II GX device,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

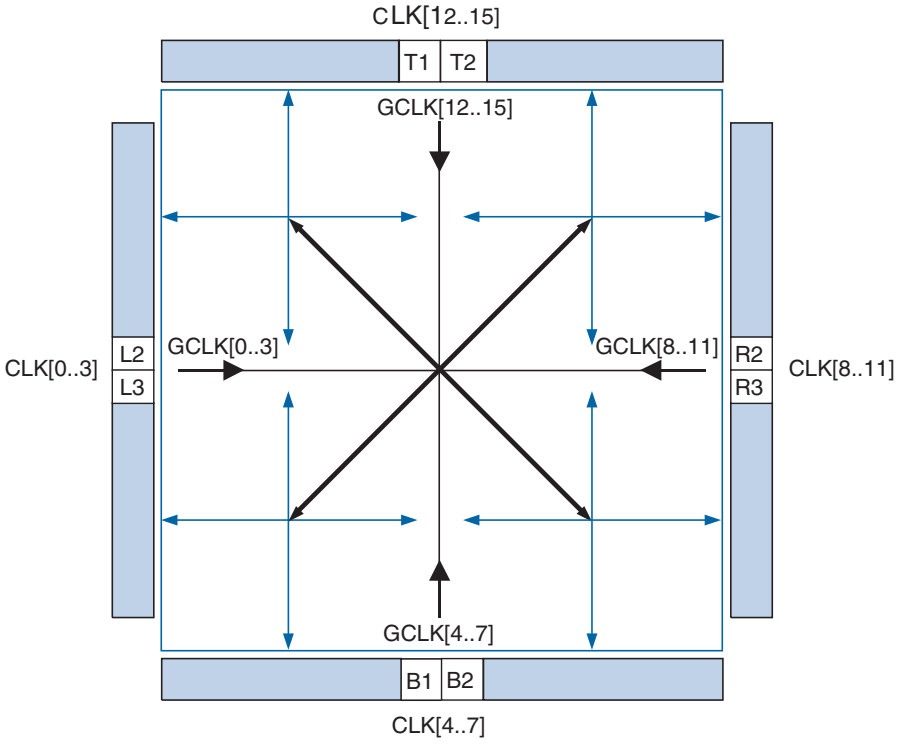



**Notes to Figure 5–1:**

(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.

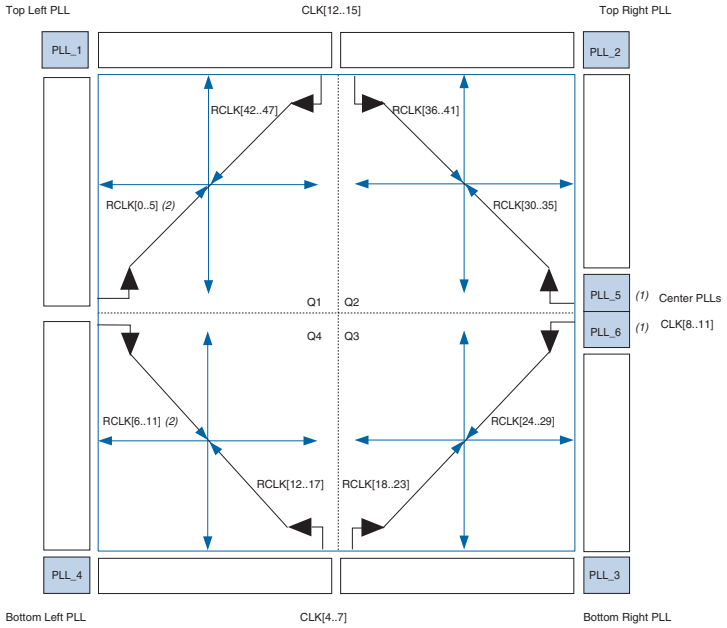
**Figure 5–2. GCLK Networks in Arria II GZ Devices**




## 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

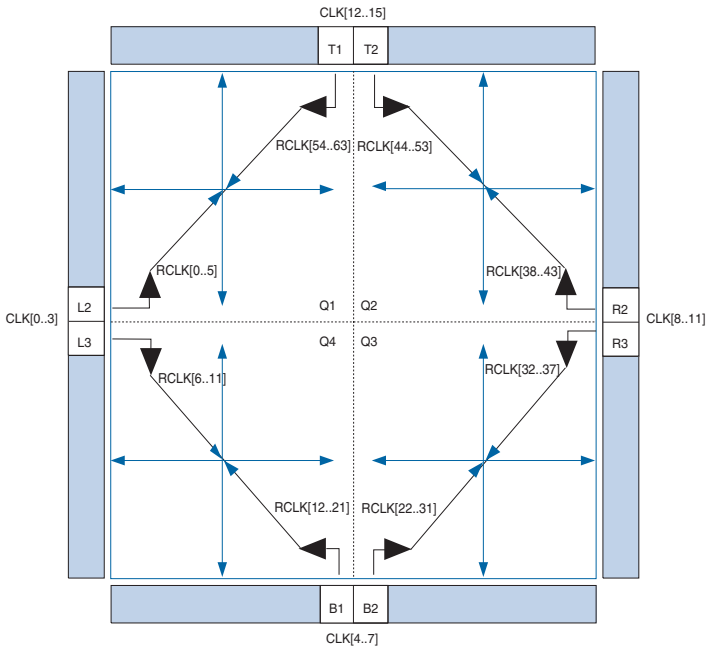



**Notes to Figure 5–3:**

(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)**

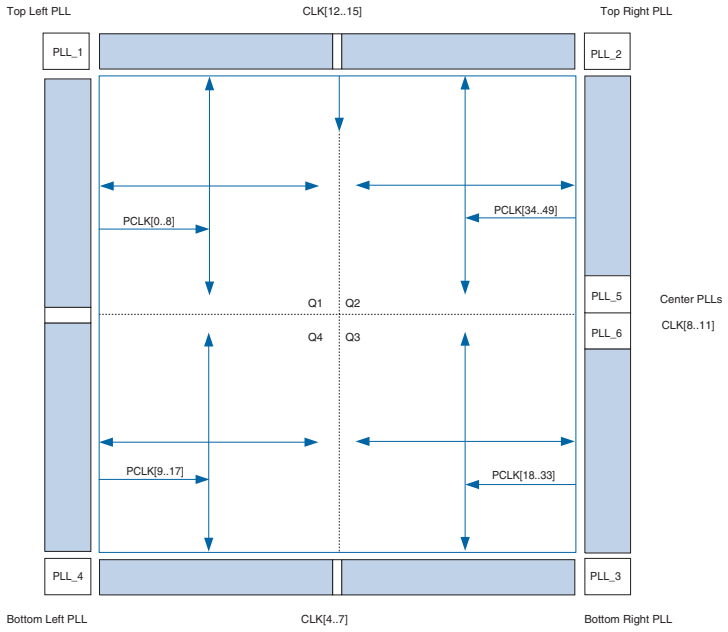


**Figure 5–6. PCLK Networks in (EP2AGX95 and EP2AGX125 Devices)**

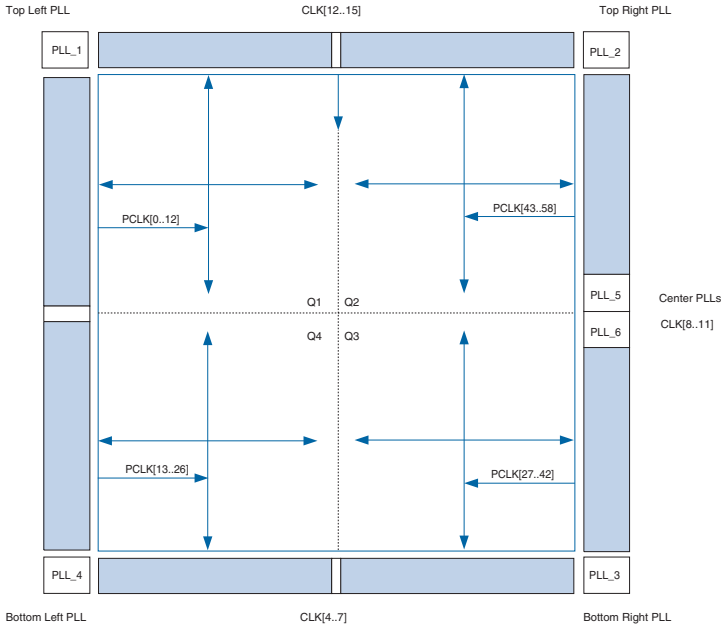



**Figure 5–7. PCLK Networks in (EP2AGX190 and EP2AGX260 Devices)**

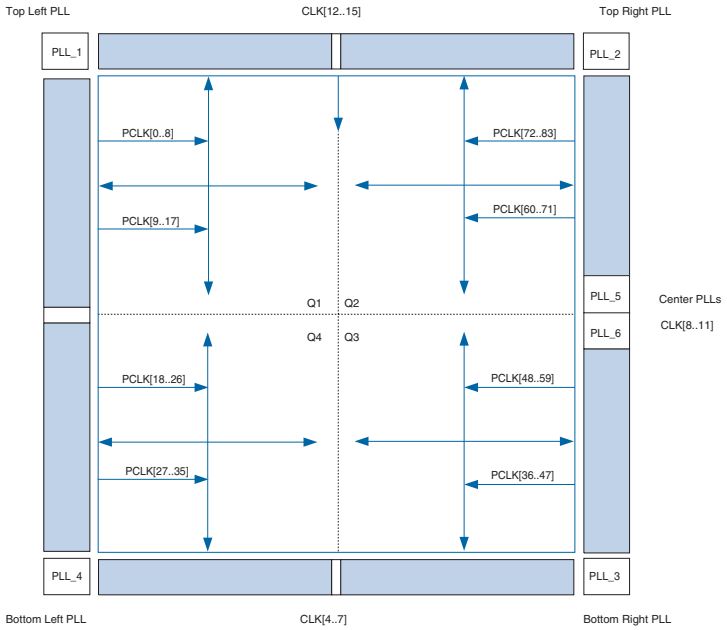
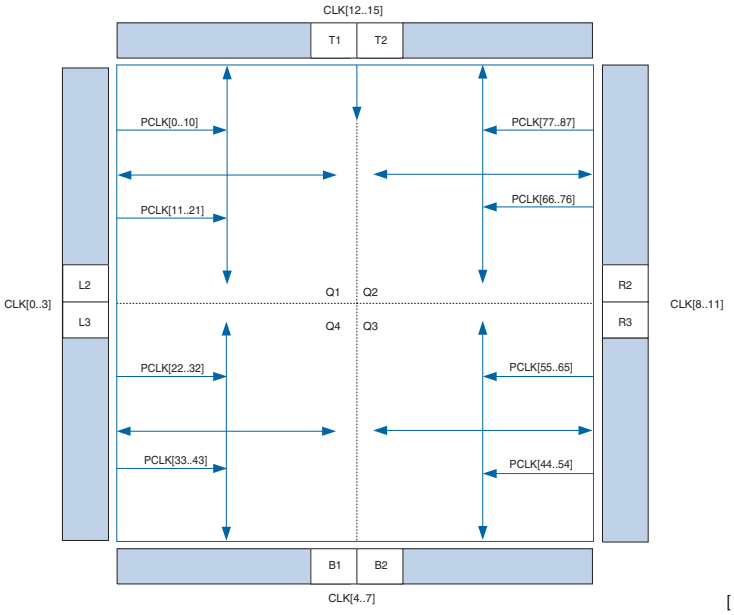


**Figure 5–8. PCLK Networks in Arria II GZ Devices**

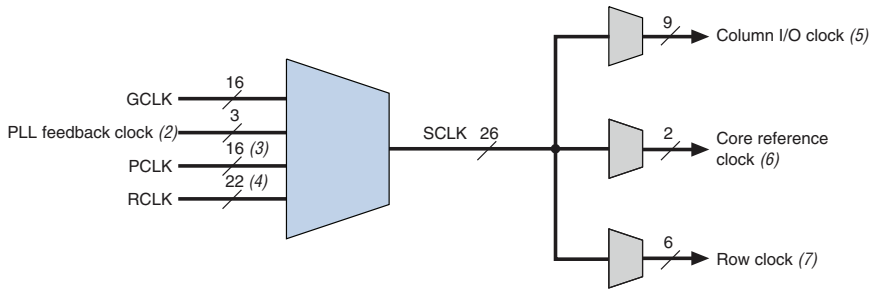



## Clock Sources Per Quadrant

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**

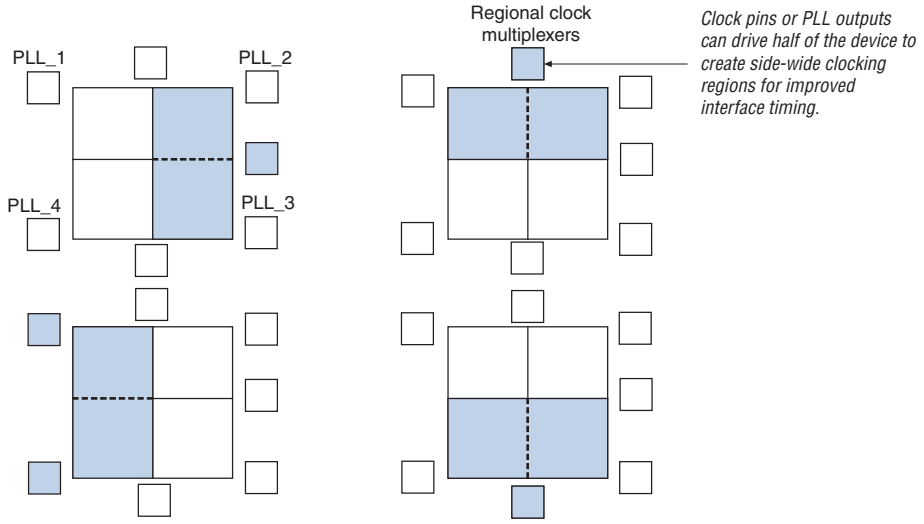
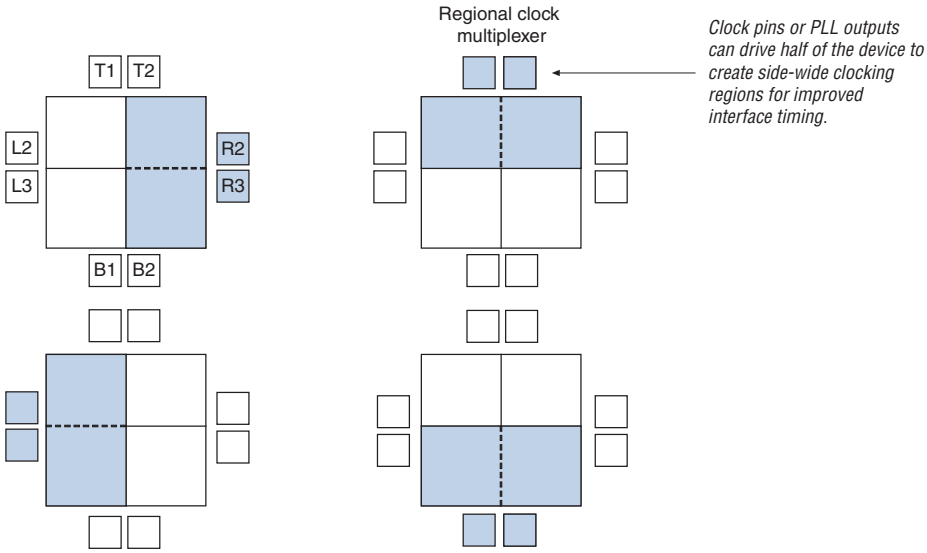



**Figure 5–11. Device Dual-Regional Clock Region for Arria II GZ Devices**



# 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] | — | — | — | — | — | — | ✓ | ✓ |

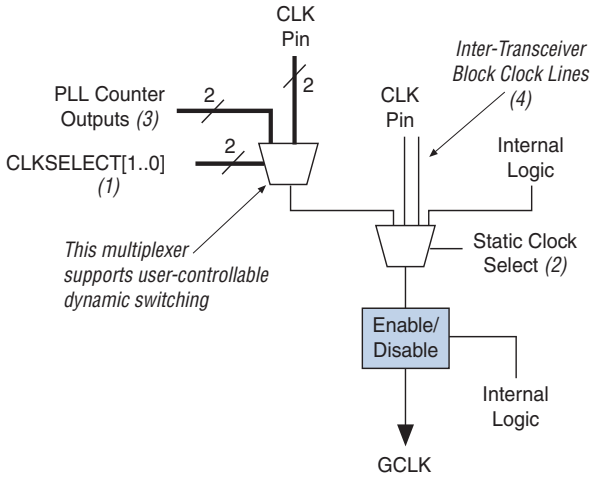
## Clock Control Block

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.<br>■ 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.<br>■ 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.

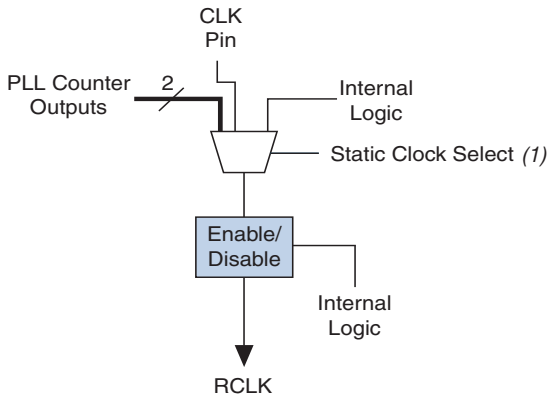
You can set the input clock sources and the clkena signals for the GCLK and RCLK clock network multiplexers through the Quartus II software with the ALTCLKCTRL megafunction. You can also enable or disable the dedicated external clock output pins with the ALTCLKCTRL megafunction.

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.

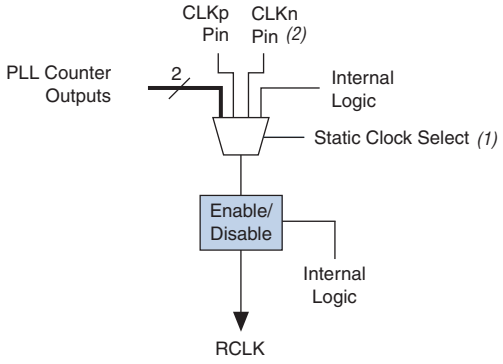
**Figure 5–13. RCLK Control Block for Arria II GX Devices**




**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**



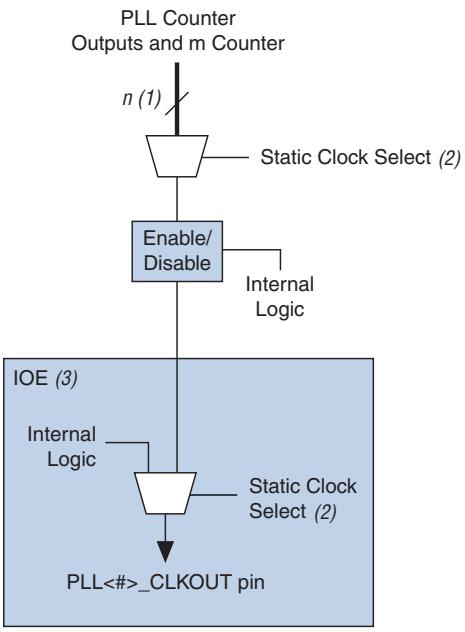

**Notes to Figure 5–14:**

(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.

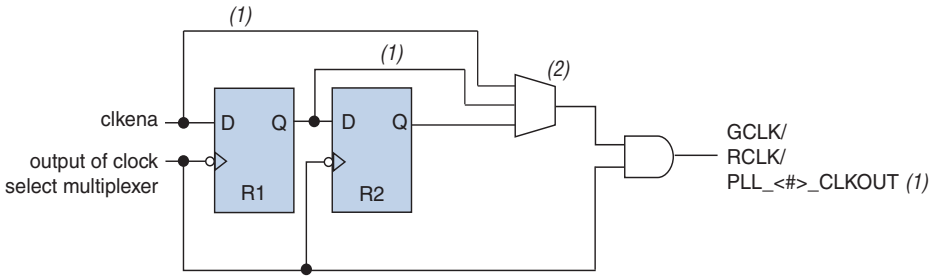
(2) When the device is in user mode, you can only set the clock select signals through a configuration file (**.sof** or **.pof**). You cannot dynamically control the clock.

(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:**

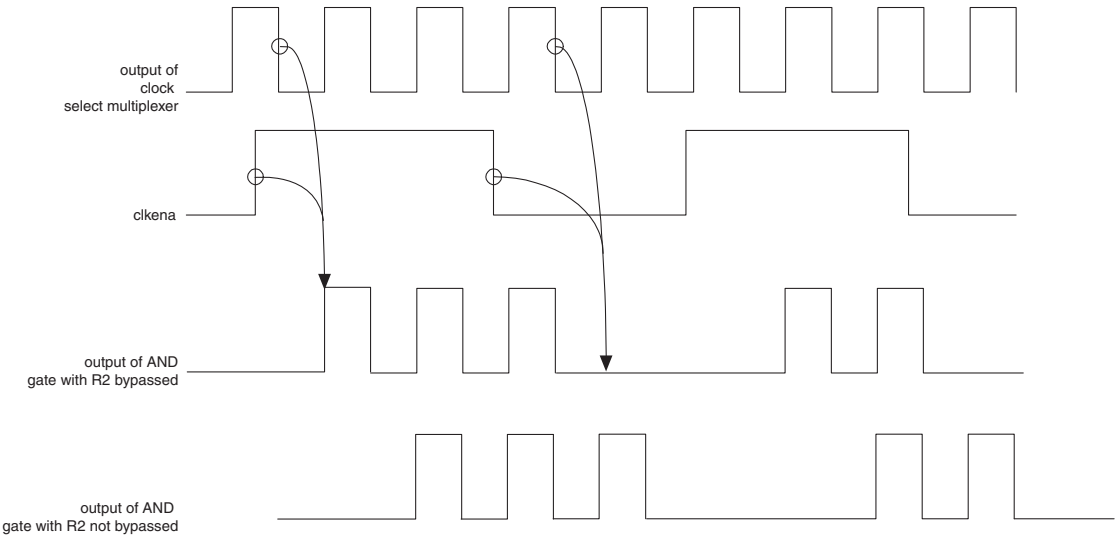
(1) The R1 and R2 bypass paths are not available for PLL external clock outputs.

(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.

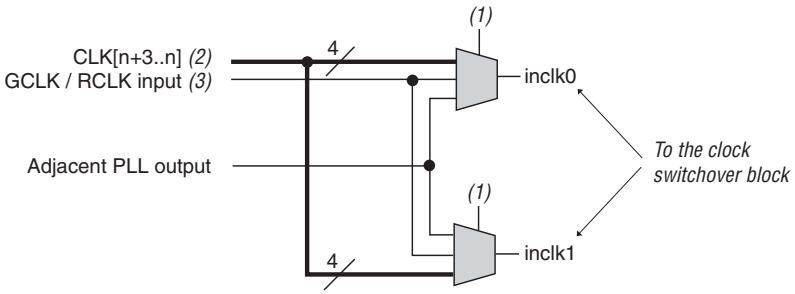
## Clock Source Control for PLLs

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**



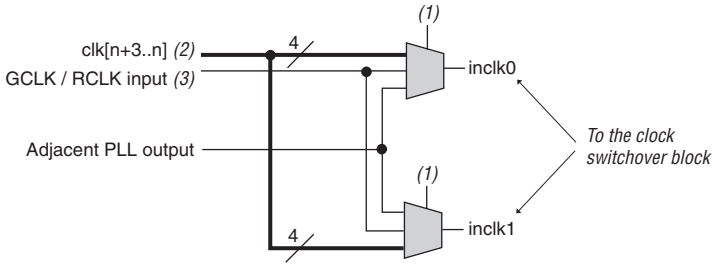

**Notes to Figure 5–18:**

(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<br>3 single-ended or 3 differential pairs *(1)*, *(2)* | 6 single-ended or<br>4 single-ended and<br>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.

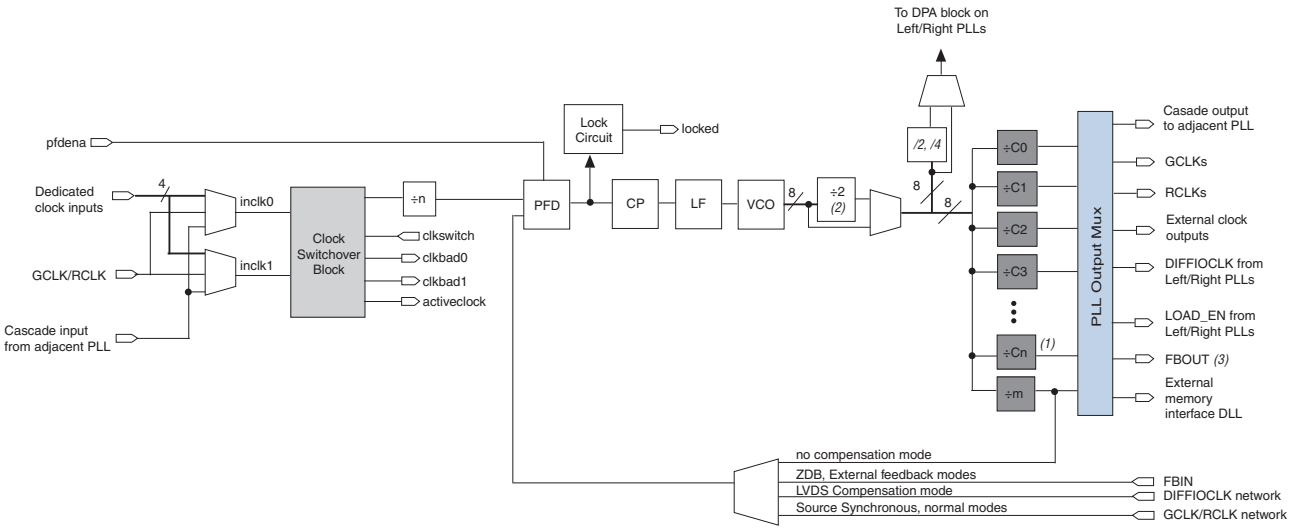
(4) The dedicated path between adjacent PLLs is not available on L1, L4, R1, and R4 PLLs.

(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.

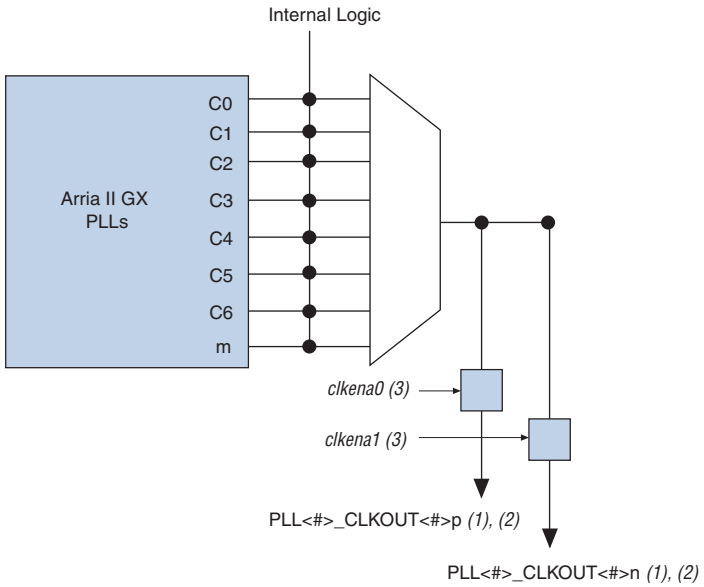
## PLL Clock I/O Pins

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:**

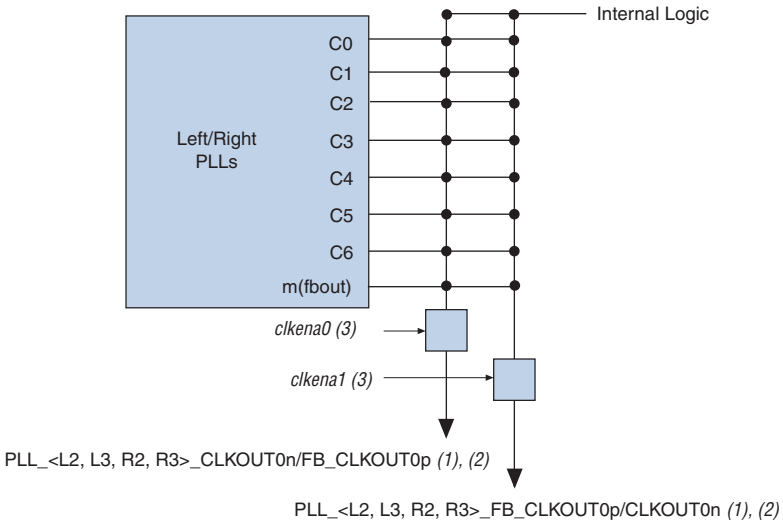
(1) You can feed these clock output pins using any one of the C[9..0], or m counters.

(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:**

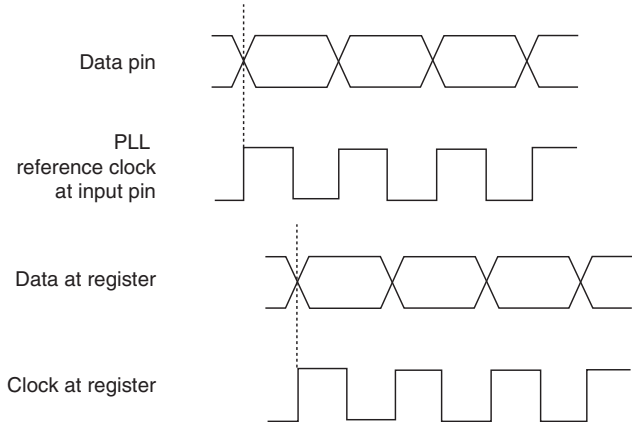
(1) ZDB mode uses 8 ns delay for compensation in Arria II GX devices.
(2) The high-bandwidth PLL setting is not supported in the external feedback mode.
(3) External feedback mode is supported for single-ended inputs and outputs only on the left and right PLLs.
(4) LVDS compensation mode is only supported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

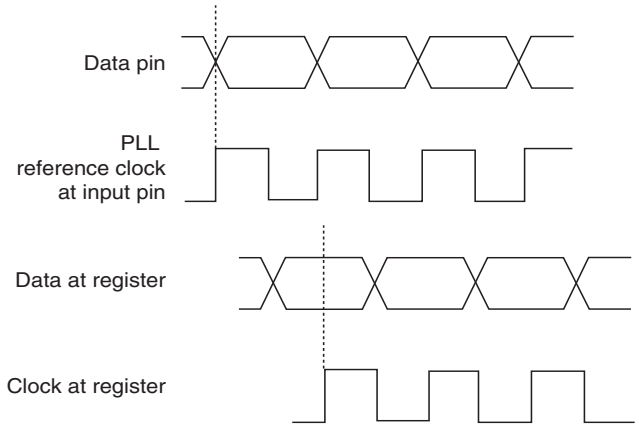
If you do not assign the **PLL Compensation** assignment, the Quartus II software automatically selects all pins driven by the compensated output of the PLL as the compensation target.

## 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.

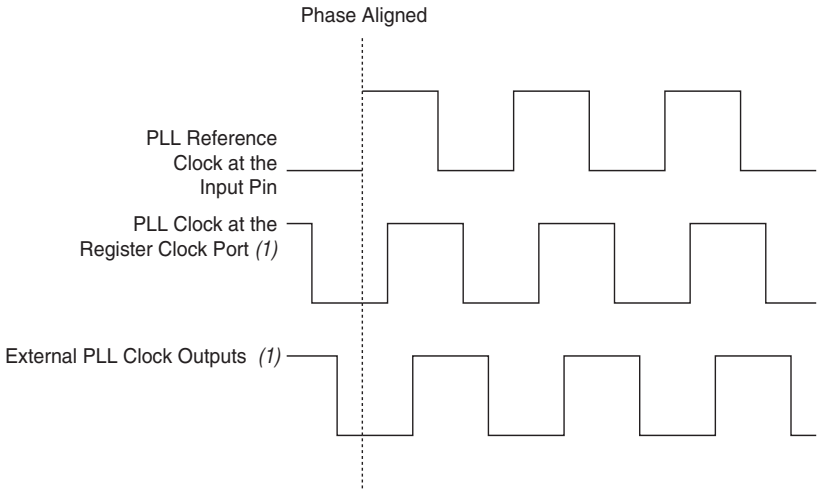
**Figure 5–25. Source-Synchronous Mode for LVDS Compensation for Arria II Devices**




## 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**



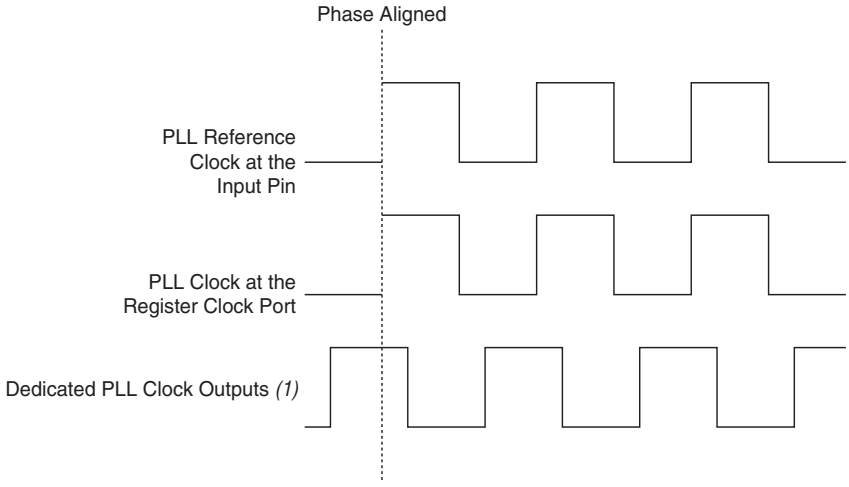

**Note to Figure 5–26:**

(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**

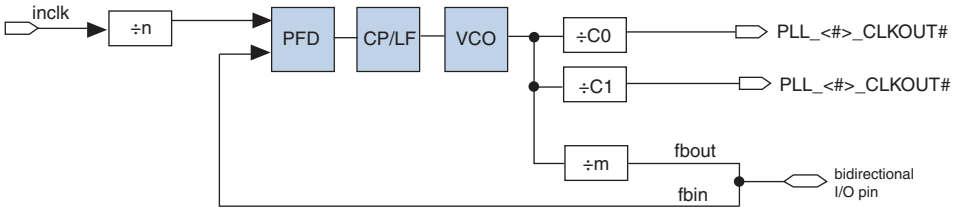
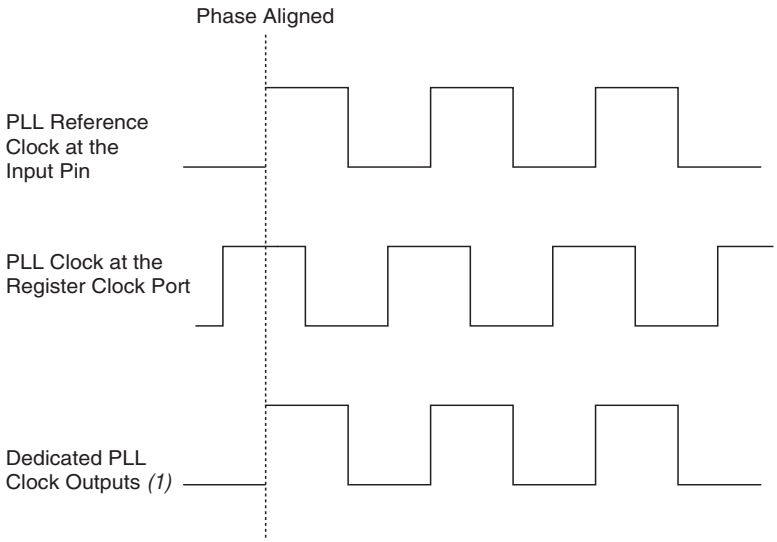


Figure 5–29 shows an example waveform of the PLL clocks' phase relationship in ZDB mode.

**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

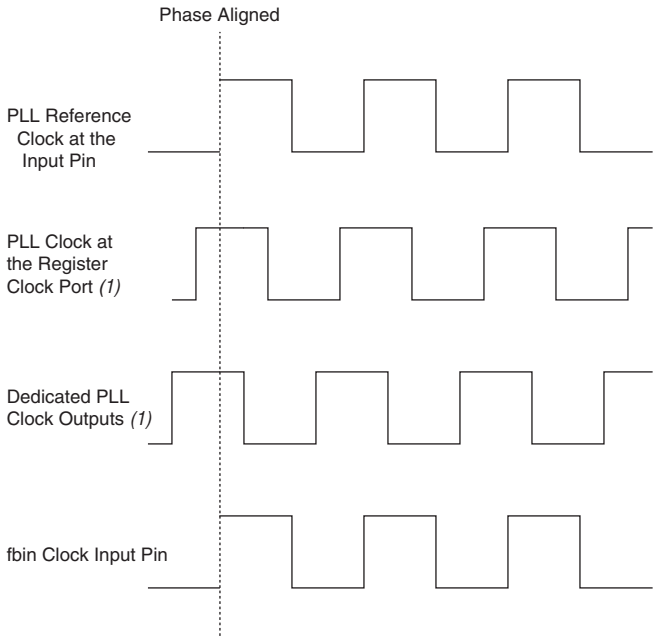
In external feedback mode, the external feedback input pin (`fbin`) is phase-aligned with the clock input pin, as shown in Figure 5–30. Aligning these clocks allows you to remove clock delay and skew between devices. This mode is supported on all Arria II GZ PLLs.

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**

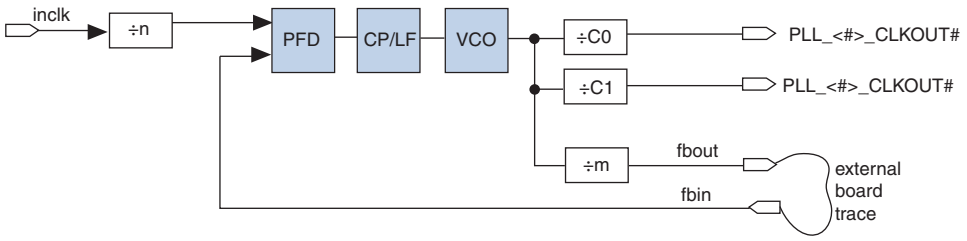



**Note to Figure 5–30:**

(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).

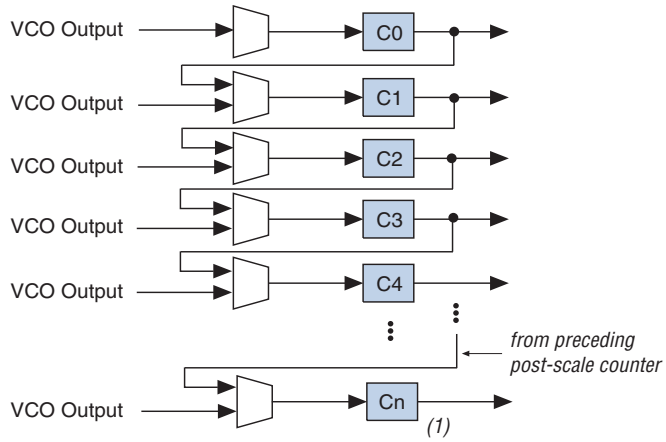
Each PLL has one pre-scale counter (N) and one multiply counter (M) with a range of 1 to 512 for both M and N. The n counter does not use duty-cycle control because the only purpose of this counter is to calculate frequency division. There are seven generic post-scale counters in each PLL that can feed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

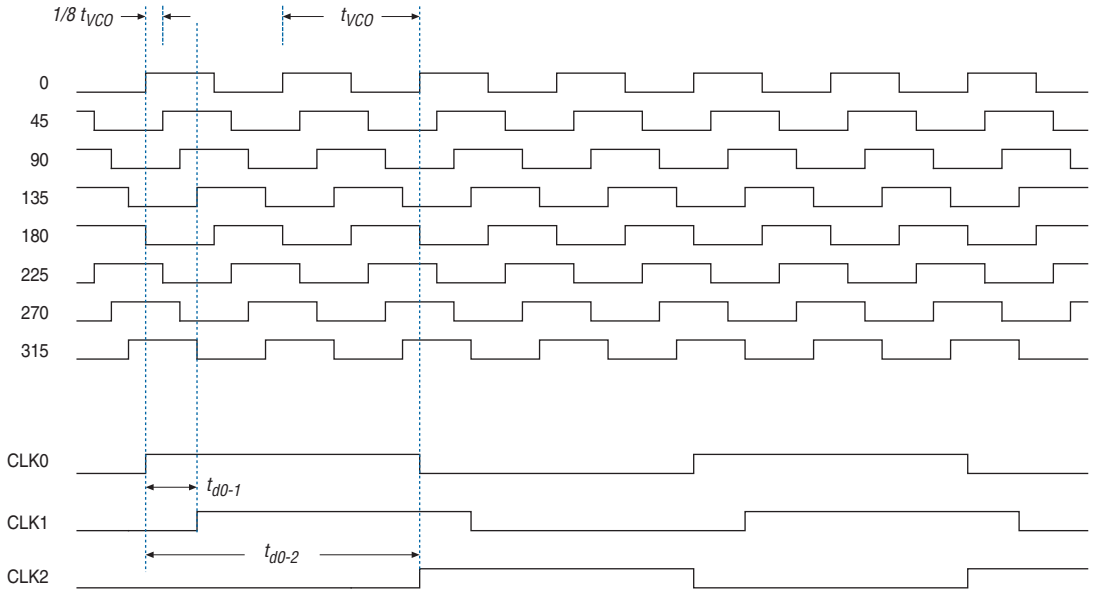
**Equation 5–2. Coarse-Resolution Phase Shifts for Arria II Devices**

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

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 $\Phi_{fine}$. 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 $\Phi_{COARSE}$ (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

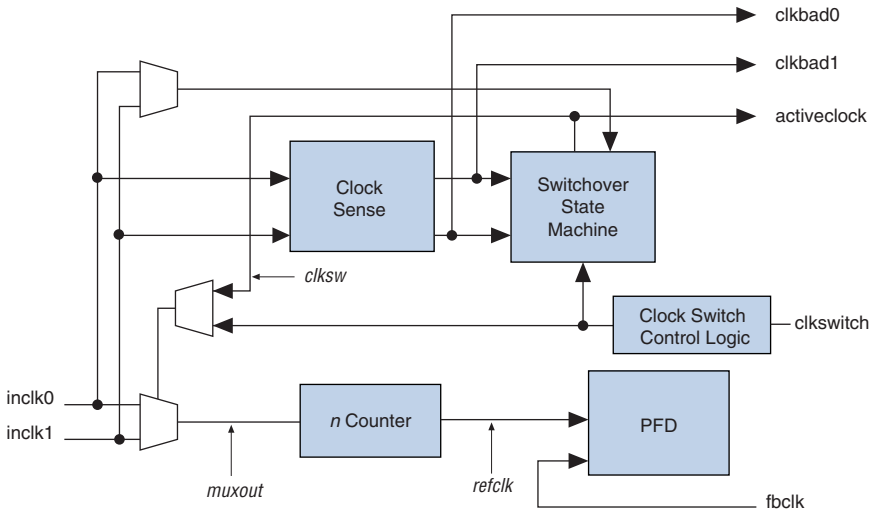
The clock switchover feature allows the PLL to switch between two reference input clocks. Use this feature for clock redundancy or for a dual-clock domain application such as in a system that turns on the redundant clock if the previous clock stops running. Your design can perform clock switchover automatically, 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.

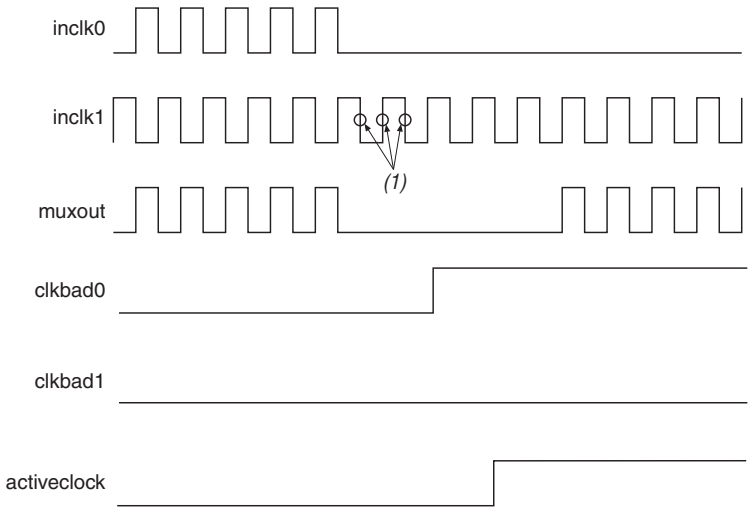
Altera recommends resetting the PLL with the `areset` signal to maintain the phase relationships between the PLL input and output clocks when you use clock switchover.

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.

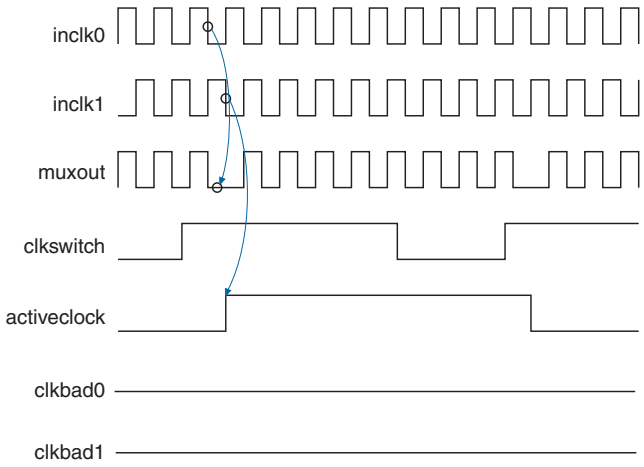
### Manual Override Mode

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.

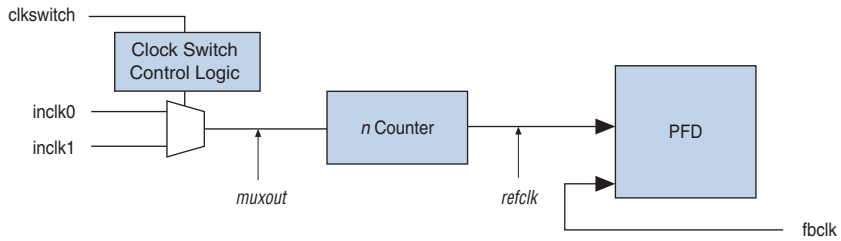
In automatic switchover with manual override mode, the activeclock signal mirrors the clkswitch signal. As both clocks are still functional during the manual switch, neither clkbad signal goes high. Because the switchover circuit is positive-edge sensitive, the falling edge of the clkswitch signal does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

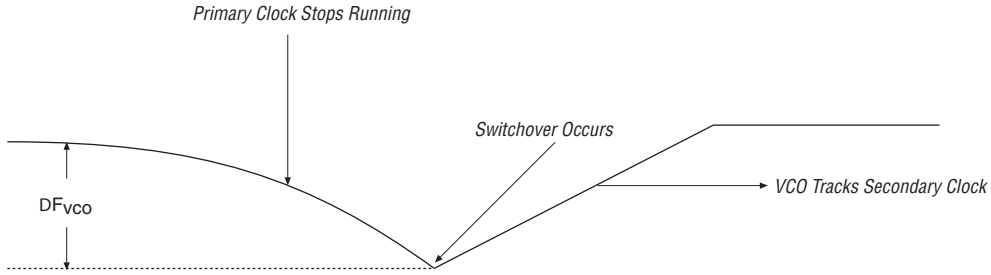
- Automatic clock switchover requires that the inclk0 and inclk1 frequencies be in 100% (2x) of each other. Failing to meet this requirement causes the clkbad[0] and clkbad[1] signals to not function properly.
- When you use manual clock switchover mode, the difference between inclk0 and inclk1 can be 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.

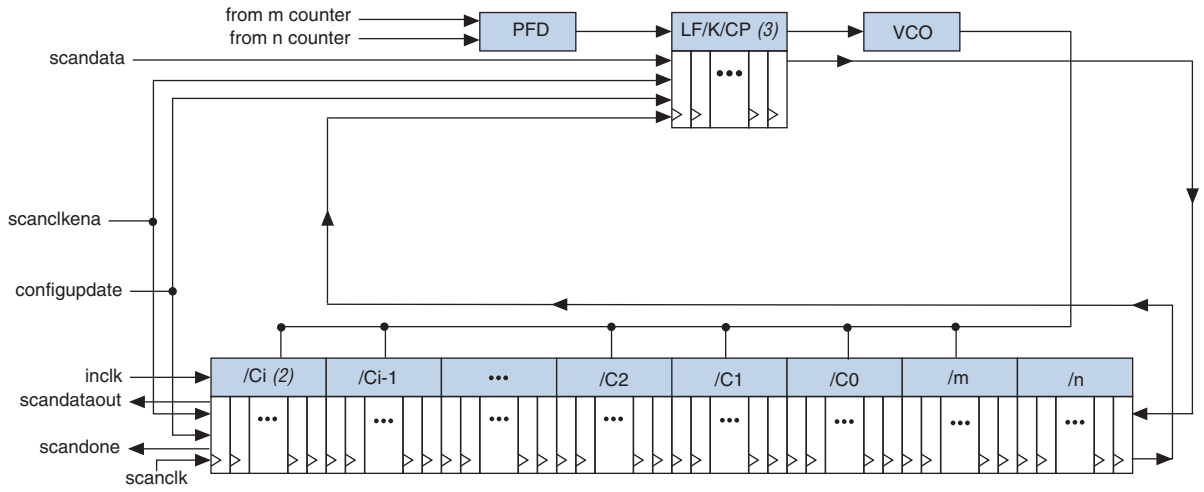
## PLL Reconfiguration Hardware Implementation

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*.

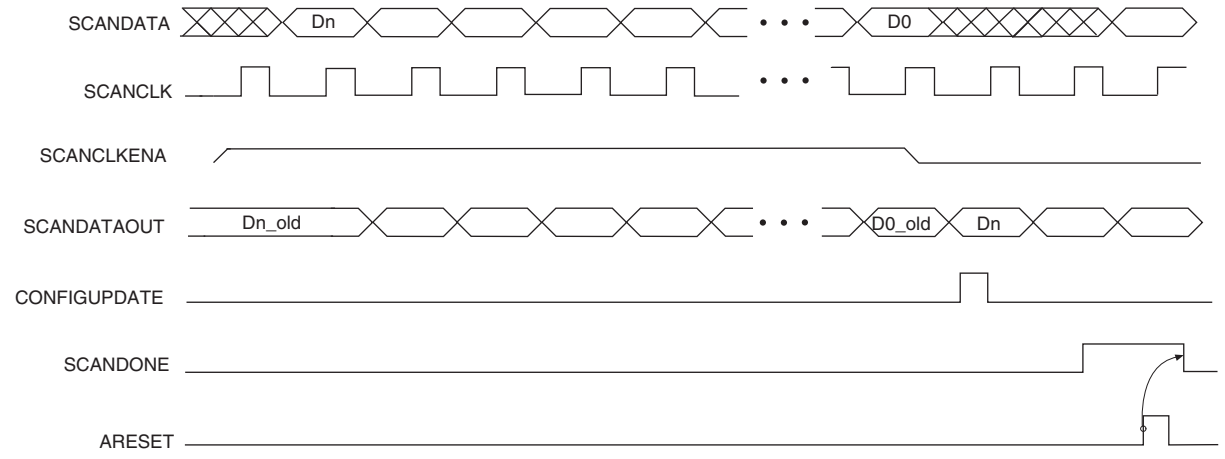
The counter settings are updated synchronously to the clock frequency of the individual counters. Therefore, all counters are not simultaneously updated.

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**

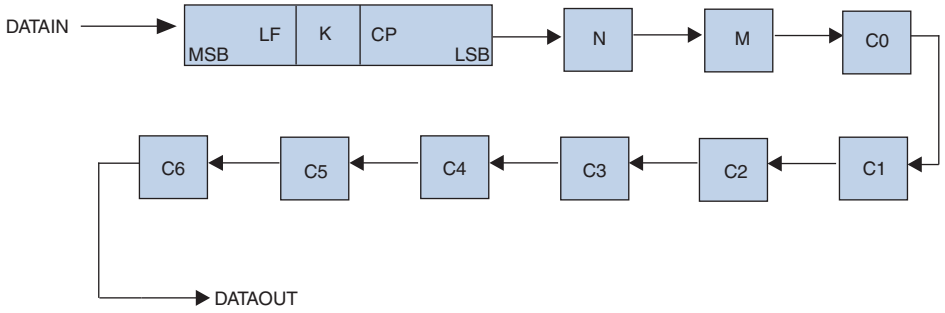
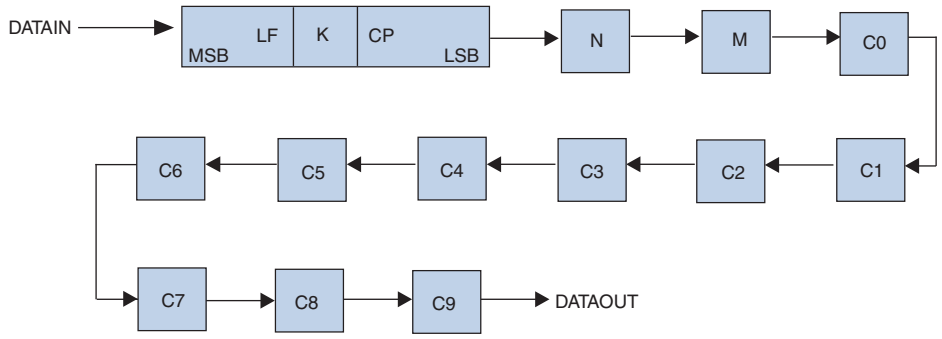



Figure 5–42 shows the scan chain order of PLL components for the top and bottom Arria II GZ PLLs.

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

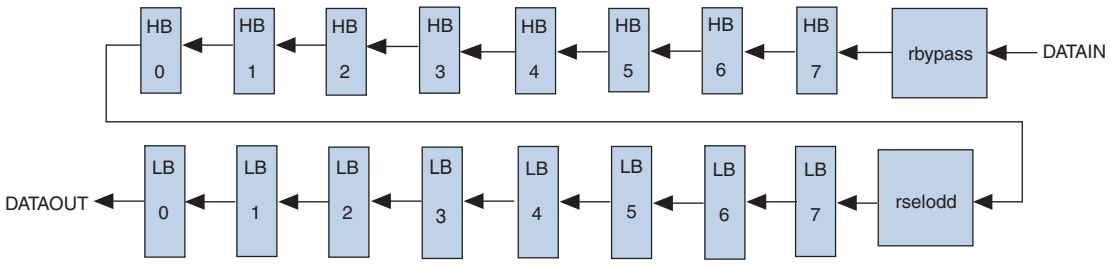



**Note to Figure 5–43:**

(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.

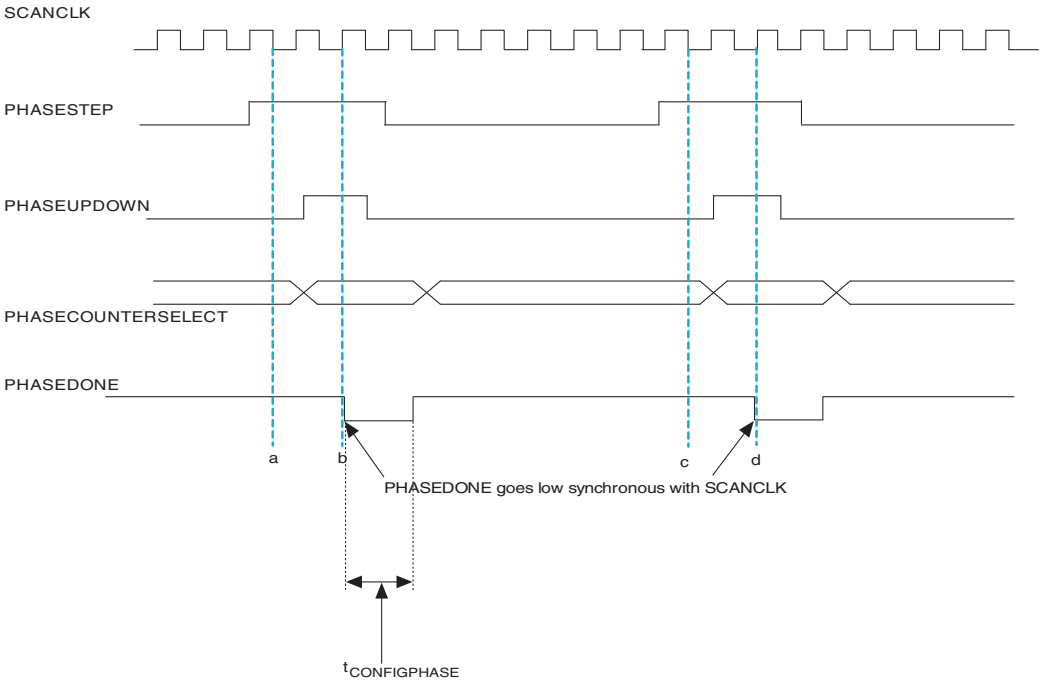
You can repeat dynamic phase-shifting indefinitely. For example, in a design where the VCO frequency is set to 1,000 MHz and the output clock frequency is set to 100 MHz, performing 40 dynamic phase shifts (each one yields 125 ps phase shift) results in shifting the output clock by 180°, in other words, 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.<br>■ Updated Figure 5–44.<br>■ Updated "Dynamic Phase-Shifting" section.<br>■ Added Figure 5–5, Figure 5–6, Figure 5–7, and Figure 5–8.<br>■ Minor text edits. |
| December 2010 | 4.0 | ■ Updated for the Quartus II software version 10.1 release.<br>■ Added Arria II GZ devices information.<br>■ Updated Table 5–1, Table 5–12, Table 5–20, and Table 5–21.<br>■ 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.<br>■ Added Table 5–5, Table 5–7, Table 5–9, Table 5–11, andTable 5–16.<br>■ Added "Clock Sources Per Quadrant" and "External Feedback Mode" sections.<br>■ Minor text edit. |
| July 2010 | 3.0 | Updated for Arria II GX v10.0 release:<br>■ Updated "Clock Regions" and "Arria II PLL Hardware Overview" sections.<br>■ Updated Figure 5–44.<br>■ Removed sub-regional clock references.<br>■ Minor text edit. |
| November 2009 | 2.0 | Updated for Arria II GX v9.1 release:<br>■ Updated Table 5–1.<br>■ Updated Figure 5–14.<br>■ Updated the "Periphery Clock (PCLK) Networks" and "Cascading PLLs" sections.<br>■ Minor text edit. |
| June 2009 | 1.1 | ■ Updated Table 5–8.<br>■ Updated Figure 5–13 and Figure 5–14.<br>■ 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

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.

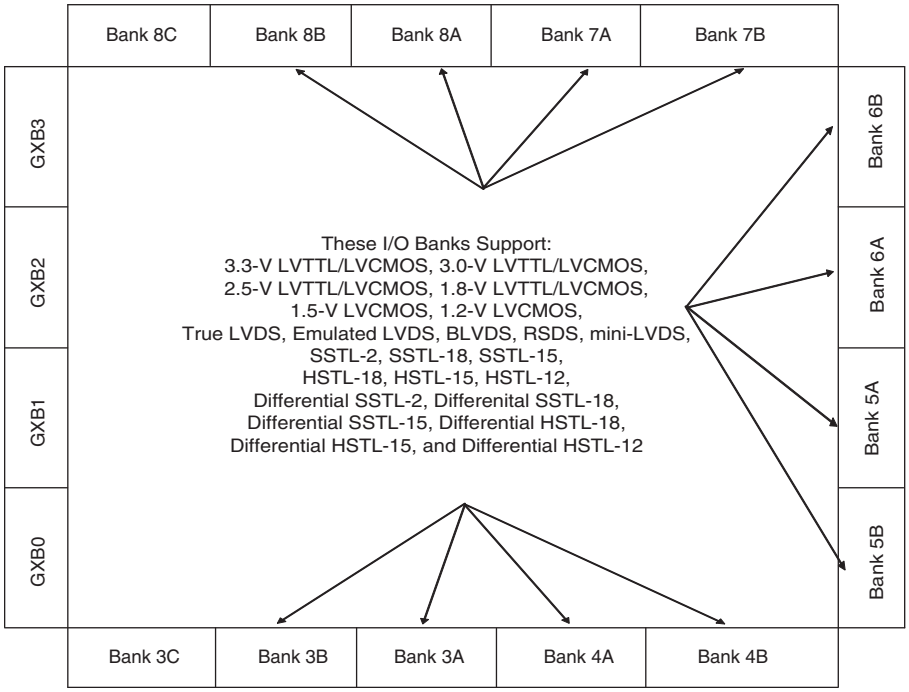
(8) The emulated differential output standard that supports the tri-state feature includes: LVDS_E_1R, LVDS_E_3R, RSDS_E_1R, RSDS_E_3R, Mini_LVDS_E_1R, and Mini_LVDS_E_3R. For more information, refer to the *I/O Buffer (ALTIOBUF) Megafunction User Guide*.

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.

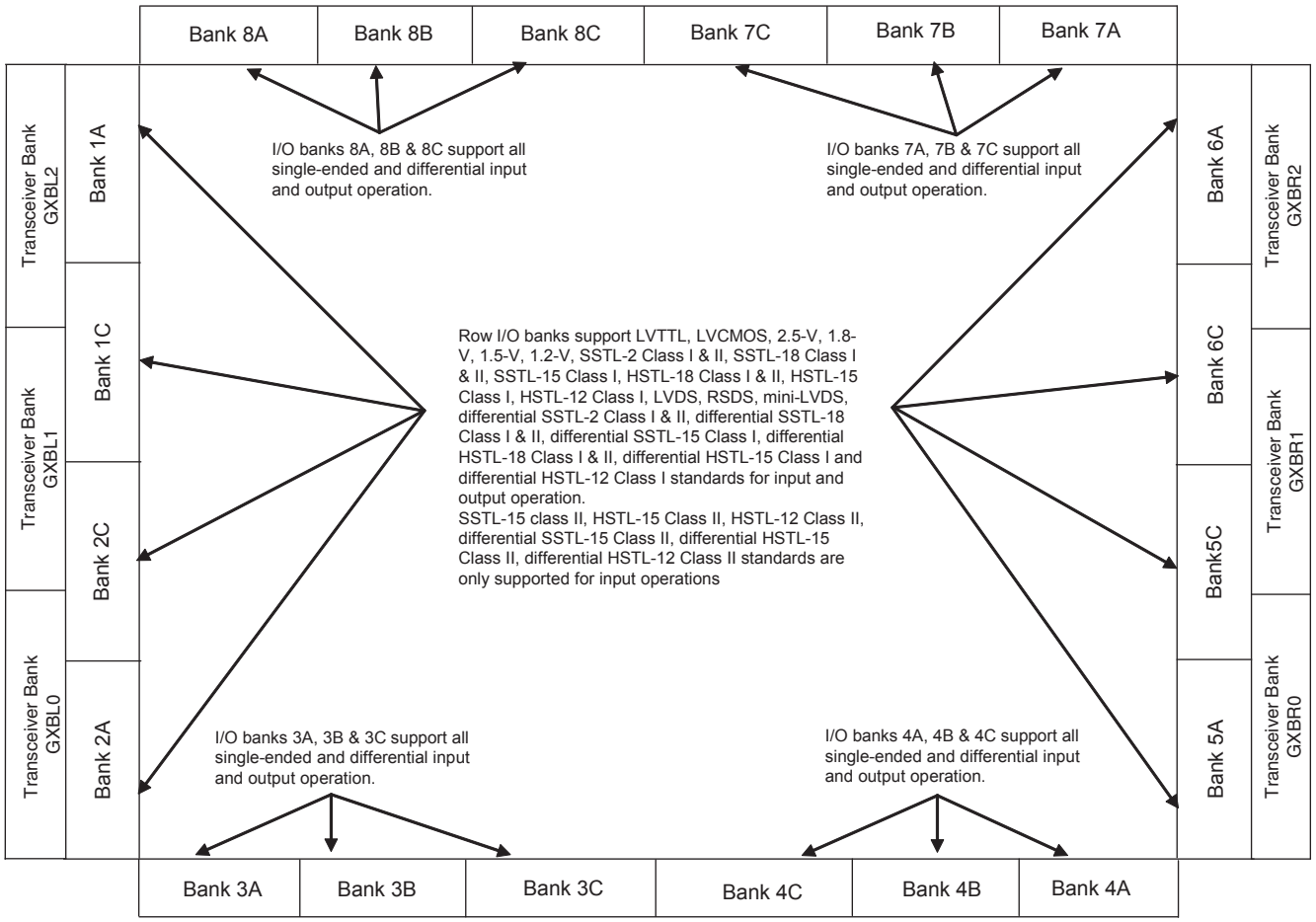
(5) Differential HSTL and SSTL outputs are not true differential outputs. They use two single-ended outputs with the second output programmed as inverted.

(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**

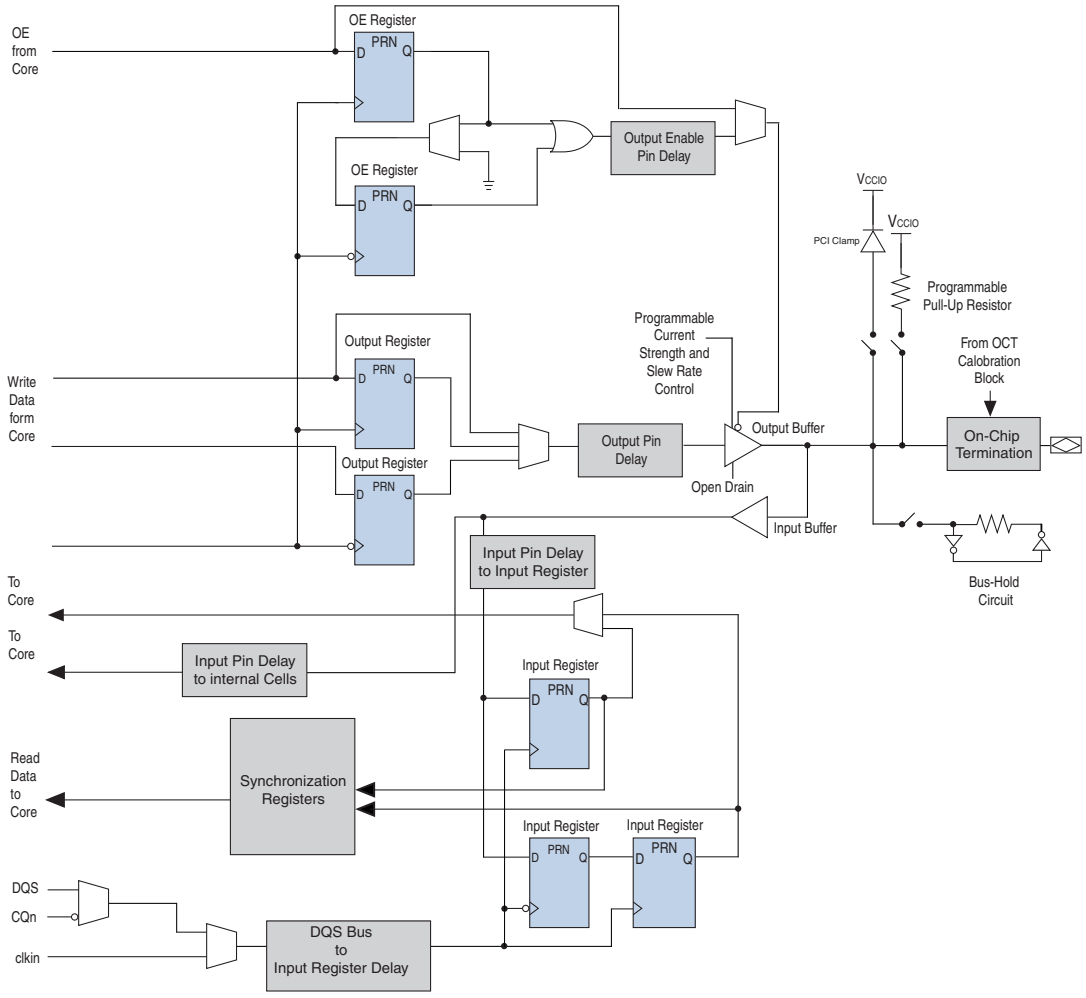
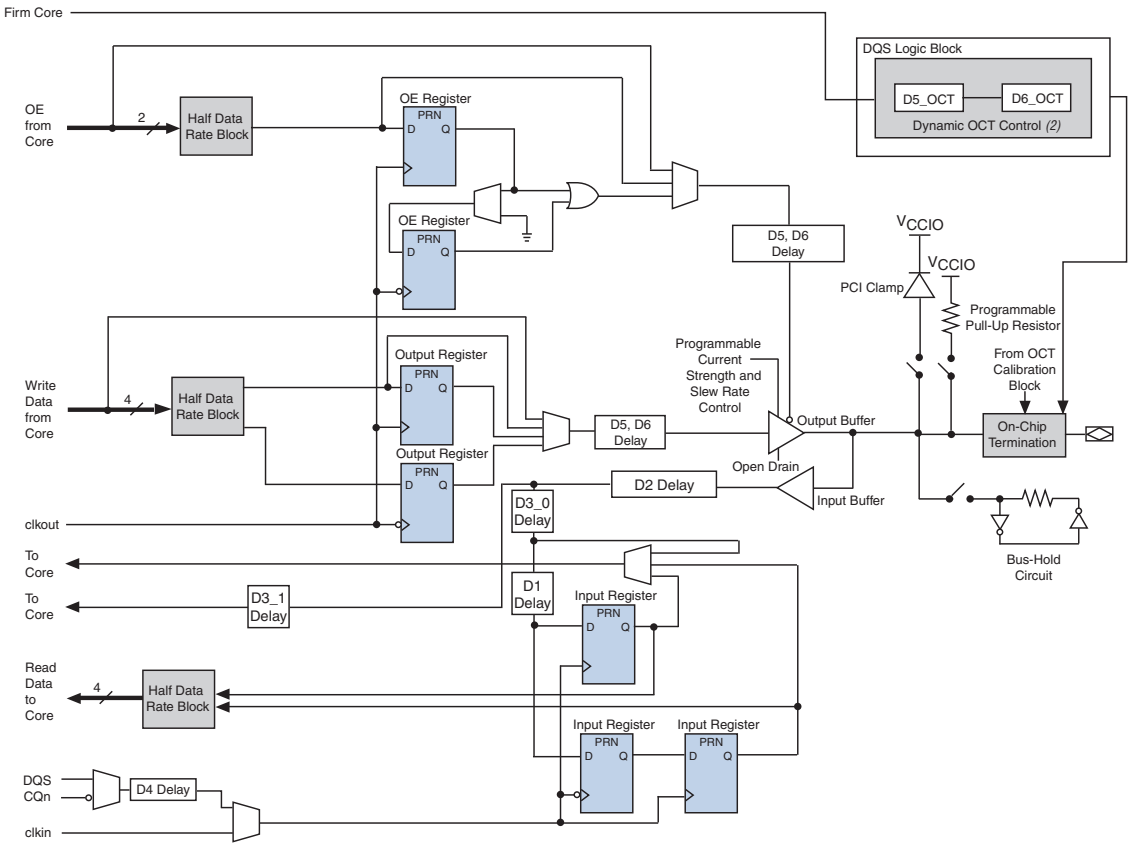


**Figure 6–4. IOE Structure for Arria II GZ Devices *(Note 1)*, *(2)***




**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:

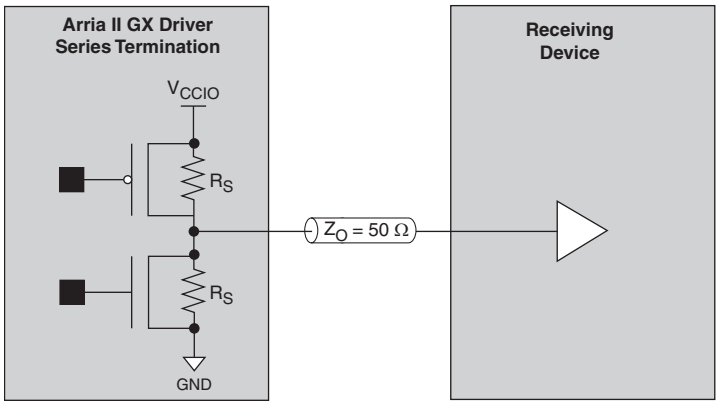
- The RUP pin is connected to $V_{CCIO}$ through an external 50-Ω ±1% resistor.
- The RDN pin is connected to GND through an external 50-Ω ±1% resistor.

## $R_S$ 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**

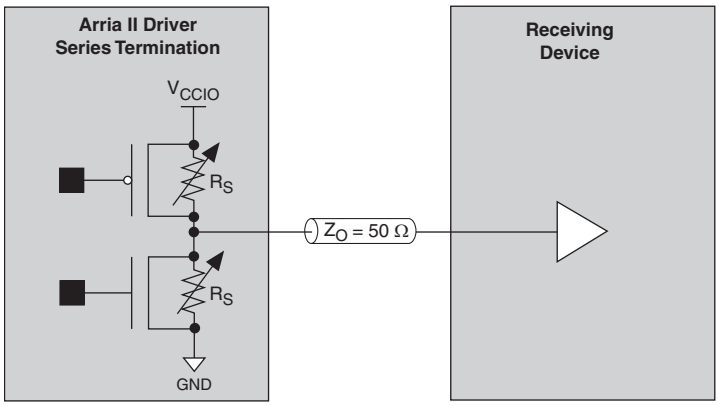


Table 6–11 lists the I/O standards that support $R_S$ OCT with and without calibration.

**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**

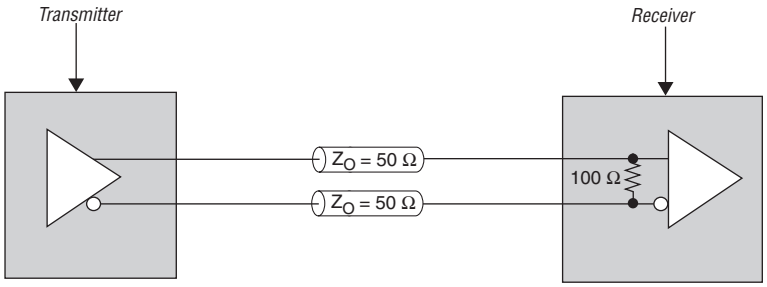
| I/O Standard | Expanded $R_S$ OCT Range | |
|---|---|---|
| | Row I/O (Ω) | Column I/O (Ω) |
| 3.3-V LVTTL/LVCMOS | 20–60 | 20–60 |
| 2.5-V LVTTL/LVCMOS | 20–60 | 20–60 |
| 1.8-V LVTTL/LVCMOS | 20–60 | 20–60 |
| 1.5-V LVTTL/LVCMOS | 40–60 | 20–60 |
| 1.2-V LVTTL/LVCMOS | 40–60 | 20–60 |
| SSTL-2 | 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**

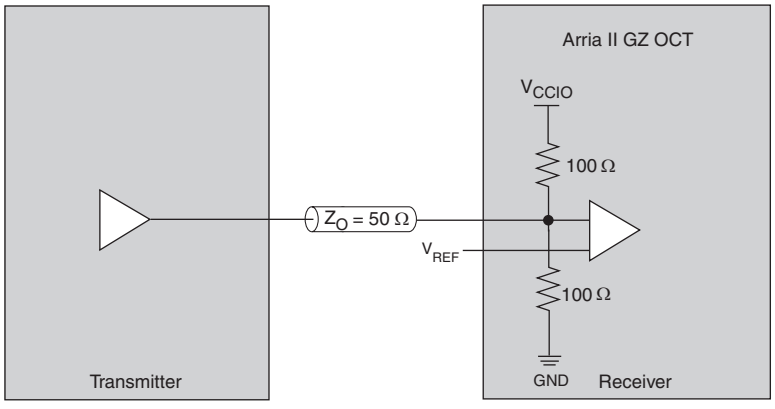



For more information about $R_D$ OCT, refer to the *High-Speed Differential I/O Interfaces and DPA in Arria II Devices* chapter.

## $R_T$ 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**

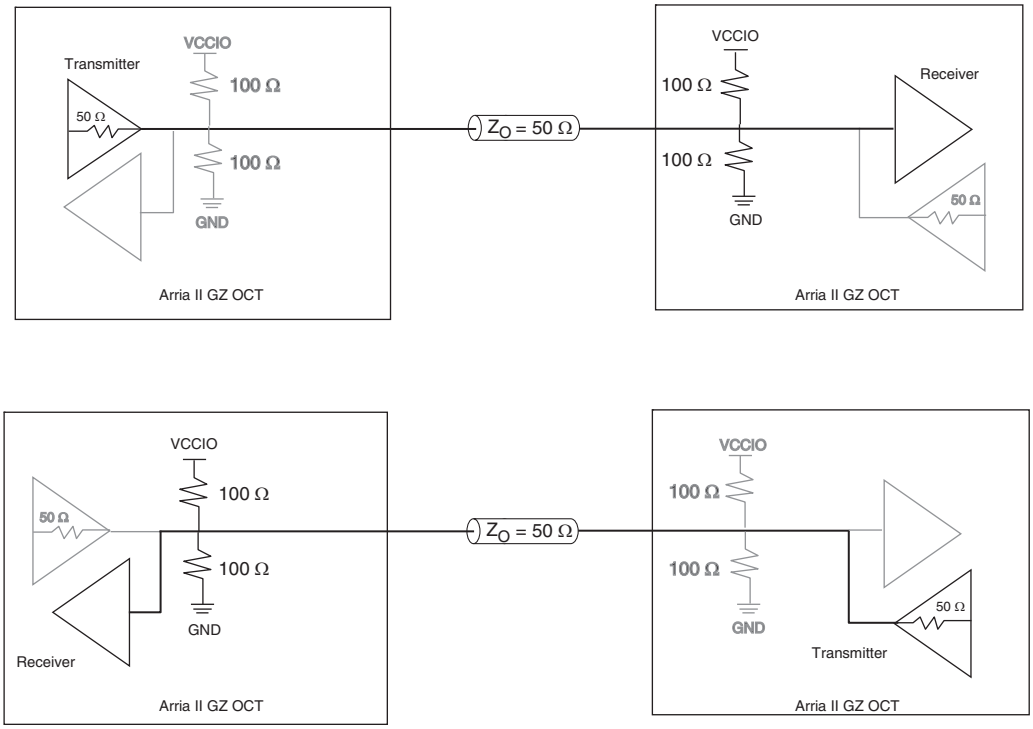
| I/O Standard | $R_T$ OCT Setting (Column I/O) (Ω) | $R_T$ OCT Setting (Row I/O) (Ω) |
|---|---|---|
| SSTL-2 Class I, II | 50 | 50 |
| SSTL-18 Class I, II | 50 | 50 |
| SSTL-15 Class I, II | 50 | 50 |
| HSTL-18 Class I, II | 50 | 50 |
| HSTL-15 Class I, II | 50 | 50 |
| HSTL-12 Class I, II | 50 | 50 |
| 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

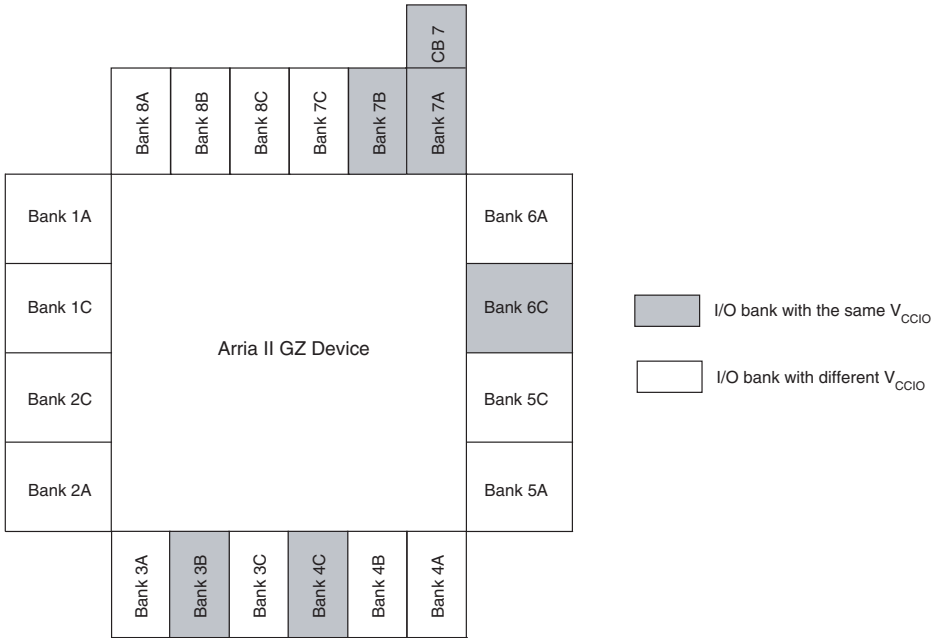
An OCT calibration block has the same $V_{CCIO}$ as the I/O bank that contains the block. $R_S$ OCT calibration is supported on all user I/O banks with different $V_{CCIO}$ voltage standards, up to the number of available OCT calibration blocks. You can configure I/O banks to receive calibrated codes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

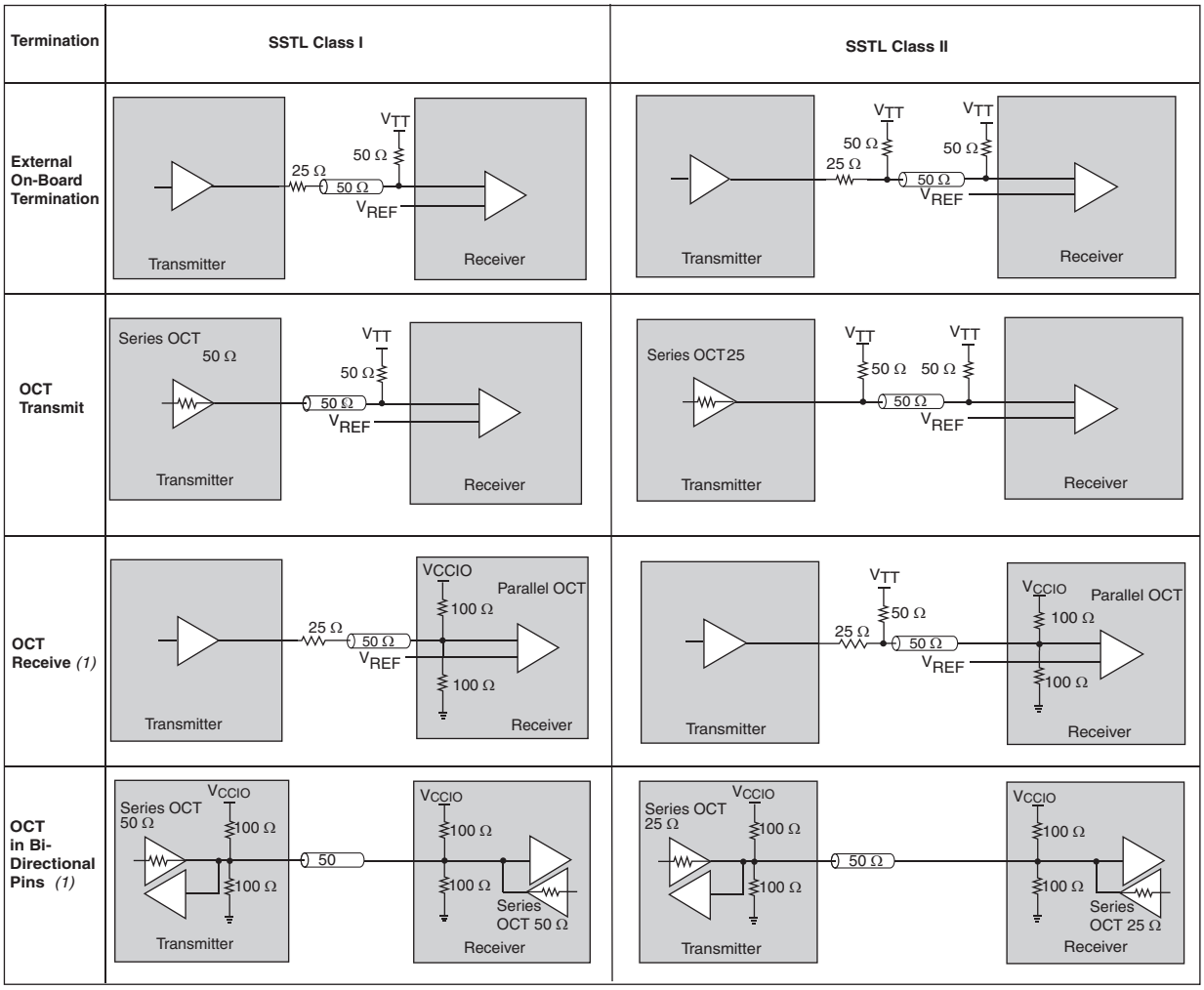
The following section describes the different termination schemes for I/O standards used in Arria II devices.

## 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**

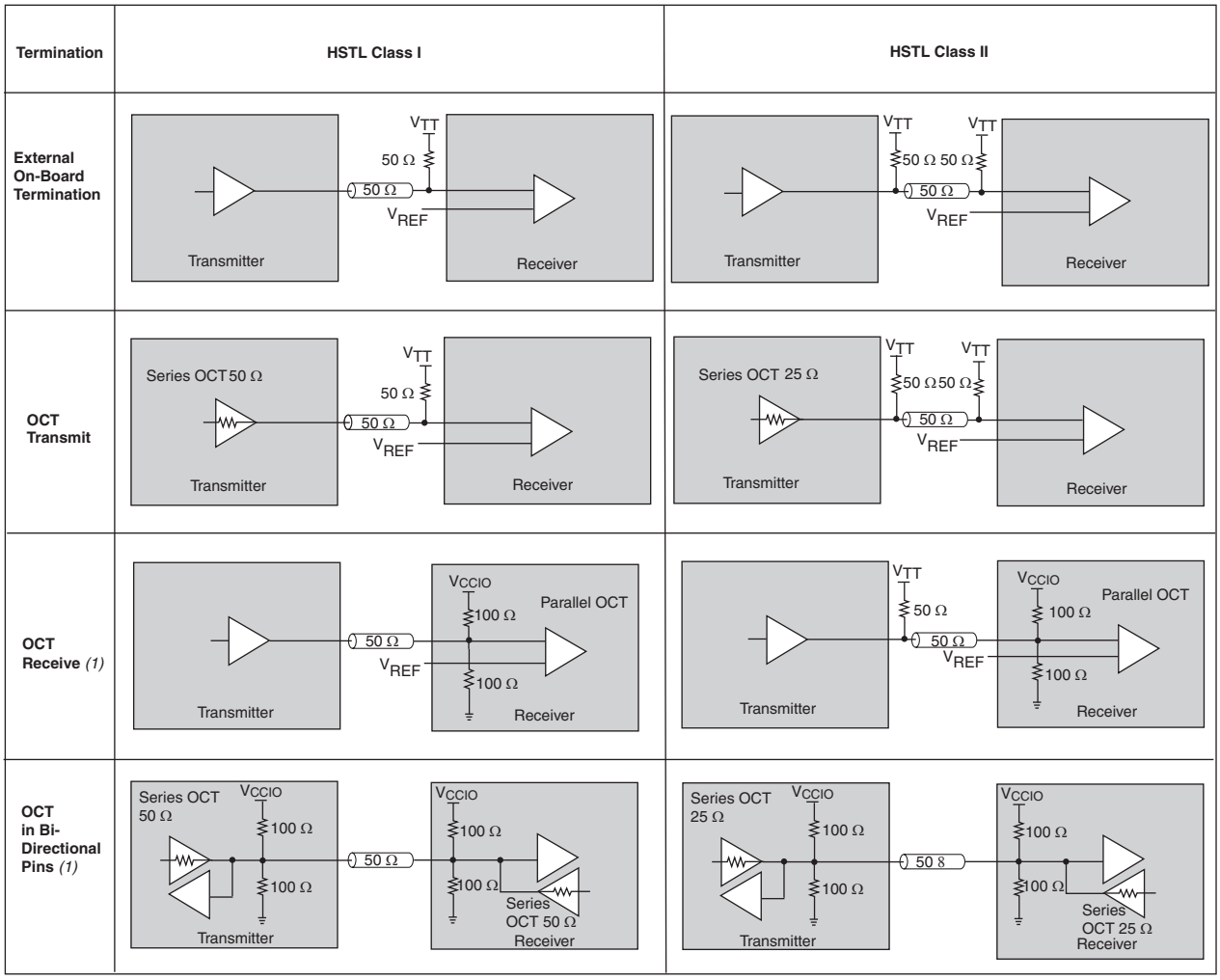



**Note to Figure 6–11:**

(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**

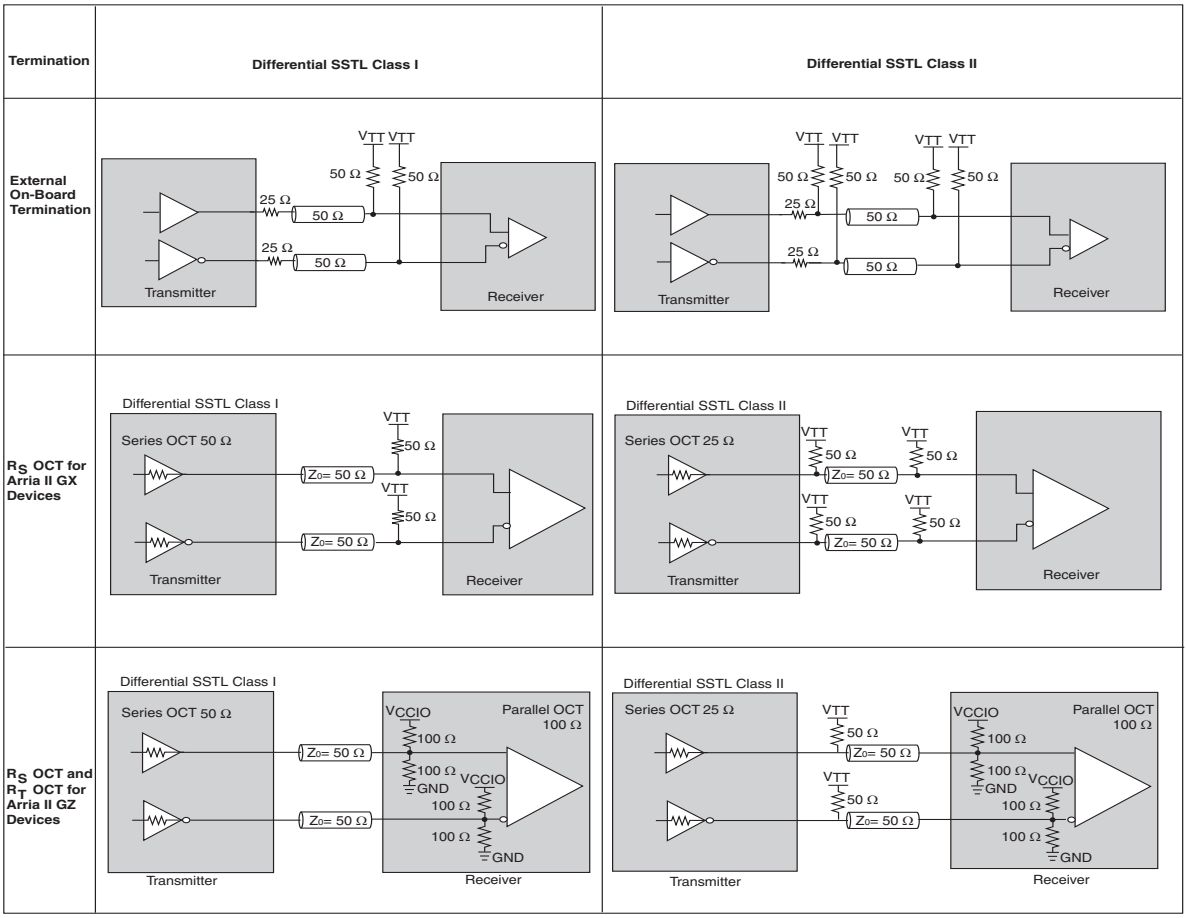
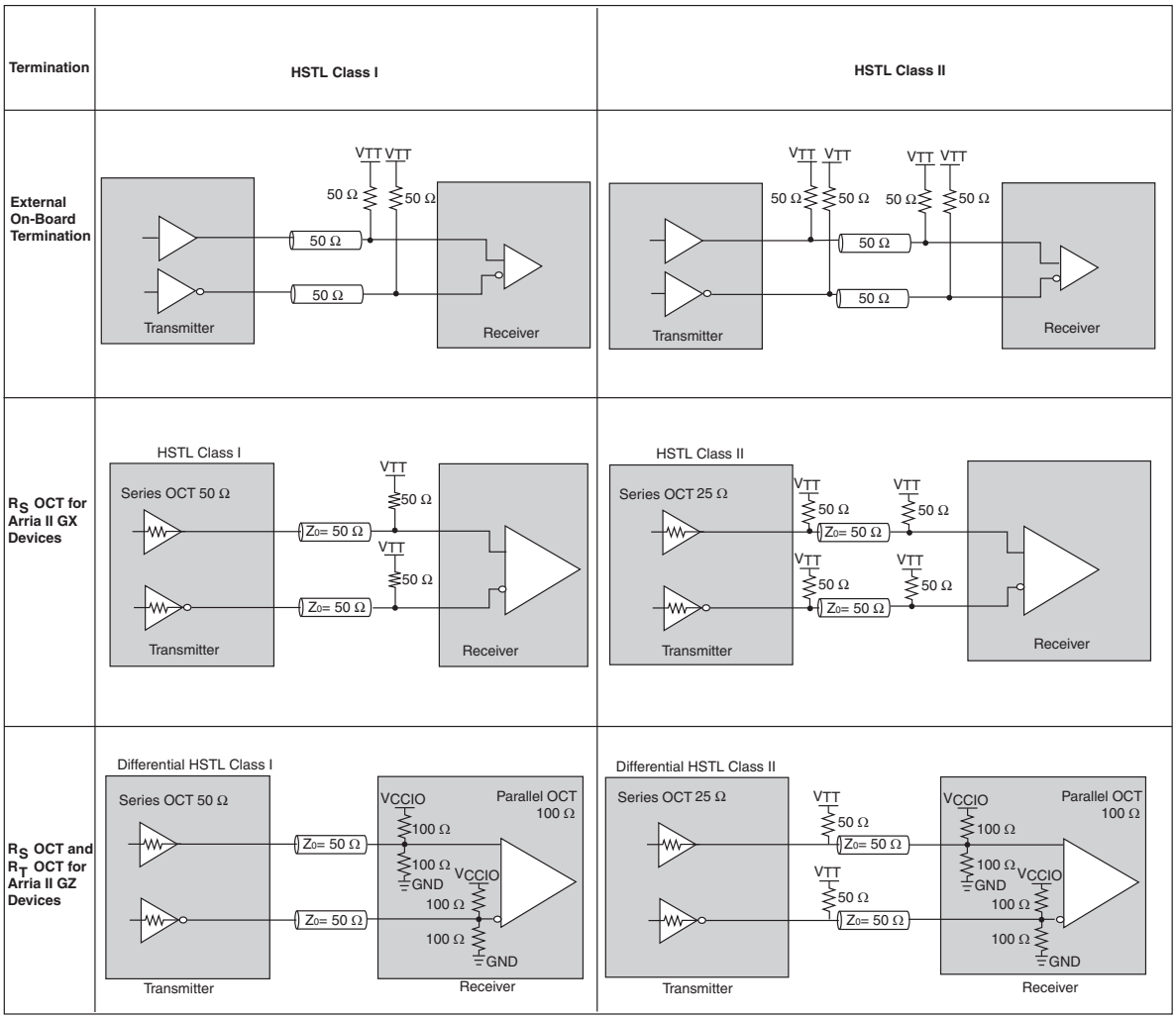


Figure 6–14 shows the details of differential HSTL I/O standard termination on Arria II devices.

**Figure 6–14. Differential HSTL I/O Standard Termination for Arria II Devices**

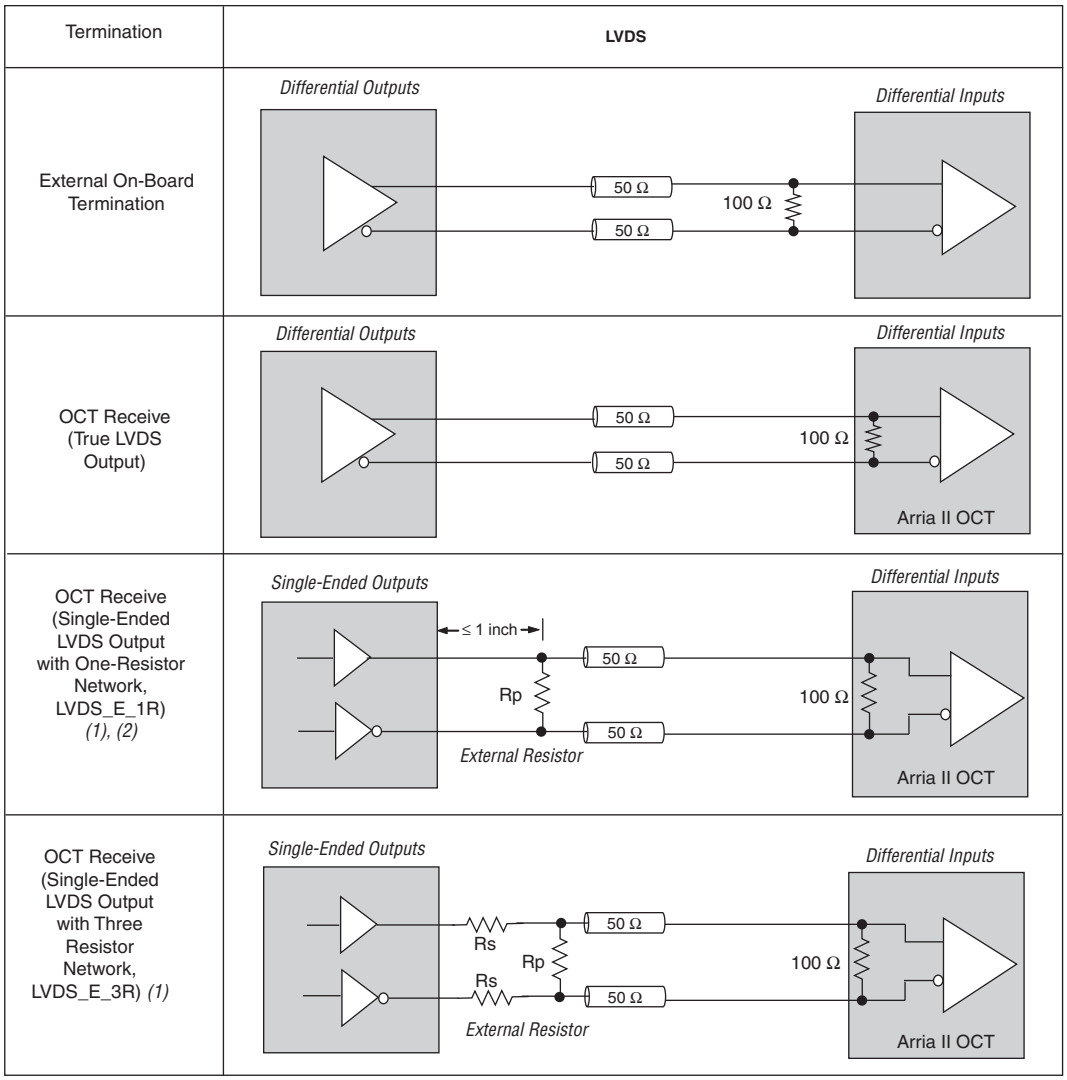


## LVDS

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)***



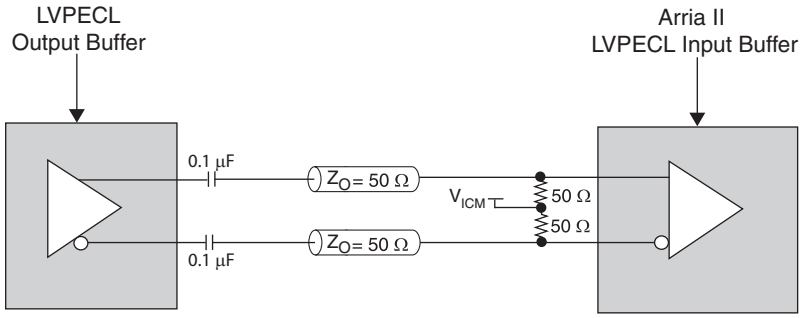

**Notes to Figure 6–15:**

(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

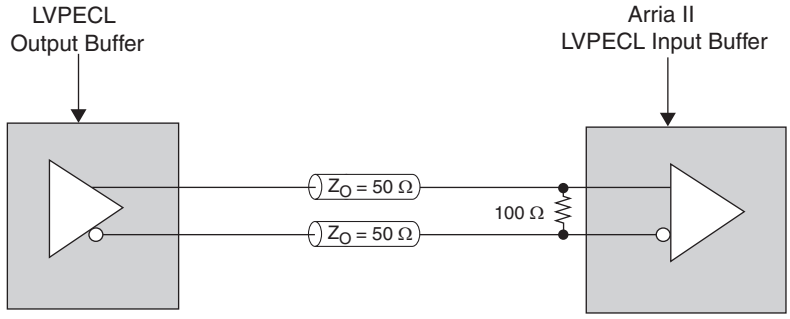
Arria II devices support the LVPECL I/O standard on input clock pins only. LVPECL output operation is not supported. LVDS input buffers are used to support LVPECL input operation. AC-coupling is required when the LVPECL common mode voltage of the output buffer is higher than Arria II LVPECL input common 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.

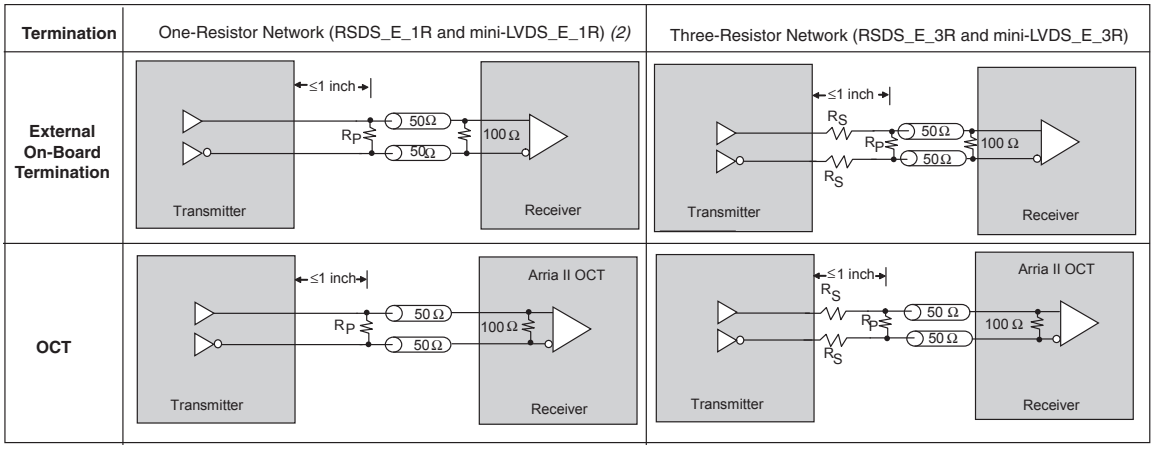
## mini-LVDS

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.<br>■ Minor text edits. |
| June 2011 | 4.1 | ■ Updated Table 6–9 and Table 6–10.<br>■ Updated Figure 6–3 and Figure 6–4.<br>■ 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:<br>■ Added Arria II GZ device information.<br>■ 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.<br>■ Added Figure 6–1. |
| July 2010 | 3.0 | Updated for Arria II GX v10.0 release:<br>■ Updated Table 6–4, Table 6–5, and Table 6–6.<br>■ Updated Figure 6–1.<br>■ Updated "Overview" section. |
| October 2009 | 2.0 | Updated for Arria II GX v9.1 release:<br>■ Updated Table 6–2 and Table 6–3.<br>■ Updated Figure 6–2, Figure 6–13, and Figure 6–14<br>■ Minor text edits. |
| June 2009 | 1.1 | ■ Updated Table 6–1, Table 6–4 and Table 6–5.<br>■ Updated "Programmable Slew Rate Control", "Programmable Differential Output Voltage", "Mini-LVDS", "RSDS", "OCT Calibration Block", and "I/O Placement Guidelines" sections.<br>■ 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:



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.

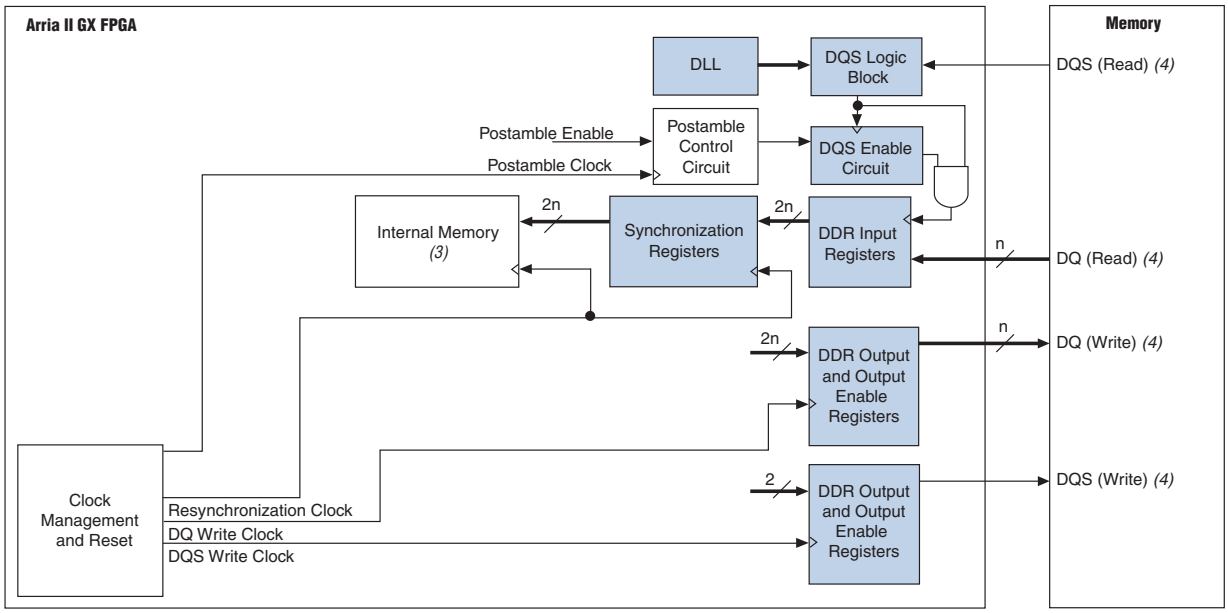
RLDRAM II is only available in Arria II GZ devices.

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*.

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:**

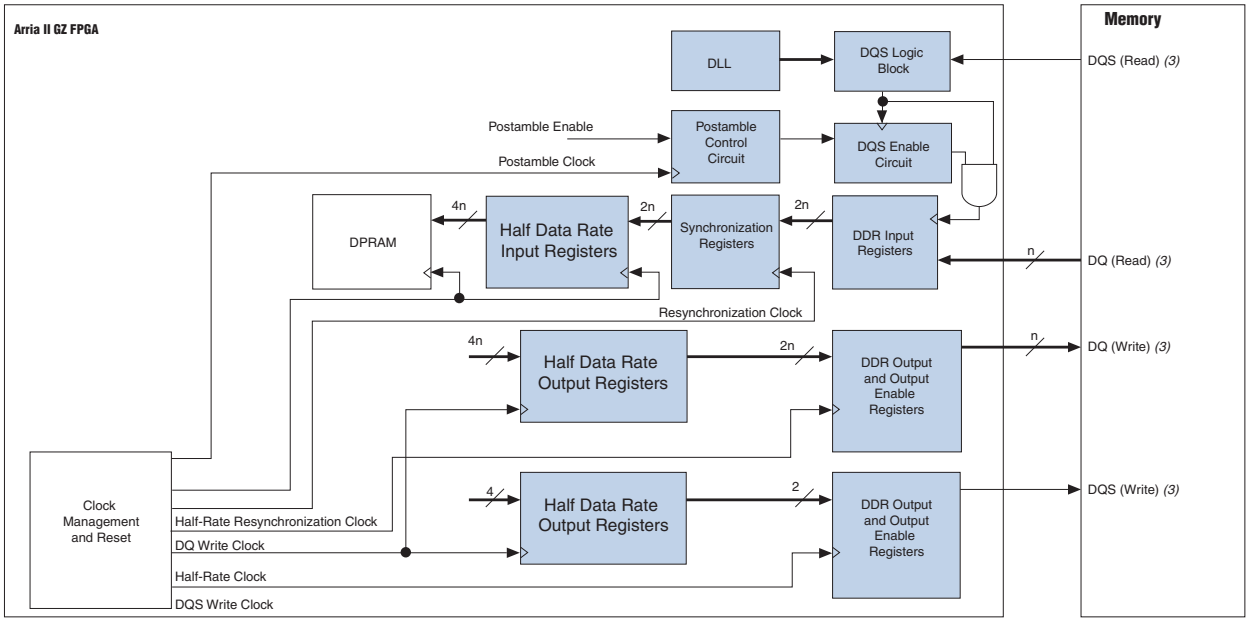
(1) You can bypass each register block.

(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.

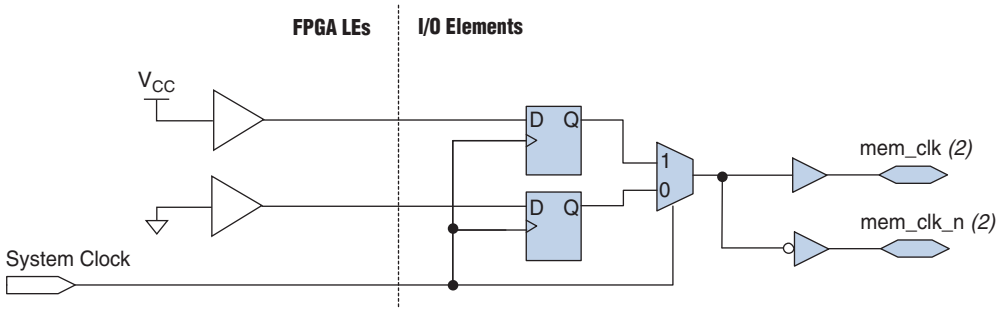
For more information about pin connections, refer to the *Arria II Device Family Pin Connection Guidelines*.

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.

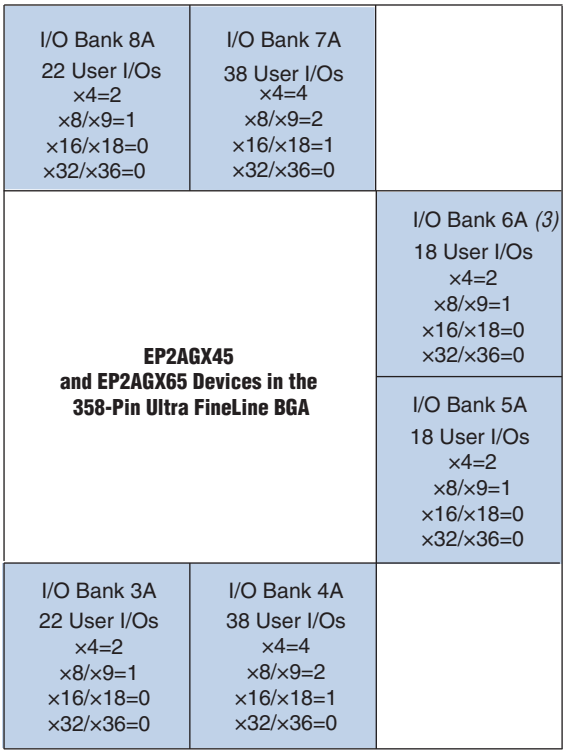
(2) To interface with a ×36 QDR II+/QDR II SRAM device in a Arria II GZ FPGA that does not support the ×32/×36 DQ/DQS group, refer to "Combining ×16/×18 DQ/DQS Groups for ×36 QDR II+/QDR II SRAM Interface" on page 7–21.

(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:**

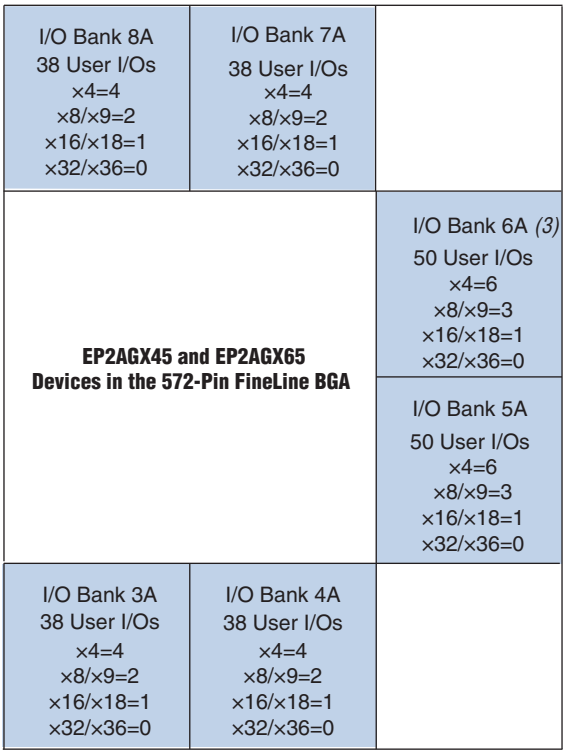
(1) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(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:**

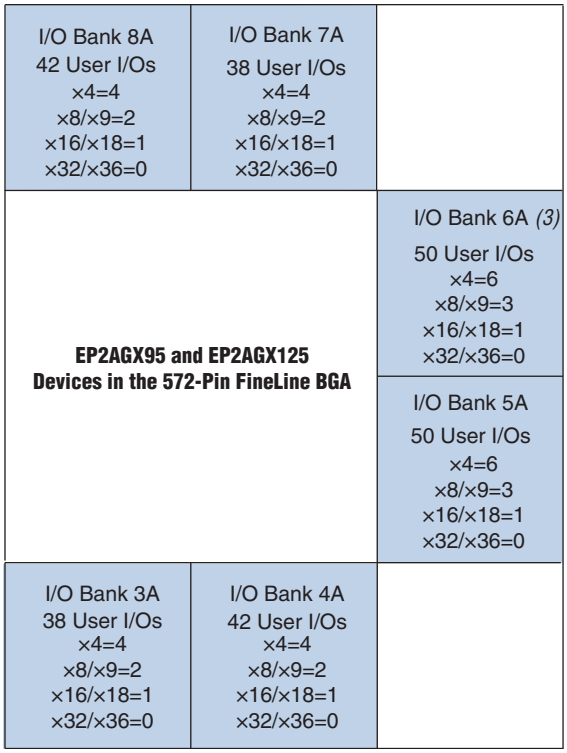
(1) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(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:**

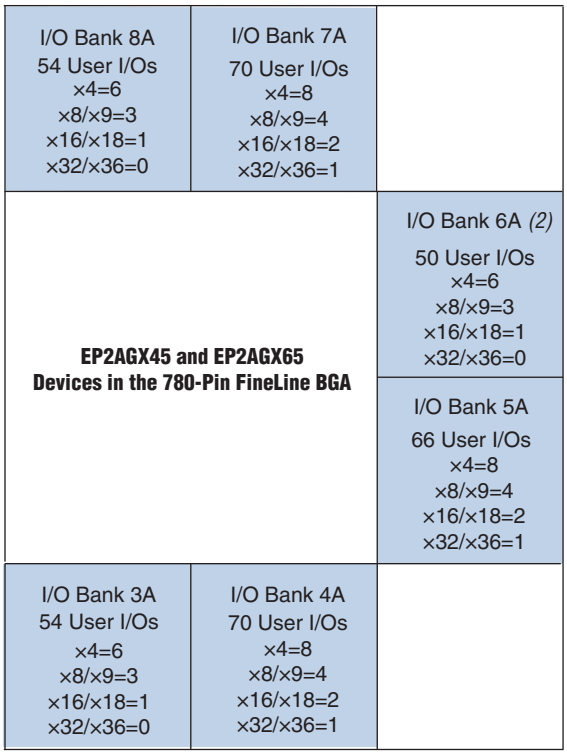
(1) All I/O pin counts include 12 dedicated clock inputs (`CLK4` to `CLK15`) that you can use for data inputs.

(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)***



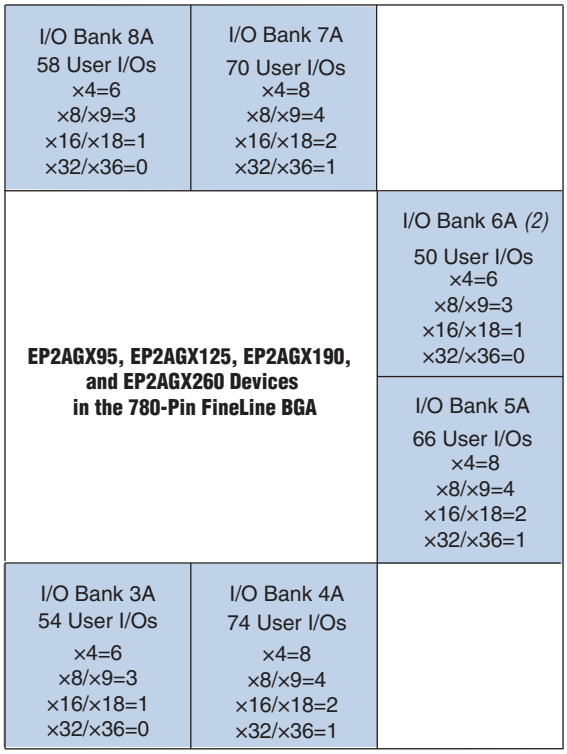

**Notes to Figure 7–7:**

(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)***



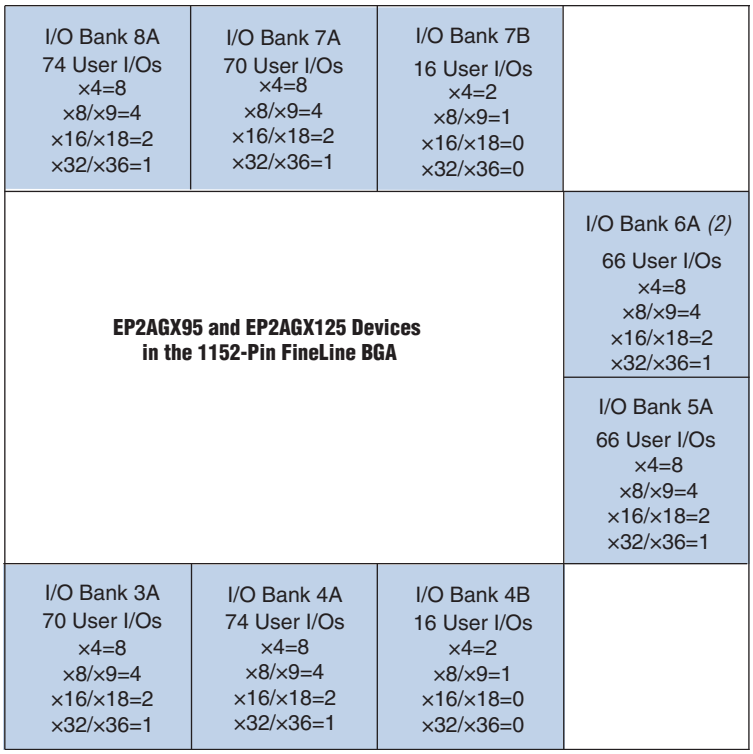

**Notes to Figure 7–8:**

(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)***



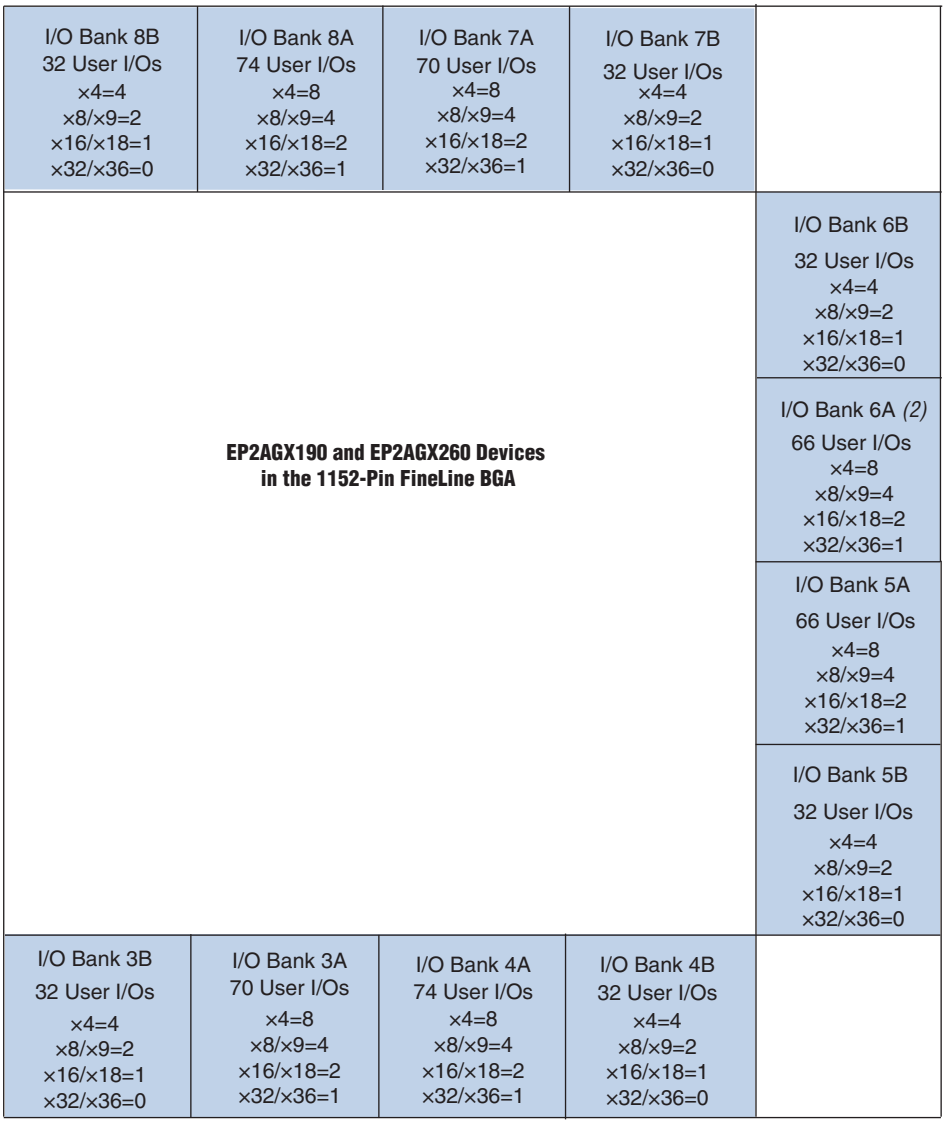

**Notes to Figure 7–9:**

(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)***



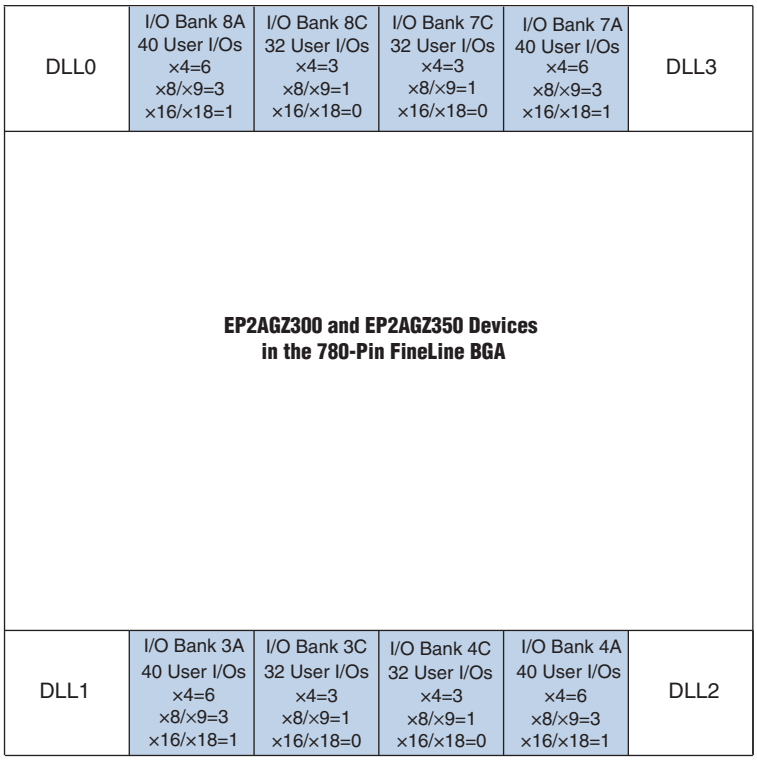

**Notes to Figure 7–10:**

(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)***

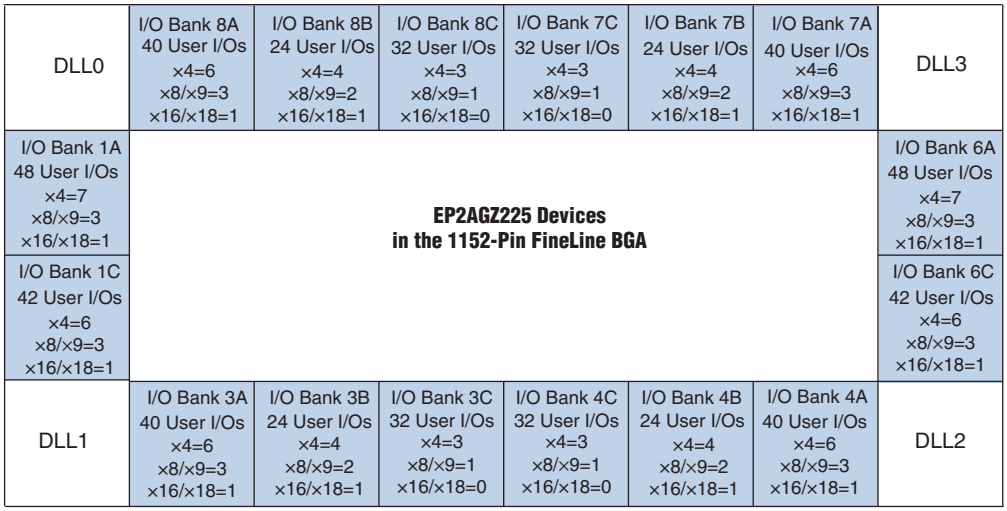



**Note to Figure 7–11:**

(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.

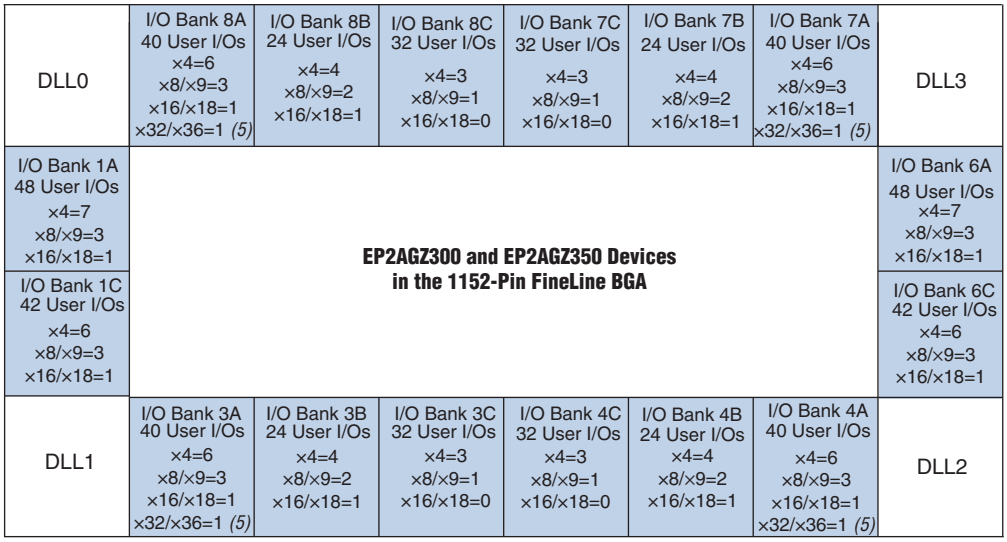
(2) You can also use DQS/DQSn pins in some of the ×4 groups as $R_{UP}$ and $R_{DN}$ pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as $R_{UP}$ and $R_{DN}$ pins for OCT calibration. If two pins of a ×4 group are used as $R_{UP}$ and $R_{DN}$ pins for OCT 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.

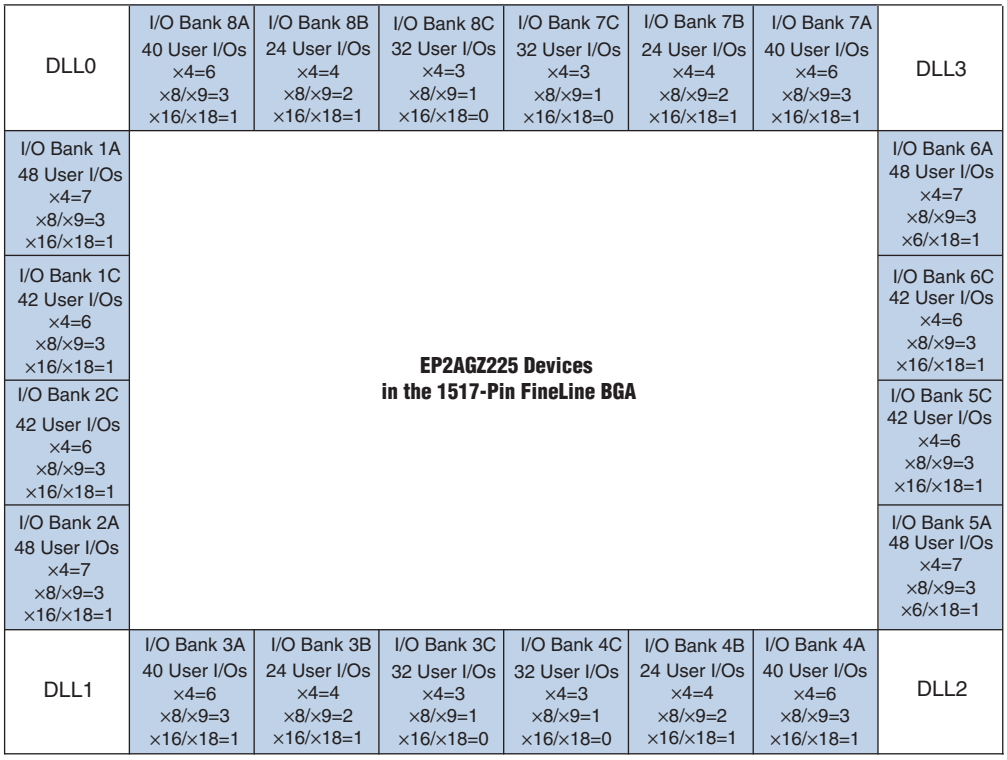
(2) All I/O pin counts include dedicated clock inputs that you can use for data inputs.

(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:**

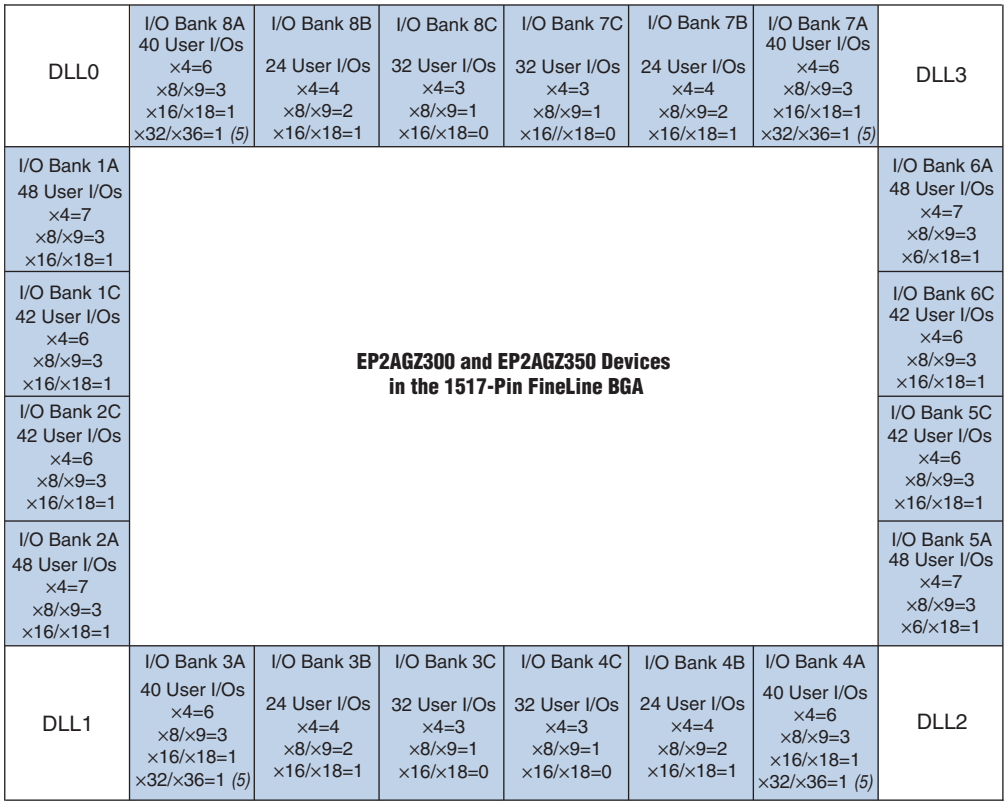
(1) EP2AGZ225 devices do not support ×32/×36 mode. To interface with a ×36 QDR II+/QDR II SRAM device, refer to "Combining ×16/×18 DQ/DQS Groups for ×36 QDR II+/QDR II SRAM Interface" on page 7–21.

(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.

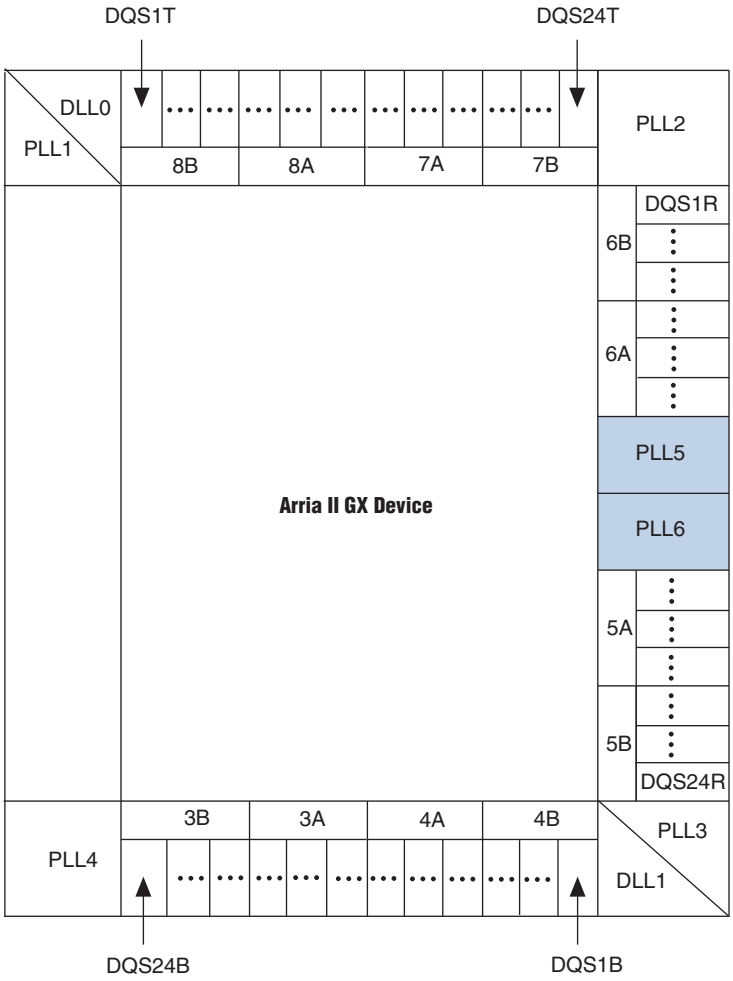
The DQS and DQSn pins are listed in the Arria II pin tables as `DQSXY` and `DQSnXY`, respectively, where `X` denotes the DQ/DQS grouping number and `Y` denotes whether the group is located on the top (T), bottom (B), left (L), or right (R) side of the device. The DQ/DQS pin numbering is based on ×4 mode.

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.

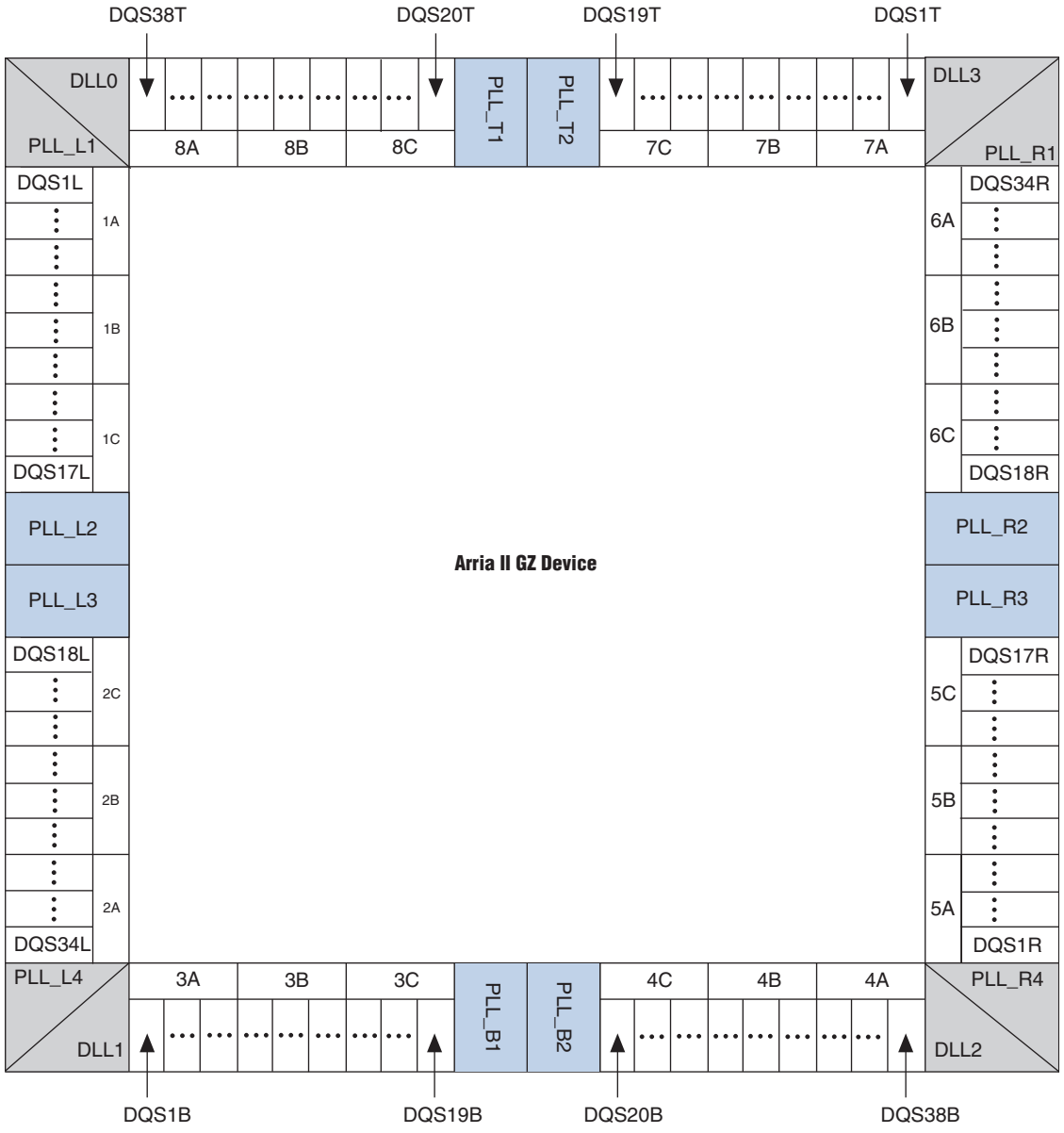
The numbering scheme starts from the top-left side of the device going clockwise in a die top view. Figure 7–16 shows how the DQ/DQS groups are numbered in a die top view of the largest Arria II GX device.

**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<br>■ EP2AGX65 | 4A and 7A (Top and Bottom I/O banks) *(1)* |
| | 572-Pin FineLine BGA | ■ EP2AGX45<br>■ EP2AGX65<br>■ EP2AGX95<br>■ EP2AGX125 | 7A and 8A (Top I/O banks)<br>5A and 6A (Right I/O banks)<br>3A and 4A (Bottom I/O banks) |
| | 780-Pin FineLine BGA *(2)* | ■ EP2AGX45<br>■ EP2AGX65<br>■ EP2AGX95<br>■ EP2AGX125<br>■ EP2AGX190<br>■ EP2AGX260 | 7A and 8A (Top I/O banks)<br>5A and 6A (Right I/O banks)<br>3A and 4A (Bottom I/O banks) |
| | 1152-Pin FineLine BGA *(2)* | ■ EP2AGX95<br>■ EP2AGX125 | 7A and 8A (Top I/O banks)<br>5A and 6A (Right I/O banks)<br>3A and 4A (Bottom I/O banks) |
| | | ■ EP2AGX190<br>■ EP2AGX260 | Combine any two banks from each side of I/O banks |
| Arria II GZ | 780-Pin FineLine BGA | ■ EP2AGZ300<br>■ EP2AGZ350 | 3A and 4A, 7A and 8A (bottom and top I/O banks) *(3)* |
| | 1152-Pin FineLine BGA | ■ EP2AGZ225<br>■ EP2AGZ300 *(4)*<br>■ EP2AGZ350 *(4)* | 1A and 1C, 6A and 6C (left and right I/O banks)<br>3A and 3B, 4A and 4B (bottom I/O banks)<br>7A and 7B, 8A and 8B (top I/O banks) |
| | 1517-Pin FineLine BGA | ■ EP2AGZ225<br>■ EP2AGZ300 *(4)*<br>■ EP2AGZ350 *(4)* | 1A and 1C, 2A and 2C (left I/O banks)<br>3A and 3B, 4A and 4B (bottom I/O banks)<br>5A and 5C, 6A and 6C (right I/O banks)<br>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.

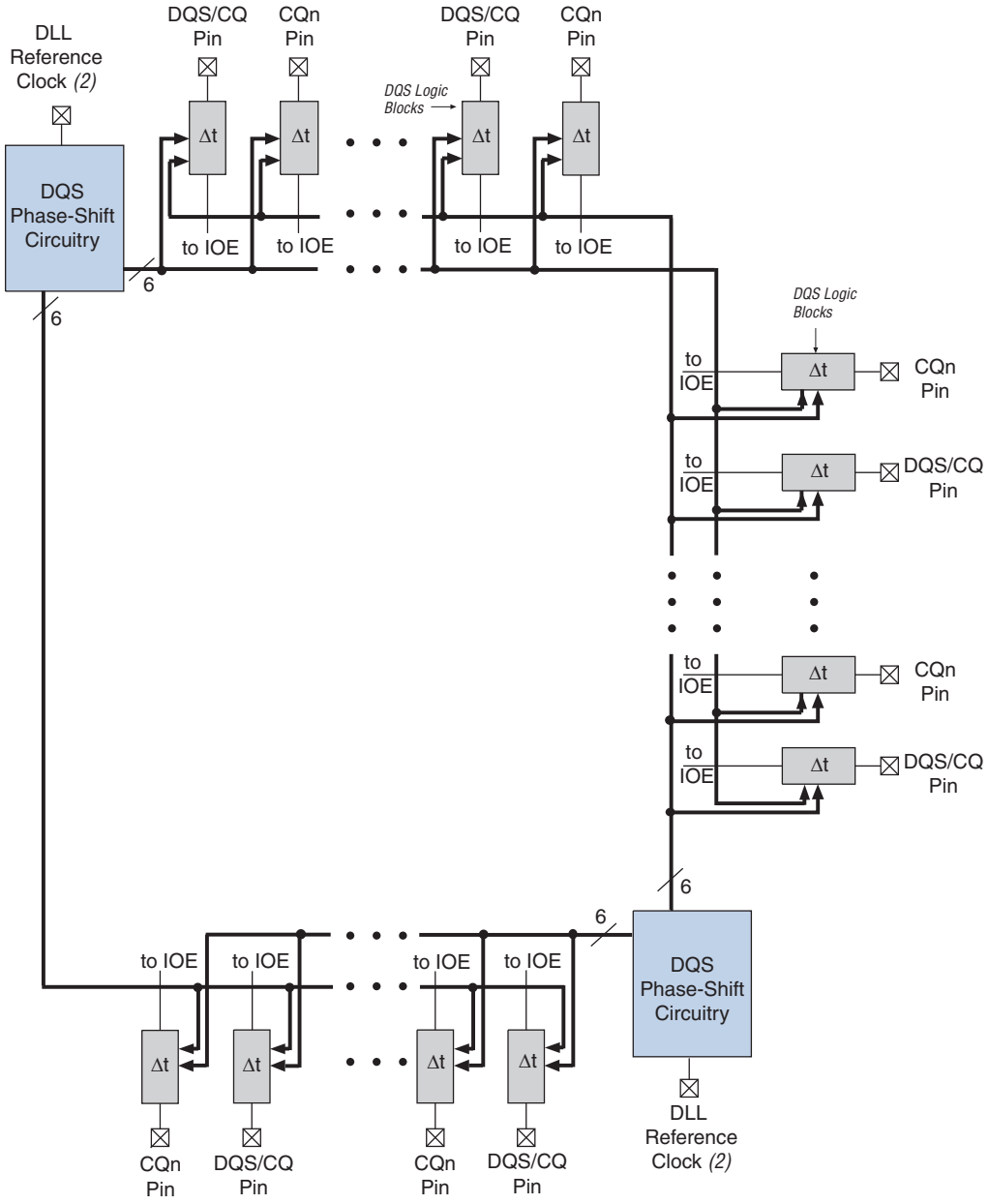
For more information about supported external memory IPs, refer to *Section III: External Memory Interface System Specification* in volume 1 of the *External Memory Handbook*.

## 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)***

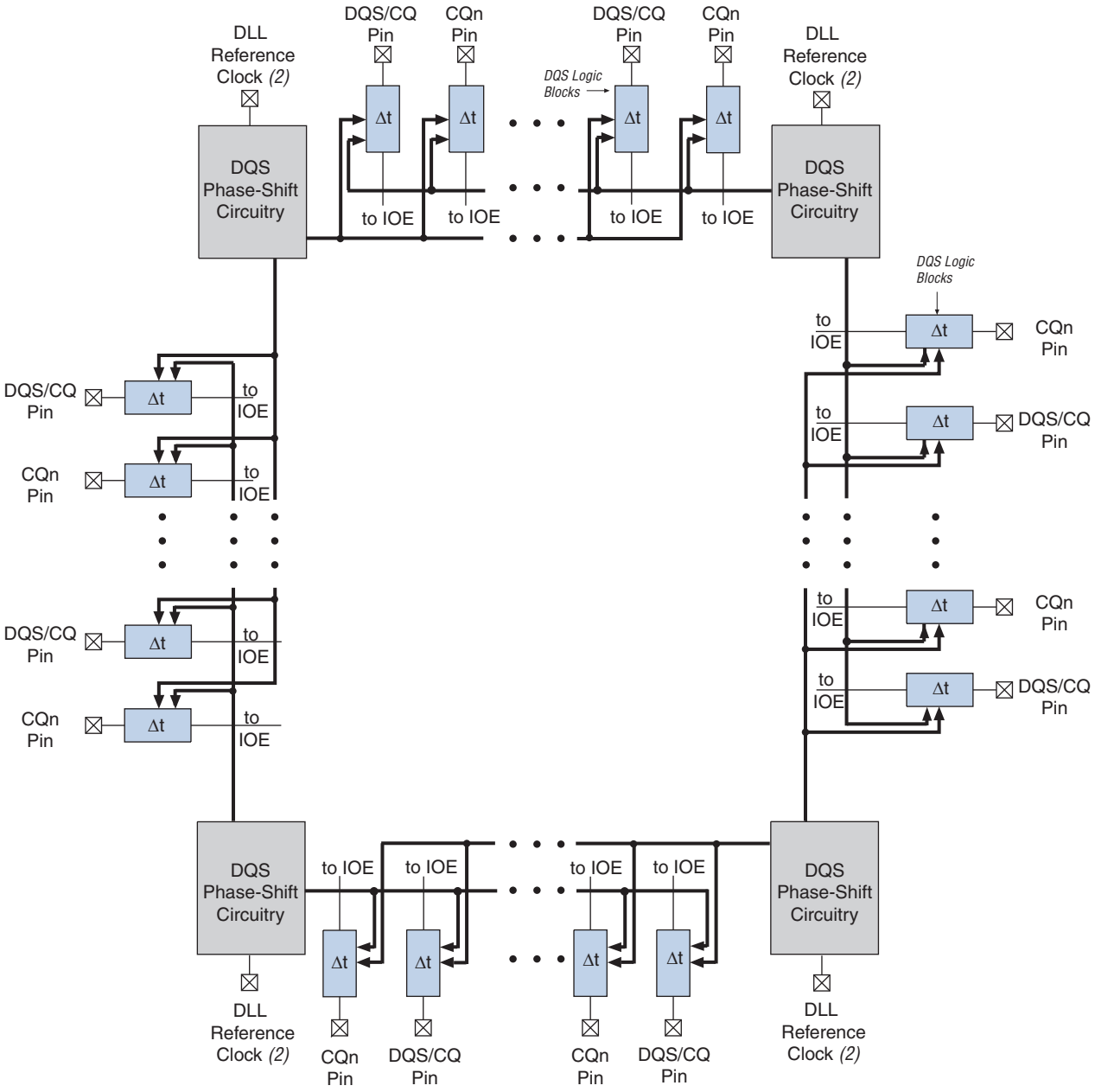



**Notes to Figure 7–18:**

(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<br>CLK13<br>CLK14<br>CLK15 | — | PLL1 |
| DLL1 | CLK4<br>CLK5<br>CLK6<br>CLK7 | CLK8<br>CLK9<br>CLK10<br>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<br>CLK13P<br>CLK14P<br>CLK15P | — | PLL_T1 | — | — |
| DLL1 | CLK4P<br>CLK5P<br>CLK6P<br>CLK7P | — | PLL_B1 | — | — |
| DLL2 | CLK4P<br>CLK5P<br>CLK6P<br>CLK7P | — | PLL_B2 | — | — |
| DLL3 | CLK12P<br>CLK13P<br>CLK14P<br>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<br>CLK13P<br>CLK14P<br>CLK15P | CLK0P<br>CLK1P | PLL_T1 | PLL_L2 | — |
| DLL1 | CLK4P<br>CLK5P<br>CLK6P<br>CLK7P | CLK0P<br>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<br>CLK5P<br>CLK6P<br>CLK7P | CLK10P<br>CLK11P | PLL_B2 | — | — |
| DLL3 | CLK12P<br>CLK13P<br>CLK14P<br>CLK15P | CLK10P<br>CLK11P | PLL_T2 | PLL_R2 | — |

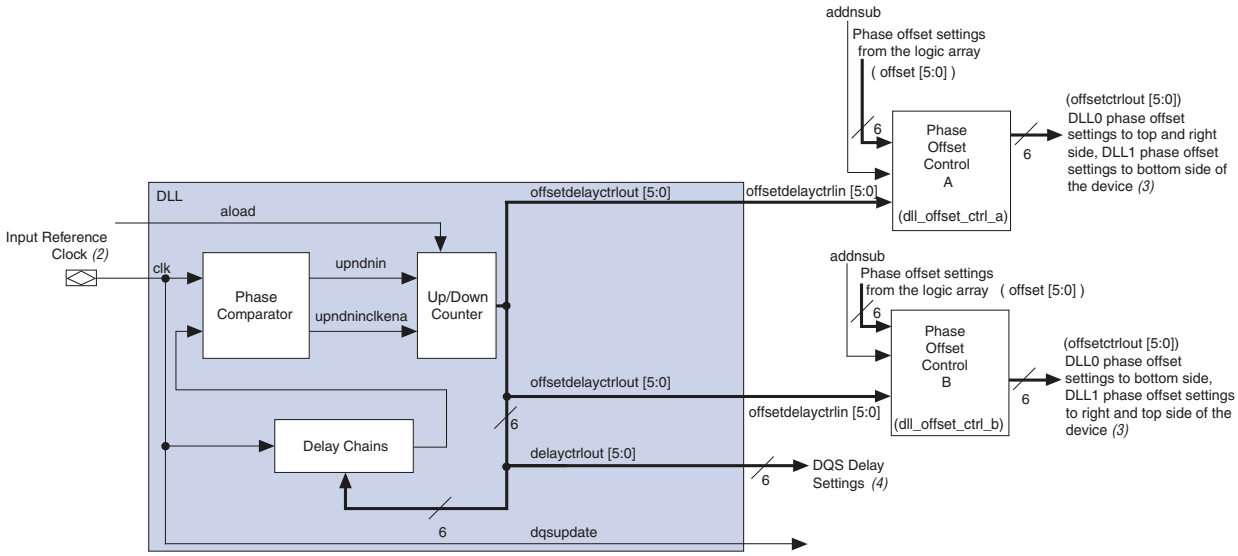
**Table 7–9. DLL Reference Clock Input for EP2AGZ225, EP2AGZ300, and EP2AGZ350 Devices in the 1517-Pin FineLine BGA Package**

| DLL | CLKIN (Top/Bottom) | CLKIN (Left/Right) | PLL (Top/Bottom) | PLL (Left/Right) | PLL (Corner) |
|---|---|---|---|---|---|
| DLL0 | CLK12P<br>CLK13P<br>CLK14P<br>CLK15P | CLK0P<br>CLK1P<br>CLK2P<br>CLK3P | PLL_T1 | PLL_L2 | — |
| DLL1 | CLK4P<br>CLK5P<br>CLK6P<br>CLK7P | CLK0P<br>CLK1P<br>CLK2P<br>CLK3P | PLL_B1 | PLL_L3 | — |
| DLL2 | CLK4P<br>CLK5P<br>CLK6P<br>CLK7P | CLK8P<br>CLK9P<br>CLK10P<br>CLK11P | PLL_B2 | PLL_R3 | — |
| DLL3 | CLK12P<br>CLK13P<br>CLK14P<br>CLK15P | CLK8P<br>CLK9P<br>CLK10P<br>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.

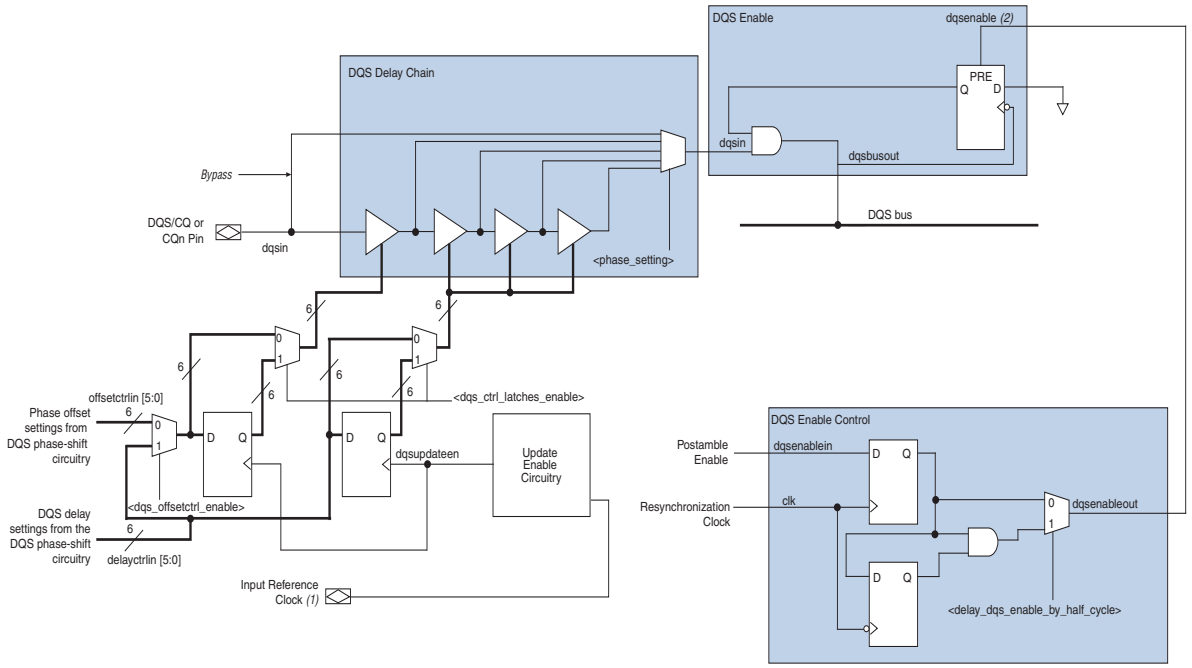
For more information about the value for each step, refer to the *Device Datasheet for Arria II Devices*.

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.

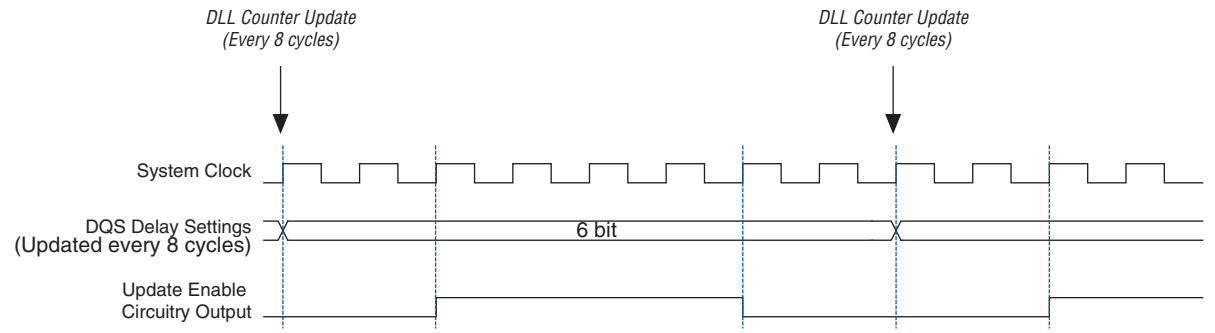
The delay elements in the DQS logic block have the same characteristics as the delay elements in the DLL. When the DLL is not used to control the DQS delay chains, you can input your own Gray-coded 6-bit or 5-bit settings with the `dqs_delayctrlin[5..0]` signals available in the ALTMEMPHY megafunction 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.

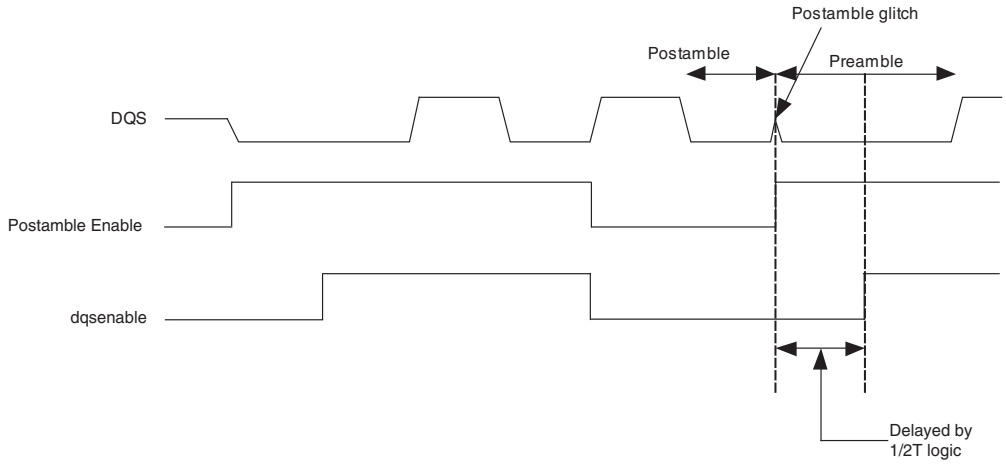
Arria II devices have dedicated postamble registers that you can control to ground the shifted DQS signal used to clock the DQ input registers at the end of a read operation. This ensures that any glitches on the DQS input signals at the end of the read postamble time do not affect the DQ IOE registers.

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**

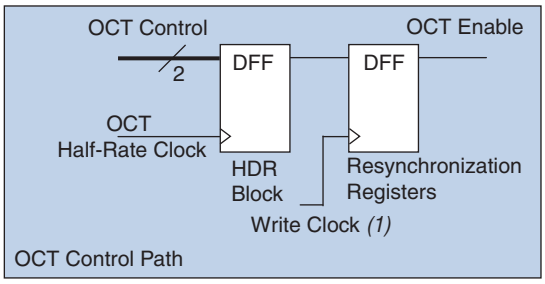


## Arria II GZ Dynamic On-Chip Termination Control

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:**

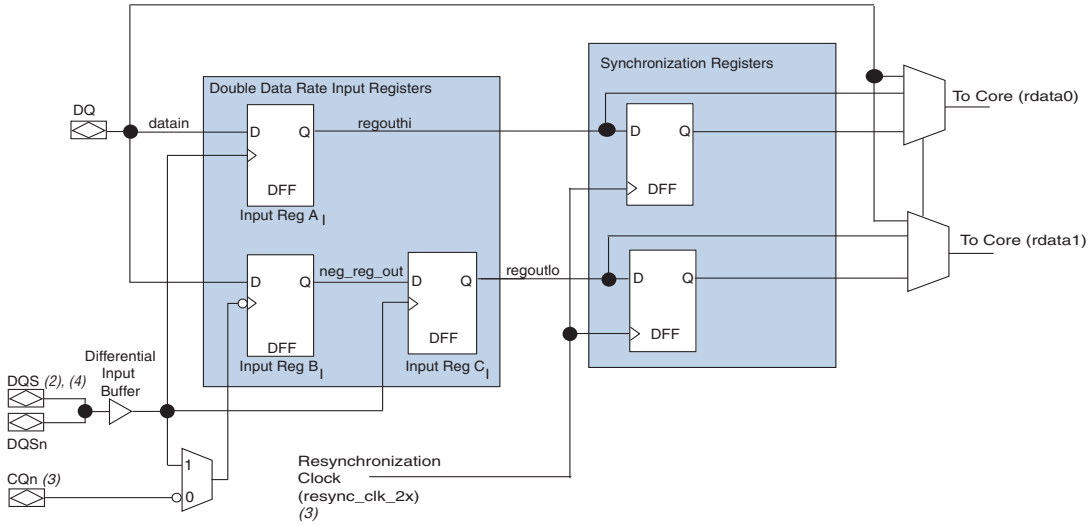
(1) The write clock comes from the PLL.

## 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.

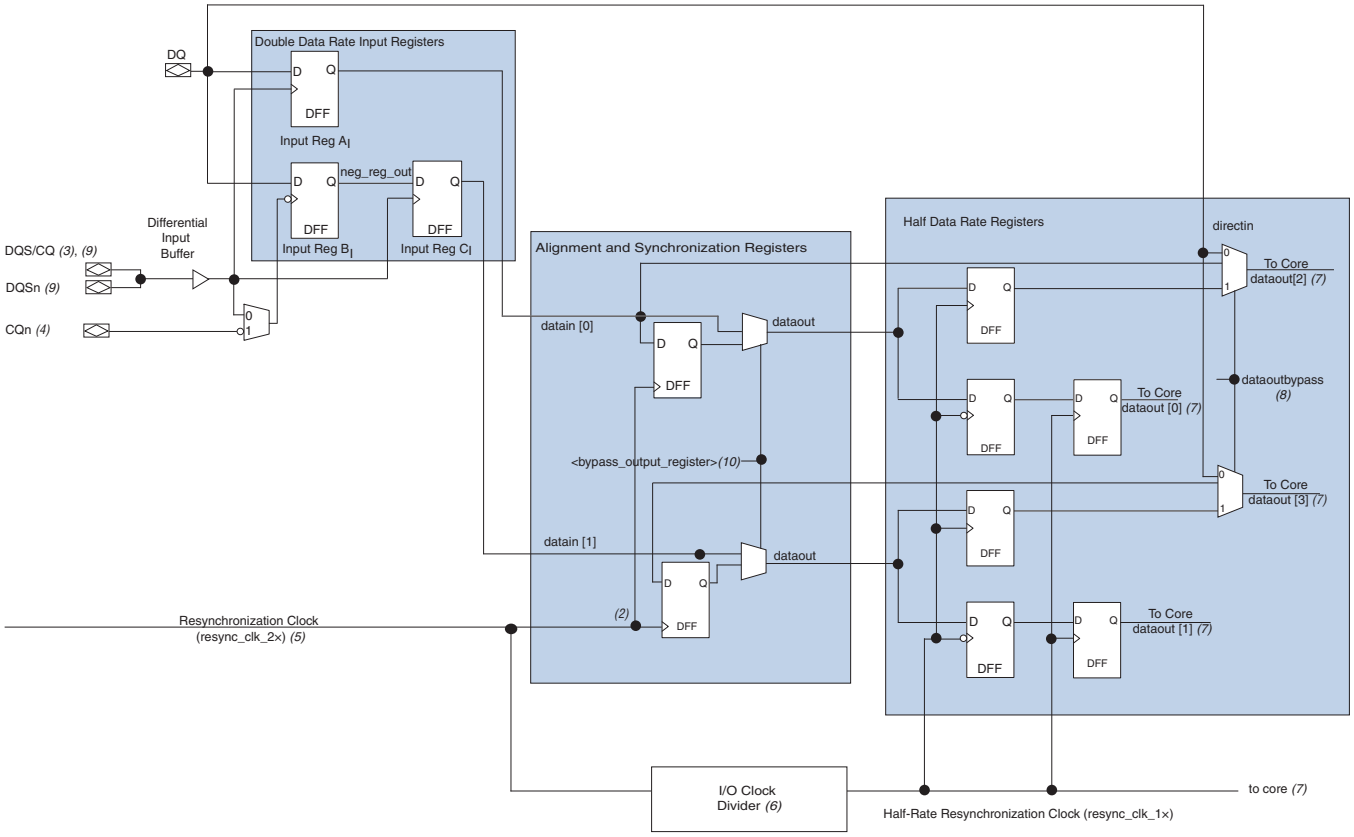
(2) The input clock can be from the DQS logic block (whether the postamble circuitry is bypassed or not) or from a global clock line.

(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`.

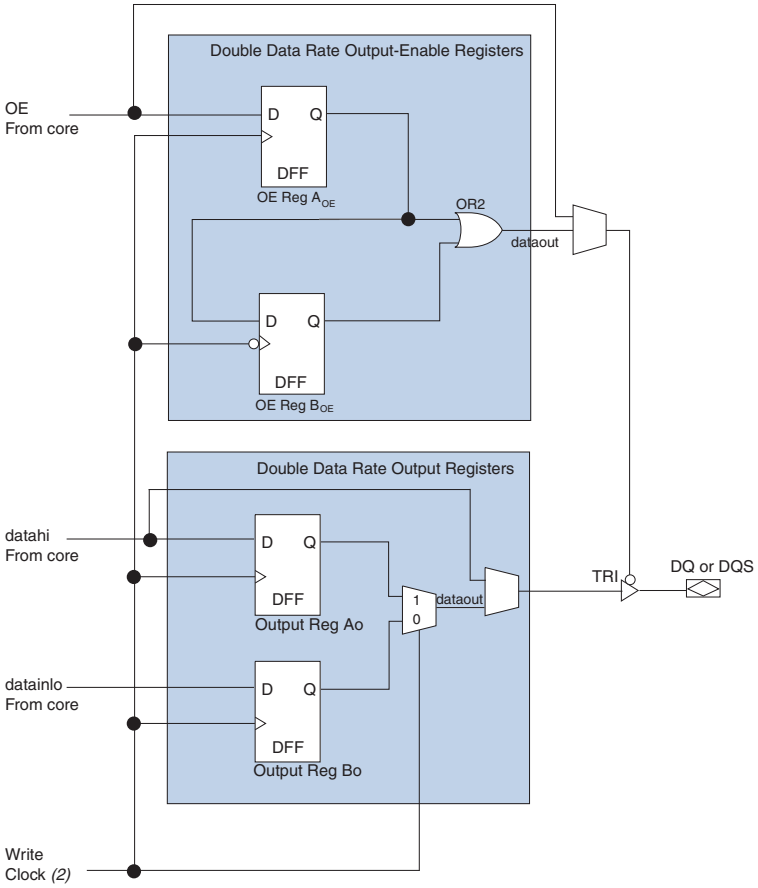
There are three registers in the DDR input registers block. Two registers capture data on the positive and negative edges of the clock, and the third register aligns the captured data. You can choose to use the same clock for the positive edge and negative edge registers, or two complementary clocks (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.

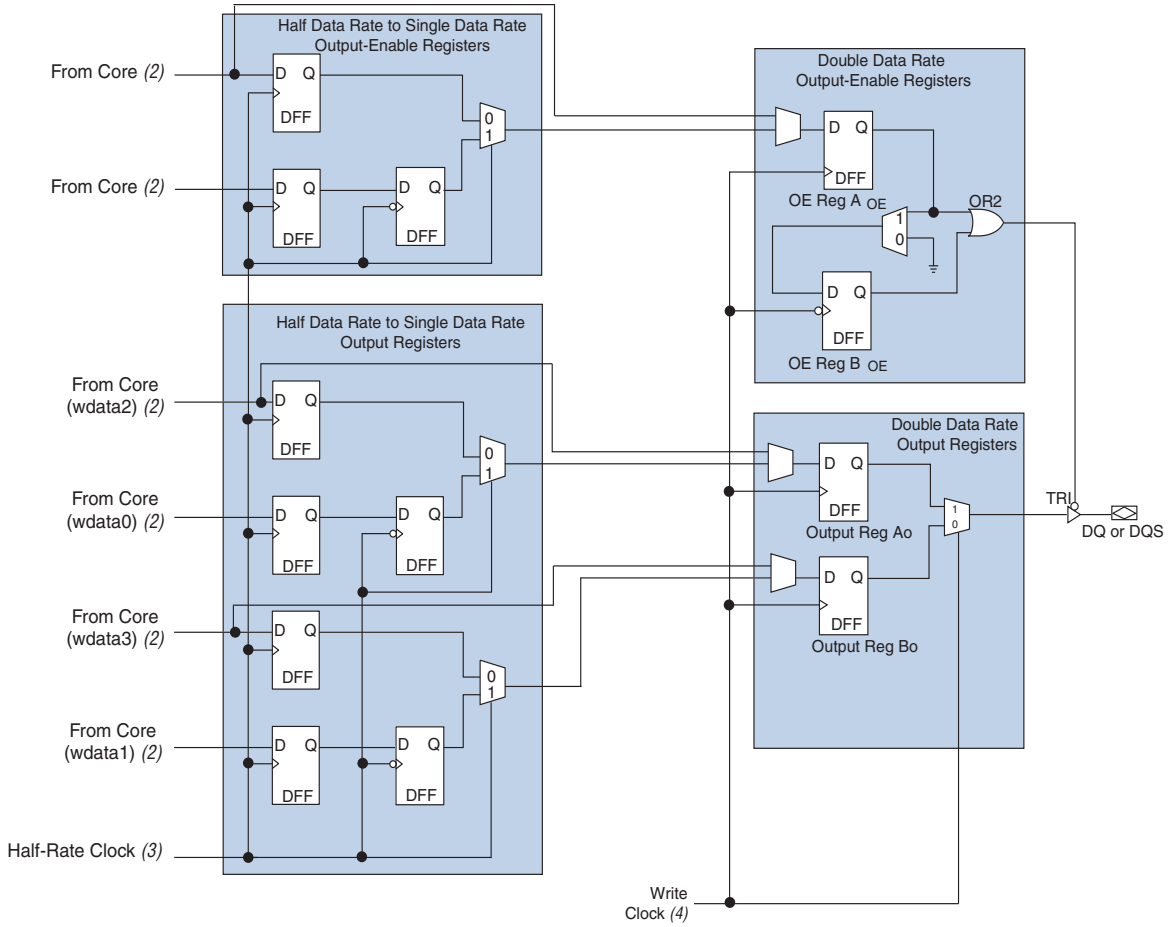
(2) The write clock comes from the PLL. The DQ write clock and DQS write clock have a 90° offset between them.

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.<br>■ Updated Figure 7–11, Figure 7–12, Figure 7–13, Figure 7–14, and Figure 7–15.<br>■ Minor text edits. |
| December 2010 | 4.0 | ■ Updated for the Quartus II software version 10.1 release.<br>■ Added Arria II GZ devices information.<br>■ 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.<br>■ 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.<br>■ Updated Table 7–10.<br>■ 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.<br>■ Minor text edits. |
| July 2010 | 3.0 | Updated for Arria II GX v10.0 release:<br>■ 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".<br>■ Update DLL numbering to match with the Quartus II software.<br>■ Minor text edits. |
| November 2009 | 2.0 | Updated for Arria II GX v9.1 release:<br>■ Updated Table 7–1, Table 7–2, and Table 7–5.<br>■ 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.<br>■ Updated the "Arria II GX External Memory Interface Features" section.<br>■ Added new "Combining ×16/×18 DQ/DQS Groups for ×36 QDR II+/QDR II SRAM Interface" section.<br>■ Minor text edits. |

**Table 7–11. Document Revision History (Part 2 of 2)**

| Date | Version | Changes |
|---|---|---|
| June 2009 | 1.2 | ■ Added Table 7–2.<br>■ Updated Table 7–1, Table 7–3, and Table 7–5.<br>■ 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.<br>■ 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:



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.

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**.

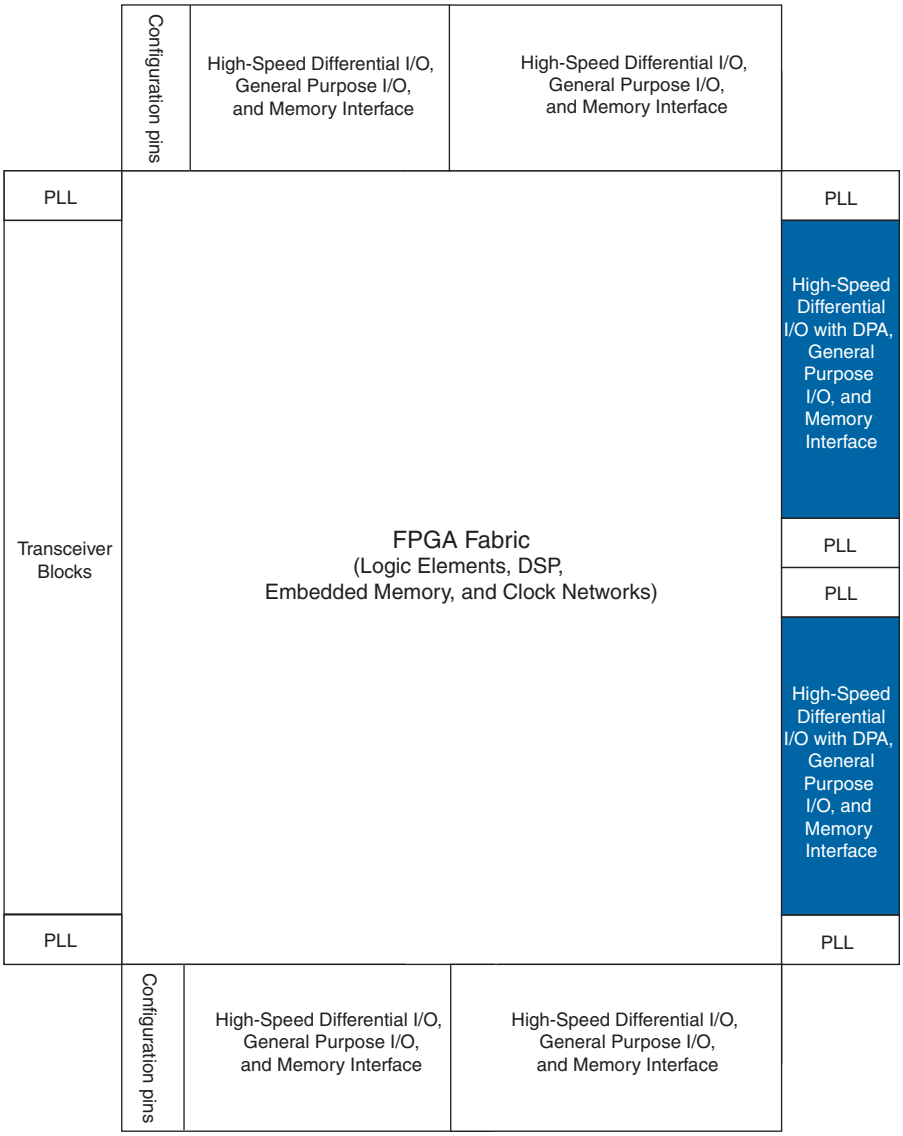
For more information about I/O banks, refer to the *I/O Features in Arria II Devices* chapter.

# 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)***

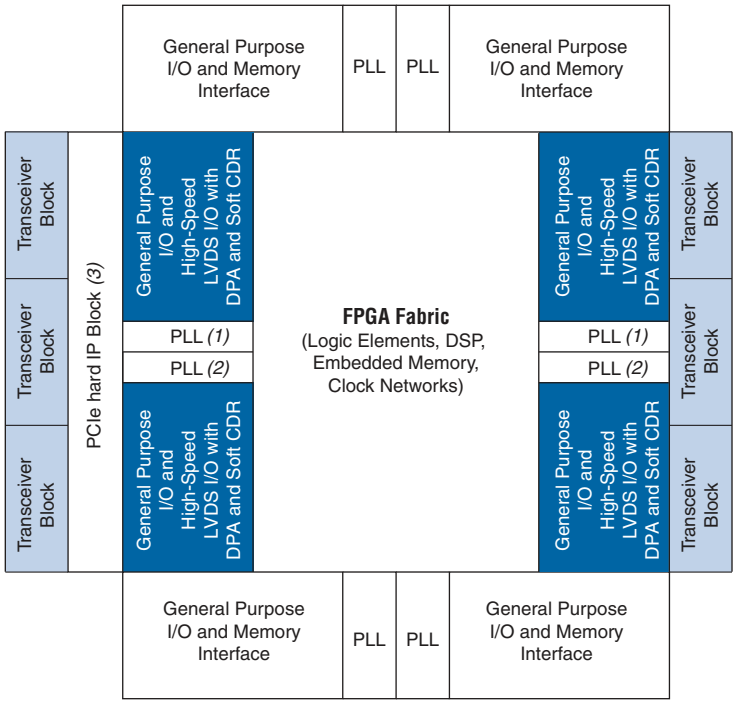



**Notes to Figure 8–1:**

(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:**

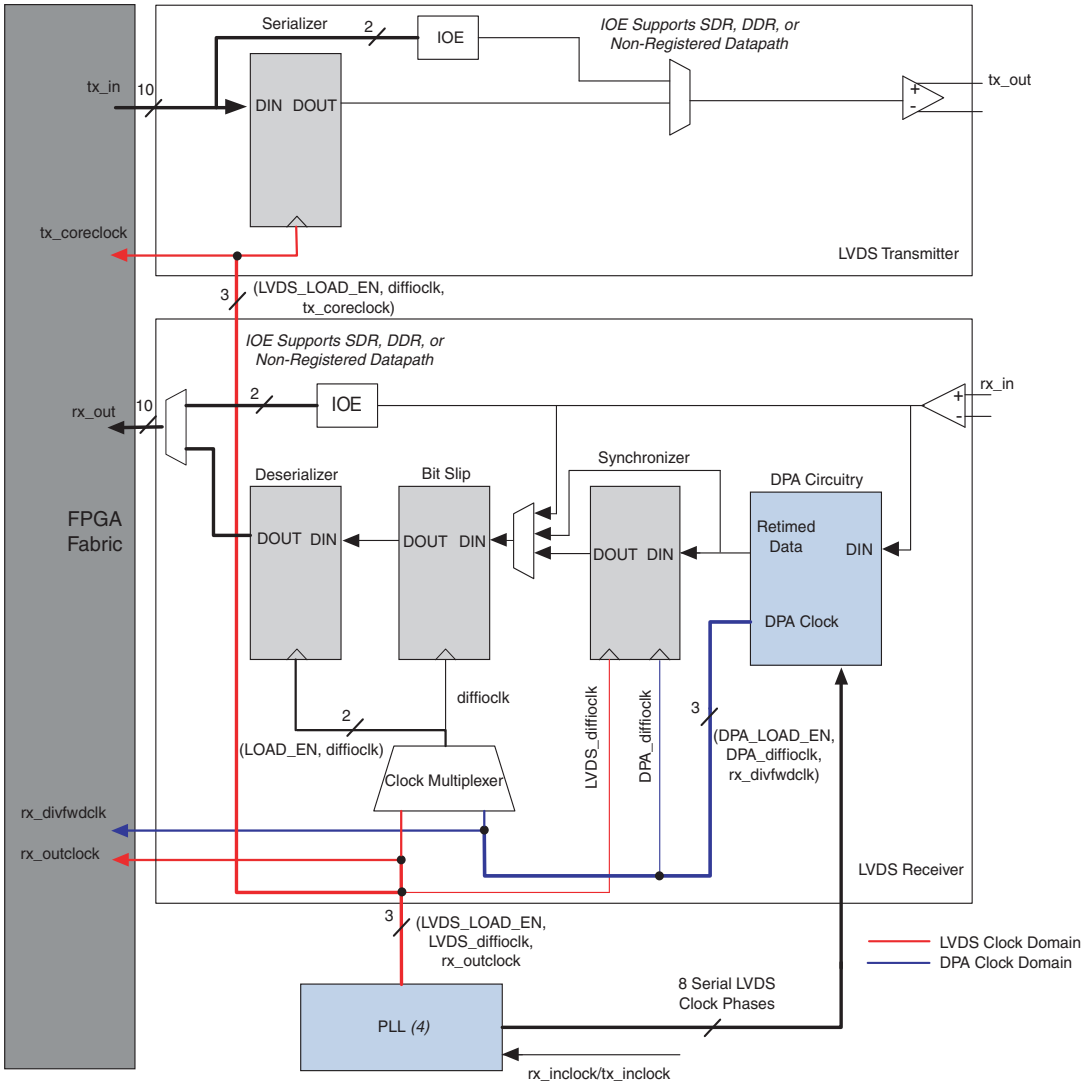
(1) Rx = true LVDS input buffers without $R_D$ OCT, eTx = emulated LVDS output buffers (either `LVDS_E_1R` or `LVDS_E_3R`).
(2) The LVDS Rx and Tx channels are equally divided between the top and bottom sides of the device.
(3) The LVDS channel count does not include dedicated clock input pins.

# 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.

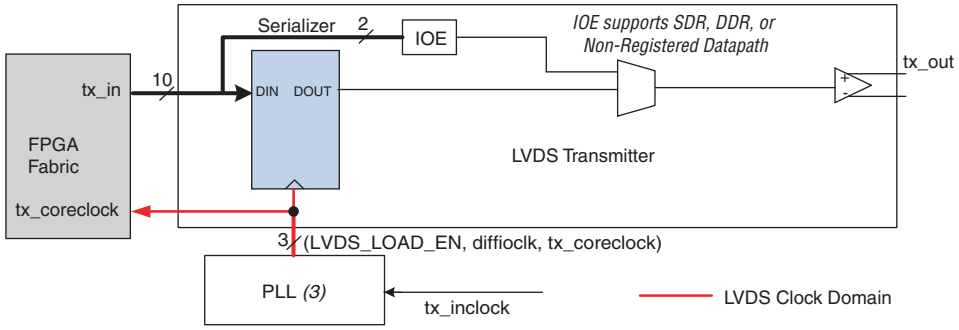
(4) Arria II GX center/corner PLL or Arria II GZ left/right PLL.

# 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:**

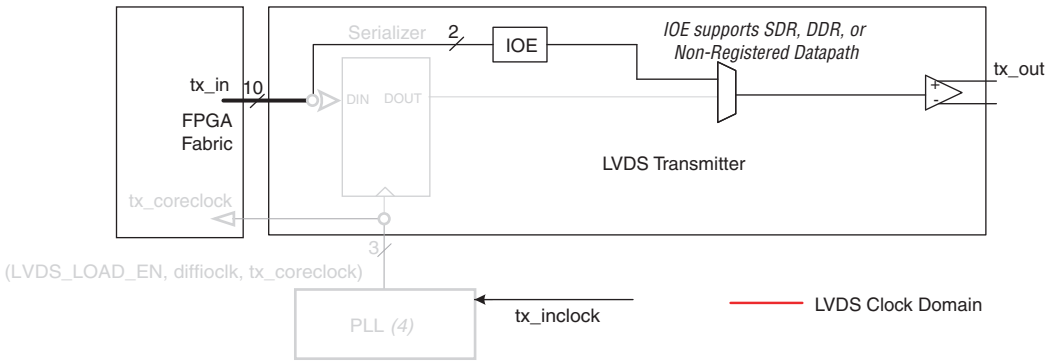
(1) In SDR and DDR modes, the data width is 1 and 2 bits, respectively.
(2) The `tx_in` port has a maximum data width of 10 bits.
(3) Arria II GX center/corner PLL or Arria II GZ left/right PLL.

## 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)***



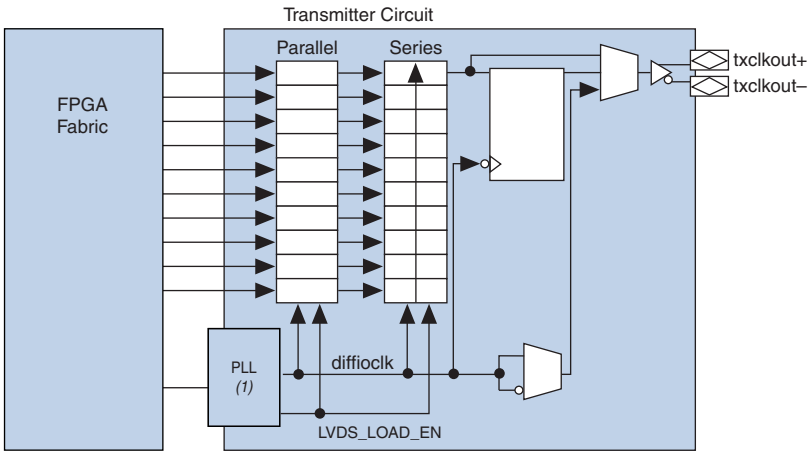

**Notes to Figure 8–5:**

(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**



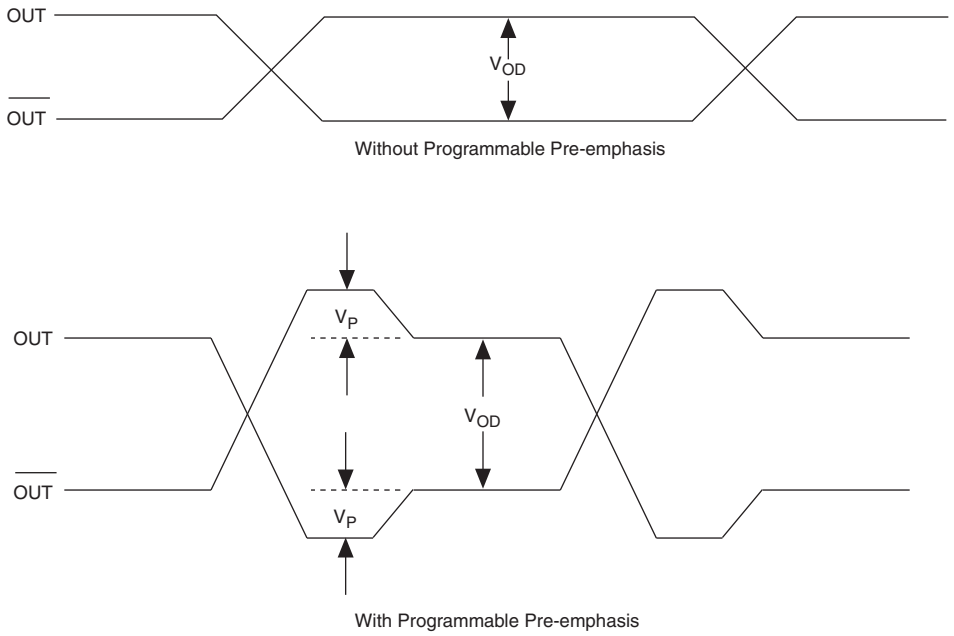

**Note to Figure 8–6:**

(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),<br>1 (medium low),<br>2 (medium high),<br>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.

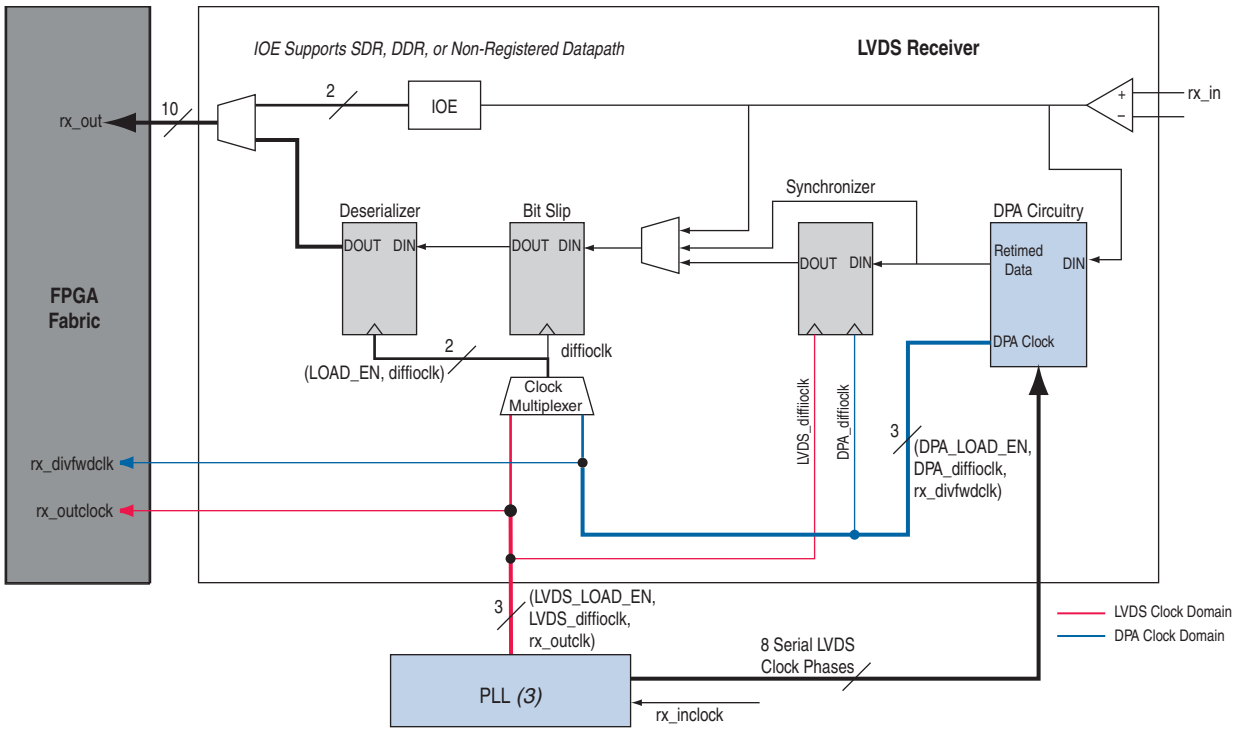
**Table 8–6. Programmable $V_{OD}$ Settings in Quartus II Software Assignment Editor**

| 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:

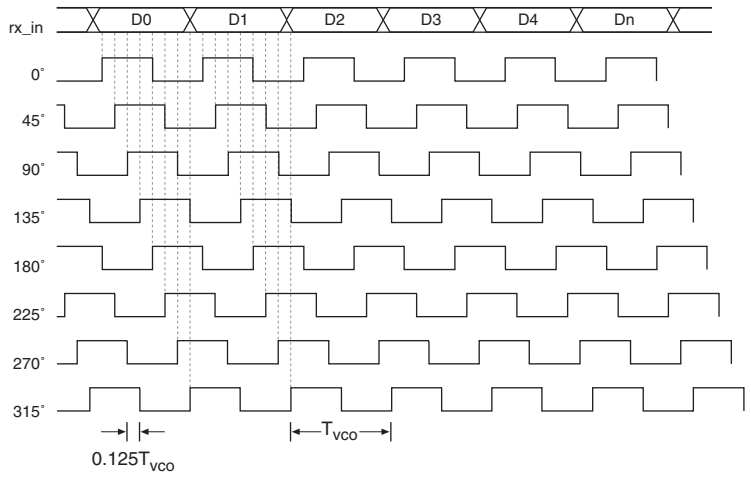
- "DPA" on page 8–12
- "Synchronizer" on page 8–13
- "Data Realignment Block (Bit Slip)" on page 8–14
- "Deserializer" on page 8–15

### 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)

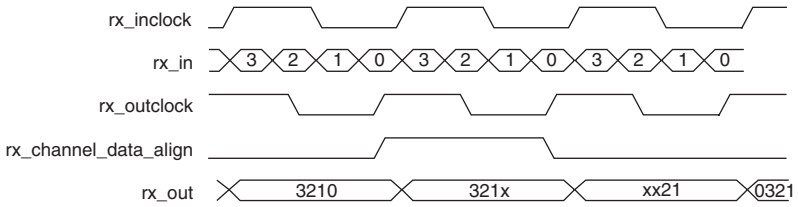
Skew in the transmitted data along with skew added by the link causes channel-to-channel skew on the received serial data streams. If you enabled the DPA block, the received data is captured with different clock phases on each channel and might cause the received data to be misaligned from channel to 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**.

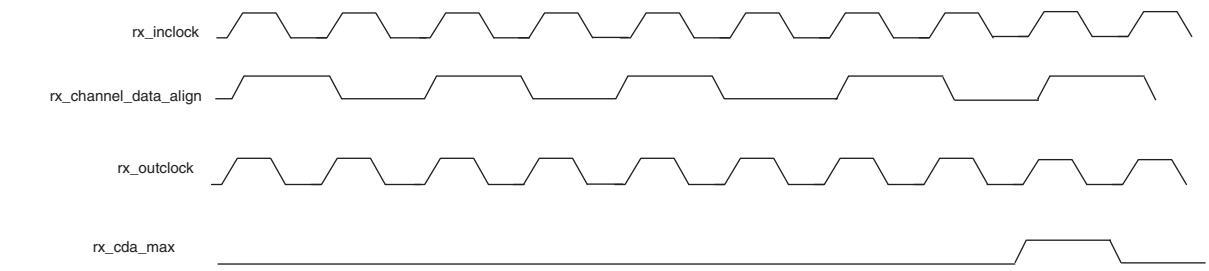
**Figure 8–10. Data Realignment Timing**




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.

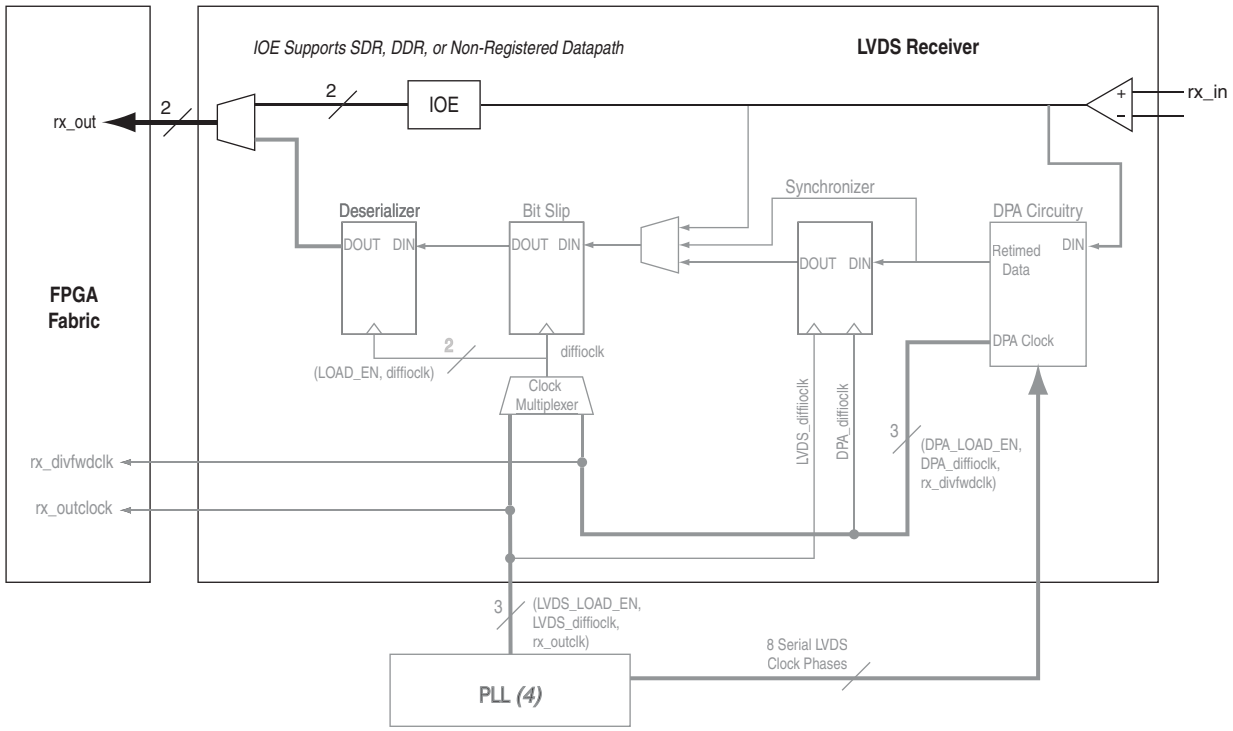
**Figure 8–11. Receiver Data Re-Alignment Rollover**




## 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

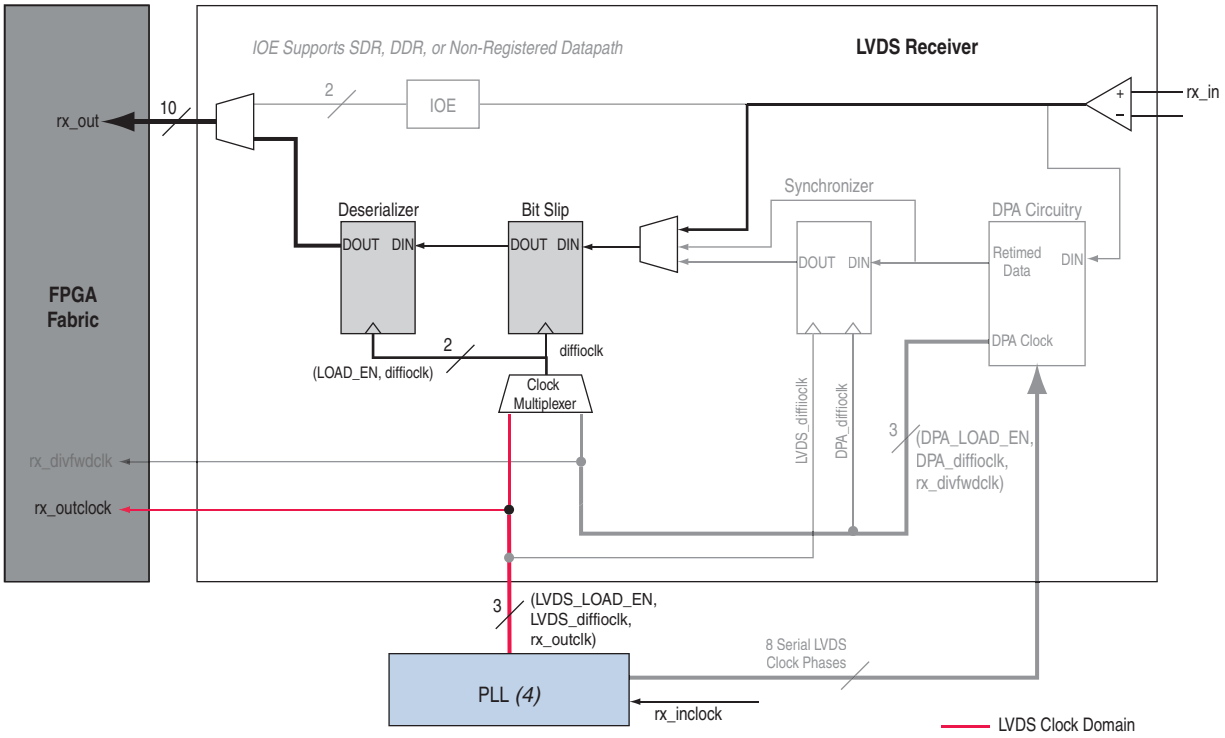
Non-DPA mode allows you to statically select the optimal phase between the source-synchronous reference clock and the input serial data to compensate for any skew between the two signals. The reference clock must be a differential signal. Figure 8–13 shows the non-DPA datapath block diagram. Input serial 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)***



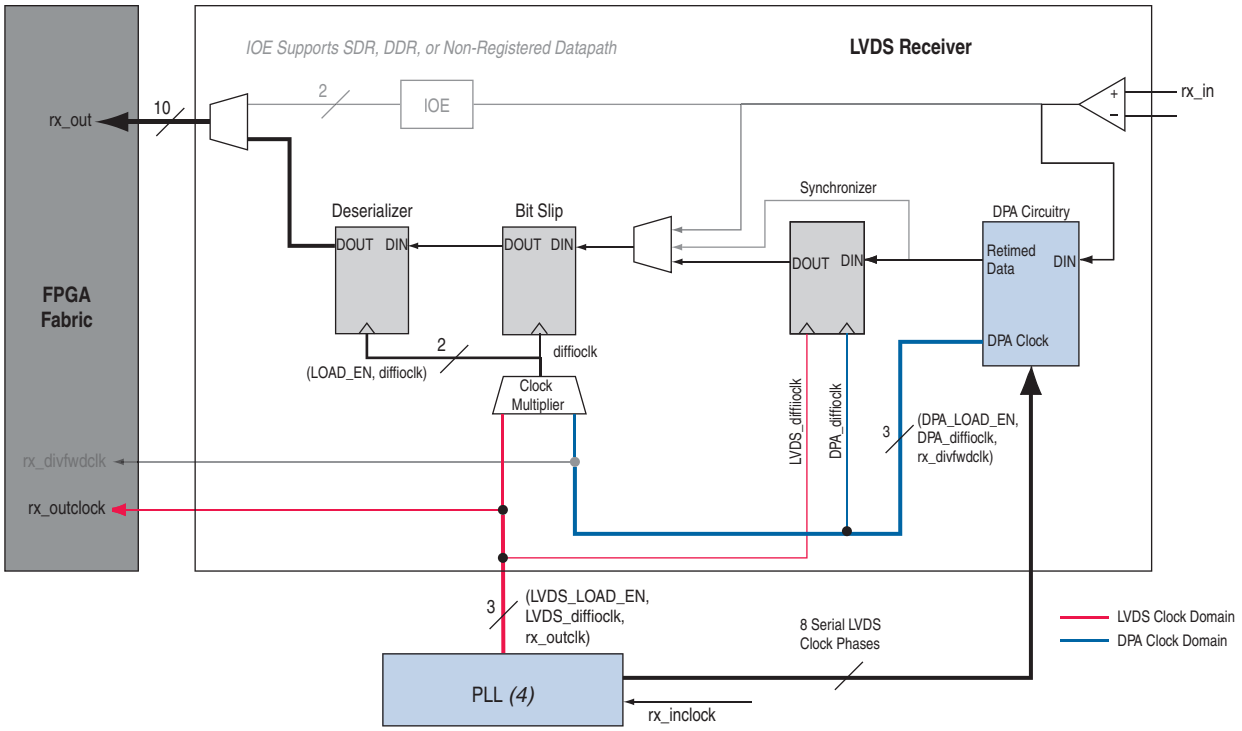

**Notes to Figure 8–13:**

(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)***



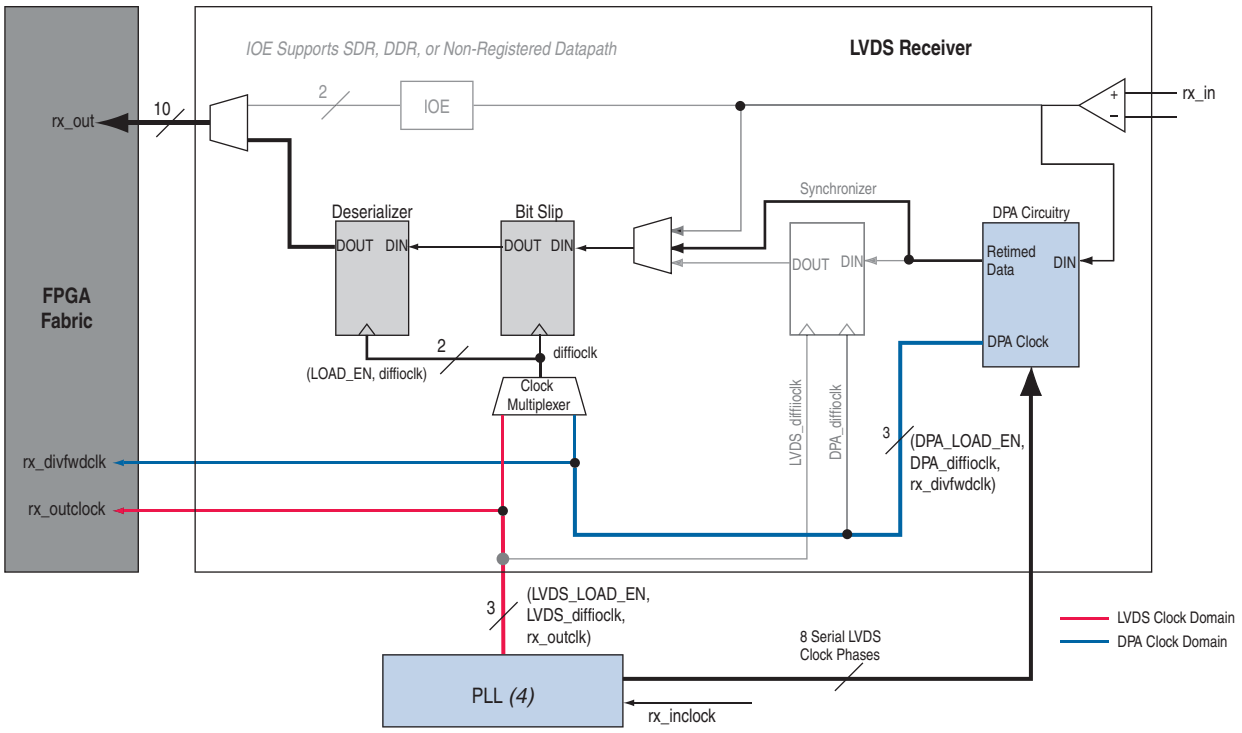

**Notes to Figure 8–14:**

(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**

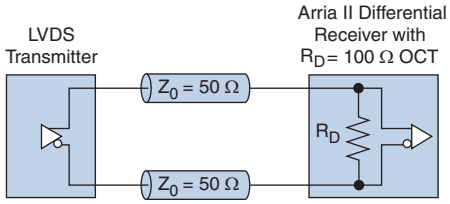



Table 8–7 lists the assignment name and its value for differential input OCT in the Quartus II software Assignment Editor.

**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**

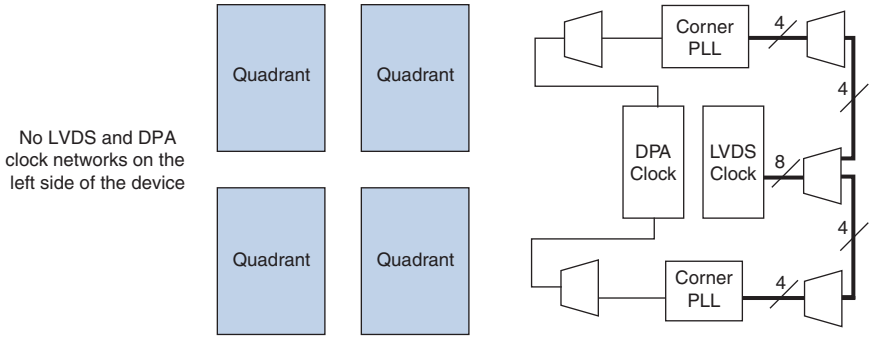
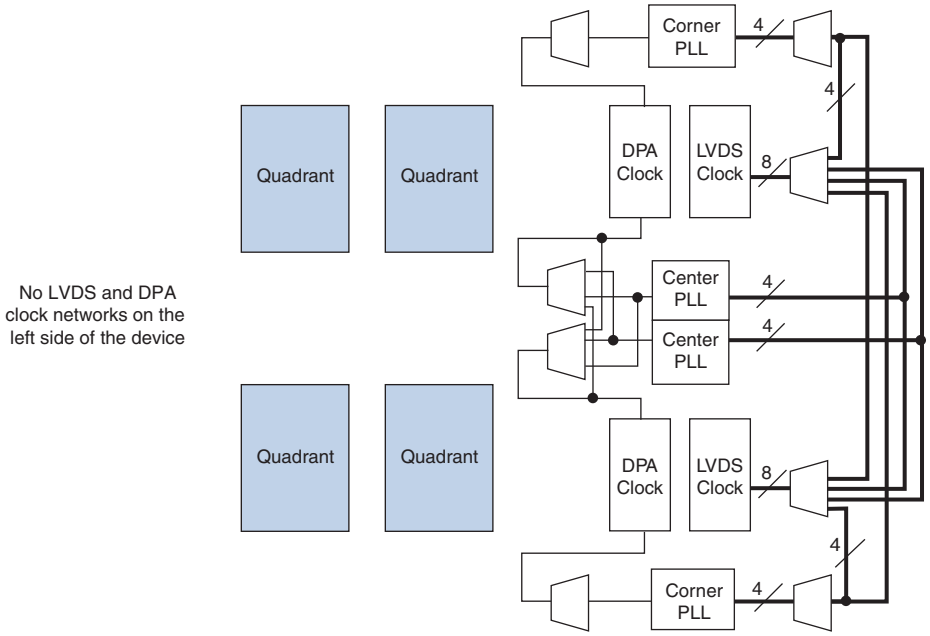


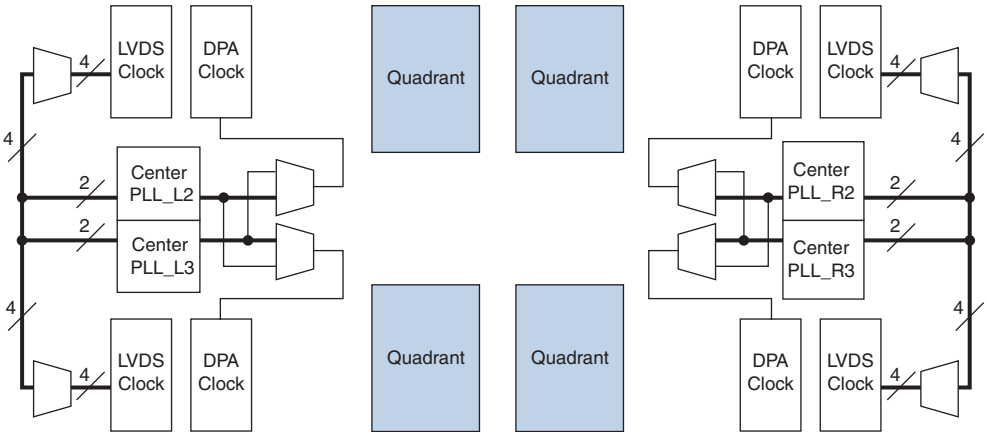
**Figure 8–18. LVDS and DPA Clock Networks in the Arria II GX Devices with Center PLLs**




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

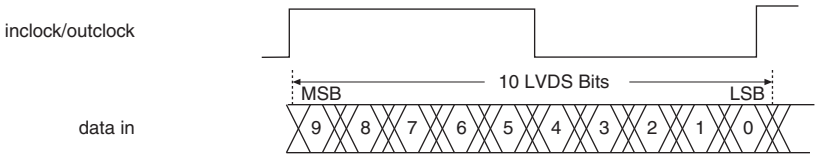
This section describes the timing budget, waveforms, and specifications for source-synchronous signaling in Arria II devices. Timing analysis for the differential block is different from traditional synchronous timing analysis techniques. Therefore, it is important to understand how to analyze timing 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**



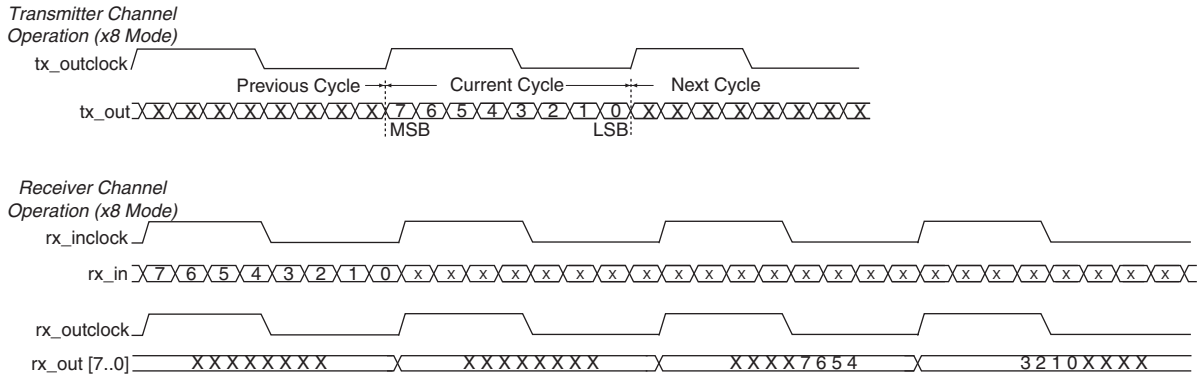

## Differential I/O Bit Position

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.**

$$RSKM = \frac{TUI - SW - TCCS}{2}$$

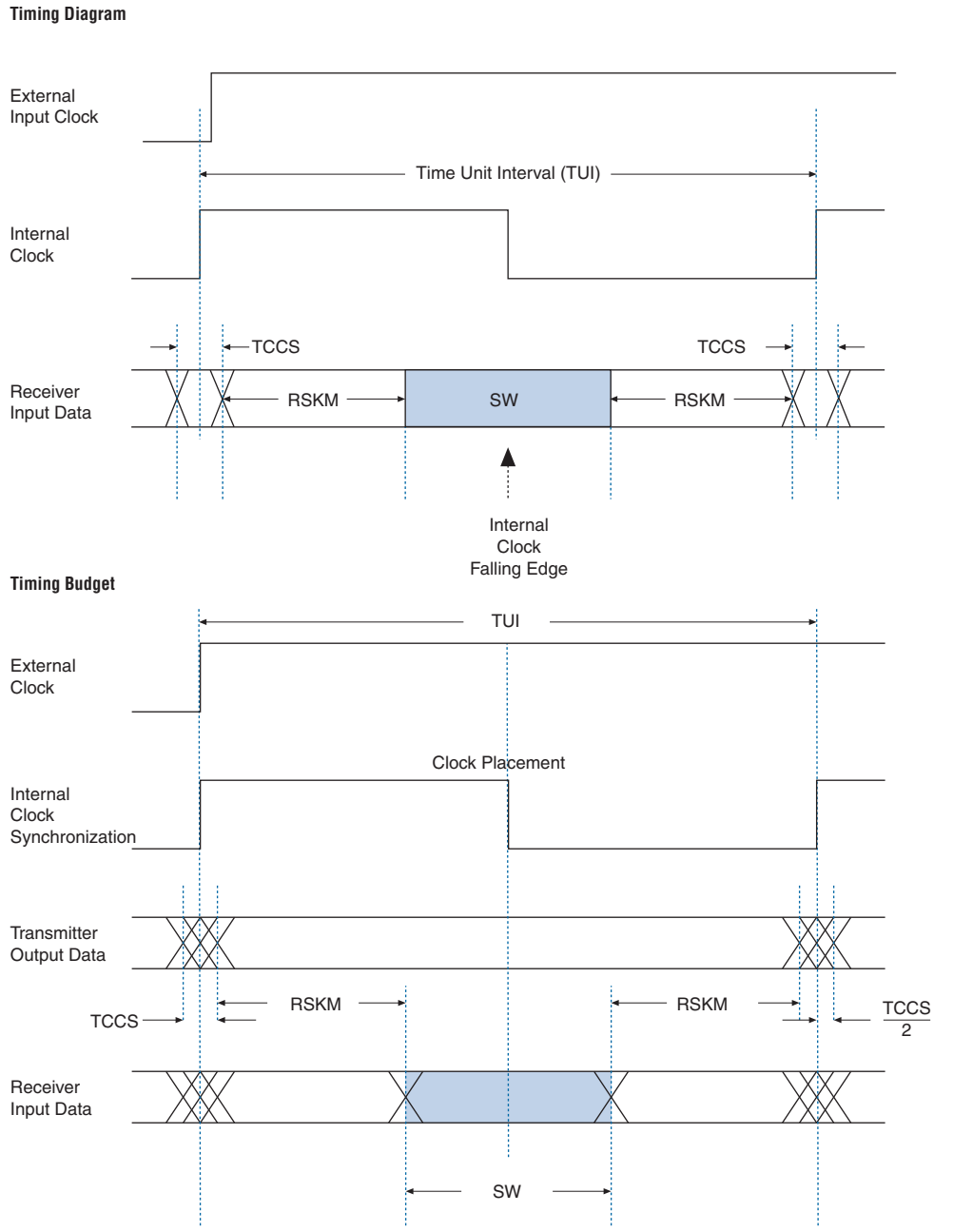
Where:

- 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

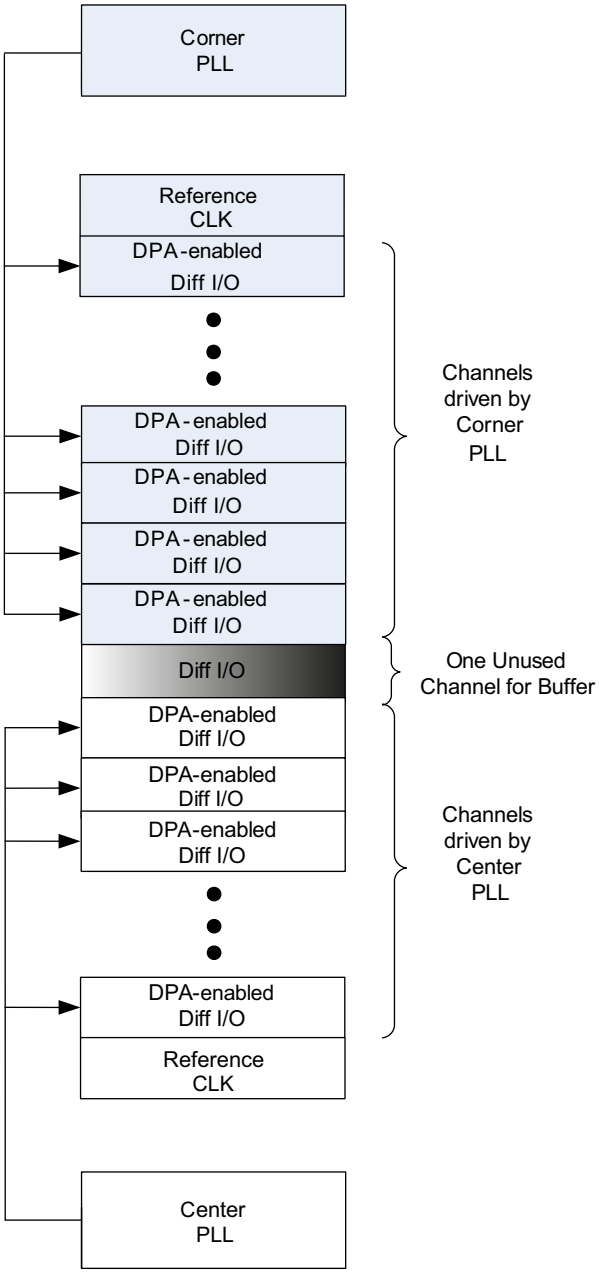
If the number of DPA-enabled channels driven by each center or corner PLL exceeds 25 LAB rows, Altera recommends implementing data realignment (bit slip) circuitry for all the DPA channels.

### 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**



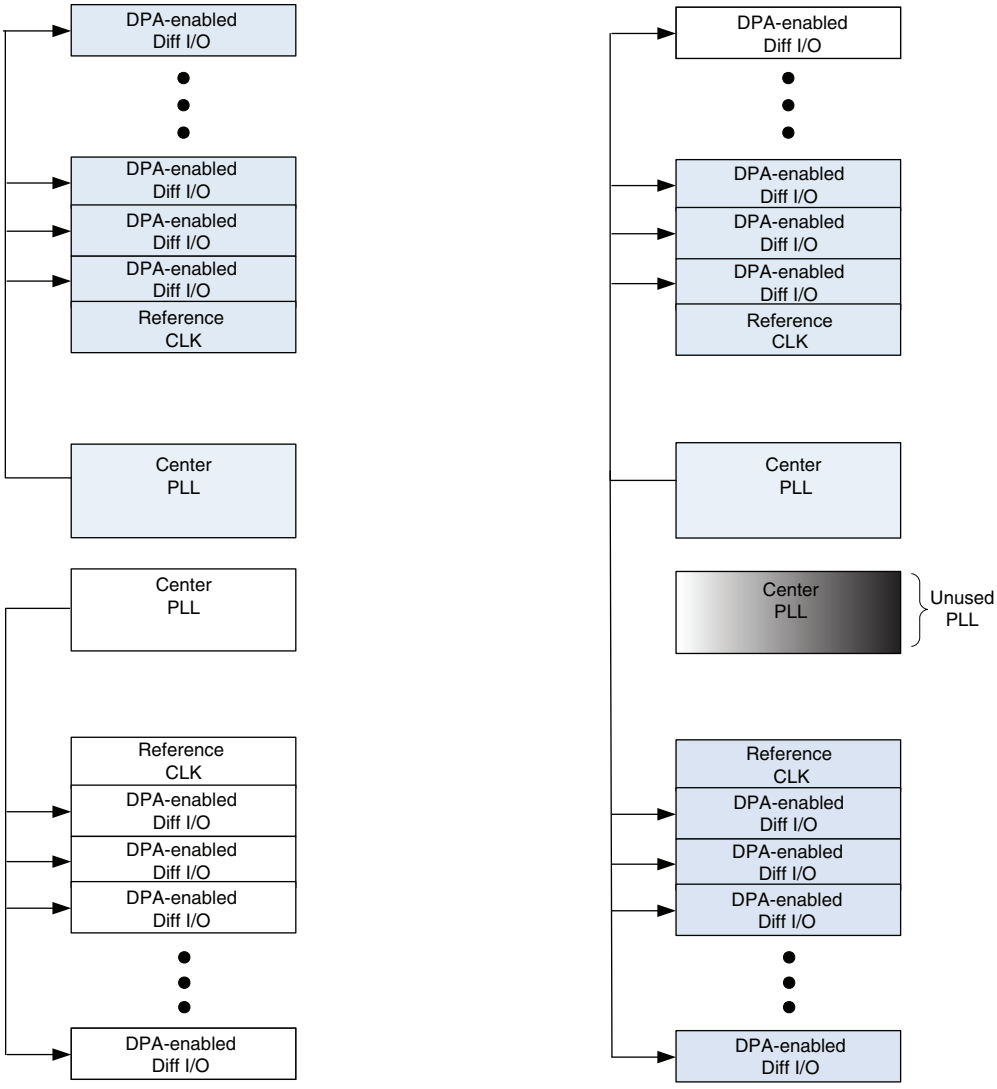
## Using Both Center PLLs

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.

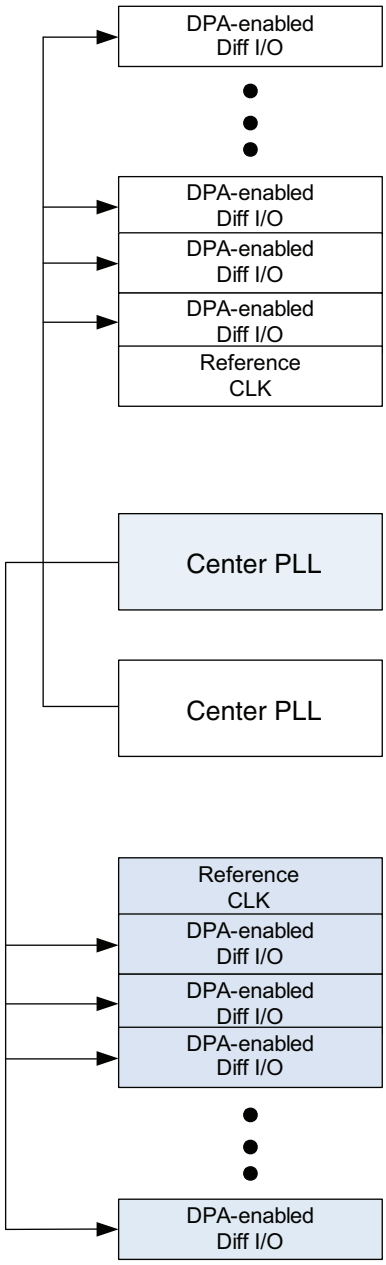
If one of the center PLLs drives the DPA-enabled channels in the upper and lower I/O banks, you cannot use the other center PLL for DPA-enabled channels, as shown in Figure 8–24.

**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**



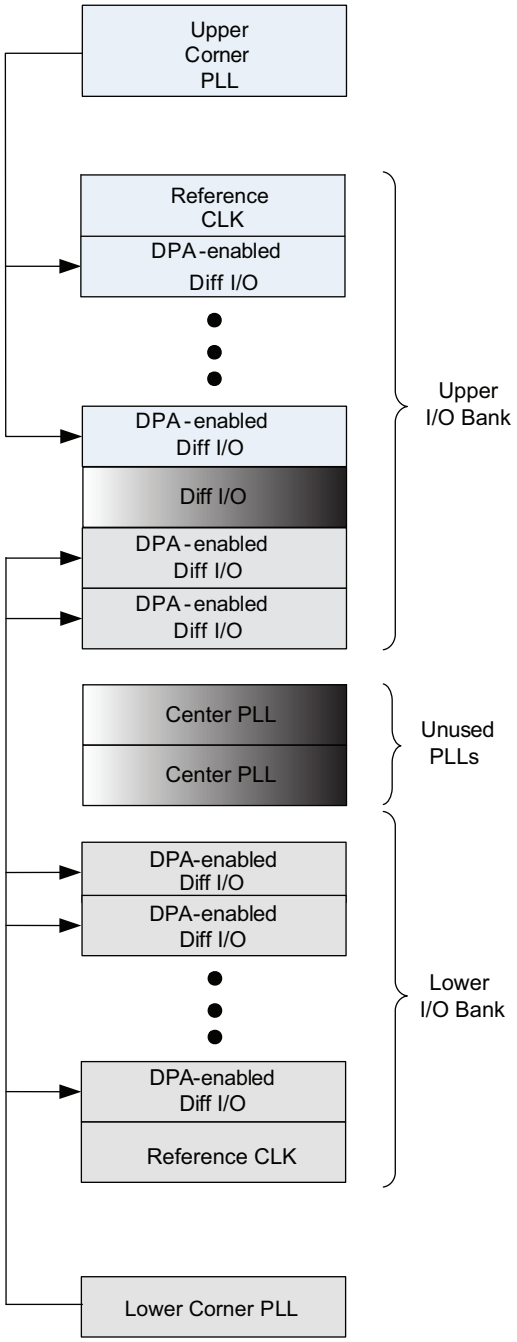
## Using Both Corner PLLs in Arria II GX Devices

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.

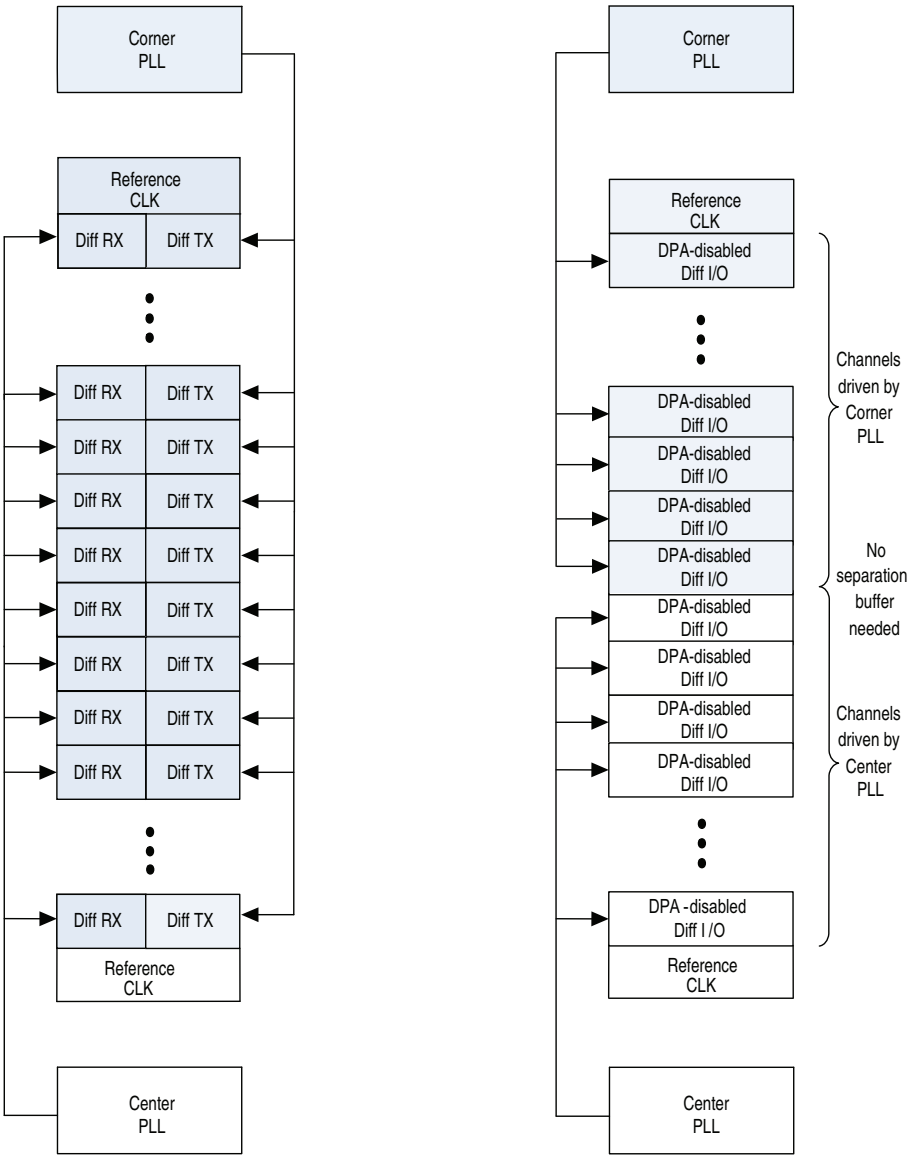
## DPA-Disabled Channel Driving Distance

Each PLL can drive all the DPA-disabled channels in the entire bank.

## Using Corner and Center PLLs in Arria II GX Devices

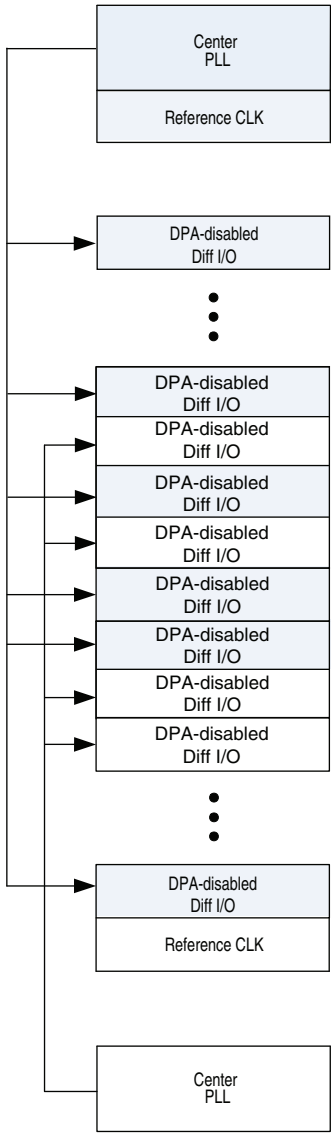
You can use a corner PLL to drive all transmitter channels and you can use a center PLL to drive all DPA-disabled receiver channels in the same I/O bank. In other words, you can drive a transmitter channel and a receiver channel in the same LAB row by two different PLLs, as shown in Figure 8–27.

**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**

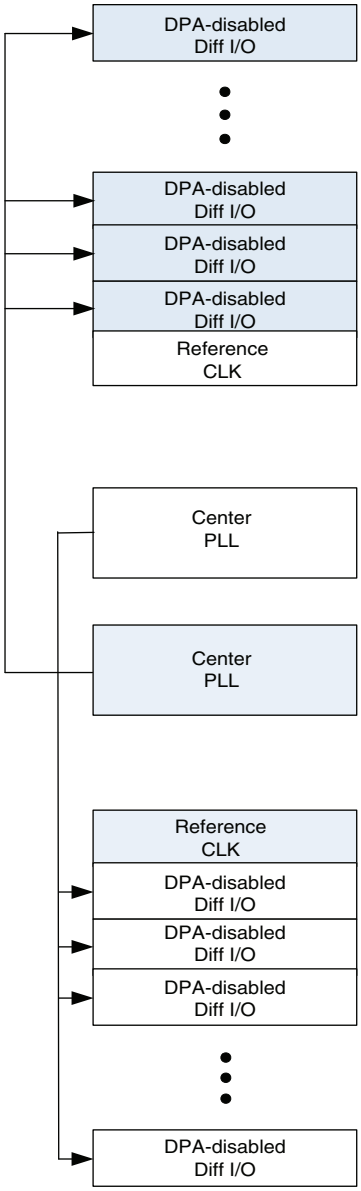


## Using Both Center PLLs

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.<br>■ Minor text edits. |
| June 2011 | 4.1 | ■ Updated Figure 8–2.<br>■ Minor text edits. |
| December 2010 | 4.0 | Updated for the Quartus II software version 10.1 release:<br>■ Added Arria II GZ device information.<br>■ Updated Table 8–3 and Table 8–4.<br>■ 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:<br>■ Updated Table 8–1 and Table 8–2.<br>■ Updated Figure 8–1 and Figure 8–5.<br>■ Updated "Non-DPA Mode" section.<br>■ Removed Table 8–1: Supported Data Range.<br>■ Minor text edit. |
| November 2009 | 2.0 | Updated for Arria II GX v9.1 release:<br>■ Updated Table 8–1 and Table 8–2.<br>■ Updated Figure 8–1.<br>■ Updated "LVDS Channels" and "Non-DPA Mode" sections.<br>■ Minor text edit. |
| June 2009 | 1.1 | ■ Updated Table 8–2 and Table 8–3.<br>■ 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 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.<br>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.<br>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.<br>$V_{CCPD}$ 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.<br>You must connect $V_{CCPD}$ according to the I/O standard used in the same bank:<br>■ For 3.3-V I/O standards, connect $V_{CCPD}$ to 3.3 V.<br>■ For 3.0-V I/O standards, connect $V_{CCPD}$ to 3.0 V.<br>■ 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).<br>$V_{CCPD}$ 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.<br>$V_{CCPD}$ must be greater than or equal to $V_{CCIO}$ of the same bank:<br>■ If the $V_{CCIO}$ of the bank is powered to 3.0 V, $V_{CCPD}$ must be powered up to 3.0 V.<br>■ 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

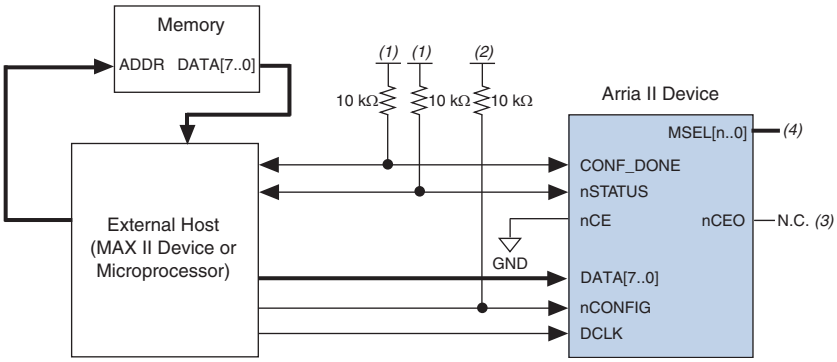
FPP configuration using an external host provides the fastest method to configure Arria II devices. In this configuration scheme, you can use a MAX II device or microprocessor as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target 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.

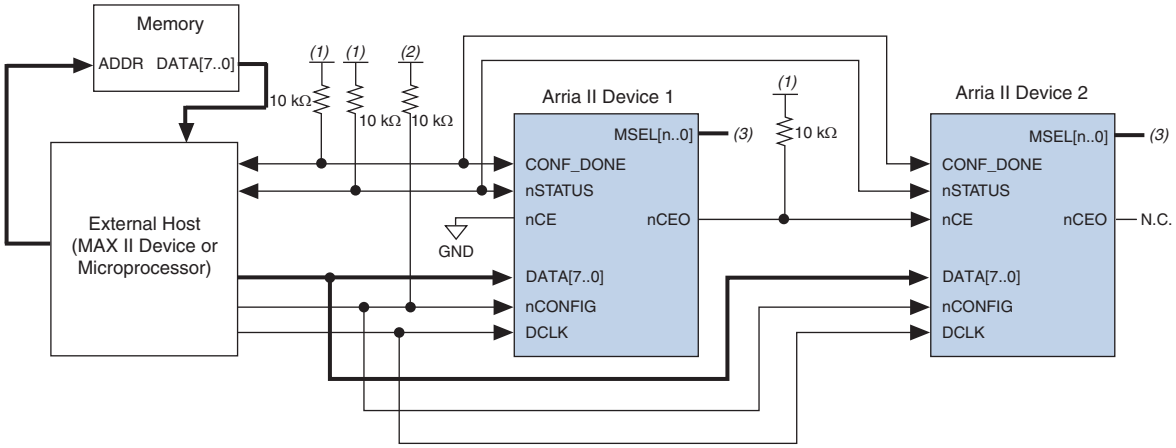
Arria II devices receive configuration data on the DATA[7..0] pins and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. If you are using the Arria II decompression, design security, or both features, configuration data is latched on the rising edge of 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.

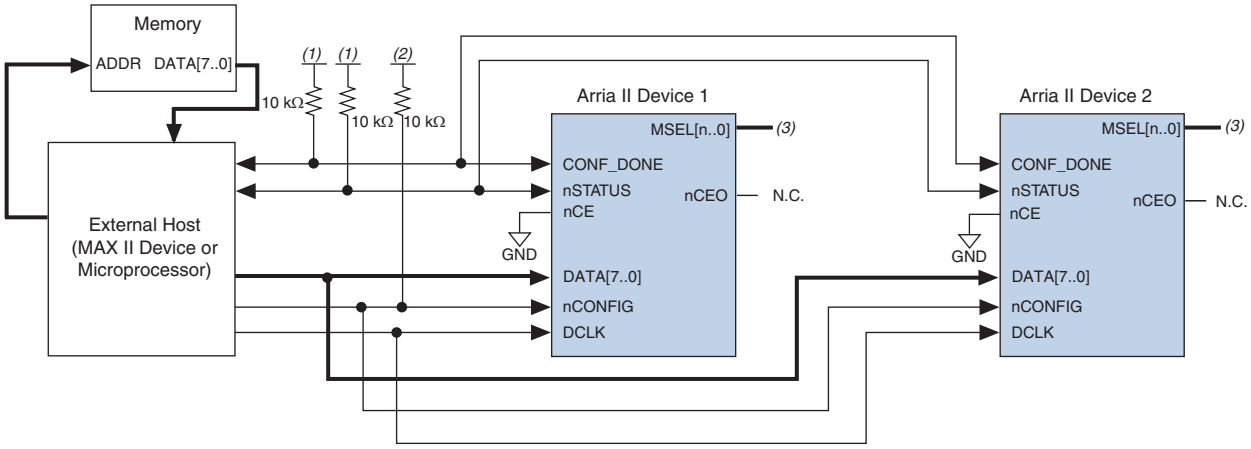
After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the nCE pin of the second device, which prompts the second device to begin configuration. The second device in the chain begins configuration in one clock cycle; therefore, the transfer 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.

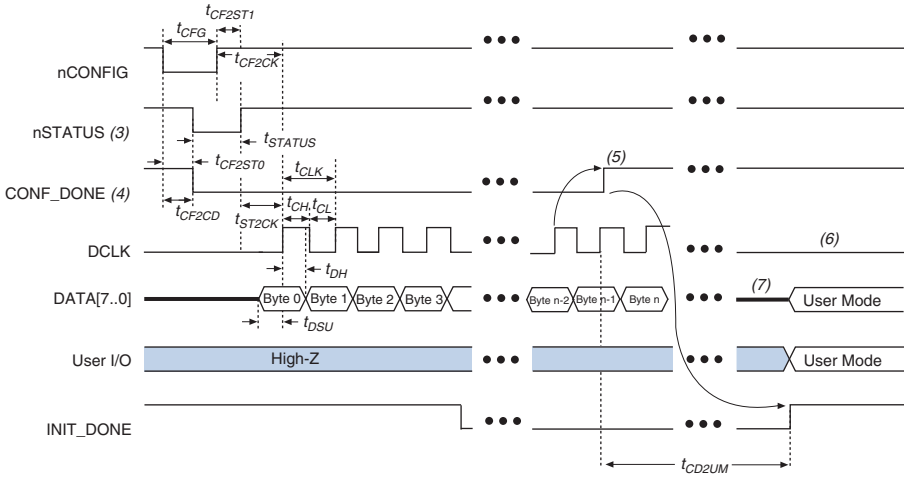
You can use a single configuration chain to configure Arria II devices with other Altera devices that support FPP configuration. To ensure that all devices in the chain complete configuration at the same time, or that an error flagged by one device initiates reconfiguration in all devices, tie all of 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.

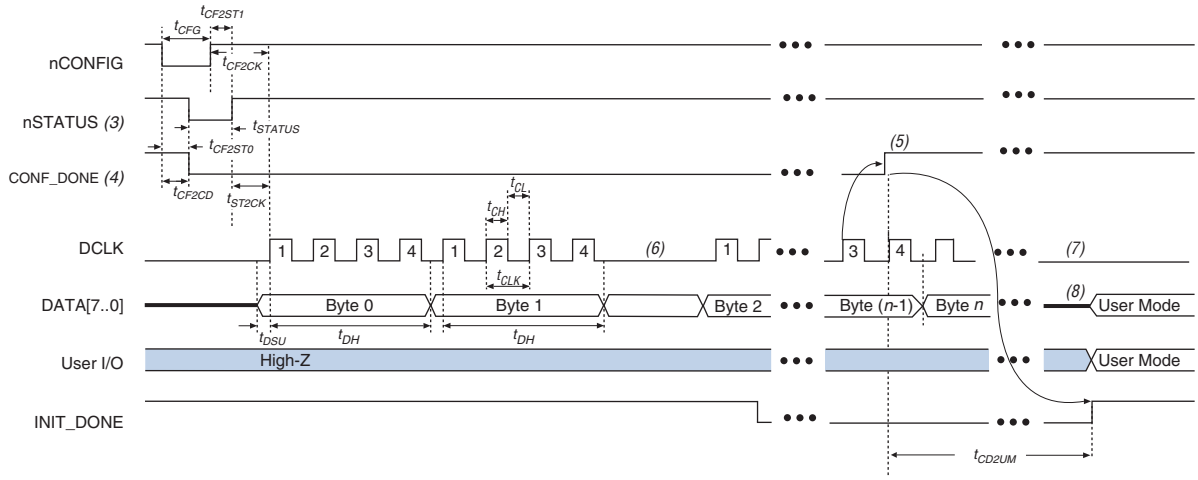
(3) You can obtain this value if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(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.

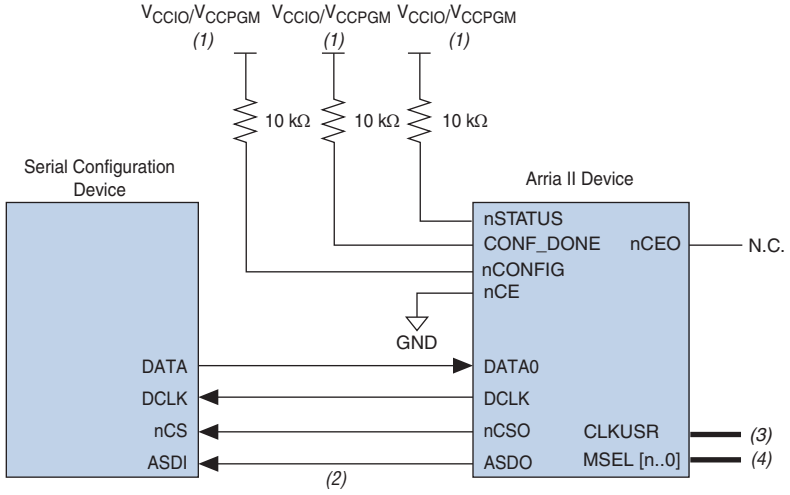
For more information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

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.

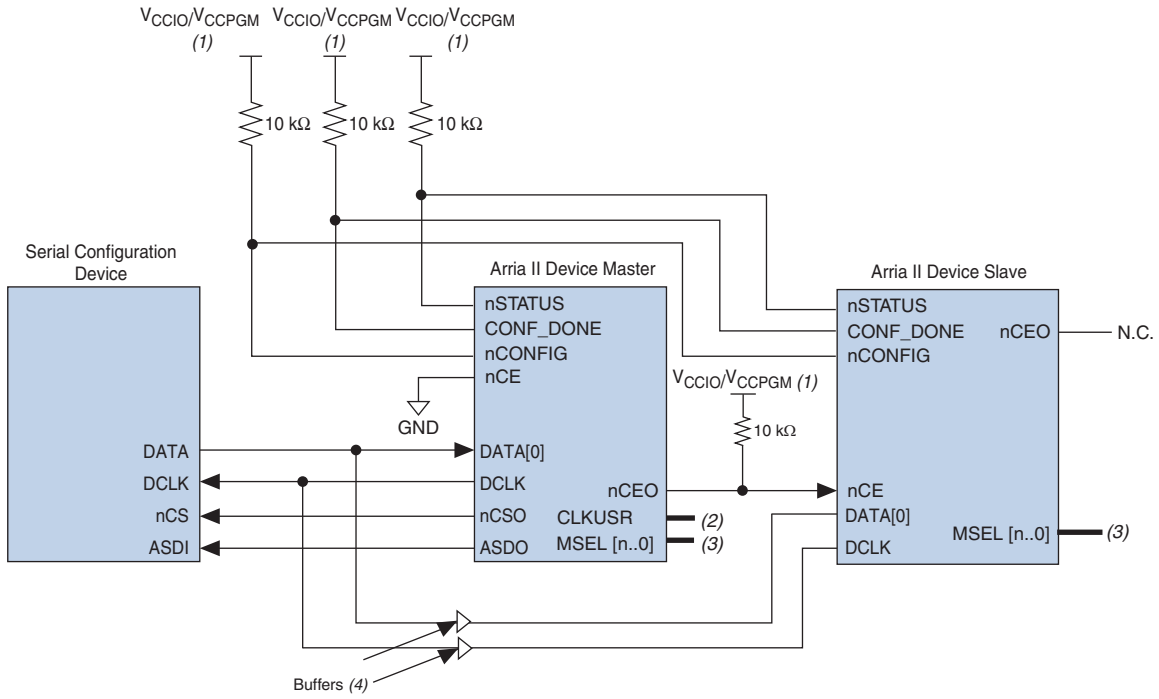
In configuration mode, Arria II devices enable the serial configuration device by driving the nCSO output pin low, which connects to the chip select (nCS) pin of the configuration device. The Arria II device uses the serial clock (DCLK) and serial data output (ASDO) pins to send operation commands, read 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.

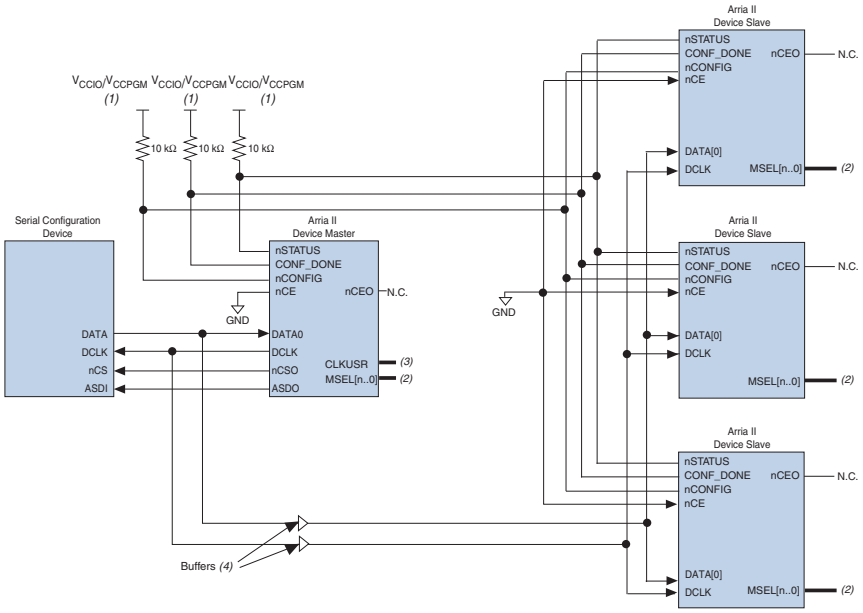
If the configuration bitstream size exceeds the capacity of a serial configuration device, you must select a larger configuration device, enable the compression feature, or both. When configuring multiple devices, the size of the bitstream is the sum of the configuration bitstreams of the individual 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.

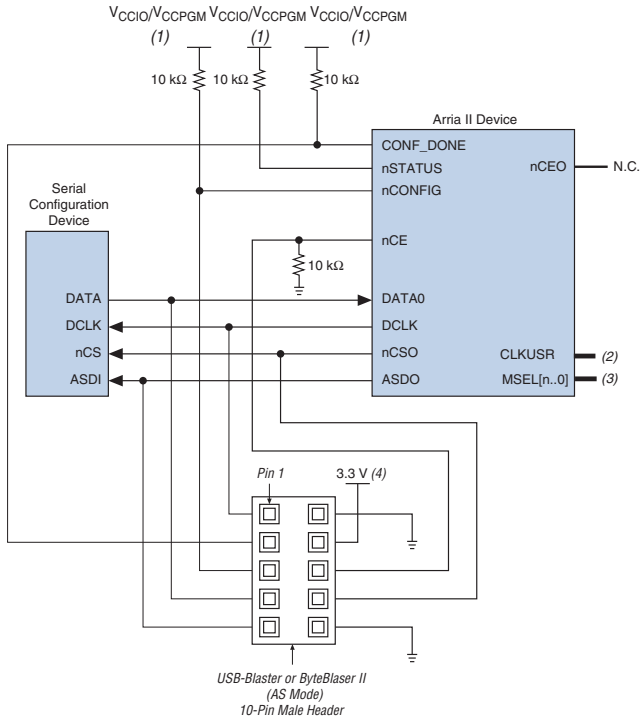
For more information about SFL, refer to *AN 370: Using the Serial FlashLoader with Quartus II Software*.

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.

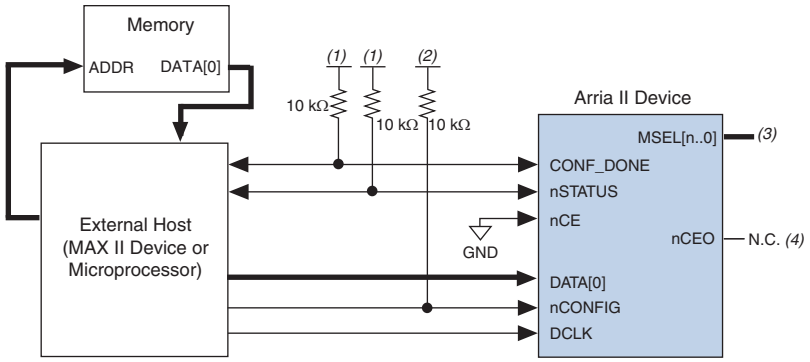
The Arria II decompression and design security features are available when configuring your Arria II device using PS mode.

## 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.

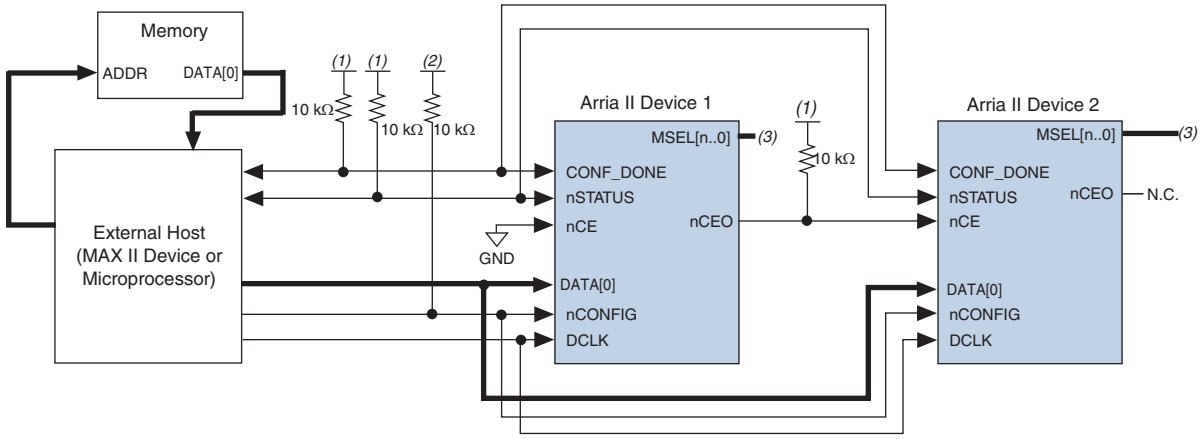
(3) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

(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.

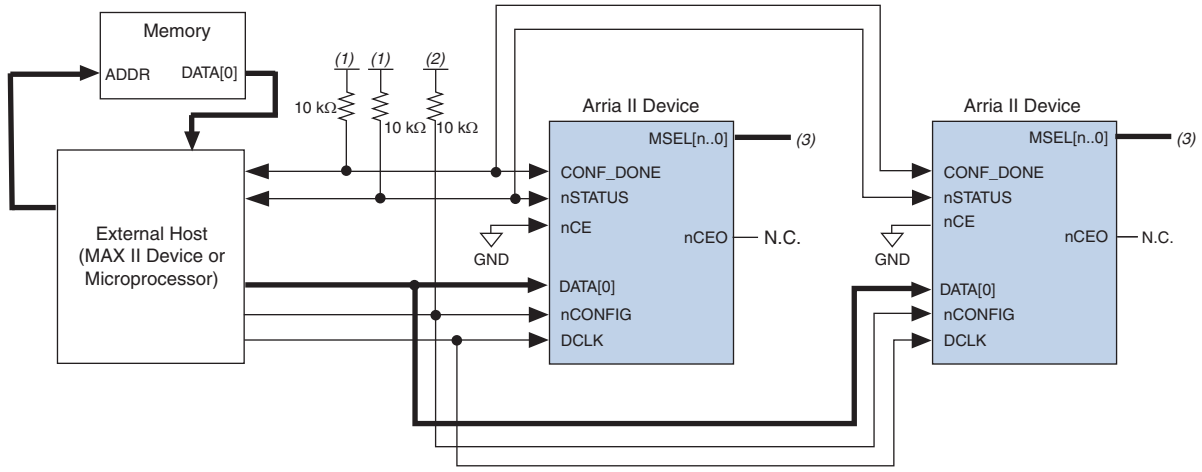
After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration in one clock cycle. Therefore, the transfer 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 $V_{CCIO}$/$V_{CCPGM}$.

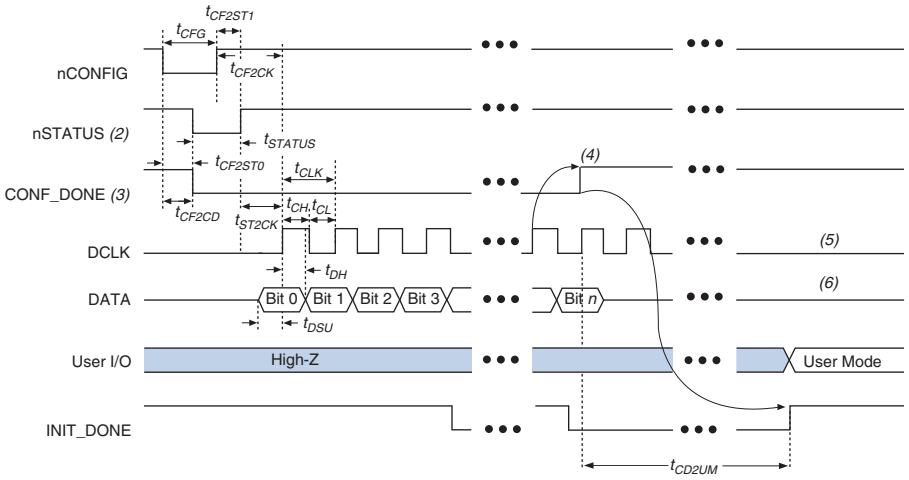
(2) A pull-up resistor to $V_{CCIO}$/$V_{CCPGM}$ or a pull-down resistor keeps the nCONFIG line in a known state when the external host is not driving the line.

(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.

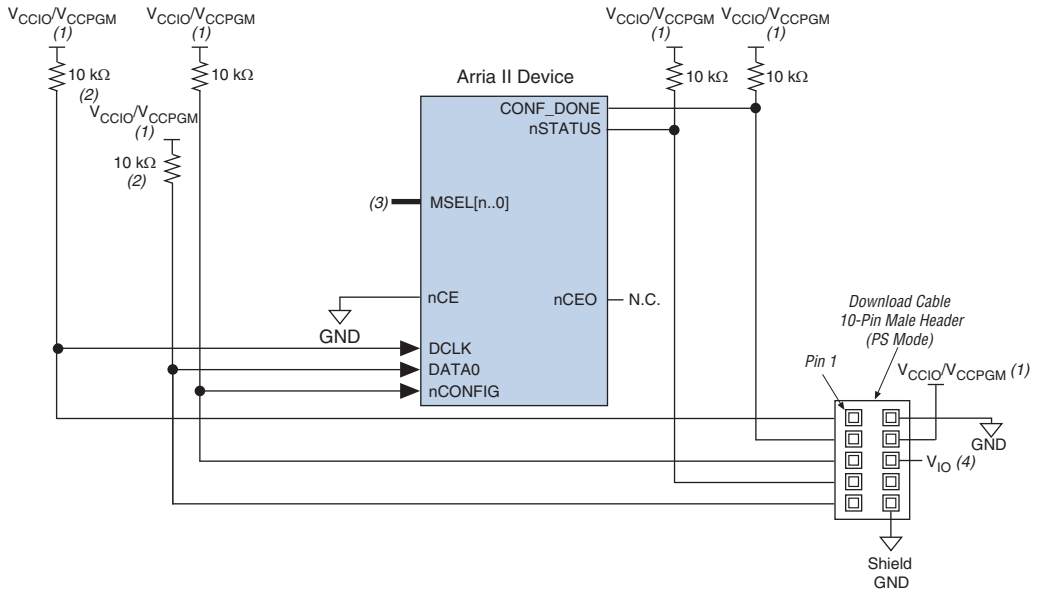
In a PS configuration with a download cable, an intelligent host (such as a PC) transfers data from a storage device to the Arria II device using the download cable.

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.

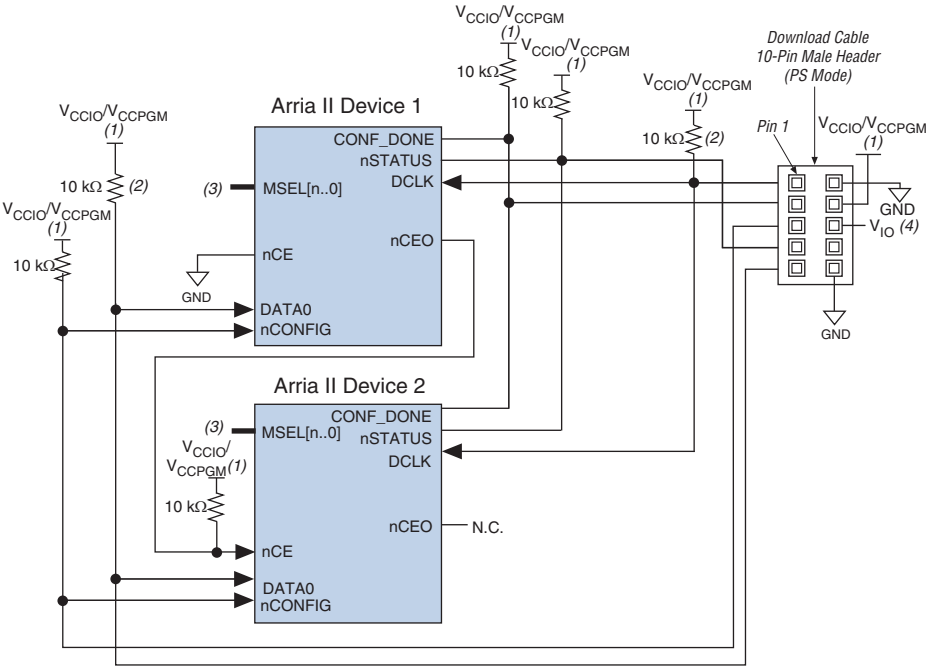
(4) In the USB-Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for AS programming; otherwise, it is a no connect.

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. |

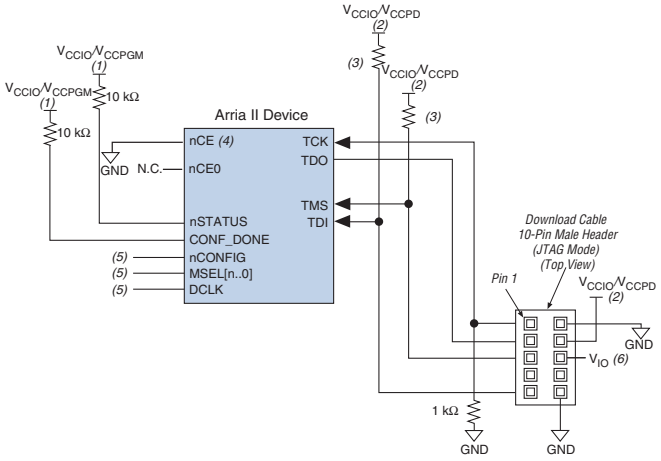
**Note to Table 9–13:**

(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)**

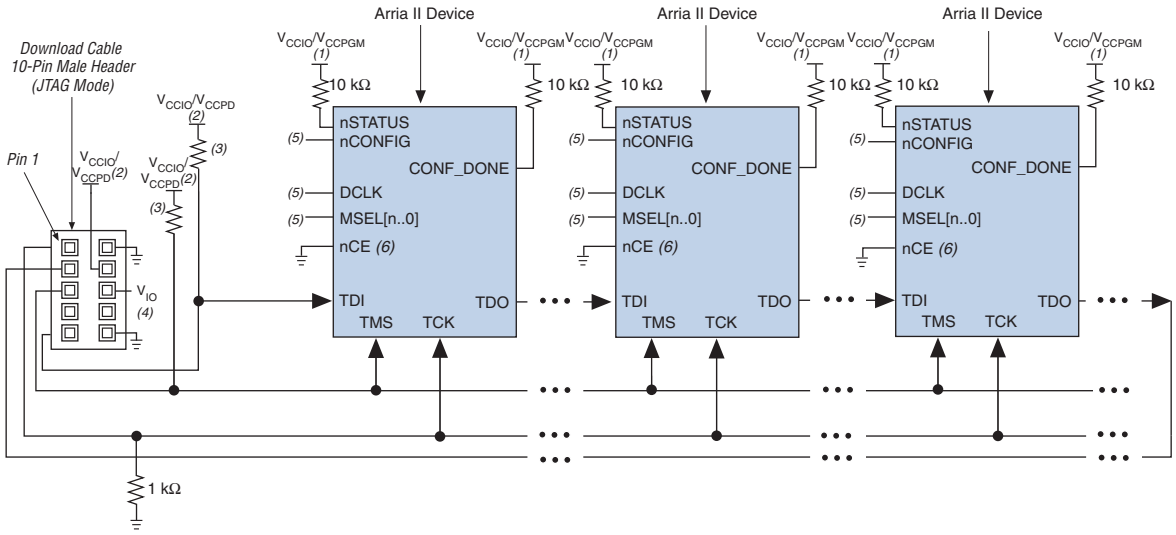
| Signal | Description |
|---|---|
| nSTATUS | Pull to $V_{CCIO}$ or $V_{CCPGM}$ using a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin must be pulled up to $V_{CCIO}$ or $V_{CCPGM}$ individually. |
| CONF_DONE | Pull to $V_{CCIO}$ or $V_{CCPGM}$ using a 10-kΩ 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.

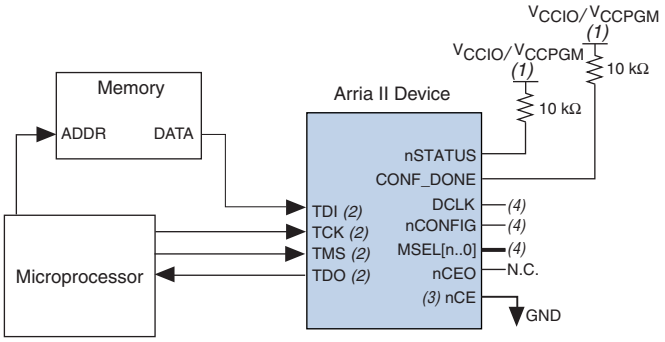
For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera Device Chains* chapter in volume 2 of the *Configuration Handbook*.

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.<br>You must connect $V_{CCPD}$ according to the I/O standard used in the same bank:<br>■ For 3.3-V I/O standards, connect $V_{CCPD}$ to 3.3 V<br>■ For 3.0-V I/O standards, connect $V_{CCPD}$ to 3.0 V<br>■ For 2.5-V and below I/O standards, connect $V_{CCPD}$ to 2.5 V<br>$V_{CCPD}$ 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.<br>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.<br>You must hardwire these pins to $V_{CCPGM}$ or GND.<br>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.<br>You must hardwire these pins to $V_{CCPD}$ or GND.<br>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.<br>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.<br>During user mode and regular configuration, this pin is pulled high by an external 10-kΩ resistor.<br>This pin, when driven low by the Arria II device, indicates that the device has encountered an error during configuration.<br>■ Status output—If an error occurs during configuration, nSTATUS is pulled low by the target device.<br>■ Status input—If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state.<br>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.<br>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:<br>■ $V_{CCIO}/V_{CCPGM}$ 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.<br>■ $V_{CCIO}/V_{CCPGM}$ 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.<br>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.<br>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.<br>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.<br>The nCEO pin is powered by $V_{CCIO}$ for Arria II GX devices and $V_{CCPGM}$ for Arria II GZ devices.<br>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.<br>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.<br>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.<br>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.<br>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.<br>In AS mode, DATA0 has an internal pull-up resistor that is always active.<br>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.<br>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].<br>In serial configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated.<br>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.<br>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.<br>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.<br>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.<br>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.<br>It is expected that the clock input waveform have a nominal 50% duty cycle.<br>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.<br>Hold TMS at one or keep TCK static while TRST is changed from 0 to 1.<br>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.

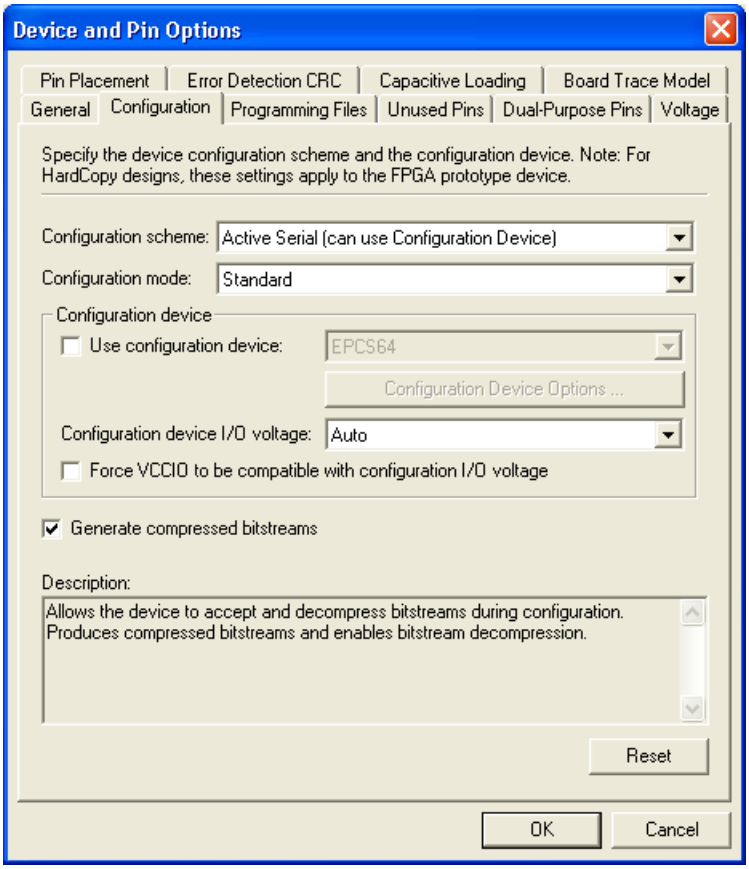
When you enable compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash memory and decreases the time needed to transmit the bitstream to the Arria II device. The 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**



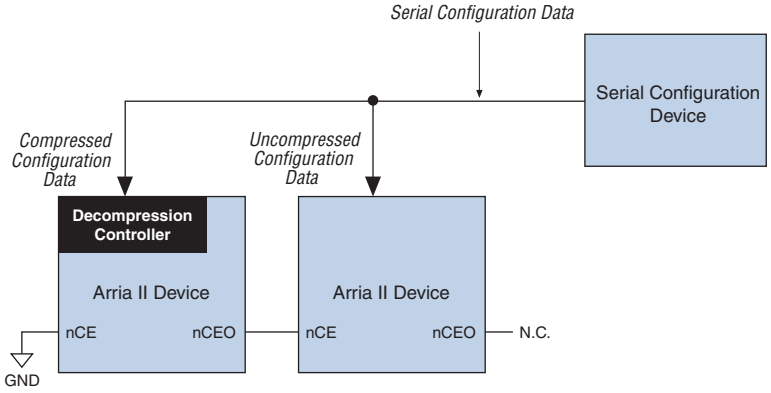

You can also enable compression when creating programming files from the **Convert Programming Files** window. To do this, follow these steps:

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

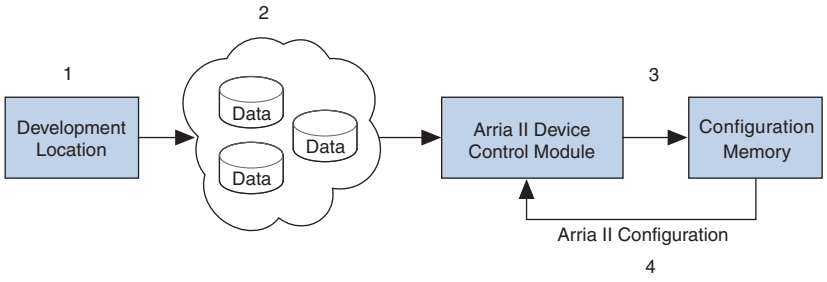
The dedicated remote system upgrade circuitry in Arria II devices manages remote configuration and provides error detection, recovery, and status information. User logic or a Nios II processor implemented in the Arria II device logic array provides access to the remote configuration data source and an 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.)

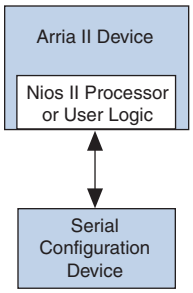
**Figure 9–21. Functional Diagram of Arria II Remote System Upgrade**




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.

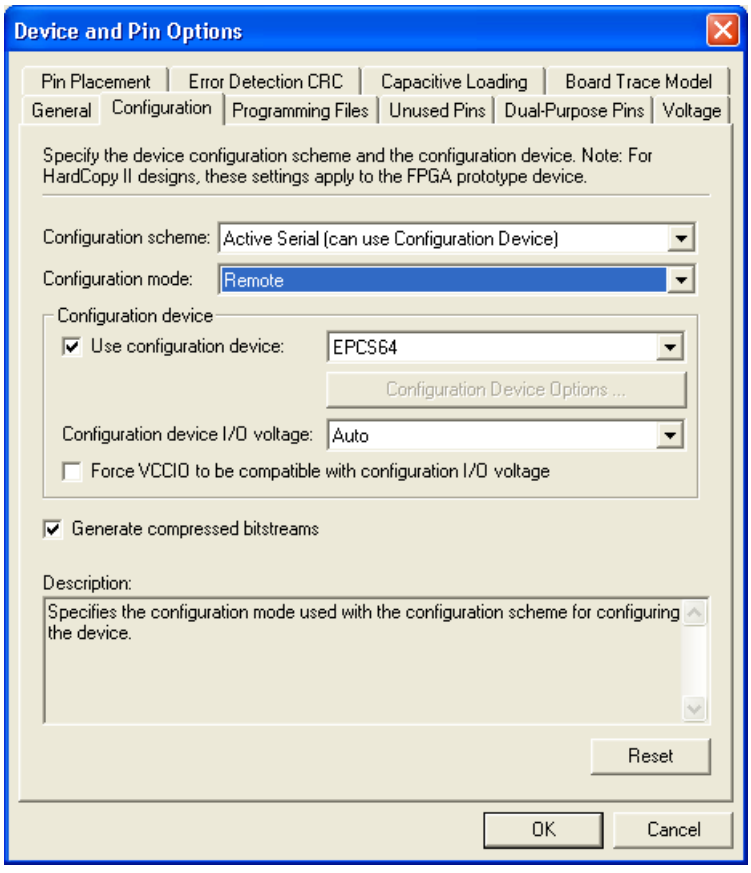
When using AS mode, you must select **remote update mode** in the Quartus II software and insert the ALTREMOTE_UPDATE megafunction to access the circuitry. For more information, refer to "ALTREMOTE_UPDATE Megafunction" on page 9–60.

## 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.

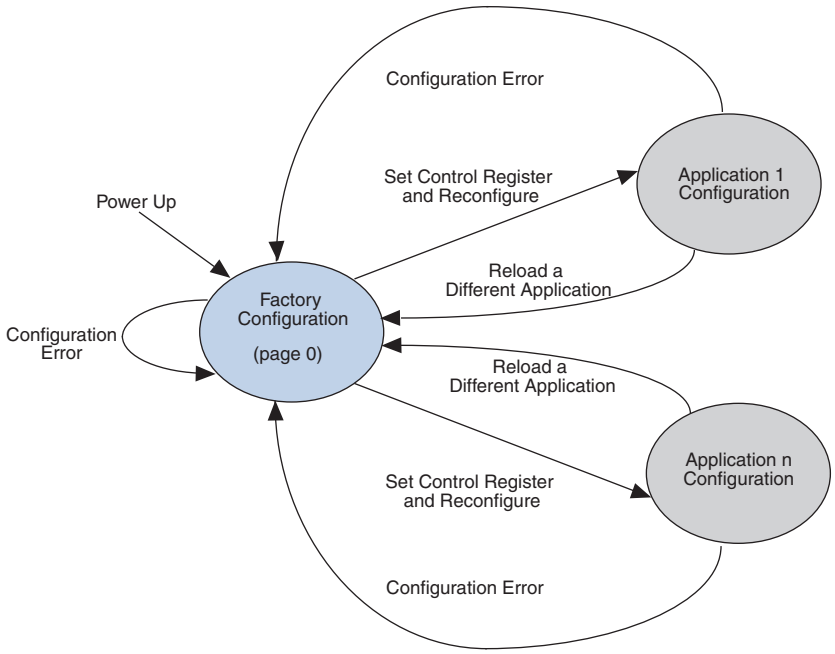
When an Arria II device is first powered up in remote update mode, it loads the factory configuration located at page zero (page registers `PGM[23..0] = 24'b0`). Always store the factory configuration image for your system at page address zero. This corresponds to the start address location 0×000000 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.

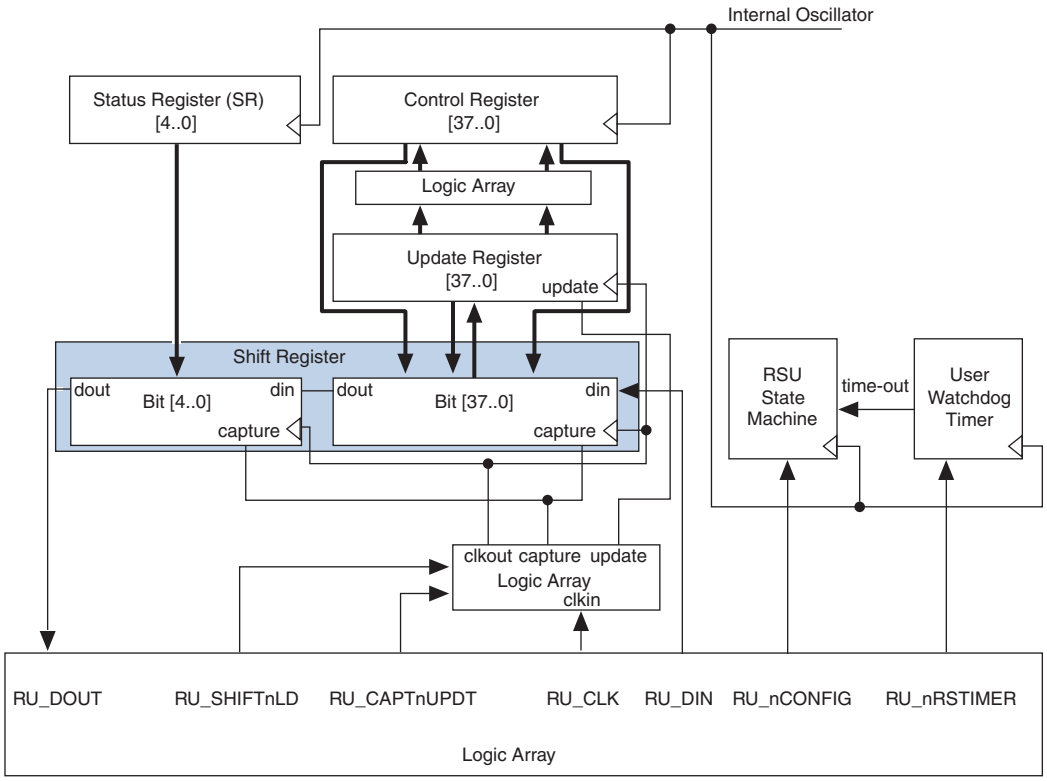
# Dedicated Remote System Upgrade Circuitry

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. |

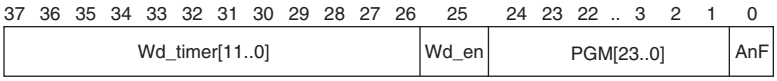
The remote system upgrade control and status registers are clocked by the 10-MHz internal oscillator (the same oscillator that controls the user watchdog timer). However, the remote system upgrade shift and update registers are clocked by the user clock input (`RU_CLK`).

### 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**

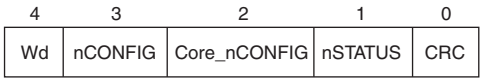


Table 9–21 lists the status register contents for remote system upgrade.

**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.

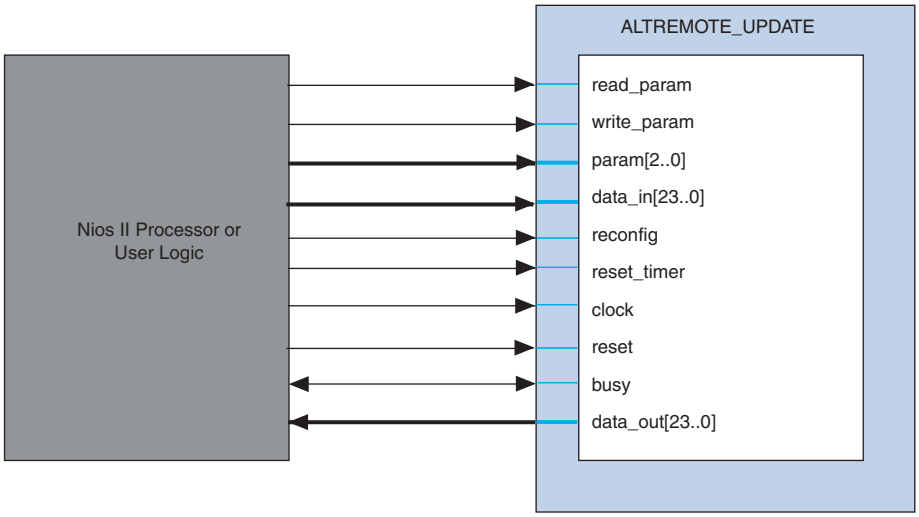
Use the ALTREMOTE_UPDATE megafunction option in the Quartus II software as the interface between the remote system upgrade circuitry and the device logic array interface. Using the megafunction block instead of creating your own logic saves design time and offers more efficient logic synthesis and device 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.<br>Tamper resistant if volatile tamper protection bit is set. *(2)* | Secure against copying and reverse engineering.<br>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.

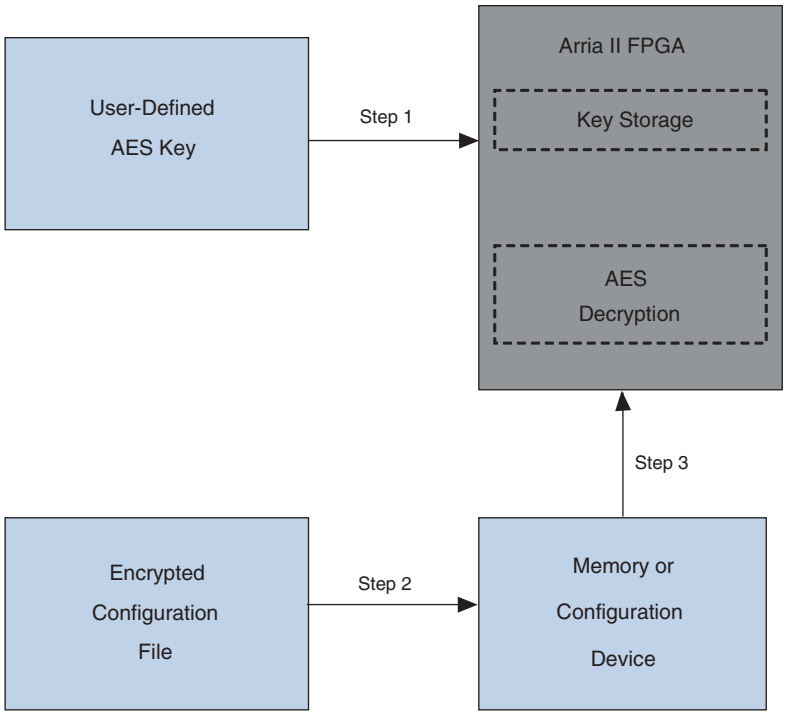
2. Encrypt the configuration file and store it in the external memory.

   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:**

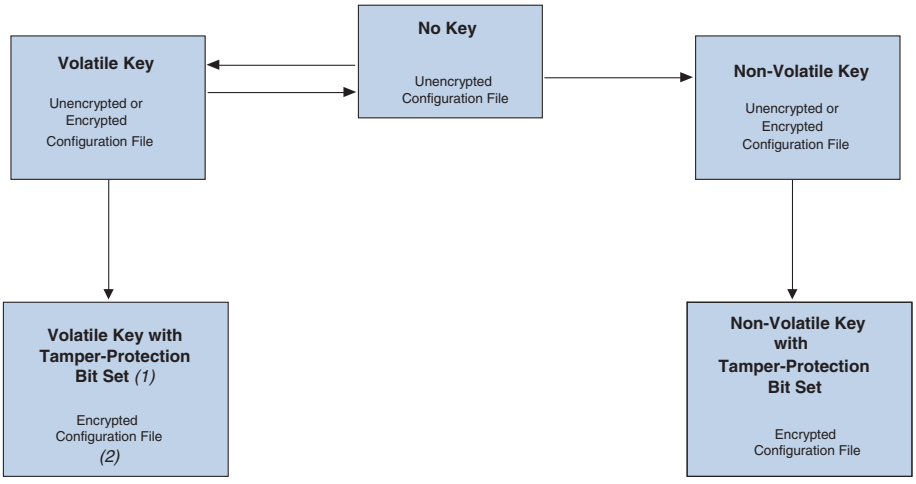
(1) In No key operation, only the unencrypted configuration file is supported.
(2) Arria II GZ devices do not support this feature.
(3) The tamper protection bit setting does not prevent the device from being reconfigured.

## 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)<br>■ FPP with AES (and/or with decompression)<br>■ Remote update AS or fast AS with AES (and/or with decompression)<br>■ 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)<br>■ FPP with AES (and/or with decompression)<br>■ Remote update AS or fast AS with AES (and/or with decompression)<br>■ 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)<br>■ FPP with AES (and/or with decompression)<br>■ Remote update AS or fast AS with AES (and/or with decompression)<br>■ 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.<br>■ 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.<br>■ Updated Figure 9–16 and Figure 9–17.<br>■ Updated "Configuration" and "FPP Configuration Using a MAX II Device as an External Host" sections.<br>■ Minor text edits. |
| June 2011 | 4.1 | ■ Updated Table 9–9, Table 9–10, Table 9–12, Table 9–18, and Table 9–23.<br>■ Updated the "Programming Serial Configuration Devices" and "Configuration Data Decompression" sections.<br>■ Removed references to the "ByteBlaster MV" and "MasterBlaster" cables as they are discontinued.<br>■ Minor text edits. |
| December 2010 | 4.0 | ■ Updated for the Quartus II software version 10.1 release.<br>■ Added Arria II GZ devices information.<br>■ Minor text edits. |
| July 2010 | 3.0 | Updated for Arria II GX v10.0 release:<br>■ Updated Table 9–9 and Table 9–17.<br>■ Updated Figure 9–4, Figure 9–5, Figure 9–13, Figure 9–16, Figure 9–17, Figure 9–21, and Figure 9–30.<br>■ Updated "AS and Fast AS Configuration (Serial Configuration Devices)" and "Flexible Security Key Storage" sections.<br>■ Added "Guidelines for Connecting Serial Configuration Device to Arria II Devices on an Active Serial Interface" section.<br>■ Minor text edits. |
| November 2009 | 2.0 | Updated for Arria II GX v9.1 release:<br>■ Updated Table 9–3, Table 9–10, Table 9–11, Table 9–13.<br>■ Updated Figure 9–2, Figure 9–3, and Figure 9–6.<br>■ Updated "VCCPD Pins", "JTAG Configuration", "Remote System Upgrade Mode", "Remote System Upgrade State Machine", "User Watchdog Timer" sections.<br>■ 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.<br>■ Updated Figure 9–6, Figure 9–11, and Figure 9–16.<br>■ Updated "VCCIO Pins for I/O Banks 3C and 8C", "FPP Configuration Using an External Host", and "Programming Serial Configuration Devices" sections.<br>■ 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

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.

# 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.<br>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.<br>■ 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.<br>■ If an error occurs, the search engine finds the location of the error.<br>■ The error messages can be shifted out through the JTAG instruction or core interface logics while the error detection block continues running.<br>■ 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.<br>■ 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.<br>■ 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.<br>■ 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**

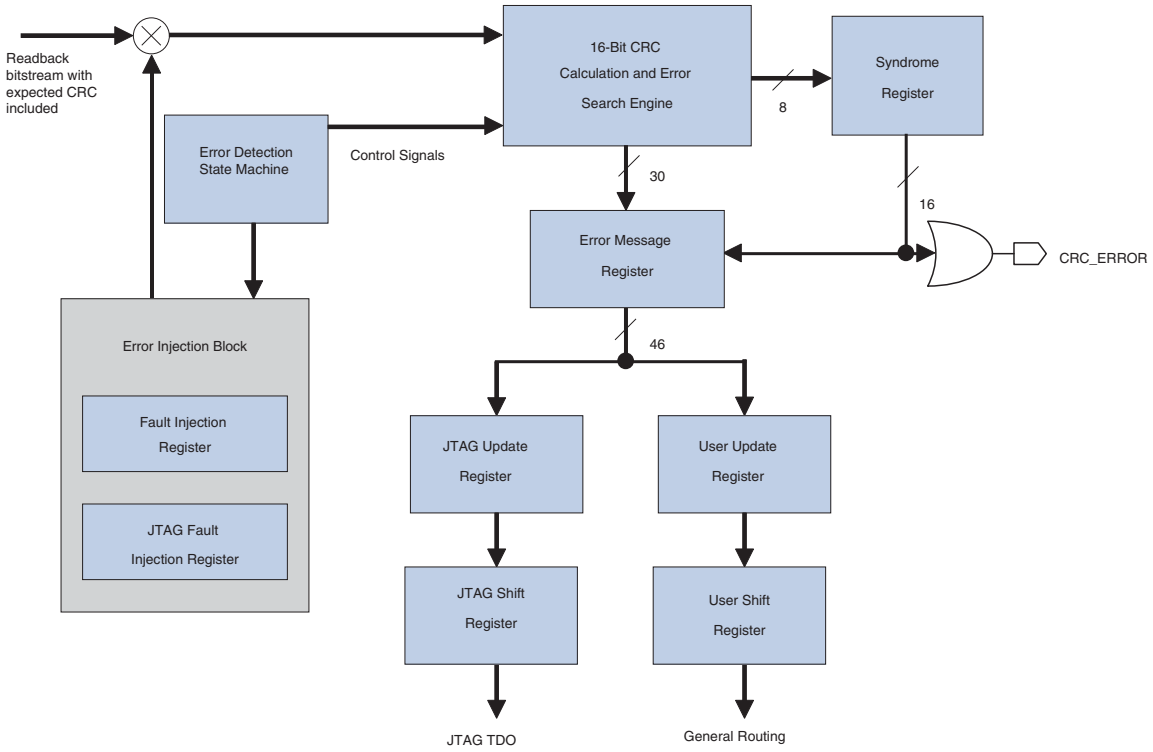


Table 10–5 lists the registers shown in Figure 10–1.

**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.<br>■ Minor text edits. |
| December 2010 | 4.0 | ■ Updated for the Quartus II software version 10.1 release.<br>■ Added Arria II GZ devices information.<br>■ Minor text edits. |
| July 2010 | 3.0 | Updated for Arria II GX v10.0 release:<br>■ 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:<br>■ Updated Table 10–7 and Table 10–8.<br>■ 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:



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.

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**

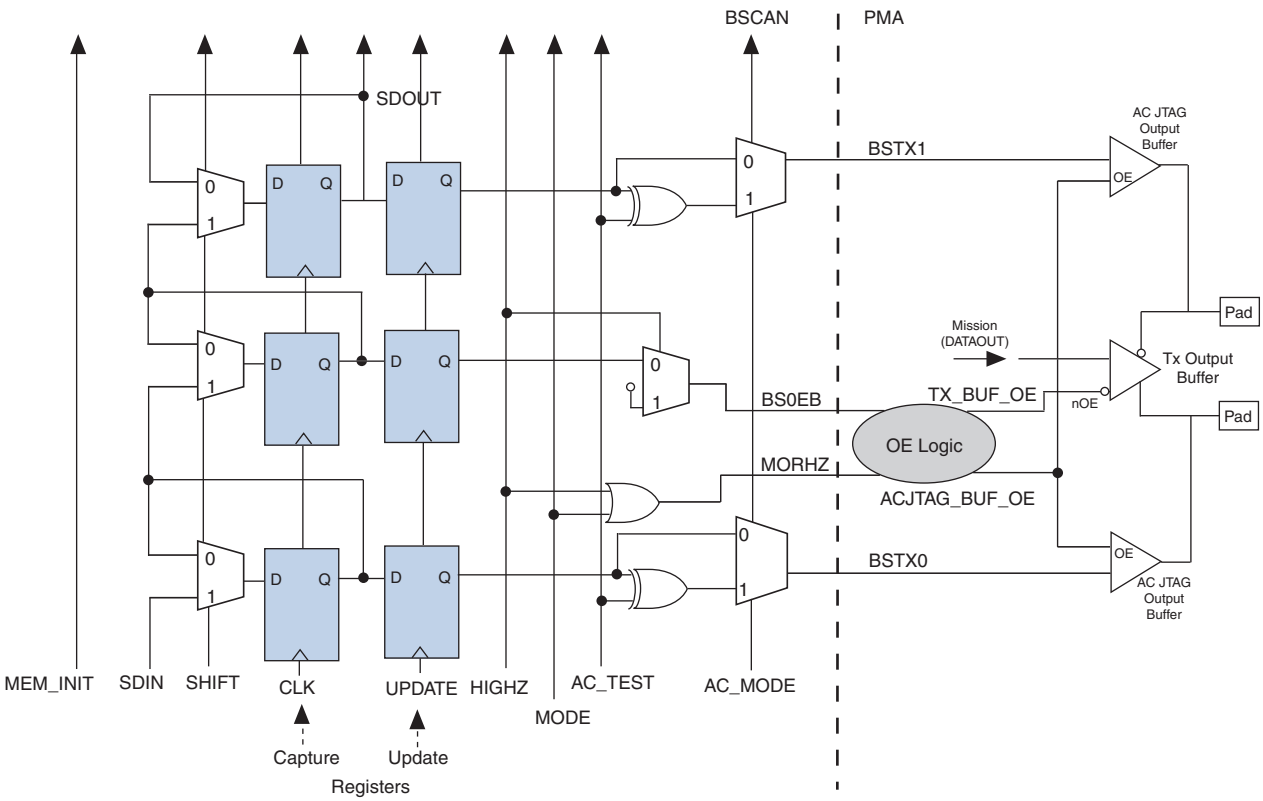
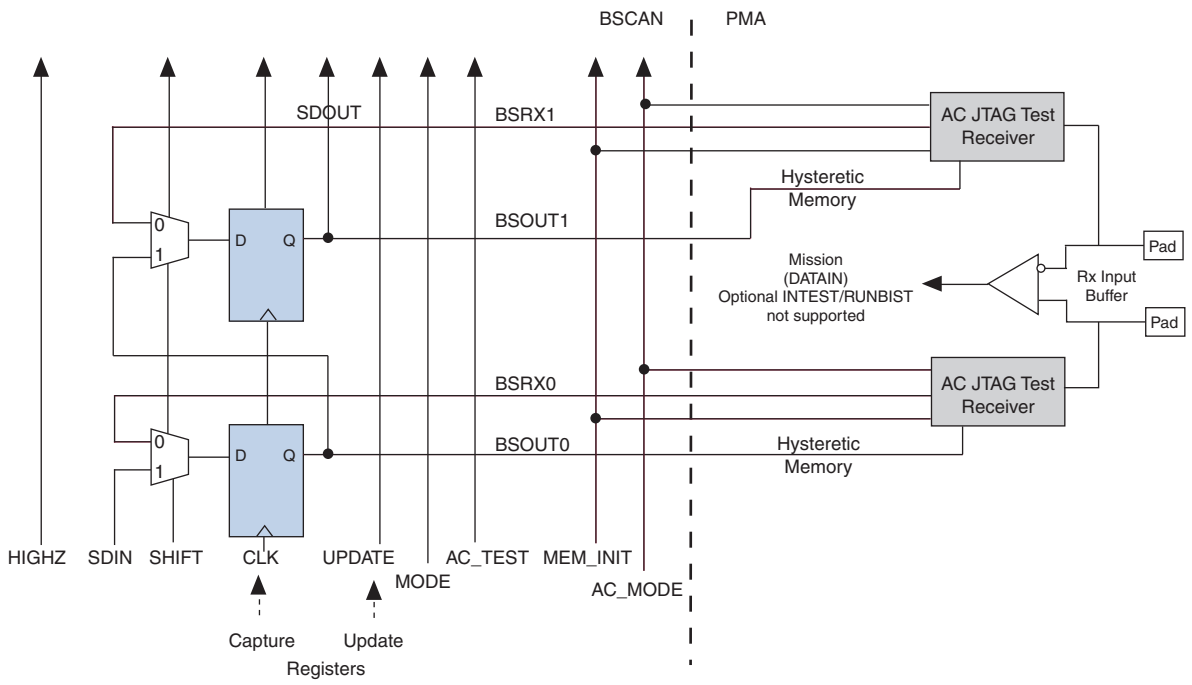


Figure 11–2 shows the Arria II GX HSSI receiver/input clock buffer BSC.

**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.<br>■ Added Arria II GZ devices information.<br>■ Added "BST Architecture for Arria II Devices" and "Disabling IEEE Std. 1149.1 BST Circuitry" sections.<br>■ Added Table 11–3 and Table 11–5.<br>■ Updated Table 11–1 and Table 11–2.<br>■ Minor text edits. |
| July 2010 | 3.0 | Updated for Arria II GX v10.0 release:<br>■ Updated "BST Operation Control" section.<br>■ Minor text edits. |
| November 2009 | 2.0 | Updated for Arria II GX v9.1 release:<br>■ Updated Table 11–1 and Table 11–2.<br>■ Updated "I/O Voltage Support in a JTAG Chain" section.<br>■ 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

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.

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.<br>■ Updated "Insertion or Removal of an Arria II Device from a Powered-Up System" and "Hot-Socketing Feature Implementation" sections.<br>■ Minor text edits. |
| December 2010 | 3.0 | ■ Updated for the Quartus II software version 10.1 release.<br>■ Added Arria II GZ devices information.<br>■ 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.<br>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`.<br>Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`.<br>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<br>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. |