Arria V Device Handbook

Volume 1: Device Overview and Datasheet

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

AV-5V1-1.3

Document last updated for Altera Complete Design Suite version: 11.1
Document publication date: February 2012

Contents

Additional Information

Chapter Revision Dates

The chapters in this document, Volume 1: Device Overview and Datasheet, 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 V Device Family
Revised: *February 2012*
Part Number: *AV51001-1.3*

Chapter 2. Device Datasheet for Arria V Devices
Revised: *February 2012*
Part Number: *AV-51002-1.3*

1. Overview for the Arria V Device Family

AV51001-1.3

Built on the 28-nm low-power process technology, Arria® V devices offer the lowest power and lowest system cost for mainstream applications. Arria V devices include unique innovations such as the lowest static power in its class, the lowest power transceivers of any midrange family, support for serial data rates up to 10.3125 gigabits per second (Gbps), a powerful collection of integrated hard intellectual property (IP), and a power-optimized core architecture, making Arria V devices ideal for the following applications:

- Power sensitive wireless infrastructure equipment
- 20G/40G bridging, switching, and packet processing applications
- High-definition video processing and image manipulation
- Intensive digital signal processing (DSP) applications

Arria V devices are available in the following variants:

- Arria V GX—FPGA with integrated 6-Gbps transceivers, this variant provides bandwidth, cost, and power levels that are optimized for high-volume data and signal-processing applications.
- Arria V GT—FPGA with integrated 10-Gbps transceivers, this variant provides enhanced high-speed serial I/O bandwidth for cost-sensitive data and signal processing applications.
- Arria V SX—system-on-a-chip (SoC) FPGA with integrated Arria V FPGA and ARM®-based hard processor system (HPS).
- Arria V ST—SoC FPGA with integrated Arria V FPGA, ARM-based HPS, and 10-Gbps transceivers.

The Arria V SoC FPGA variants feature an FPGA integrated with an HPS that consists of a dual-core ARM Cortex™-A9 MPCore™ processor, a rich set of peripherals, and a shared multiport SDRAM memory controller.

The unique feature set in Arria V devices was chosen to optimize power, cost, and performance. These features include a redesigned adaptive logic module (ALM), distributed memory, new 10-Kbit (M10K) internal memory blocks, variable-precision DSP blocks, and fractional clock synthesis phase-locked loops (PLLs) with a highly flexible clocking network, all interconnected by a power-optimized MultiTrack routing architecture.

Arria V devices provide interface support flexibility with up to 10-Gbps transceivers, 1.25-Gbps **LVDS**, 1.333-Gbps memory interfaces with low latency, and support for all mainstream single-ended and differential I/O standards, including 3.3 V. Arria V devices also offer the lowest system cost by requiring only three power supplies to operate the devices and a thermal composite flip chip ball-grid array (BGA) packaging option. Arria V devices also support innovative features, such as configuration via protocol (CvP), partial reconfiguration, and design security.

Arria V devices provide the power, features, and cost you require to succeed with your designs. With these innovations, Arria V devices deliver ideal performance and capability for a wide range of applications.

Arria V Feature Summary

Table 1–1 lists the Arria V device features.

Table 1–1. Feature Summary for Arria V Devices (Part 1 of 3)

Feature	Details
Technology	■ 28-nm TSMC low-power process technology ■ Lowest static power in its class (less than 800 mW for 500 K logic elements (LEs) at 85°C junction under typical conditions) ■ 1.1-V core nominal voltage
Lowest-power serial transceivers of any midrange FPGA	■ 611-Mbps to 10.3125-Gbps integrated transceivers ■ Transmit pre-emphasis and receiver equalization ■ Dynamic reconfiguration of individual channels
FPGA General-purpose I/Os (GPIOs)	■ 1.25-Gbps **LVDS** ■ 667-MHz/1.333-Gbps external memory interface ■ On-chip termination (OCT) ■ 3.3-V support
Embedded transceiver hard IP	■ Custom implementation up to 10.3125 Gbps ■ PCI Express® (PCIe®) Gen1 and Gen2 ■ Gbps Ethernet (GbE) and XAUI physical coding sublayer (PCS) ■ Common Public Radio Interface (CPRI) PCS ■ Gigabit-capable passive optical network (GPON) PCS

Table 1–1. Feature Summary for Arria V Devices (Part 2 of 3)

Feature	Details
HPS (Arria V SX and ST devices only)	■ Dual-core ARM Cortex-A9 MPCore processor. Up to 800 MHz maximum frequency that supports symmetric and asymmetric multiprocessing ■ Interface peripherals—10/100/1000 Ethernet media access control (MAC), USB 2.0 On-The-Go (OTG) controller, Quad SPI flash controller, NAND flash controller, and SD/MMC/SDIO controller, UART, serial peripheral interface (SPI), I2C interfaces, and up to 86 GPIO interfaces ■ System peripherals—general-purpose and watchdog timers, direct memory access (DMA) controller, FPGA configuration manager, and clock and reset managers ■ On-chip RAM and boot ROM ■ HPS–FPGA bridges—include the FPGA-to-HPS, HPS-to-FPGA, and lightweight HPS-to-FPGA bridges that allow the FPGA fabric to master transactions to slaves in the HPS, and vice versa ■ FPGA-to-HPS SDRAM controller subsystem—provides a configurable interface to the multiport front end of the HPS SDRAM controller ■ ARM CoreSight™ JTAG debug, trace port, and on-chip trace storage ■ Three fractional PLLs
Physical medium attachment (PMA) with soft PCS	■ 10GBASE-R ■ 9.8304-Gbps CPRI
High-performance core fabric	■ Enhanced ALM with four registers ■ Improved routing architecture to reduce congestion and improve compilation time
Variable-precision DSP blocks	■ Natively supports three-signal processing precision ranging from 9 x 9, 18 x 19, or 27 x 27 in the same DSP block ■ 64-bit accumulator and cascade for systolic finite impulse responses (FIRs) ■ Embedded internal coefficient memory ■ Pre-adder/subtractor improves efficiency
Internal memory blocks	■ M10K, 10 Kbit with soft error correction code (ECC) ■ Memory logic array block (MLAB), 640-bit distributed LUTRAM—you can use up to 25% of the LEs as MLAB memory ■ Hardened double data rate3 (DDR3) and DDR2 memory controllers
High-resolution Fractional PLLs	■ Integer mode and fractional mode ■ Precision clock synthesis, clock delay compensation, and zero delay buffering (ZDB)
Clock networks	■ 625-MHz global clock network ■ Global, quadrant, and peripheral clock networks ■ Unused clock networks can be powered down to reduce dynamic power

Table 1–1. Feature Summary for Arria V Devices (Part 3 of 3)

Feature	Details
Configuration	■ Partial and dynamic reconfigurations ■ CvP ■ Configuration via HPS ■ Serial and parallel flash interface ■ Enhanced advanced encryption standard (AES) design security features ■ Tamper protection ■ Remote system upgrade
Packaging	■ Thermal composite flip chip BGA packaging ■ Multiple device densities with identical package footprints for seamless migration between different device densities ■ Lead, lead-free (Pb-free), and RoHS-compliant options

Arria V Family Plan

Arria V devices offer various thermal composite flip chip BGA packaging options with differing price and performance points. Table 1–2 and Table 1–3 list the Arria V devices features.

Table 1–2. Maximum Resource Counts for Arria V GX Devices —*Preliminary*

Feature	Arria V GX Device							
	5AGXA1	5AGXA3	5AGXA5	5AGXA7	5AGXB1	5AGXB3	5AGXB5	5AGXB7
ALMs	28,302	56,100	71,698	91,680	113,208	136,880	158,491	190,240
LE (K)	75	148	190	242	300	362	420	504
M10K memory blocks	800	1,051	1,180	1,366	1,510	1,726	2,054	2,414
MLAB memory (Kbit)	463	873	1,173	1,448	1,852	2,098	2,532	2,906
Block memory (Kbit)	8,000	10,510	11,800	13,660	15,100	17,260	20,540	24,140
Variable-precision DSP blocks	240	396	600	800	920	1,045	1,092	1,139
18 x 19 multipliers	480	792	1,200	1,600	1,840	2,090	2,184	2,278
Fractional PLLs (1)	10	10	12	12	12	12	16	16
GPIO	480	480	544	544	704	704	704	704
LVDS transmitter (TX) (2)	68	68	120	120	160	160	156	160
LVDS receiver (RX) (2)	80	80	136	136	176	176	172	176
PCIe hard IP blocks	1	1	2	2	2	2	2	2
Hard memory controllers	2	2	4	4	4	4	4	4

Notes to Table 1–2:

(1) The total number of available fractional PLLs is a combination of general-purpose and transceiver PLLs. Transceiver fractional PLLs that are not used by the transceiver I/O can be used as general-purpose fractional PLLs.

(2) For the LVDS channels count for each package, refer to the *High-Speed Differential I/O Interfaces with DPA in Arria V Devices* chapter.

Table 1–3. Maximum Resource Counts for Arria V GT, SX, and ST Devices—*Preliminary*

Feature	Arria V GT Device		Arria V SX Device		Arria V ST Device	
	5AGTD3	5AGTD7	5ASXB3	5ASXB5	5ASTD3	5ASTD5
ALMs	136,880	190,240	132,075	174,340	132,075	174,340
LE (K)	362	504	350	462	350	462
M10K memory blocks	1,726	2,414	1,729	2,282	1,729	2,282
MLAB memory (Kb)	2,098	2,906	2,014	2,658	2,014	2,658
Block memory (Kb)	17,260	24,140	17,288	22,820	17,288	22,820
Variable-precision DSP blocks	1,045	1,156	809	1,068	809	1,068
18 x 19 multipliers	2,090	2,312	1,618	2,186	1,618	2,186
FPGA Fractional PLLs (1)	12	16	TBD	TBD	TBD	TBD
HPS PLLs (1)	—	—	TBD	TBD	TBD	TBD
FPGA GPIO	704	704	528	528	528	528
HPS I/O	—	—	216	216	216	216
LVDS TX (2)	160	160	120	120	120	120
LVDS RX (2)	176	176	120	120	120	120
PCIe hard IP blocks	2	2	2	2	2	2
Hard memory controllers	4	4	3	3	3	3
HPS memory controllers	—	—	1	1	1	1
ARM Cortex–A9 MPCore processor	—	—	Dual-core	Dual-core	Dual-core	Dual-core

Notes to Table 1–3:

(1) The total number of available fractional PLLs is a combination of general-purpose and transceiver PLLs. Transceiver fractional PLLs, when not used by the transceiver I/O, can be used as a general-purpose fractional PLL.

(2) For the LVDS channels count for each package, refer to the *High-Speed Differential I/O Interfaces with DPA in Arria V Devices* chapter.

Table 1–4 lists the Arria V package plan. The package plan shows the GPIO counts, the maximum number of 6-Gbps transceivers available, and the maximum number of 10-Gbps transceivers available per density and package. Various combinations of 6-Gbps and 10-Gbps transceiver counts are available.

Table 1–4. Package Plan for Arria V Devices —*Preliminary* *(1)*

Varients	Devices	F672 (27 mm) Flip Chip		F896 (31 mm) Flip Chip			F1152 (35 mm) Flip Chip			F1517 (40 mm) Flip Chip		
		GPIO	XCVR	GPIO *(2)*	HPS I/O	XCVR	GPIO	HPS I/O	XCVR	GPIO	HPS I/O	XCVR
Arria V GX *(3)*	5AGXA1	336	9	480	—	12	—	—	—	—	—	—
	5AGXA3	336	9	480	—	12	—	—	—	—	—	—
	5AGXA5	336	9	384	—	18	544	—	24	—	—	—
	5AGXA7	336	9	384	—	18	544	—	24	—	—	—
	5AGXB1	—	—	384	—	18	544	—	24	704	—	24
	5AGXB3	—	—	384	—	18	544	—	24	704	—	24
	5AGXB5	—	—	—	—	—	544	—	24	704	—	36
	5AGXB7	—	—	—	—	—	544	—	24	704	—	36
Arria V GT *(4)*, *(5)*	5AGTD3	—	—	384	—	12, 2	544	—	12, 4	704	—	12, 4
	5AGTD7	—	—	—	—	—	544	—	12, 4	704	—	12, 8
Arria V SX *(3)*	5ASXB3	—	—	178	216	12	350	216	18	528	216	30
	5ASXB5	—	—	178	216	12	350	216	18	528	216	30
Arria V ST *(4)*, *(5)*	5ASTD3	—	—	178	216	6, 2	350	216	12, 2	528	216	12, 6
	5ASTD5	—	—	178	216	6, 2	350	216	12, 2	528	216	12, 6

Notes to Table 1–4:

(1) The arrows indicate the package vertical migration capability. Vertical migration allows you to migrate across device densities for devices having the same dedicated pins, configuration pins, and power pins for a given package.

(2) In the F896 package, the PCIe hard IP block on the right side of the 5AGXA5, 5AGXA7, 5AGXB1, 5AGXB3, and 5AGTD3 devices supports x1 for Gen1 and Gen2 data rates.

(3) The transceiver counts listed are for 6-Gbps transceivers.

(4) The transceiver counts listed are for 6-Gbps and 10-Gbps transceivers, respectively.

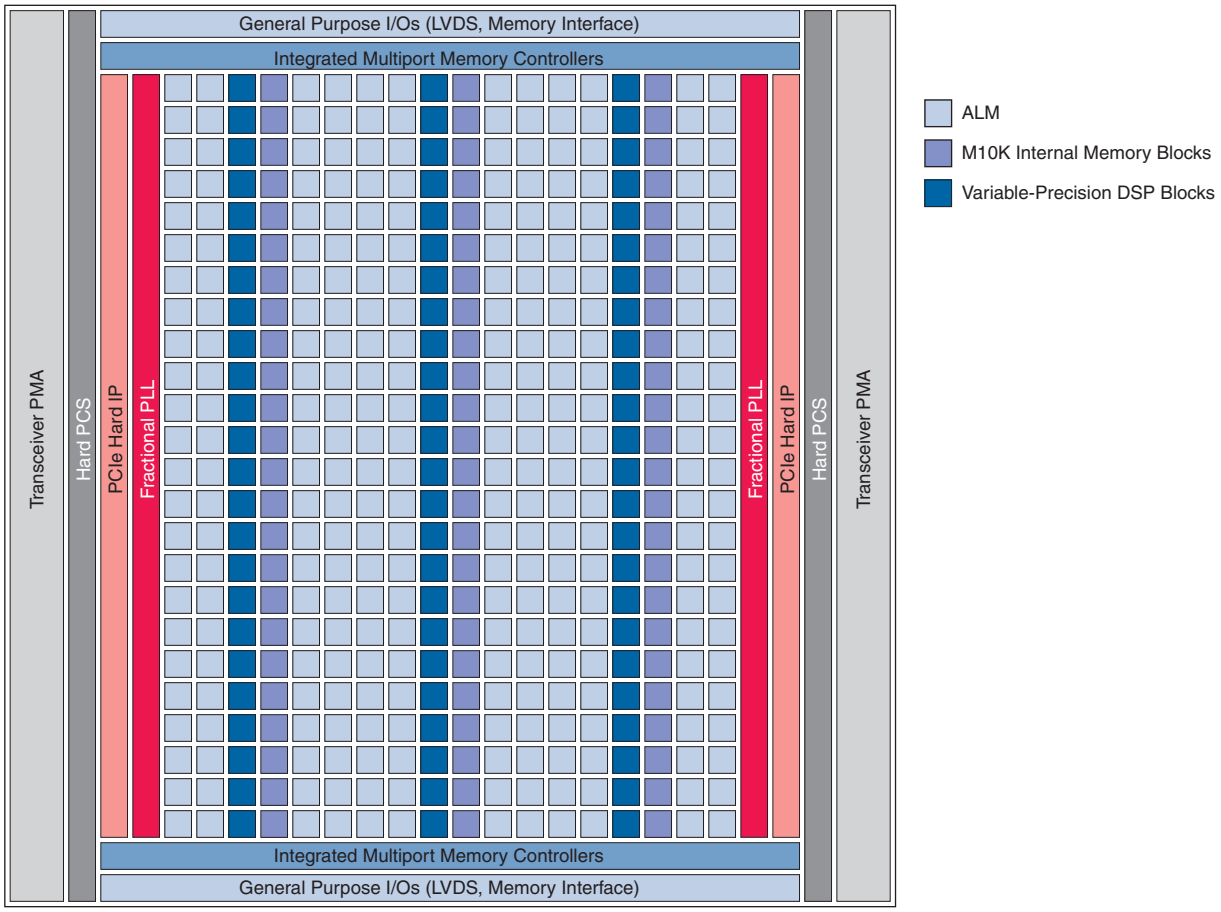
(5) You can alternatively configure any pair of 10-Gbps channels as six 6-Gbps channels. For instance, you can alternatively configure the 5AGTD7 device in the F1517 package as eighteen 6-Gbps and six 10-Gbps, twenty-four 6-Gbps and four 10-Gbps, or thirty 6-Gbps and two 10-Gbps channels.

Low-Power Serial Transceivers

Arria V devices deliver the industry's lowest power 10-Gbps transceivers at less than 140 mW and 6-Gbps transceivers at less than 100 mW power consumption per channel. Arria V transceivers are designed to be standard compliant for a wide range of protocols and data rates.

The transceivers are positioned on the left and right outer edges of the device, as shown in Figure 1–1.

Figure 1–1. Device Chip Overview for Arria V Devices *(1), (2)*



Notes to Figure 1–1:

(1) This figure represents an Arria V device with transceivers. Other Arria V devices may have a different floor plan than the one shown here.

(2) This figure is a graphical representation of a top view of the silicon die, which corresponds to a reverse view for flip chip packages.

PMA Support

To prevent core and I/O noise from coupling into the transceivers, the PMA block is isolated from the rest of the chip, ensuring optimal signal integrity. The transceiver channels consist of the PMA, PCS, and clock networks. You can also use the unused receiver PMA channels as additional transmit PLLs.

Table 1–5 lists the transceiver PMA features.

Table 1–5. Transceiver PMA Features for Arria V Devices

Features	Capability
Backplane support	Up to 16” FR4 PCB fabric drive capability at up to 6.5536 Gbps
Chip-to-chip support	Up to 10.3125 Gbps
PLL-based clock recovery	Superior jitter tolerance
Programmable serializer and deserializer (SERDES)	Flexible SERDES width
Equalization and pre-emphasis	Up to 6 dB of pre-emphasis and 4 dB of equalization
Ring oscillator transmit PLLs	611 Mbps to 10.3125 Gbps
Input reference clock range	27 MHz to 710 MHz
Transceiver dynamic reconfiguration	Allows reconfiguration of single channels without affecting operation of other channels

PCS Support

The Arria V core logic connects to the PCS through an 8-, 10-, 16-, 20-, 32-, or 40-bit interface, depending on the transceiver data rate and protocol. Arria V devices contain PCS hard IP to support PCIe Gen1 and Gen2, XAUI, GbE, Serial RapidIO® (SRIO), and CPRI protocols. All other standard and proprietary protocols from 611 Mbps to 6.5536 Gbps are supported through the custom double-width mode (up to 6.5536 Gbps) and custom single-width mode (up to 3.75 Gbps) transceiver PCS hard IP. A dedicated 80-bit interface to the core logic connects directly from the PMA, bypassing the PCS hard IP, to support all protocols beyond 6.5536 Gbps up to 10.3125 Gbps.

Table 1–6 lists the transceiver PCS features.

Table 1–6. Transceiver PCS Features for Arria V Devices (Part 1 of 2)

PCS Support (1)	Data Rates (Gbps)	Transmitter Data Path	Receiver Data Path
Custom single- and double-width modes	0.61 to ~6.5536	Phase compensation FIFO, byte serializer, and 8B/10B encoder	Word aligner, 8B/10B decoder, byte deserializer, and phase compensation FIFO
PCIe Gen1: x1, x2, x4, x8 PCIe Gen2: x1, x2, x4 (2)	2.5 and 5.0	The same as custom single- and double-width modes, plus PIPE 2.0 interface to the core logic	The same as custom single- and double-width modes, plus rate match FIFO and PIPE 2.0 interface to the core logic
GbE	1.25	The same as custom single- and double-width modes	The same as custom single- and double-width modes, plus rate match FIFO

Table 1–6. Transceiver PCS Features for Arria V Devices (Part 2 of 2)

PCS Support (1)	Data Rates (Gbps)	Transmitter Data Path	Receiver Data Path
XAUI	3.125	The same as custom single- and double-width modes, plus the XAUI state machine for bonding four channels	The same as custom single- and double-width modes, plus the XAUI state machine for realigning four channels, and deskew FIFO circuitry
SRIO	1.25 to 6.25	The same as custom single- and double-width modes	The same as custom single- and double-width modes
SDI	0.27 (3), 1.485, 2.97	Phase compensation FIFO, byte serializer	Byte deserializer and phase compensation FIFO
Serial ATA	1.5, 3.0, 6.0	Phase compensation FIFO, byte serializer, 8B/10B encoder	Phase compensation FIFO, byte deserializer, word aligner, and 8B/10B decoder
CPRI (4)	0.6144 to 6.144	The same as custom single- and double-width modes, plus the TX deterministic latency	The same as custom single- and double-width modes, plus the RX deterministic latency
GPON (5)	1.25 and 2.5	Phase compensation FIFO and byte serializer	Phase compensation FIFO and byte deserializer

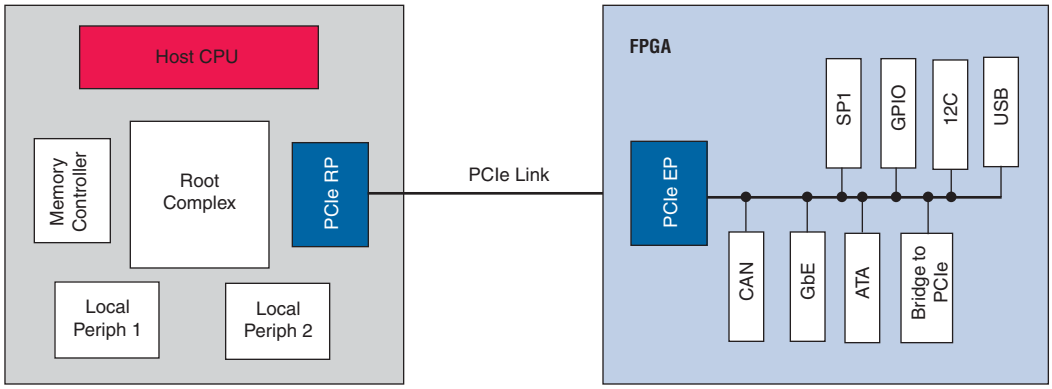
Notes to Table 1–6:

(1) Data rates above 6.5536 Gbps up to 10.3125 Gbps, such as 10GBASE-R, are supported through soft PCS.
(2) PCIe Gen2 is supported with the PCIe hard IP only.
(3) The 0.27-Gbps data rate is supported using oversampling user logics that you must implement in the FPGA fabric.
(4) CPRI data rates above 6.5536 Gbps, such as 9.8304 Gbps, are supported through soft PCS.
(5) The GPON standard does not support burst mode.

PCIe Gen1 and Gen2 Hard IP

Arria V devices contain PCIe hard IP designed for performance, ease-of-use, and increased functionality. The PCIe hard IP consists of the PHY MAC, data link, and transaction layers. The PCIe hard IP supports PCIe Gen2 end point and root port for up to x4 lane configurations, and PCIe Gen1 end point and root port for up to x8 lane configurations. PCIe endpoint support includes multifunction support for up to eight functions, as shown in Figure 1–2.

Figure 1–2. PCIe Multifunction for Arria V Devices



The Arria V PCIe hard IP operates independently from the core logic, which allows the PCIe link to wake up and complete link training in less than 100 ms, while the Arria V device completes loading the programming file for the rest of the device. In addition, the Arria V PCIe hard IP has improved end-to-end data path protection using ECC.

FPGA GPIOs

Arria V devices offer highly configurable GPIOs. The following list describes the many features of the GPIOs:

- Programmable bus hold and weak pull-up.
- **LVDS** output buffer with programmable differential output voltage (V_{OD}) and programmable pre-emphasis.
- Dynamic on-chip parallel termination (R_T OCT) for all I/O banks with OCT calibration to limit the termination impedance variation.
- On-chip dynamic termination to swap between serial and parallel termination, depending on whether there is reading or writing on a common bus for signal integrity.
- Configurable unused voltage reference (`VREF`) pins as user I/Os.
- Easy timing closure support using the hardened read FIFO in the input register path, and delay-locked loop (DLL) delay chain with fine and coarse architecture.

External Memory

Arria V devices support up to four hardened memory controllers for DDR3 and DDR2 SDRAM. Each controller supports 8- to 32-bit components up to 4 gigabits (Gb) in density with two-chip select and optional ECC. Arria V devices do not support DDR3 SDRAM leveling.

Arria V devices also support soft memory controllers for DDR3, DDR2, LPDDR2, and LPDDR SDRAM, RLDRAM II, QDR II, and QDR II+ SRAM for maximum flexibility.

Table 1–7 lists the external memory interface block performance.

Table 1–7. External Memory Interface Performance for Arria V Devices

Interface	Voltage (V)	Hard Controller (MHz)	Soft Controller (MHz)
DDR3 SDRAM	1.5	533	667
	1.35	533	667
	1.25	400	400
DDR2 SDRAM	1.8	400	400
	1.5	400	400
RLDRAM II	1.8	(1)	400
QDR II+ SRAM	1.8	(1)	400
	1.5	(1)	400
QDR II SRAM	1.8	(1)	400
	1.5	(1)	400
DDR II+ SRAM (2)	1.8	(1)	400
	1.5	(1)	400
LPDDR SDRAM (2)	1.8	(1)	200
LPDDR2 SDRAM (2)	1.2	(1)	400

Notes to Table 1–7:

(1) These memory interfaces are not supported in the hard memory controller.

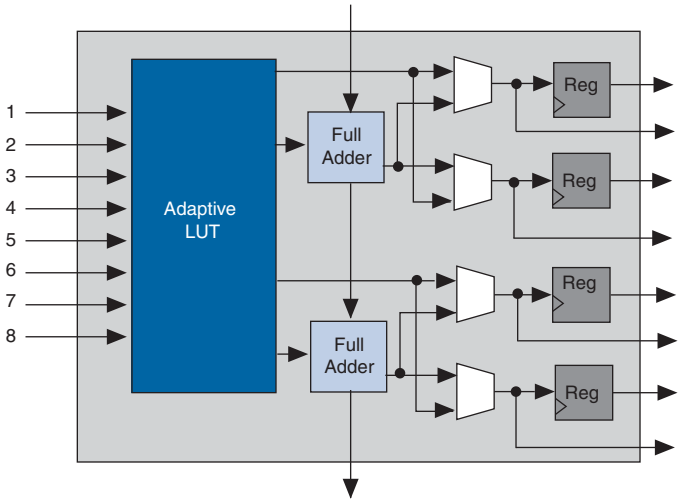
(2) These memory interfaces are not available as Altera® IP.

ALM

Arria V devices use a 28-nm ALM as the basic building block of the device fabric. The ALM shown in Figure 1–3 uses an 8-input fracturable look-up table (LUT) with four dedicated registers to help improve timing closure in register-rich designs and achieve an even higher design packing capability than previous generations.

You can configure up to 25% of the ALMs in Arria V devices as distributed MLABs. For more information, refer to "Embedded Memory" on page 1–14.

Figure 1–3. ALM for Arria V Devices



Variable-Precision DSP Block

Arria V devices feature a variable-precision DSP block that you can configure to support signal processing with precision ranging from 9 x 9, 18 x 19, and 27 x 27 bits natively.

You can independently configure each DSP block during compilation as a triple 9 x 9, a dual 18 x 19 multiply, or a single 27 x 27. With a dedicated 64-bit cascade bus, you can cascade multiple variable-precision DSP blocks to implement even higher precision DSP functions efficiently.

The variable precision DSP block also supports these features:

- 64-bit accumulator that is the largest in the industry,
- Double accumulator
- Hard pre-adder that is available in both 18- and 27-bit modes
- Cascaded output adders for efficient systolic FIR filters
- Dynamic coefficients
- 18-bit internal coefficient register banks
- Enhanced independent multiplier operation
- Efficient support for single floating point arithmetic
- Inferability of all modes by the Altera Complete Design Suite

Table 1–8 lists the accommodation of different configurations in a DSP block.

Table 1–8. Variable-Precision DSP Block Configurations for Arria V Devices

Multiplier Size (Bit)	DSP Block Resources	Expected Usage
Three 9 x 9	1 variable-precision DSP block	Low precision fixed point for video applications
Two18 x 19	1 of variable-precision DSP block	Medium precision fixed point in FIR filters
Two 18 x 19 with accumulate	1 variable-precision DSP block	FIR filters
One 27 x 27	1 variable-precision DSP block	Single precision floating point

Table 1–9 lists the number of multipliers in Arria V devices.

Table 1–9. Number of Multipliers in Arria V Devices

Variants	Devices	Variable Precision DSP Blocks	Independent Input and Output Multiplications Operator			18 x 19 Multiplier Adder Mode	18 x 18 Multiplier Adder Summed with 36-bit Input
			9 x 9 Multipliers	18 x 19 Multipliers	27 x 27 Multipliers		
Arria V GX	5AGXA1	240	720	480	240	240	240
	5AGXA3	396	1,188	792	396	396	396
	5AGXA5	600	1,800	1,200	600	600	600
	5AGXA7	800	2,400	1,600	800	800	800
	5AGXB1	920	2,760	1,840	920	920	920
	5AGXB3	1,045	3,135	2,090	1,045	1,045	1,045
	5AGXB5	1,092	3,276	2,184	1,092	1,092	1,092
	5AGXB7	1,139	3,417	2,278	1,139	1,139	1,139
Arria V GT	5AGTD3	1,045	3,135	2,090	1,045	1,045	1,045
	5AGTD7	1,139	3,417	2,278	1,139	1,139	1,139
Arria V SX	5ASXB3	809	2,427	1,618	809	809	809
	5ASXB5	1,068	3,204	2,136	1068	1,068	1,068
Arria V ST	5ASTD3	809	2,427	1,618	809	809	809
	5ASTD5	1,068	3,204	2,136	1068	1,068	1,068

Embedded Memory

The Arria V memory blocks are flexible and designed to provide an optimal amount of small- and large-sized memory arrays. Arria V devices contain two types of embedded memory blocks:

- 640-bit MLAB blocks—for wide and shallow memories. You can use up to 25% of the device LABs as MLAB. The MLAB operates at up to 500 MHz.
- 10-Kb M10K blocks—for larger memory configurations. The M10K embedded memory operates at up to 400 MHz.

Table 1–10 lists the supported memory configurations for Arria V devices.

Table 1–10. Embedded Memory Block Configuration for Arria V Devices

Memory Block	Depth (bits)	Programmable Widths
MLAB	32	x16, x18, or x20
M10K	256	x40 or x32
	512	x20 or x16
	1K	x10 or x8
	2K	x5 or x4
	4K	x2
	8K	x1

Dynamic and Partial Reconfiguration

Dynamic reconfiguration enables transceiver data rates or encoding schemes to be changed dynamically while maintaining data transfer on adjacent transceiver channels in Arria V devices. Dynamic reconfiguration is ideal for applications requiring on-the-fly multi-protocol or multi-rate support. You can reconfigure the PMA, PCS, and PCIe hard IP blocks with dynamic reconfiguration.

Partial reconfiguration allows you to reconfigure part of the device while other sections remain running. Partial reconfiguration is required in systems where the uptime is critical because it allows you to make updates or adjust functionality without disrupting other services. While lowering power and cost, partial reconfiguration also increases the effective logic density by removing the necessity to place the device functions that do not operate simultaneously. Instead, you can store these functions in external memory and load them as required. This reduces the size of the required device by allowing multiple applications on a single device, which saves board space and reduces power consumption.

Altera simplifies the time-intensive task of partial reconfiguration by building the partial reconfiguration capability on top of the proven incremental compile and design flow in the Quartus® II software. With this Altera solution, you do not need to know all the intricate device architecture details to perform a partial reconfiguration.

Partial reconfiguration is supported through the FPP x16 configuration interface. You can seamlessly use partial reconfiguration in tandem with dynamic reconfiguration to enable partial reconfiguration of both the core and transceiver simultaneously.

Clock Networks and PLL Clock Sources

The Arria V clock network architecture is based on Altera's proven global, quadrant, and peripheral clock structure, which is supported by dedicated clock input pins and fractional PLLs. Arria V devices have 16 global clock networks capable of up to 625 MHz operation. The Quartus II software identifies all unused sections of the clock network and powers them down, which reduces power consumption.

Arria V devices have up to 16 PLLs with 18 output counters per PLL. One fractional PLL can use up to 18 output counters and two adjacent fractional PLLs share the 18 output counters. You can use fractional PLLs to reduce the number of oscillators required on your board, as well as reduce the clock pins used in the device by synthesizing multiple clock frequencies from a single reference clock source. You can use the PLLs for frequency synthesis, on-chip clock deskew, jitter attenuation, dynamic phase-shift, zero delay buffers, counters reconfiguration, bandwidth reconfiguration, programmable output clock duty cycles, PLL cascading, and reference clock switchover.

Arria V devices use a fractional PLL architecture in addition to the historical integer PLL. When you use fractional PLL mode, you can use the PLLs for precision fractional-N frequency synthesis—removing the need for an off-chip reference clock. Transceiver fractional PLLs, when not used by the Transceiver I/O, can be used as general-purpose fractional PLLs by the FPGA fabric.

Enhanced Configuration and Configuration via Protocol

Arria V devices support 3.3-V programming voltage and the following configuration modes:

- active serial (AS)
- passive serial (PS)
- fast passive parallel (FPP)
- CvP
- Configuration via HPS
- configuration through JTAG

You can configure Arria V devices through PCIe using CvP instead of an external flash or ROM. The CvP mode offers the fastest configuration rate and flexibility with the easy-to-use PCIe hard IP block interface. The Arria V CvP implementation conforms to the PCIe 100-ms power-up-to-active time requirement.

For more information regarding CvP, refer to the *Configuration via Protocol (CvP) Implementation in Altera FPGAs User Guide*.

Table 1–11 lists the configuration modes that Arria V devices support.

Table 1–11. Configuration Modes and Features for Arria V Devices

Mode	Data Width (Bit)	Maximum Clock Rate (MHz)	Maximum Data Rate (Mbps)	Decompression	Design Security	Remote System Update	Partial Reconfiguration
AS	1, 4	100	—	✓	✓	✓	—
PS	1	125	125	✓	✓	—	—
FPP	8, 16	125	—	✓	✓	Parallel flash loader	16-bit only
CvP	x1, x2, x4, x8 (1)	—	—	✓	✓	✓	✓
HPS	32	125	—	✓	✓	Parallel flash loader	✓
JTAG	1	33	33	—	—	—	—

Note to Table 1–11:

(1) Number of lanes instead of bits.

Power Management

Arria V devices leverage FPGA architectural features and process technology advancements to reduce the total device core power consumption by as much as 50% when compared with Stratix IV devices at the same performance level.

Additionally, Arria V devices have a number of hard IP blocks that not only reduce logic resources but also deliver substantial power savings when compared with soft implementations. The list includes PCIe Gen1 and Gen2, XAUI, GbE, SRIO, GPON and CPRI protocols. The hard IP blocks consume up to 25% less power than equivalent soft implementations.

Arria V transceivers are also designed for power efficiency. As a result, the transceiver channels consume 50% less power than the previous generation of Arria devices.

SoC FPGA with HPS

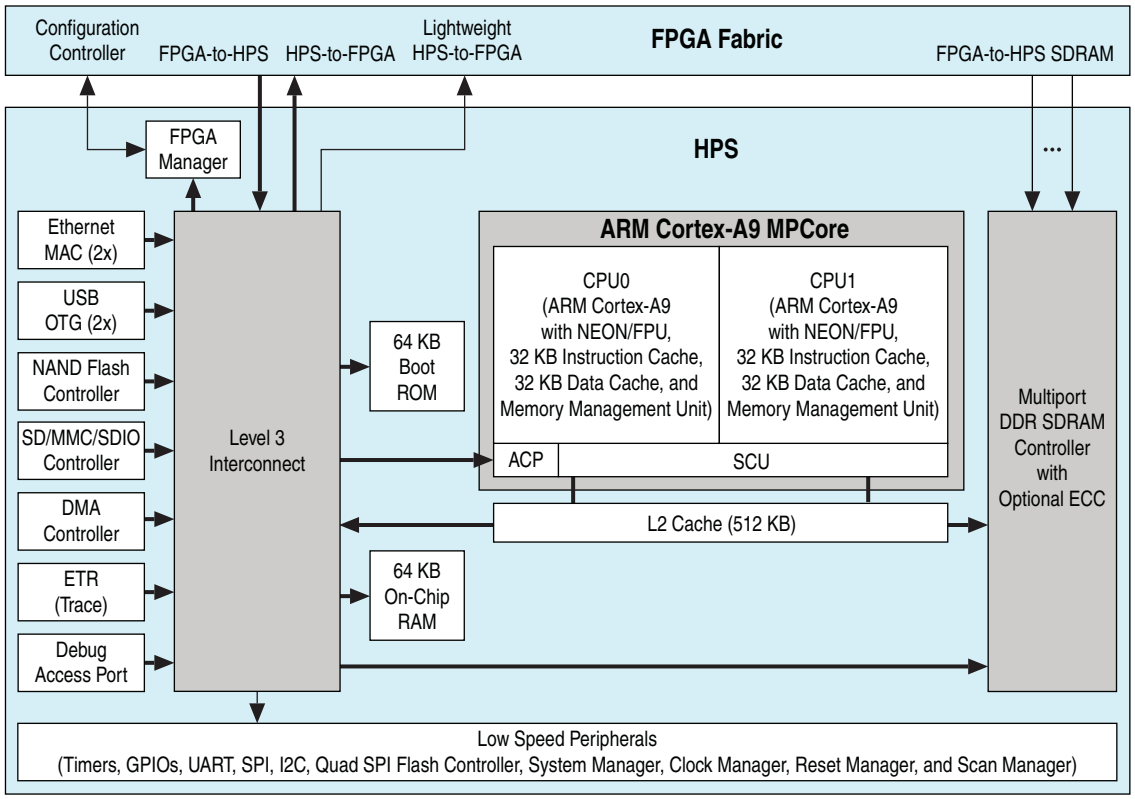
Each SoC FPGA device combines an FPGA fabric and an HPS in a single device. This combination delivers the flexibility of programmable logic with the power and cost savings of hard IP in the following ways:

- Reduces board space, system power, and bill of materials cost by eliminating a discrete embedded processor
- Allows you to differentiate the end product in both the hardware and software, and to support virtually any interface standard
- Extends the product life and revenue through in-field hardware and software updates

Features of the HPS

The HPS consists of a dual-core ARM Cortex-A9 MPCore processor, a rich set of peripherals, and a shared multiport SDRAM memory controller, as shown in Figure 1–4..

Figure 1–4. HPS with Dual-Core ARM Cortex-A9 MPCore Processor



System Peripherals

The Ethernet MAC, USB OTG controller, NAND flash controller and SD/MMC/SDIO controller modules have an integrated DMA controller. For modules without an integrated DMA controller, an additional DMA controller module provides up to eight channels for high-bandwidth data transfers. The debug access port provides interfaces to industry standard JTAG debug probes and supports ARM CoreSight debug and core traces to facilitate software development.

HPS-FPGA AXI Bridges

The HPS–FPGA bridges, which support the Advanced Microcontroller Bus Architecture (AMBA®) Advanced eXtensible Interface (AXI™) specifications, consist of the following bridges:

- FPGA-to-HPS AXI bridge—a high-performance bus supporting 32-, 64-, and 128-bit data widths that allows the FPGA fabric to master transactions to the slaves in the HPS
- HPS-to-FPGA AXI bridge—a high-performance bus supporting 32-, 64-, and 128-bit data widths that allows the HPS to master transactions to the slaves in the FPGA fabric.
- Lightweight HPS-to-FPGA AXI bridge—a lower performance 32-bit width bus that allows the HPS to master transactions to the slaves in the FPGA fabric.

The HPS–FPGA AXI bridges also allow the FPGA fabric to access the memory shared by one or both microprocessors, and provide asynchronous clock crossing with the clock from the FPGA fabric.

HPS SDRAM Controller Subsystem

The HPS SDRAM controller subsystem contains a multiport SDRAM controller and DDR PHY that is shared between the FPGA fabric (through the FPGA-to-HPS SDRAM interface), the level 2 (L2) cache, and the level 3 (L3) system interconnect. The FPGA-to-HPS SDRAM interface supports AMBA AXI and Avalon® Memory-Mapped (Avalon-MM) interface standards, and provides up to four ports with separate read and write directions.

To maximize memory performance, the HPS SDRAM controller subsystem supports command and data reordering, deficit round-robin arbitration with aging, and high-priority bypass features. The HPS SDRAM controller subsystem supports DDR2, DDR3, LPDDR, or LPDDR2 devices up to 4 Gb and runs up to 533 MHz (1066 Mbps data rate).

For easy migration, the FPGA-to-HPS SDRAM interface is compatible with the interface of the soft SDRAM memory controller IPs and hard SDRAM memory controllers in the FPGA fabric.

FPGA Configuration and Processor Booting

The FPGA fabric and HPS in the SoC FPGA are powered independently. You can reduce the clock frequencies or gate the clocks to reduce dynamic power, or shut down the entire FPGA fabric to reduce total system power.

You can configure the FPGA fabric and boot the HPS independently, in any order, providing you with more design flexibility:

- You can boot the HPS before you power up and configure the FPGA fabric. After the system is running, the HPS reconfigures the FPGA fabric at any time under program control or through the FPGA configuration controller.
- You can power up both the HPS and the FPGA fabric together, configure the FPGA fabric first, and then upload the boot code to the HPS from the FPGA fabric.

Hardware and Software Development

For hardware development, you can configure the HPS and connect your soft logic in the FPGA fabric to the HPS interfaces using the Qsys system integration tool in the Quartus II software.

For software development, the ARM-based SoC FPGA devices inherit the rich software development ecosystem available for the ARM Cortex-A9 MPCore processor. The software development process for Altera SoC FPGAs follows the same steps as those for other SoC devices. Altera also provides support for the Linux and VxWorks® operating systems.

You can begin device-specific firmware and software development on the Altera SoC FPGA Virtual Target. The Virtual Target is a PC-based fast-functional simulation of a target development system—a model of a complete development board that runs on a PC. The Virtual Target enables the development of device-specific production software that can run unmodified on actual hardware.

Ordering Information

This section describes the ordering information for Arria V devices.

Figure 1–5 and Figure 1–6 show the ordering codes for Arria V devices.

Figure 1–5. Ordering Information for Arria V GX and GT Devices

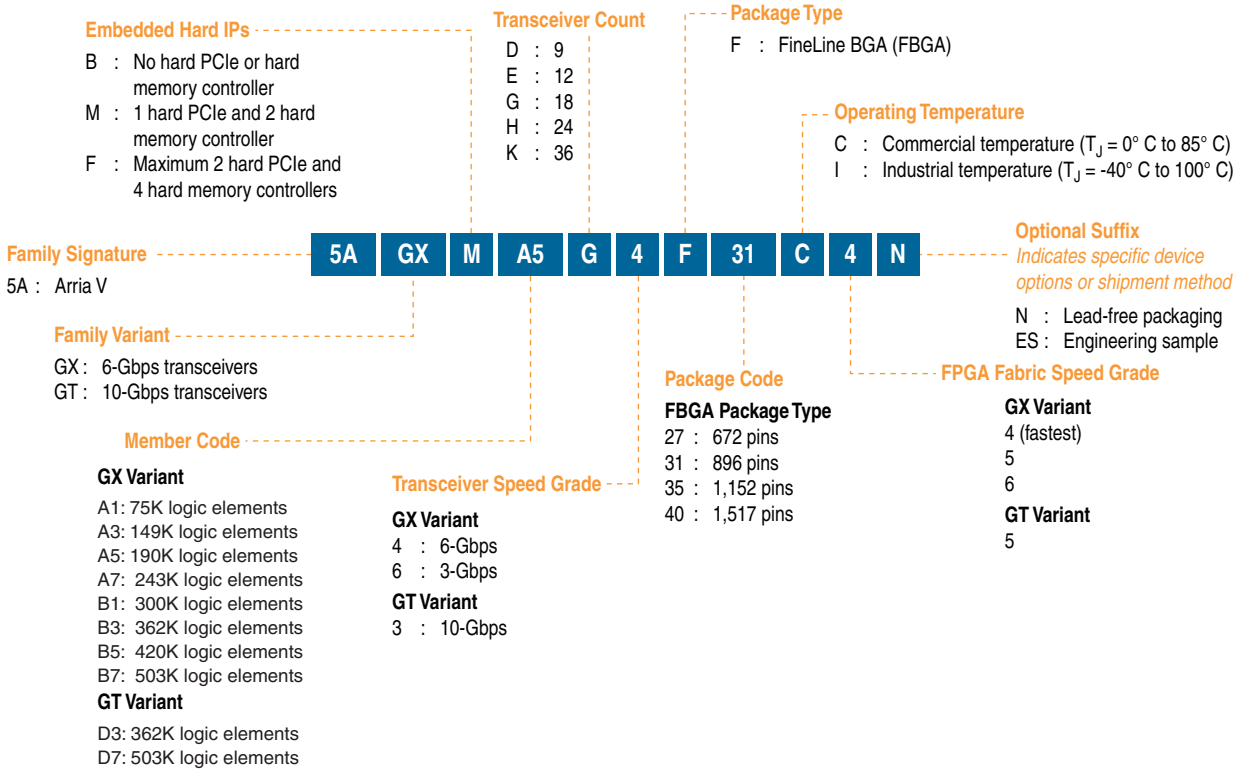
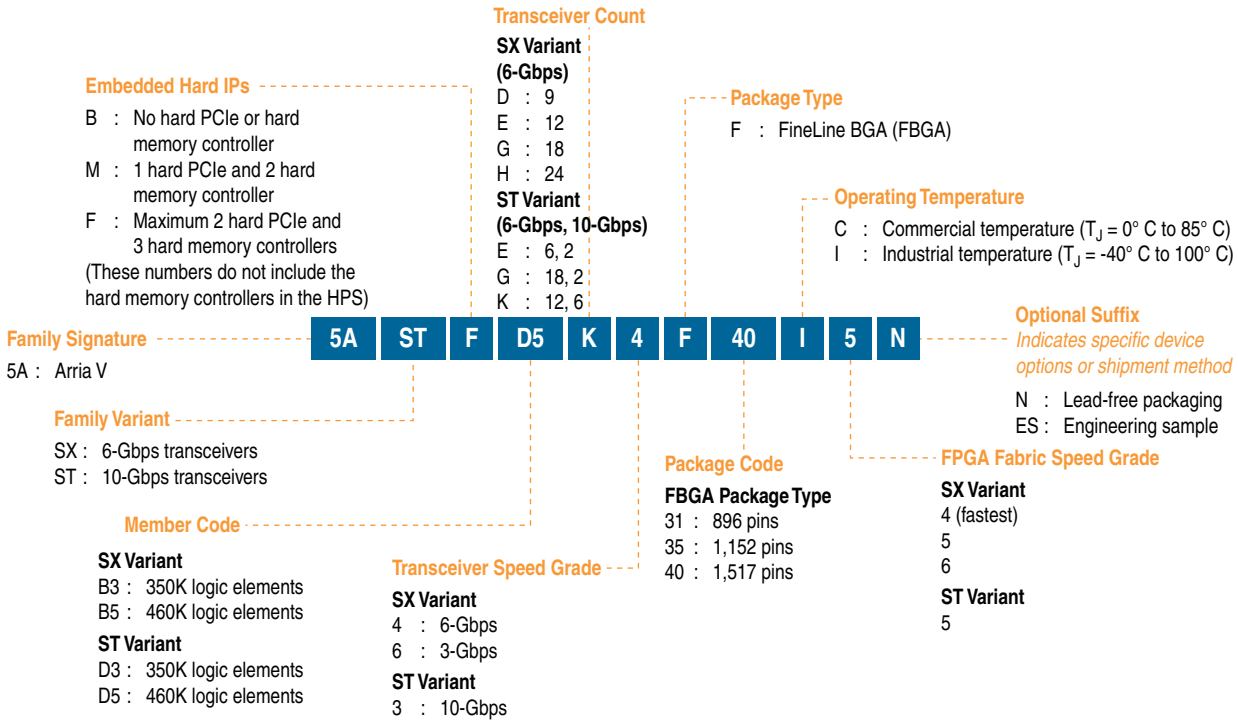


Figure 1–6. Ordering Information for Arria V SX and ST Devices



Document Revision History

Table 1–12 lists the revision history for this chapter.

Table 1–12. Document Revision History

Date	Version	Changes
February 2012	1.3	■ Updated Table 1–2 and Table 1–3. ■ Updated Figure 1–5 and Figure 1–6. ■ Minor text edits.
December 2011	1.2	■ Minor text edits.
November 2011	1.1	■ Updated Table 1–1, Table 1–2, Table 1–3, Table 1–4, Table 1–6, Table 1–7, Table 1–9, and Table 1–10. ■ Added “SoC FPGA with HPS” section. ■ Updated “Clock Networks and PLL Clock Sources” and “Ordering Information” sections. ■ Updated Figure 1–5. ■ Added Figure 1–6. ■ Minor text edits.
August 2011	1.0	Initial release.

2. Device Datasheet for Arria V Devices

AV-51002-1.3

This chapter describes the electrical characteristics, switching characteristics, and configuration specifications for Arria® V devices. Electrical characteristics include operating conditions and power consumption. Switching characteristics include transceiver specifications, and core and periphery performance. Configuration specifications include power-on reset (POR) specification, initialization clock source option and timing, various configuration mode timing parameters, remote system upgrades timing, and user watchdog internal oscillator frequency specification. This chapter also describes I/O timing, including programmable I/O element (IOE) delay and programmable output buffer delay.

For more information about the densities and packages of devices in the Arria V family, refer to the *Overview for Arria V Device Family* chapter.

Electrical Characteristics

The following sections describe the electrical characteristics of Arria V devices.

Operating Conditions

When you use Arria V devices, they are rated according to a set of defined parameters. To maintain the highest possible performance and reliability of the Arria V devices, you must consider the operating requirements described in this chapter.

Arria V devices are offered in commercial and industrial grades. Commercial devices are offered in –4 (fastest), –5, and –6 speed grades. Industrial grade devices are offered in the –5 speed grade.

Absolute Maximum Ratings

Absolute maximum ratings define the maximum operating conditions for Arria V devices. The values are based on experiments conducted with the devices and theoretical modeling of breakdown and damage mechanisms. The functional operation of the device is not implied for these conditions.

CAUTION

Conditions other than those listed in Table 2–1 may cause permanent damage to the device. Additionally, device operation at the absolute maximum ratings for extended periods of time may have adverse effects on the device.

Table 2–1. Absolute Maximum Ratings for Arria V Devices—*Preliminary*

Symbol	Description	Minimum	Maximum	Unit
V_{CC}	Core voltage power supply	–0.5	1.35	V
V_{CCP}	Periphery circuitry, PCIe® hard IP block, and transceiver physical coding sublayer (PCS) power supply	–0.5	1.35	V
V_{CCPGM}	Configuration pins power supply	–0.5	3.75	V
V_{CCAUX}	Auxiliary supply	–0.5	3.75	V
V_{CCBAT}	Battery back-up power supply for design security volatile key register	–0.5	3.75	V
V_{CCPD}	I/O pre-driver power supply	–0.5	3.75	V
V_{CCIO}	I/O power supply	–0.5	3.9	V
V_{CCD_FPLL}	Phase-locked loop (PLL) digital power supply	–0.5	1.8	V
V_{CCA_FPLL}	PLL analog power supply	–0.5	3.75	V
V_{CCA_GXB}	Transceiver high voltage power	–0.5	3.75	V
V_{CCH_GXB}	Transmitter output buffer power	–0.5	1.8	V
V_{CCR_GXB}	Receiver power	–0.5	1.21	V
V_{CCT_GXB}	Transmitter power	–0.5	1.21	V
V_{CCL_GXB}	Clock network power	–0.5	1.21	V
V_I	DC input voltage	–0.5	4	V
I_{OUT}	DC output current per pin	–25	40	mA
T_J	Operating junction temperature	–55	125	°C
T_{STG}	Storage temperature (No bias)	–65	150	°C
V_{CC_HPS}	Core voltage power supply	–0.5	1.35	V
V_{CCPD_HPS}	I/O pre-driver power supply	–0.5	3.75	V
V_{CCIO_HPS}	I/O power supply	–0.5	3.9	V
$V_{CCRSTCLK\_HPS}$	Configuration pins power supply	–0.5	3.75	V
V_{CCPLL_HPS}	PLL analog power supply	–0.5	3.75	V

Maximum Allowed Overshoot and Undershoot Voltage

During transitions, input signals may overshoot to the voltage listed in Table 2–2 and undershoot to –2.0 V for input currents less than 100 mA and periods shorter than 20 ns.

Table 2–2 lists the maximum allowed input overshoot voltage and the duration of the overshoot voltage as a percentage of device lifetime. The maximum allowed overshoot duration is specified as a percentage of high time over the lifetime of the device. A DC signal is equivalent to 100% duty cycle. For example, a signal that overshoots to 3.95 V can only be at 3.95 V for ~5% over the lifetime of the device; for a device lifetime of 10 years, this amounts to half a year.

Table 2–2. Maximum Allowed Overshoot During Transitions for Arria V Devices—*Preliminary*

Symbol	Description	Condition (V)	Overshoot Duration as % of High Time	Unit
Vi (AC)	AC input voltage	3.7	100	%
		3.75	59.79	%
		3.8	33.08	%
		3.85	18.45	%
		3.9	10.36	%
		3.95	5.87	%
		4	3.34	%
		4.05	1.92	%
		4.1	1.11	%

Recommended Operating Conditions

This section lists the functional operation limits for the AC and DC parameters for Arria V devices. Table 2–3 lists the steady-state voltage values expected from Arria V devices. Power supply ramps must all be strictly monotonic, without plateaus.

Table 2–3. Recommended Operating Conditions for Arria V Devices—*Preliminary*

Symbol	Description	Condition	Minimum	Typical	Maximum	Unit
V_{CC}	Core voltage power supply	—	1.07	1.1	1.13	V
V_{CCP}	Periphery circuitry, PCIe hard IP block, and transceiver PCS power supply	—	1.07	1.1	1.13	V
V_{CCAUX}	Auxiliary supply	—	2.375	2.5	2.625	V
V_{CCPD} (1)	I/O pre-driver (3.3 V) power supply	—	3.135	3.3	3.465	V
	I/O pre-driver (3.0 V) power supply	—	2.85	3.0	3.15	V
	I/O pre-driver (2.5 V) power supply	—	2.375	2.5	2.625	V
V_{CCIO}	I/O buffers (3.3 V) power supply	—	3.135	3.3	3.465	V
	I/O buffers (3.0 V) power supply	—	2.85	3.0	3.15	V
	I/O buffers (2.5 V) power supply	—	2.375	2.5	2.625	V
	I/O buffers (1.8 V) power supply	—	1.71	1.8	1.89	V
	I/O buffers (1.5 V) power supply	—	1.425	1.5	1.575	V
	I/O buffers (1.35 V) power supply	—	1.283	1.35	1.418	V
	I/O buffers (1.25 V) power supply	—	1.19	1.25	1.31	V
	I/O buffers (1.2 V) power supply	—	1.14	1.2	1.26	V
V_{CCPGM}	Configuration pins (3.3 V) power supply	—	3.135	3.3	3.465	V
	Configuration pins (3.0 V) power supply	—	2.85	3.0	3.15	V
	Configuration pins (2.5 V) power supply	—	2.375	2.5	2.625	V
	Configuration pins (1.8 V) power supply	—	1.71	1.8	1.89	V
V_{CCA_FPLL}	PLL analog voltage regulator power supply	—	2.375	2.5	2.625	V
V_{CCD_FPLL}	PLL digital voltage regulator power supply	—	1.425	1.5	1.575	V
V_{CCBAT} (2)	Battery back-up power supply (For design security volatile key register)	—	1.2	—	3.0	V
V_I	DC input voltage	—	–0.5	—	3.6	V
V_O	Output voltage	—	0	—	V_{CCIO}	V
T_J	Operating junction temperature	Commercial	0	—	85	°C
		Industrial	–40	—	100	°C
t_{RAMP} (3)	Power supply ramp time	Slow POR (PORSEL=0)	200 µs	—	100 ms	—
		Fast POR (PORSEL=1)	200 µs	—	4 ms	—

Notes to Table 2–3:

(1) V_{CCPD} must be 2.5 V when V_{CCIO} is 2.5, 1.8, 1.5, 1.35, 1.25 or 1.2 V. V_{CCPD} must be 3.0 V when V_{CCIO} is 3.0 V.

(2) If you do not use the design security feature in Arria V devices, connect V_{CCBAT} to a 1.5-V, 2.5-V or 3.0-V power supply. Arria V POR circuitry monitors V_{CCBAT}. Arria V devices do not exit POR if V_{CCBAT} stays low.

(3) When power is applied to an Arria V device, a POR occurs if the power supply reaches the recommended operating range in the maximum power supply ramp.

Table 2–4 lists the transceiver power supply recommended operating conditions for Arria V devices.

Table 2–4. Transceiver Power Supply Operating Conditions for Arria V GX and GT Devices—*Preliminary*

Symbol	Description	Minimum	Typical	Maximum	Unit
V_{CCA_GXBL}	Transceiver high voltage power (left side)	2.375	2.5	2.625	V
V_{CCA_GXBR}	Transceiver high voltage power (right side)				
V_{CCR_GXBL}	Receiver power (left side)	1.07	1.1	1.13	V
V_{CCR_GXBR}	Receiver power (right side)				
V_{CCT_GXBL}	Transmitter power (left side)	1.07	1.1	1.13	V
V_{CCT_GXBR}	Transmitter power (right side)				
V_{CCH_GXBL}	Transmitter output buffer power (left side)	1.425	1.5	1.575	V
V_{CCH_GXBR}	Transmitter output buffer power (right side)				
V_{CCL_GXBL}	Clock network power (left side)	1.07	1.1	1.13	V
V_{CCL_GXBR}	Clock network power (right side)	1.07	1.1	1.13	V

Table 2–5 lists the steady-state voltage and current values expected from Arria V system-on-a-chip (SoC) FPGA with ARM®-based hard processor system (HPS). Power supply ramps must all be strictly monotonic, without plateaus.

Table 2–5. HPS Power Supply Operating Conditions for Arria V SX and ST Devices—*Preliminary*

Symbol	Description	Minimum	Typical	Maximum	Unit
V_{CC_HPS}	HPS Core voltage and periphery circuitry power supply	1.07	1.1	1.13	V
V_{CCPD_HPS}	HPS I/O pre-driver (3.3 V) power supply	3.135	3.3	3.465	V
	HPS I/O pre-driver (3.0 V) power supply	2.85	3.0	3.15	V
	HPS I/O pre-driver (2.5 V) power supply	2.375	2.5	2.625	V
V_{CCIO_HPS}	HPS I/O buffers (3.3 V) power supply	3.135	3.3	3.465	V
	HPS I/O buffers (3.0 V) power supply	2.85	3.0	3.15	V
	HPS I/O buffers (2.5 V) power supply	2.375	2.5	2.625	V
	HPS I/O buffers (1.8 V) power supply	1.71	1.8	1.89	V
	HPS I/O buffers (1.5 V) power supply	1.425	1.5	1.575	V
	HPS I/O buffers (1.2 V) power supply	1.14	1.2	1.26	V
$V_{CCRSTCLK\_HPS}$	HPS reset and clock input pins (3.3 V) power supply	3.135	3.3	3.465	V
	HPS reset and clock input pins (3.0 V) power supply	2.85	3.0	3.15	V
	HPS reset and clock input pins (2.5 V) power supply	2.375	2.5	2.625	V
	HPS reset and clock input pins (1.8 V) power supply	1.71	1.8	1.89	V
V_{CCPLL_HPS}	HPS PLL analog voltage regulator power supply	2.375	2.5	2.625	V

DC Characteristics

This section lists the supply current, I/O pin leakage current, input pin capacitance, on-chip termination tolerance, and hot socketing specifications.

Supply Current

Standby current is the current drawn from the respective power rails used for power budgeting. Use the Excel-based Early Power Estimator (EPE) to estimate supply current for your design because these currents vary greatly with the resources you use.

For more information about power estimation tools, refer to the *PowerPlay Early Power Estimator User Guide* and the *PowerPlay Power Analysis* chapter in the *Quartus II Handbook*.

I/O Pin Leakage Current

Table 2–6 lists the Arria V I/O pin leakage current specifications.

Table 2–6. I/O Pin Leakage Current for Arria V Devices—*Preliminary*

Symbol	Description	Conditions	Min	Typ	Max	Unit
I_I	Input pin	V_I = 0 V to $V_{CCIOMAX}$	–30	—	30	µA
I_{OZ}	Tri-stated I/O pin	V_O = 0 V to $V_{CCIOMAX}$	–30	—	30	µA

Bus Hold Specifications

Table 2–7 lists the Arria V device bus hold specifications.

Table 2–7. Bus Hold Parameters for Arria V Devices—*Preliminary*[1] (Part 1 of 2)

Parameter	Symbol	Conditions	V_{CCIO} (V)												Unit
			1.2		1.5		1.8		2.5		3.0		3.3		
			Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
Bus-hold, low, sustaining current	I_{SUSL}	$V_{IN} > V_{IL}$ (max.)	8	—	12	—	30	—	50	—	70	—	70	—	µA
Bus-hold, high, sustaining current	I_{SUSH}	$V_{IN} < V_{IH}$ (min.)	–8	—	–12	—	–30	—	–50	—	–70	—	-70	—	µA
Bus-hold, low, overdrive current	I_{ODL}	0V < V_{IN} < V_{CCIO}	—	125	—	175	—	200	—	300	—	500	—	500	µA

Table 2–7. Bus Hold Parameters for Arria V Devices—*Preliminary*[1] (Part 2 of 2)

Parameter	Symbol	Conditions	V_{CCIO} (V)												Unit
			1.2		1.5		1.8		2.5		3.0		3.3		
			Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
Bus-hold, high, overdrive current	I_{ODH}	$0V < V_{IN} < V_{CCIO}$	—	-125	—	-175	—	-200	—	-300	—	-500	—	-500	µA
Bus-hold trip point	V_{TRIP}	—	0.3	0.9	0.375	1.125	0.68	1.07	0.7	1.7	0.8	2	0.8	2	V

Note to Table 2–7:

(1) The bus-hold trip points are based on calculated input voltages from the JEDEC standard.

(OCT) Specifications

If you enable on-chip termination (OCT) calibration, calibration is automatically performed at power up for I/Os connected to the calibration block. Table 2–8 lists the Arria V OCT termination calibration accuracy specifications.

Table 2–8. OCT Calibration Accuracy Specifications for Arria V Devices—*Preliminary*[1] (Part 1 of 2)

Symbol	Description	Conditions (V)	Calibration Accuracy			Unit
			C4	C5, I5	C6	
25-Ω R_S	Internal series termination with calibration (25-Ω setting)	V_{CCIO} = 3.0, 2.5, 1.8, 1.5, 1.2	±15	±15	±15	%
50-Ω R_S	Internal series termination with calibration (50-Ω setting)	V_{CCIO} = 3.0, 2.5, 1.8, 1.5, 1.2	±15	±15	±15	%
34-Ω and 40-Ω R_S	Internal series termination with calibration (34-Ω and 40-Ω setting)	V_{CCIO} = 1.5, 1.35, 1.25, 1.2	±15	±15	±15	%
48-Ω, 60-Ω, and 80-Ω R_S	Internal series termination with calibration (48-Ω, 60-Ω, and 80-Ω setting)	V_{CCIO} = 1.2	±15	±15	±15	%
50-Ω R_T	Internal parallel termination with calibration (50-Ω setting)	V_{CCIO} = 2.5, 1.8, 1.5, 1.2	–10 to +40	–10 to +40	–10 to +40	%
20-Ω, 30-Ω, 40-Ω,60-Ω, and 120-Ω R_T	Internal parallel termination with calibration (20-Ω, 30-Ω, 40-Ω, 60-Ω, and 120-Ω setting)	V_{CCIO} = 1.5, 1.35, 1.25	–10 to +40	–10 to +40	–10 to +40	%

Table 2–8. OCT Calibration Accuracy Specifications for Arria V Devices—*Preliminary*(1) (Part 2 of 2)

Symbol	Description	Conditions (V)	Calibration Accuracy			Unit
			C4	C5, I5	C6	
60-Ω and 120-Ω R_T	Internal parallel termination with calibration (60-Ω and 120-Ω setting)	V_{CCIO} = 1.2	–10 to +40	–10 to +40	–10 to +40	%
25-Ω $R_{S\_left\_shift}$	Internal left shift series termination with calibration (25-Ω $R_{S\_left\_shift}$ setting)	V_{CCIO} = 3.0, 2.5, 1.8, 1.5, 1.2	±15	±15	±15	%

Note to Table 2–8:

(1) OCT calibration accuracy is valid at the time of calibration only.

Calibration accuracy for the calibrated on-chip series termination (R_S OCT) and on-chip parallel termination (R_T OCT) are applicable at the moment of calibration. When process, voltage, and temperature (PVT) conditions change after calibration, the tolerance may change.

Table 2–9 lists the Arria V OCT without calibration resistance tolerance to PVT changes.

Table 2–9. OCT Without Calibration Resistance Tolerance Specifications for Arria V Devices—*Preliminary*

Symbol	Description	Conditions (V)	Resistance Tolerance			Unit
			C4	C5, I5	C6	
25-Ω R_S	Internal series termination without calibration (25-Ω setting)	V_{CCIO} = 3.0 and 2.5	±30	±40	±40	%
25-Ω R_S	Internal series termination without calibration (25-Ω setting)	V_{CCIO} = 1.8 and 1.5	±30	±40	±40	%
25-Ω R_S	Internal series termination without calibration (25-Ω setting)	V_{CCIO} = 1.2	±35	±50	±50	%
50-Ω R_S	Internal series termination without calibration (50-Ω setting)	V_{CCIO} = 3.0 and 2.5	±30	±40	±40	%
50-Ω R_S	Internal series termination without calibration (50-Ω setting)	V_{CCIO} = 1.8 and 1.5	±30	±40	±40	%
50-Ω R_S	Internal series termination without calibration (50-Ω setting)	V_{CCIO} = 1.2	±35	±50	±50	%
100-Ω R_D	Internal differential termination (100-Ω setting)	V_{CCIO} = 2.5	±25	TBD	TBD	%

OCT calibration is automatically performed at power up for OCT-enabled I/Os. Table 2–10 lists OCT variation with temperature and voltage after power-up calibration. Use Table 2–10 to determine the OCT variation after power-up calibration and Equation 2–1 to determine the OCT variation without recalibration.

Equation 2–1. OCT Variation Without Recalibration—*Preliminary* (1), (2), (3), (4), (5), (6)

$$R_{OCT} = R_{SCAL}\left(1 + \langle\frac{dR}{dT} \times \Delta T\rangle \pm \langle\frac{dR}{dV} \times \Delta V\rangle\right)$$

Notes to Equation 2–1:

(1) The R_{OCT} value calculated from Equation 2–1 shows the range of OCT resistance with the variation of temperature and V_{CCIO}.
(2) R_{SCAL} is the OCT resistance value at power-up.
(3) ΔT is the variation of temperature with respect to the temperature at power up.
(4) ΔV is the variation of voltage with respect to the V_{CCIO} at power up.
(5) dR/dT is the percentage change of R_{SCAL} with temperature.
(6) dR/dV is the percentage change of R_{SCAL} with voltage.

Table 2–10 lists the OCT variation after the power-up calibration.

Table 2–10. OCT Variation after Power-Up Calibration for Arria V Devices—*Preliminary* (1)

Symbol	Description	V_{CCIO} (V)	Value	Unit
dR/dV	OCT variation of voltage without recalibration	3.0	0.0297	%/mV
		2.5	0.0344	
		1.8	0.0499	
		1.5	0.0744	
		1.2	0.1241	
dR/dT	OCT variation of temperature without recalibration	3.0	0.189	%/°C
		2.5	0.208	
		1.8	0.266	
		1.5	0.273	
		1.2	0.317	

Note to Table 2–10:

(1) Valid for a V_{CCIO} range of ±5% and temperature range of 0° to 85°C.

Pin Capacitance

Table 2–11 lists the Arria V pin capacitance.

Table 2–11. Pin Capacitance for Arria V Devices

Symbol	Description	Value	Unit
C_{IOTB}	Input capacitance on top/bottom I/O pins	5.5	pF
C_{IOLR}	Input capacitance on left/right I/O pins	5.5	pF
C_{OUTFB}	Input capacitance on dual-purpose clock output/feedback pins	5.5	pF

Hot Socketing

Table 2–12 lists the hot socketing specifications for Arria V devices.

Table 2–12. Hot Socketing Specifications for Arria V Devices—*Preliminary*

Symbol	Description	Maximum
$I_{IOPIN\ (DC)}$	DC current per I/O pin	300 µA
$I_{IOPIN\ (AC)}$	AC current per I/O pin	8 mA (1)
$I_{XCVR\text{-}TX\ (DC)}$	DC current per transceiver transmitter (TX) pin	100 mA
$I_{XCVR\text{-}RX\ (DC)}$	DC current per transceiver receiver (RX) pin	50 mA

Note to Table 2–12:

(1) The I/O ramp rate is 10 ns or more. For ramp rates faster than 10 ns, $|I_{IOPIN}| = C\ dv/dt$, in which C is the I/O pin capacitance and dv/dt is the slew rate.

Internal Weak Pull-Up Resistor

Table 2–13 lists the weak pull-up resistor values for Arria V devices.

Table 2–13. Internal Weak Pull-Up Resistor Values for Arria V Devices—*Preliminary* (1), (2)

Symbol	Description	Conditions (V) (3)	Value (4)	Unit
R_{PU}	Value of the I/O pin pull-up resistor before and during configuration, as well as user mode if you have enabled the programmable pull-up resistor option.	V_{CCIO} = 3.3 ±5%	25	kΩ
		V_{CCIO} = 3.0 ±5%	25	kΩ
		V_{CCIO} = 2.5 ±5%	25	kΩ
		V_{CCIO} = 1.8 ±5%	25	kΩ
		V_{CCIO} = 1.5 ±5%	25	kΩ
		V_{CCIO} = 1.35 ±5%	25	kΩ
		V_{CCIO} = 1.25 ±5%	25	kΩ
		V_{CCIO} = 1.2 ±5%	25	kΩ

Notes to Table 2–13:

(1) All I/O pins have an option to enable weak pull-up except the configuration, test, and JTAG pins.

(2) The internal weak pull-down feature is only available for the JTAG TCK pin. The typical value for this internal weak pull-down resistor is approximately 25 kΩ.

(3) Pin pull-up resistance values may be lower if an external source drives the pin higher than V_{CCIO}.

(4) Valid with ±10% tolerances to cover changes over PVT.

I/O Standard Specifications

Table 2–14 through Table 2–19 list the input voltage (V_{IH} and V_{IL}), output voltage (V_{OH} and V_{OL}), and current drive characteristics (I_{OH} and I_{OL}) for various I/O standards supported by Arria V devices. The I/O standards tables also list the Arria V device family I/O standard specifications. The V_{OL} and V_{OH} values are valid at the corresponding I_{OH} and I_{OL}, respectively.

For an explanation of terms used in Table 2–14 through Table 2–19, refer to "Glossary" on page 2–48.

Table 2–14. Single-Ended I/O Standards for Arria V Devices—*Preliminary*

I/O Standard	V_{CCIO} (V)			V_{IL} (V)		V_{IH} (V)		V_{OL} (V)	V_{OH} (V)	I_{OL} (mA)	I_{OH} (mA)
	Min	Typ	Max	Min	Max	Min	Max	Max	Min		
3.3-V LVTTL	3.135	3.3	3.465	–0.3	0.8	1.7	3.6	0.45	2.4	4	–4
3.3-V LVCMOS	3.135	3.3	3.465	–0.3	0.8	1.7	3.6	0.2	V_{CCIO} – 0.2	2	–2
3.0-V LVTTL	2.85	3	3.15	–0.3	0.8	1.7	3.6	0.4	2.4	2	–2
3.0-V LVCMOS	2.85	3	3.15	–0.3	0.8	1.7	3.6	0.2	V_{CCIO} – 0.2	0.1	–0.1
3.0-V PCI	2.85	3	3.15	—	0.3 x V_{CCIO}	0.5 x V_{CCIO}	V_{CCIO} + 0.3	0.1 x V_{CCIO}	0.9 x V_{CCIO}	1.5	–0.5
3.0-V PCI-X	2.85	3	3.15	—	0.35 x V_{CCIO}	0.5 x V_{CCIO}	V_{CCIO} + 0.3	0.1 x V_{CCIO}	0.9 x V_{CCIO}	1.5	–0.5
2.5 V	2.375	2.5	2.625	–0.3	0.7	1.7	3.6	0.4	2	1	–1
1.8 V	1.71	1.8	1.89	–0.3	0.35 x V_{CCIO}	0.65 x V_{CCIO}	V_{CCIO} + 0.3	0.45	V_{CCIO} – 0.45	2	–2
1.5 V	1.425	1.5	1.575	–0.3	0.35 x V_{CCIO}	0.65 x V_{CCIO}	V_{CCIO} + 0.3	0.25 x V_{CCIO}	0.75 x V_{CCIO}	2	–2
1.2 V	1.14	1.2	1.26	–0.3	0.35 x V_{CCIO}	0.65 x V_{CCIO}	V_{CCIO} + 0.3	0.25 x V_{CCIO}	0.75 x V_{CCIO}	2	–2

Table 2–15. Single-Ended SSTL and HSTL I/O Reference Voltage Specifications for Arria V Devices—*Preliminary* (Part 1 of 2)

I/O Standard	V_{CCIO}(V)			V_{REF}(V)			V_{TT}(V)		
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
SSTL-2 Class I, II	2.375	2.5	2.625	0.49 x V_{CCIO}	0.5 x V_{CCIO}	0.51 x V_{CCIO}	V_{REF} – 0.04	V_{REF}	V_{REF} + 0.04
SSTL-18 Class I, II	1.71	1.8	1.89	0.833	0.9	0.969	V_{REF} – 0.04	V_{REF}	V_{REF} + 0.04
SSTL-15 Class I, II	1.425	1.5	1.575	0.49 x V_{CCIO}	0.5 x V_{CCIO}	0.51 x V_{CCIO}	0.49 x V_{CCIO}	0.5 x V_{CCIO}	0.51 x V_{CCIO}
SSTL 135 Class I, II	1.283	1.35	1.418	0.49 x V_{CCIO}	0.5 x V_{CCIO}	0.51 x V_{CCIO}	0.49 x V_{CCIO}	0.5 x V_{CCIO}	0.51 x V_{CCIO}
SSTL 125 Class I, II	1.19	1.25	1.26	0.49 x V_{CCIO}	0.5 x V_{CCIO}	0.51 x V_{CCIO}	0.49 x V_{CCIO}	0.5 x V_{CCIO}	0.51 x V_{CCIO}

Table 2–15. Single-Ended SSTL and HSTL I/O Reference Voltage Specifications for Arria V Devices—*Preliminary* (Part 2 of 2)

I/O Standard	V_{CCIO}(V)			V_{REF}(V)			V_{TT}(V)		
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
HSTL-18 Class I, II	1.71	1.8	1.89	0.85	0.9	0.95	—	V_{CCIO}/2	—
HSTL-15 Class I, II	1.425	1.5	1.575	0.68	0.75	0.9	—	V_{CCIO}/2	—
HSTL-12 Class I, II	1.14	1.2	1.26	0.47 x V_{CCIO}	0.5 x V_{CCIO}	0.53 x V_{CCIO}	—	V_{CCIO}/2	—
HSUL-12	1.14	1.2	1.3	0.49 x V_{CCIO}	0.5 x V_{CCIO}	0.51 x V_{CCIO}	—	—	—

Table 2–16. Single-Ended SSTL and HSTL I/O Standards Signal Specifications for Arria V Devices—*Preliminary* (Part 1 of 2)

I/O Standard	$V_{IL(DC)}$ (V)		$V_{IH(DC)}$ (V)		$V_{IL(AC)}$ (V)	$V_{IH(AC)}$ (V)	V_{OL} (V)	V_{OH} (V)	I_{OI} (mA)	I_{oh} (mA)
	Min	Max	Min	Max	Max	Min	Max	Min		
SSTL-2 Class I	–0.3	V_{REF} –0.15	V_{REF} + 0.15	V_{CCIO} + 0.3	V_{REF} – 0.31	V_{REF} + 0.31	V_{TT} – 0.608	V_{TT} + 0.608	8.1	–8.1
SSTL-2 Class II	–0.3	V_{REF} –0.15	V_{REF} + 0.15	V_{CCIO} + 0.3	V_{REF} – 0.31	V_{REF} + 0.31	V_{TT} – 0.81	V_{TT} + 0.81	16.2	–16.2
SSTL-18 Class I	–0.3	V_{REF} –0.125	V_{REF} + 0.125	V_{CCIO} + 0.3	V_{REF} – 0.25	V_{REF} + 0.25	V_{TT} – 0.603	V_{TT} + 0.603	6.7	–6.7
SSTL-18 Class II	–0.3	V_{REF} –0.125	V_{REF} + 0.125	V_{CCIO} + 0.3	V_{REF} – 0.25	V_{REF} + 0.25	0.28	V_{CCIO} –0.28	13.4	–13.4
SSTL-15 Class I	—	V_{REF} – 0.1	V_{REF} + 0.1	—	V_{REF} – 0.175	V_{REF} + 0.175	0.2 x V_{CCIO}	0.8 x V_{CCIO}	8	–8
SSTL-15 Class II	—	V_{REF} – 0.1	V_{REF} + 0.1	—	V_{REF} – 0.175	V_{REF} + 0.175	0.2 x V_{CCIO}	0.8 x V_{CCIO}	16	–16
SSTL 135	—	V_{REF} – 0.09	V_{REF} + 0.09	—	V_{REF} – 0.16	V_{REF} + 0.16	TBD (1)	TBD (1)	TBD (1)	TBD (1)
SSTL 125	—	V_{REF} – 0.85	V_{REF} + 0.85	—	V_{REF} – 0.15	V_{REF} + 0.15	TBD (1)	TBD (1)	TBD (1)	TBD (1)
HSTL-18 Class I	—	V_{REF} –0.1	V_{REF} + 0.1	—	V_{REF} – 0.2	V_{REF} + 0.2	0.4	V_{CCIO} – 0.4	8	–8
HSTL-18 Class II	—	V_{REF} – 0.1	V_{REF} + 0.1	—	V_{REF} – 0.2	V_{REF} + 0.2	0.4	V_{CCIO} – 0.4	16	–16
HSTL-15 Class I	—	V_{REF} – 0.1	V_{REF} + 0.1	—	V_{REF} – 0.2	V_{REF} + 0.2	0.4	V_{CCIO} – 0.4	8	–8
HSTL-15 Class II	—	V_{REF} – 0.1	V_{REF} + 0.1	—	V_{REF} – 0.2	V_{REF} + 0.2	0.4	V_{CCIO} –0.4	16	–16
HSTL-12 Class I	–0.15	V_{REF} – 0.08	V_{REF} + 0.08	V_{CCIO} + 0.15	V_{REF} – 0.15	V_{REF} + 0.15	0.25 x V_{CCIO}	0.75 x V_{CCIO}	8	–8
HSTL-12 Class II	–0.15	V_{REF} – 0.08	V_{REF} + 0.08	V_{CCIO} + 0.15	V_{REF} – 0.15	V_{REF} + 0.15	0.25 x V_{CCIO}	0.75 x V_{CCIO}	16	–16

Table 2–16. Single-Ended SSTL and HSTL I/O Standards Signal Specifications for Arria V Devices—*Preliminary* (Part 2 of 2)

I/O Standard	$V_{IL(DC)}$ (V)		$V_{IH(DC)}$ (V)		$V_{IL(AC)}$ (V)	$V_{IH(AC)}$ (V)	V_{OL} (V)	V_{OH} (V)	I_{ol} (mA)	I_{oh} (mA)
	Min	Max	Min	Max	Max	Min	Max	Min		
HSUL-12	—	V_{REF} – 0.13	V_{REF} + 0.13	—	V_{REF} – 0.22	V_{REF} + 0.22	0.1 x V_{CCIO}	0.9 x V_{CCIO}	TBD (1)	TBD (1)

Note to Table 2–16:

(1) Pending silicon characterization.

Table 2–17. Differential SSTL I/O Standards for Arria V Devices—*Preliminary*

I/O Standard	V_{CCIO} (V)			$V_{SWING(DC)}$ (V)		$V_{X(AC)}$ (V)			$V_{SWING(AC)}$ (V)		$V_{OX(AC)}$ (V)		
	Min	Typ	Max	Min	Max	Min	Typ	Max	Min	Max	Min	Typ	Max
SSTL-2 Class I, II	2.375	2.5	2.625	0.3	V_{CCIO} + 0.6	V_{CCIO}/2 – 0.2	—	V_{CCIO}/2 + 0.2	0.62	V_{CCIO} + 0.6	V_{CCIO}/2 – 0.15	—	V_{CCIO}/2 + 0.15
SSTL-18 Class I, II	1.71	1.8	1.89	0.25	V_{CCIO} + 0.6	V_{CCIO}/2 – 0.175	—	V_{CCIO}/2 + 0.175	0.5	V_{CCIO} + 0.6	V_{CCIO}/2 – 0.125	—	V_{CCIO}/2 + 0.125
SSTL-15 Class I, II	1.425	1.5	1.575	0.2	–0.2	–0.15	—	0.15	–0.35	0.35	—	V_{CCIO}/2	—
SSTL 135	1.283	1.35	1.45	0.2	–0.2	V_{REF} – 0.135	V_{CCIO}/2	V_{REF} + 0.135	TBD (1)	TBD (1)	V_{REF} –0.15	—	V_{REF} + 0.15
SSTL 125	1.19	1.25	1.31	TBD (1)	—	TBD (1)	V_{CCIO}/2	TBD (1)	TBD (1)	—	TBD (1)	TBD (1)	TBD (1)

Note to Table 2–17:

(1) Pending silicon characterization.

Table 2–18. Differential HSTL I/O Standards for Arria V Devices—*Preliminary*

I/O Standard	V_{CCIO} (V)			$V_{DIF(DC)}$ (V)		$V_{X(AC)}$ (V)			$V_{CM(DC)}$ (V)			$V_{DIF(AC)}$ (V)	
	Min	Typ	Max	Min	Max	Min	Typ	Max	Min	Typ	Max	Min	Max
HSTL-18 Class I, II	1.71	1.8	1.89	0.2	—	0.78	—	1.12	0.78	—	1.12	0.4	—
HSTL-15 Class I, II	1.425	1.5	1.575	0.2	—	0.68	—	0.9	0.68	—	0.9	0.4	—
HSTL-12 Class I, II	1.14	1.2	1.26	0.16	V_{CCIO} + 0.3	—	0.5 x V_{CCIO}	—	0.4 x V_{CCIO}	0.5 x V_{CCIO}	0.6 x V_{CCIO}	0.3	V_{CCIO} + 0.48
HSUL-12	1.14	1.2	1.3	0.26	0.26	0.5 x V_{CCIO} – 0.12	0.5 x V_{CCIO}	0.5 x V_{CCIO} +0.12	0.4 x V_{CCIO}	0.5 x V_{CCIO}	0.6 x V_{CCIO}	0.44	0.44

Table 2–19. Differential I/O Standard Specifications for Arria V Devices—*Preliminary* (1) (Part 1 of 2)

I/O Standard	V_{CCIO} (V)			V_{ID} (mV)			$V_{ICM(DC)}$ (V)		V_{OD} (V) (2)			V_{OCM} (V) (2)		
	Min	Typ	Max	Min	Condition	Max	Min	Max	Min	Typ	Max	Min	Typ	Max
PCML	(3)													

Table 2–19. Differential I/O Standard Specifications for Arria V Devices—*Preliminary* (1) (Part 2 of 2)

I/O Standard	V_{CCIO} (V)			V_{ID} (mV)			$V_{ICM(DC)}$ (V)		V_{OD} (V) (2)			V_{OCM} (V) (2)		
	Min	Typ	Max	Min	Condition	Max	Min	Max	Min	Typ	Max	Min	Typ	Max
2.5 V LVDS	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.05	1.8	0.247	—	0.6	1.125	1.25	1.375
						—	1.05	1.55	0.247	—	0.6	1.125	1.25	1.375
RSDS (HIO)	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.3	1.4	0.1	0.2	0.6	0.5	1.2	1.4
Mini-LVDS (HIO)	2.375	2.5	2.625	200	—	600	0.4	1.325	0.25	—	0.6	1	1.2	1.4
LVPECL	2.375	2.5	2.625	300	—	—	0.6	1.8	—	—	—	—	—	—

Notes to Table 2–19:

(1) The 1.4-V and 1.5-V **PCML** transceiver I/O standard specifications are described in "Transceiver Performance Specifications" on page 2–15.

(2) RL range: $90 \leq RL \leq 110\ \Omega$.

(3) Transmitter, receiver, and input reference clock pins of high-speed transceivers use the **PCML** I/O standard. For transmitter, receiver, and reference clock I/O pin specifications, refer to Table 2–20 and Table 2–21.

Power Consumption

Altera offers two ways to estimate power consumption for a design—the Excel-based Early Power Estimator (EPE) and the Quartus® II PowerPlay Power Analyzer feature.

You typically use the interactive Excel-based EPE before designing the FPGA to get a magnitude estimate of the device power. The Quartus II PowerPlay Power Analyzer provides better quality estimates based on the specifics of the design after you complete place-and-route. The PowerPlay Power Analyzer can apply a combination of user-entered, simulation-derived, and estimated signal activities that, when combined with detailed circuit models, yields very accurate power estimates.

For more information about power estimation tools, refer to the *PowerPlay Early Power Estimator User Guide* and the *PowerPlay Power Analysis* chapter in the *Quartus II Handbook*.

Switching Characteristics

This section provides performance characteristics of Arria V core and periphery blocks for commercial grade devices.

These characteristics can be designated as Preliminary or Final.

- Preliminary characteristics are created using simulation results, process data, and other known parameters. The title of these tables show the designation as "Preliminary."
- Final numbers are based on actual silicon characterization and testing. The numbers reflect the actual performance of the device under worst-case silicon process, voltage, and junction temperature conditions. There are no preliminary designations on finalized tables.

Transceiver Performance Specifications

This section describes transceiver performance specifications.

Table 2–20 and Table 2–21 list the Arria V transceiver specifications.

Table 2–20. Transceiver Specifications for Arria V GX Devices—*Preliminary* (1) (Part 1 of 3)

Symbol/ Description	Conditions	–4 Commercial Speed Grade			–5 Commercial/Industrial Speed Grade			–6 Commercial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Reference Clock											
Supported I/O Standards	**1.2 V PCML, 1.4 V PCML, 1.5 V PCML, 2.5 V PCML, Differential LVPECL** (2), **HCSL**, and **LVDS**										
Input frequency from REFCLK input pins	—	27	—	710	27	—	710	27	—	710	MHz
Duty cycle	—	45	—	55	45	—	55	45	—	55	%
Peak-to-peak differential input voltage	—	200	—	2000	200	—	2000	200	—	2000	mV
Spread-spectrum modulating clock frequency	PCI Express® (PCIe®)	30	—	33	30	—	33	30	—	33	kHz
Spread-spectrum downspread	PCIe	—	0 to –0.5%	—	—	0 to –0.5%	—	—	0 to –0.5%	—	—
On-chip termination resistors	—	—	100	—	—	100	—	—	100	—	Ω
V_{ICM} (AC coupled)	—		1.1 (3)			1.1 (3)			1.1 (3)		V
V_{ICM} (DC coupled)	**HCSL** I/O standard for the PCIe reference clock	250	—	550	250	—	550	250	—	550	mV
R_{REF}	—	—	2000 ±1%	—	—	2000 ±1%	—	—	2000 ±1%	—	Ω
Transceiver Clocks											
fixedclk clock frequency	PCIe Receiver Detect	—	125	—	—	125	—	—	125	—	MHz
Avalon®-Memory-Mapped (Avalon-MM) PHY management clock frequency	< 150										MHz

Table 2–20. Transceiver Specifications for Arria V GX Devices—*Preliminary* (1) (Part 2 of 3)

Symbol/ Description	Conditions	–4 Commercial Speed Grade			–5 Commercial/Industrial Speed Grade			–6 Commercial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Receiver											
Supported I/O Standards	**1.5 V PCML, 2.5 V PCML, LVPECL**, and **LVDS**										
Data rate	—	611	—	6553	611	—	6553	611	—	3125	Mbps
Absolute V_{MAX} for a receiver pin (4)	—	—	—	1.2	—	—	1.2	—	—	1.2	V
Absolute V_{MIN} for a receiver pin	—	–0.4	—	—	–0.4	—	—	–0.4	—	—	V
Maximum peak-to-peak differential input voltage V_{ID} (diff p-p) before device configuration	—	—	—	1.6	—	—	1.6	—	—	1.6	V
Maximum peak-to-peak differential input voltage V_{ID} (diff p-p) after device configuration	—	—	—	2.2	—	—	2.2	—	—	2.2	V
Minimum differential eye opening at the receiver serial input pins (5)	—	85	—	—	85	—	—	85	—	—	mV
Differential on-chip termination resistors	85-Ω setting	—	85	—	—	85	—	—	85	—	Ω
	100-Ω setting	—	100	—	—	100	—	—	100	—	Ω
	120-Ω setting	—	120	—	—	120	—	—	120	—	Ω
	150-Ω setting	—	150	—	—	150	—	—	150	—	Ω
Differential and common mode return loss	PCIe (Gen1 and Gen2), GIGE, XAUI, SDI, CPRI, OBSAI	Compliant									—
Programmable ppm detector (6)	—	±62.5, 100, 125, 200, 250, 300, 500, and 1000									ppm
Run Length	—	—	—	200	—	—	200	—	—	200	UI
Programmable equalization	—	—	—	4	—	—	4	—	—	4	dB
Programmable DC gain	DC Gain Setting = 0	—	0	—	—	0	—	—	0	—	dB
	DC Gain Setting = 1	—	3	—	—	3	—	—	3	—	dB

Table 2–20. Transceiver Specifications for Arria V GX Devices—*Preliminary* (1) (Part 3 of 3)

Symbol/ Description	Conditions	–4 Commercial Speed Grade			–5 Commercial/Industrial Speed Grade			–6 Commercial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Transmitter											
Supported I/O Standards	1.5 V PCML										
Data rate	—	611	—	6553	611	—	6553	611	—	3125	Mbps
V_{OCM}	—	—	650	—	—	650	—	—	650	—	mV
Differential on-chip termination resistors	85-Ω setting	—	85	—	—	85	—	—	85	—	Ω
	100-Ω setting	—	100	—	—	100	—	—	100	—	Ω
	120-Ω setting	—	120	—	—	120	—	—	120	—	Ω
	150-Ω setting	—	150	—	—	150	—	—	150	—	Ω
Rise time (7)	—	30	—	160	30	—	160	30	—	160	ps
Fall time (7)	—	30	—	160	30	—	160	30	—	160	ps
CMU PLL											
Supported data range	—	611	—	6553	611	—	6553	611	—	3125	Mbps
Transceiver-FPGA Fabric Interface											
Interface speed (single-width mode)	—	25	—	187.50	25	—	163.84	25	—	156.25	MHz
Interface speed (double-width mode)	—	25	—	163.84	25	—	163.84	25	—	156.25	MHz

Notes to Table 2–20:

(1) Speed grades shown in Table 2–20 refer to the Transceiver Speed Grade in the device ordering code. For more information about device ordering codes, refer to the *Overview for Arria V Device Family* chapter.

(2) Differential LVPECL signal levels must comply to the minimum and maximum peak-to-peak differential input voltage specified in this table.

(3) The reference clock common mode voltage is equal to the V_{CCR_GXB} power supply level.

(4) The device cannot tolerate prolonged operation at this absolute maximum.

(5) The differential eye opening specification at the receiver input pins assumes that you have disabled the **Receiver Equalization** feature. If you enable the **Receiver Equalization** feature, the receiver circuitry can tolerate a lower minimum eye opening, depending on the equalization level.

(6) The rate match FIFO supports only up to ±300 parts per million (ppm).

(7) The Quartus II software automatically selects the appropriate slew rate depending on the configured data rate or functional mode.

Table 2–21. Transceiver Specifications for Arria V GT Devices—*Preliminary* (1) (Part 1 of 2)

Symbol/ Description	Conditions	–5 Industrial Speed Grade			Unit
		Min	Typ	Max	
Reference Clock					
Supported I/O Standards	**1.2 V PCML**, **1.4 V PCML**, **1.5 V PCML**, **2.5 V PCML**, **Differential LVPECL** (2), **HCSL**, and **LVDS**				
Input frequency from REFCLK input pins	—	27	—	710	MHz
Duty cycle	—	45	—	55	%
Peak-to-peak differential input voltage	—	200	—	2000	mV
Spread-spectrum modulating clock frequency	PCI Express® (PCIe®)	30	—	33	kHz
Spread-spectrum downspread	PCIe	—	0 to –0.5%	—	—
On-chip termination resistors	—	—	100	—	Ω
V_{ICM} (AC coupled)	—	1.1 (3)			V
V_{ICM} (DC coupled)	**HCSL** I/O standard for the PCIe reference clock	250	—	550	mV
R_{REF}	—	—	2000 ±1%	—	Ω
Transceiver Clocks					
fixedclk clock frequency	PCIe Receiver Detect	—	125	—	MHz
Avalon-MM PHY management clock frequency	< 150				MHz
Receiver					
Supported I/O Standards	**1.5 V PCML**, **2.5 V PCML**, **LVPECL**, and **LVDS**				
Data rate (6-Gbps Transceiver)	—	611	—	6375	Mbps
Data rate (10-Gbps transceiver)	—	6.376	9.8304	10.3125	Gbps
Absolute V_{MAX} for a receiver pin (4)	—	—	—	1.2	V
Absolute V_{MIN} for a receiver pin	—	–0.4	—	—	V
Maximum peak-to-peak differential input voltage V_{ID} (diff p-p) before device configuration	—	—	—	1.6	V
Maximum peak-to-peak differential input voltage V_{ID} (diff p-p) after device configuration	—	—	—	2.2	V
Minimum differential eye opening at the receiver serial input pins (5)	—	85	—	—	mV
Differential on-chip termination resistors	85-Ω setting	85			Ω
	100-Ω setting	100			Ω
	120-Ω setting	120			Ω
	150-Ω setting	150			Ω

Table 2–21. Transceiver Specifications for Arria V GT Devices—*Preliminary* (1) (Part 2 of 2)

Symbol/ Description	Conditions	–5 Industrial Speed Grade			Unit
		Min	Typ	Max	
Differential and common mode return loss	PCIe (Gen1 and Gen2), GIGE, XAUI, SDI, CPRI, OBSAI, SFI	Compliant			—
Programmable ppm detector (6)	—	±62.5, 100, 125, 200, 250, 300, 500, and 1000			ppm
Run Length	—	—	—	200	UI
Programmable equalization	—	—	—	4	dB
Programmable DC gain	DC Gain Setting = 0	—	0	—	dB
	DC Gain Setting = 1	—	3	—	dB
Transmitter					
Supported I/O Standards	**1.5 V PCML**				
Data rate (6-Gbps transceiver)	—	611	—	6375	Mbps
Data rate (10-Gbps transceiver)	—	6.376	9.8304	10.3125	Gbps
V_{OCM}	—	—	650	—	mV
Differential on-chip termination resistors	85-Ω setting	—	85	—	Ω
	100-Ω setting	—	100	—	Ω
	120-Ω setting	—	120	—	Ω
	150-Ω setting	—	150	—	Ω
Rise time (7)	—	30	—	160	ps
Fall time (7)	—	30	—	160	ps
CMU PLL					
Supported data range	—	0.611	—	10.3125	Gbps
Transceiver-FPGA Fabric Interface					
Interface speed (80-bit mode)	—	25	—	159.375	MHz
Interface speed (single-width mode)	—	25	—	156.25	MHz
Interface speed (double-width mode)	—	25	—	159.375	MHz

Notes to Table 2–21:

(1) Speed grades shown in Table 2–21 refer to the Transceiver Speed Grade in the device ordering code. For more information about device ordering codes, refer to the *Overview for Arria V Device Family* chapter.

(2) Differential **LVPECL** signal levels must comply to the minimum and maximum peak-to-peak differential input voltage specified in this table.

(3) The reference clock common mode voltage is equal to the V_{CCR_GXB} power supply level.

(4) The device cannot tolerate prolonged operation at this absolute maximum.

(5) The differential eye opening specification at the receiver input pins assumes that you have disabled the **Receiver Equalization** feature. If you enable the **Receiver Equalization** feature, the receiver circuitry can tolerate a lower minimum eye opening, depending on the equalization level.

(6) The rate match FIFO supports only up to ±300 ppm.

(7) The Quartus II software automatically selects the appropriate slew rate depending on the configured data rate or functional mode.

Table 2–22 and Table 2–23 list the transceiver block jitter specification for Arria V devices.

Table 2–22. Transceiver Block Jitter Specification for Arria V GX Devices—*Preliminary* (Part 1 of 4)

Symbol/ Description	Conditions	–4 Commercial Speed Grade			–5 Commercial/ Industrial Speed Grade			–6 Commercial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
CPRI Transmit Jitter Generation (1)											
Total Jitter	E.6.HV, E.12.HV Pattern = CJPAT	—	—	0.279	—	—	0.279	—	—	0.279	UI
	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJTPAT	—	—	0.35	—	—	0.35	—	—	0.35	UI
Deterministic Jitter	E.6.HV, E.12.HV Pattern = CJPAT	—	—	0.14	—	—	0.14	—	—	0.14	UI
	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJTPAT	—	—	0.17	—	—	0.17	—	—	0.17	UI
CPRI Receiver Jitter Tolerance (1)											
Total jitter tolerance	E.6.HV, E.12.HV Pattern = CJPAT	> 0.66			> 0.66			> 0.66			UI
Deterministic jitter tolerance	E.6.HV, E.12.HV Pattern = CJPAT	> 0.4			> 0.4			> 0.4			UI
Total jitter tolerance	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJTPAT	> 0.65			> 0.65			> 0.65			UI
Deterministic jitter tolerance	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJTPAT	> 0.37			> 0.37			> 0.37			UI
Combined deterministic and random jitter tolerance	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJTPAT	> 0.55			> 0.55			> 0.55			UI
OBSAI Transmit Jitter Generation (2)											
Total jitter at 768 Mbps, 1536 Mbps, and 3072 Mbps	REFCLK = 153.6 MHz Pattern = CJPAT	—	—	0.35	—	—	0.35	—	—	0.35	UI
Deterministic jitter at 768 Mbps, 1536 Mbps, and 3072 Mbps	REFCLK = 153.6 MHz Pattern = CJPAT	—	—	0.17	—	—	0.17	—	—	0.17	UI

Table 2–22. Transceiver Block Jitter Specification for Arria V GX Devices—*Preliminary* (Part 2 of 4)

Symbol/ Description	Conditions	–4 Commercial Speed Grade			–5 Commercial/ Industrial Speed Grade			–6 Commercial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
OBSAI Receiver Jitter Tolerance (2)											
Deterministic jitter tolerance at 768 Mbps, 1536 Mbps, and 3072 Mbps	Pattern = CJPAT		> 0.37			> 0.37			> 0.37		UI
Combined deterministic and random jitter tolerance at 768 Mbps, 1536 Mbps, and 3072 Mbps	Pattern = CJPAT		> 0.55			> 0.55			> 0.55		UI
Sinusoidal Jitter tolerance at 768 Mbps	Jitter Frequency = 5.4 KHz Pattern = CJPAT		> 8.5			> 8.5			> 8.5		UI
	Jitter Frequency = 460 MHz to 20 MHz Pattern = CJPAT		> 0.1			> 0.1			> 0.1		UI
Sinusoidal Jitter tolerance at 1536 Mbps	Jitter Frequency = 10.9 KHz Pattern = CJPAT		> 8.5			> 8.5			> 8.5		UI
	Jitter Frequency = 921.6 MHz to 20 MHz Pattern = CJPAT		> 0.1			> 0.1			> 0.1		UI
Sinusoidal Jitter tolerance at 3072 Mbps	Jitter Frequency = 21.8 KHz Pattern = CJPAT		> 8.5			> 8.5			> 8.5		UI
	Jitter Frequency = 1843.2 MHz to 20 MHz Pattern = CJPAT		> 0.1			> 0.1			> 0.1		UI

Table 2–22. Transceiver Block Jitter Specification for Arria V GX Devices—*Preliminary* (Part 3 of 4)

Symbol/ Description	Conditions	–4 Commercial Speed Grade			–5 Commercial/ Industrial Speed Grade			–6 Commercial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Serial RapidIO® (SRIO) Transmit Jitter Generation *(3)*											
Deterministic jitter (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	—	—	0.17	—	—	0.17	—	—	0.17	UI
Total jitter (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	—	—	0.35	—	—	0.35	—	—	0.35	UI
SRIO Receiver Jitter Tolerance *(3)*											
Deterministic jitter tolerance (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.37			> 0.37			> 0.37			UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.55			> 0.55			> 0.55			UI
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 22.1 KHz Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 8.5			> 8.5			> 8.5			UI
	Jitter Frequency = 1.875 MHz Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.1			> 0.1			> 0.1			UI
	Jitter Frequency = 20 MHz Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.1			> 0.1			> 0.1			UI
GIGE Transmit Jitter Generation *(4)*											
Deterministic jitter (peak-to-peak)	Pattern = CRPAT	—	—	0.14	—	—	0.14	—	—	0.14	UI
Total jitter (peak-to-peak)	Pattern = CRPAT	—	—	0.279	—	—	0.279	—	—	0.279	UI

Table 2–22. Transceiver Block Jitter Specification for Arria V GX Devices—*Preliminary* (Part 4 of 4)

Symbol/ Description	Conditions	–4 Commercial Speed Grade			–5 Commercial/ Industrial Speed Grade			–6 Commercial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
GIGE Receiver Jitter Tolerance *(4)*											
Deterministic jitter tolerance (peak-to-peak)	Pattern = CJPAT		> 0.4			> 0.4			> 0.4		UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Pattern = CJPAT		> 0.66			> 0.66			> 0.66		UI
HiGig Transmit Jitter Generation *(5)*											
Deterministic jitter (peak-to-peak)	Data Rate = 3.75 Gbps Pattern = CJPAT	—	—	0.17	—	—	—	—	—	—	UI
Total jitter (peak-to-peak)	Data Rate = 3.75 Gbps Pattern = CJPAT	—	—	0.35	—	—	—	—	—	—	UI

Notes to Table 2–22:

(1) The jitter numbers for CPRI are compliant to the CPRI Specification V3.0.
(2) The jitter numbers for OBSAI are compliant to the OBSAI RP3 Specification V4.1.
(3) The jitter numbers for SRIO are compliant to the RapidIO Specification 1.3.
(4) The jitter numbers for GIGE are compliant to the IEEE802.3-2002 Specification.
(5) The jitter numbers for HiGig are compliant to the IEEE802.3ae-2002 Specification.

Table 2–23. Transceiver Block Jitter Specification for Arria V GT Devices—*Preliminary* (Part 1 of 3)

Symbol/ Description	Conditions	–5 Industrial Speed Grade			Unit
		Min	Typ	Max	
SFI Transmit Jitter Generation (1)					
Deterministic Jitter	Data Rate = 9.8304, 10.3125 Gbps	—	—	0.1	UI
Total Jitter	Pattern = PRBS31	—	—	0.28	UI
SFI Receive Jitter Tolerance (1)					
99% Jitter Tolerance	Data Rate = 9.8304, 10.3125 Gbps		>0.42		UI
Total Jitter	Pattern = PRBS31		>0.70		UI
CPRI Transmit Jitter Generation (2)					
Total Jitter	E.6.HV, E.12.HV Pattern = CJPAT	—	—	0.279	—
	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJTPAT	—	—	0.35	—
Deterministic Jitter	E.6.HV, E.12.HV Pattern = CJPAT	—	—	0.14	—
	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJTPAT	—	—	0.17	—
CPRI Receive Jitter Generation (2)					
Total jitter tolerance	E.6.HV, E.12.HV Pattern = CJPAT		> 0.66		> 0.66
Deterministic jitter tolerance	E.6.HV, E.12.HV Pattern = CJPAT		> 0.4		> 0.4
Total jitter tolerance	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJTPAT		> 0.65		> 0.65
Deterministic jitter tolerance	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJTPAT		> 0.37		> 0.37
Combined deterministic and random jitter tolerance	E.6.LV, E.12.LV, E.24.LV, E.30.LV Pattern = CJTPAT		> 0.55		> 0.55
OBSAI Transmit Jitter Generation (3)					
Total jitter at 768 Mbps, 1536 Mbps, and 3072 Mbps	REFCLK = 153.6MHz Pattern = CJPAT	—	—	0.35	—
Deterministic jitter at 768 Mbps, 1536 Mbps, and 3072 Mbps	REFCLK = 153.6MHz Pattern = CJPAT	—	—	0.17	—

Table 2–23. Transceiver Block Jitter Specification for Arria V GT Devices—*Preliminary* (Part 2 of 3)

Symbol/ Description	Conditions	–5 Industrial Speed Grade			Unit
		Min	Typ	Max	
OBSAI Receiver Jitter Tolerance *(3)*					
Deterministic jitter tolerance at 768 Mbps, 1536 Mbps, and 3072 Mbps	Pattern = CJPAT		> 0.37		> 0.37
Combined deterministic and random jitter tolerance at 768 Mbps, 1536 Mbps, and 3072 Mbps	Pattern = CJPAT		> 0.55		> 0.55
Sinusoidal Jitter tolerance at 768 Mbps	Jitter Frequency = 5.4 KHz Pattern = CJPAT		> 8.5		> 8.5
	Jitter Frequency = 460 MHz to 20 MHz Pattern = CJPAT		> 0.1		> 0.1
Sinusoidal Jitter tolerance at 1536 Mbps	Jitter Frequency = 10.9 KHz Pattern = CJPAT		> 8.5		> 8.5
	Jitter Frequency = 921.6 MHz to 20 MHz Pattern = CJPAT		> 0.1		> 0.1
Sinusoidal Jitter tolerance at 3072 Mbps	Jitter Frequency = 21.8 KHz Pattern = CJPAT		> 8.5		> 8.5
	Jitter Frequency = 1843.2 MHz to 20 MHz Pattern = CJPAT		> 0.1		> 0.1
SRIO Transmit Jitter Generation *(4)*					
Deterministic jitter (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	—	—	0.17	UI
Total jitter (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	—	—	0.35	UI

Table 2–23. Transceiver Block Jitter Specification for Arria V GT Devices—*Preliminary* (Part 3 of 3)

Symbol/ Description	Conditions	–5 Industrial Speed Grade			Unit
		Min	Typ	Max	
SRIO Receiver Jitter Tolerance (4)					
Deterministic jitter tolerance (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT		> 0.37		UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT		> 0.55		UI
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 22.1 KHz Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT		> 8.5		UI
	Jitter Frequency = 1.875 MHz Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT		> 0.1		UI
	Jitter Frequency = 20 MHz Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT		> 0.1		UI
GIGE Transmit Jitter Generation (5)					
Deterministic jitter (peak-to-peak)	Pattern = CRPAT	—	—	0.14	UI
Total jitter (peak-to-peak)	Pattern = CRPAT	—	—	0.279	UI
GIGE Receiver Jitter Tolerance (5)					
Deterministic jitter tolerance (peak-to-peak)	Pattern = CJPAT		> 0.4		UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Pattern = CJPAT		> 0.66		UI
HiGig Transmit Jitter Generation (6)					
Deterministic jitter (peak-to-peak)	Data Rate = 3.75 Gbps Pattern = CJPAT	—	—	0.17	UI
Total jitter (peak-to-peak)	Data Rate = 3.75 Gbps Pattern = CJPAT	—	—	0.35	UI

Notes to Table 2–23:

(1) The jitter numbers for SFI are compliant to SFF-8431 Specification.
(2) The jitter numbers for CPRI are compliant to the CPRI Specification V3.0.
(3) The jitter numbers for OBSAI are compliant to the OBSAI RP3 Specification V4.1.
(4) The jitter numbers for SRIO are compliant to the RapidIO Specification 1.3.
(5) The jitter numbers for GIGE are compliant to the IEEE802.3-2002 Specification.
(6) The jitter numbers for HiGig are compliant to the IEEE802.3ae-2002 Specification.

Core Performance Specifications

This section describes the clock tree, phase-locked loop (PLL), digital signal processing (DSP), memory blocks and temperature sensing diode specifications.

Clock Tree Specifications

Table 2–24 lists the clock tree specifications for Arria V devices.

Table 2–24. Clock Tree Performance for Arria V Devices—*Preliminary*

Performance				Unit
Symbol	–C4 Speed Grade	–C5, I5 Speed Grade	–C6 Speed Grade	
Global clock and Regional clock	625	625	525	MHz
Peripheral clock	450	400	350	MHz

PLL Specifications

Table 2–25 lists the Arria V PLL specifications when operating in both the commercial junction temperature range (0° to 85°C) and the industrial junction temperature range (–40° to 100°C).

Table 2–25. PLL Specifications for Arria V Devices—*Preliminary* (1) (Part 1 of 3)

Symbol	Parameter	Min	Typ	Max	Unit
f_{IN}	Input clock frequency (–4 speed grade)	5	—	670 (2)	MHz
	Input clock frequency (–5 speed grade)	5	—	622 (2)	MHz
	Input clock frequency (–6 speed grade)	5	—	500 (2)	MHz
f_{INPFD}	Integer input clock frequency to the PFD	5	—	325	MHz
f_{FINPFD}	Fractional input clock frequency to the PFD	50	—	TBD (1)	MHz
f_{VCO}	PLL VCO operating range (–4 speed grade)	600	—	1600	MHz
	PLL VCO operating range (–5 speed grade)	600	—	1400	MHz
	PLL VCO operating range (–6 speed grade)	600	—	1300	MHz
$t_{EINDUTY}$	Input clock or external feedback clock input duty cycle	40	—	60	%
f_{OUT}	Output frequency for internal global or regional clock (–4 speed grade)	—	—	500 (3)	MHz
	Output frequency for internal global or regional clock (–5 speed grade)	—	—	500 (3)	MHz
	Output frequency for internal global or regional clock (–6 speed grade)	—	—	400 (3)	MHz
f_{OUT_EXT}	Output frequency for external clock output (–4 speed grade)	—	—	670 (3)	MHz
	Output frequency for external clock output (–5 speed grade)	—	—	622 (3)	MHz
	Output frequency for external clock output (–6 speed grade)	—	—	500 (3)	MHz
$t_{OUTDUTY}$	Duty cycle for external clock output (when set to 50%)	45	50	55	%
t_{FCOMP}	External feedback clock compensation time	—	—	10	ns
$t_{CONFIGPHASE}$	Time required to reconfigure phase shift	—	—	TBD (1)	—
$t_{DYCONFIGCLK}$	Dynamic Configuration Clock	—	—	100	MHz

Table 2–25. PLL Specifications for Arria V Devices—*Preliminary* (1) (Part 2 of 3)

Symbol	Parameter	Min	Typ	Max	Unit
t_{LOCK}	Time required to lock from end-of-device configuration or deassertion of areset	—	—	1	ms
t_{DLOCK}	Time required to lock dynamically (after switchover or reconfiguring any non-post-scale counters/delays)	—	—	1	ms
f_{CLBW}	PLL closed-loop low bandwidth	—	0.3	—	MHz
	PLL closed-loop medium bandwidth	—	1.5	—	MHz
	PLL closed-loop high bandwidth (8)	—	4	—	MHz
t_{PLL_PSERR}	Accuracy of PLL phase shift	—	—	±50	ps
t_{ARESET}	Minimum pulse width on the areset signal	10	—	—	ns
t_{INCCJ} (4), (5)	Input clock cycle-to-cycle jitter ($F_{REF} \geq 100$ MHz)	—	—	0.15	UI (p-p)
	Input clock cycle-to-cycle jitter ($F_{REF} < 100$ MHz)	—	—	+750	ps (p-p)
t_{OUTPJ_DC} (6)	Period jitter for dedicated clock output ($F_{OUT} \geq 100$ MHz)	—	—	TBD (1)	ps (p-p)
	Period jitter for dedicated clock output ($F_{OUT} < 100$ MHz)	—	—	TBD (1)	mUI (p-p)
t_{OUTCCJ_DC} (6)	Cycle-to-cycle jitter for dedicated clock output ($F_{OUT} \geq 100$ MHz)	—	—	TBD (1)	ps (p-p)
	Cycle-to-cycle jitter for dedicated clock output ($F_{OUT} < 100$ MHz)	—	—	TBD (1)	mUI (p-p)
t_{OUTPJ_IO} (6), (9)	Period Jitter for clock output on the regular I/O ($F_{OUT} \geq 100$ MHz)	—	—	TBD (1)	ps (p-p)
	Period Jitter for clock output on the regular I/O ($F_{OUT} < 100$ MHz)	—	—	TBD (1)	mUI (p-p)
t_{OUTCCJ_IO} (6), (9)	Cycle-to-cycle jitter for clock output on the regular I/O ($F_{OUT} \geq 100$ MHz)	—	—	TBD (1)	ps (p-p)
	Cycle-to-cycle jitter for clock output on the regular I/O ($F_{OUT} < 100$ MHz)	—	—	TBD (1)	mUI (p-p)
$t_{OUTPJ\_DC\_F}$	Period jitter for dedicated clock output in fractional mode	—	—	TBD (1)	—
$t_{OUTCCJ\_DC\_F}$	Cycle-to-cycle jitter for dedicated clock output in fractional mode	—	—	TBD (1)	—
$t_{OUTPJ\_IO\_F}$	Period Jitter for clock output on the regular I/O in fractional mode	—	—	TBD (1)	—
$t_{OUTCCJ\_IO\_F}$	Cycle-to-cycle jitter for clock output on the regular I/O in fractional mode	—	—	TBD (1)	—
$t_{CASC\_OUTPJ\_DC}$ (6), (7)	Period jitter for dedicated clock output in cascaded PLLs ($F_{OUT} \geq 100$ MHz)	—	—	TBD (1)	ps (p-p)
	Period jitter for dedicated clock output in cascaded PLLs ($F_{OUT} < 100$ MHz)	—	—	TBD (1)	mUI (p-p)
t_{DRIFT}	Frequency drift after PFDENA is disabled for a duration of 100 µs	—	—	±10	%

Table 2–25. PLL Specifications for Arria V Devices—*Preliminary* (1) (Part 3 of 3)

Symbol	Parameter	Min	Typ	Max	Unit
dK_{BIT}	Bit number of Delta Sigma Modulator (DSM)	—	24	—	bits
k_{VALUE}	Numerator of Fraction	TBD (1)	8388608	TBD (1)	—
f_{RES}	Resolution of VCO frequency (f_{INPFD} =100 MHz)	—	5.96	—	Hz

Notes to Table 2–25:

(1) Pending silicon characterization.

(2) This specification is limited in the Quartus II software by the I/O maximum frequency. The maximum I/O frequency is different for each I/O standard.

(3) This specification is limited by the lower of the two: I/O f_{MAX} or F_{OUT} of the PLL.

(4) A high input jitter directly affects the PLL output jitter. To have low PLL output clock jitter, you must provide a clean clock source < 120 ps.

(5) F_{REF} is fIN/N when N = 1.

(6) Peak-to-peak jitter with a probability level of 10^{-12} (14 sigma, 99.99999999974404 % confidence level). The output jitter specification applies to the intrinsic jitter of the PLL, when an input jitter of 30 ps is applied. The external memory interface clock output jitter specifications use a different measurement method and are available in Table 2–33 on page 2–35.

(7) The cascaded PLL specification is only applicable with the following conditions:
a. Upstream PLL: 0.59 MHz ≤ Upstream PLL BW < 1 MHz
b. Downstream PLL: Downstream PLL BW > 2 MHz

(8) High bandwidth PLL settings are not supported in external feedback mode.

(9) External memory interface clock output jitter specifications use a different measurement method, which are available in Table 2–33 on page 2–35.

DSP Block Specifications

Table 2–26 lists the Arria V DSP block performance specifications.

Table 2–26. DSP Block Performance Specifications for Arria V Devices—*Preliminary*

Mode	Performance			Unit
	–C4 Speed Grade	–C5, I5 Speed Grade	–C6 Speed Grade	
Modes using One DSP Block				
Independent 9 x 9 Multiplication	370	310	220	MHz
Independent 18 x 19 Multiplication	370	310	220	MHz
Independent 18 x 18 Multiplication	370	310	220	MHz
Independent 27 x 27 Multiplication	310	250	200	MHz
Independent 18 x 25 Multiplication	370	310	220	MHz
Independent 20 x 24 Multiplication	370	310	220	MHz
Two 18 x 19 Multiplier Adder Mode	370	310	220	MHz
18 x 18 Multiplier Added Summed with 36-bit Input	370	310	220	MHz
Modes using Two DSP Blocks				
Complex 18 x 19 multiplication	370	310	220	MHz
Two 27 x 27 Multiplier Adder	310	250	200	MHz
Four 18 x 19 Multiplier Adder	370	310	220	MHz

Memory Block Specifications

Table 2–27 lists the Arria V memory block specifications.

Table 2–27. Memory Block Performance Specifications for Arria V Devices—*Preliminary* (1), (2)

Memory	Mode	Resources Used		Performance			Unit
		ALUTs	Memory	C4 Speed Grade	C5,I5 Speed Grade	C6 Speed Grade	
MLAB	Single port, all supported widths	0	1	500	450	400	MHz
	Simple dual-port, all supported widths	0	1	500	450	400	MHz
	Simple dual-port with read and write at the same address	0	1	400	350	300	MHz
	ROM, all supported width	—	—	500	450	400	MHz
M10K Block	Single-port, all supported widths	0	1	400	350	285	MHz
	Simple dual-port, all supported widths	0	1	400	350	285	MHz
	Simple dual-port with the read-during-write option set to **Old Data**, all supported widths	0	1	315	275	240	MHz
	True dual port, all supported widths	0	1	400	350	285	MHz
	ROM, all supported widths	0	1	400	350	285	MHz
	Min Pulse Width (clock high time)	—	—	1,275	1,360	1,445	ps
	Min Pulse Width (clock low time)	—	—	850	1,060	1,175	ps

Notes to Table 2–27:

(1) To achieve the maximum memory block performance, use a memory block clock that comes through global clock routing from an on-chip PLL set to **50%** output duty cycle. Use the Quartus II software to report timing for this and other memory block clocking schemes.

(2) When you use the error detection cyclical redundancy check (CRC) feature, there is no degradation in f_{MAX}.

Temperature Sensing Diode Specifications

Table 2–28 lists the specifications for the Arria V internal temperature sensing diode.

Table 2–28. Internal Temperature Sensing Diode Specifications for Arria V Devices—*Preliminary*

Temperature Range	Accuracy	Offset Calibrated Option	Sampling Rate	Conversion Time	Resolution	Minimum Resolution with no Missing Codes
–40 to 100°C	±8°C	No	Frequency: 1 MHz	< 100 ms	8 bits	8 bits

Periphery Performance

This section describes periphery performance and the high-speed I/O and external memory interface.

I/O performance supports several system interfaces, such as the LVDS high-speed I/O interface, external memory interface, and the PCI/PCI-X bus interface. GPIO standards such as 3.3-, 2.5-, 1.8-, and 1.5-V **LVTTL/LVCMOS** are capable of a typical 167 MHz and 1.2 **LVCMOS** at 100 MHz interfacing frequency with a 10 pF load.

☞ Actual achievable frequency depends on design- and system-specific factors. You must perform HSPICE/IBIS simulations based on your specific design and system setup to determine the maximum achievable frequency in your system.

High-Speed I/O Specification

Table 2–29 lists high-speed I/O timing for Arria V devices.

Table 2–29. High-Speed I/O Specifications for Arria V Devices—*Preliminary* (1), (2), (3) (Part 1 of 3)

Symbol	Conditions	–4 Speed Grade			–5 Speed Grade			–6 Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
f_{HSCLK_in} (input clock frequency) True Differential I/O Standards	Clock boost factor W = 1 to 40 (5)	5	—	625	5	—	625	5	—	TBD	MHz
f_{HSCLK_in} (input clock frequency) Single Ended I/O Standards (4)	Clock boost factor W = 1 to 40 (5)	5	—	625	5	—	625	5	—	TBD	MHz
f_{HSCLK_in} (input clock frequency) Single Ended I/O Standards (3)	Clock boost factor W = 1 to 40 (5)	5	—	TBD	5	—	TBD	5	—	TBD	MHz
f_{HSCLK_OUT} (output clock frequency)	—	5	—	625 (6)	5	—	625 (6)	5	—	TBD (6)	MHz

Table 2–29. High-Speed I/O Specifications for Arria V Devices—*Preliminary*(1), (2), (3) (Part 2 of 3)

Symbol	Conditions	–4 Speed Grade			–5 Speed Grade			–6 Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Transmitter											
True Differential I/O Standards - f_{HSDR} (data rate)	SERDES factor J = 3 to 10	(7)	—	1250	(7)	—	1250	(7)	—	1050	Mbps
	SERDES factor J = 1 to 2, Uses DDR Registers	(7)	—	(7)	(7)	—	(7)	(7)	—	(7)	Mbps
Emulated Differential I/O Standards with Three External Output Resistor Networks - f_{HSDR} (data rate) (8)	SERDES factor J = 4 to 10	(7)	—	TBD	(7)	—	TBD	(7)	—	TBD	Mbps
$t_{x\ Jitter}$ - True Differential I/O Standards	Total Jitter for Data Rate, 600 Mbps - 1.25 Gbps	—	—	160	—	—	160	—	—	160	ps
	Total Jitter for Data Rate, < 600 Mbps	—	—	0.1	—	—	0.1	—	—	0.1	UI
$t_{x\ Jitter}$ - Emulated Differential I/O Standards with Three External Output Resistor Network	Total Jitter for Data Rate, 600 Mbps – 1.25 Gbps	—	—	TBD	—	—	TBD	—	—	TBD	ps
	Total Jitter for Data Rate < 600 Mbps	—	—	TBD	—	—	TBD	—	—	TBD	UI
t_{DUTY}	TX output clock duty cycle for both True and Emulated Differential I/O Standards	45	50	55	45	50	55	45	50	55	%
t_{RISE} & t_{FALL}	True Differential I/O Standards	—	—	200	—	—	200	—	—	200	ps
	Emulated Differential I/O Standards with Three External Output Resistor Networks	—	—	250	—	—	250	—	—	300	ps
TCCS	True Differential I/O Standards	—	—	150	—	—	150	—	—	150	ps
	Emulated Differential I/O Standards	—	—	300	—	—	300	—	—	300	ps

Table 2–29. High-Speed I/O Specifications for Arria V Devices—*Preliminary*(1), (2), (3) (Part 3 of 3)

Symbol	Conditions	–4 Speed Grade			–5 Speed Grade			–6 Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Receiver											
True Differential I/O Standards - $f_{HSDRDPA}$ (data rate)	SERDES factor J = 3 to 10	—	—	1250	—	—	1250	—	—	1050	Mbps
f_{HSDR} (data rate)	SERDES factor J = 3 to 10	(7)	—	(9)	(7)	—	(9)	(7)	—	(9)	Mbps
	SERDES factor J = 1 to 2, Uses DDR Registers	(7)	—	(7)	(7)	—	(7)	(7)	—	(7)	Mbps
DPA Mode											
DPA run length	—	—	—	10000	—	—	10000	—	—	10000	UI
Soft CDR mode											
Soft-CDR ppm tolerance	—	—	—	300	—	—	300	—	—	300	± ppm
Non DPA Mode											
Sampling Window	—	—	—	300	—	—	300	—	—	300	ps

Notes to Table 2–29:

(1) When J = 3 to 10, use the serializer/deserializer (SERDES) block.
(2) When J = 1 or 2, bypass the SERDES block.
(3) This applies to LVDS source synchronous mode only.
(4) This applies to DPA and soft-CDR modes only.
(5) Clock Boost Factor (W) is the ratio between the input data rate and the input clock rate.
(6) This is achieved by using the LVDS clock network.
(7) The minimum specification depends on the clock source (for example, the PLL and clock pin) and the clock routing resource (global, regional, or local) that you use. The I/O differential buffer and input register do not have a minimum toggle rate.
(8) You must calculate the leftover timing margin in the receiver by performing link timing closure analysis. You must consider the board skew margin, transmitter channel-to-channel skew, and receiver sampling margin to determine the leftover timing margin.
(9) You can estimate the achievable maximum data rate for non-DPA mode by performing link timing closure analysis. You must consider the board skew margin, transmitter delay margin, and receiver sampling margin to determine the maximum data rate supported.

Figure 2–1 shows the DPA lock time specifications with the DPA PLL calibration option enabled.

Figure 2–1. DPA Lock Time Specification with DPA PLL Calibration Enabled

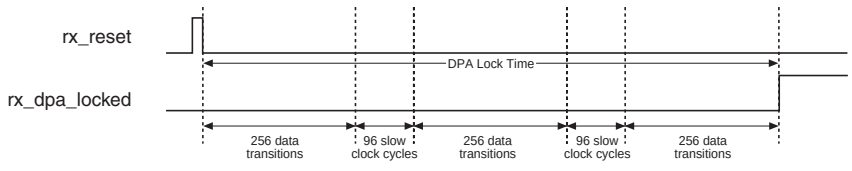


Table 2–30 lists the DPA lock time specifications for Arria V devices.

Table 2–30. DPA Lock Time Specifications for Arria V Devices—*Preliminary* *(1), (2), (3)*

Standard	Training Pattern	Number of Data Transitions in One Repetition of the Training Pattern	Number of Repetitions per 256 Data Transitions *(4)*	Maximum
SPI-4	00000000001111111111	2	128	640 data transitions
Parallel Rapid I/O	00001111	2	128	640 data transitions
	10010000	4	64	640 data transitions
Miscellaneous	10101010	8	32	640 data transitions
	01010101	8	32	640 data transitions

Notes to Table 2–30:

(1) The DPA lock time is for one channel.
(2) One data transition is defined as a 0-to-1 or 1-to-0 transition.
(3) The DPA lock time stated in this table applies to both commercial and industrial grades.
(4) This is the number of repetitions for the stated training pattern to achieve the 256 data transitions.

Figure 2–2 shows the LVDS soft-CDR/DPA sinusoidal jitter tolerance specification for a data rate equal to 1.25 Gbps.

Figure 2–2. LVDS Soft-CDR/DPA Sinusoidal Jitter Tolerance Specification for a Data Rate Equal to 1.25 Gbps

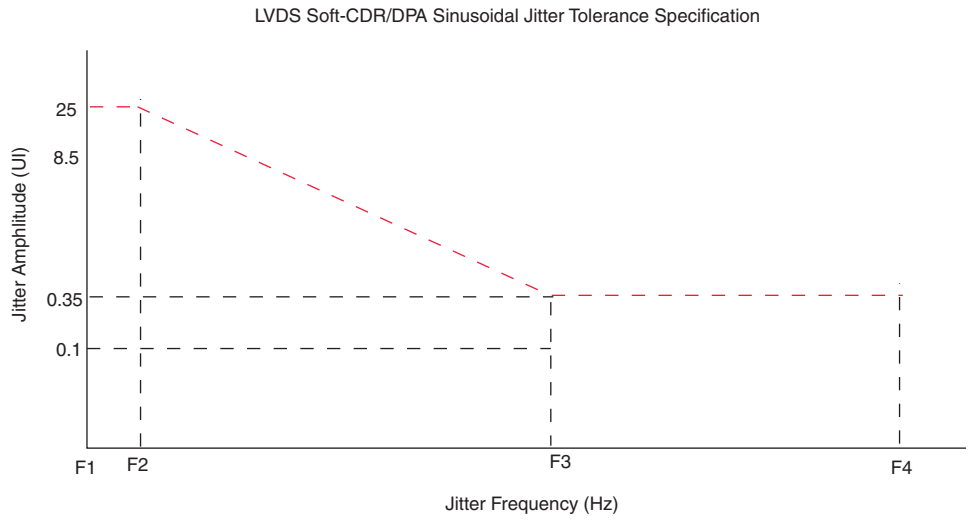


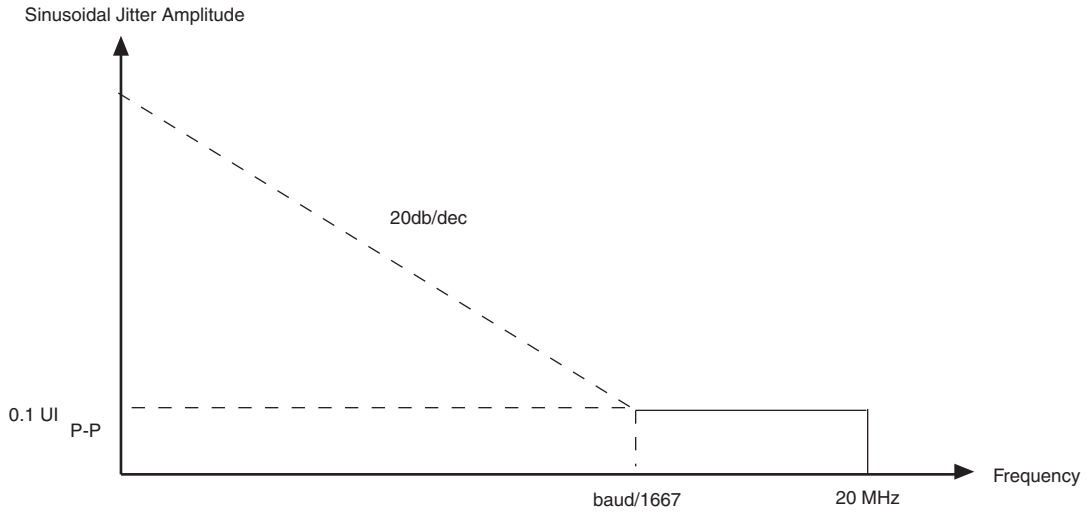
Table 2–31 lists the LVDS soft-CDR/DPA sinusoidal jitter tolerance specification for a data rate equal to 1.25 Gbps.

Table 2–31. LVDS Soft-CDR/DPA Sinusoidal Jitter Mask Values for a Data Rate Equal to 1.25 Gbps—*Preliminary*

Jitter Frequency (Hz)		Sinusoidal Jitter (UI)
F1	10,000	25.000
F2	17,565	25.000
F3	1,493,000	0.350
F4	50,000,000	0.350

Figure 2–3 shows the LVDS soft-CDR/DPA sinusoidal jitter tolerance specification for a data rate less than 1.25 Gbps.

Figure 2–3. LVDS Soft-CDR/DPA Sinusoidal Jitter Tolerance Specification for a Data Rate Less than 1.25 Gbps



DQS Logic Block and Memory Output Clock Jitter Specifications

Table 2–32 lists the DQS phase shift error for Arria V devices.

Table 2–32. DQS Phase Shift Error Specification for DLL-Delayed Clock (t_{DQS_PSERR}) for Arria V Devices—*Preliminary* (1), (2)

Number of DQS Delay Buffer	–C4 Speed Grade	–C5, I5 Speed Grade	–C6 Speed Grade	Unit
2	57	58	74	ps

Notes to Table 2–32:

(1) The numbers are preliminary pending silicon characterization.

(2) This error specification is the absolute maximum and minimum error. For example, skew on two DQS delay buffers in a –4 speed grade is 58 ps or ±29 ps.

Table 2–33 lists the memory output clock jitter specifications for Arria V devices.

Table 2–33. Memory Output Clock Jitter Specification for Arria V Devices—*Preliminary* (1) (Part 1 of 2)

Parameter	Clock Network	Symbol	–4 Speed Grade		–5 Speed Grade		–6 Speed Grade		Unit
			Min	Max	Min	Max	Min	Max	
Clock period jitter	Regional	$t_{JIT(per)}$	–50	50	–55	55	–55	55	ps
Cycle-to-cycle period jitter	Regional	$t_{JIT(cc)}$	–100	100	–110	110	–110	110	ps
Duty cycle jitter	Regional	$t_{JIT(duty)}$	–50	50	–82.5	82.5	–82.5	82.5	ps
Clock period jitter	Global	$t_{JIT(per)}$	–75	75	–82.5	82.5	–82.5	82.5	ps
Cycle-to-cycle period jitter	Global	$t_{JIT(cc)}$	–150	150	–165	165	–165	165	ps

Table 2–33. Memory Output Clock Jitter Specification for Arria V Devices—*Preliminary* (1) (Part 2 of 2)

Parameter	Clock Network	Symbol	–4 Speed Grade		–5 Speed Grade		–6 Speed Grade		Unit
			Min	Max	Min	Max	Min	Max	
Duty cycle jitter	Global	$t_{JIT(duty)}$	–75	75	–90	90	–90	90	ps

Note to Table 2–33:

(1) The memory output clock jitter measurements are for 200 consecutive clock cycles, as specified in the JEDEC DDR2/DDR3 SDRAM standard.

OCT Calibration Block Specifications

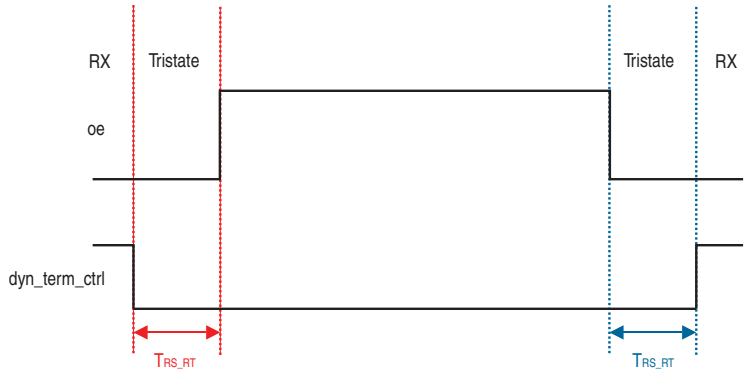
Table 2–34 lists the OCT calibration block specifications for Arria V devices.

Table 2–34. OCT Calibration Block Specifications for Arria V Devices—*Preliminary*

Symbol	Description	Min	Typ	Max	Unit
OCTUSRCLK	Clock required by OCT calibration blocks	—	—	20	MHz
T_{OCTCAL}	Number of OCTUSRCLK clock cycles required for R_S OCT /R_T OCT calibration	—	1000	—	Cycles
$T_{OCTSHIFT}$	Number of OCTUSRCLK clock cycles required for OCT code to shift out	—	32	—	Cycles
T_{RS_RT}	Time required between the `dyn_term_ctrl` and `oe` signal transitions in a bidirectional I/O buffer to dynamically switch between R_S OCT and R_T OCT	—	2.5	—	ns

Figure 2–4 shows the T_{RS_RT} for `dyn_term_ctrl` and `oe` signals.

Figure 2–4. Timing Diagram for dyn_term_ctrl and oe Signals



Duty Cycle Distortion (DCD) Specifications

Table 2–35 lists the worst-case DCD for Arria V devices.

Table 2–35. Worst-Case DCD on Arria V I/O Pins—*Preliminary*

Symbol	–C4 Speed Grade		–C5,I5 Speed Grade		–C6 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
Output Duty Cycle	45	55	45	55	45	55	%

Configuration Specification

This section provides configuration specifications and timing for Arria V devices.

These characteristics can be designated as Preliminary or Final.

- Preliminary characteristics are created using simulation results, process data, and other known parameters. The title of these tables show the designation as "Preliminary."
- Final numbers are based on actual silicon characterization and testing. The numbers reflect the actual performance of the device under worst-case silicon process, voltage, and junction temperature conditions. There are no designations on finalized tables.

POR Specifications

Table 2–36 lists the specifications for fast and standard POR for Arria V devices.

Table 2–36. Fast and Standard POR Delay Specification for Arria V Devices *(1)*

POR Delay	Minimum (ms)	Maximum (ms)
Fast *(2)*	4	12
Standard *(3)*	100	300

Notes to Table 2–36:

(1) Select the POR delay based on the MSEL setting as described in the "Configuration Schemes for Arria V Devices" table in the *Configuration, Design Security, and Remote System Upgrades in Arria V Devices* chapter.

(2) When the PORSEL signal is **high**, the device is in fast POR delay.

(3) When the PORSEL signal is **low**, the device is in standard POR delay.

JTAG Configuration Timing

Table 2–37 lists the JTAG timing parameters and values for Arria V devices.

Table 2–37. JTAG Timing Parameters and Values for Arria V Devices—*Preliminary*

Symbol	Description	Min	Max	Unit
t_{JCP}	TCK clock period	30	—	ns
t_{JCH}	TCK clock high time	14	—	ns
t_{JCL}	TCK clock low time	14	—	ns
$t_{JPSU\ (TDI)}$	TDI JTAG port setup time	1	—	ns
$t_{JPSU\ (TMS)}$	TMS JTAG port setup time	3	—	ns
t_{JPH}	JTAG port hold time	5	—	ns
t_{JPCO}	JTAG port clock to output	—	11 *(1)*	ns
t_{JPZX}	JTAG port high impedance to valid output	—	14 *(1)*	ns
t_{JPXZ}	JTAG port valid output to high impedance	—	14 *(1)*	ns

Note to Table 2–37:

(1) A 1-ns adder is required for each V_{CCIO} voltage step down from 3.0 V. For example, t_{JPCO} = 12 ns if V_{CCIO} of the TDO I/O bank = 2.5 V, or 13 ns if it equals 1.8 V.

FPP Configuration Timing

This section describes the fast passive parallel (FPP) configuration timing parameters for Arria V devices.

DCLK-to-DATA[] Ratio (r) for FPP Configuration

FPP configuration requires a different DCLK-to-DATA[] ratio when you turn on encryption or the compression feature.

Table 2–38 lists the DCLK-to-DATA[] ratio for each combination.

Table 2–38. DCLK-to-DATA[] Ratio for Arria V Devices *(1)*

Configuration Scheme	Encryption	Compression	DCLK-to-DATA[] ratio (r)
FPP (8-bit wide)	Off	Off	1
	On	Off	1
	Off	On	2
	On	On	2
FPP (16-bit wide)	Off	Off	1
	On	Off	2
	Off	On	4
	On	On	4

Note to Table 2–38:

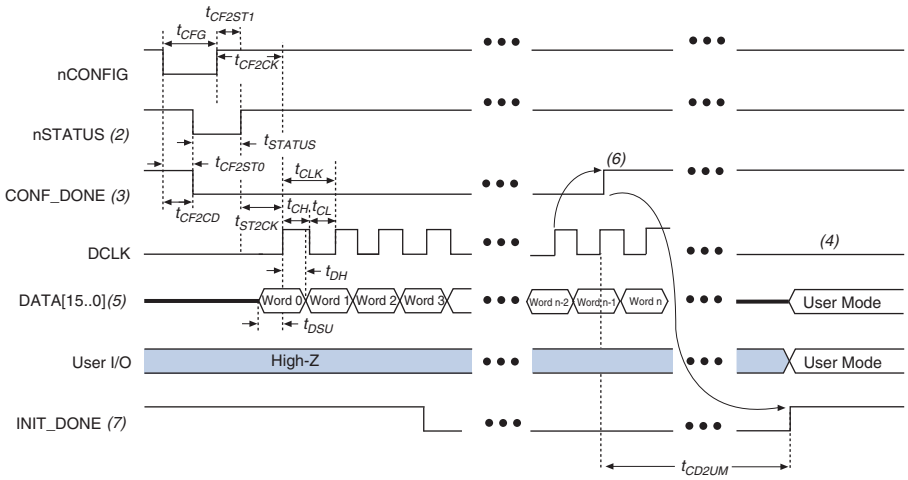
(1) Depending on the DCLK-to-DATA[] ratio, the host must send a DCLK frequency that is r times the DATA[] rate in byte per second (Bps) or word per second (Wps). For example, in FPP x16 where the r is 2, the DCLK frequency must be 2 times the DATA[] rate in Wps.

FPP Configuration Timing when DCLK to DATA[] = 1

Figure 2–5 shows the timing waveform for a FPP configuration when using a MAX® II device as an external host. This timing waveform shows timing when the DCLK-to-DATA[] ratio is 1.

When you enable decompression or the design security feature, the DCLK-to-DATA[] ratio varies for FPP x8 and FPP x16. For the respective DCLK-to-DATA[] ratio, refer to Table 2–38.

Figure 2–5. DCLK-to-DATA[] FPP Configuration Timing Waveform When the Ratio is 1 *(1)*



Notes to Figure 2–5:

(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 V device holds nSTATUS low for the time of the POR delay.

(3) After power up, before and during configuration, CONF_DONE is low.

(4) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(5) For FPP x16, use DATA[15..0]. For FPP x8, use DATA[7..0]. DATA[15..0] are available as a user I/O pin after configuration. The state of this pin depends on the dual-purpose pin settings.

(6) To ensure a successful configuration, send the entire configuration data to the Arria V device. CONF_DONE is released high when the Arria V device receives all the configuration data successfully. After CONF_DONE goes high, send two additional falling edges on DCLK to begin initialization and enter user mode.

(7) After the option bit to enable the INIT_DONE pin is configured into the device, the INIT_DONE goes low.

Table 2–39 lists the timing parameters for Arria V devices for FPP configuration when the DCLK-to-DATA[] ratio is 1.

Table 2–39. DCLK-to-DATA[] FPP Timing Parameters for Arria V Devices When the Ratio is 1—*Preliminary* (1)

Symbol	Parameter	Minimum	Maximum	Unit
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	600	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	600	ns
t_{CFG}	nCONFIG low pulse width	2	—	µs
t_{STATUS}	nSTATUS low pulse width	268	1506 (2)	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	1506 (3)	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	1506	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	DATA[] setup time before rising edge on DCLK	5.5	—	ns
t_{DH}	DATA[] hold time after rising edge on DCLK	0	—	ns
t_{CH}	DCLK high time	0.45 x $1/f_{MAX}$	—	ns
t_{CL}	DCLK low time	0.45 x $1/f_{MAX}$	—	ns
t_{CLK}	DCLK period	$1/f_{MAX}$	—	ns
f_{MAX}	DCLK frequency (FPP x8/ x16)	—	125	MHz
t_R	Input rise time	—	40	ns
t_F	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode (4)	175	437	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} + (T_{init} x CLKUSR period)	—	—
T_{init}	Number of clock cycles required for device initialization	17,408	—	Cycles

Notes to Table 2–39:

(1) Use these timing parameters when the DCLK-to-DATA[] ratio is 1. To find the DCLK-to-DATA[] ratio for your system, refer Table 2–38 on page 2–38.

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

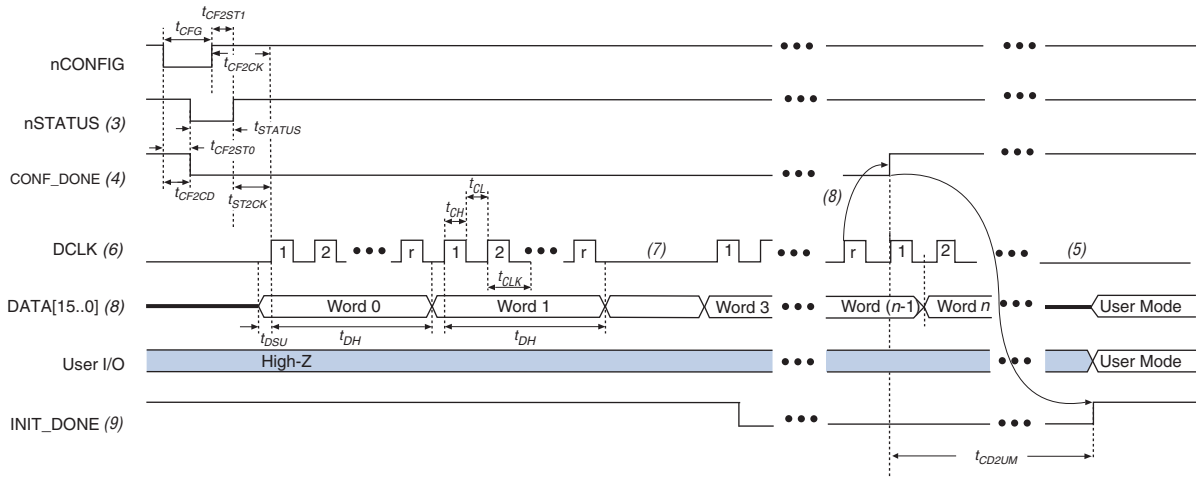
(3) You can obtain this value if you do not delay configuration by externally holding the nSTATUS low.

(4) The minimum and maximum numbers apply only if you chose the internal oscillator as the clock source for initializing the device.

FPP Configuration Timing when DCLK to DATA[] > 1

Figure 2–6 shows the timing waveform for a FPP configuration when using a MAX II device or microprocessor as an external host. This waveform shows timing when the DCLK-to-DATA[] ratio is more than 1.

Figure 2–6. FPP Configuration Timing Waveform When the DCLK-to-DATA[] Ratio is >1 *(1), (2)*



Notes to Figure 2–6:

(1) To find the DCLK-to-DATA[] ratio for your system, refer Table 2–38 on page 2–38.

(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 V device holds nSTATUS low for the time as specified by the POR delay.

(4) After power up, before and during configuration, CONF_DONE is low.

(5) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(6) "r" denotes the DCLK-to-DATA[] ratio. For the DCLK-to-DATA[] ratio based on the decompression and the design security feature enable settings, refer to Table 2–38 on page 2–38.

(7) If needed, pause DCLK by holding it low. When DCLK restarts, the external host must provide data on the DATA[15..0] pins prior to sending the first DCLK rising edge.

(8) To ensure a successful configuration, send the entire configuration data to the Arria V device. CONF_DONE is released high after the Arria V device receives all the configuration data successfully. After CONF_DONE goes high, send two additional falling edges on DCLK to begin initialization and enter user mode.

(9) After the option bit to enable the INIT_DONE pin is configured into the device, the INIT_DONE goes low.

Table 2–40 lists the timing parameters for Arria V devices when the DCLK-to-DATA[] ratio is more than 1.

Table 2–40. DCLK-to-DATA[] FPP Timing Parameters for Arria V Devices When the Ratio is >1—*Preliminary*(1)

Symbol	Parameter	Minimum	Maximum	Unit
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	600	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	600	ns
t_{CFG}	nCONFIG low pulse width	2	—	µs
t_{STATUS}	nSTATUS low pulse width	268	1506 (2)	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	1506 (3)	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	1506	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	DATA[] setup time before rising edge on DCLK	5.5	—	ns
t_{DH}	DATA[] hold time after rising edge on DCLK	3 x $1/f_{DCLK}$	—	ns
t_{CH}	DCLK high time	0.45 x $1/f_{MAX}$	—	ns
t_{CL}	DCLK low time	0.45 x $1/f_{MAX}$	—	ns
t_{CLK}	DCLK period	$1/f_{MAX}$	—	ns
f_{MAX}	DCLK frequency (FPP x8/ x16)	—	125	MHz
t_R	Input rise time	—	40	ns
t_F	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode (4)	175	437	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} + (T_{init} x CLKUSR period)	—	—
T_{init}	Number of clock cycles required for device initialization	17,408	—	Cycles

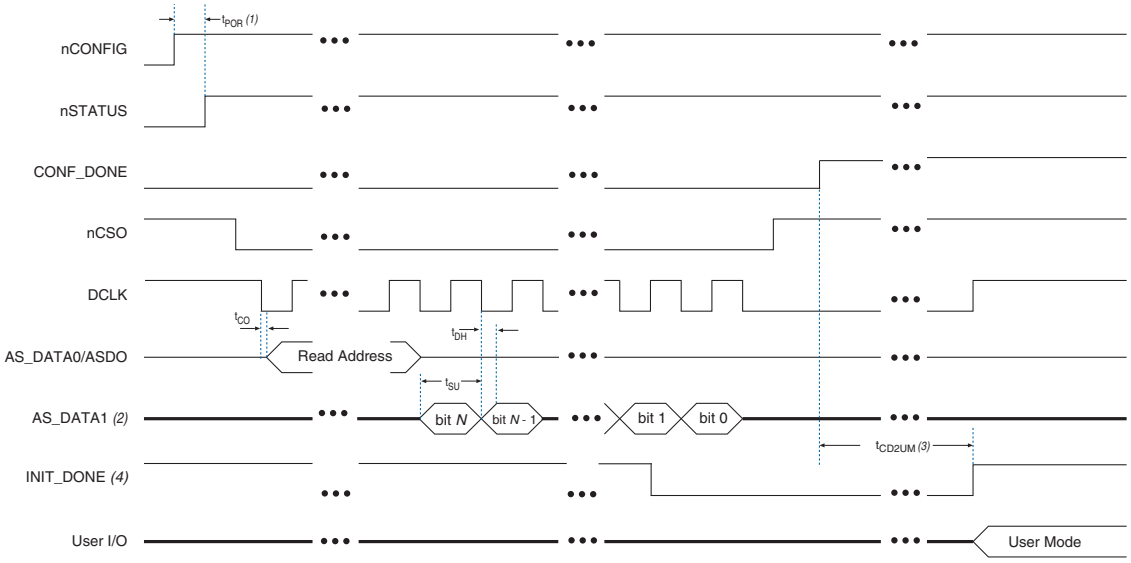
Notes to Table 2–40:

(1) Use these timing parameters when you use decompression and the design security features.
(2) This value can be obtained if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(3) This value can be obtained if you do not delay configuration by externally holding nSTATUS low.
(4) The minimum and maximum numbers apply only if you chose the internal oscillator as the clock source for initializing the device.

AS Configuration Timing

Figure 2–7 shows the timing waveform for the active serial (AS) x1 mode and AS x4 mode configuration timing.

Figure 2–7. AS Configuration Timing



Notes to Figure 2–7:

(1) The AS scheme supports standard and fast POR delay (t_{POR}). For t_{POR} delay information, refer to the "POR Delay Specification" section in the *Configuration, Design Security, and remote System Upgrades in Arria V Devices* chapter.

(2) If you are using AS x4 mode, this signal represents the `AS_DATA[3..0]` and EPCQ sends in 4-bits of data for each `DCLK` cycle.

(3) The initialization clock can be from the internal oscillator or `CLKUSR` pin.

(4) After the option bit to enable the `INIT_DONE` pin is configured into the device, the `INIT_DONE` goes low.

Table 2–41 lists the timing parameters for AS x1 and AS x4 configurations in Arria V devices.

Table 2–41. AS Timing Parameters for AS x1 and x4 Configurations in Arria V Devices—*Preliminary* (1), (2)

Symbol	Parameter	Minimum	Maximum	Unit
t_{CO}	DCLK falling edge to the `AS_DATA0/ASDO` output	—	4	µs
t_{SU}	Data setup time before the rising edge on `DCLK`	1.5	—	ns
t_H	Data hold time after the rising edge on `DCLK`	0	—	ns
t_{CD2UM}	`CONF_DONE` high to user mode	175	437	µs
t_{CD2CU}	`CONF_DONE` high to `CLKUSR` enabled	4 x maximum `DCLK` period	—	—
t_{CD2UMC}	`CONF_DONE` high to user mode with `CLKUSR` option on	t_{CD2CU} + (T_{init} x `CLKUSR` period)	—	—
T_{init}	Number of clock cycles required for device initialization	17,408	—	Cycles

Notes to Table 2–41:

(1) The minimum and maximum numbers apply only if you choose the internal oscillator as the clock source for initializing the device.

(2) The t_{CF2CD}, t_{CF2ST0}, t_{CFG}, t_{STATUS}, and t_{CF2ST1} timing parameters are identical to the timing parameters for PS mode listed in Table 2–43 on page 2–45.

Table 2–42 lists the internal clock frequency specification for the AS configuration scheme.

Table 2–42. DCLK Frequency Specification in the AS Configuration Scheme—*Preliminary* (1), (2)

Minimum	Typical	Maximum	Unit
5.3	7.9	12.5	MHz
10.6	15.7	25.0	MHz
21.3	31.4	50.0	MHz
42.6	62.9	100.0	MHz

Notes to Table 2–42:

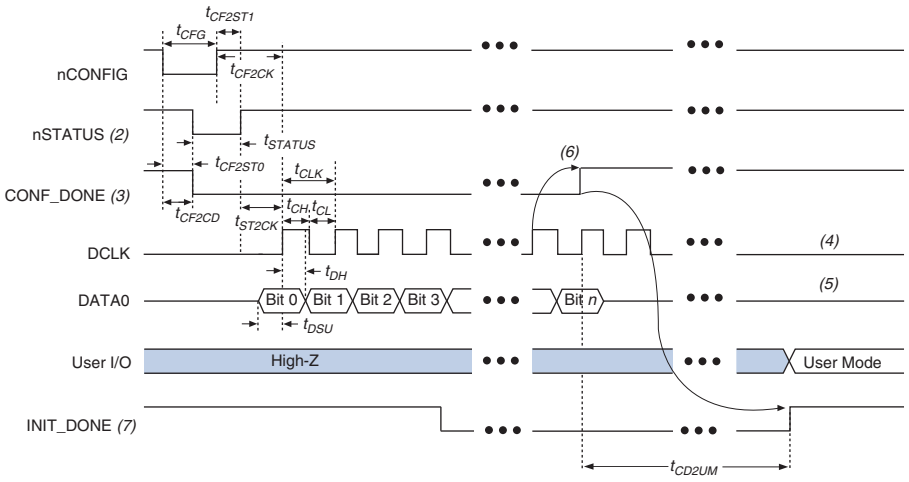
(1) This applies to the DCLK frequency specification when using the internal oscillator as the configuration clock source.

(2) The AS multi-device configuration scheme does not support DCLK frequency of 100 MHz.

PS Configuration Timing

Figure 2–8 shows the timing waveform for a passive serial (PS) configuration when using a MAX II device or microprocessor as an external host.

Figure 2–8. PS Configuration Timing Waveform (1)



Notes to Figure 2–8:

(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 V device holds nSTATUS low for the time of the POR delay.

(3) After power up, before and during configuration, CONF_DONE is low.

(4) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(5) DATA0 is available as a user I/O pin after configuration. The state of this pin depends on the dual-purpose pin settings in the **Device and Pins Option**.

(6) To ensure a successful configuration, send the entire configuration data to the Arria V device. CONF_DONE is released high after the Arria V device receives all the configuration data successfully. After CONF_DONE goes high, send two additional falling edges on DCLK to begin initialization and enter user mode.

(7) After the option bit to enable the INIT_DONE pin is configured into the device, the INIT_DONE goes low.

Table 2–43 lists the PS timing parameter for Arria V devices.

Table 2–43. PS Timing Parameters for Arria V Devices—*Preliminary*

Symbol	Parameter	Minimum	Maximum	Unit
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	600	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	600	ns
t_{CFG}	nCONFIG low pulse width	2	—	µs
t_{STATUS}	nSTATUS low pulse width	268	1506 (1)	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	1506 (2)	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	1506	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	DATA[] setup time before rising edge on DCLK	5.5	—	ns
t_{DH}	DATA[] hold time after rising edge on DCLK	0	—	ns
t_{CH}	DCLK high time	0.45 x $1/f_{MAX}$	—	ns
t_{CL}	DCLK low time	0.45 x $1/f_{MAX}$	—	ns
t_{CLK}	DCLK period	$1/f_{MAX}$	—	ns
f_{MAX}	DCLK frequency	—	125	MHz
t_R	Input rise time	—	40	ns
t_F	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode (3)	175	437	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 x maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (T_{init} x CLKUSR period)	—	—
T_{init}	Number of clock cycles required for device initialization	17,408	—	Cycles

Notes to Table 2–43:

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

(2) You can obtain this value if you do not delay configuration by externally holding nSTATUS low.

(3) The minimum and maximum numbers apply only if you chose the internal oscillator as the clock source for initializing the device.

Remote System Upgrades Circuitry Timing Specification

Table 2–44 lists the timing parameter specifications for the remote system upgrade circuitry.

Table 2–44. Remote System Upgrade Circuitry Timing Specification

Parameter	Minimum	Maximum	Unit
$t_{MAX\_RU\_CLK}$ (1)	—	40	MHz
$t_{RU\_nCONFIG}$ (2)	250	—	ns
$t_{RU\_nRSTIMER}$ (3)	250	—	ns

Notes to Table 2–44:

(1) This clock is user-supplied to the remote system upgrade circuitry. If you are using the ALTREMOTE_UPDATE megafunction, the clock user-supplied to the ALTREMOTE_UPDATE megafunction must meet this specification.

(2) This is equivalent to strobing the reconfiguration input of the ALTREMOTE_UPDATE megafunction high for the minimum timing specification. For more information, refer to the "Remote System Upgrade State Machine" section in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

(3) This is equivalent to strobing the reset timer input of the ALTREMOTE_UPDATE megafunction high for the minimum timing specification. For more information, refer to the "User Watchdog Timer" section in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

User Watchdog Internal Oscillator Frequency Specification

Table 2–45 lists the frequency specifications for the user watchdog internal oscillator.

Table 2–45. User Watchdog Internal Oscillator Frequency Specifications—*Preliminary*

Minimum	Typical	Maximum	Unit
5.3	7.9	12.5	MHz

I/O Timing

Altera offers two ways to determine I/O timing—the Excel-based I/O Timing and the Quartus II Timing Analyzer.

Excel-based I/O timing provides pin timing performance for each device density and speed grade. The data is typically used prior to designing the FPGA to get an estimate of the timing budget as part of the link timing analysis. The Quartus II Timing Analyzer provides a more accurate and precise I/O timing data based on the specifics of the design after you complete place-and-route.

You can download the Excel-based I/O Timing spreadsheet from the Arria V Devices Literature webpage.

Programmable IOE Delay

Table 2–46 lists the Arria V IOE programmable delay settings.

Table 2–46. IOE Programmable Delay for Arria V Devices *(1)*

Parameter	Available Settings	Minimum Offset	Fast Model		Slow Model			Unit
			Industrial	Commercial	C4	C5, I5	C6	
TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	ns
TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	ns
TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	ns
TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	ns
TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	ns

Note to Table 2–46:

(1) Pending the Quartus II software extraction.

Programmable Output Buffer Delay

Table 2–47 lists the delay chain settings that control the rising and falling edge delays of the output buffer. The default delay is 0 ps.

Table 2–47. Programmable Output Buffer Delay—*Preliminary* *(1), (2)*

Symbol	Parameter	Typical	Unit
D_{OUTBUF}	Rising and/or falling edge delay	0 (default)	ps
		50	ps
		100	ps
		150	ps

Notes to Table 2–47:

(1) Pending the Quartus II software extraction.

(2) You can set the programmable output buffer delay in the Quartus II software by setting the **Output Buffer Delay Control** assignment to either positive, negative, or both edges, with the specific values stated here (in ps) for the **Output Buffer Delay** assignment.

Glossary

Table 2–48 lists the glossary for this chapter.

Table 2–48. Glossary Table (Part 1 of 4)

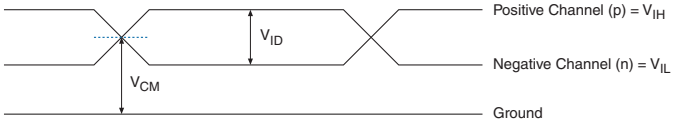
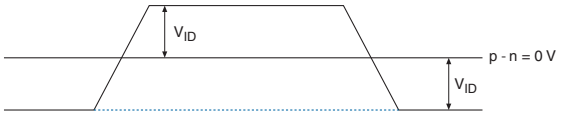
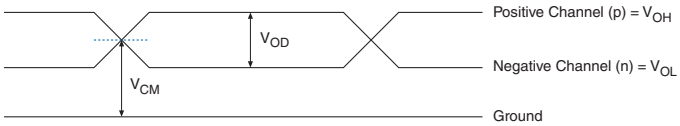
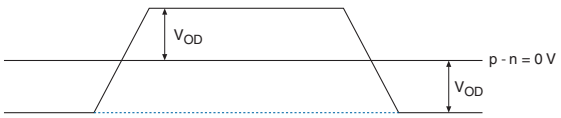
Letter	Subject	Definitions
A B C	—	—
D	Differential I/O Standards	*Receiver Input Waveforms* **Single-Ended Waveform**  **Differential Waveform**  *Transmitter Output Waveforms* **Single-Ended Waveform**  **Differential Waveform** 
E	—	—
F	f_{HSCLK}	Left/right PLL input clock frequency.
	f_{HSDR}	High-speed I/O block—Maximum/minimum LVDS data transfer rate (f_{HSDR} = 1/TUI), non-DPA.
	$f_{HSDRDPA}$	High-speed I/O block—Maximum/minimum LVDS data transfer rate ($f_{HSDRDPA}$ = 1/TUI), DPA.
G H I	—	—

Table 2–48. Glossary Table (Part 2 of 4)

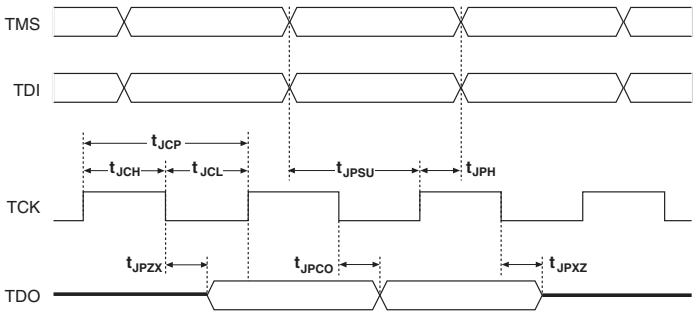
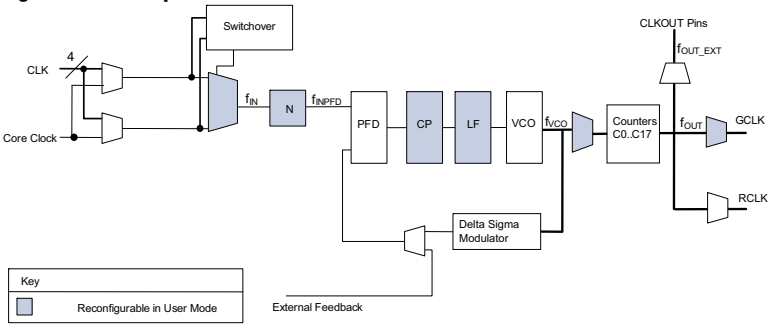
Letter	Subject	Definitions
J	J	High-speed I/O block—Deserialization factor (width of parallel data bus).
	JTAG Timing Specifications	JTAG Timing Specifications: 
K **L** **M** **N** **O**	—	—
P	**PLL Specifications**	**Diagram of PLL Specifications** *(1)*  **Note:** (1) `Core Clock` can only be fed by dedicated clock input pins or PLL outputs.
Q	—	—
R	R_L	Receiver differential input discrete resistor (external to the Arria V device).

Table 2–48. Glossary Table (Part 3 of 4)

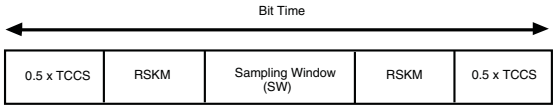
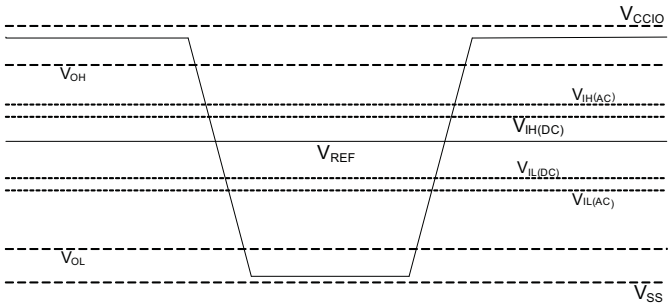
Letter	Subject	Definitions
S	**Sampling window (SW)**	Timing Diagram—the period of time during which the data must be valid in order to capture it correctly. The setup and hold times determine the ideal strobe position in the sampling window, as shown: 
	Single-ended voltage referenced I/O standard	The JEDEC standard for the SSTI and HSTL I/O defines both the AC and DC input signal values. The AC values indicate the voltage levels at which the receiver must meet its timing specifications. The DC values indicate the voltage levels at which the final logic state of the receiver is unambiguously defined. After the receiver input has crossed the AC value, the receiver changes to the new logic state. The new logic state is then maintained as long as the input stays beyond the AC threshold. This approach is intended to provide predictable receiver timing in the presence of input waveform ringing, as shown: *Single-Ended Voltage Referenced I/O Standard* 
T	t_C	High-speed receiver/transmitter input and output clock period.
	TCCS (channel-to-channel-skew)	The timing difference between the fastest and slowest output edges, including the t_{CO} variation and clock skew, across channels driven by the same PLL. The clock is included in the TCCS measurement (refer to the *Timing Diagram* figure under **SW** in this table).
	t_{DUTY}	High-speed I/O block—Duty cycle on high-speed transmitter output clock.
		Timing Unit Interval (TUI) The timing budget allowed for skew, propagation delays, and the data sampling window. (TUI = 1/(Receiver Input Clock Frequency Multiplication Factor) = t_C/w)
	t_{FALL}	Signal high-to-low transition time (80–20%)
	t_{INCCJ}	Cycle-to-cycle jitter tolerance on the PLL clock input
	t_{OUTPJ_IO}	Period jitter on the GPIO driven by a PLL
	t_{OUTPJ_DC}	Period jitter on the dedicated clock output driven by a PLL
	t_{RISE}	Signal low-to-high transition time (20–80%)
U	—	—

Table 2–48. Glossary Table (Part 4 of 4)

Letter	Subject	Definitions
V	$V_{CM(DC)}$	DC Common mode input voltage.
	V_{ICM}	Input Common mode voltage—The common mode of the differential signal at the receiver.
	V_{ID}	Input differential voltage swing—The difference in voltage between the positive and complementary conductors of a differential transmission at the receiver.
	$V_{DIF(AC)}$	AC differential input voltage—Minimum AC input differential voltage required for switching.
	$V_{DIF(DC)}$	DC differential input voltage— Minimum DC input differential voltage required for switching.
	V_{IH}	Voltage input high—The minimum positive voltage applied to the input which is accepted by the device as a logic high.
	$V_{IH(AC)}$	High-level AC input voltage
	$V_{IH(DC)}$	High-level DC input voltage
	V_{IL}	Voltage input low—The maximum positive voltage applied to the input which is accepted by the device as a logic low.
	$V_{IL(AC)}$	Low-level AC input voltage
	$V_{IL(DC)}$	Low-level DC input voltage
	V_{OCM}	Output Common mode voltage—The common mode of the differential signal at the transmitter.
	V_{OD}	Output differential voltage swing—The difference in voltage between the positive and complementary conductors of a differential transmission at the transmitter.
	V_{SWING}	Differential input voltage
	V_X	Input differential cross point voltage
	V_{OX}	Output differential cross point voltage
W	W	High-speed I/O block—Clock Boost Factor
X, Y, Z	—	—

Document Revision History

Table 2–49 lists the revision history for this chapter.

Table 2–49. Document Revision History

Date	Version	Changes
February 2012	1.3	■ Updated Table 2–1. ■ Updated Transceiver-FPGA Fabric Interface rows in Table 2–20. ■ Updated V_{CCP} description.
December 2011	1.2	Updated Table 2–1, and Table 2–3.
November 2011	1.1	■ Updated Table 2–1, Table 2–19, Table 2–26, and Table 2–36. ■ Added Table 2–5. ■ Added Figure 2–4.
August 2011	1.0	Initial release.

This chapter provides additional information about the document and Altera.

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
Nontechnical support (general)	Email	nacomp@altera.com
(software licensing)	Email	authorization@altera.com

Note to Table:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, **\qdesigns** directory, **D:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicate document titles. For example, *Stratix IV Design Guidelines*.
italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>***.pof** file.
Initial Capital Letters	Indicate keyboard keys and menu names. For example, the Delete key and the Options menu.
"Subheading Title"	Quotation marks indicate references to sections in a document and titles of Quartus II Help topics. For example, "Typographic Conventions."

Visual Cue	Meaning
Courier type	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. The suffix `n` denotes an active-low signal. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
↵	An angled arrow instructs you to press the Enter key.
1., 2., 3., and a., b., c., and so on	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■ ■	Bullets indicate a list of items when the sequence of the items is not important.
	The hand points to information that requires special attention.
	A question mark directs you to a software help system with related information.
	The feet direct you to another document or website with related information.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The envelope links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.

Arria V Device Handbook

Volume 2: Device Interfaces and Integration

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

AV-5V2-1.3

Contents

Chapter Revision Dates

The chapters in this document, Volume 2: 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. Logic Array Blocks and Adaptive Logic Modules in Arria V Devices
Revised: *November 2011*
Part Number: *AV-52001-1.1*

Chapter 2. Memory Blocks in Arria V Devices
Revised: *November 2011*
Part Number: *AV-52003-1.1*

Chapter 3. Variable-Precision DSP Blocks in Arria V Devices
Revised: *November 2011*
Part Number: *AV-52003-1.1*

Chapter 4. Clock Networks and PLLs in Arria V Devices
Revised: *November 2011*
Part Number: *AV-52004-1.0*

Chapter 5. I/O Features in Arria V Devices
Revised: *February 2012*
Part Number: *AV52005-1.2*

Chapter 6. High-Speed Differential I/O Interfaces and DPA in Arria V Devices
Revised: *November 2011*
Part Number: *AV52006-1.1*

Chapter 7. External Memory Interfaces in Arria V Devices
Revised: *November 2011*
Part Number: *AV52007-1.1*

Chapter 8. Configuration, Design Security, and Remote System Upgrades in Arria V Devices
Revised: *November 2011*
Part Number: *AV-52008-1.1*

Chapter 9. SEU Mitigation in Arria V Devices
Revised: *November 2011*
Part Number: *AV-52009-1.1*

Chapter 10. JTAG Boundary-Scan Testing in Arria V Devices
Revised: *February 2012*
Part Number: *AV-52010-1.2*

Chapter 11. Power Management in Arria V Devices
Revised: *February 2012*
Part Number: *AV-52011-1.3*

Section I. Device Core for Arria V Devices

This section provides a complete overview of all features relating to the Arria® V device family. This section includes the following chapters:

- Chapter 1, Logic Array Blocks and Adaptive Logic Modules in Arria V Devices
- Chapter 2, Memory Blocks in Arria V Devices
- Chapter 3, Variable-Precision DSP Blocks in Arria V Devices
- Chapter 4, Clock Networks and PLLs in Arria V Devices

Revision History

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

1. Logic Array Blocks and Adaptive Logic Modules in Arria V Devices

AV-52001-1.1

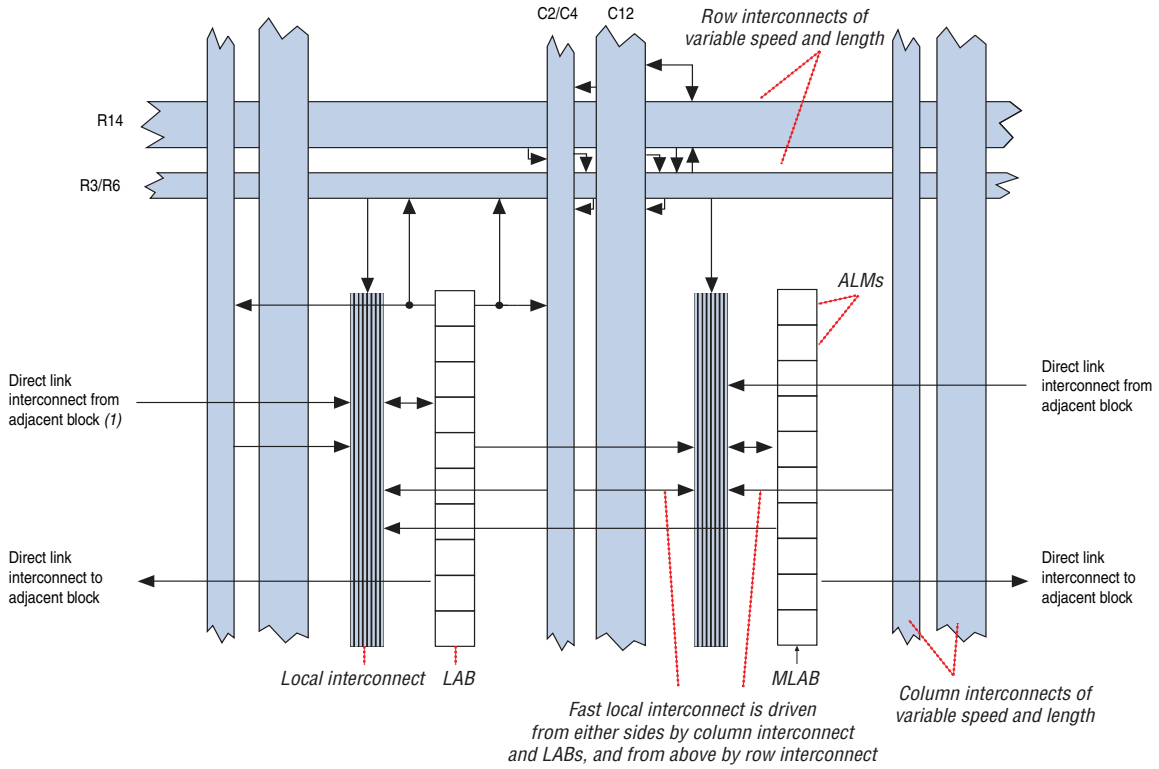
This chapter describes the features of the logic array block (LAB) in the Arria® V core fabric. The LAB is composed of basic building blocks known as adaptive logic modules (ALMs) that you can configure to implement logic functions, arithmetic functions, and register functions.

This chapter contains the following sections:

Logic Array Blocks

Figure 1–1 shows the Arria V LAB structure and the LAB interconnects.

Figure 1–1. LAB Structure and Interconnects in Arria V Devices



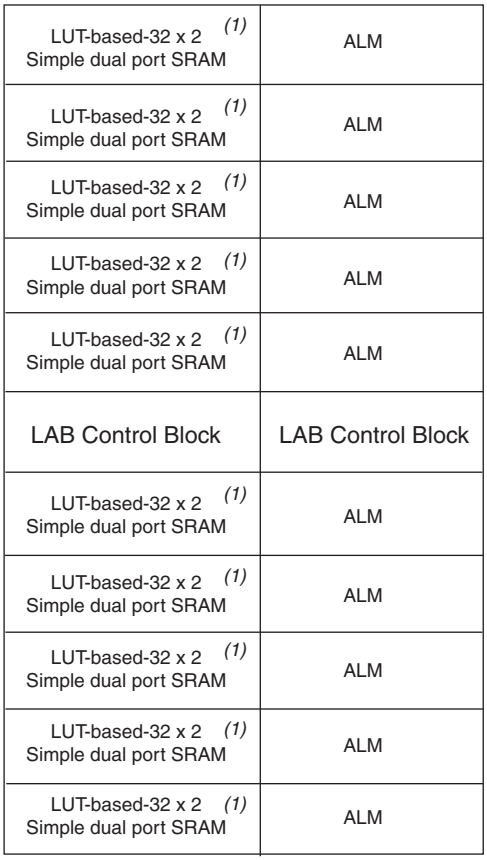
Note to Figure 1–1:

(1) Connects to adjacent LABs, memory blocks, digital signal processing (DSP) blocks, or I/O element (IOE) outputs.

You can configure each ALM in an MLAB as a 32 x 2 memory block, resulting in a configuration of 32 x 20 simple dual-port SRAM blocks. MLAB is a superset of the LAB and includes all the LAB features. Figure 1–2 shows an overview of the LAB and MLAB topology.

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

Figure 1–2. LAB and MLAB Structure for Arria V Devices



Note to Figure 1–2:

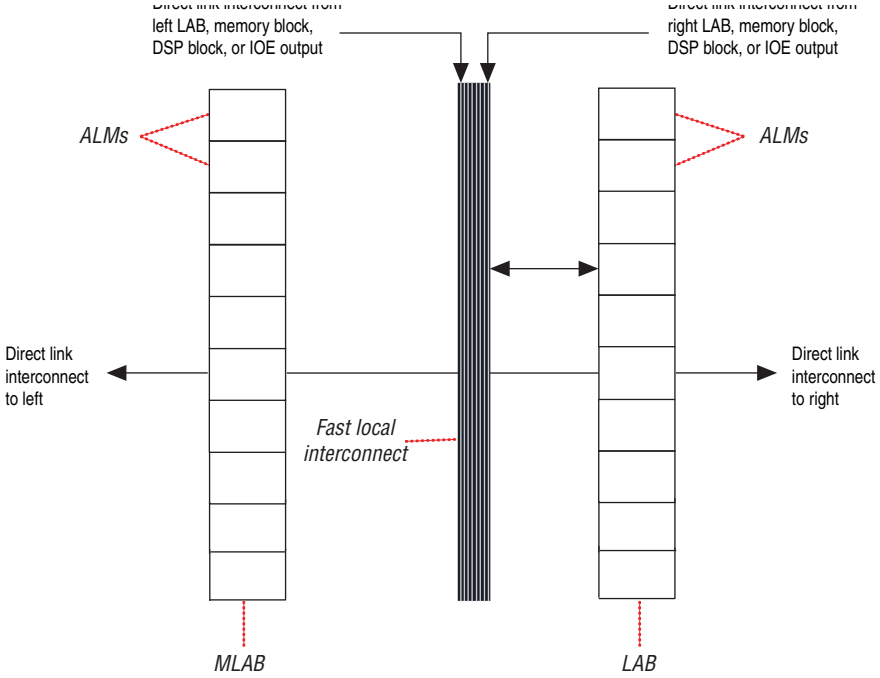
(1) You can use an MLAB ALM as a regular LAB ALM or configure it as a dual-port SRAM.

LAB Interconnects

The LAB local interconnect drives the ALMs in the same LAB. 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 1–3 shows the direct link connection, which connects adjacent LABs, memory blocks, DSP blocks, or IOE outputs.

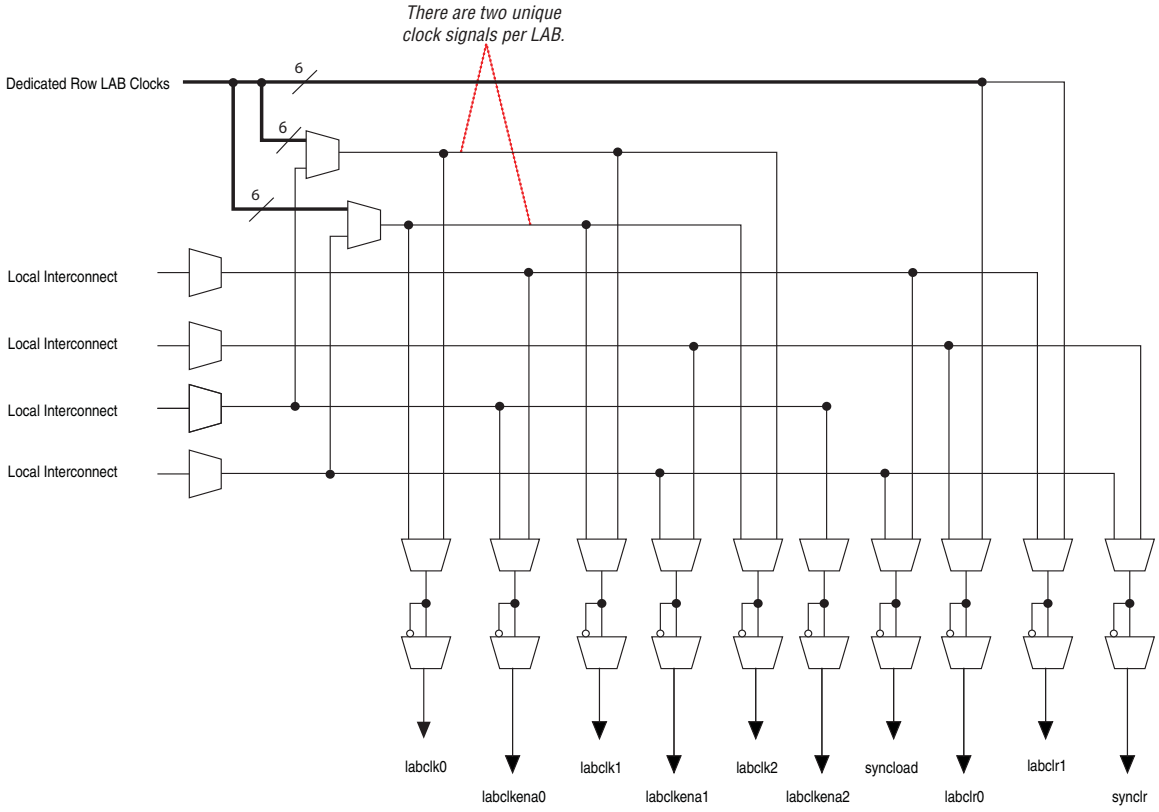
Figure 1–3. Direct Link Connection for Arria V Devices



LAB Control Signals

Each LAB contains dedicated logic for driving the control signals to its ALMs, and has two unique clock sources and three clock enable signals, as shown in Figure 1–4. The LAB control block generates up to three clocks using the two clock sources and three clock enable signals. Each clock and the clock enable signals are linked. De-asserting the clock enable signal turns off the corresponding LAB-wide clock.

Figure 1–4. LAB-Wide Control Signals for Arria V Devices *(1)*



Note to Figure 1–4:

(1) For more information, refer to Figure 1–5 on page 1–6.

For more information about LABs, refer to "Logic Array Blocks" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

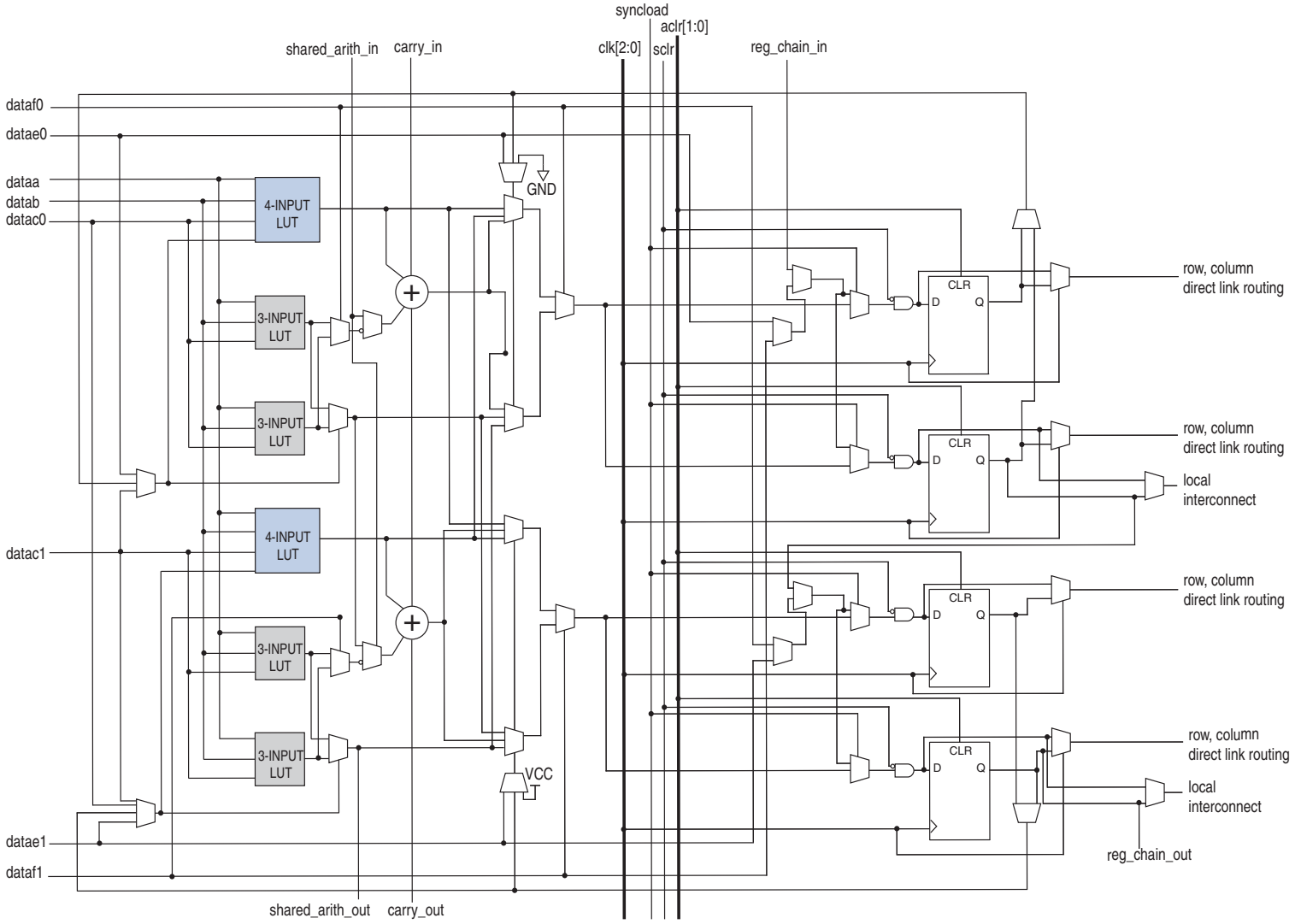
Adaptive Logic Modules

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

Each ALM has two sets of outputs that drive the local, row, and column routing resources. The LUT, adder, or register output can drive the ALM outputs (refer to Figure 1–5). 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.

Figure 1–5 shows a detailed view of all the connections in an ALM.

Figure 1–5. ALM Connection Details for Arria V Devices



ALM Operating Modes

The Arria V ALM operates in any of the following modes:

- "Normal Mode" on page 1–7
- "Extended LUT Mode" on page 1–7
- "Arithmetic Mode" on page 1–8
- "Shared Arithmetic Mode" on page 1–9

For more information about ALMs, refer to "Adaptive Logic Modules" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

Normal Mode

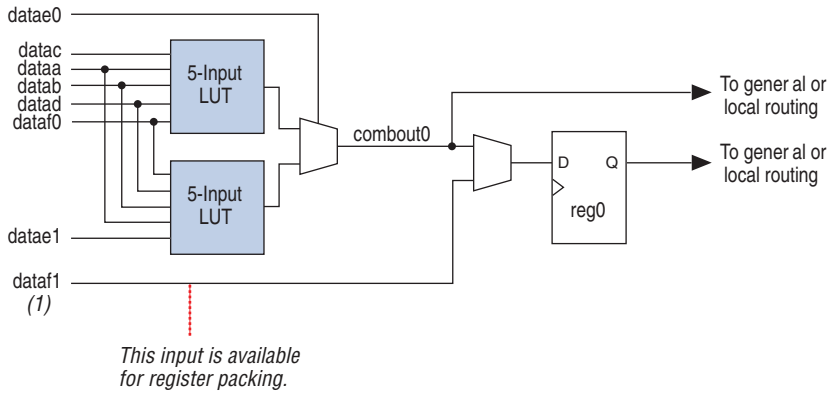
In this mode, up to eight data inputs from the LAB local interconnect are inputs to the combinational logic. Normal mode allows two functions to be implemented in one Arria V ALM, or a single function of up to six inputs. The ALM can support certain combinations of completely independent functions and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

Extended LUT Mode

Figure 1–6 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 1–6, often appear in designs as "if-else" statements in Verilog HDL or VHDL code.

Figure 1–6. Template for Supported 7-Input Functions in Extended LUT Mode in Arria V Devices



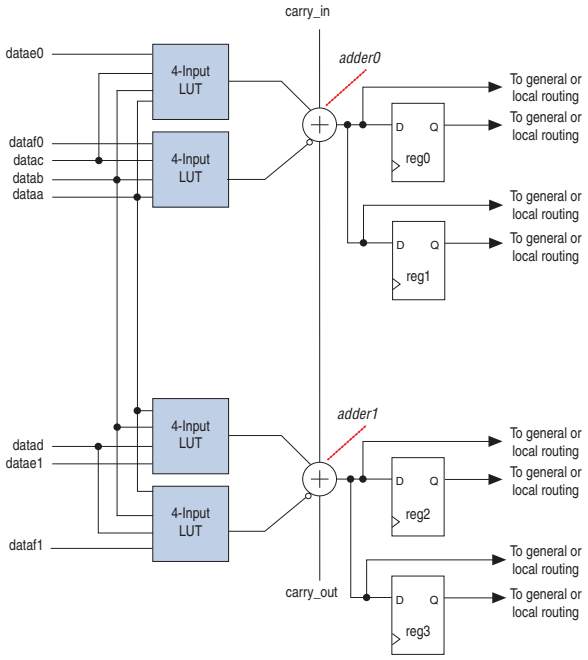
Note to Figure 1–6:

(1) If the 7-input function is unregistered, the unused eighth input is available for register packing.

Arithmetic Mode

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. Figure 1–7 shows an ALM in arithmetic mode.

Figure 1–7. ALM in Arithmetic Mode for Arria V Devices



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 the combinational logic output provides resource savings of up to 50% for functions that can use this mode.

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 V 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.

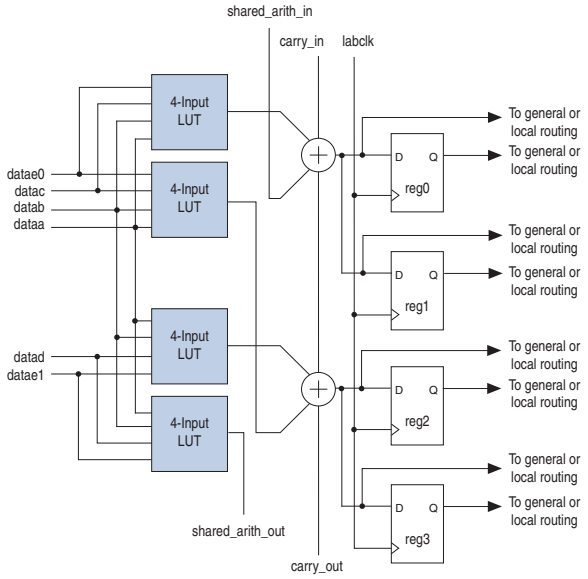
The Quartus II Compiler creates carry chains longer than 20 ALMs (10 ALMs in arithmetic or shared arithmetic mode) by linking LABs together automatically. For enhanced fitting, a long carry chain runs vertically, allowing fast horizontal connections to the TriMatrix memory and DSP blocks. A carry chain 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. You can bypass the top-half of the LAB columns and bottom-half of the MLAB columns.

Shared Arithmetic Mode

In shared arithmetic mode, the ALM can implement a 3-input add in the ALM. In this mode, configure the ALM 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 1–8 shows the ALM using this feature.

Figure 1–8. ALM in Shared Arithmetic Mode for Arria V Devices



Shared Arithmetic Chain

The shared arithmetic chain available in enhanced arithmetic mode allows the ALM to implement a 3-input adder. This significantly reduces the resources necessary to implement large adder trees or correlator functions.

The shared arithmetic chain 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 enhanced 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 carry chains, the top and bottom half of the 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. In every LAB column is top-half bypassable; while in MLAB columns are bottom-half bypassable.

Document Revision History

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

Table 1–1. Document Revision History

Date	Version	Changes
November 2011	1.1	Restructured chapter.
May 2011	1.0	Initial release.

2. Memory Blocks in Arria V Devices

AV-52003-1.1

This chapter describes the embedded memory blocks in Arria® V devices. The embedded memory blocks provide different sizes of embedded SRAM to address the Arria V device design requirements efficiently.

This chapter contains the following sections:

Overview

Arria V devices contain two types of memory blocks:

- Memory logic array blocks (MLABs)—640-bit enhanced memory blocks that are optimized for implementation of shift registers for digital signal processing (DSP) applications, wide shallow FIFO buffers, and filter delay lines.
- M10K blocks—10-kilobit (Kb) memory blocks that you can use to create designs with larger memory configurations.

Table 2–1 lists the features supported by the embedded memory blocks.

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

Feature	MLABs	M10K Blocks
Maximum performance	300 MHz	380 MHz
Total RAM bits (including parity bits)	640	10,240
Configurations (depth × width)	32 x 16 32 x 18 32 x 20	8K x 1 4K x 2 2K x 4 2K x 5 1K x 8 1K x 10 512 x 16 512 x 20 256 x 32 256 x 40
Parity bits (1)	✓	✓

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

Feature	MLABs	M10K Blocks
Byte enable (1)	✓	✓
Packed mode	—	✓
Address clock enable (2)	✓	✓
Single-port memory (2)	✓	✓
Simple dual-port memory (2)	✓	✓
True dual-port memory (2)	—	✓
Embedded shift register (3)	✓	✓
ROM (2)	✓	✓
FIFO buffer	✓ (4)	✓
Simple dual-port mixed width support (2)	—	✓
True dual-port mixed width support (2)	—	✓
Memory Initialization File (**.mif**)	✓	✓
Mixed-clock mode	✓	✓
Power-up condition	Outputs cleared if registered, otherwise reads memory contents	Outputs cleared
Register clears	Output registers	Output registers
Write/Read operation triggering	Write and Read: Rising clock edges	Write and Read: Rising clock edges
Same-port read-during-write	Outputs set to **don't care**	Outputs set to **new data** or **don't care** (5)
Mixed-port read-during-write	Outputs set to **old data**, **new data**, or **don't care** (6)	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

Notes to Table 2–1:

(1) For more information about parity bit and byte enable support, refer to "Parity Bit Support" and "Byte Enable Support" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

(2) For more information about these memory features, refer to *Internal Memory (RAM and ROM) User Guide*.

(3) For more information about implementing the shift register mode, refer to *RAM-Based Shift Register (ALTSHIFT_TAPS) Megafunction User Guide*.

(4) MLABs do not support mixed-width FIFO mode. For more information about implementing FIFO buffers, refer to *SCFIFO and DCFIFO Megafunctions User Guide*.

(5) Don't care mode is only available in single-port RAM.

(6) MLAB outputs are only set to **don't care** when the output register is enabled.

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

Table 2–2. Memory Capacity and Distribution in Arria V Devices (Part 1 of 2)

Device	MLABs	M10K Blocks	Total Dedicated RAM Bits (M10K Blocks Only) (Kb)	Total RAM Bits (Including LABs) (Kb)
5AGXA1	741	800	8,000	8,463
5AGXA3	1,398	1,051	10,510	11,383
5AGXA5	1,877	1,180	11,800	12,973
5AGXA7	2,318	1,366	13,660	15,108

Table 2–2. Memory Capacity and Distribution in Arria V Devices (Part 2 of 2)

Device	MLABs	M10K Blocks	Total Dedicated RAM Bits (M10K Blocks Only) (Kb)	Total RAM Bits (Including LABs) (Kb)
5AGXB1	2,963	1,510	15,100	16,952
5AGXB3	3,358	1,726	17,260	19,358
5AGXB5	4,051	2,054	20,540	23,072
5AGXB7	4,650	2,414	24,140	27,046
5AGTD3	3,358	1,726	17,260	19,358
5AGTD7	4,650	2,414	24,140	27,046

Memory Modes

The M10K blocks do not support asynchronous memory (unregistered inputs) but the MLABs support asynchronous (flow-through) read operations. Depending on which memory blocks you target, the embedded memory blocks in the Arria V devices allow you to implement fully synchronous SRAM memory in these modes of operation:

- Single-port RAM
- Simple dual-port mode
- True dual-port mode (only supported in M10K blocks)
- Shift-register mode
- ROM mode
- FIFO mode

To avoid corrupting the memory contents, do not violate the setup or hold time on any of the memory block input registers during read or write operations. This is applicable if you use the memory blocks in ROM, single-port, simple dual-port, or true dual-port mode.

Simple Dual-Port Mode

In the simple dual-port mode, you can simultaneously perform one read and one write operations to different locations where Port A is used for the write operation and Port B is used for the read operation.Table 2–3 lists the M10K block mixed-width configurations in simple dual-port mode.

Table 2–3. M10K Block Mixed-Width Configurations (Simple Dual-Port Mode) (Part 1 of 2)

Read Port	Write Port									
	8K x 1	4K x 2	2K x 4	2K x 5	1K x 8	1K x 10	512 x 16	512 x 20	256 x 32	256 x 40
8K × 1	✓	✓	✓	—	✓	—	✓	—	✓	—
4K × 2	✓	✓	✓	—	✓	—	✓	—	✓	—
2K × 4	✓	✓	✓	—	✓	—	✓	—	✓	—
2K × 5	—	—	—	✓	—	✓	—	✓	—	✓
1K × 8	✓	✓	✓	—	✓	—	✓	—	✓	—
1K × 10	—	—	—	✓	—	✓	—	✓	—	✓

Table 2–3. M10K Block Mixed-Width Configurations (Simple Dual-Port Mode) (Part 2 of 2)

Read Port	Write Port									
	8K x 1	4K x 2	2K x 4	2K x 5	1K x 8	1K x 10	512 x 16	512 x 20	256 x 32	256 x 40
512 × 16	✓	✓	✓	—	✓	—	✓	—	✓	—
512 × 20	—	—	—	✓	—	✓	—	✓	—	✓
256 × 32	✓	✓	✓	—	✓	—	✓	—	✓	—
256 × 40	—	—	—	✓	—	✓	—	✓	—	✓

True Dual-Port Mode

In true dual-port mode, the M10K block can perform any combination of two port operations—two read operations, two write operations, or one write operation at two different clock frequencies. Table 2–4 lists the M10K block mixed-width configurations in true dual-port mode.

Table 2–4. M10K Block Mixed-Width Configurations (True Dual-Port Mode)

Port B	Port A							
	8K x 1	4K x 2	2K x 4	2K x 5	1K x 8	1K x 10	512 x 16	512 x 20
8K x 1	✓	✓	✓	—	✓	—	✓	—
4K x 2	✓	✓	✓	—	✓	—	✓	—
2K x 4	✓	✓	✓	—	✓	—	✓	—
2K x 5	—	—	—	✓	—	✓	—	✓
1K x 8	✓	✓	✓	—	✓	—	✓	—
1K x 10	—	—	—	✓	—	✓	—	✓
512 x 16	✓	✓	✓	—	✓	—	✓	—
512 x 20	—	—	—	✓	—	✓	—	✓

Clocking Modes

Arria V embedded memory blocks support the following clocking modes:

- Independent clock mode
- Input/output clock mode
- Read/write clock mode
- Single clock mode

CAUTION

To avoid corrupting the memory contents, do not violate the setup or hold time on the memory block address registers during read or write operations.

Table 2–5 lists the internal memory clock modes supported for each memory mode.

Table 2–5. Internal Memory Clock Modes

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

For more information about each clock mode, refer to *Internal Memory (RAM and ROM) User Guide*.

Design Considerations

When designing with Arria V embedded memory blocks, you must consider the following factors:

- Embedded memory blocks selection—consider the manual or automatic memory blocks selection by the Quartus II software, and also the dual-purpose architecture of the MLABs.
- Conflict resolution—implement external conflict resolution logic to the memory block to avoid unknown data being written to the address if you use memory blocks in true dual-port mode.
- Read-during-write behavior—customize the read-during-write behavior of the embedded memory blocks to suit your design requirements.
- Power-up conditions and memory initialization—consider the power up conditions of the embedded memory if you are designing logic that evaluates the initial power-up values.
- Power management—reduce power consumption in your design by controlling the clocking of each memory block.

For more information about design considerations, refer to "Design Consideration" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

Document Revision History

Table 2–6 lists the revision history for this chapter.

Table 2–6. Document Revision History

Date	Version	Changes
November 2011	1.1	■ Updated Table 2–1. ■ Restructured chapter.
May 2011	1.0	Initial release.

3. Variable-Precision DSP Blocks in Arria V Devices

AV-52003-1.1

This chapter describes how the variable-precision digital signal processing (DSP) blocks in Arria® V devices are optimized to support higher bit precision in high-performance DSP applications.

This chapter contains the following sections:

Features

Architectural highlights of the Arria V variable-precision DSP block include:

- High-performance, power-optimized, and fully registered multiplication operations
- Supports 9-bit, 18-bit, and 27-bit word lengths
- Supports 18 x 19 complex multiplications
- Supports floating-point arithmetic formats (32-bit for single precision, 64-bit for double precision, and 43- to 63-bit for single-extended precision)
- Built-in addition, subtraction, and dual 64-bit accumulation unit to combine multiplication results efficiently
- Cascading 19-bit or 27-bit form tap-delay line for filtering applications
- Cascading 64-bit output bus to propagate output results from one block to the next block without external logic support
- Hard pre-adder supported in 19-bit and 27-bit mode for symmetric filters
- Internal coefficient register bank for filter implementation
- Supports 18-bit and 27-bit systolic finite impulse response (FIR) filters with distributed output adder

For more information about the number of multipliers in each Arria V device, refer to the *Overview for Arria V Device Family* chapter.

Supported Operational Modes

Table 3–1 summarizes the operational modes that are supported by Arria V variable-precision DSP blocks.

Table 3–1. Variable-Precision DSP Blocks Operational Modes for Arria V Devices

Variable-Precision DSP Block Resources	Operation Mode	Supported Instance	Pre-Adder Support	Coefficient Support	Input Cascade Support	Chainout Support
1 variable-precision DSP block	Independent 9 x 9 multiplication	3	No	No	No	No
	Independent 18 x 18 multiplication	2	Yes	Yes	Yes (1)	No
	Independent 18 x 19 multiplication	2	Yes	Yes	Yes (1)	No
	Independent 18 x 25 multiplication	1	Yes	Yes	Yes (1)	Yes
	Independent 20 x 24 multiplication	1	Yes	Yes	Yes (1)	Yes
	Independent 27 x 27 multiplication	1	Yes	Yes	Yes (1)	Yes
	Two 18 x 19 multiplier adder mode	1	Yes	Yes	Yes (1)	Yes
	18 x 18 multiplier adder summed with 36-bit input	1	Yes	No	No	Yes
2 variable-precision DSP blocks	Complex 18 x 19 multiplication	1	No	No	Yes	No
	Two 27 x 27 multiplier adder	1	Yes	Yes	Yes (1)	Yes
	Four 18 x 19 multiplier adder	1	Yes	Yes	Yes (1)	Yes

Note to Table 3–1:

(1) When you enable the pre-adder feature, the input cascade support is not available.

The Quartus® II software includes megafunctions that you can use to control the operation mode of the multipliers. After making the appropriate parameter settings with the MegaWizard™ Plug-In Manager, the Quartus II software automatically configures the variable-precision DSP block.

For more information, refer to *Megafunction Overview User Guide*, *Integer Arithmetic Megafunctions User Guide* and *Floating-Point Megafunctions User Guide*.

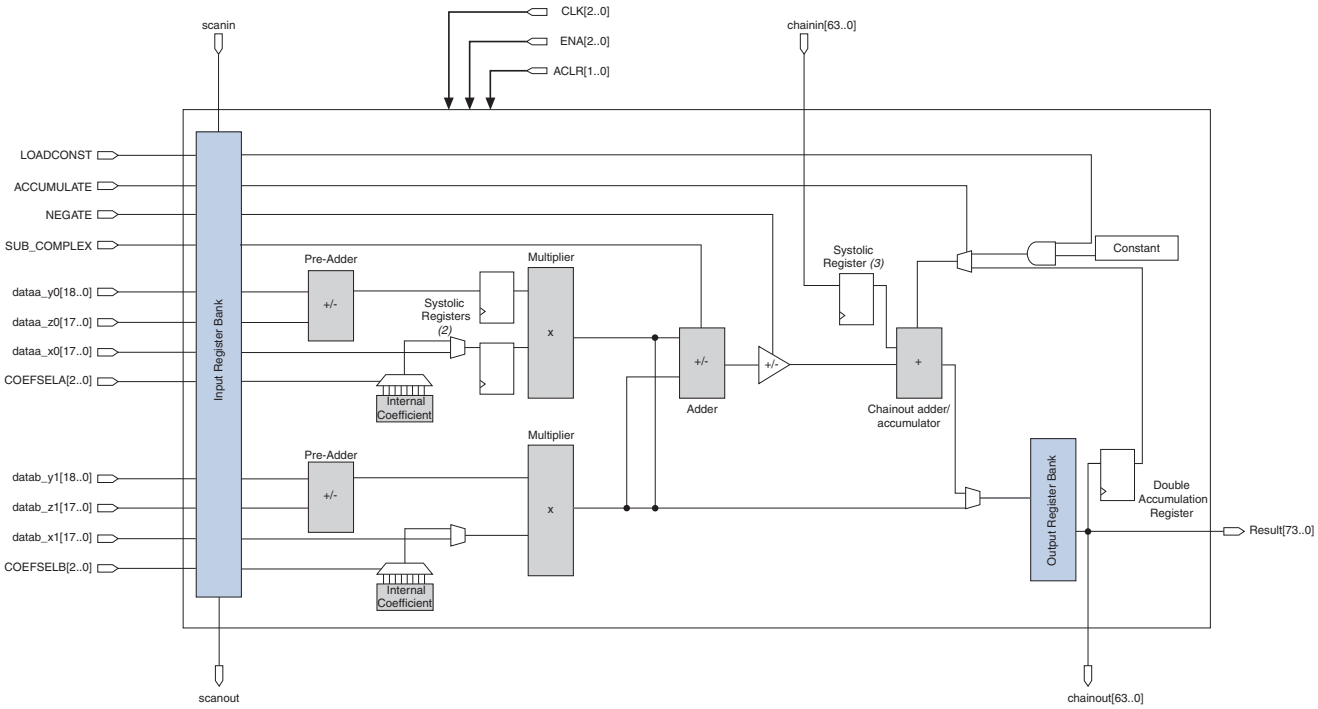
Resource Descriptions

The Arria V variable-precision DSP block consists of the following elements:

- "Input Register Bank" on page 3–4
- "Pre-Adder and Internal Coefficient" on page 3–5
- "Multipliers" on page 3–6
- "Accumulator and Adder" on page 3–6
- "Systolic Registers" on page 3–7
- "Output Register Bank" on page 3–7

Figure 3–1 shows a detailed overall architecture of the Arria V variable-precision DSP block.

Figure 3–1. Variable Precision DSP Block Architecture for Arria V Devices (1)



Notes to Figure 3–1:

(1) If the variable-precision DSP block is not configured in systolic FIR mode, both systolic registers are bypassed.
(2) When enabled, systolic registers are clocked with the same clock source as the multiplier inputs.
(3) When enabled, systolic registers are clocked with the same clock source as the output register bank.

Input Register Bank

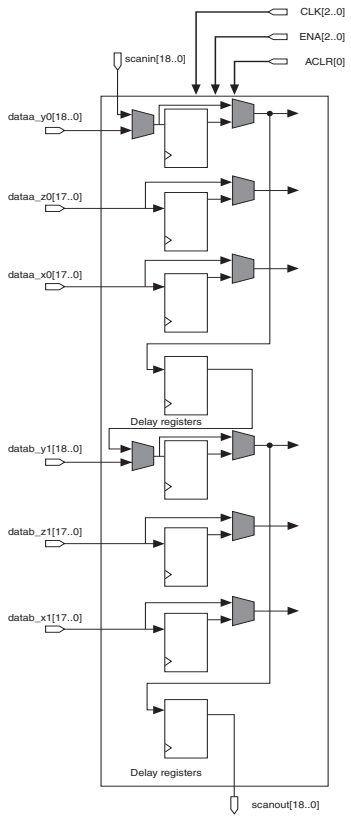
All the registers in the DSP blocks are positive-edge triggered and cleared on power up. Each multiplier operand can feed an input register or a multiplier directly, bypassing the input registers. The following variable-precision DSP block signals control the input registers within the variable-precision DSP block:

- CLK[2..0]
- ENA[2..0]
- ACLR[0]

Besides the registers for the data and dynamic control signals, there are also two sets of delay registers in the input register bank. In 18 x 19 mode, you can use these delay registers to balance the latency requirements when you use both the input cascade and chainout features.

The tap-delay line feature allows you to drive the top leg of the multiplier input, dataa_y0 and datab_y1 in 18 x 19 mode and dataa_y0 only in 27 x 27 mode, from the general routing or cascade chain. Figure 3–2 and Figure 3–3 show the input register for 18 x19 mode and 27 x 27 mode for Arria V devices.

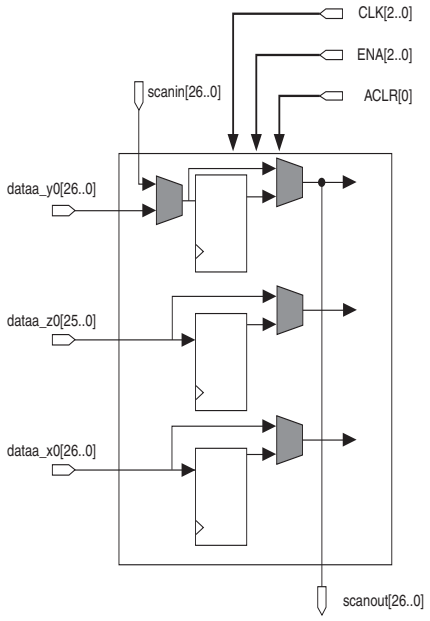
Figure 3–2. Input Register of a Variable Precision DSP Block in 18 x 19 Mode for Arria V Devices *(1)*



Note to Figure 3–2:

(1) Figure 3–2 shows only the data registers. Registers for the control signals are not shown.

Figure 3–3. Input Register of a Variable Precision DSP Block in 27 x 27 Mode for Arria V Devices *(1)*



Note to Figure 3–3:

(1) Figure 3–3 shows only the data registers. Registers for the control signals are not shown.

Pre-Adder and Internal Coefficient

The pre-adder supports both addition, substraction, and the following:

- 18-bit (signed) addition or subtraction for 18 x 19 mode
- 17-bit (unsigned) addition or subtraction for 18 x 19 mode
- 26-bit addition or subtraction for 27 x 27 mode

Each variable-precision DSP block has two 19-bit pre-adders. You can configure these pre-adders as two 19-bit pre-adders or one 27-bit pre-adder.

The Arria V variable-precision DSP block has the flexibility of selecting the multiplicand from either the dynamic input or internal coefficient. The internal coefficient can support up to eight constant coefficients for the multiplicands in 18-bit and 27-bit modes. When you enable the internal coefficient feature, `COEFSELA/COEFSELB` are used to control the selection of the coefficient multiplexer.

In both 18-bit and 27-bit modes, you can use the coefficient feature and pre-adder feature independently.

When you enable the pre-adder feature, all input data and multipliers must have the same clock setting.

Multipliers

There are two multipliers per variable-precision DSP block. You can configure these two multipliers to work as a 27 x 27 multiplier, two 18 x 19 multipliers, or three 9 x 9 multipliers, depending on the operational mode. A single variable-precision DSP block can perform many multiplications in parallel, depending on the data width of the multiplier.

For more information, refer to "Variable-Precision DSP Blocks" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

Accumulator and Adder

The Arria V variable-precision DSP block supports a 64-bit accumulator. You can dynamically control the function of the accumulator by three control signals—NEGATE, LOADCONST, and ACCUMULATE. Table 3–2 lists how these dynamic signals control the accumulator functions in Arria V devices.

Table 3–2. Dynamic Control Signals for 64-Bit Accumulator in Arria V Devices

Function	Description	NEGATE	LOADCONST	ACCUMULATE
Zeroing	Disables the accumulator.	0	0	0
Preload	Loads an initial value to the accumulator. Only one bit of the 64-bit preload value can be "1". It can be used as rounding the DSP result to any position of the 64-bit result.	0	1	0
Accumulation	Adds the current result to the previous accumulate result.	0	X (1)	1
Decimation	This function takes the current result, converts it into two's compliment, and adds it to the previous result.	1	X (1)	X (1)

Note to Table 3–2:

(1) X denotes a "don't care" value.

The adder is a 74-bit adder. Because the accumulator size is 64-bit, you can use the adder as one 64-bit adder in most cases. You can also use it as several small adders with various sizes, depending on the operational mode.

- In two 18 x 19 mode, the 74-bit adder is divided into two 37-bit adders to produce the full 37-bit result of each independent 18 x 19 multiplication.
- In three 9 x 9 mode, you can use the adder as three 18-bit adders to produce three 9 x 9 multiplication results independently.
- Use the adder for adding results from other DSP blocks by using the output chaining path.

By enabling the 64-bit double accumulation registers located between the output register bank and the accumulator, the accumulator supports double accumulation. The double accumulation registers are set statically in the programming file.

Systolic Registers

There are two systolic registers per variable-precision DSP block. The first set of systolic registers consists of 18-bit and 19-bit registers that are used to register the 18-bit and 19-bit inputs of the upper multiplier, respectively. You must clock these registers with the same clock source as the multiplier inputs. The second set of systolic registers are used to delay the `chainout` output to the next variable-precision DSP block. You must clock this register with the same clock source as the output register bank. If the variable-precision DSP block is not configured in systolic FIR mode, both systolic registers are bypassed.

Output Register Bank

The positive edge of the clock signal triggers the 74-bit bypassable output register bank and is cleared after power up. The following variable-precision DSP block signals control the output register per variable-precision DSP block:

- `CLK[2..0]`
- `ENA[2..0]`
- `ACLR[1]`

Document Revision History

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

Table 3–3. Document Revision History

Date	Version	Changes
November 2011	1.1	Restructured chapter.
May 2011	1.0	Initial release.

4. Clock Networks and PLLs in Arria V Devices

AV-52004-1.0

This chapter describes the advanced features of hierarchical clock networks and phase-locked loops (PLLs) in Arria® V devices. The Quartus® II software enables the PLLs and their features without external devices.

The chapter contains the following sections:

Clock Networks in Arria V Devices

The global clock networks (GCLKs), regional clock networks (RCLKs), and periphery clock networks (PCLKs) available in Arria V devices are organized into hierarchical clock structures. The clock networks provide up to 348 unique clock domains (16 GCLKs + 88 RCLKs + 244 PCLKs) within the Arria V device and allow up to 99 unique GCLK, RCLK, and PCLK clock sources (16 GCLKs + 22 RCLKs + 61 PCLKs) per device quadrant.

Table 4–1 lists the clock resources available in Arria V devices.

Table 4–1. Clock Resources in Arria V Devices—*Preliminary*

Clock Resource	Number of Resources Available	Source of Clock Resource
Clock input pins	40 Single-ended (20 Differential) (1) or 48 Single-ended (24 Differential) (2)	`CLK[0..19]p` and `CLK[0..19]n` pins (1) or `CLK[0..23]p` and `CLK[0..23]n` pins (2)
GCLK networks	16	`CLK[0..23]p` and `CLK[0..23]n` pins, PLL clock outputs, and logic array
RCLK networks	88	`CLK[0..23]p` and `CLK[0..23]n` pins, PLL clock outputs, and logic array
PCLK networks	104,184, 224, and 244 (3)	DPA clock outputs, PLD-transceiver interface clocks, I/O pins, and logic array
GCLKs and RCLKs per quadrant	38	16 GCLKs + 22 RCLKs
GCLKs and RCLKs per device	104	16 GCLKs + 88 RCLKs

Notes to Table 4–1:

(1) This only applies to 5AGXA1 and 5AGXA3 devices.

(2) This applies to all Arria V devices except for 5AGXA1 and 5AGXA3 devices.

(3) There are 104 PCLKs in 5AGXA1 and 5AGXA3 devices, 184 PCLKs in 5AGXA5 and 5AGXA7 devices, 224 PCLKs in 5AGXB1, 5AGXB3, and 5AGTD3 devices, and 244 PCLKs in 5AGXB5, 5AGXB7, and 5AGTD7 devices.

Arria V PLLs cannot be driven by internally-generated GCLKs, RCLKs, or PCLKs. The input clock to the PLL has to come from dedicated clock input pins, PLL-fed GCLKs, or PLL-fed RCLKs.

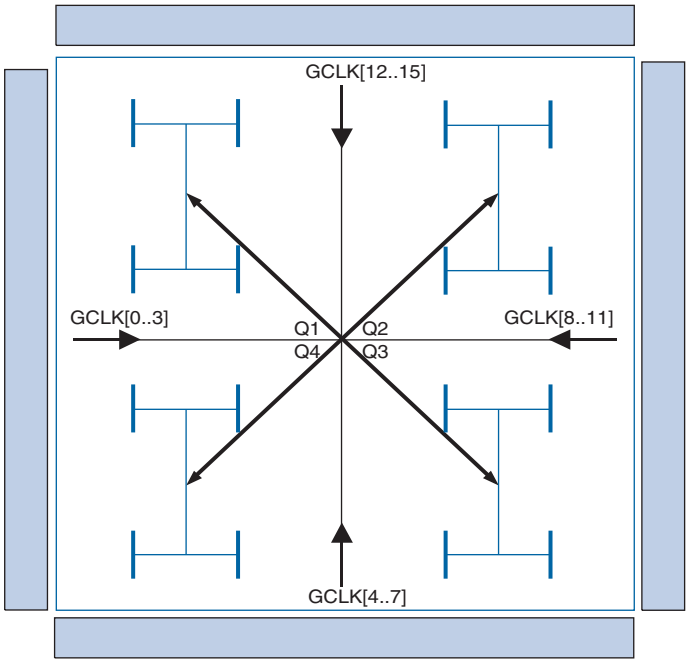
Arria V devices have up to 48 dedicated single-ended clock pins or 24 dedicated differential clock pins (CLK[0..23]p and CLK[0..23]n) that can drive either the GCLK or RCLK networks.

For more information about how to connect the clock input pins, refer to *Arria V Device Family Pin Connection Guidelines*.

Global Clock Networks

Arria V 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 V device I/O elements (IOEs) and internal logic can also drive GCLKs to create internally generated global clocks and other high fan-out control signals; for example, synchronous or asynchronous clears and clock enables. Figure 4–1 shows the GCLK networks in Arria V devices.

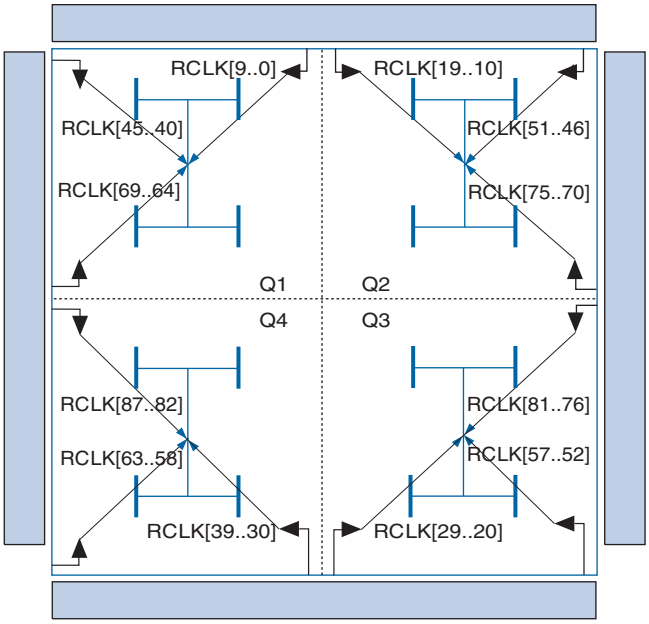
Figure 4–1. GCLK Networks in Arria V Devices



Regional Clock Networks

RCLK networks only pertain to the quadrant they drive into. RCLK networks provide the lowest clock insertion delay and skew for logic contained within a single device quadrant. The Arria V device IOEs and internal logic within a given quadrant can also drive RCLKs to create internally generated regional clocks and other high fan-out control signals; for example, synchronous or asynchronous clears and clock enables. Figure 4–2 shows the RCLK networks in Arria V devices.

Figure 4–2. RCLK Networks in Arria V Devices



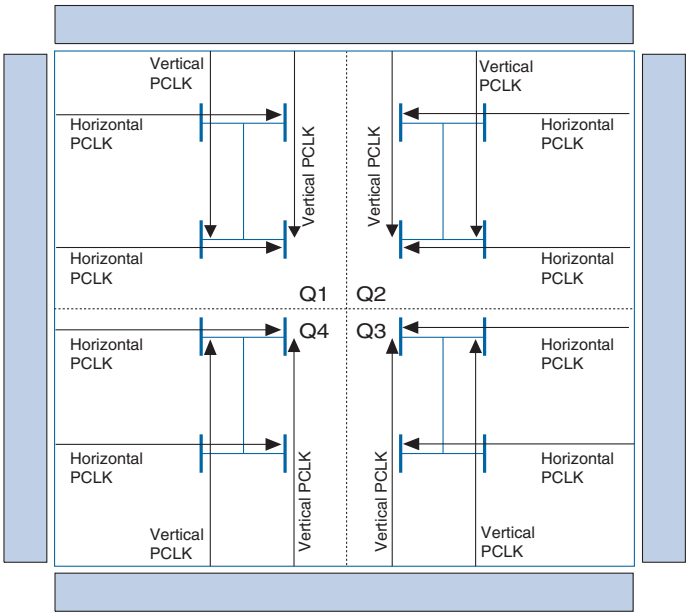
Periphery Clock Networks

Figure 4–3 shows the PCLK networks in Arria V devices. Depending on the routing direction, there are vertical PCLKs from the top and bottom periphery and horizontal PCLKs from the left and right periphery. Clock outputs from the dynamic phase aligner (DPA) block, programmable logic device (PLD)-transceiver interface clocks, I/O pins, and internal logic can drive the PCLK networks.

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

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

Figure 4–3. PCLK Networks in Arria V Devices



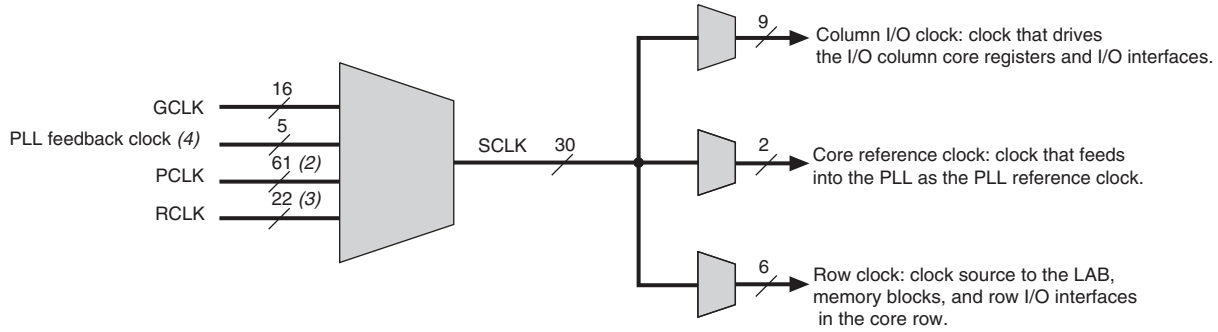
Clock Sources Per Quadrant

There are 30 section clock (SCLK) networks available in each spine clock per quadrant that can drive six row clocks in each logic array block (LAB) row, nine column I/O clocks, and two core reference clocks. The SCLKs are the clock resources to the core functional blocks, PLLs, and I/O interfaces of the device.

A spine clock is another layer of routing between the GCLKs, RCLKs, and PCLK networks 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 PCLK networks.

Figure 4–4 shows SCLKs driven by the GCLK, RCLK, PCLK, or the PLL feedback clock networks in each spine clock per quadrant.

Figure 4–4. Hierarchical Clock Networks in Each Spine Clock Per Quadrant *(1)*



Notes to Figure 4–4:

(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 61 PCLKs that can drive the SCLKs in each spine clock per quadrant in the largest device.
(3) There are up to 22 RCLKs that can drive the SCLKs in each spine clock per quadrant in the largest device.
(4) The PLL feedback clock is the clock from the PLL that drives into the SCLKs.

For more information about device clock regions, clock control block, and clock enable signals, refer to "Clock Networks and PLLs" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

Clock Control Block

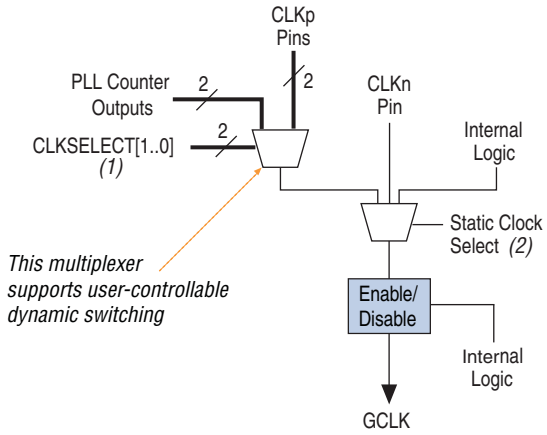
Every GCLK, RCLK, and PCLK network has its own clock control block. The control block provides the following features:

- Clock source selection (dynamic selection available only for GCLKs)
- Global clock multiplexing
- Clock power down (static or dynamic clock enable or disable available only for GCLKs and RCLKs)

Figure 4–5, Figure 4–6, and Figure 4–7 show the GCLK, RCLK, and PCLK control blocks, respectively.

You can select the clock source for the GCLK select block either statically or dynamically using internal logic to drive the multiplexer-select inputs. When selecting the clock source dynamically, you can select either PLL outputs (such as `C0` or `C1`) or a combination of clock pins or PLL outputs.

Figure 4–5. GCLK Control Block for Arria V Devices



Notes to Figure 4–5:

(1) When the device is in user mode, you can dynamically control the clock select signals through internal logic.

(2) When the device is in user mode, you can only set the clock select signals through a configuration file (SRAM object file [**.sof**] or programmer object file [**.pof**]) because the signals cannot be controlled dynamically.

You can set the input clock sources and the `clkena` signals for the GCLK and RCLK network multiplexers through the Quartus II software using the ALTCLKCTRL megafunction.

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

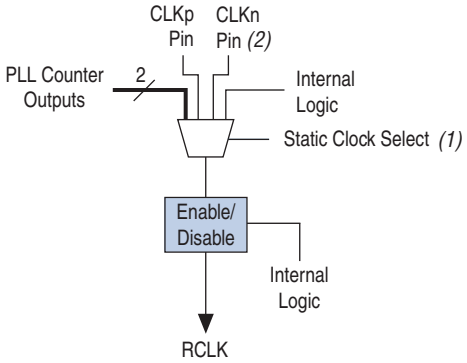
Table 4–1 lists the mapping between the input clock pins, PLL counter outputs, and clock control block inputs.

Table 4–1. Mapping Between the Input Clock Pins, PLL Counter Outputs, and Clock Control Block Inputs

Clock	Fed by
`inclk[0]` and `inclk[1]`	Any of the four dedicated clock pins on the same side of the Arria V device
`inclk[2]`	Arria V device
`inclk[3]`	Arria V device

CAUTION You cannot use corner PLLs for dynamic clock control selection.

Figure 4–6. RCLK Control Block



Notes to Figure 4–6:

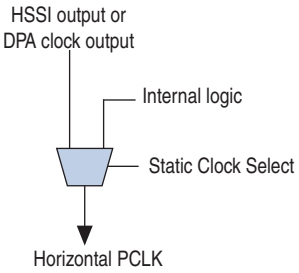
(1) When the device is in user mode, you can only set the clock select signals through a configuration file (**.sof** or **.pof**); they cannot be dynamically controlled.

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

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

You can select the HSSI output or internal logic to drive the HSSI horizontal PCLK control block. Alternatively, you can also use the DPA clock output or internal logic to drive the DPA horizontal PCLK. You can only use the DPA output to generate the vertical PCLK to the core.

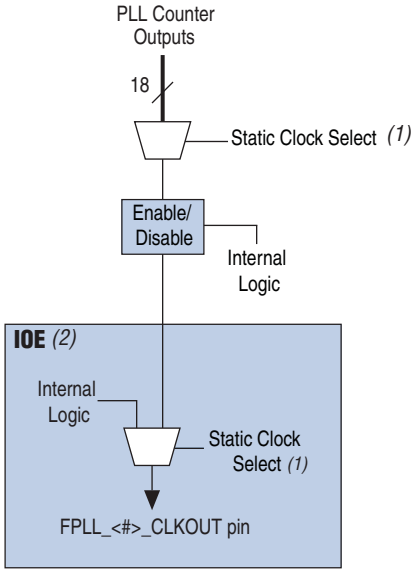
Figure 4–7. Horizontal PCLK Control Block



You can power down the Arria V GCLK and RCLK clock networks using both static and dynamic approaches. When a clock network is powered down, all the logic fed by the clock network is in off-state, thereby reducing the overall power consumption of the device. The unused GCLK, RCLK, and PCLK networks are automatically powered down through configuration bit settings in the configuration file (**.sof** or **.pof**) generated by the Quartus II software. The dynamic clock enable or disable feature allows the internal logic to control power-up or power-down synchronously on the GCLK and RCLK networks, including dual-regional clock regions.

You can enable or diable the dedicated external clock output pins using the ALTCLKCTRL megafunction. F shows the external PLL output clock control block.

Figure 4–8. External PLL Output Clock Control Block for Arria V Devices



Notes to Figure 4–8:

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

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

Arria V PLLs

The Arria V device family contains fractional PLLs in addition to the existing integer PLLs that provide robust clock management and synthesis for device clock management, external system clock management, and high-speed I/O interfaces. Two adjacent fractional PLLs share 18 output counters that support integer or fractional frequency synthesis.

Arria V devices offer up to 16 fractional PLLs in the larger densities. All Arria V fractional PLLs have the same core analog structure and features support.

Table 4–2 lists the features in Arria V PLLs.

Table 4–2. PLL Features in Arria V Devices —*Preliminary*

Feature	Support
Integer PLL	Yes
Fractional PLL	Yes
C output counters	18 (1)
M, N, C counter sizes	1 to 512
Dedicated external clock outputs	4 single-ended or 2 single-ended and 1 differential
Clock input pins	4 single-ended or 4 differential
External feedback input pin	Single-ended or differential
Spread-spectrum input clock tracking	Yes (2)
Source synchronous compensation	Yes
Direct compensation	Yes
Normal compensation	Yes
Zero-delay buffer compensation	Yes
External feedback compensation	Yes
LVDS compensation	Yes
VCO output drives the DPA clock	Yes
Phase shift resolution	78.125 ps (3)
Programmable duty cycle	Yes

Notes to Table 4–2:

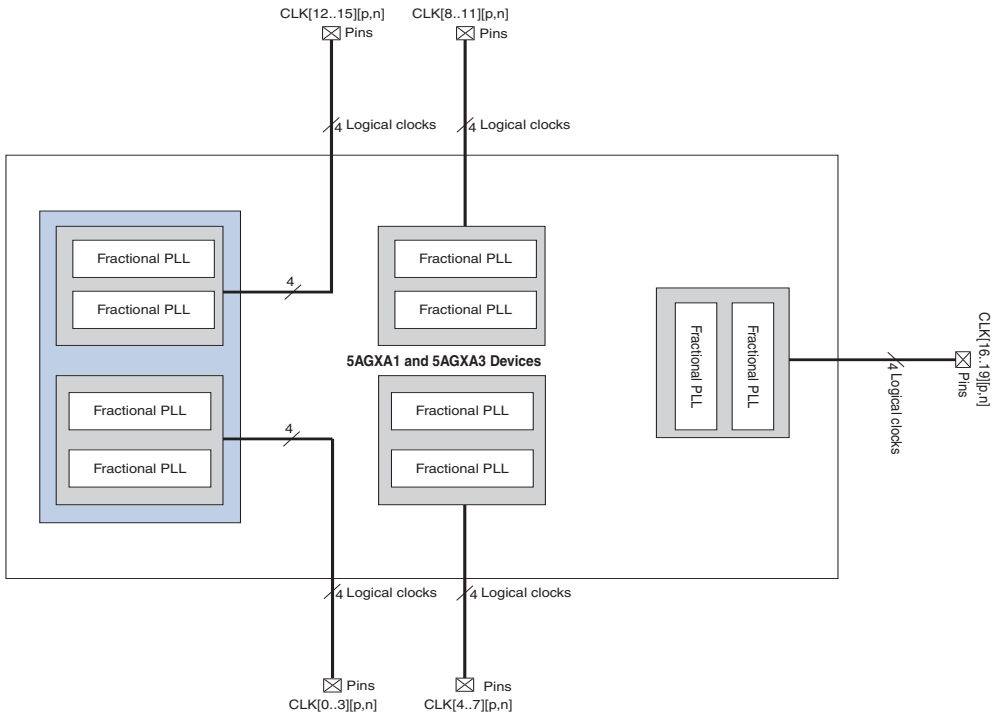
(1) Two adjacent fractional PLLs share 18 output counters.

(2) Provided input clock jitter is within input jitter tolerance specifications and the modulation frequency of the input clock is below the PLL bandwidth which is specified in the Fitter report.

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

Figure 4–9 through Figure 4–12 on page 4–13 show the physical locations of the fractional PLLs.

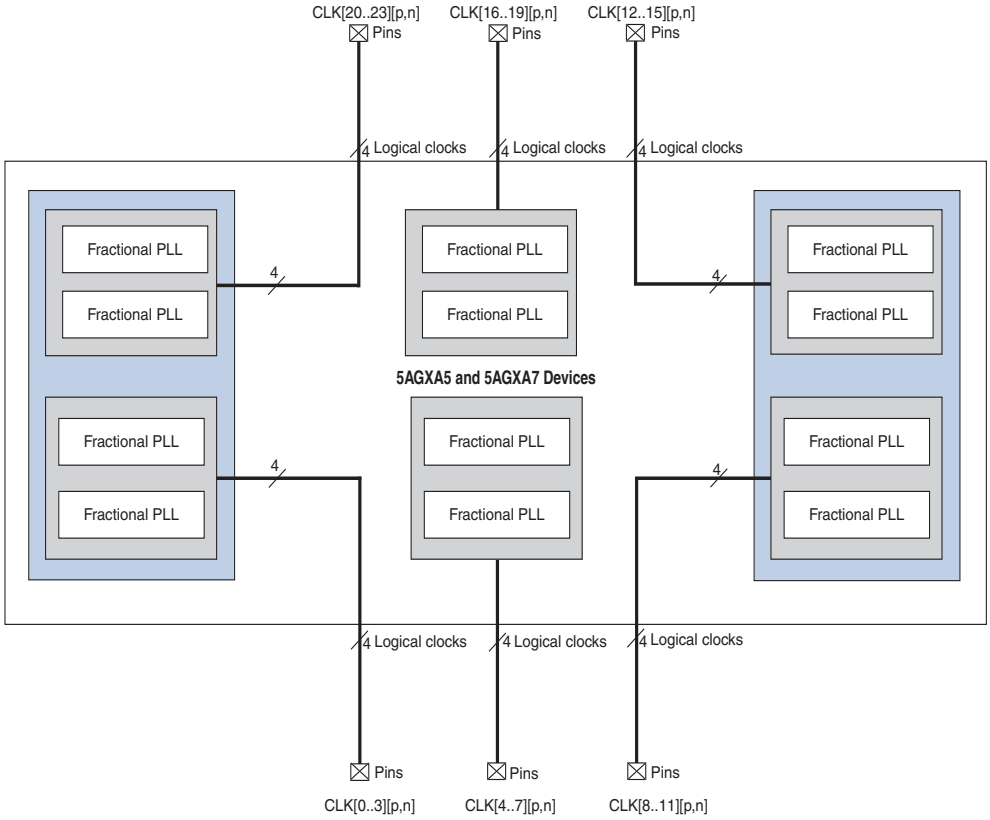
Figure 4–9. PLL Locations for 5AGXA1 and 5AGXA3 Devices *(1)*



Note to Figure 4–9:

(1) Fractional PLLs coordinates will be finalized in the future release of the Quartus II software.

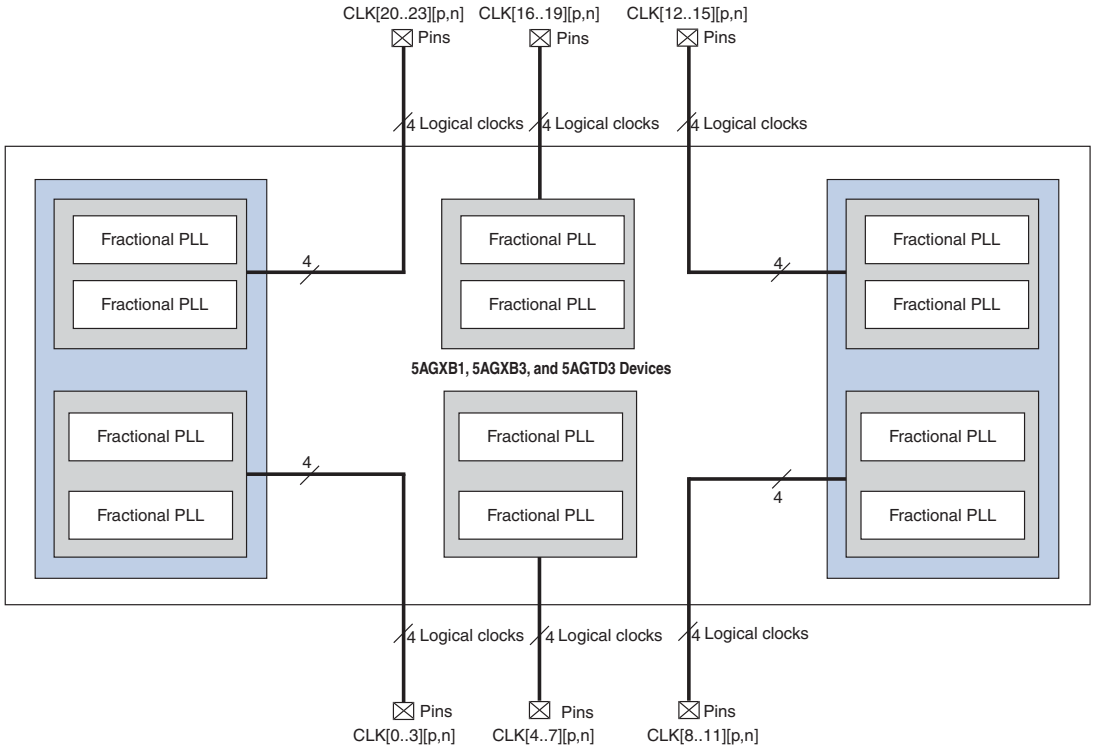
Figure 4–10. PLL Locations for 5AGXA5 and 5AGXA7 Devices *(1)*



Note to Figure 4–10:

(1) Fractional PLL coordinates will be finalized in the future release of the Quartus II software.

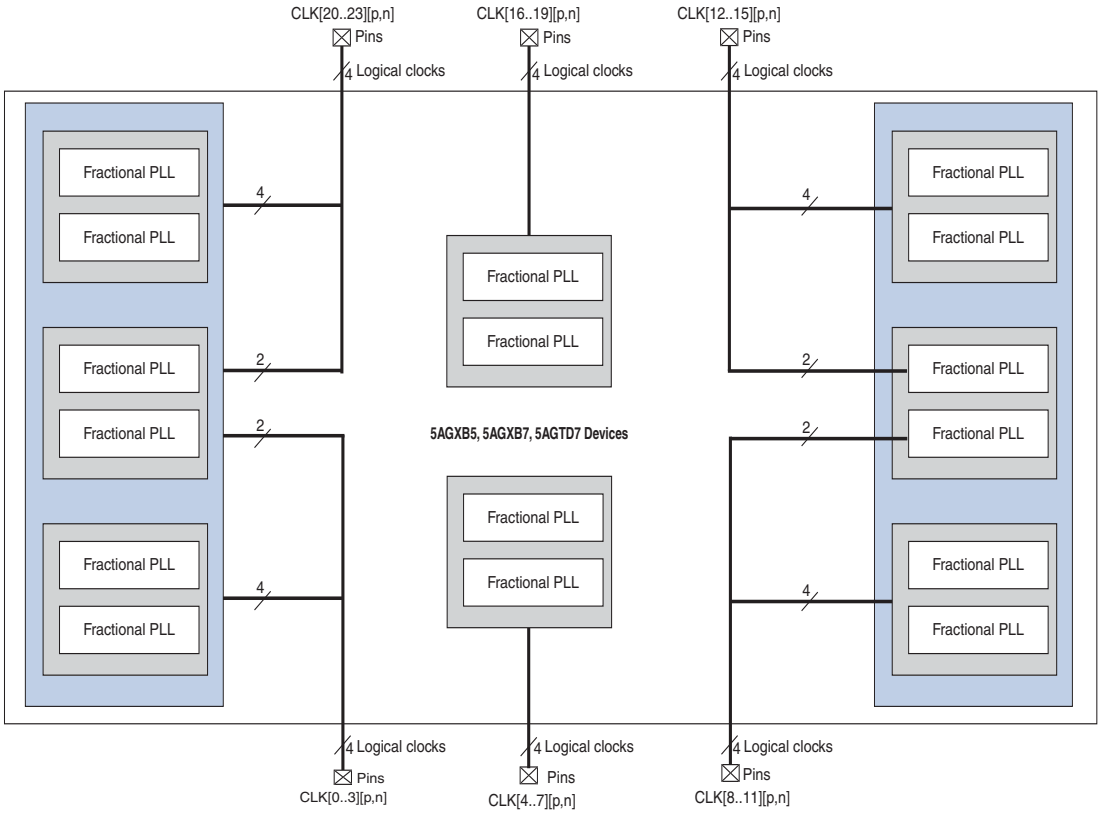
Figure 4–11. PLL Locations for 5AGXB1, 5AGXB3, and 5AGTD3 Devices *(1)*



Note to Figure 4–11:

(1) Fractional PLL coordinates will be finalized in the future release of the Quartus II software.

Figure 4–12. PLL Locations for 5AGXB5, 5AGXB7, and 5AGTD7 Devices *(1)*



Note to Figure 4–12:

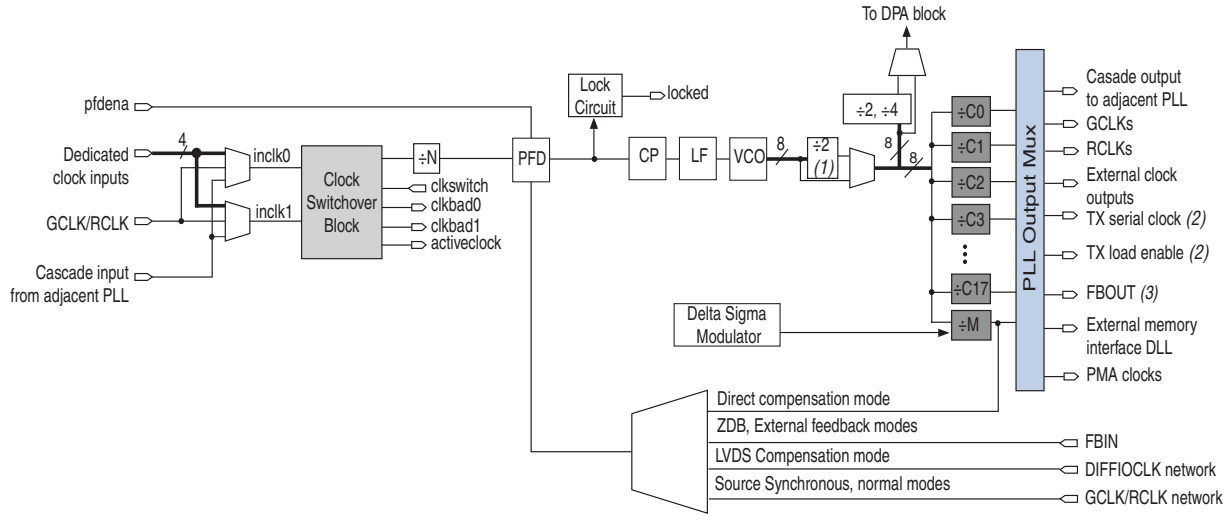
(1) Fractional PLL coordinates will be finalized in the future release of the Quartus II software.

For more information about fractional PLL architecture and usage, refer to "Clock Networks and PLLs in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

Fractional PLL Architecture

Figure 4–13 shows the high-level block diagram of the Arria V fractional PLL.

Figure 4–13. Fractional PLL High-Level Block Diagram



Notes to Figure 4–13:

(1) This is the VCO post-scale counter K.
(2) Only C0, C2, C15, and C17 can drive the TX serial clock and C1, C3, C14, and C16 can drive the TX load enable.
(3) This FBOUT port is fed by the M counter in the Arria V PLLs.

Fractional PLL Usage

You can configure the fractional PLL as either an integer or as enhanced fractional mode. One fractional PLL can use up to 18 output counters and all external clock outputs. Two adjacent fractional PLLs share the 18 output counters.

You can use fractional PLLs to reduce the number of required oscillators on the board, as well as to reduce the clock pins used in the FPGA by synthesizing multiple clock frequencies from a single reference clock source. In addition, you can use fractional PLLs for clock network delay compensation, zero-delay buffering, and transmit clocking for transceivers.

Clock Feedback Modes

This section describes the clock feedback modes that Arria V PLLs support and the specific implementation of zero-delay buffer (ZDB) mode and external feedback (EFB) mode.

Arria V PLLs support the following clock feedback modes:

- Source synchronous mode
- LVDS compensation
- Direct compensation
- Normal mode
- ZDB mode

- EFB mode

For more information about clock feedback modes, refer to "Clock Feedback Modes" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

PLL External Clock I/O Pins

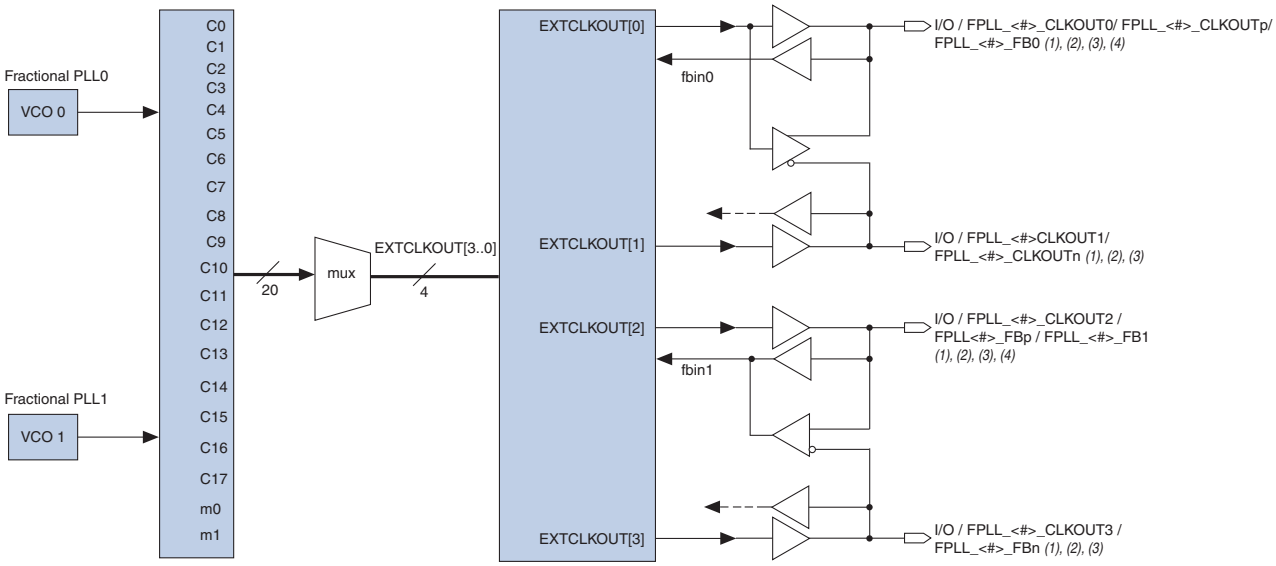
Two adjacent fractional PLLs share four dual-purpose clock I/O pins, organized as one of the following combinations:

- Four single-ended clock outputs
- Two single-ended outputs and one differential clock output
- Four single-ended clock outputs and two single-ended feedback inputs within the I/O driver feedback for ZDB support
- Two single-ended clock outputs and two single-ended feedback inputs for single-ended external feedback (EFB) support
- One differential clock output and one differential feedback input for differential EFB support (only one of the two adjacent fractional PLLs can support differential EFB while the other fractional PLL can be used for general-purpose clocking)

The center fractional PLLs on the left and right sides of 5AGXB5, 5AGXB7, and 5AGTD7 devices do not support external clock outputs.

Figure 4–14 shows the dual-purpose clock I/O pins associated with the PLL for Arria V devices.

Figure 4–14. Dual-Purpose Clock I/O Pins Associated with PLL for Arria V Devices



Notes to Figure 4–14:

(1) You can feed these clock output pins using any one of the `C[17..0]` or `M` counters. When not used as external clock outputs, these clock output pins can be used as regular user I/Os.

(2) The `FPLL_<#>_CLKOUT0`, `FPLL_<#>_CLKOUT1`, `FPLL_<#>_CLKOUT2`, and `FPLL_<#>_CLKOUT3` pins are single-ended clock output pins.

(3) The `FPLL_<#>_CLKOUTp` and `FPLL_<#>_CLKOUTn` pins are differential output pins while the `FPLL_<#>_FBp` and `FPLL_<#>_FBn` pins are differential feedback input pins to support differential EFB only in VCO 1.

(4) The `FPLL_<#>_FB0` and `FPLL_<#>_FB1` pins are single-ended feedback input pins.

Figure 4–14 shows that any of the output counters (`C[17..0]`) on the PLLs or the `M` counter can feed the dedicated external clock outputs. Therefore, one counter or frequency can drive all output pins available from a given PLL.

Each pin of a single-ended output pair can be either in-phase or 180° out-of-phase. The Quartus II software places the NOT gate in the design into the IOE to implement the 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, 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 *I/O Features in Arria V Devices* chapter.

Arria V 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.

For more information about clock feedback modes, clock multiplication and division, and programmable duty cycle, refer to "Clock Networks and PLLs in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

Document Revision History

Table 4–3 lists the revision history for this chapter.

Table 4–3. Document Revision History

Date	Version	Changes
November 2011	1.1	Restructured chapter.
May 2011	1.0	Initial release.

Section II. I/O Interfaces for Arria V Devices

This section provides information about Arria® V device I/O features, external memory interfaces, and high-speed differential I/O interfaces and dynamic phase aligner (DPA). This section includes the following chapters:

- Chapter 5, I/O Features in Arria V Devices
- Chapter 6, High-Speed Differential I/O Interfaces and DPA in Arria V Devices
- Chapter 7, External Memory Interfaces in Arria V Devices

Revision History

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

5. I/O Features in Arria V Devices

AV52005-1.2

This chapter provides details about the features of the Arria® V I/O elements (IOEs) and how the IOEs work.

Arria V I/Os support a wide range of features:

- Single-ended, non voltage-referenced, and voltage-referenced I/O standards
- **Low-voltage differential signaling** (**LVDS**), **RSDS**, **mini-LVDS**, **HSTL**, and **SSTL**
- Hard DPA block with serializer/deserializer (SERDES)
- Programmable output current strength
- Programmable slew rate
- Programmable bus-hold
- Programmable pull-up resistor
- Open-drain output
- On-chip series termination (R_S OCT)
- On-chip parallel termination (R_T OCT)
- On-chip differential termination (R_D OCT)
- Programmable pre-emphasis
- Programmable I/O delay
- Programmable voltage output differential (V_{OD})

All information in this chapter is applicable to all Arria V variants, unless noted otherwise.

This chapter contains the following sections:

For more information about design considerations, refer to "I/O Interface Design Considerations" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

I/O Standards Support

Table 5–1 lists the supported I/O standards and typical V_{CCIO}, V_{CCPD}, V_{REF}, and board V_{TT} values for input and output.

V_{CCPD} powers the single-ended **HSTL/SSTL/HSUL**, **differential SSTL/HSTL/HSUL**, and **LVDS** input buffers. **Differential HSTL**, **SSTL**, and **HSUL** outputs are not true differential outputs. These I/O standards use two single-ended outputs with the second output programmed as inverted. **Differential HSTL**, **SSTL**, and **HSUL** inputs use **LVDS** differential input buffers. However, R_D support is only available if the I/O standard is **LVDS**.

Table 5–1. Arria V I/O Standards and Voltage Levels (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			
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	3.0/2.5	3.0	3.0	—	—
2.5-V LVCMOS	JESD8-5	3.0/2.5	2.5	2.5	—	—
1.8-V 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	JESD8-9B	V_{CCPD}	2.5	2.5	1.25	1.25
SSTL-2 Class II	JESD8-9B	V_{CCPD}	2.5	2.5	1.25	1.25
SSTL-18 Class I	JESD8-15	V_{CCPD}	1.8	2.5	0.90	0.90
SSTL-18 Class II	JESD8-15	V_{CCPD}	1.8	2.5	0.90	0.90
SSTL-15 Class I	—	V_{CCPD}	1.5	2.5	0.75	0.75
SSTL-15 Class II	—	V_{CCPD}	1.5	2.5	0.75	0.75
SSTL-15	JESD79-3D	V_{CCPD}	1.5	2.5	0.75	*(2)*
SSTL-135	—	V_{CCPD}	1.35	2.5	0.675	*(2)*
SSTL-125	—	V_{CCPD}	1.25	2.5	0.625	*(2)*
1.8-V HSTL Class I	JESD8-6	V_{CCPD}	1.8	2.5	0.90	0.90
1.8-V HSTL Class II	JESD8-6	V_{CCPD}	1.8	2.5	0.90	0.90
1.5-V HSTL Class I	JESD8-6	V_{CCPD}	1.5	2.5	0.75	0.75
1.5-V HSTL Class II	JESD8-6	V_{CCPD}	1.5	2.5	0.75	0.75
1.2-V HSTL Class I	JESD8-16A	V_{CCPD}	1.2	2.5	0.6	0.6
1.2-V HSTL Class II	JESD8-16A	V_{CCPD}	1.2	2.5	0.6	0.6
HSUL-12	—	V_{CCPD}	1.2	2.5	0.6	*(2)*
Differential SSTL-2 Class I	JESD8-9B	V_{CCPD}	2.5	2.5	—	1.25
Differential SSTL-2 Class II	JESD8-9B	V_{CCPD}	2.5	2.5	—	1.25
Differential SSTL-18 Class I	JESD8-15	V_{CCPD}	1.8	2.5	—	0.90
Differential SSTL-18 Class II	JESD8-15	V_{CCPD}	1.8	2.5	—	0.90

Table 5–1. Arria V I/O Standards and Voltage Levels (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			
Differential SSTL-15 Class I	—	V_{CCPD}	1.5	2.5	—	0.75
Differential SSTL-15 Class II	—	V_{CCPD}	1.5	2.5	—	0.75
Differential 1.8-V HSTL Class I	JESD8-6	V_{CCPD}	1.8	2.5	—	0.90
Differential 1.8-V HSTL Class II	JESD8-6	V_{CCPD}	1.8	2.5	—	0.90
Differential 1.5-V HSTL Class I	JESD8-6	V_{CCPD}	1.5	2.5	—	0.75
Differential 1.5-V HSTL Class II	JESD8-6	V_{CCPD}	1.5	2.5	—	0.75
Differential 1.2-V HSTL Class I	JESD8-16A	V_{CCPD}	1.2	2.5	—	0.60
Differential 1.2-V HSTL Class II	JESD8-16A	V_{CCPD}	1.2	2.5	—	0.60
Differential SSTL-15	JESD79-3D	V_{CCPD}	1.5	2.5	—	*(2)*
Differential SSTL-135	—	V_{CCPD}	1.35	2.5	—	*(2)*
Differential SSTL-125	—	V_{CCPD}	1.25	2.5	—	*(2)*
Differential HSUL-12	—	V_{CCPD}	1.2	2.5	—	*(2)*
LVDS	ANSI/TIA/EIA-644	V_{CCPD}	2.5	2.5	—	—
RSDS	—	V_{CCPD}	2.5	2.5	—	—
Mini-LVDS	—	V_{CCPD}	2.5	2.5	—	—
LVPECL	—	*(3)*	—	2.5	—	—

Notes to Table 5–1:

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

(2) This I/O standard typically does not require board termination.

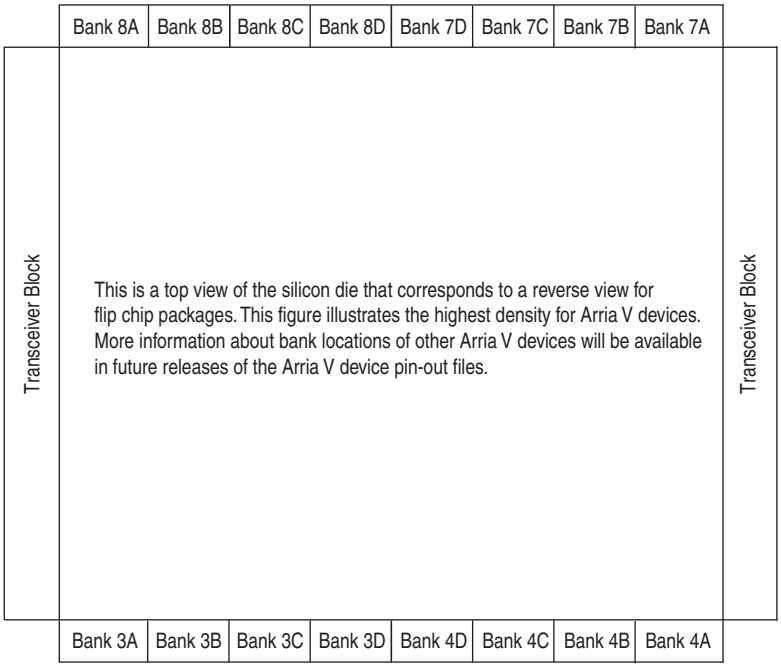
(3) The support for the **LVPECL** I/O standard is only for input clock operation. V_{CCPD} powers the differential clock input buffers.

All I/O banks support the true **LVDS**, **RSDS**, and **mini-LVDS** I/O standards by using true **LVDS** output buffers without resistor networks. The I/O banks also support **emulated LVDS**, **RSDS**, and **mini-LVDS** I/O standards by using two single-ended output buffers with a three-resistor (**LVDS_E_3R**, **RSDS_E_3R**, and **mini-LVDS_E_3R**) network. The emulated differential output standards that support the tri-state feature includes **LVDS_E_3R**, **RSDS_E_3R**, and **mini_LVDS_E_3R**.

I/O Banks

Figure 5–1 shows a top view of the silicon die, displaying the I/O banks in Arria V devices.

Figure 5–1. I/O Banks for Arria V Devices—*Preliminary*



All I/O banks in Arria V devices contain true differential input and output buffers, and dedicated circuitry to support differential I/O standards.

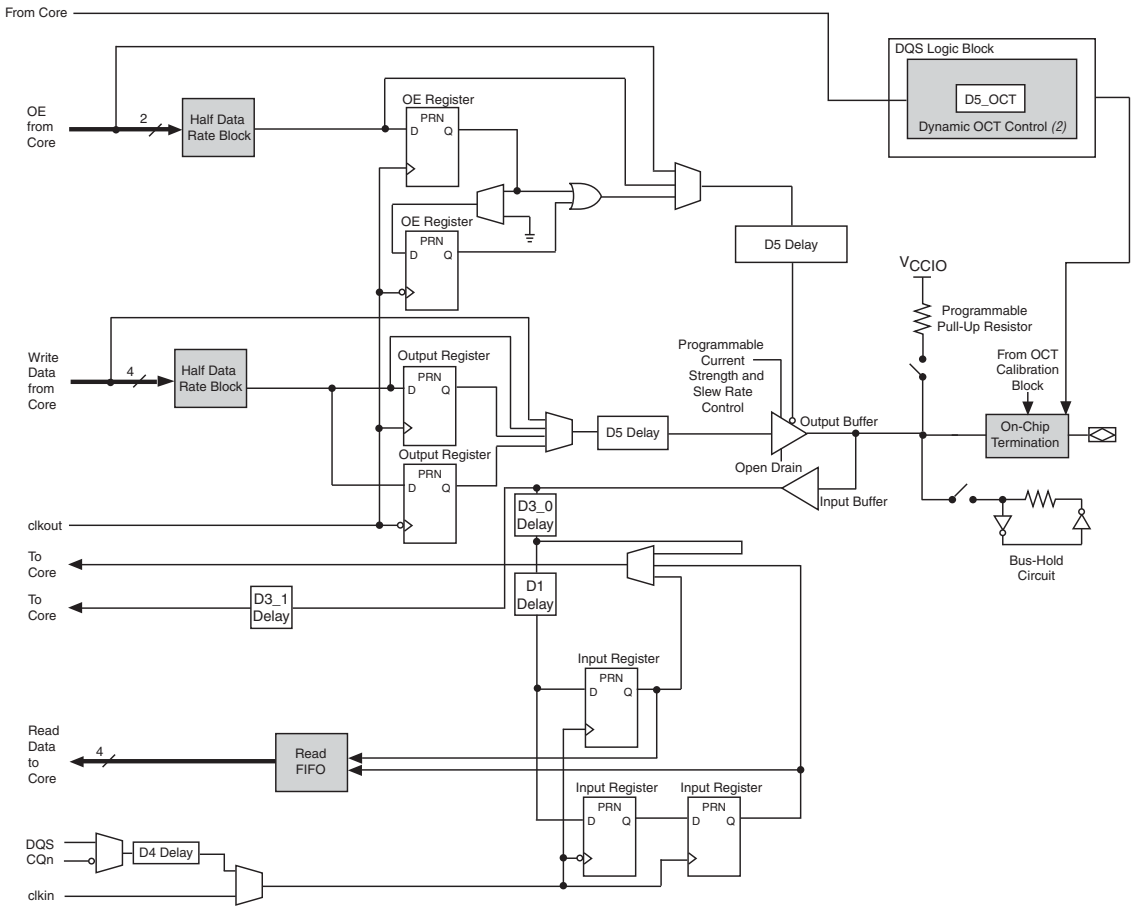
Each I/O bank in Arria V devices supports a high-performance external memory interface. The I/O pins are organized in pairs to support differential I/O standards. Each I/O pin pair can support both differential input and output buffers.

The I/O pins in Arria V devices are arranged in groups called modular I/O banks. The number of Arria V I/O banks in a particular device ranges from 16 to 18, depending on the device density.

I/O Element Features

The IOEs in Arria V devices contain a bidirectional I/O buffer and I/O registers to support a complete 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 V device. Figure 5–2 shows the Arria V IOE structure.

Figure 5–2. IOE Structure for Arria V Devices *(1), (2)*



Notes to Figure 5–2:

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

(2) One dynamic on-chip termination (OCT) control is available for each DQ/DQS group.

I/O Programmable Features

The Arria V I/O supports programmable features, as listed in Table 5–2.

Table 5–2. Supported I/O Features and Settings (Part 1 of 2)

Feature *(1)*	Setting	Condition
Slew Rate Control	0 = Slow, 1 = Fast (default)	Disabled when you use the R_S OCT feature.
I/O Delay	*(2)*	—
Open-Drain Output	On, Off (default)	—
Bus-Hold	On, Off (default)	Disabled when you use the weak pull-up resistor feature.

Table 5–2. Supported I/O Features and Settings (Part 2 of 2)

Feature (1)	Setting	Condition
Weak Pull-up Resistor	On, Off (default)	Disabled when you use the bus-hold feature.
Pre-Emphasis	0 = Disabled, 1 = Enabled (default)	—
Differential Output Voltage	0 = low, 1 = medium (default), 2 = high	—
On-Chip Clamping Diode	On, Off	—

Notes to Table 5–2:

(1) For more information about each supported feature, refer to "I/O Element Features" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

(2) For information about the programmable IOE and output buffer delays, refer to *Device Datasheet for Arria V Devices* chapter.

Current Strength

The output buffer for each Arria V device I/O pin has a programmable current strength control for certain I/O standards. You can use the programmable current strength to mitigate the effects of high signal attenuation that is caused by a long transmission line or a legacy backplane.

Table 5–3 lists the programmable current strength settings for Arria V devices.

Table 5–3. Programmable Current Strength Settings

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

Altera recommends that you perform IBIS or SPICE simulations to determine the best current strength setting for your specific application.

MultiVolt I/O Interface

The Arria V architecture supports the MultiVolt I/O interface feature that allows Arria V devices in all packages to interface with systems of different supply voltages. You must connect the Arria V VCCPD power pins to a 2.5- or 3.0-V power supply to increase the performance of the output pins. Table 5–4 lists Arria V MultiVolt I/O support.

Table 5–4. MultiVolt I/O Support in Arria V Devices (1)

V_{CCIO} (V)	Input Signal (V)								Output Signal (V)							
	1.2	1.25	1.35	1.5	1.8	2.5	3.0	3.3	1.2	1.25	1.35	1.5	1.8	2.5	3.0	3.3
1.2	✓	—	—	—	—	—	—	—	✓	—	—	—	—	—	—	—
1.25	—	✓	—	—	—	—	—	—	—	✓	—	—	—	—	—	—
1.35	—	—	✓	—	—	—	—	—	—	—	✓	—	—	—	—	—
1.5	—	—	—	✓	✓	—	—	—	—	—	—	✓	—	—	—	—
1.8	—	—	—	✓	✓	—	—	—	—	—	—	—	✓	—	—	—
2.5	—	—	—	—	—	✓	✓ (2)	✓ (2)	—	—	—	—	—	✓	—	—
3.0	—	—	—	—	—	✓	✓ (2)	✓ (2)	—	—	—	—	—	—	✓	—
3.3	—	—	—	—	—	✓	✓ (2)	✓ (2)	—	—	—	—	—	—	—	✓

Notes to Table 5–4:

(1) The pin current may be slightly higher than the default value. Verify that the V_{OL} maximum and V_{OH} minimum voltages of the driving device do not violate the applicable V_{IL} maximum and V_{IH} minimum voltage specifications of the Arria V device.

(2) Altera recommends using the on-chip clamp diode on the I/O pins when the input signal is 3.0 V or 3.3 V.

OCT Support

Arria V devices feature dynamic R_S and R_T OCT to provide I/O impedance matching and termination capabilities. OCT maintains signal quality, saves board space, and reduces external component costs. Arria V devices support OCT in all I/O banks.

Table 5–5 lists the OCT schemes supported in Arria V devices.

Table 5–5. OCT Schemes in Arria V Devices

Direction	OCT Schemes
Output	OCT R_S with calibration (1)
	OCT R_S without calibration (1)
Input	OCT R_T with calibration (1)
	OCT R_D (**LVDS** I/O standard only)
Bidirectional	Dynamic OCT R_S and OCT R_T

Note to Table 5–5:

(1) For information about the selectable I/O standards, refer to Table 5–6 on page 5–8.

For more information about each OCT scheme, refer to "OCT Support" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

You can use R_S and R_T OCT in the same I/O bank for different I/O standards if the R_S and R_T OCT use the same VCCIO supply voltage. You cannot configure both the R_S OCT and the programmable current strength for the same I/O buffer.

The Arria V device uses the RZQ pin available in each calibration block during the calibration process. Connect the RZQ pin to the GND pin through a resistor with the specified value. The RZQ pin shares the same VCCIO supply voltage with the I/O bank where the pin is located.

Table 5–6 lists the input and output termination settings for calibrated and uncalibrated OCT on different I/O standards.

Table 5–6. Selectable I/O Standards for R_S and R_T OCT with and without Calibration (Part 1 of 2)

I/O Standard	Output Termination			Input Termination	
	Uncalibrated OCT Setting	Calibrated OCT Setting		Calibrated OCT Setting	
	R_S (Ω)	R_S (Ω)	RZQ (Ω)	R_T (Ω)	RZQ (Ω)
3.3-V LVTTL/3.3-V LVCMOS	—	—	—	—	—
3.0-V LVVTL/3.0-V LVCMOS	25/50	25/50	100	—	—
2.5-V LVCMOS	25/50	25/50	100	—	—
1.8-V LVCMOS	25/50	25/50	100	—	—
1.2-V LVCMOS	25/50	25/50	100	—	—
SSTL-2 Class I	50	50	100	50	100
SSTL-2 Class II	25	25	100	50	100
SSTL-18 Class I	50	50	100	50	100
SSTL-18 Class II	25	25	100	50	100
SSTL-15 Class I	50	50	100	50	100
SSTL-15 Class II	25	25	100	50	100
1.8-V HSTL Class I	50	50	100	50	100
1.8-V HSTL Class II	25	25	100	50	100
1.5-V HSTL Class I	50	50	100	50	100
1.5-V HSTL Class II	25	25	100	50	100
1.2-V HSTL Class I	50	50	100	50	100
1.2-V HSTL Class II	25	25	100	50	100
SSTL-15	—	25/50	100	20, 30, 40, 60, 120	240
		34/40	240		
SSTL-135	—	34/40	240	20, 30, 40, 60, 120	240
SSTL-125	—	34/40	240	20, 30, 40, 60, 120	240
HSUL-12	—	34/40/48/60/80	240	—	—
Differential SSTL-2 Class I	50	50	100	50	100
Differential SSTL-2 Class II	25	25	100	50	100
Differential SSTL-18 Class I	50	50	100	50	100
Differential SSTL-18 Class II	25	25	100	50	100

Table 5–6. Selectable I/O Standards for R_S and R_T OCT with and without Calibration (Part 2 of 2)

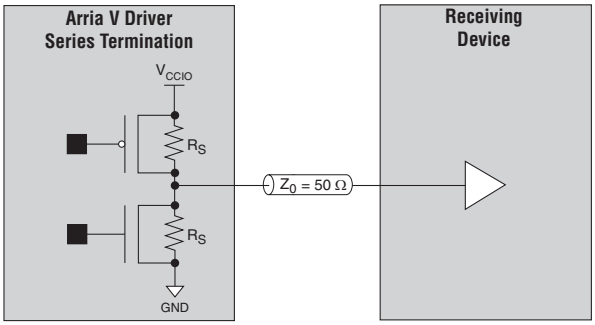
I/O Standard	Output Termination			Input Termination	
	Uncalibrated OCT Setting	Calibrated OCT Setting		Calibrated OCT Setting	
	R_S (Ω)	R_S (Ω)	RZQ (Ω)	R_T (Ω)	RZQ (Ω)
Differential SSTL-15 Class I	50	50	100	50	100
Differential SSTL-15 Class II	25	25	100	50	100
Differential 1.8-V HSTL Class I	50	50	100	50	100
Differential 1.8-V HSTL Class II	25	25	100	50	100
Differential 1.5-V HSTL Class I	50	50	100	50	100
Differential 1.5-V HSTL Class II	25	25	100	50	100
Differential 1.2-V HSTL Class I	50	50	100	50	100
Differential 1.2-V HSTL Class II	25	25	100	50	100
Differential SSTL-15	—	25/50	100	20, 30, 40, 60, 120	240
		34/40	240		
Differential SSTL-135	—	34/40	240	20, 30, 40, 60, 120	240
Differential SSTL-125	—	34/40	240	20, 30, 40, 60, 120	240
Differential HSUL-12	—	34/40/48/60/80	240	—	—

R_S OCT Without Calibration

Arria V 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 signal reflections on PCB traces.

Arria V devices support R_S OCT for single-ended and voltage-referenced I/O standards. Figure 5–3 shows the R_S as the intrinsic impedance of the output transistors. When you select matching impedance, current strength is no longer selectable.

Figure 5–3. R_S OCT Without Calibration

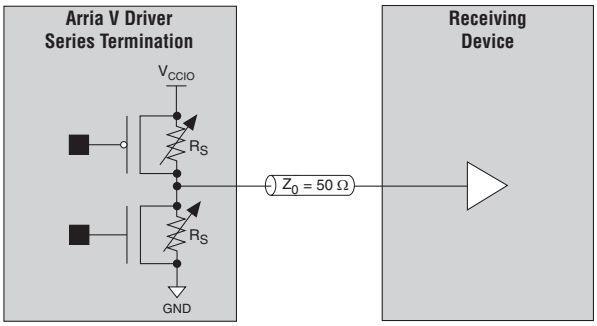


R_S OCT with Calibration

Arria V devices support R_S OCT with calibration in all banks. The R_S OCT calibration circuit compares the total impedance of the I/O buffer to the external reference resistor connected to the RZQ pin and dynamically enables or disables the transistors until they match.

The R_S is the intrinsic impedance of the transistors, as shown in Figure 5–4. Calibration occurs at the end of device configuration. When the calibration circuit finds the correct impedance, the circuit powers down and stops changing the characteristics of the drivers.

Figure 5–4. R_S OCT with Calibration

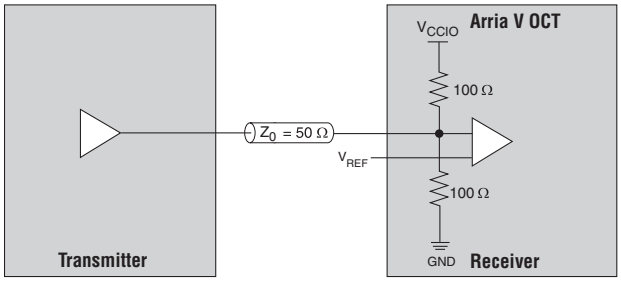


R_T OCT with Calibration

Arria V devices support R_T OCT with calibration in all banks. This support is available only for configuration of input and bidirectional pins. Output pin configurations do not support R_T OCT with calibration.

Figure 5–5 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 you enable the R_T OCT.

Figure 5–5. R_T OCT with Calibration



The R_T OCT calibration circuit compares the total impedance of the I/O buffer to the external resistor connected to the RZQ pin. The circuit dynamically enables or disables the transistors until the total impedance of the I/O buffer matches the external resistor.

Calibration occurs at the end of the device configuration. When the calibration circuit finds the correct impedance, the circuit powers down and stops changing the characteristics of the drivers.

Dynamic OCT

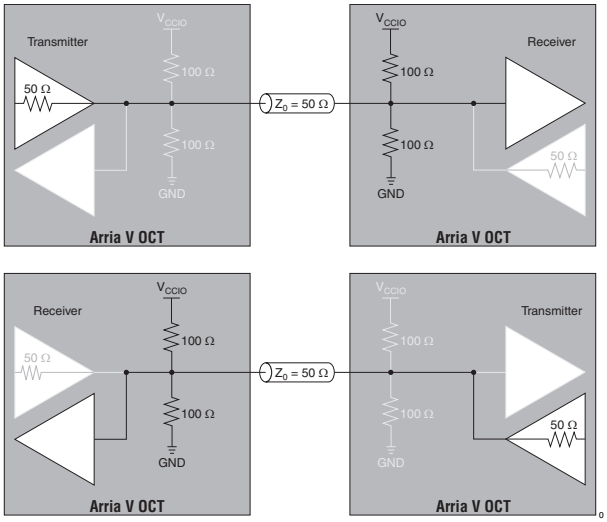
Dynamic R_T OCT or R_S OCT is enabled or disabled based on whether the bidirectional I/O acts as a receiver or driver, as listed in Table 5–7. This feature is useful for terminating any high-performance, bidirectional path because signal integrity is optimized depending on the direction of the data.

Table 5–7. Dynamic OCT Based on Bidirectional I/O

Dynamic OCT	Bidirectional I/O	State
Dynamic R_T OCT	Acts as a receiver	Enabled
	Acts as a driver	Disabled
Dynamic R_S OCT	Acts as a receiver	Disabled
	Acts as a driver	Enabled

Figure 5–6 shows the dynamic R_T OCT supported in the device.

Figure 5–6. Dynamic R_T OCT in Arria V Devices

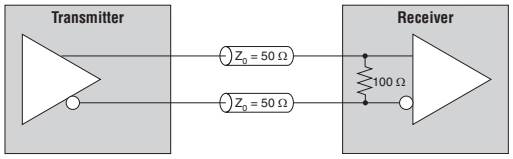


Altera recommends that you use dynamic OCT for the DDR3 memory interface if you use the **SSTL-15**, **SSTL-135**, and **SSTL-125** I/O standards. These I/O standards save board space by reducing the number of external termination resistors used.

LVDS Input R_D OCT

Arria V devices support OCT for differential **LVDS** input buffers with a nominal resistance value of 100 Ω, as shown in Figure 5–7. There is support for R_D OCT in all I/O banks. You can use R_D OCT when you set both the V_{CCIO} and V_{CCPD} to 2.5 V.

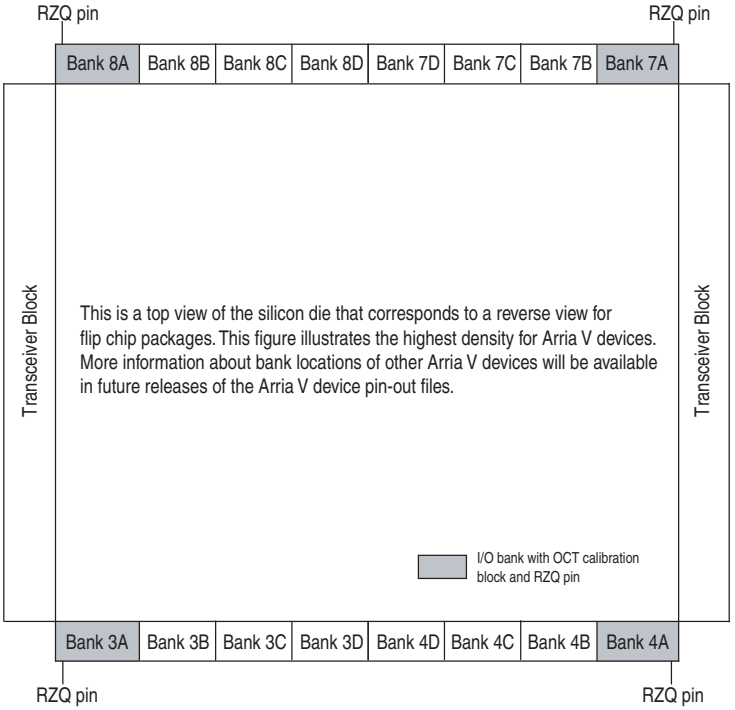
Figure 5–7. Differential Input OCT



OCT Calibration

Figure 5–8 shows the location of I/O banks with OCT calibration blocks and RZQ pins.

Figure 5–8. OCT Calibration Block and RZQ Pin Location—*Preliminary*



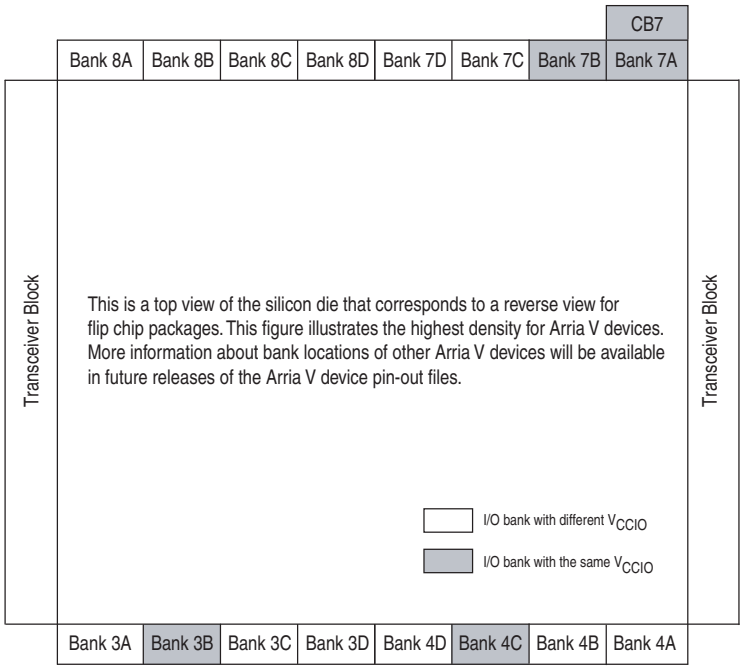
Arria V devices support calibrated R_S and calibrated R_T on all I/O pins with exceptions for dedicated configuration pins. You can calibrate using any of the available four OCT calibration blocks for each device. Each calibration block contains one RZQ pin.

Sharing an OCT Calibration Block on Multiple I/O Banks

An OCT calibration block has the same V_{CCIO} as the I/O bank that contains the block. All I/O banks support OCT calibration with different V_{CCIO} voltage standards, up to the number of available OCT calibration blocks. You can configure the I/O banks to receive calibration 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 5–9 shows a group of I/O banks that has the same V_{CCIO} voltage. This figure does not show transceiver calibration blocks.

Figure 5–9. Example of Calibrating Multiple I/O Banks with One Shared OCT Calibration Block—*Preliminary*



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, and 7B have the same V_{CCIO} as bank 7A, you can calibrate all four I/O banks (3B, 4C, 7A, and 7B) with the OCT calibration block (CB7) located in bank 7A. To enable this calibration, serially shift out the R_S OCT calibration codes from the OCT calibration block in bank 7A to the I/O banks around the periphery.

Calibration blocks are shared between I/O banks that have and do not have calibration blocks.

For more information about the OCT calibration block, refer to *Dynamic Calibrated On-Chip Termination (ALTOCT) Megafunction User Guide*.

Termination Schemes for I/O Standards

The following sections describe the different termination schemes for the I/O standards used in Arria V devices. Table 5–8 lists the external termination schemes for the different I/O standards.

Table 5–8. I/O Standards External Termination Scheme (Part 1 of 2)

I/O Standard	External Termination Scheme
3.3-V LVTTL/3.3-V LVCMOS	No external termination required
3.0-V LVVTL/3.0-V LVCMOS	
2.5-V LVCMOS	
1.8-V LVCMOS	
1.2-V LVCMOS	
3.0-V PCI	
3.0-V PCI-X	
SSTL-2 Class I	Single-Ended **SSTL** I/O Standard Termination
SSTL-2 Class II	
SSTL-18 Class I	
SSTL-18 Class II	
SSTL-15 Class I	
SSTL-15 Class II	
1.8-V HSTL Class I	Single-Ended **HSTL** I/O Standard Termination
1.8-V HSTL Class II	
1.5-V HSTL Class I	
1.5-V HSTL Class II	
1.2-V HSTL Class I	
1.2-V HSTL Class II	
SSTL-15 (1)	No external termination required
SSTL-135 (1)	
SSTL-125 (1)	
HSUL-12	
Differential SSTL-2 Class I	**Differential SSTL** I/O Standard Termination
Differential SSTL-2 Class II	
Differential SSTL-18 Class I	
Differential SSTL-18 Class II	
Differential SSTL-15 Class I	
Differential SSTL-15 Class II	
Differential 1.8-V HSTL Class I	**Differential HSTL** I/O Standard Termination
Differential 1.8-V HSTL Class II	
Differential 1.5-V HSTL Class I	
Differential 1.5-V HSTL Class II	
Differential 1.2-V HSTL Class I	
Differential 1.2-V HSTL Class II	

Table 5–8. I/O Standards External Termination Scheme (Part 2 of 2)

I/O Standard	External Termination Scheme
Differential SSTL-15 (1)	No external termination required
Differential SSTL-135 (1)	
Differential SSTL-125 (1)	
Differential HSUL-12	
LVDS	**LVDS** I/O Standard Termination
RSDS (2)	**RSDS/mini-LVDS** I/O Standard Termination
Mini-LVDS (3)	
LVPECL	**Differential LVPECL** I/O Standard Termination

Notes to Table 5–8:

(1) Altera recommends using dynamic OCT with these I/O standards to save board space and cost by reducing the number of external termination resistors.

(2) Arria V devices support the true **RSDS** output standard with data rates of up to 230 Mbps using true **LVDS** output buffer types on all I/O banks.

(3) Arria V devices support the true **mini-LVDS** output standard with data rates of up to 340 Mbps using true **LVDS** output buffer types on all I/O banks.

For more information about each external termination scheme, refer to "Termination Schemes for I/O Standards" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

Document Revision History

Table 5–9 lists the revision history for this chapter.

Table 5–9. Document Revision History

Date	Version	Changes
February 2012	1.2	Updated Table 5–4.
November 2011	1.1	■ Restructured chapter. ■ Updated Table 5–3.
May 2011	1.0	Initial release.

6. High-Speed Differential I/O Interfaces and DPA in Arria V Devices

AV52006-1.1

This chapter describes the Arria® V high-speed differential I/O interfaces and dynamic phase aligner (DPA). The chapter explains the advantages of these features over single-ended I/Os and their contribution to the overall system bandwidth achievable with Arria V FPGAs.

This chapter contains the following sections:

Features

Arria V devices feature built-in serializer/deserializer (SERDES) circuitry that supports high-speed **LVDS** interfaces at data rates of up to 1.25 Gbps. You can configure the SERDES circuitry to support source-synchronous communication protocols such as RapidIO®, XSBI, serial peripheral interface (SPI), and asynchronous protocols such as Gigabit Ethernet (GbE).

The following dedicated circuitries are available in the Arria V device family to support high-speed differential I/O:

- Differential I/O buffer
- Transmitter serializer
- Receiver deserializer
- Data realignment
- DPA
- Synchronizer (FIFO buffer)
- Fractional phase-locked loops (PLLs)

The Arria V device family supports the following differential I/O standards for high-speed differential interfaces:

- **LVDS**
- **Mini-LVDS**

- Reduced swing differential signaling (**RSDS**)

Locations of the I/O Banks

The dedicated SERDES and DPA circuitry that supports high-speed differential I/Os is located in the top and bottom banks of the Arria V devices. Figure 6–1 and Figure 6–2 show the high-level SERDES/DPA location in the Arria V device.

Figure 6–1. High-Speed Differential I/Os with DPA Locations for 5AGXA1 and 5AGXA3 Devices

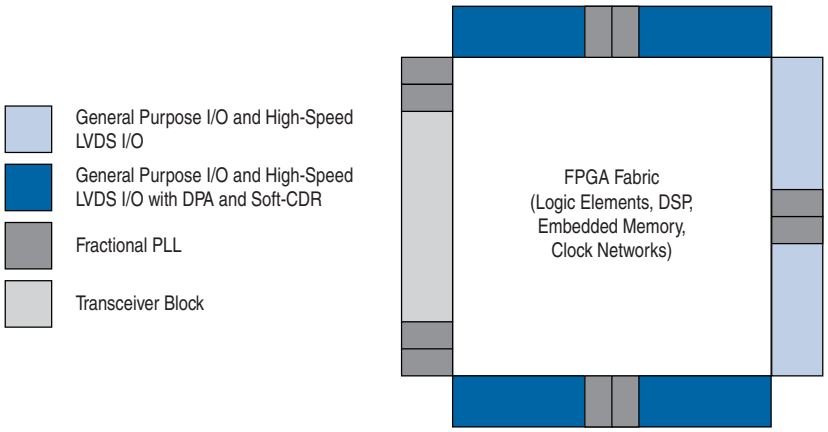
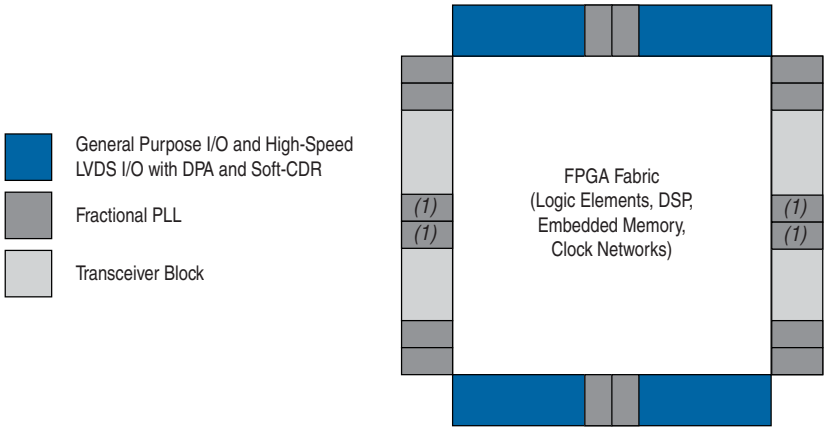


Figure 6–2. High-Speed Differential I/Os with DPA Locations for 5AGXA5, 5AGXA7, 5AGXB3, 5AGXB5, 5AGXB7, 5AGTD3, and 5AGTD7 Devices



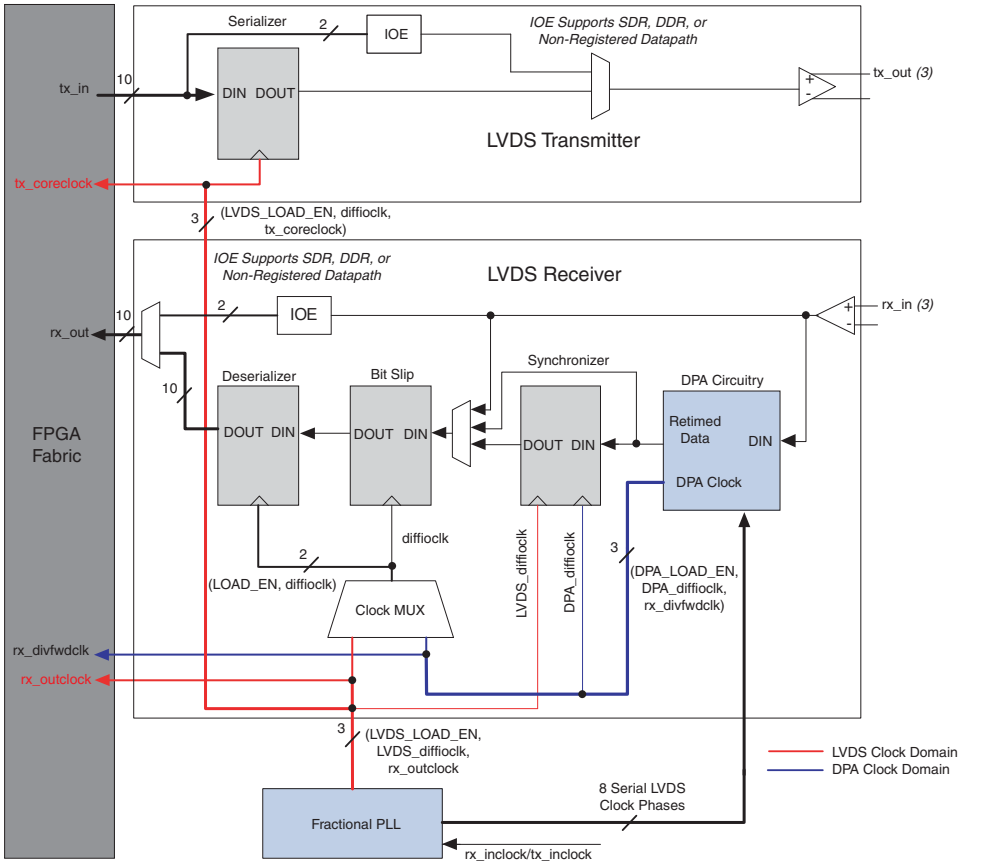
Note to Figure 6–2:

(1) Applicable only to 5AGXB5 and 5AGXB7 devices.

LVDS Channels and Dedicated Circuitry

Figure 6–3 shows a transmitter and receiver block diagram for the **LVDS** SERDES circuitry. This diagram shows the interface signals of the transmitter and receiver data path. For more information, refer to "Differential Transmitter" on page 6–5 and "Differential Receiver" on page 6–9.

Figure 6–3. LVDS SERDES *(1), (2)*



Notes to Figure 6–3:

(1) This diagram shows a shared PLL between the transmitter and receiver. If the transmitter and receiver do not share the same PLL, you require two fractional PLLs.

(2) In single data rate (SDR) and double data rate (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.

For more information about the **LVDS** transmitter and receiver port list and settings using ALTLVDS, refer to *LVDS SERDES Transmitter/Receiver (ALTLVDS_RX/TX) Megafunction User Guide*.

LVDS Channels

The Arria V device family supports **LVDS** on all I/O banks. Both row and column I/Os support true **LVDS** input buffers with R_D OCT and true **LVDS** output buffers. Alternatively, you can configure the **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 V devices offer single-ended I/O reference clock support for the **LVDS** SERDES.

☞ Emulated differential output buffers support tri-state capability.

Table 6–1 lists the number of true **LVDS** buffers supported in Arria V devices. You can use the unutilized true **LVDS** buffers as emulated **LVDS** output buffers (eTX). For example, with the 672-pin 5AGXA5 device, if you use 30 pairs of true **LVDS** input buffer (RX), you can use the remaining input buffer pairs as eTX.

Table 6–1. LVDS Channels Supported in Arria V Devices—*Preliminary* (1) (Part 1 of 2)

Device	Package	Side (2)	TX (3)	RX (3)
5AGXA1 5AGXA3 (4)	672-pin FineLine BGA, Flip Chip	Top	29	34
		Bottom	29	34
	896-pin FineLine BGA, Flip Chip	Top	34	40
		Bottom	34	40
5AGXA5 5AGXA7	672-pin FineLine BGA, Flip Chip	Top	31	36
		Bottom	31	36
	896-pin FineLine BGA, Flip Chip	Top	42	48
		Bottom	42	48
	1152-pin FineLine BGA, Flip Chip	Top	60	68
		Bottom	60	68
5AGXB1 5AGXB3	896-pin FineLine BGA, Flip Chip	Top	42	48
		Bottom	42	48
	1152-pin FineLine BGA, Flip Chip	Top	60	68
		Bottom	60	68
	1517-pin FineLine BGA, Flip Chip	Top	80	88
		Bottom	80	88
5AGXB5 5AGXB7	1152-pin FineLine BGA, Flip Chip	Top	58	66
		Bottom	58	66
	1517-pin FineLine BGA, Flip Chip	Top	78	86
		Bottom	78	86

Table 6–1. LVDS Channels Supported in Arria V Devices—*Preliminary*[1] (Part 2 of 2)

Device	Package	Side[2]	TX[3]	RX[3]
5AGTD3	896-pin FineLine BGA, Flip Chip	Top	42	48
		Bottom	42	48
	1152-pin FineLine BGA, Flip Chip	Top	60	68
		Bottom	60	68
	1517-pin FineLine BGA, Flip Chip	Top	80	88
		Bottom	80	88
5AGTD7	1152-pin FineLine BGA, Flip Chip	Top	58	66
		Bottom	58	66
	1517-pin FineLine BGA, Flip Chip	Top	78	86
		Bottom	78	86

Notes to Table 6–1:

(1) **LVDS** channel count does not include dedicated clock pins.
(2) Dedicated SERDES and DPA is available for top and bottom banks only.
(3) You can use all unutilized TX and RX as eTX.
(4) The right side I/O banks do not contain true **LVDS** output buffer. However, emulated **LVDS** output buffer (eTX) is still supported.

Differential Transmitter

The Arria V differential transmitter buffers support the following features:

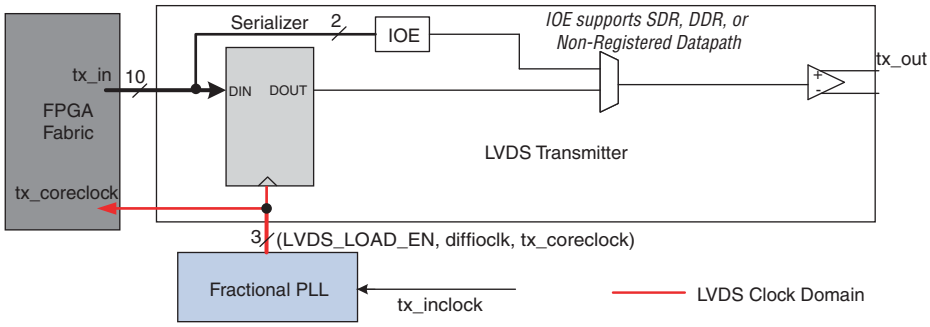
- **LVDS** signaling that can drive out **LVDS**, **mini-LVDS**, and **RSDS** signals
- Programmable V_{OD} and programmable pre-emphasis

The dedicated circuitry consists of a true differential buffer, a serializer, and fractional PLLs that you can share between the transmitter and receiver. The serializer takes up to 10 bits wide parallel data from the FPGA fabric, clocks it into the load registers, and serializes it using shift registers that are clocked by the fractional PLL before sending the data to the differential buffer. The MSB of the parallel data is transmitted first.

When you use emulated **LVDS** I/O standards at the differential transmitter, implement the SERDES circuitry in logic cells instead of the hard SERDES.

The fractional PLL generates the load enable (LVDS_LOAD_EN) signal and the diffioclk signal (the clock running at serial data rate) that clocks the load and shift registers. You can statically set the serialization factor to ×3, ×4, ×5, ×6, ×7, ×8, ×9, or ×10 using the Quartus® II software. The load enable signal is derived from the serialization factor setting. Figure 6–4 shows a block diagram of the Arria V transmitter.

Figure 6–4. Arria V Transmitter (1), (2)



Notes to Figure 6–4:

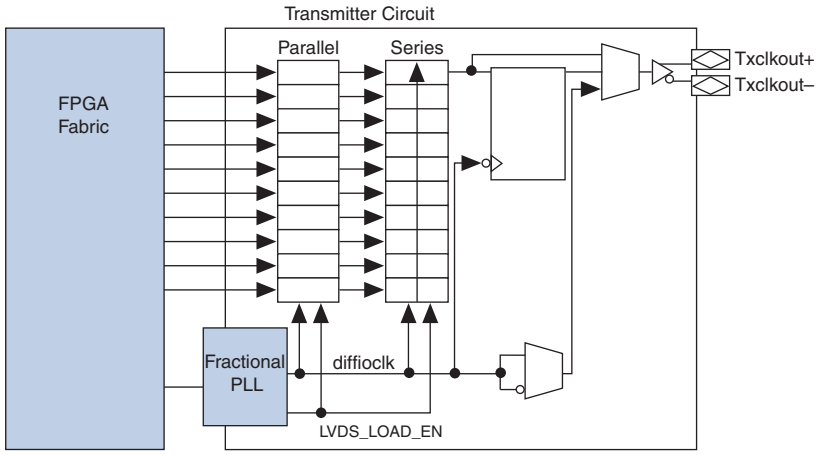
(1) In SDR and DDR modes, the data width is 1 and 2 bits, respectively.

(2) The tx_in port has a maximum data width of 10 bits.

You can configure any Arria V transmitter data channel 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. Different applications often require specific clock-to-data alignments or specific data-rate-to-clock-rate factors. The transmitter can output a clock signal at the same rate as the data—with a maximum output clock frequency that each speed grade of the device supports. You can also divide the output clock by a factor of 1, 2, 4, 6, 8, or 10, depending on the serialization factor.

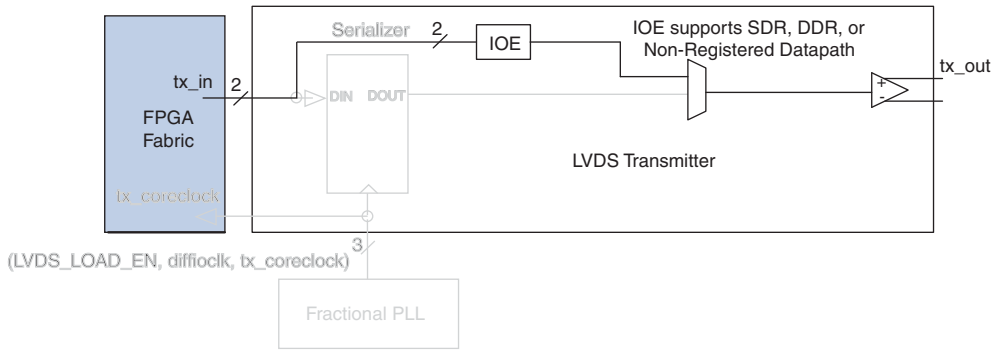
You can set the phase of the clock in relation to the data at 0° or 180° (edge or center aligned). The fractional PLLs provide additional support for other phase shifts in 45° increments. You can specify these settings statically in the Quartus II MegaWizard™ Plug-In Manager. Figure 6–5 shows the Arria V transmitter in clock output mode. In clock output mode, you can use an **LVDS** channel as a clock output channel.

Figure 6–5. Arria V Transmitter in Clock Output Mode



You can bypass the Arria V serializer to support DDR (×2) and SDR (×1) operations to achieve a serialization factor of 2 and 1, respectively. The I/O element (IOE) contains two data output registers that can each operate in either DDR or SDR mode. Figure 6–6 shows the serializer bypass path.

Figure 6–6. Arria V Serializer Bypass *(1), (2), (3)*



Notes to Figure 6–6:

(1) All disabled blocks and signals are grayed out.
(2) In DDR mode, `tx_inclock` clocks the IOE register. In SDR mode, data is passed directly through the IOE.
(3) In SDR and DDR modes, the data width to the IOE is 1 and 2 bits, respectively.

Programmable V_{OD} and Programmable Pre-Emphasis

Arria V **LVDS** transmitters support programmable differential output voltage (V_{OD}) and programmable pre-emphasis.

Programmable V_{OD}

You can statically adjust the V_{OD} of the differential signal by changing the V_{OD} settings in the Assignment Editor. Figure 6–7 shows the V_{OD} of the differential **LVDS** output.

Figure 6–7. Differential V_{OD}

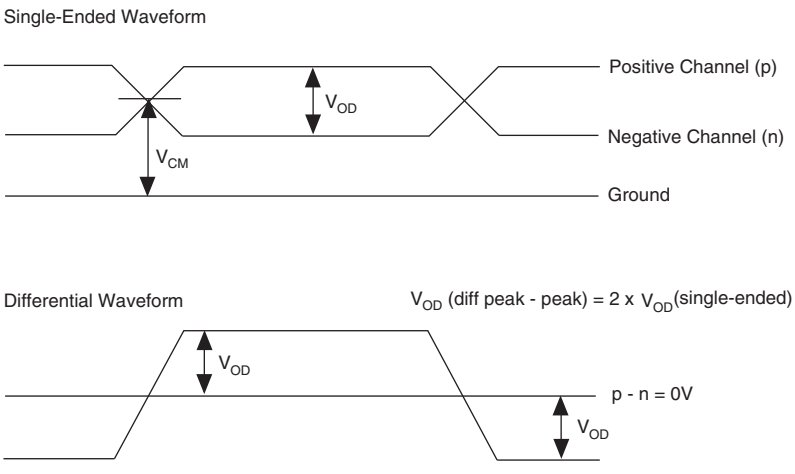


Table 6–2 lists the assignment name for programmable V_{OD} and its possible values in the Quartus II software Assignment Editor.

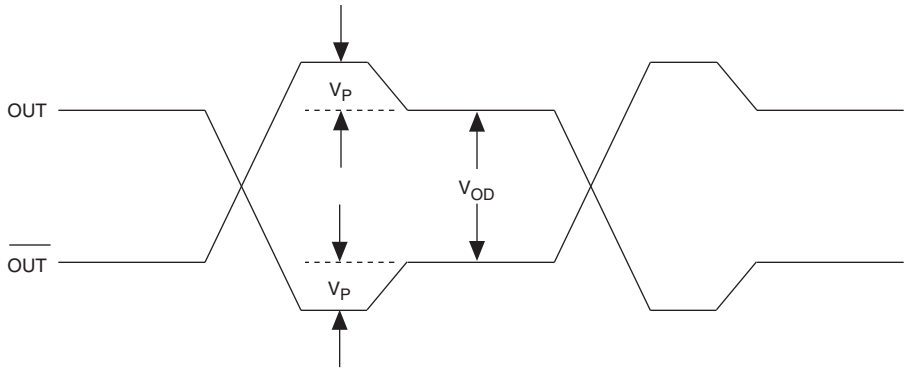
Table 6–2. Quartus II Software Assignment Editor—Programmable V_{OD}

Field	Assignment
To	tx_out
Assignment name	Programmable Differential Output Voltage (V_{OD})
Allowed values	0, 1, 2, 3

Programmable Pre-Emphasis

Pre-emphasis increases the amplitude of the high-frequency component of the output signal, and thus helps to compensate for the frequency-dependent attenuation along the transmission line. Figure 6–8 shows the **LVDS** output with pre-emphasis.

Figure 6–8. Programmable Pre-Emphasis *(1)*



Note to Figure 6–8:

(1) V_P— voltage boost from pre-emphasis. V_{OD}— differential output voltage (peak–peak).

Table 6–3 lists the assignment name for programmable pre-emphasis and its possible values in the Quartus II software Assignment Editor.

Table 6–3. Quartus II Software Assignment Editor—Programmable Pre-Emphasis

Field	Assignment
To	tx_out
Assignment name	Programmable Pre-emphasis
Allowed values	0 (disable) and 1 (enable)

For more information about programmable pre-emphasis, refer to "Programmable Pre-Emphasis" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

Differential Receiver

The receiver has a differential buffer and fractional PLLs that you can share among the transmitter and receiver, a DPA block, a synchronizer, a data realignment block, and a deserializer. The differential buffer can receive **LVDS**, **mini-LVDS**, and **RSDS** signal levels. You can statically set the I/O standard of the receiver pins to **LVDS**, **mini-LVDS**, or **RSDS** in the Quartus II software Assignment Editor.

Figure 6–9 shows the hardware blocks of the Arria V receiver.

The fractional PLL receives the external clock input and generates different phases of the same clock. The DPA block automatically chooses one of the clocks from the fractional PLL and aligns the incoming data on each channel.

The synchronizer circuit is a 1-bit wide by 6-bit deep FIFO buffer that compensates for any phase difference between the DPA clock and the data realignment block. 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 includes shift registers and parallel load registers, and sends a maximum of 10 bits to the internal logic.

The Arria V device family supports three different receiver modes:

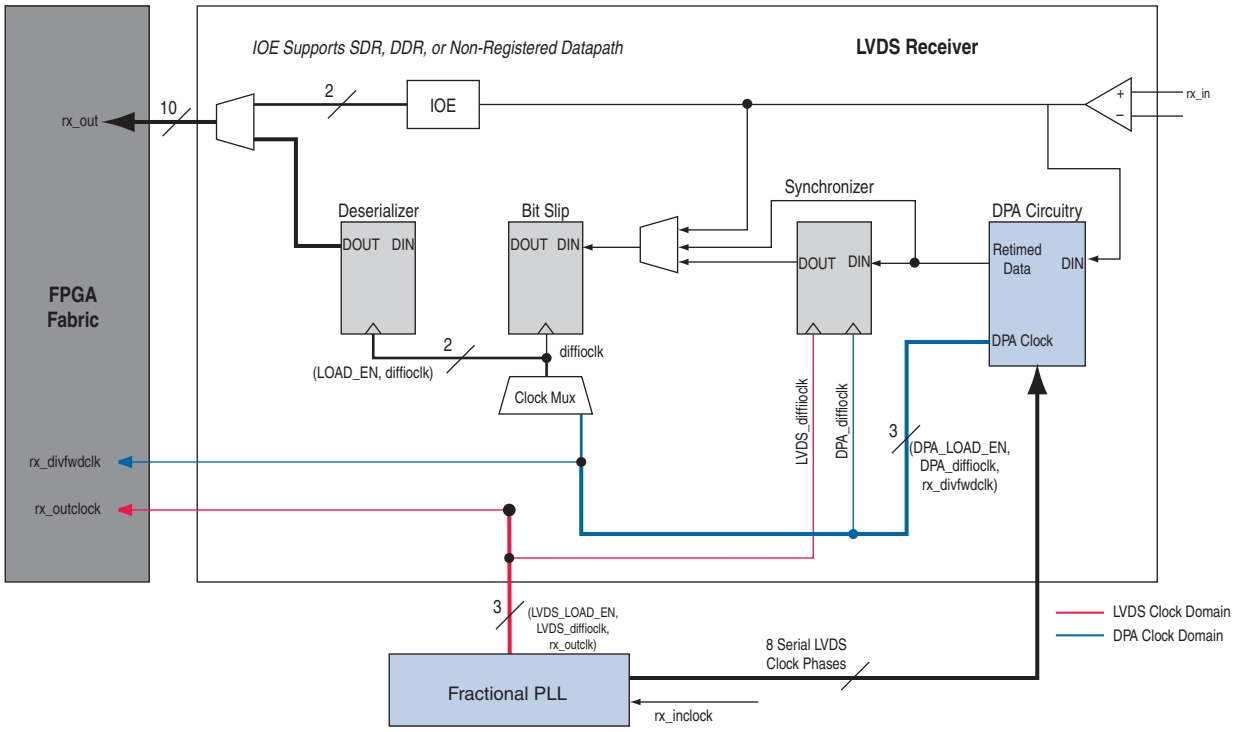
- "Non-DPA Mode" on page 6–15
- "DPA Mode" on page 6–16
- "Soft-CDR Mode" on page 6–17

The physical medium connecting the transmitter and receiver **LVDS** channels may introduce skew between the serial data and the source-synchronous clock. The instantaneous skew between each **LVDS** channel and the clock also varies with the jitter on the data and clock signals as seen by the receiver. The three different modes—non-DPA, DPA, and soft-CDR—provide different options to overcome skew between the source synchronous clock (non-DPA, DPA) /reference clock (soft-CDR) and the serial data.

Only the non-DPA mode requires manual skew adjustment.

Non-DPA mode allows you to statically select the optimal phase between the source synchronous clock and the received serial data to compensate skew. In DPA mode, the DPA circuitry automatically chooses the best phase to compensate for the skew between the source synchronous clock and the received serial data. Soft-CDR mode provides opportunities for synchronous and asynchronous applications for chip-to-chip and short reach board-to-board applications for SGMII protocols.

Figure 6–9. Receiver Block Diagram *(1), (2)*



Notes to Figure 6–9:

(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.

Differential I/O Termination

The Arria V device family provides a 100-Ω, on-chip differential termination option on each differential receiver channel for **LVDS** standards. On-chip termination saves board space by eliminating the need to add external resistors on the board. You can enable on-chip termination in the Quartus II software Assignment Editor.

All I/O pins and dedicated clock input pins support on-chip differential termination.

Figure 6–10 shows device on-chip termination.

Figure 6–10. On-Chip Differential I/O Termination

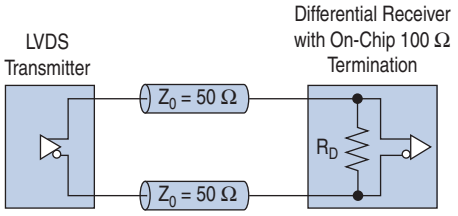


Table 6–4 lists the assignment name for on-chip differential termination in the Quartus II software Assignment Editor.

Table 6–4. Quartus II Software Assignment Editor—On-Chip Differential Termination

Field	Assignment
To	rx_in
Assignment name	Input Termination
Value	Differential

Receiver Hardware Blocks

The differential receiver has the following hardware blocks:

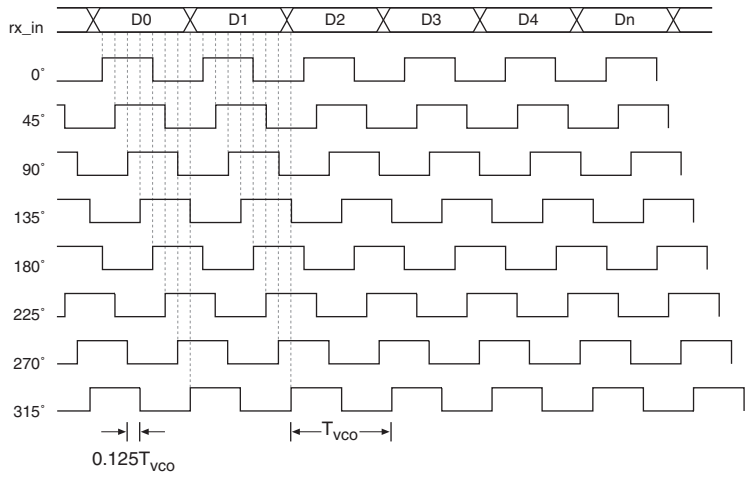
- "DPA Block" on page 6–11
- "Synchronizer" on page 6–12
- "Data Realignment Block (Bit Slip)" on page 6–13
- "Deserializer" on page 6–14

DPA Block

The DPA block takes in high-speed serial data from the differential input buffer and selects one of the eight phases that the fractional PLLs generate to sample the data. The DPA chooses a phase closest to the phase of the serial data. The maximum phase offset between the received data and the selected phase is 1/8 UI, which is the maximum quantization error of the DPA. The eight phases of the clock are equally divided, offering a 45° resolution.

Figure 6–11 shows the possible phase relationships between the DPA clocks and the incoming serial data.

Figure 6–11. DPA Clock Phase to Serial Data Timing Relationship *(1)*



Note to Figure 6–11:

(1) T_{VCO} is defined as the PLL serial clock period.

The DPA block continuously monitors the phase of the incoming serial data and selects a new clock phase if it is required. You can prevent the DPA from selecting a new clock phase by asserting the optional `RX_DPLL_HOLD` port, which is available for each channel.

DPA circuitry does not require a fixed training pattern to lock to the optimum phase out of the eight phases. After reset or power up, the DPA circuitry requires transitions on the received data to lock to the optimum phase. An optional output port, `RX_DPA_LOCKED`, is available to indicate an initial DPA lock condition to the optimum phase after power up or reset. This signal is not deasserted if the DPA selects a new phase out of the eight clock phases to sample the received data. Do not use the `rx_dpa_locked` signal to determine a DPA loss-of-lock condition. Use data checkers such as a cyclic redundancy check (CRC) or diagonal interleaved parity (DIP-4) to validate the data.

An independent reset port, `RX_RESET`, is available to reset the DPA circuitry. You must retrain the DPA circuitry after reset.

The DPA block is bypassed in non-DPA mode.

Synchronizer

The synchronizer is a 1-bit wide and 6-bit deep FIFO buffer that compensates for the phase difference between `DPA_diffioclk`—the optimal clock that the DPA block selects—and the `LVDS_diffioclk` that the fractional PLLs produce. The synchronizer can only compensate for phase differences, not frequency differences, between the data and the receiver's input reference clock.

An optional port, RX_FIFO_RESET, is available to the internal logic to reset the synchronizer. The synchronizer is automatically reset when the DPA first locks to the incoming data. Altera recommends using RX_FIFO_RESET to reset the synchronizer when the data checker indicates corrupted, received data.

The synchronizer circuit is bypassed in non-DPA and soft-CDR mode.

Data Realignment Block (Bit Slip)

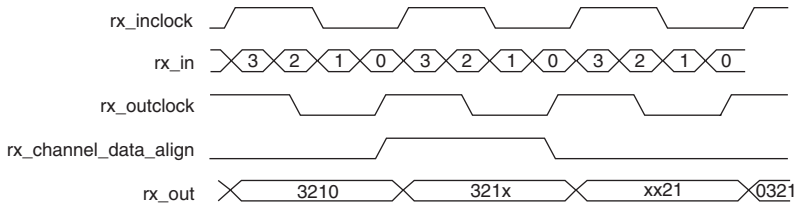
Skew in the transmitted data along with skew added by the link causes channel-to-channel skew on the received serial data streams. If you enable the DPA, the received data is captured with different clock phases on each channel. This difference may cause misalignment of the received data from channel to channel. To compensate for this channel-to-channel skew and establish the correct received word boundary at each channel, each receiver channel has a dedicated data realignment circuit that realigns the data by inserting bit latencies into the serial stream.

An optional RX_CHANNEL_DATA_ALIGN port 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 requirements for the RX_CHANNEL_DATA_ALIGN signal include:

- 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.
- The signal is an edge-triggered signal.
- The valid data is available two parallel clock cycles after the rising edge of RX_CHANNEL_DATA_ALIGN.

Figure 6–12 shows receiver output (RX_OUT) after one bit slip pulse with the deserialization factor set to **4**.

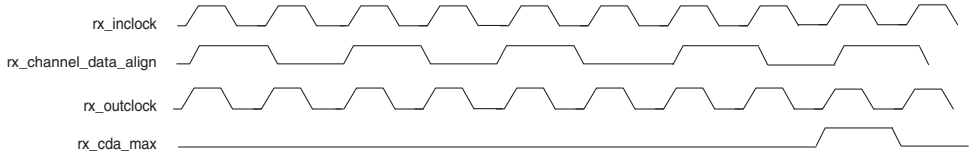
Figure 6–12. Data Realignment Timing



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. Set the programmable bit rollover point 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 MegaWizard Plug-In Manager. An optional status port, RX_CDA_MAX, is available to the FPGA fabric from each channel to indicate the reaching of the preset rollover point.

Figure 6–13 shows a preset value of four bit-times before rollover occurs. The rx_cda_max signal pulses for one rx_outclock cycle to indicate that rollover has occurred.

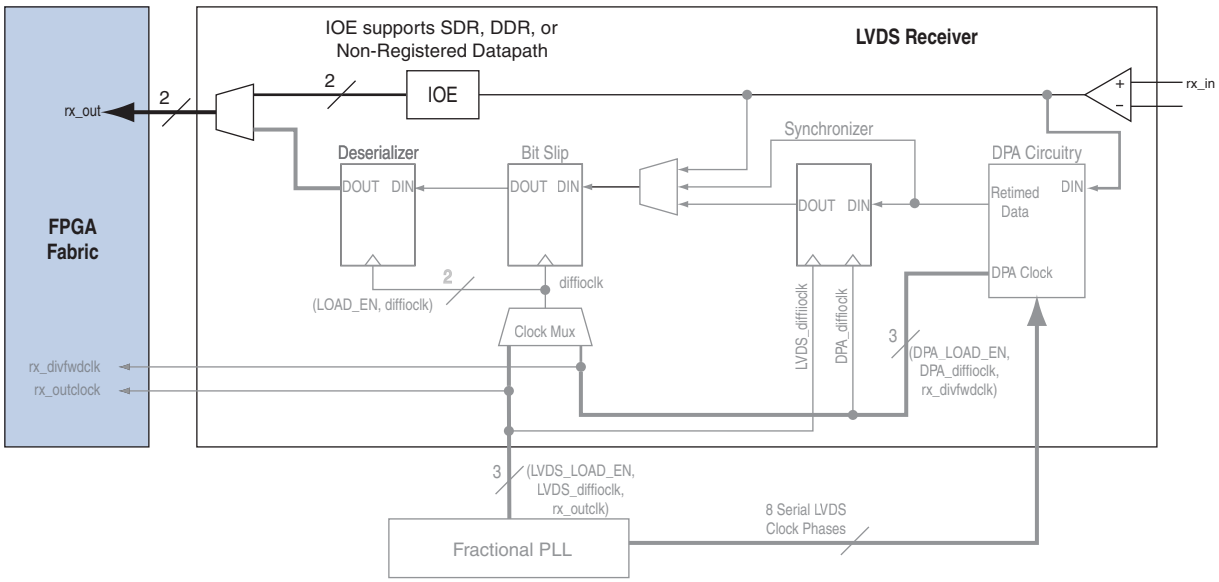
Figure 6–13. Receiver Data Realignment Rollover



Deserializer

You can statically set the deserialization factor to ×3, ×4, ×5, ×6, ×7, ×8, ×9, or ×10 by using the Quartus II software. You can bypass the Arria V deserializer in the Quartus II MegaWizard Plug-In Manager to support DDR (×2) or SDR (×1) operations, as shown Figure 6–14. You cannot use the DPA and data realignment circuit when you bypass the deserializer. The IOE contains two data input registers that can operate in DDR or SDR mode.

Figure 6–14. Deserializer Bypass *(1), (2), (3)*



Notes to Figure 6–14:

(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.

Receiver Data Path Modes

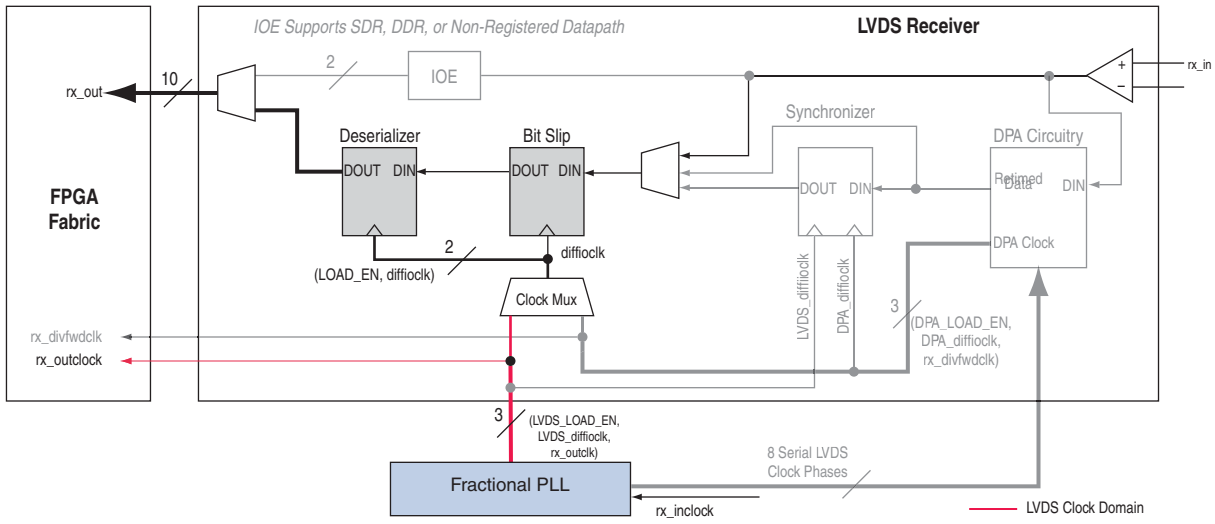
The Arria V device family supports three receiver datapath modes: non-DPA mode, DPA mode, and soft-CDR mode.

Non-DPA Mode

Figure 6–15 shows the non-DPA datapath block diagram. The non-DPA mode disables the DPA and synchronizer blocks. Input serial data is registered at the rising or falling edge of the serial `LVDS_diffioclk` clock that is produced by the left and right PLLs.

You can select the rising or falling edge option with the Quartus II MegaWizard Plug-In Manager. The `LVDS_diffioclk` clock that is generated by the left and right PLLs clocks the data realignment and deserializer blocks.

Figure 6–15. Receiver Data Path in Non-DPA Mode *(1), (2), (3)*



Notes to Figure 6–15:

(1) All disabled blocks and signals are grayed out.

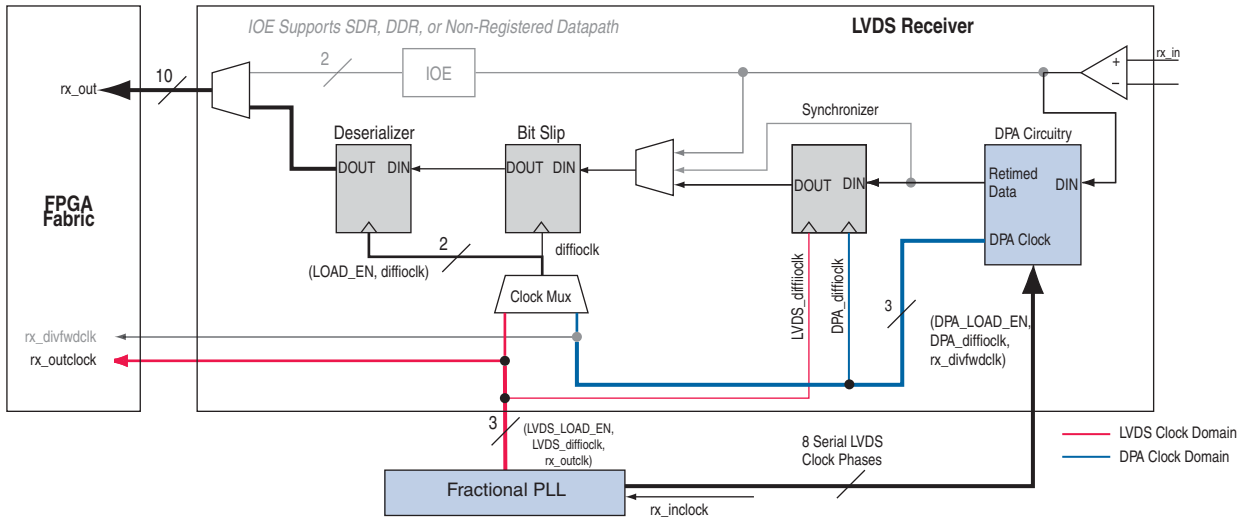
(2) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.

(3) The `rx_out` port has a maximum data width of 10 bits.

DPA Mode

Figure 6–16 shows the DPA mode datapath, where all the hardware blocks mentioned in "Receiver Hardware Blocks" on page 6–11 are active. The DPA block chooses the best possible clock (DPA_diffioclk) from the eight fast clocks that the fractional PLL sent. This serial DPA_diffioclk clock is used for writing the serial data into the synchronizer. A serial LVDS_diffioclk clock is used for reading the serial data from the synchronizer. The same LVDS_diffioclk clock is used in data realignment and deserializer blocks.

Figure 6–16. Receiver Datapath in DPA Mode *(1), (2), (3)*



Notes to Figure 6–16:

(1) All disabled blocks and signals are grayed out.

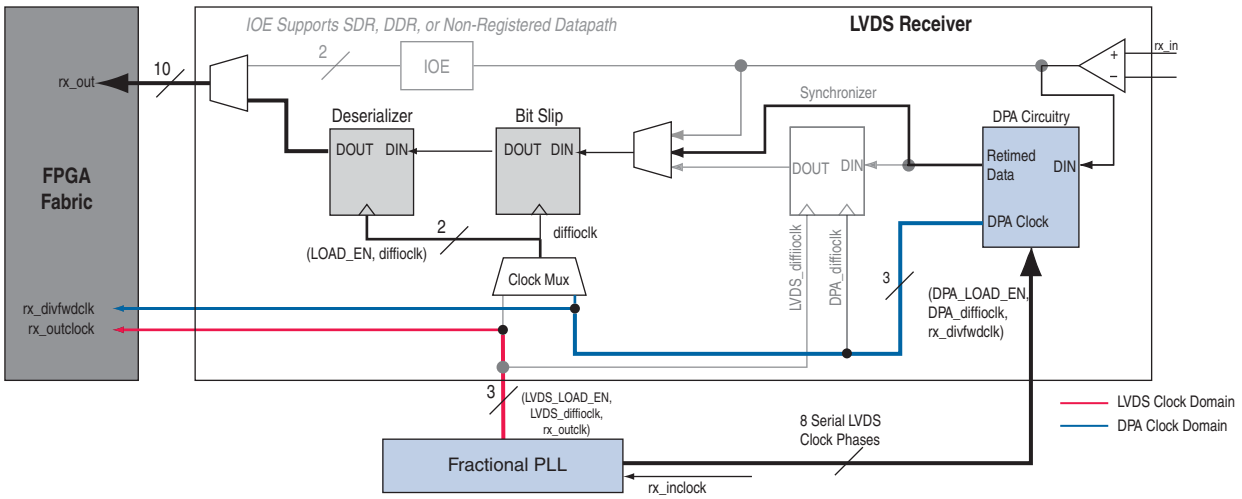
(2) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.

(3) The rx_out port has a maximum data width of 10 bits.

Soft-CDR Mode

The Arria V **LVDS** channel offers the soft-CDR mode to support the GbE and SGMII protocols. A receiver PLL uses the local clock source for reference. Figure 6–17 shows the soft-CDR mode datapath.

Figure 6–17. Receiver Datapath in Soft-CDR Mode *(1), (2), (3)*



Notes to Figure 6–17:

(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.

In soft-CDR mode, the synchronizer block is inactive. The DPA circuitry selects an optimal DPA clock phase to sample the data. Use the selected DPA clock for bit-slip operation and deserialization. The DPA block also forwards the selected DPA clock, divided by the deserialization factor called `rx_divfwdclk`, to the FPGA fabric, along with the deserialized data. This clock signal is put on the periphery clock (PCLK) network.

When you use the soft-CDR mode, do not assert the `rx_reset` port after you assert the `rx_dpa_lock`. The DPA continuously chooses new phase taps from the PLL to track parts per million (PPM) differences between the reference clock and incoming data.

For more information about PCLK networks, refer to the *Clock Networks and PLLs in Arria V Devices* chapter.

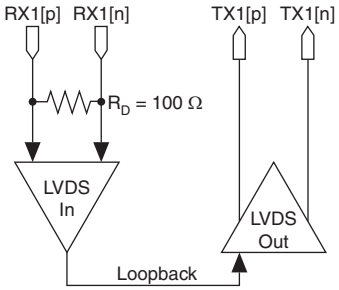
You can use every **LVDS** channel in soft-CDR mode and drive the FPGA fabric using the PCLK network in the Arria V device family. The `rx_dpa_locked` signal is not valid in soft-CDR mode because the DPA continuously changes its phase to track PPM differences between the upstream transmitter and the local receiver input reference clocks. The parallel clock, `rx_outclock`, generated by the left and right PLLs, is also forwarded to the FPGA fabric.

For more information, refer to "Differential Transmitter" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

LVDS Direct Loopback Mode

The Arria V device family supports direct loopback mode for the **LVDS** driver and receiver pairs only within the same **LVDS** module. Figure 6–18 shows the true **LVDS** input and output buffer from an I/O pair from the same module. **LVDS** direct loopback mode allows you to verify the **LVDS** driver and receiver pair by checking the incoming **LVDS** data from the true **LVDS** input buffer into the true **LVDS** output buffer.

Figure 6–18. LVDS Direct Loopback Path *(1)*



Note to Figure 6–18:

(1) The R_D value is pending characterization.

You can turn the **LVDS** direct loopback mode on or off with the assignment editor in the Quartus II software.

This option is available only for true differential I/O standards.

You can apply the option on the **LVDS** output pair that is already being used in the design. Turning the **LVDS** direct loopback mode option to **On** overrides the connection from the core with the signal from the true differential input buffer in the same I/O module. You can disable this option after verifying the **LVDS** driver receiver pair by turning the **LVDS** direct loopback mode option to **Off** and recompiling your design.

Fractional PLLs and Arria V Clocking

The Arria V device family supports fractional PLLs on each side of the device. Figure 6–1 on page 6–2 and Figure 6–2 on page 6–2 show the location of the fractional PLLs supported for the high-speed differential I/O receiver and transmitter channels to generate the parallel clocks (`rx_outclock` and `tx_outclock`) and high-speed clocks (`diffioclk`).

The center or corner fractional PLLs can drive the **LVDS** receiver and driver channels. The clock tree network cannot cross over to different I/O regions. For example, the top left corner fractional PLL cannot cross over to drive the LVDS receiver and driver channels on the top right I/O bank.

For more information about the fractional PLL clocking restrictions, refer to "Differential Pin Placement Guidelines" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

The MegaWizard Plug-In Manager provides an option for implementing the **LVDS** interface with the external PLL mode. With this option enabled you can control the PLL settings, such as dynamically reconfiguring the PLL to support different data rates, dynamic phase shift, and other settings. You also must instantiate the appropriate megafunction to generate the various clock and load enable signals.

For more information about the external PLL mode, refer to "Generating Clock Signals for LVDS Interface" in the *LVDS SERDES Transmitter/Receiver (ALTLVDS_RX and ALTLVDS_TX) Megafunction User Guide*.

Source-Synchronous Timing Budget

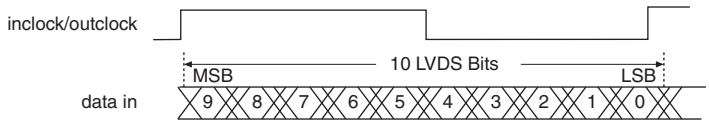
This section describes the timing budget, waveforms, and specifications for source-synchronous signaling in the Arria V device family. The **LVDS** I/O standards enable the transmission of data at a high speed. This high transmission rate of data results in better overall system performance. To take advantage of fast system performance, you must understand how to analyze timing for these high-speed signals. Timing analysis for the differential block is different from traditional synchronous timing analysis techniques.

The basis of the source synchronous timing analysis is the skew between the data and the clock signals instead of the clock-to-output setup times. High-speed differential data transmission requires the use of timing parameters provided by IC vendors and is strongly influenced by board skew, cable skew, and clock jitter. This section defines the source-synchronous differential data orientation timing parameters, the timing budget definitions for the Arria V device family, and how to use these timing parameters to determine a design's maximum performance.

Differential Data Orientation

There is a set relationship between an external clock and the incoming data. For operations at 1 Gbps and a serialization factor of 10, the external clock is multiplied by 10. You can set 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 6–19 shows the data bit orientation of the x10 mode.

Figure 6–19. Bit Orientation in the Quartus II Software

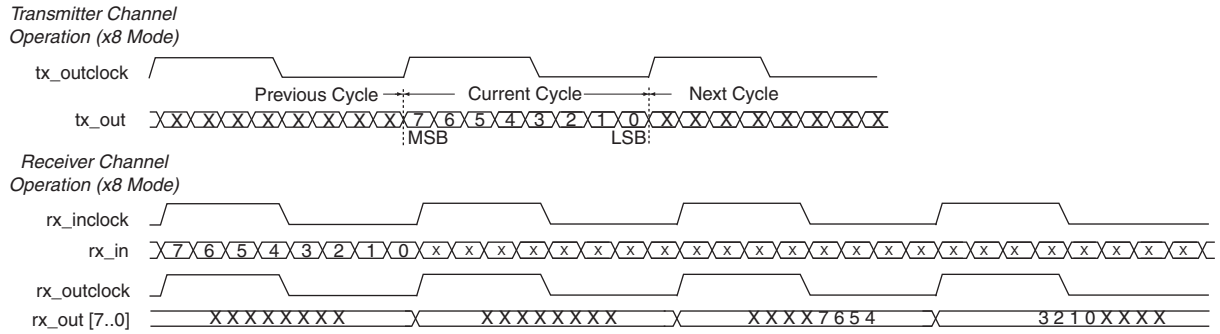


Differential I/O Bit Position

Data synchronization is necessary for successful data transmission at high frequencies. Figure 6–20 shows the data bit orientation for a channel operation and is based on the following conditions:

- The serialization factor is equal to the clock multiplication factor.
- The phase alignment uses edge alignment.
- The operation is implemented in hard SERDES.

Figure 6–20. Bit-Order and Word Boundary for One Differential Channel *(1)*



Note to Figure 6–20:

(1) These waveforms are only functional waveforms and do not convey timing information.

For other serialization factors, use the Quartus II software tools to find the bit position within the word.

Table 6–5 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 6–5. 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

The receiver skew margin calculation uses the transmitter channel-to-channel skew (TCCS)—an important parameter based on the Arria V transmitter in a source-synchronous differential interface. For more information, refer to "Receiver Skew Margin for Non-DPA Mode".

TCCS is the difference between the fastest and slowest data output transitions, including the T_{CO} 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 in the TimeQuest Timing Analyzer, or from the *Device Datasheet for Arria V Devices* chapter.

Receiver Skew Margin for Non-DPA Mode

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 the receiver skew margin (RSKM).Innon-DPA LVDS mode, use RSKM, TCCS, and sampling window (SW) specifications for high-speed source-synchronous differential signals in the receiver data path.

For more information about the RSKM equation and calculation, refer to "Receiver Skew Margin for Non-DPA Mode" in the *LVDS SERDES Transmitter/Receiver (ALTLVDS_RX and ALTLVDS_TX) Megafunction User Guide*.

For **LVDS** receivers, the Quartus II software provides an RSKM report showing the SW, TUI, and RSKM values fornon-DPA LVDS mode. You can generate the RSKM report by executing the `report_RSKM` command in the TimeQuest Timing Analyzer. You can find the RSKM report in the Quartus II compilation report in the TimeQuest Timing Analyzer section.

To obtain the RSKM value, assign the input delay to the **LVDS** receiver through the TimeQuest Timing Analyzer constraints menu. The input delay is determined according to the data arrival time at the **LVDS** receiver port, with respect to the reference clock. When setting the input delay in the settings parameters for the **Set Input Delay** option, the clock name must reference the source synchronous clock that feeds the **LVDS** receiver.

If you do not set any input delay in the TimeQuest Timing Analyzer, the receiver channel-to-channel skew (RCCS) defaults to zero. You can also directly set the input delay in a Synopsys Design Constraint file (**.sdc)** using the `set_input_delay` command.

For more information about **.sdc** commands and the TimeQuest Timing Analyzer, refer to the *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Development Software Handbook*.

Differential Pin Placement Guidelines

When DPA-enabled or DPA-disabled differential channels in the differential banks are used, you must adhere to the differential pin placement guidelines to ensure the proper high-speed operation.

For more information, refer to "Differential Pin Placement Guidelines" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

Document Revision History

Table 6–6 lists the revision history for this chapter.

Table 6–6. Document Revision History

Date	Version	Changes
November 2011	1.1	■ Updated Table 6–1. ■ Restructured chapter.
May 2011	1.0	Initial release.

7. External Memory Interfaces in Arria V Devices

AV52007-1.1

This chapter describes external memory interfaces available with Arria® V devices, as well as the silicon capabilities of the devices to support external memory interfaces.

The design of the I/Os provides high-performance support for existing and emerging external double data rate (DDR) memory standards.

Memory interfaces use Arria V device features such as delay-locked loops (DLLs), dynamic on-chip termination (OCT) control, and I/O features such as programmable I/O delay chains, read FIFO blocks, slew rate adjustment, and programmable drive strength.

This chapter contains the following sections:

For more information, refer to External Memory Interfaces section of the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

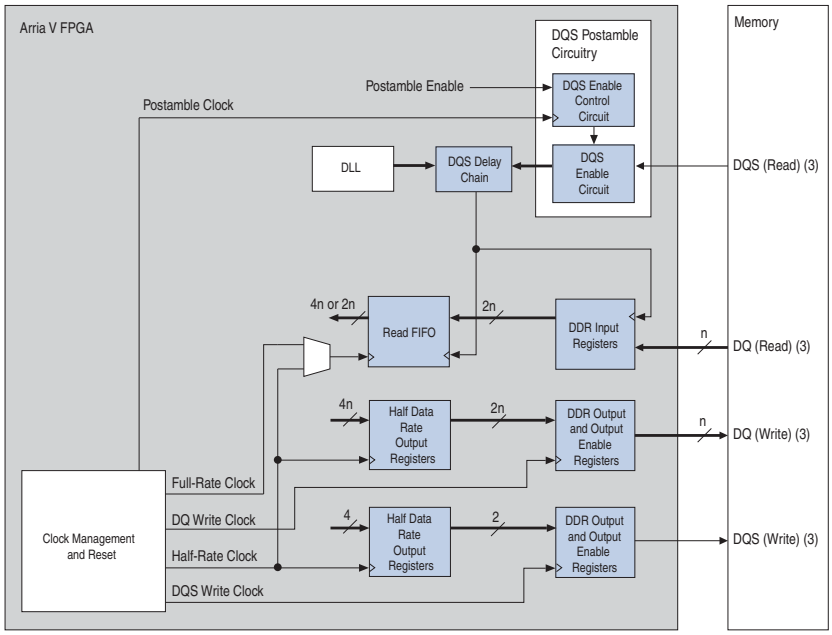
You can use Altera's External Memory Interface Spec Estimator tool to estimate the external memory system performance specifications. For more information, refer to the External Memory Interface Spec Estimator page of the Altera website.

For more information about board design guidelines, timing analysis, simulation, and debugging information, refer to *External Memory Interface Handbook*.

For more information about the UniPHY intelectual property (IP), refer to *Volume 3: Implementing Altera Memory Interface IP* of the *External Memory Interface Handbook*.

Figure 7–1 shows an overview of the memory interface datapath that uses the Arria V I/O element (IOE) features.

Figure 7–1. External Memory Interface Datapath Overview for Arria V Devices *(1), (2)*



Notes to Figure 7–1:

(1) You can bypass each register block.

(2) The blocks for each memory interface may differ slightly. The shaded blocks are part of the Arria V 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 Interface Pin Support

This section describes the Arria V I/O pins that can be used for external memory interface. This section also provides the number of DQ groups available on each device side for all Arria V devices.

LPDDR, LPDDR2, DDR2, and DDR3 SDRAM, and RLDRAM II devices use the CK and CK# signals to capture the address and command signals. Generate these signals to mimic the write-data strobe with Arria V DDR I/O (DDRIO) registers to ensure that you meet the timing relationships between the CK/CK# and DQS signals (t_{DQSS}, t_{DSS}, and t_{DSH} in DDR3 and DDR2 SDRAM devices or t_{CKDK} in RLDRAM II devices).

Arria V devices offer differential input buffers for differential read-data strobe and clock operations. In the Arria V pin tables, the DQS and DQSn pins denote the differential data strobe/clock pin pairs, while the CQ and CQn pins denote the complementary echo clock signals.

Table 7–1 lists pin support per DQ/DQS bus mode, including the DQS/CQ and DQSn/CQn pin pair.

Table 7–1. DQ/DQS Bus Mode Pins for Arria V Devices

Mode	DQSn Support	Parity or Data Mask (Optional)	QVLD (Optional) (1)	Typical Number of Data Pins per Group	Maximum Number of Data Pins per Group (2)
×4/×8/×9 (3)	Yes	Yes	Yes	4, 8, or 9	11
×16/×18 (4)	Yes	Yes	Yes	16 or 18	23
×32/×36 (5)	Yes	Yes	Yes	32 or 36	47

Notes to Table 7–1:

(1) The QVLD pin is not used in the UniPHY megafunction.

(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. If you use differential or complementary DQS signaling, the maximum number of data pins 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 and DDR2 interfaces, the number of pins is further reduced for an interface larger than ×8 because you require one DQS pin for each ×8/×9 group to form the ×16/×18 and ×32/×36 groups.

(3) The ×4 mode uses ×8/×9 groups in Arria V devices.

(4) Two ×8 DQ/DQS groups are stitched to create a ×16/×18 group; so there are a total of 24 pins in this group.

(5) Four ×8 DQ/DQS groups are stitched to create a ×32/×36 group, so there are a total of 48 pins in this group.

Table 7–2 lists the number of DQ/DQS groups available per side in each Arria V device.

Table 7–2. Number of DQ/DQS Groups in Arria V Devices per Side (1) (Part 1 of 2)

Device	Package	Side	×8/×9	×16/×18	×32/×36
5AGXA1 5AGXA3	672-pin FineLine BGA, Flip Chip	Right	6	2	0
		Top/Bottom	7	3	0
5AGXA1 5AGXA3	896-pin FineLine BGA, Flip Chip	Right	6	2	0
		Top/Bottom	12	6	2
5AGXA5 5AGXA7	672-pin FineLine BGA, Flip Chip	Right	0	0	0
		Top/Bottom	8	3	0
5AGXA5 5AGXA7 5AGXB1 5AGXB3 5AGTD3	896-pin FineLine BGA, Flip Chip	Right	0	0	0
		Top/Bottom	12	5	1
5AGXA5 5AGXA7 5AGXB1 5AGXB3 5AGTD3	1152-pin FineLine BGA, Flip Chip	Right	0	0	0
		Top/Bottom	17	8	2

Table 7–2. Number of DQ/DQS Groups in Arria V Devices per Side (1) (Part 2 of 2)

Device	Package	Side	×8/×9	×16/×18	×32/×36
5AGXB1 5AGXB3 5AGTD3	1517-pin FineLine BGA, Flip Chip	Right	0	0	0
		Top/Bottom	22	10	4
5AGXB5 5AGXB7 5AGTD7	1152-pin FineLine BGA, Flip Chip	Right	0	0	0
		Top/Bottom	16	7	2
5AGXB5 5AGXB7 5AGTD7	1517-pin FineLine BGA, Flip Chip	Right	0	0	0
		Top/Bottom	21	9	3

Note to Table 7–2:

(1) These numbers are preliminary until the devices are available.

The pin table lists the parity, DM, BWSn, NWSn, ECC, and QVLD pins as DQ pins.

Design Considerations for External Memory Interfaces

Memory interface circuitry is available in every I/O bank that does not support transceivers. All the memory interface pins support the I/O standards required for the supported memory interfaces.

To ensure the success of your design, consider the following aspects:

- Arria V devices support DQ and DQS signals with DQ bus modes of ×4/×8/×9, ×16/×18, or ×32/×36. If you do not use any of these pins for memory interfacing, you can use these pins as user I/Os. In addition, you can use the DQSn or CQn pins that are not used for clocking as DQ pins.
- Memory clock pins in Arria V devices are generated with double data rate input/output (DDRIO) registers.
- You can use some of the DQ pins as RZQ pins. However, when you use these pins as RZQ pins, you cannot use them as DQ pins in an external memory interface.
- You must manually assign DQ and DQS pins for ×8, ×16/×18, or ×32/×36 DQ/DQS groups whose members are used as RZQ pins. The Quartus® II software might not be able to place DQ and DQS pins without manual pin assignments, which results in a “no-fit” error.
- Use differential DQS signaling for DDR2 SDRAM interfaces running at 333 MHz and higher.

External Memory Interface Features

This section describes the following Arria V device features that are used in external memory interfaces:

- "DQS Phase-Shift Circuitry" on page 7–6
- "PHY Clock (PHYCLK) Networks" on page 7–9
- "DQS Logic Block" on page 7–11
- "Dynamic OCT Control" on page 7–13
- "IOE Registers" on page 7–13
- "Delay Chain" on page 7–16
- "Hard Memory Controllers" on page 7–17

For more information about the UniPHY megafunction, refer to *Volume 3: Implementing Altera Memory Interface IP* of the *External Memory Interface Handbook*.

DQS Phase-Shift Circuitry

The Arria V DLL provides phase shift to the DQS/CQ pins on read transactions if the DQS/CQ pins are acting as input clocks or strobes to the FPGA. Figure 7–2 and Figure 7–3 show how the DLLs are connected to the DQS/CQ pins in the device, where memory interfaces are supported on the right side of the Arria V device.

Figure 7–2. DQS/CQ Pins and DLLs in 5AGXA1 and 5AGXA3 Devices

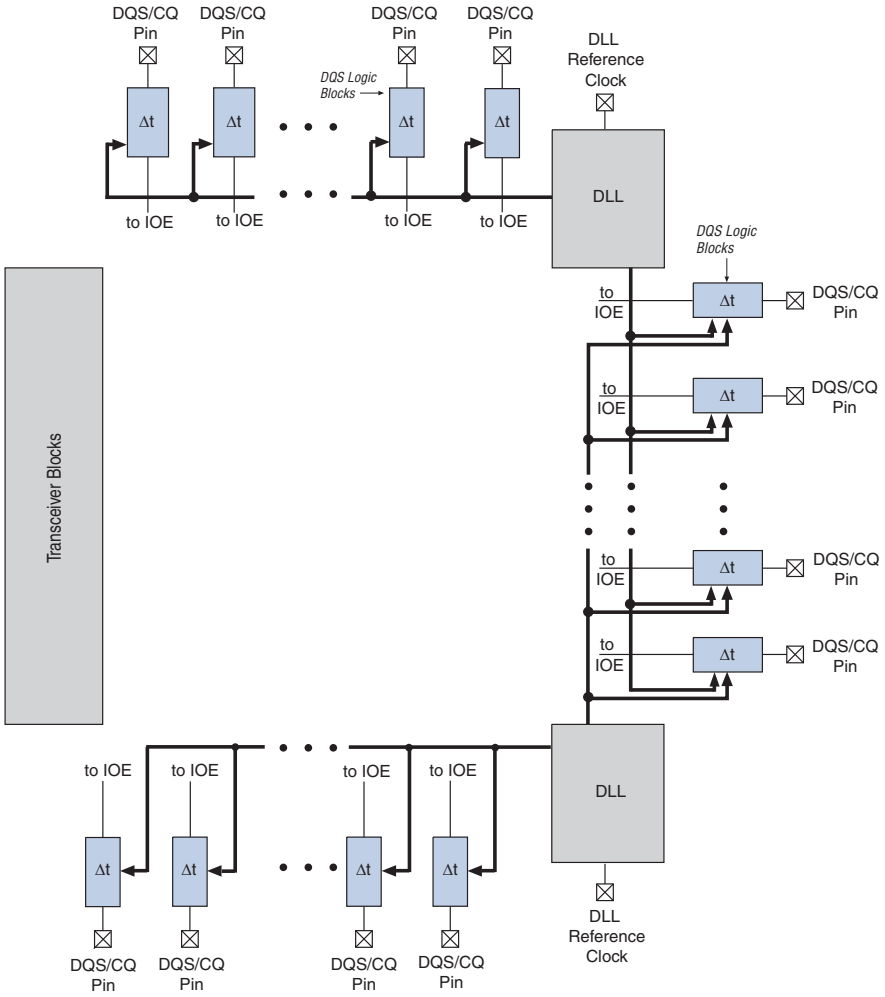
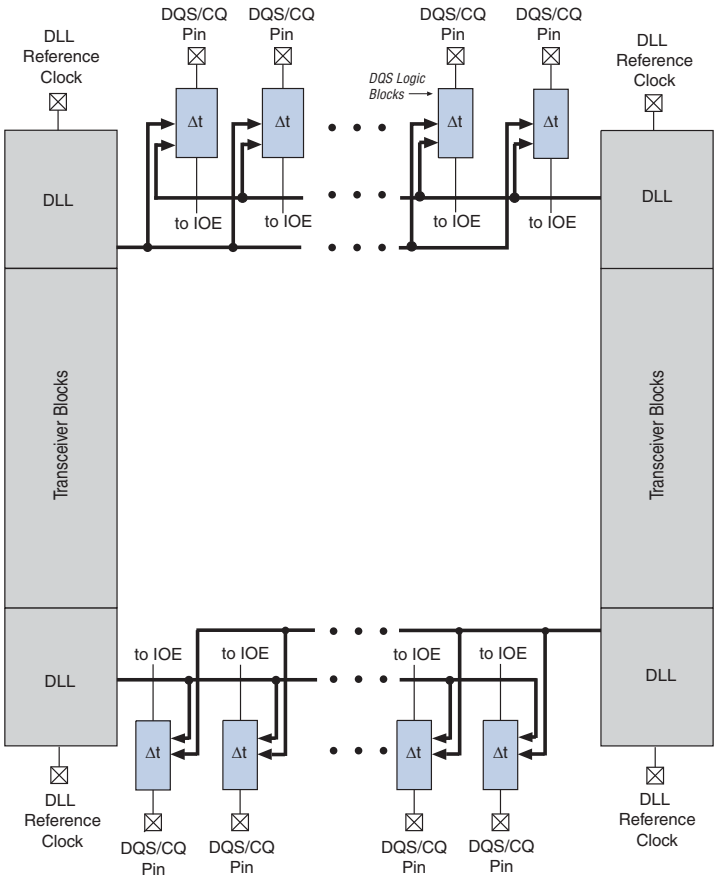


Figure 7–3. DQS/CQ Pins and DLLs in 5AGXB1, 5AGXB3, 5AGXA5, 5AGXA7,5AGXB5, 5AGXB7, 5AGTD3, and 5AGTD7 Devices



Delay-Locked Loop

There are a maximum of four DLLs, located in each corner, in some Arria V devices. These four DLLs support a maximum of four unique frequencies, with each DLL running at one frequency.

The DLLs can access the two adjacent sides from its location in the device. 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.

I/O banks between two DLLs have the flexibility to create multiple frequencies and multiple-type interfaces. These banks can use settings from either or both adjacent DLLs. For example, DQS1R can get its phase-shift settings from DLL_TR, while DQS2R can get its phase-shift settings from DLL_BR. The reference clock for each DLL may come from the PLL output clocks or clock input pins.

The DLL can shift the incoming DQS signals by 0°, 45°, or 90°. Depending on the number of DQS delay chains used, the shifted DQS signal is then used as the clock for the DQ IOE input registers.

All DQS/CQ 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 45° phase shift on DQS2T, referenced from a 200-MHz clock.

However, 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 45° (up to 90°).

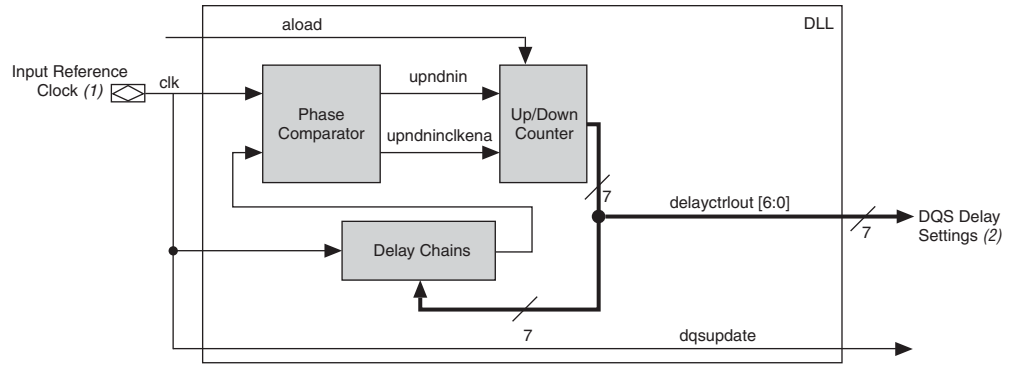
The 7-bit DQS delay settings from the DLL vary with PVT to implement the phase-shift delay.

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 if a 0° shift is implemented. You can feed the DQS delay settings to the DQS logic block and logic array.

The shifted DQS/CQ signal goes to the DQS bus to clock the IOE input registers of the DQ pins. The signal can also go into the logic array for resynchronization if you are not using IOE read FIFO for resynchronization.

Figure 7–14 shows a simple block diagram of the DLL. The input reference clock goes into the DLL to a chain of up to eight 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-code counter. This signal increments or decrements a 7-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 7–4. Simplified Diagram of the DLL



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 2,560 clock cycles for the DLL to lock before you can capture the data properly.

The DLL phase comparator requires 2,560 clock cycles to lock and calculate the correct input clock period. Altera recommends that you do not send data during these clock cycles because there is no guarantee that the data will be captured properly until this lock period has elapsed.

For more information about the DLL, refer to "Delay-Locked Loop" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

PHY Clock (PHYCLK) Networks

The PHYCLK network is a dedicated high-speed, low-skew balanced clock tree designed for high-performance external memory interface. The top and bottom side of the Arria V devices have up to four PHYCLK networks respectively. There are two PHYCLK networks on the left and right side I/O banks. Each PHYCLK network spans across one I/O bank and is driven by one of the PLLs located adjacent to the I/O bank.

The PHYCLK networks can be used to drive I/O sub-banks in each I/O bank. Each I/O sub-bank can be driven by only one PHYCLK network. As such, all I/O pins in an I/O sub-bank are driven by the same PHYCLK network. The UniPHY intellectual property (IP) for Arria V devices uses the PHYCLK network for better performance. Figure 7–5 and Figure 7–6 show the PHYCLK networks available in the Arria V devices.

Figure 7–5. PHYCLK Networks in 5AGXA1 and 5AGXA3 Device

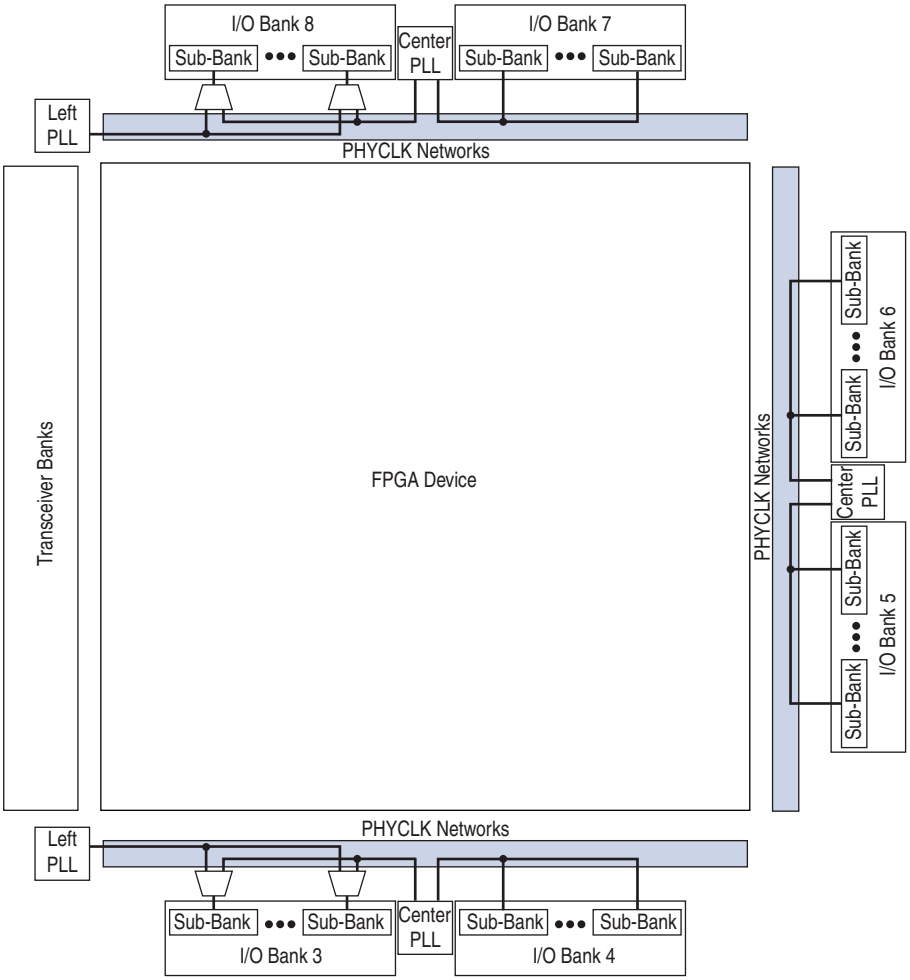
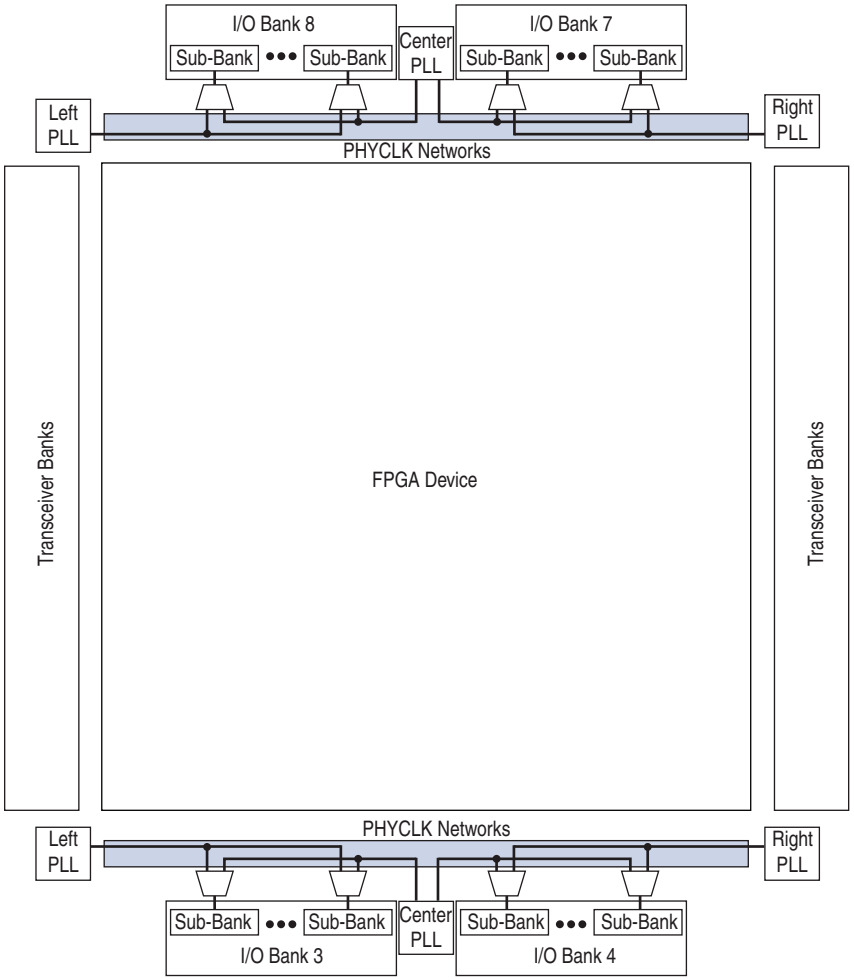


Figure 7–6. PHYCLK Networks in 5AGXB1, 5AGXB3, 5AGXA5, 5AGXA7, 5AGXB5, and 5AGXB7 Devices



For PHYCLK networks using either hard or soft memory controllers, two interfaces can share an I/O sub-bank (for example, sub-bank 4A) for pin placement if the two interfaces share a PLL. These two interfaces must use the same memory protocol (for example, DDR3), frequency, controller rate (for example, half rate), and phase requirements (for example, additional core-to-periphery clock phase of 90°).

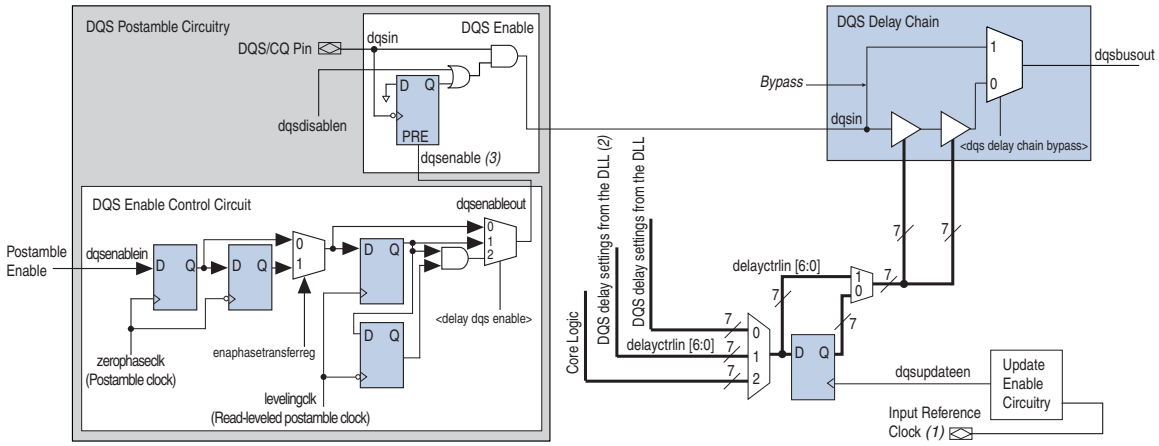
Two interfaces can share an I/O bank (for example, I/O bank 4) for pin placement, regardless of whether they share a PLL, if there are two PLLs that feed that I/O bank. If there is only one PLL that feeds into an I/O bank, two interfaces can share the I/O bank (for example, I/O bank 5) if they share the PLL.

☞ PHYCLK networks support interface at the same side of I/O banks only.

DQS Logic Block

Each DQS/CQ pin is connected to a separate DQS logic block, which consists of the DQS delay chains, update enable circuitry, and DQS postamble circuitry, as shown in Figure 7–7.

Figure 7–7. DQS Logic Block in an Arria V Device



Notes to Figure 7–7:

(1) The input reference clock for the DQS phase-shift circuitry can come from a PLL output clock or an input clock pin.
(2) Only applicable if the DQS delay settings come from a side with two DLLs.
(3) The `dqsenable` signal can also come from the Arria V FPGA fabric.

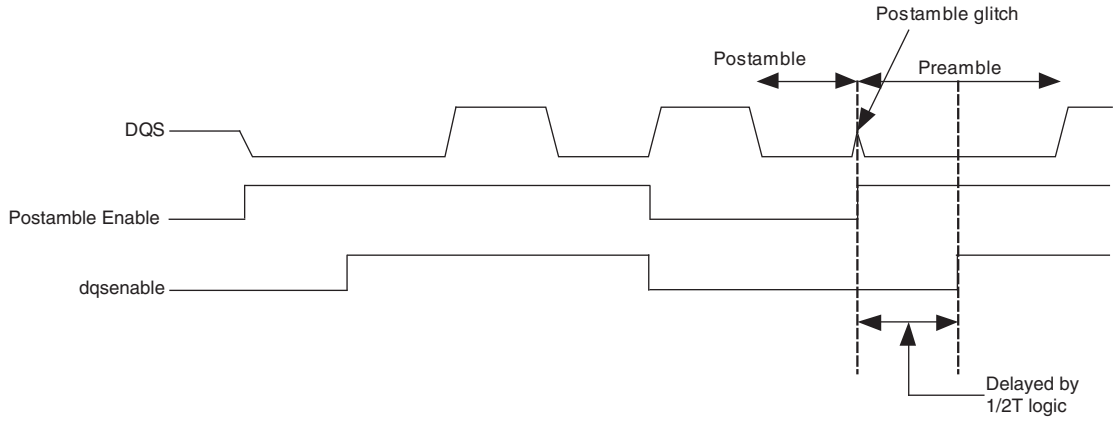
DQS Postamble Circuitry

Arria V devices have dedicated postamble registers that you can control to ground the shifted DQS signal that is used to clock the DQ input registers at the end of a read operation. This function ensures that any glitches on the DQS input signal during the end of a read operation and occurring while DQS is in a postamble state do not affect the DQ IOE registers.

In addition to the dedicated postamble register, Arria V devices also have an half data rate (HDR) block in its 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. There is an AND gate after the postamble register outputs to avoid postamble glitches from a previous read burst on a non-consecutive read burst. This scheme allows half-a-clock cycle latency for dqsenable assertion and zero latency for dqsenable deassertion, as shown in Figure 7–8.

Figure 7–8. Avoiding Glitch on a Non-Consecutive Read Burst Waveform



DQS Delay Chain

The number of delay chains required is transparent because the UniPHY IP automatically sets it when you choose the operating frequency.

The delay elements in the DQS logic block have the same characteristics as the delay elements in the DLL. When you do not use the DLL to control the DQS delay chains, you can input your own Gray-coded 7-bit settings using the `delayctrlin[6..0]` signals available in the UniPHY IP.

The UniPHY IP can also dynamically choose the number of DQS delay chains that are required for the system. The amount of delay is equal to the sum of the intrinsic delay of the delay element and the product of the number of delay steps and the value of the delay steps.

Update Enable Circuitry

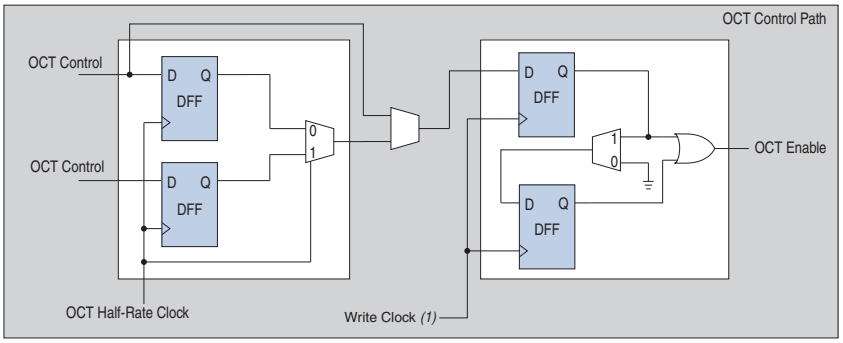
Both the DQS delay settings and the phase-offset settings pass through a register before going into the DQS delay chains. The registers are controlled by the update enable circuitry to allow enough time for any changes in the DQS delay setting bits to arrive at all the delay elements, which allows them to be adjusted at the same time.

For more information about the DQS delay chain, update enable circuitry, and DQS postamble circuitry, refer to the "DQS Logic Block" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

Dynamic OCT Control

Figure 7–9 shows the dynamic OCT control block.

Figure 7–9. Dynamic OCT Control Block for Arria V Devices



Note to Figure 7–9:

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

The control block includes all the registers that are required to dynamically turn on-chip parallel termination (R_T OCT) on during a read and turn R_T OCT off during a write.

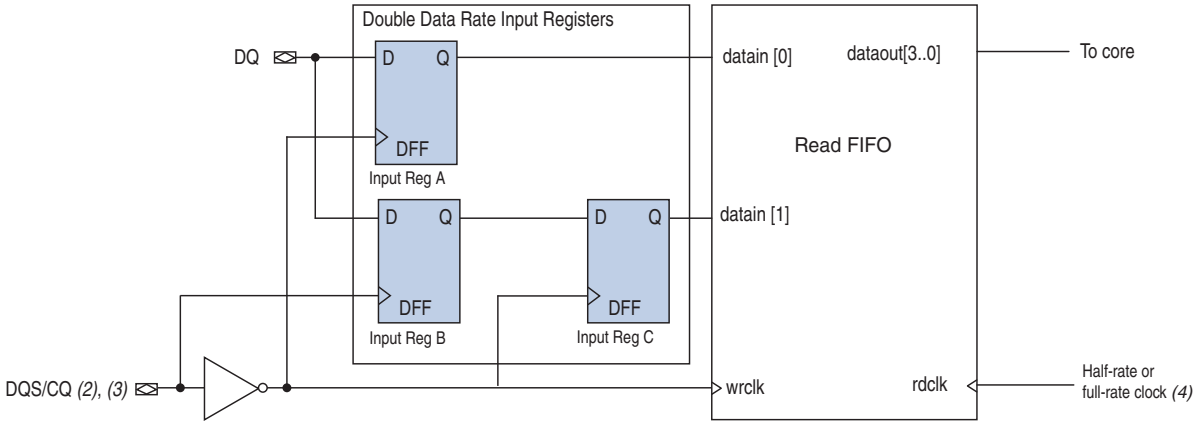
For more information about dynamic OCT control, refer to the *I/O Features in Arria V Devices* chapter.

IOE Registers

The IOE registers are expanded to allow source-synchronous systems to have faster register-to-FIFO transfers and resynchronization. All top, bottom, and right IOEs have the same capability.

Figure 7–10 shows the registers available in the Arria V input path. The input path consists of the DDR input registers and the read FIFO block. You can bypass each block of the input path.

Figure 7–10. IOE Input Registers for Arria V Devices *(1)*



Notes to Figure 7–10:

(1) You can bypass the register or Read FIFO block in this path.

(2) The input clock can be from the DQS logic block or from a global clock line.

(3) The DQS and DQSn signals must be inverted for DDR3 and DDR2 SDRAM interfaces. When using Altera's memory interface IPs, the DQS and DQSn signals are automatically inverted.

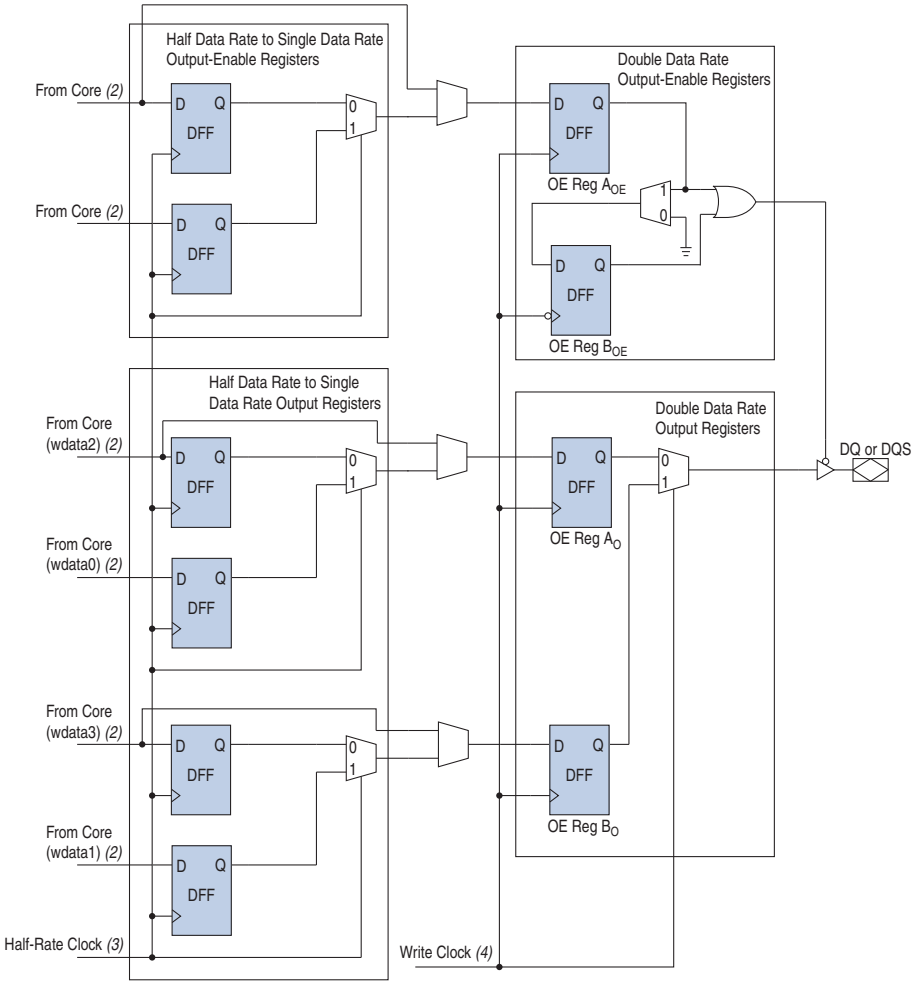
(4) This half-rate or full-rate read clock comes from a PLL through the clock network.

There are three registers in the DDR input registers block. Registers A and B capture data on the positive and negative edges of the clock while register C aligns the captured data. Register C that aligns the captured data uses the same clock as Register A.

The read FIFO block resynchronizes the data to the system clock domain and lowers the data rate to half rate.

Figure 7–11 shows the registers available in the Arria V output and output-enable paths. The path is divided into the HDR block, and output and output-enable registers. The device can bypass each block of the output and output-enable path.

Figure 7–11. IOE Output and Output-Enable Path Registers for Arria V Devices *(1)*



Notes to Figure 7–11:

(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 full-rate write clock can come from the PLL. The DQ write clock and DQS write clock have a 90° offset between them.

The output path is designed to route combinatorial or registered single data rate (SDR) outputs and full-rate or half-rate DDR outputs from the FPGA core. Half-rate data is converted to full-rate with the HDR block, clocked by the half-rate clock from the PLL.

The output-enable path has a structure similar to the output path—ensuring that the output-enable path goes through the same delay and latency as the output path. You can have a combinatorial or registered output in SDR applications and you can use half- or full-rate operation in DDR applications.

Delay Chain

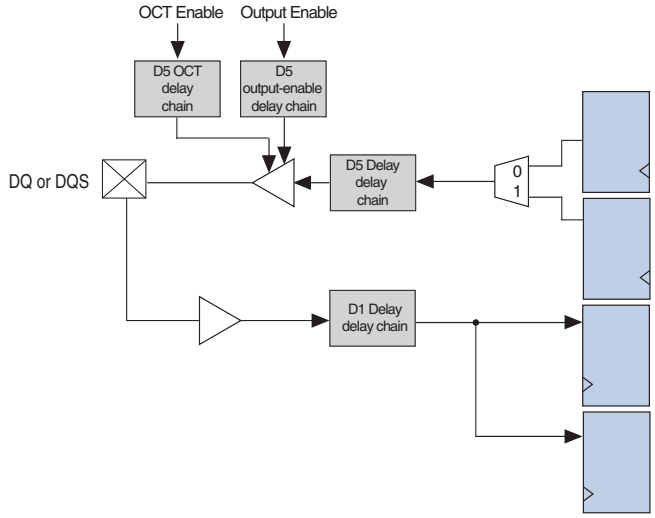
Arria V devices have run-time adjustable delay chains in the I/O blocks and the DQS logic blocks. You can control the delay chain setting through the I/O or the DQS configuration block output.

Every I/O block contains a delay chain between the following elements:

- The output registers and output buffer
- The input buffer and input register
- The output enable and output buffer
- The R_T OCT enable-control register and output buffer

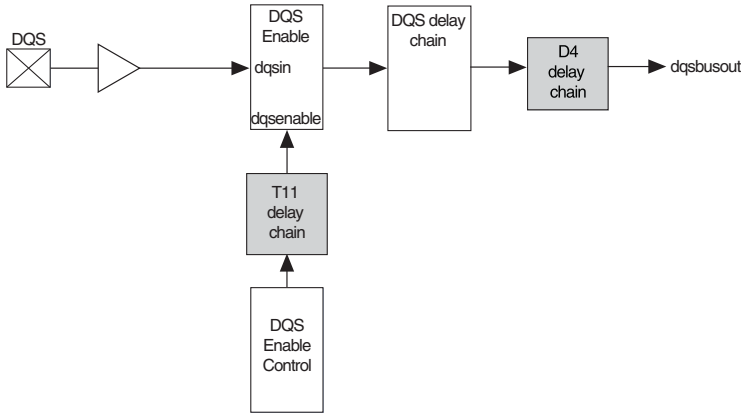
Figure 7–12 shows the delay chains in an I/O block.

Figure 7–12. Delay Chains in an I/O Block



Each DQS logic block contains a delay chain after the `dqsbusout` output and another delay chain before the `dqsenable` input. Figure 7–13 shows the delay chains in the DQS input path.

Figure 7–13. Delay Chains in the DQS Input Path



Hard Memory Controllers

Arria V devices feature hard memory controllers. These dedicated memory controllers allow support for higher memory interface frequencies in comparison to the memory controllers implemented using core logic.

You can use the dedicated memory controllers for DDR2 and DDR3 SDRAM interfaces. The hard memory controller supports other features similar to the DDR2 and DDR3 SDRAM High-Performance Controller II.

The hard memory controllers in Arria V devices use dedicated I/O pins as data, address, command, control, and clock pins for the SDRAM interface. If you do not use the hard memory controllers, you can use these dedicated pins as regular I/O pins.

Apart from the hard memory controllers, you can also implement memory controllers using core logic.

Features of the Hard Memory Controller

Table 7–3 lists the features of the hard memory controller in Arria V devices.

Table 7–3. Features of the Arria V Hard Memory Controller (Part 1 of 2)

Feature	Description
Memory Interface Data Width	■ 8-, 16-, and 32-bit data ■ 16-bit data + 8-bit ECC ■ 32-bit data + 8-bit ECC
Memory Density	The controller supports up to four gigabits density parts and two chip selects.
Memory Burst Length	■ DDR3—Burst length of 8 and burst chop of 4 ■ DDR2—Burst lengths of 4 and 8 ■ LPDDR—Burst lengths of 2, 4, 8, and 16 ■ LPDDR2—Burst lengths of 2, 4, 8, and 16
Command and Data Reordering	The controller increases efficiency through the support for out-of-order execution of DRAM commands—with address collision detection—and in-order return of results.
Starvation Control	A starvation counter ensures that all requests are served after a predefined time-out period. This function ensures that data with low priority access are not left behind when reordering data for efficiency.
User-Configurable Priority Support	When the controller detects a high priority request, it allows the request to bypass the current queuing request. This request is processed immediately and thus reduces latency.
Avalon®-MM Data Slave Local Interface	By default, the controller supports the Avalon Memory-Mapped protocol.
Bank Management	By default, the controller provides closed-page bank management on every access. The controller intelligently keeps a row open based on incoming traffic. This feature improves the efficiency of the controller especially for random traffic.
Streaming Reads and Writes	The controller can issue reads or writes continuously to sequential addresses every clock cycle if the bank is open. This function allows for very high efficiencies with large amounts of data.
Bank Interleaving	The controller can issue reads or writes continuously to ‘random’ addresses.

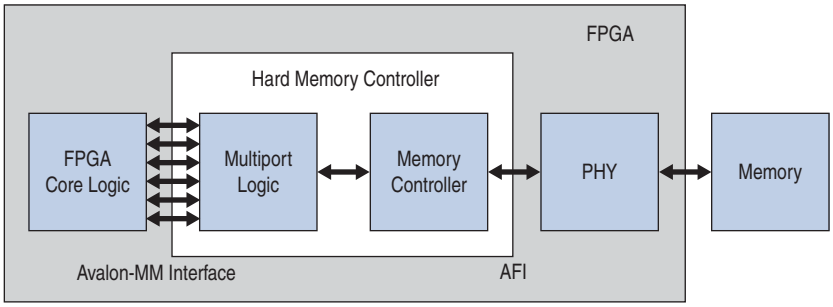
Table 7–3. Features of the Arria V Hard Memory Controller (Part 2 of 2)

Feature	Description
Predictive Bank Management	The controller can issue bank management commands early so that the correct row is open when the read or write happens. This increases efficiency.
Multiport Interface	The interface allows you to connect up to six data masters to access the memory controller through the local interface. You can update the multiport scheduling configuration without interrupting traffic on a port.
Built-in Burst Adaptor	The controller can accept bursts of arbitrary sizes on its local interface and map these bursts to efficient memory commands.
Run-time Configuration of the Controller	This feature provides support for updates to the timing parameters without requiring reconfiguration of the FPGA, apart from the standard compile-time setting of the timing parameters.
On-Die Termination	The controller controls the on-die termination (ODT) in the memory, which improves signal integrity and simplifies your board design.
User-Controlled Refresh Timing	You can optionally control when refreshes occur—allowing the refreshes to avoid clashing of important reads or writes with the refresh lock-out time.
Low Power Modes	You can optionally request the controller to put the memory into the self-refresh or deep power-down modes.
Partial Array Self-Refresh	You can select the region of memory to refresh during self-refresh through the mode register to save power.
ECC	Standard Hamming single error correction, double error detection (SECDED) error correction code (ECC) support: ■ 32-bit data + 8-bit ECC ■ 16-bit data + 8-bit ECC
Additive Latency	With additive latency, the controller can issue a READ/WRITE command after the ACTIVATE command to the bank prior to t_{RCD} to increase the command efficiency.
Write Acknowledgement	The controller supports write acknowledgment on the local interface.
User Control of Memory Controller Initialization	The controller supports initialization of the memory controller under the control of user logic—for example, through the software control in the user system if a processor is present.
Controller Bonding Support	You can bond two controllers to achieve wider data width for higher bandwidth applications.

Multiport Logic

Multiport logic allows up to six local interfaces from the core logic to access a controller. Figure 7–14 shows a simplified diagram of the Arria V hard memory controller with the multiport logic.

Figure 7–14. Simplified Diagram of the Arria V Hard Memory Interface



Bonding Support

You can bond two hard memory controllers to support wider data widths. When you bond two hard memory controllers, the data going out of the controllers to the user logic is synchronized. However, the data going out of the controllers to the memory is not synchronized.

The bonding controllers are not synchronized and remain independent with two separate address busses and two independent command busses. These busses are calibrated separately.

Only one bonding feature is available per package, either through the dedicated routing or through the core fabric. Memory interface using the bonding feature will have higher average latency. Bonding through the core fabric will also cause a higher latency compared to bonding through the dedicated routing.

Figure 7–15 and Figure 7–16 show the number of hard memory controllers, the locations, and the bonding support for the hard memory controllers in Arria V devices.

Figure 7–15. Hard Memory Controllers in 5AGXA1 and 5AGXA3 Devices

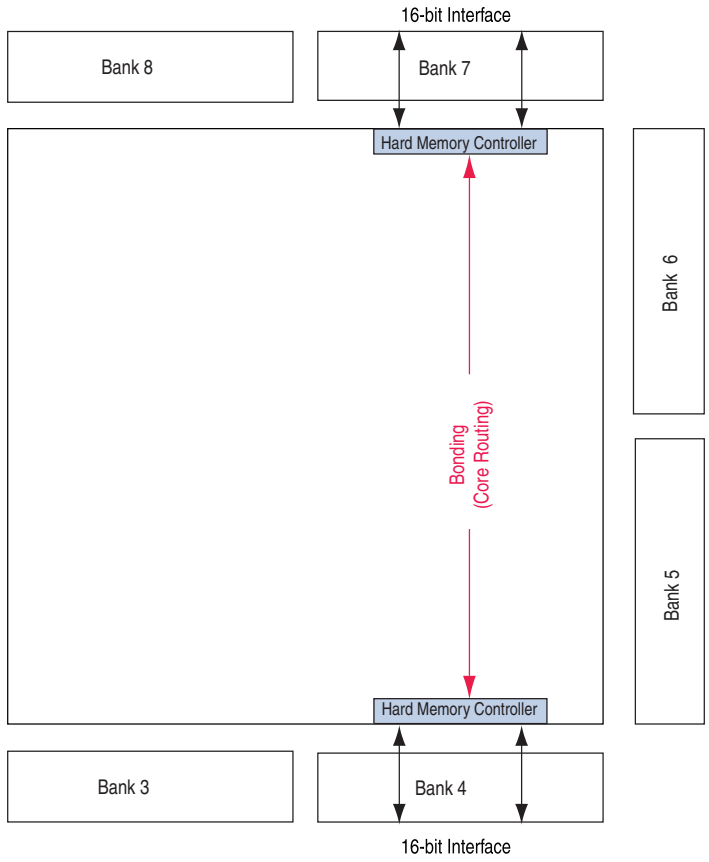
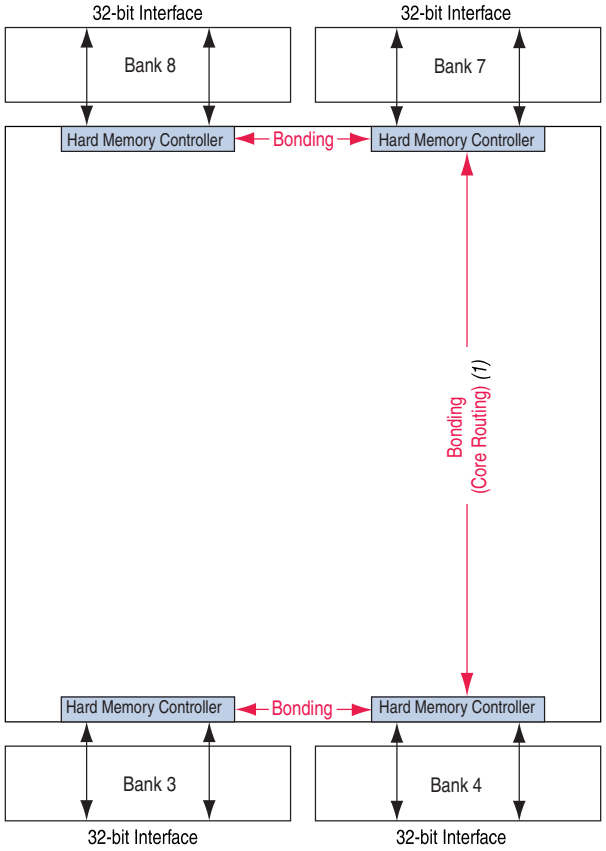


Figure 7–16. Hard Memory Controllers in 5AGXA5, 5AGXA7, 5AGXB1, 5AGXB3, 5AGXB5, 5AGXB7, 5AGTD3, and 5AGTD7 Devices



Note to Figure 7–16:

(1) This is enabled only for single hard memory controller bond out per side. This bonding is available only if you do not use the hard memory controllers in banks 4 and 7 for bonding with other banks.

For more information about the dedicated pins, refer to *Arria V Device Family Pin Connection Guidelines*.

UniPHY IP

The UniPHY IP is optimized to take advantage of the Arria V I/O structure and the Quartus II software TimeQuest Timing Analyzer. Memory controllers with UniPHY IP ensures the highest reliable frequency of operation across process, voltage, and temperature (PVT) variations.

The UniPHY IP instantiates a PLL to generate related clocks for the memory interface.

Memory controllers with the UniPHY IP and the Altera memory controller MegaCore® functions can run at half the frequency of the I/O interface of the memory devices to allow better timing management in high-speed memory interfaces. Arria V devices have built-in circuitry in the IOE to convert data from full rate (the I/O frequency) to half rate (the controller frequency) and vice versa.

For more information about the UniPHY IP, refer to *Volume 3: Implementing Altera Memory Interface IP* of the *External Memory Interface Handbook*.

Document Revision History

Table 7–4 lists the revision history for this chapter.

Table 7–4. Document Revision History

Date	Version	Changes
November 2011	1.1	■ Updated Table 7–2. ■ Added "PHY Clock (PHYCLK) Networks" and "UniPHY IP" sections. ■ Restructured chapter.
May 2011	1.0	Initial release.

Section III. System Integration for Arria V Devices

This section provides information about Arria® V device configuration, design security, remote system upgrades, SEU mitigation, JTAG, and power requirements. This section includes the following chapters:

- Chapter 8, Configuration, Design Security, and Remote System Upgrades in Arria V Devices
- Chapter 9, SEU Mitigation in Arria V Devices
- Chapter 10, JTAG Boundary-Scan Testing in Arria V Devices
- Chapter 11, Power Management in Arria V Devices

Revision History

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

8. Configuration, Design Security, and Remote System Upgrades in Arria V Devices

AV-52008-1.1

This chapter describes the supported configuration schemes for Arria® V devices, instructions for executing the required configuration schemes, and all 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 V devices.

Arria V devices use SRAM cells to store configuration data. Because SRAM memory is volatile, you must download the configuration data to the Arria V device each time the device powers up. You can configure Arria V devices using one of four configuration schemes:

- Fast passive parallel (FPP) (x8 and x16)
- Active serial (AS) (x1 and x4)
- Passive serial (PS)
- JTAG

All configuration schemes use either an external controller (for example, a MAX® II device or microprocessor), a configuration device, or a download cable. For more information about the configuration features, refer to "Features".

This chapter contains the following sections:

Features

Arria V device soffer the following configuration features:

- Decompression—allows Arria V devices to receive a compressed configuration bitstream and decompress this data in real-time, reducing storage requirements and configuration time.
- Design security—offers protection to your Arria V device.
- Remote system upgrade—allows you to upgrade your Arria V designs remotely in real-time.
- Partial reconfiguration—allows you to reconfigure part of the device while other sections remain running.

Table 8–1 lists which configuration features you can use in each configuration scheme.

Table 8–1. Configuration Features for Arria V Devices

Configuration Scheme	Decompression	Design Security	Remote System Upgrade	Partial Reconfiguration
FPP (x8 and x16)	✓ (1)	✓ (1)	✓ (2)	✓ (3)
AS (x1, and x4)	✓	✓	✓	—
PS	✓	✓	✓ (2)	—
JTAG	—	—	—	—

Notes to Table 8–1:

(1) In these configuration schemes, the host system must accommodate a different DCLK-to-DATA[] ratio. For more information, refer to "Fast Passive Parallel Configuration" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

(2) FPP and PS modes support remote system upgrade feature through the parallel flash loader (PFL) megafunction.

(3) FPP x16 I/O interface supports partial reconfiguration.

For more information about FPP, AS, PS, and JTAG, refer to "Configuration, Design Security, and Remote System Upgrades" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

Power-On Reset

The following sections describe the power-on reset (POR) circuit and the power supply for the configuration pins.

POR 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 all the power supplies monitored by the POR circuitry are above the device's POR trip point. On power down, brown-out occurs if any of the power supplies monitored by the POR circuitry drops below the threshold level of the hot-socket circuitry.

For more information about which power supplies are monitored by the POR circuitry, refer to *Power Management in Arria V Devices* chapter.

POR Delay Specification

POR delay is defined as the delay between the time when all the power supplies monitored by the POR circuitry reach the minimum recommended operating voltage to the time when nSTATUS is released high and your device is ready to begin configuration.

For more information about the POR delay specification, refer to *Device Datasheet for Arria V Devices* chapter.

Power Supply

The following sections describe the configuration power supplies.

V_{CCPGM} Supply

The V_{CCPGM} is for all dedicated configuration pins and dual-purpose pins.

Use the V_{CCPGM} pin to power all the dedicated configuration inputs, dedicated configuration outputs, dedicated configuration bidirectional pins, and the dual-purpose pins that you use for configuration. The configuration input buffers do not have to share power lines with the regular I/O buffer in Arria V devices.

The operating voltage for the configuration input pin is independent of the I/O banks power supply, V_{CCIO}, during configuration. Therefore, Arria V devices do not require configuration voltage constraints on V_{CCIO}. The supported configuration voltages are 1.8, 2.5, 3.0, and 3.3 V.

For more information about the configuration pins connections, refer to *Arria V Device Family Pin Connection Guidelines*.

V_{CCPD} Supply

The V_{CCPD} supply is a dedicated programming power supply to power the I/O pre-drivers and JTAG I/O pins (TCK, TMS, TDI, and TDO). The supported configuration voltages are 2.5, 3.0, and 3.3 V.

V_{CCPD} must be greater than or equal to V_{CCIO}. If V_{CCIO} is set to 3.3 V, V_{CCPD} must be powered up to 3.3 V. If V_{CCIO} is set to 3.0 V, V_{CCPD} must be powered up to 3.0 V. If V_{CCIO} is set to 2.5 V or lower, V_{CCPD} must be powered up to 2.5 V. This applies for all the banks containing the VCCPD and VCCIO pins.

For more information about the configuration pins power supply, refer to "Device Configuration Pins" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

Configuration Schemes

Table 8–2 lists the configuration schemes for Arria V devices.

Table 8–2. Configuration Schemes for Arria V Devices (Part 1 of 2)

Configuration Scheme	Decompression Feature	Design Security Feature	Configuration Voltage Standard (V) (1)	POR Delay (2)	MSEL[4..0]
FPP x8	Disabled	Disabled	1.8/2.5/3.0/3.3	Fast	10100
				Standard	11000
	Disabled	Enabled	1.8/2.5/3.0/3.3	Fast	10101
				Standard	11001
	Enabled	Optional (3)	1.8/2.5/3.0/3.3	Fast	10110
				Standard	11010

Table 8–2. Configuration Schemes for Arria V Devices (Part 2 of 2)

Configuration Scheme	Decompression Feature	Design Security Feature	Configuration Voltage Standard (V) (1)	POR Delay (2)	MSEL[4..0]
FPP x16	Disabled	Disabled	1.8/2.5/3.0/3.3	Fast	00000
				Standard	00100
	Disabled	Enabled	1.8/2.5/3.0/3.3	Fast	00001
				Standard	00101
	Enabled	Optional (3)	1.8/2.5/3.0/3.3	Fast	00010
				Standard	00110
PS	Optional (3)	Optional (3)	1.8/2.5/3.0/3.3	Fast	10000
				Standard	10001
AS (x1, x4) (4)	Optional (3)	Optional (3)	1.8/3.0/3.3	Fast	10010
				Standard	10011
JTAG-based configuration (5)	Disabled	Disabled	—	—	(6)

Notes to Table 8–2:

(1) The configuration voltage standard applied to the V_{CCPGM} power supply that powers all the configuration pins during configuration.

(2) For POR delay specifications, refer to "POR Delay Specification" on page 8–2.

(3) You can enable or disable this feature.

(4) The AS configuration scheme supports the remote system upgrade feature. For more information about the remote system upgrade feature, refer to "Remote System Upgrades" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

(5) JTAG-based configuration takes precedence over other configuration schemes. This means the MSEL pin settings are ignored. The JTAG-based configuration does not support the design security or decompression features.

(6) Do not leave the MSEL pins floating. Connect them to V_{CCPGM} or GND. This pin supports the non-JTAG configuration scheme used in production. If you only use the JTAG configuration, Altera recommends connecting the MSEL pins to GND.

MSEL Pin Settings

Select the configuration scheme by driving the Arria V device MSEL pins either high or low (refer to Table 8–2). The V_{CCPGM} power supply powers the MSEL input buffers. The MSEL[4..0] pins have 5-kΩ internal pull down resistors that are always active. During POR and during reconfiguration, the MSEL pins must be at the LVTTL V_{IL} and V_{IH} levels to be considered logic low and logic high, respectively.

Altera recommends hardwiring the MSEL pins to V_{CCPGM} or GND without pull-up or pull-down resistors. Do not drive the MSEL pins with a microprocessor or another device.

Raw Binary File Size

Table 8–3 lists the uncompressed raw binary file (**.rbf**) sizes for Arria V devices.

Table 8–3. Uncompressed .rbf Sizes for Arria V Devices (1) (Part 1 of 2)

Device	Configuration .rbf Size (bits)
5AGXA1	69,034,936
5AGXA3	69,034,936
5AGXA5	101,511,640

Table 8–3. Uncompressed .rbf Sizes for Arria V Devices (1) (Part 2 of 2)

Device	Configuration .rbf Size (bits)
5AGXA7	101,511,640
5AGXB1	138,416,696
5AGXB3	138,416,696
5AGXB5	185,327,416
5AGXB7	185,327,416
5AGTD3	138,416,696
5AGTD7	185,327,416

Note to Table 8–3:

(1) These values are preliminary.

Use the data in Table 8–3 to estimate the file size before design compilation. Different configuration file formats, such as a hexadecimal file (**.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 *Device Configuration Options* and *Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

Device Configuration Pins

This section describes the connections and functionality of all the configuration-related pins on the Arria V devices.

Table 8–4 lists the Arria V configuration pins and their power supply.

Table 8–4. Configuration Pin Summary for Arria V Devices (Part 1 of 2)

Description	Input/Output	User Mode	Powered By	Configuration Scheme
TDI	Input	—	V_{CCPD} (4)	JTAG
TMS	Input	—	V_{CCPD} (4)	JTAG
TCK	Input	—	V_{CCPD} (4)	JTAG
TDO	Output	—	V_{CCPD} (4)	JTAG
CLKUSR	Input	I/O (1)	V_{CCPGM}/V_{CCIO} (5)	All schemes
CRC_ERROR	Output	I/O (1)	Pull-up	Optional, all schemes
CONF_DONE	Bidirectional	—	V_{CCPGM}/Pull-up (6)	All schemes
DCLK	Input	—	V_{CCPGM}	FPP, PS
	Output	—	V_{CCPGM}	AS
DEV_OE	Input	I/O (1)	V_{CCPGM}/V_{CCIO} (5)	Optional, all schemes
DEV_CLRn	Input	I/O (1)	V_{CCPGM}/V_{CCIO} (5)	Optional, all schemes
INIT_DONE	Output	I/O (1)	Pull-up	Optional, all schemes
MSEL[4..0]	Input	—	V_{CCPGM}	All schemes

Table 8–4. Configuration Pin Summary for Arria V Devices (Part 2 of 2)

Description	Input/Output	User Mode	Powered By	Configuration Scheme
nSTATUS	Bidirectional	—	V_{CCPGM}/Pull-up (6)	All schemes
nCE	Input	—	V_{CCPGM}	All schemes
nCEO	Output	I/O (2)	Pull-up	All schemes
nCONFIG	Input	—	V_{CCPGM}	All schemes
DATA[15..5]	Input	I/O (3)	V_{CCPGM}/V_{CCIO} (5)	FPP
nCSO/DATA4	Output	—	V_{CCPGM}	AS
	Bidirectional	—	V_{CCPGM}/V_{CCIO} (5)	FPP
AS_DATA[3..1]/DATA[3..1]	Bidirectional	—	V_{CCPGM}	AS
	Bidirectional	—	V_{CCPGM}/V_{CCIO} (5)	FPP
AS_DATA0/DATA0/ASDO	Bidirectional	—	V_{CCPGM}	AS
	Bidirectional	—	V_{CCPGM}/V_{CCIO} (5)	FPP, PS

Notes to Table 8–4:

(1) This is a dual-purpose pin. This pin is available as an I/O if the associated option that enables this pin is turned off from the **Configuration** panel in the **Device and Pins Option** settings. For example, `DEV_OE` is available as a user I/O if the **Enable device-wide output enable** option is turned off.

(2) This pin is available as an I/O if this pin is not feeding the next device's `nCE` in a multi-device configuration. To use this pin to feed the next device's `nCE` in a multi-device chain, turn on the **Enable nCEO pin** option under the **Device and Pins Option**, **General** panel in the Quartus II software.

(3) This is a dual-purpose pin. The state of this pin in user mode depends on the **Dual-purpose Pins** settings in the **Device and Pins Option**.

(4) This pin is powered up by V_{CCPD} of the bank in which the pin resides.

(5) This pin is powered up by V_{CCPGM} during configuration. It is powered up by V_{CCIO} of the bank in which the pin resides if it is used as a regular I/O in user mode.

(6) The pin is an open-drain output and is pulled up by V_{CCPGM}.

For more information about configuration pins, refer to "Device Configuration Pins" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

Document Revision History

Table 8–5 lists the revision history for this chapter.

Table 8–5. Document Revision History

Date	Version	Changes
November 2011	1.1	Restructured chapter.
May 2011	1.0	Initial release.

9. SEU Mitigation in Arria V Devices

AV-52009-1.1

This chapter describes how to activate and use the error detection cyclic redundancy check (CRC) feature when your Arria® V device is in user mode and how to recover from configuration errors caused by CRC errors. The error detection feature is enhanced in Arria V devices.

In critical applications such as avionics, telecommunications, system control, and military applications, you can use the error detection feature to confirm that the stored configuration data stored is correct, and alert the system if a configuration error occurs.

This chapter contains the following section:

- "Error Detection Timing"

For more information about activating and using the error detection CRC feature in the user mode of your Arria V device, and how to recover from configuration errors caused by CRC errors, refer to "SEU Mitigation" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

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.

If an error is detected within a frame, CRC_ERROR is driven high at the end of the error location search, after the error message register (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 9–1 lists the minimum and maximum error detection frequencies in Arria V devices.

Table 9–1. Minimum and Maximum Error Detection Frequencies in Arria V Devices

Error Detection Frequency	Maximum Error Detection Frequency	Minimum Error Detection Frequency	Valid Divisors (n)
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. 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 9–1).

Equation 9–1. Error Detection Frequency

$$\text{Error Detection Frequency} = \frac{100\ \text{MHz}}{2^n}$$

For more information, refer to "Software Support" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

The error detection frequency reflects the frequency of the error detection process for a frame because the CRC calculation in Arria V 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 of the 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 single event upset (SEU) event.

Table 9–2 lists the estimated minimum interval time between each update for the EMR in Arria V devices.

Table 9–2. Minimum Update Interval for Error Message Register in Arria V Devices (1)

Family	Device	Timing Interval (µs)
Arria V GX	5AGXA1	2.73
	5AGXA3	2.73
	5AGXA5	2.79
	5AGXA7	2.79
	5AGXB1	2.99
	5AGXB3	2.99
	5AGXB5	3.59
	5AGXB7	3.59
Arria V GT	5AGTD3	2.99
	5AGTD7	3.59

Note to Table 9–2:

(1) These timing numbers are preliminary.

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.

The minimum CRC calculation time is calculated using the maximum error detection frequency with a divisor factor of 1. The maximum CRC calculation time is calculated using the minimum error detection frequency with a divisor factor of 8.

Table 9–3 lists the minimum and maximum estimated clock frequency time for each CRC calculation for Arria V devices.

Table 9–3. CRC Calculation Time for Arria V Devices (1)

Family	Device	Minimum Time (µs)	Maximum Time (ms)
Arria V GX	5AGXA1	11.52	188.01
	5AGXA3	11.52	188.01
	5AGXA5	15.37	196.43
	5AGXA7	15.37	196.43
	5AGXB1	20.57	225.82
	5AGXB3	20.57	225.82
	5AGXB5	27.49	326.27
	5AGXB7	27.49	326.27
Arria V GT	5AGTD3	20.57	225.82
	5AGTD7	27.49	326.27

Note to Table 9–3:

(1) These timing numbers are preliminary.

Document Revision History

Table 9–4 lists the revision history for this chapter.

Table 9–4. Document Revision History

Date	Version	Changes
November 2011	1.1	Restructured chapter.
May 2011	1.0	Initial release.

10. JTAG Boundary-Scan Testing in Arria V Devices

AV-52010-1.2

This chapter describes the boundary-scan test (BST) features that are supported in Arria® V devices.

Arria V devices support IEEE Std. 1149.1.

This chapter includes the following sections:

IEEE Std.1149.1 BST Architecture

An Arria V device operating in IEEE Std. 1149.1 BST mode uses four required pins, `TDI`, `TDO`, `TMS`, and `TCK`. The `TCK` pin has an internal weak pull-down resistor, while the `TDI` and `TMS` pins have weak internal pull-up resistors. The 3.3-, 3.0-, and 2.5-V_{CCPD} supply of I/O bank 3A powers the `TDO` output pin and all the JTAG input pins. All user I/O pins are tri-stated during JTAG configuration. Arria V devices do not support the optional pin, `TRST`.

For more information about the following IEEE Std. 1149.1 BST features, refer to "JTAG Boundary-Scan Testing" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter:

- IEEE Std. 1149.1 BST architecture
- IEEE Std. 1149.1 boundary-scan register
- IEEE Std. 1149.1 BST operation control
- Using IEEE Std. 1149.1 BST circuitry
- Disabling IEEE Std. 1149.1 BST circuitry
- IEEE Std. 1149.1 BST guidelines

JTAG Instruction

Table 10–1 lists the JTAG instructions that are supported by Arria V devices.

Table 10–1. JTAG Instructions Supported by Arria V Devices

JTAG Instruction	Instruction Code	Description
SAMPLE/PRELOAD (1)	00 0000 0101	Allows you to capture and examine a snapshot of signals at the device pins during normal device operation and permits an initial data pattern to be an output at the device pins. The SignalTap™ II Embedded Logic Analyzer also uses this instruction.
EXTEST (1), (2)	00 0000 1111	Allows you to test the external circuit and board-level interconnects by forcing a test pattern at the output pins and capturing the test results at the input pins.
BYPASS (1)	11 1111 1111	Places the 1-bit bypass register between the TDI and TDO pins, allowing the BST data to pass synchronously through selected devices to adjacent devices during normal device operation.
USERCODE (1)	00 0000 0111	Selects the 32-bit USERCODE register and places it between the TDI and TDO pins, allowing USERCODE to be serially shifted out of TDO.
IDCODE (1)	00 0000 0110	Selects the IDCODE register and places it between the TDI and TDO pins, allowing IDCODE to be serially shifted out of TDO. IDCODE is the default instruction at power up and in the TAP RESET state.
HIGHZ (1), (2)	00 0000 1011	Places the 1-bit bypass register between the TDI and TDO pins, allowing the BST data to pass synchronously through selected devices to adjacent devices during normal device operation while tri-stating all of the I/O pins.
CLAMP (1), (2)	00 0000 1010	Places the 1-bit bypass register between the TDI and TDO pins, allowing the BST data to pass synchronously through selected devices to adjacent devices during normal device operation while holding the I/O pins to a state defined by the data in the boundary-scan register.
PULSE_NCONFIG (3)	00 0000 0001	Emulates pulsing the nCONFIG pin low to trigger reconfiguration even though the physical pin is unaffected.
CONFIG_IO (3)	00 0000 1101	Allows I/O reconfiguration (after or during reconfigurations) through the JTAG ports using IOCSR for JTAG testing. The nSTATUS pin must go high before you can issue the CONFIG_IO instruction.
LOCK (3), (4)	01 1111 0000	Put the device in JTAG secure mode. In this mode, only BYPASS, SAMPLE/PRELOAD, EXTEST, IDCODE, and SHIFT_EDERROR_REG instructions are supported.
UNLOCK (3), (4)	11 0011 0001	Release the device from the JTAG secure mode to enable access to all other JTAG instructions.

Notes to Table 10–1:

(1) For more information about this instruction mode, refer to "IEEE Std. 1149.1 BST Operation Control" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

(2) Bus hold and weak pull-up resistor features override the high-impedance state of HIGHZ, CLAMP, and EXTEST.

(3) For more information about this instruction mode, refer to "JTAG Configuration" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

(4) You can only issue this instruction through JTAG core access in user mode.

When the tamper-protection bit is enabled, the Arria V device is in the JTAG secure mode after power up. Only the three mandatory JTAG instructions (BYPASS, SAMPLE/PRELOAD, and EXTEST), the IDCODE optional instruction, and the SHIFT_EDERROR_REG private instruction are supported by the JTAG pins in JTAG secure mode. To enable access to all other JTAG instructions, you must issue the UNLOCK JTAG instruction.

The IDCODE instruction is the default instruction when the TAP controller is in the reset state. Without loading any instructions, you can go to the SHIFT_DR state and shift out the JTAG device ID.

If the device is in a reset state and 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 nCONFIG and nSTATUS signals are high.

You must not invoke the following private instructions at any instance:

- 1100010000
- 0011001001
- 1100010011
- 1100010111
- 0111100000
- 1110110011

CAUTION

These private instructions can potentially damage the device, rendering the device useless. If you require the use of these instructions, contact Altera® Applications.

Arria V IDCODE

The IDCODE is unique for each Arria V device. You can shift out the 32-bit IDCODE to determine the correct device during BST by issuing an IDCODE instruction.

Table 10–2 lists the IDCODE information for Arria V devices.

Table 10–2. IDCODE Information for Arria V Devices (Part 1 of 2)

Family	Device	IDCODE (32 Bits) (1)			
		Version (4 Bits)	Part Number (16 Bits)	Manufacture Identity (11 Bits)	LSB (1 Bit) (2)
Arria V GX	5AGXA1	0000	0010 1010 0001 0001	000 0110 1110	1
	5AGXA3	0000	0010 1010 0000 0001	000 0110 1110	1
	5AGXA5	0000	0010 1010 0001 0010	000 0110 1110	1
	5AGXA7	0000	0010 1010 0000 0010	000 0110 1110	1
	5AGXB1	0000	0010 1010 0001 0011	000 0110 1110	1
	5AGXB3	0000	0010 1010 0000 0011	000 0110 1110	1
	5AGXB5	0000	0010 1010 0001 0100	000 0110 1110	1
	5AGXB7	0000	0010 1010 0000 0110	000 0110 1110	1

Table 10–2. IDCODE Information for Arria V Devices (Part 2 of 2)

Family	Device	IDCODE (32 Bits) (1)			
		Version (4 Bits)	Part Number (16 Bits)	Manufacture Identity (11 Bits)	LSB (1 Bit) (2)
Arria V GT	5AGTD3	0000	0010 1010 0000 0011	000 0110 1110	1
	5AGTD7	0000	0010 1010 0000 0110	000 0110 1110	1

Notes to Table 10–2:

(1) The MSB is on the left.

(2) The LSB of the IDCODE is always 1.

I/O Voltage Support in a JTAG Chain

The JTAG chain supports 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 10–3 lists the supported TDO and TDI voltage combination to ensure proper JTAG chain operation.

Table 10–3. Supported TDO and TDI Voltage Combinations

Device	TDI Input Buffer Power (V)	Arria V TDO V_{CCPD}		
		V_{CCPD} = 3.3 V (1)	V_{CCPD} = 3.0 V (1)	V_{CCPD} = 2.5 V (2)
Arria V	V_{CCPD} = 3.3	✓	✓	✓
	V_{CCPD} = 3.0	✓	✓	✓
	V_{CCPD} = 2.5	✓	✓	✓
Non-Arria V	V_{CC} = 3.3	✓ (3)	✓ (4)	✓ (5)
	V_{CC} = 2.5	✓ (3)	✓ (4)	✓ (5)
	V_{CC} = 1.8	✓ (3)	✓ (4)	✓ (5)
	V_{CC} = 1.5	✓ (3)	✓ (4)	✓ (5)

Notes to Table 10–3:

(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) Input buffer must be 3.3-V tolerant.

(4) Input buffer must be 3.0-V tolerant.

(5) Input buffer must be 2.5-V tolerant.

Document Revision History

Table 10–4 lists the revision history for this chapter.

Table 10–4. Document Revision History

Date	Version	Changes
February 2012	1.2	Updated Table 10–2.
November 2011	1.1	Minor text edits.
May 2011	1.0	Initial release.

11. Power Management in Arria V Devices

AV-52011-1.3

This chapter describes power management, hot-socketing specifications, power-on reset (POR) requirements, and their implementation in Arria® V devices.

The PowerPlay Power Analyzer in the Quartus® II software optimizes all designs with Arria V 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 thermal management, use the Arria V internal temperature sensing diode (TSD) with built-in analog-to-digital converter (ADC) circuitry to easily incorporate this feature in your designs. Being able to monitor the junction temperature of the device at any time also allows you to alter the system environment in response (for example, changing the air flow to the device) to optimize thermal conditions and potentially save power for the whole system.

Arria V devices offer hot-socketing, also known as hot plug-in or hot swap, and power sequencing support without the use of external devices.

This chapter contains the following sections:

External Power Supply Requirements

This section provides the external power supplies required to power Arria V devices. You can use the same external power supply for the power supply pins with the same voltage level requirements.

For each Altera-recommended power supply's operating conditions, refer to *Device Datasheet for Arria V Devices* chapter.

For more information about power supply pin connection guidelines and power regulator sharing, refer to *Arria V Device Family Pin Connection Guidelines*.

For more information about power supply design requirements, refer to Board Design Resource Center page.

Internal Temperature Sensing Diode

The Arria V TSD uses the characteristics of a PN junction diode to determine die temperature. Knowing the junction temperature is crucial for thermal management. Historically, junction temperature is calculated using ambient or case temperature, junction-to-ambient (ja) or junction-to-case (jc) thermal resistance, and device power consumption. Arria V devices can monitor its die temperature with the internal TSD with built-in ADC circuitry.

You can use the Arria V internal TSD in two modes of operation—power-up mode and user mode. For power-up mode, the internal TSD reads the die's temperature during configuration if you enable the ALTTEMP_SENSE megafunction in your design. The ALTTEMP_SENSE megafunction allows temperature sensing during device user mode by asserting the `clken` signal to the internal TSD circuitry. To reduce power consumption, disable the internal TSD with built-in ADC circuitry when not in use.

For more information about using the ALTTEMP_SENSE megafunction, refer to *Temperature Sensor (ALTTEMP_SENSE) Megafunction User Guide*.

For more information about internal TSD specification, refer to *Device Datasheet for Arria V Devices* chapter.

Hot-socketing

Arria V I/O pins are hot-socketing compliant without the need for external components or special design requirements. Hot-socketing support in Arria V 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 the other operating buses.
- You can insert or remove an Arria V device from a powered-up system board without damaging or interfering with normal system and board operation.

The hot-socketing circuit monitors the V_{CCIO}, V_{CCPD}, V_{CC}, and V_{CCP} power supplies. All V_{CCIO} and V_{CCPD} banks are monitored by the hot-socketing circuitry and only two pins are exempted from the hot-socketing feature—`CONF_DONE` and `nSTATUS` pins in bank 8A. You can power up or power down these power supplies in any sequence. During hot-socketing, the I/O pin capacitance is less than 15 pF and the clock pin capacitance is less than 20 pF.

Ensure that all power supplies for Arria V devices meet the ramp-up and ramp-down rate specification of 100 ns to 100 ms. Each power supply must meet the ramp-up and ramp-down specifications, but there can be any amount of time between power supply ramp up or ramp down. The specification is applied to the ramp rate of each power supply.

The hot-socketing feature removes some of the difficulty when using Arria V devices on PCBs that contain a mixture of 3.3-, 3.0-, 2.5-, 1.8-, 1.5-, 1.35-, 1.25-, 1.20-, and 1.10-V devices.

For more information about the hot-socketing feature, refer to "Arria V Hot-Socketing Specifications" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

For more information about the hot-socketing specifications, refer to *Device Datasheet for Arria V Devices* chapter.

Power-On Reset Specifications

Table 11–1 lists the power supplies that the POR circuit monitors.

Table 11–1. Power Supplies Monitored by the POR Circuitry for Arria V Devices

Power Supply	Description	Setting (V)
V_{CCAUX}	Auxiliary supply	2.5
V_{CCBAT} (1)	Battery back-up power supply for the design security volatile key register	3.0–1.2
V_{CC}	Core power supply	1.1
V_{CCP}	Periphery circuitry, PCIe® hard IP block, and transceiver physical coding sublayer (PCS) power supply	1.1
V_{CCPD}	I/O pre-driver power supply	2.5, 3.0, 3.3
V_{CCPGM}	Configuration pins power supply	1.8, 2.5, 3.0, 3.3

Note to Table 11–1:

(1) V_{CCBAT} must be powered, even if the volatile key is not used, in order for the device to exit POR.

Table 11–2 lists the power supplies that the POR circuit does not monitor.

Table 11–2. Power Supplies Not Monitored by the POR Circuitry for Arria V Devices

Power Supply	Description	Setting (V)
V_{CCT_GXB}	Transmitter power	1.1
V_{CCH_GXB}	Transmitter output buffer power	1.5
V_{CCR_GXB}	Receiver power	1.1
V_{CCA_GXB}	Transceiver high voltage power	2.5
V_{CCIO}	I/O power supply	1.2, 1.25, 1.35, 1.5, 1.8, 2.5, 3.0, 3.3
V_{CCA_FPLL}	Phase-locked loop (PLL) analog global power supply	2.5
V_{CCD_FPLL}	PLL digital power supply	1.5

The POR specification is designed to ensure that all the circuits in the Arria V device are at certain known states during power up.

For more information about the POR specification and `MSEL` pin settings, refer to *Configuration, Design Security, and Remote System Upgrades with Arria V Devices* and *Device Datasheet for Arria V Devices* chapters.

For more information about the POR circuitry, refer to "Power-On Reset Circuitry" in the *Device Interfaces and Integration Basics for Arria V Devices* chapter.

Document Revision History

Table 11–3 lists the revision history for this chapter.

Table 11–3. Document Revision History

Date	Version	Changes
February 2012	1.3	Updated V_{CCP} description.
December 2011	1.2	■ Added V_{CCP} information. ■ Updated Table 11–1.
November 2011	1.1	Restructured chapter.
May 2011	1.0	Initial release.

This chapter provides additional information about the document and Altera.

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
Nontechnical support (general)	Email	nacomp@altera.com
(software licensing)	Email	authorization@altera.com

Note to Table:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, **\qdesigns** directory, **D:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicate document titles. For example, *Stratix IV Design Guidelines*.
italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>***.pof** file.
Initial Capital Letters	Indicate keyboard keys and menu names. For example, the Delete key and the Options menu.
"Subheading Title"	Quotation marks indicate references to sections in a document and titles of Quartus II Help topics. For example, "Typographic Conventions."

Visual Cue	Meaning
Courier type	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. The suffix `n` denotes an active-low signal. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
↵	An angled arrow instructs you to press the Enter key.
1., 2., 3., and a., b., c., and so on	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■ ■	Bullets indicate a list of items when the sequence of the items is not important.
	The hand points to information that requires special attention.
	A question mark directs you to a software help system with related information.
	The feet direct you to another document or website with related information.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The envelope links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.

Arria V Device Handbook

Volume 3: Transceivers

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

AV-5V3-1.1

Document last updated for Altera Complete Design Suite version: 11.1
Document publication date: November 2011

Contents

Chapter 4. Transceiver Protocol Configurations in Arria V Devices

Chapter 5. Transceiver Custom Configurations in Arria V Devices

Chapter 6. Transceiver Loopback Support in Arria V Devices

Chapter 7. Dynamic Reconfiguration in Arria V Devices

Additional Information

Chapter Revision Dates

The chapters in this document, Arria V Device Handbook Volume 3: Transceivers, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1. Transceiver Architecture in Arria V Devices
Revised: *November 2011*
Part Number: *AV53001-1.1*

Chapter 2. Transceiver Clocking in Arria V Devices
Revised: *November 2011*
Part Number: *AV53002-1.1*

Chapter 3. Transceiver Reset Control and Power-Down in Arria V Devices
Revised: *November 2011*
Part Number: *AV53003-1.1*

Chapter 4. Transceiver Protocol Configurations in Arria V Devices
Revised: *November 2011*
Part Number: *AV53004-1.1*

Chapter 5. Transceiver Custom Configurations in Arria V Devices
Revised: *November 2011*
Part Number: *AV53005-1.1*

Chapter 6. Transceiver Loopback Support in Arria V Devices
Revised: *November 2011*
Part Number: *AV53006-1.1*

Chapter 7. Dynamic Reconfiguration in Arria V Devices
Revised: *November 2011*
Part Number: *AV53007-1.1*

1. Transceiver Architecture in Arria V Devices

AV53001-1.1

This chapter describes the Arria® V transceiver architecture, channels, and transmitter and receiver channel datapaths.

Altera® 28-nm Arria V FPGAs provide integrated transceivers with the lowest power requirement at 10- and 6-Gigabits per second (Gbps). These transceivers comply with a wide range of protocols and data rate standards.

Arria V devices are available in two variants:

- Arria V GX with integrated backplane-capable transceivers supporting serial data rates between 611 megabits per second (Mbps) and 6.5536 Gbps.
- Arria V GT with integrated backplane-capable transceivers supporting serial data rates between 611 Mbps and 6.5536 Gbps, and has chip-to-chip and chip-to-module capabilities up to 10.3125 Gbps.

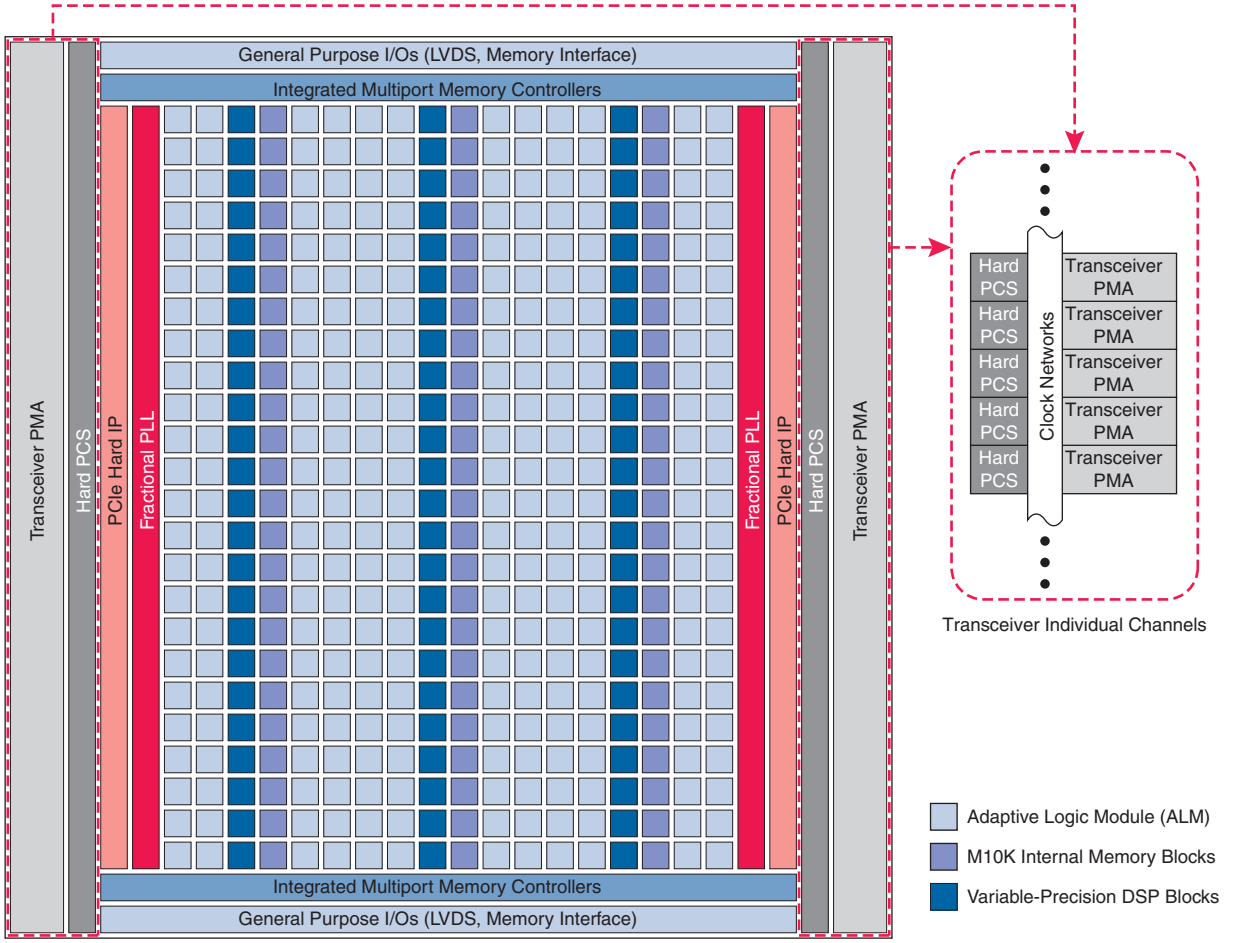
All Arria V transceiver channels are identical and fully duplexed, with a physical coding sublayer (PCS) and physical medium attachment (PMA) layer.

This chapter contains the following sections:

Architecture Overview

Figure 1–1 shows the columns of transceivers on the left and right sides of the Arria V device.

Figure 1–1. Basic Layout of Transceivers in Arria V Devices (1), (2)



Notes to Figure 1–1:

(1) Figure 1–1 represents one variant of an Arria V device. Other variants may have transceivers and PCI Express® (PCIe®) hard IP only on the left side of the device.

(2) Figure 1–1 is a graphical representation of a top view of the silicon die, which corresponds to a reverse view for flip chip packages.

The Arria V hard IP for PCIe implements the PCIe protocol stack including the following layers.

- Physical interface/media access control (PHY/MAC) layer
- Data link layer
- Transaction layer

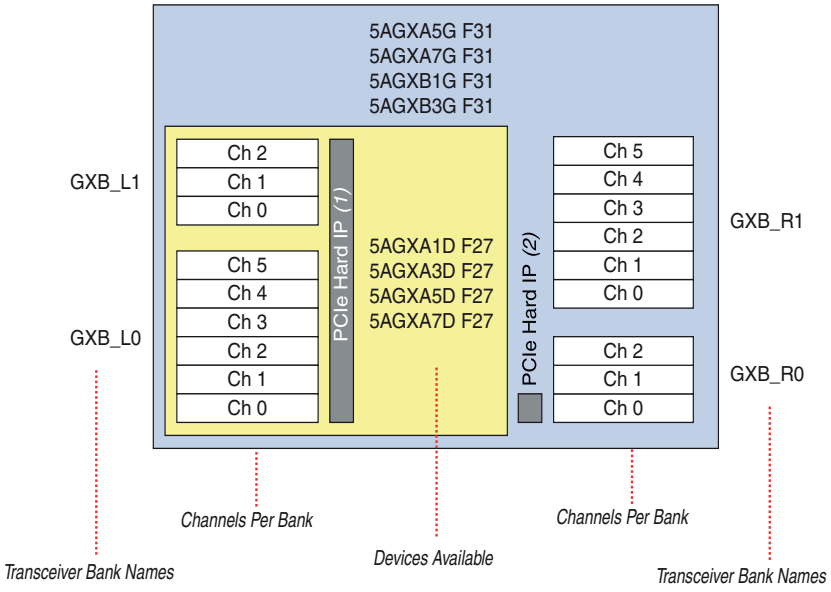
The embedded hard IP saves significant FPGA resources, reduces design risk, and reduces the time required to achieve timing closure. The hard IP complies with the PCI Express Base Specification 1.1 and 2.0 for Gen1 and Gen2 signaling data rates, respectively.

For more information about the PCIe hard IP block architecture, refer to the *Arria V Hard IP for PCI Express User Guide*.

Transceiver Banks

The columns of Arria V transceivers are categorized in banks of three and six channels for 6-Gbps transceivers and in banks of two channels for 10-Gbps transceivers. The transceiver bank boundaries are important for clocking resources, bonding channels, and fitting. Figure 1–2 to Figure 1–5 on page 1–6 show the location of the transceiver banks in Arria V devices. In some package variations, the total transceiver count is reduced.

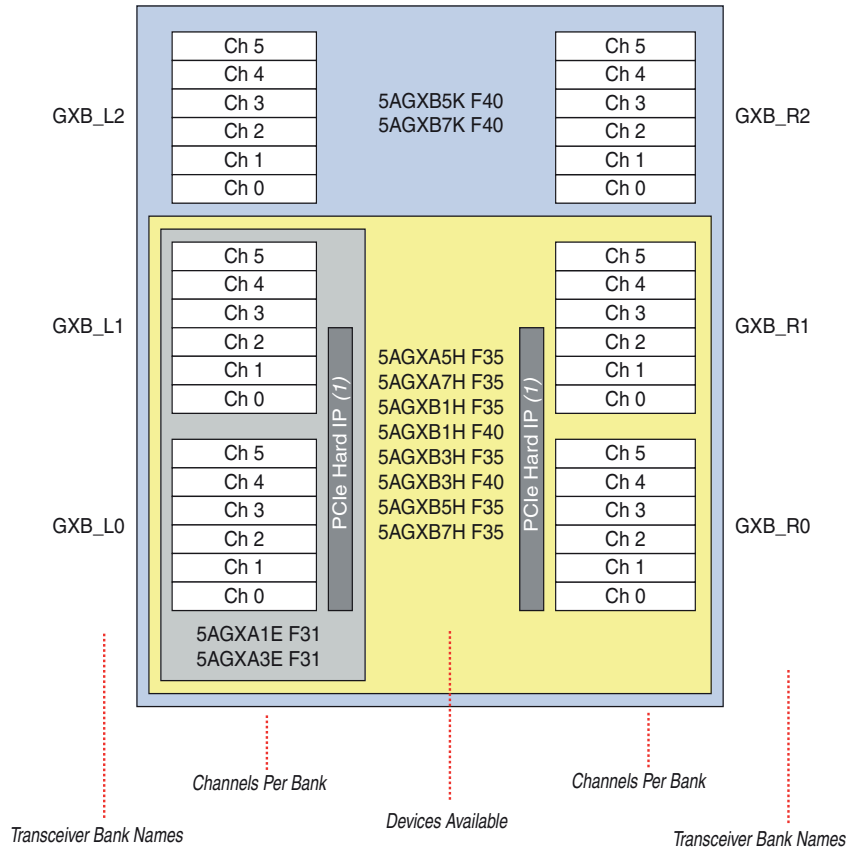
Figure 1–2. Transceiver Bank Location for Arria V GX Devices with 6-Gbps Transceivers (Total of 9 or 18 Channels)



Notes to Figure 1–2:

(1) Supports hard IP implementation of PCIe up to Gen1 x8 and Gen2 x4.
(2) Supports hard IP implementation of PCIe Gen1 x1 and Gen2 x1 only.

Figure 1–3. Transceiver Bank Location for Arria V GX Devices with 6-Gbps Transceivers (Total of 12, 24, or 36 Channels)

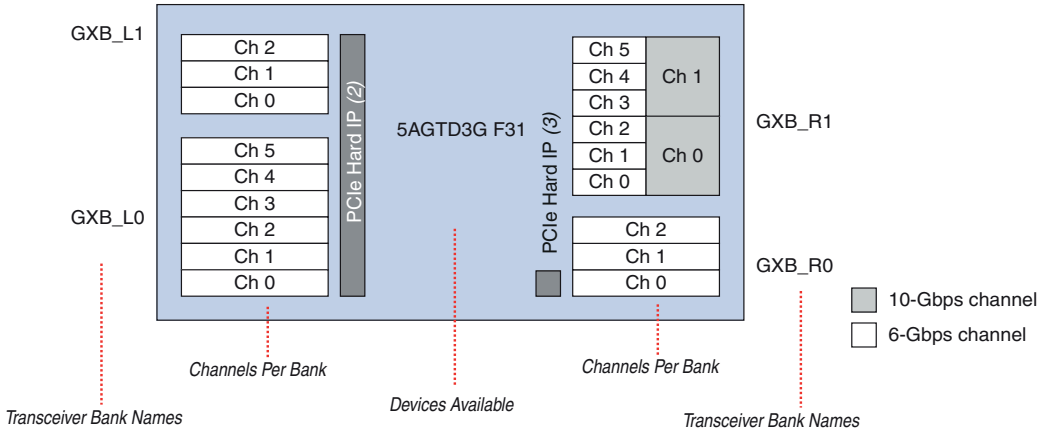


Note to Figure 1–3:

(1) Supports hard IP implementation of PCIe up to Gen1 x8 and Gen2 x4.

In Arria V GT devices as shown in Figure 1–4 and Figure 1–5, if you do not use the 10-Gbps channels in the transceiver bank for data rates above 6.5536 Gbps, you can use both 10-Gbps channels in the bank as six additional 6-Gbps channels.

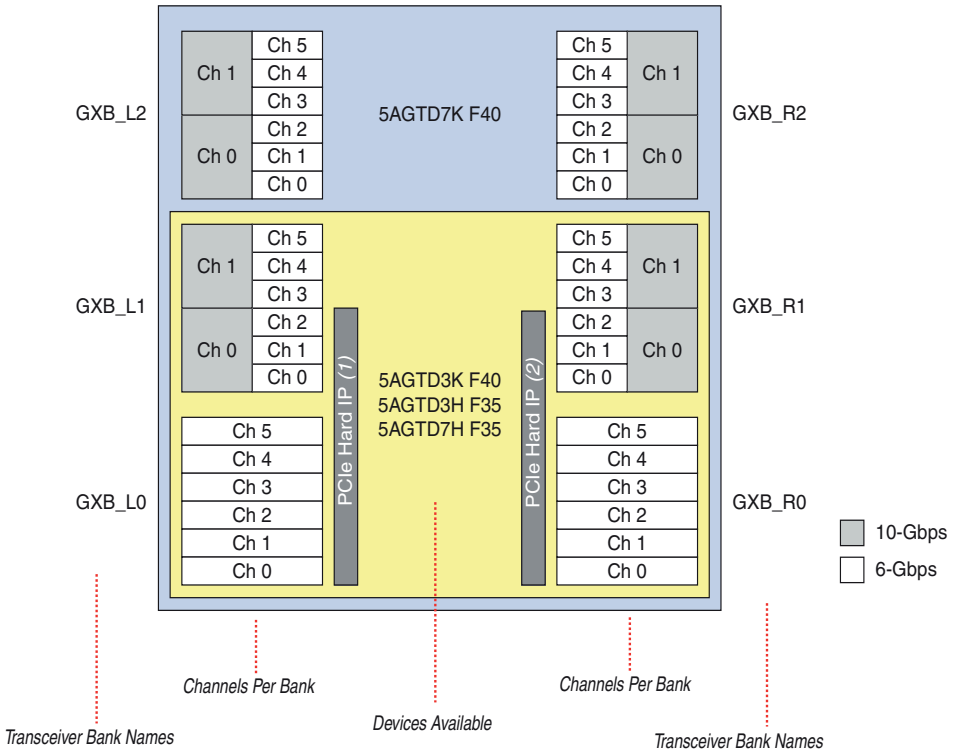
Figure 1–4. Transceiver Bank Location for an Arria V GT Device with 10-Gbps Transceivers (Total of 2 Channels) and 6-Gbps Transceivers (Total of 12 Channels) (1)



Notes to Figure 1–4:

(1) For more information about Arria V GT 10-Gbps transceivers, refer to "10-Gbps Transceiver Channels" on page 1–7.
(2) Supports hard IP implementation of PCIe up to Gen1 x8 and Gen2 x4.
(3) Supports hard IP implementation of PCIe Gen1 x1 and Gen2 x1 only.

Figure 1–5. Transceiver Bank Location for Arria V GT Devices with 10-Gbps Transceivers (Total of 4 or 8 Channels) and 6-Gbps Transceivers (Total of 12 Channels)



Notes to Figure 1–5:

(1) Supports hard IP implementation of PCIe up to Gen1 x4 and Gen2 x4. Gen1 x8 is supported only if you do not use the 10-Gbps channel in GXB_L1 for data rates above 6.5536 Gbps.

(2) Supports hard IP implementation of PCIe up to Gen1 x4 and Gen2 x4. Gen1 x8 is supported only if you do not use the 10-Gbps channel in GXB_R1 for data rates above 6.5536 Gbps.

The 10-Gbps and 6-Gbps transceiver channels are comprised of a transmitter and receiver that can operate individually or simultaneously—providing a full-duplex physical layer implementation for high-speed serial interfacing.

The transmitter and receiver in a channel are structured into PMA and PCS sections:

- PMA—converts serial to parallel data and vice versa for connecting the FPGA to a serial transmission medium.
- PCS—contains hard digital logic implementation that prepares the parallel data for transmission across a physical medium or restores the data to its original form.

The PCS is not available for 10G transceivers.

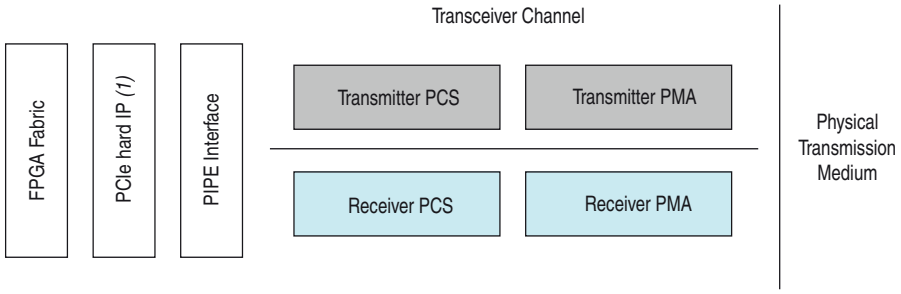
6-Gbps Transceiver Channels

The 6-Gbps transceiver channels support the following interface methods with the FPGA fabric:

- Directly—bypassing the PIPE interface for the PCIe interface and the PCIe hard IP block.
- Through the PIPE interface and the PCIe hard IP block—for hard IP implementation of the PCIe protocol stacks (PHY/MAC, data link layer, and transaction layer).
- Through the PIPE interface but bypassing the PCIe hard IP block—for soft IP implementation of the PCIe protocol stacks.

Figure 1–6 shows a 6-Gbps transceiver channel.

Figure 1–6. 6-Gbps Transceiver Channel for Arria V Devices



Note to Figure 1–6:

(1) Only certain transceiver channels support interfacing to the PCIe hard IP block. For more information, refer to Figure 1–2 on page 1–3 through Figure 1–5 on page 1–6.

You can bond multiple channels to implement a multilane link.

For more information about the transceiver channel and PLL placement restrictions for implementation with the PCIe hard IP block, refer to the *Arria V Hard IP for PCI Express User Guide*.

10-Gbps Transceiver Channels

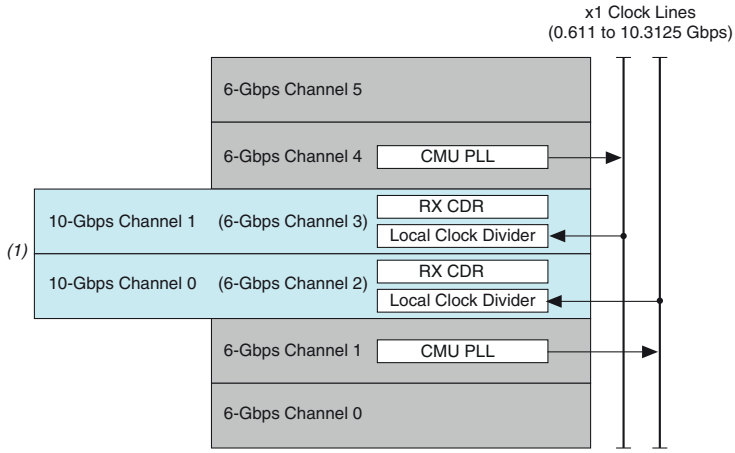
Arria V GT devices offer 10-Gbps-capable channel banks that operate in two different modes—two 10-Gbps transceiver channels or six 6-Gbps transceiver channels. The two 10-Gbps transceiver channels support full-duplex operation.

Each of the following 10-Gbps channels share the same pin resource with its associated 6-Gbps channel:

- 10-Gbps channel 0—associated with 6-Gbps channel 2
- 10-Gbps channel 1—associated with 6-Gbps channel 3

Figure 1–7 shows the full-duplex architecture of the 10-Gbps-capable transceiver banks in Arria V GT devices.

Figure 1–7. 10-Gbps-Capable Transceiver Bank Architecture in Arria V GT Devices



Note to Figure 1–7:

(1) You can use the 10-Gbps channels in a transceiver bank as six 6-Gbps channels.

To achieve full-duplex operation, use the channel phase-locked loop (PLL) resources from the following inactive 6-Gbps transceiver channels:

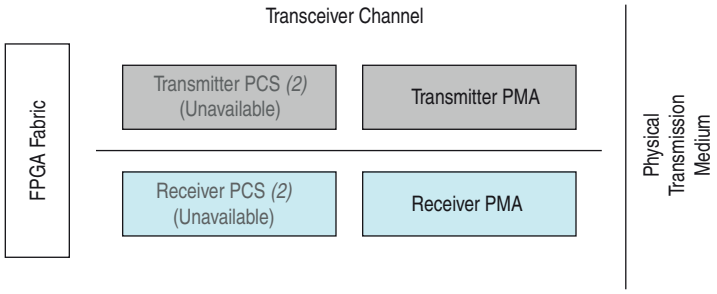
- Channels 1 and 4 provide the dedicated clock multiplier unit (CMU) PLL functionality and generate the high-speed serial clock that drive the x1 clock lines to the 10-Gbps serializer/deserializer (SERDES).
- Channels 2 and 3 use the receiver clock data recovery (CDR) functions.

The 10-Gbps transceiver channels support the following interfacing methods with the FPGA fabric:

- Through the PCS—for data rates of 6.5536 Gbps and lower.
- Directly—bypassing the PCS for data rates above 6.5536 Gbps. Only the PMA is available in the transceiver datapath. You must implement the PCS functions required for the interface with user logic in the FPGA fabric.

Figure 1–8 shows a 10-Gbps transceiver channel.

Figure 1–8. 10-Gbps Transceiver Channel for Arria V Devices (1)



Notes to Figure 1–8:

(1) If you do not use the 10-Gbps channels in a 10-Gbps-capable transceiver bank for serial data rates above 6.5536 Gbps, you can use these channels as 6-Gbps channels that interface with the FPGA fabric through the PCS. For more information, refer to Figure 1–4 on page 1–5 and Figure 1–5 on page 1–6.

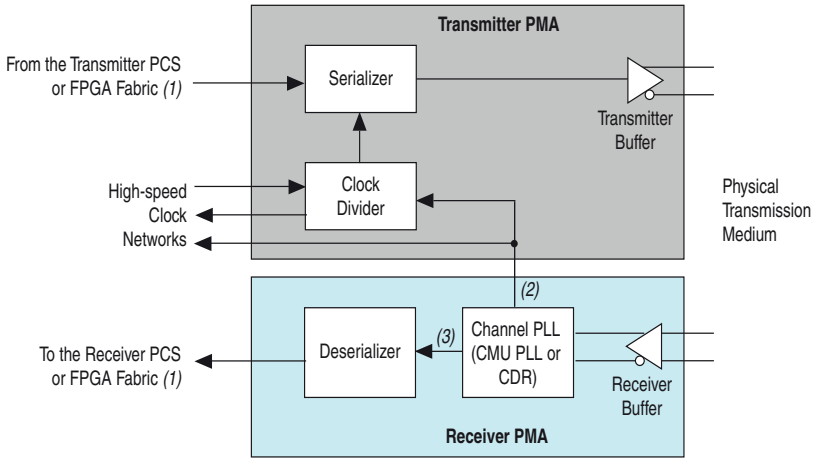
(2) Applicable only if you use the 10-Gbps channel as 6-Gbps channels.

PMA Architecture

The PMA includes the transmitter and receiver datapaths, CMU PLL (configured from the channel PLL), and the clock divider. The analog circuitry and differential on-chip termination (OCT) in the PMA requires the calibration block to compensate for process, voltage, and temperature variations (PVT).

Figure 1–9 shows the PMA transceiver channel for Arria V devices.

Figure 1–9. Transceiver Channel PMA for Arria V Devices



Notes to Figure 1–9:

(1) For 10-Gbps channels, the PMA directly interfaces with the FPGA fabric at serial data rates above 6.5536 Gbps.

(2) The channel PLL provides the serial clock when configured as a CMU PLL.

(3) The channel PLL recovers the clock and serial data stream when configured as a CDR.

Transmitter PMA Datapath

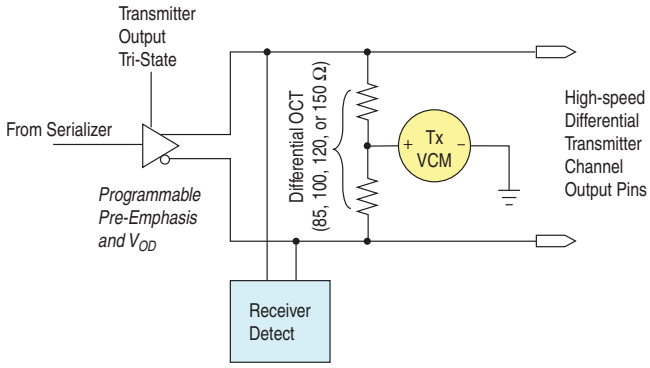
Table 1–1 lists the blocks of the transmitter PMA datapath.

Table 1–1. Functional Blocks in the Transmitter PMA Datapath

Block	Functionality
Serializer	■ Converts the incoming low-speed parallel data from the transmitter PCS to high-speed serial data and sends the data LSB first to the transmitter buffer. ■ Supports 8-, 10-, 16-, and 20-bit serialization factors. ■ Supports the optional polarity inversion and bit reversal features. ■ Supports the 80-bit serialization factor for 10-Gbps transceiver channels for data rates above 6.5536 Gbps.
Transmitter Buffer	■ The 1.5-V pseudo current mode logic (**PCML**) output buffer conditions the high-speed serial data for transmission into the physical medium, as shown in Figure 1–10. ■ Supports the following features: ■ Programmable differential output voltage (V_{OD}) ■ Programmable pre-emphasis ■ Programmable V_{CM} current strength ■ Programmable slew rate ■ On-chip biasing for common-mode voltage (TX V_{CM}) ■ Differential OCT (85, 100, 120 and 150 Ω) ■ Transmitter output tristate (for the PCIe electrical idle function) ■ Receiver detect (for the PCIe receiver detection function)

Figure 1–10 shows the transmitter buffer in Arria V devices.

Figure 1–10. Transmitter Buffer in Arria V Devices



For more information about the specifications for the transmitter buffer features, refer to the *Device Datasheet for Arria V Devices* chapter.

For more information about the features of each block in the transmitter PMA datapath, refer to the "Transmitter PMA Datapath" section in the *Transceiver Basics for Arria V Devices* chapter.

Receiver PMA Datapath

Table 1–2 lists the blocks of the receiver PMA datapath.

Table 1–2. Functional Blocks in the Receiver PMA Datapath

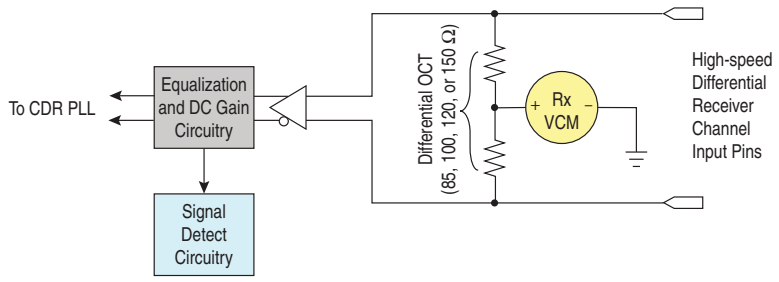
Block	Functionality
Receiver Buffer (1)	■ Receives the serial data stream and feeds the stream to the channel PLL if you configure the channel PLL as a CDR, as shown in Figure 1–11. ■ Supports the following features: ■ Programmable equalization ■ Programmable DC gain ■ Programmable V_{CM} current strength ■ On-chip biasing for common-mode voltage (RX V_{CM}) ■ I/O standard (1.5 V **PCML**, 2.5 V **PCML**, **LVDS**, **LVPECL**) ■ Differential OCT (85, 100, 120 and 150 Ω) ■ Signal detect (for PCIe electrical idle exit and unexpected idle detection).
Channel PLL	■ Recovers the clock and serial data stream if you configure the channel PLL as a CDR. ■ Requires offset cancellation to correct the analog offset voltages. ■ If you do not use the channel PLL as a CDR, you can configure the channel PLL as a CMU PLL for clocking the transceivers. For more information about the channel PLL configured as a CMU PLL, refer to "CMU PLL" on page 1–12.
Deserializer	■ Converts the incoming high-speed serial data from the receiver buffer to low-speed parallel data for the receiver PCS. ■ Receives serial data in LSB-to-MSB order. ■ Supports 8-, 10-, 16-, and 20-bit deserialization factors. ■ Supports the optional clock-slip feature for applications with stringent latency uncertainty requirement. ■ Supports the 80-bit deserialization factor for 10-Gbps transceiver channels for data rates above 6.5536 Gbps.

Note for Figure 1–3:

(1) The receiver can be AC-coupled only. The on-chip or off-chip receiver termination and biasing circuitry automatically restores the required receiver V_{CM} level.

Figure 1–11 shows the receiver buffer in Arria V devices.

Figure 1–11. Receiver Buffer in Arria V Devices



For more information about the specifications for the receiver buffer features, refer to the *Device Datasheet for Arria V Devices* chapter.

For more information about the features of each block in the receiver PMA datapath, refer to the "Receiver PMA Datapath" section in the *Transceiver Basics for Arria V Devices* chapter.

CMU PLL

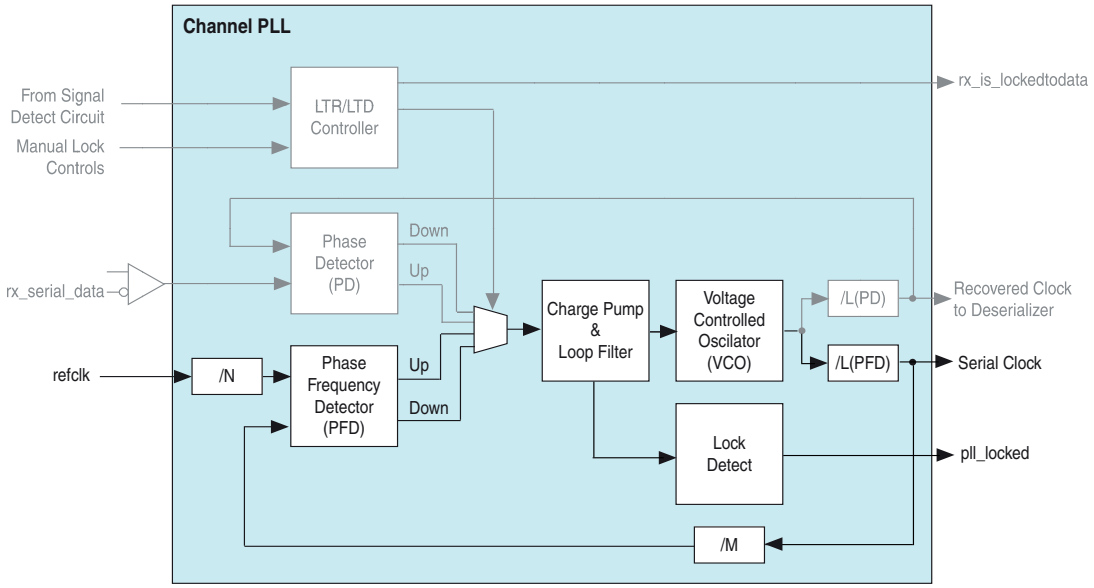
If you do not use the channel PLL as a CDR, you can independently configure every Arria V channel PLL as a CMU PLL for clocking the transceivers.

CDR functionality for the receiver is not available when you configure the channel PLL as a CMU PLL—you can use the transceiver channel only as a transmitter.

The 10-Gbps transceiver channels always support full-duplex operation. The 10-Gbps transmitter CMU PLL is sourced from the channel PLL of the inactive 6-Gbps channels 1 and 4. The 10-Gbps receiver CDR is sourced directly from the channel PLL of the 6-Gbps channels 2 and 3.

The CMU PLL operates only in lock-to-reference (LTR) mode and supports the full range of data rates. Figure 1–12 shows the Arria V channel PLL configured as a CMU PLL.

Figure 1–12. CMU PLL in Arria V Devices



For more information about the architecture of the channel PLL, refer to the "Channel PLL Architecture" section in the *Transceiver Basics for Arria V Devices* chapter.

Using the input reference clock, the CMU PLL synthesizes the serial clock with a frequency that is half of the data rate. The CMU PLL output serial clock feeds the clock divider that resides in the transmitter of the same transceiver channel. Depending on the channel location in a transceiver bank, the CMU PLL of channels 1 and 4 feeds the output clock to the x1 clock lines.

For more information about the input reference clock and transmit PLL, refer to the *Transceiver Clocking in Arria V Devices* chapter.

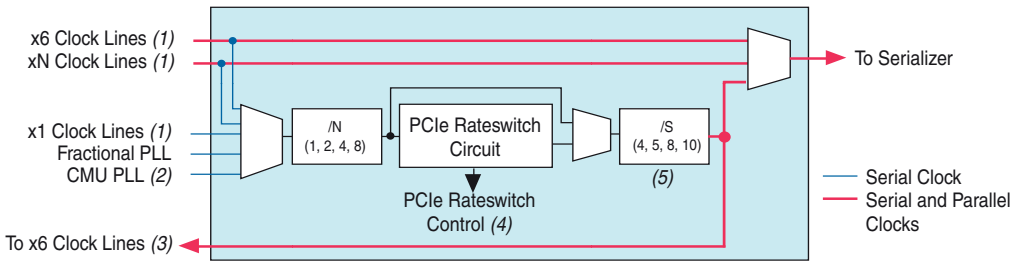
Clock Divider

Each Arria V transmitter channel has a clock divider. There are two types of clock dividers, depending on the channel location in a transceiver bank:

- Local clock divider—channels 0, 2, 3, and 5 provide serial and parallel clocks to the PMA
- Central clock divider—channels 1 and 4 can drive the x6 clock lines

Figure 1–13 shows the clock divider for a 6-Gbps channel in Arria V devices.

Figure 1–13. Clock Divider for a 6-Gbps Channel in Arria V Devices



Notes to Figure 1–13:

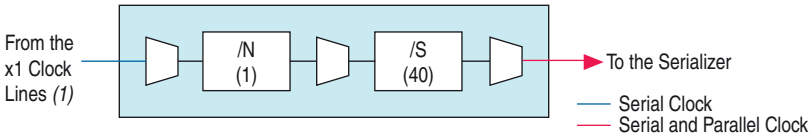
(1) For information about the x1, x6, and xN clock lines, refer to the *Transceiver Clocking in Arria V Devices* chapter.
(2) Only from the channel PLL in the same transceiver channel configured as a CMU PLL.
(3) Applicable for central clock dividers only (clock dividers in channels 1 and 4).
(4) The PCIe rateswitch circuit allows dynamic switching between Gen2 and Gen1 line rates in a PCIe Gen2 design.
(5) The divider settings are configured automatically depending on the serialization factor. The selected divider setting is half of the serialization factor.

Both types of clock dividers can divide the serial clock input to provide the parallel and serial clocks for the serializer in the channel if you use clocks from the clock lines or transmit PLLs. The central clock divider can additionally drive the x6 clock lines used to bond multiple channels.

In bonded channel configurations, both types of clock dividers can feed the serializer with the parallel and serial clocks directly, without dividing them from the x6 or xN clock lines.

Figure 1–14 shows the clock divider for a 10-Gbps channel in Arria V devices.

Figure 1–14. Clock Divider for a 10-Gbps Channel in Arria V Devices



Note to Figure 1–14:

(1) The CMU PLL that drives the x1 clock line is sourced from the adjacent CMU PLL of the inactive 6-Gbps transceiver channels 1 or 4, depending on the location of the 10-Gbps channel.

Each 10-Gbps transmitter channel receives the serial clock exclusively from its dedicated CMU PLL. Because the 10-Gbps transmitter channels 0 and 1 are associated with the 6-Gbps channels 2 and 3, respectively, the 10-Gbps transmitter channels use the local clock dividers only and do not use the central clock dividers. The local clock divider divides the serial clock input to provide the parallel clock and high-speed serial clock for the serializer in the channel.

Calibration Block

Up to two calibration blocks are available for the Arria V transceiver PMA. The calibration block calibrates the differential OCT resistance and analog circuitry in the transceiver PMA to ensure the functionality is independent of PVT.

Figure 1–15 and Figure 1–16 show the calibration block location, the transceiver banks calibrated by the calibration block, and the required external connection on the RREF pin.

Figure 1–15. Calibration Block Location and Connections in Arria V Devices with Transceivers on the Left Side of the Device Only

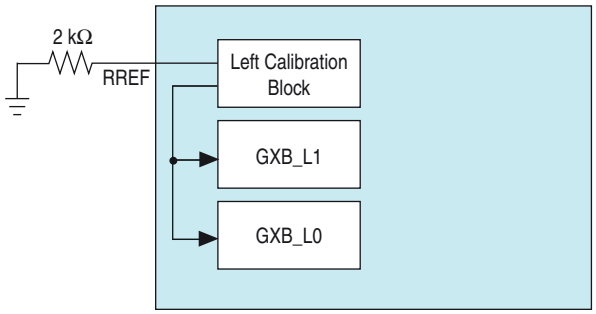
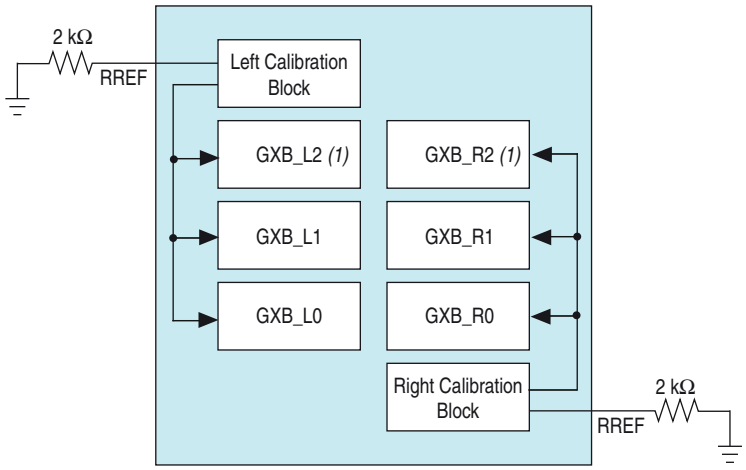


Figure 1–16. Calibration Block Location and Connections in Arria V Devices with Transceivers on Both Sides of the Device



Note to Figure 1–16:

(1) The right transceiver bank is available only in Arria V 5AGXB5KF40, 5AGXB7KF40, and 5AGTD7KF40 devices.

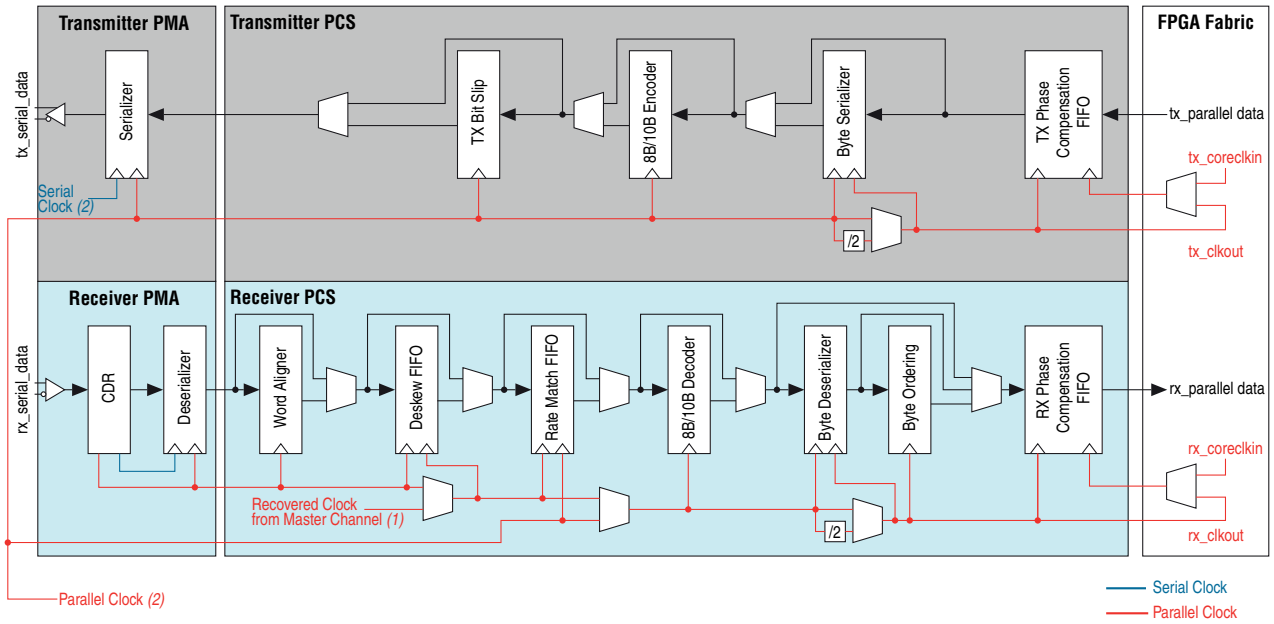
The calibration block generates a constant internal reference voltage that is independent of PVT variations using the external reference resistor. The resulting reference currents are used to calibrate the transceiver banks.

You must connect a separate 2 kΩ (tolerance maximum of ±1%) external resistor on each RREF pin to ground. To ensure the calibration block operates properly, the RREF resistor connection in the board must be free from external noise.

PCS Architecture

Figure 1–17 shows the PCS transceiver channel in Arria V devices.

Figure 1–17. Transceiver Channel PCS in Arria V Devices



Notes to Figure 1–17:

(1) Used for XAUI configurations only. For more information, refer to the *Transceiver Protocol Configurations in Arria V Devices* chapter.

(2) The serial and parallel clocks are sourced from the clock divider.

The PCS is not available when using the 10-Gbps channels; only the PMA is available. You must implement the PCS functions required for the interface using user logic in the FPGA fabric with an 80-bit FPGA fabric–transceiver interface width.

Table 1–3 lists the two transceiver channel PCS datapath configurations based on the transceiver channel PMA-PCS width (or the SERDES factor).

Table 1–3. PCS Datapath Configurations

Parameter	Single-Width	Double-Width
PMA–PCS Interface Width	8 or 10 bit	16 or 20 bit
FPGA Fabric–Transceiver Interface Width	8 or 10 bit 16 or 20 bit (1)	16 or 20 bit 32 or 40 bit (1)

Note to Table 1–3:

(1) The byte serializer and deserializer are enabled.

Transmitter PCS Datapath

Table 1–4 lists the blocks of the Arria V transmitter PCS datapath.

Table 1–4. Functional Blocks in the 6-Gbps Transmitter PCS Datapath

Block	Functionality
Transmitter Phase Compensation FIFO	■ Compensates for the phase difference between the low-speed parallel clock and the FPGA fabric interface clock when interfacing the transmitter PCS with the FPGA fabric directly or with the PCIe hard IP block ■ Supports operation in phase compensation and registered modes
Byte Serializer	■ Divides the FPGA fabric–transceiver interface frequency in half at the transmitter channel by doubling the transmitter input datapath width ■ Allows the transmitter channel to operate at higher data rates with the FPGA fabric–transceiver interface frequency that is within the maximum limit ■ Supports operation in single- and double-width modes
8B/10B Encoder	■ Generates 10-bit code groups from 8-bit data and 1-bit control identifier, in compliance with Clause 36 of the IEEE 802.3 specification ■ Supports operation in single- and double-width modes, and running disparity control
Transmitter Bit-Slip	■ Enables user-controlled, bit-level delay in the data prior to serialization for serial transmission ■ Supports operation in single- and double-width modes

For more information about the transmitter PCS datapath, refer to the "Transmitter PCS Datapath" section in the *Transceiver Basics for Arria V Devices* chapter.

Receiver PCS Datapath

Table 1–5 lists the blocks of the Arria V receiver PCS datapath.

Table 1–5. Functional Blocks in the 6-Gbps Receiver PCS Datapath (Part 1 of 2)

Block	Functionality
Word Aligner	■ Searches for a predefined alignment pattern in the deserialized data to identify the correct boundary and restores the word boundary during link synchronization ■ Supports an alignment pattern length of 7, 8, 10, 16, 20, or 32 bits ■ Supports operation in four modes—manual alignment, bit-slip, automatic synchronization state machine, and deterministic latency state machine—in single- and double-width configurations ■ Supports the optional programmable run-length violation detection, polarity inversion, bit reversal, and byte reversal features
Deskew FIFO	■ Aligns code groups between four bonded receiver channels ■ Supports operations that are compliant to the XAUI protocol
Rate Match FIFO	■ Compensates for small clock frequency differences of up to ±300 parts per million (ppm)—600 ppm total—between the upstream transmitter and the local receiver clocks by inserting or deleting skip symbols when necessary ■ Supports operation that is compliant to the clock rate compensation function in supported protocols
8B/10B Decoder	■ Receives 10-bit data and decodes the data into an 8-bit data and a 1-bit control identifier—in compliance with Clause 36 of the IEEE 802.3 specification ■ Supports operation in single- and double-width modes

Table 1–5. Functional Blocks in the 6-Gbps Receiver PCS Datapath (Part 2 of 2)

Block	Functionality
Byte Deserializer	■ Divides the FPGA fabric–transceiver interface frequency in half at the receiver channel by doubling the receiver output datapath width ■ Allows the receiver channel to operate at higher data rates with the FPGA fabric–transceiver interface frequency that is within the maximum limit ■ Supports operation in single- and double-width modes
Byte Ordering	■ Searches for a predefined pattern that must be ordered to the LSByte position in the parallel data going to the FPGA fabric when you enable the byte deserializer
Receiver Phase Compensation FIFO	■ Compensates for the phase difference between the low-speed parallel clock and the FPGA fabric interface clock when interfacing the receiver PCS with the FPGA fabric directly or with the PCIe hard IP block ■ Supports operation in phase compensation and registered modes

For more information about the receiver PCS datapath, refer to the "Receiver PCS Datapath" section in the *Transceiver Basics for Arria V Devices* chapter.

Channel Bonding

The following factors contribute to the transmitter channel-to-channel skew:

- High-speed serial and low-speed parallel clock skew between channels
- Unequal latency in the transmitter phase compensation FIFO

Bonded transmitter datapath clocking provides low channel-to-channel skew when compared with non-bonded channel configurations.

For more information about the bonded and non-bonded channel clocking, refer to the *Transceiver Clocking in Arria V Devices* chapter.

Bonded Channel Configurations

In bonded channel configurations, the serial and parallel clocks are generated by the transmit PLL and the central clock divider.

There is equal latency in the transmitter phase compensation FIFO of all bonded channels because they share common pointers and control logic generated in the central clock divider.

The lower transceiver clock skew and equal latency in the transmitter phase compensation FIFO in all channels result in a lower channel-to-channel skew.

Bonded channel configurations are available only for 6-Gbps transceivers. You can bond (up to) all channels on the same side.

Non-Bonded Channel Configurations

In a non-bonded channel configuration, the parallel clock in each channel is generated independently by its local clock divider.

There may be unequal latency in the transmitter phase compensation FIFO of each channel because each channel has its own pointers and control logic. The higher transceiver clock skew and unequal latency in the transmitter phase compensation FIFO in each channel may result in a higher channel-to-channel skew.

Non-bonded channel configurations are available for 10-Gbps and 6-Gbps transceivers.

PLL Sharing

In a Quartus® II design, you can merge two different protocol configurations to share the same CMU PLL resources. These configurations must fit in the same transceiver bank. The input `refclk` and PLL output frequencies must be identical.

Clock merging is not applicable to 10-Gbps transceiver channels, because they are driven by a dedicated CMU PLL resource.

Document Revision History

Table 1–6 lists the revision history for this chapter.

Table 1–6. Document Revision History

Date	Version	Changes
November 2011	1.1	■ Updated the maximum data rate to 6.5536 Gbps. ■ Changed the "IP Compiler for PCI Express User Guide" links to the "Arria V Hard IP for PCI Express User Guide" links. ■ Updated the "Transceiver Banks", "10-Gbps Transceiver Channels", "CMU PLL", "Clock Divider", "Calibration Block" sections. ■ Updated Table 1–1, Table 1–2, and Table 1–5. ■ Updated Figure 1–1, Figure 1–2, Figure 1–3, Figure 1–4, Figure 1–5, Figure 1–7, Figure 1–9, Figure 1–13, and Figure 1–17. ■ Removed Table 1-3 and Table 1-4. ■ Minor text edits.
August 2011	1.0	Initial release.

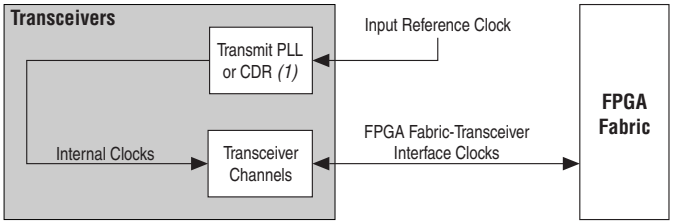
2. Transceiver Clocking in Arria V Devices

AV53002-1.1

This chapter provides information about the Arria® V transceiver clocking architecture. The chapter describes the clocks that are required for transceiver operation, the transceiver's internal clocking architecture, and the available clocking options when the transceiver interfaces with the FPGA fabric.

Figure 2–1 shows an overview of the clocking architecture.

Figure 2–1. Transceiver Clocking Architecture Overview



Note to Figure 2–1

(1) The transmit phase-locked loop (PLL) can be a CMU PLL (channel PLL) or a fractional PLL.

This chapter contains the following sections:

Input Reference Clocking

This section describes how the reference clock for the channel PLL is provided to generate the clocks required for transceiver operation.

Each 6-Gbps transceiver channel has a channel PLL that you can configure in one of the following schemes:

- Transmitter clock multiplier unit (CMU) PLL—independently synthesizes the input reference clock to generate the high-speed serial clock for clocking the transceivers.
- Receiver clock data recovery (CDR)—independently recovers the clock and data from the incoming serial data.

The CDR is not available when the channel PLL is configured as a CMU PLL. Without a CDR, you can use the channel only as a transmitter channel. Alternatively, you can use a fractional PLL as the transmit PLL, which would allow you to use the channel PLL as a CDR.

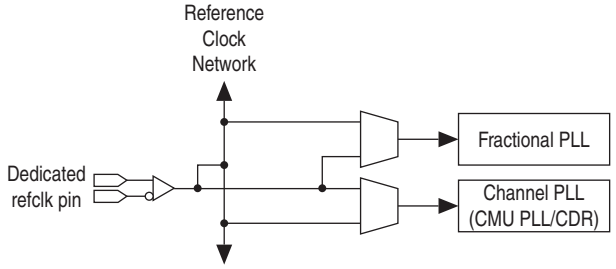
Only integer frequency synthesis is allowed if the fractional PLL is used as a transmit PLL.

10-Gbps transceiver channels always support full-duplex operation. Each 10-Gbps transceiver channel configures its channel PLL as a CDR PLL for receiver operation, and uses a serial clock provided by the CMU PLL of an adjacent and inactive 6-Gbps channel for transmitter operation. Fractional PLLs are not available for 10-Gbps transceivers.

For more information about configuring the channel PLL as a CMU PLL or CDR, refer to the *Transceiver Architecture in Arria V Devices* chapter.

The transceiver channel PLL and fractional PLL derive the input clock from a dedicated `refclk` pin or from the reference clock network. Figure 2–2 shows an overview of the input to the transceiver channel from the dedicated input reference clock.

Figure 2–2. Input Reference Clock Sources to Transceiver Channel



Dedicated Reference Clock Pins

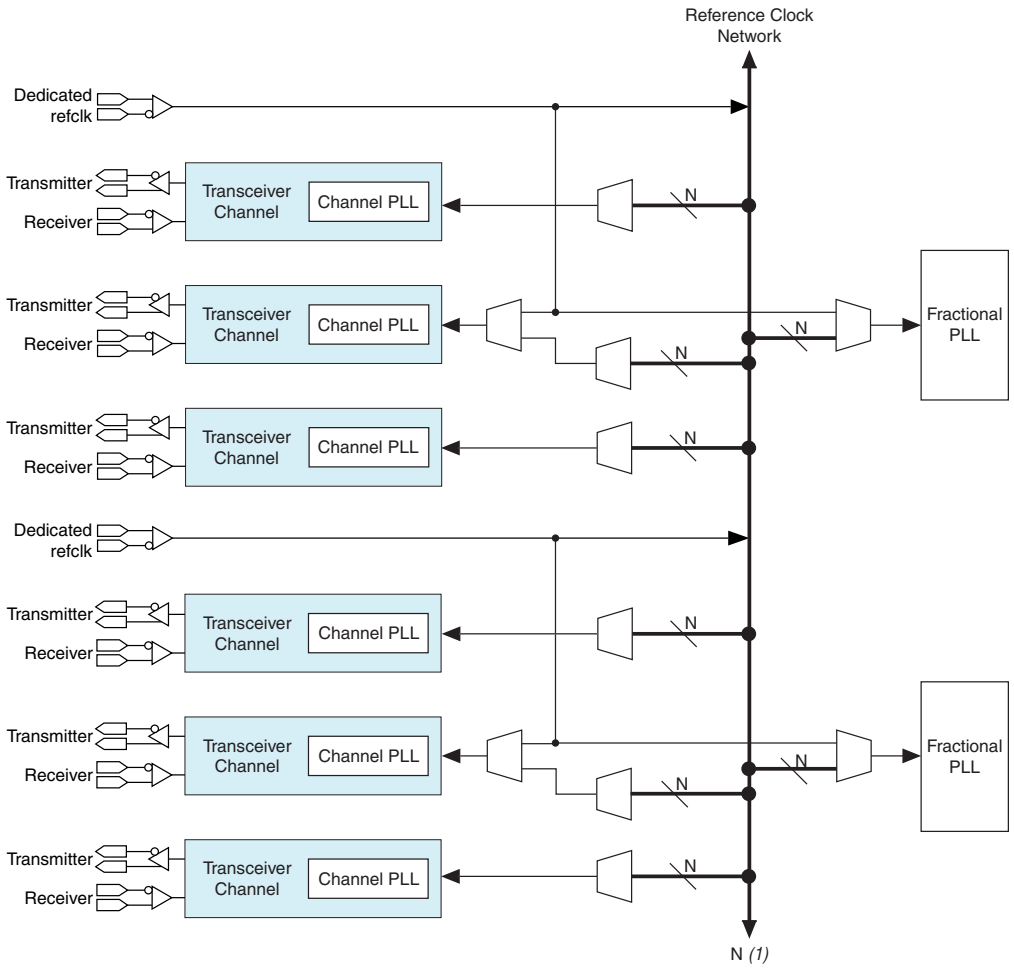
Arria V devices have one dedicated `refclk` pin for each 10-Gbps transceiver channel, or each group of three 6-Gbps transceiver channels. Every dedicated reference clock pin drives a reference clock network spanning the side of the device.

For specifications about the input frequency supported by the `refclk` pins, refer to the *Device Datasheet for Arria V Devices* chapter.

Dedicated Reference Clock Using the Reference Clock Network

A single dedicated `refclk` pin can provide the reference clock to multiple channel PLLs or fractional PLLs that require the same clock frequency through the reference clock network. Figure 2–3 shows the input reference clock sources for a 6-Gbps transceiver bank and two fractional PLLs.

Figure 2–3. Input Reference Clock Sources for 6-Gbps Transceiver Channels in a Transceiver Bank

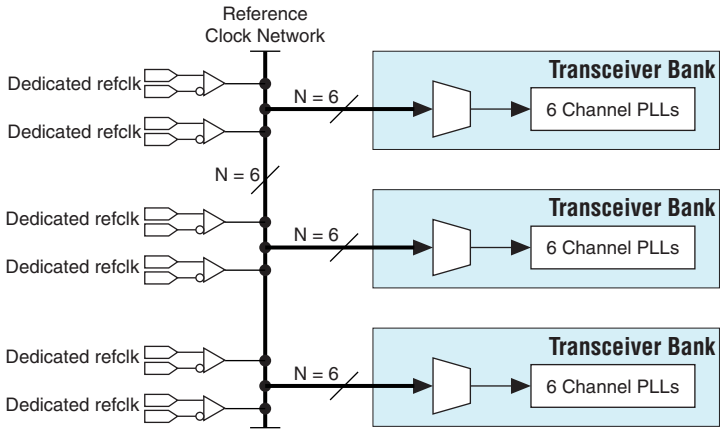


Note to Figure 2–3:

(1) N is the number of dedicated `refclk` pins, which is equal to the number of transceiver channels on a side divided by 3.

Figure 2–4 shows the input reference clock sources for eighteen 6-Gbps channel PLLs on the left side of a 5AGXB5KF40 device. For the eighteen 6-Gbps channels, the total number of clock lines is six (N = 18/3). There are similar input reference clock resources on the right side of the device.

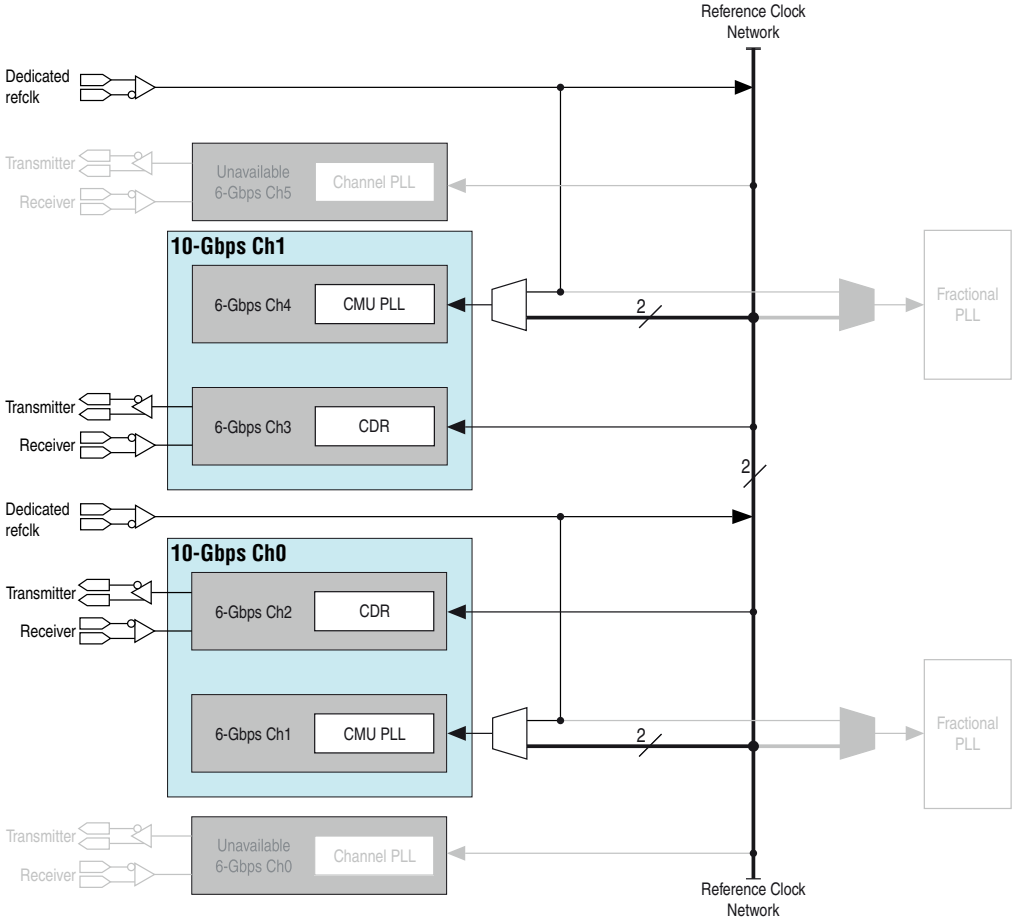
Figure 2–4. Input Reference Clock Sources in a 5AGXB5KF40 Device



Arria V GT devices use the same dedicated `refclk` input pins and reference clock network sources as the 6-Gbps transceiver channels for each of the 10-Gbps channel PLLs.

Figure 2–5 shows the input reference clock sources for two full-duplex 10-Gbps channels. 10-Gbps channel 0 is 6-Gbps channel 2 and 10-Gbps channel 1 is 6-Gbps channel 3. The two 10-Gbps channels configure their channel PLLs as CDRs for receiver functionality. For transmitter functionality, 10-Gbps channel 0 uses the CMU PLL from 6-Gbps channel 1, while 10-Gbps channel 1 uses the CMU PLL from 6-Gbps channel 4.

Figure 2–5. Input Reference Clock Sources for Two 10-Gbps Transceiver Channels in a 10 Gbps-capable Transceiver Bank *(1)*

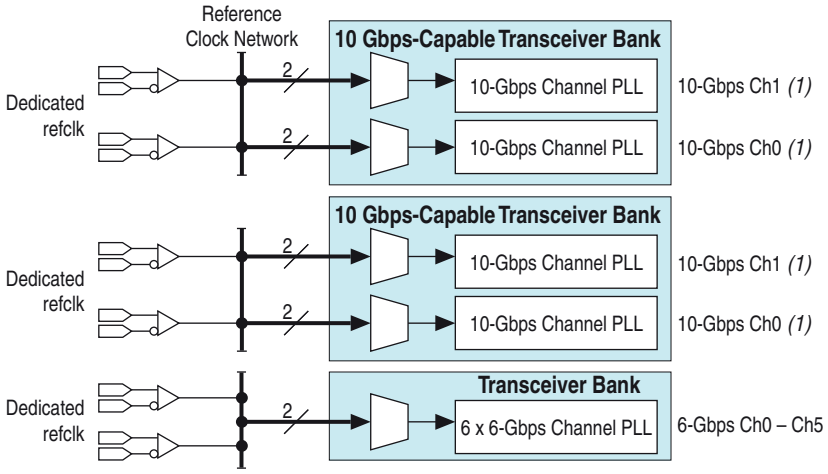


Notes to Figure 2–5:

(1) You can use the transceiver channels in a 10 Gbps-capable transceiver bank as six 6-Gbps channels.

There are six dedicated reference clock inputs for each side (left and right) of a 5AGTD7KF40 device—four are for the two 10 Gbps-capable transceiver banks, and two are for the standard 6-Gbps transceiver banks.

Figure 2–6. Input Reference Clock Sources for Left or Right Side of the 5AGTD7KF40 Device



Notes to Figure 2–6:

(1) You can use the transceivers in a 10 Gbps-capable transceiver bank as six 6-Gbps channels.

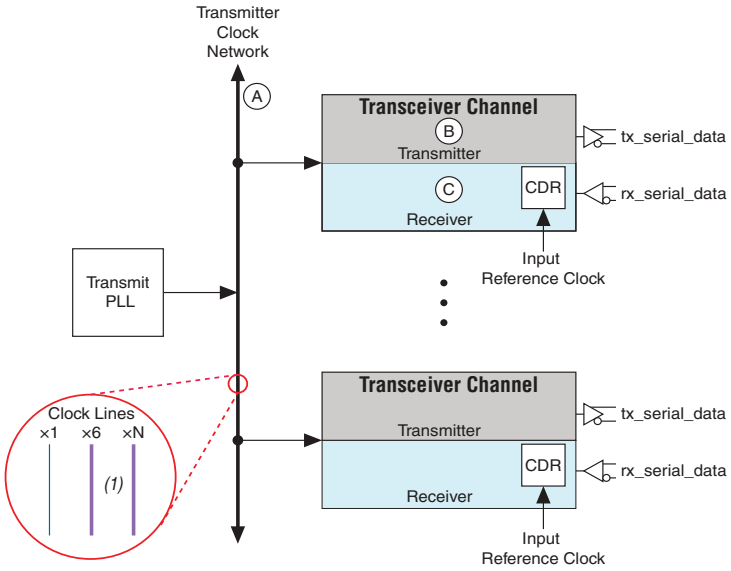
Internal Clocking

This section describes the clocking architecture internal to Arria V transceivers. Different physical coding sublayer (PCS) configurations and channel bonding options result in various transceiver clock paths.

The following labels, as shown in Figure 2–7, mark the sections of the transceiver internal clocking:

- **A**—Transmitter Clock Network
- **B**—Transmitter Clocking
- **C**—Receiver Clocking

Figure 2–7. Internal Clocking



Note to Figure 2–7:

(1) The x6 and xN clock lines are supported only by the Arria V 6-Gbps transceivers.

For the transmitter channel, the CMU PLL or fractional PLL (fractional PLLs are available only in 6-Gbps transceiver channels) use the input reference clock. The CMU PLL provides a serial clock to the transmitter clock network, which is distributed to the transceiver channels.

This section describes clocking that is internal to the transceiver. The Quartus® II software primarily performs the clock routing based on the transceiver configuration that you select.

Transmitter Clock Network

The transmitter clock network routes the clock from the transmit PLL to the transmitter channel, as shown in Figure 2–7. A clock divider provides two clocks to the transmitter channel:

- Serial clock—high-speed clock for the serializer
- Parallel clock—low-speed clock for the serializer and the PCS

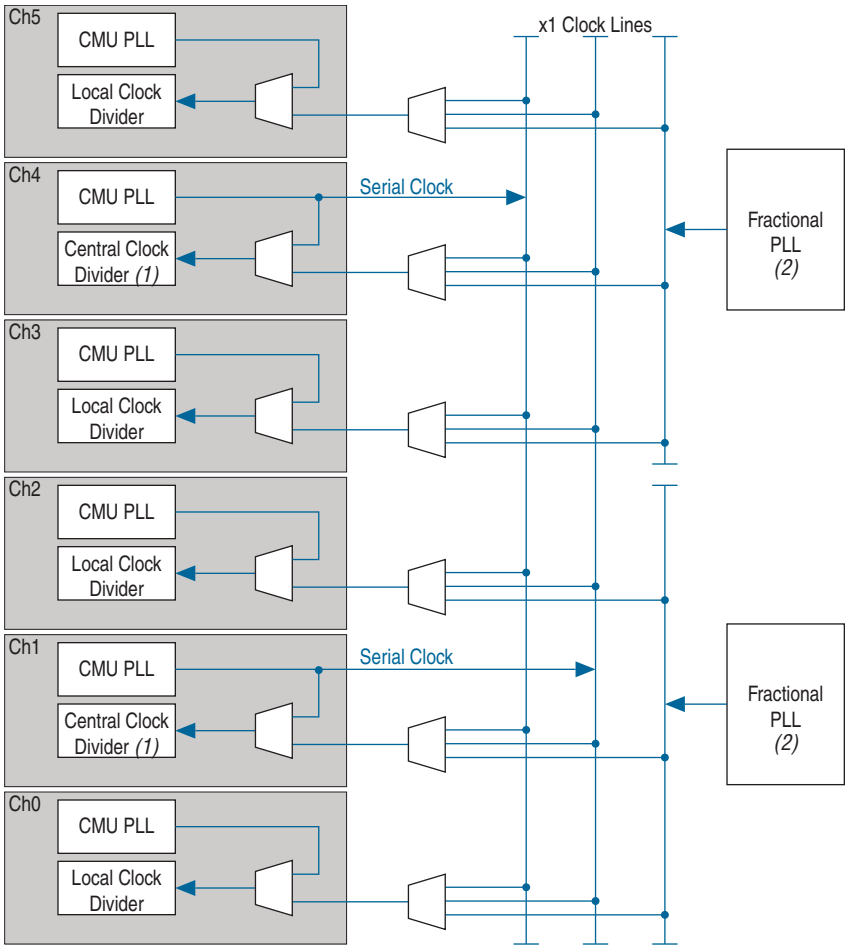
Arria V transceivers support non-bonded and bonded transceiver clocking configurations:

- Non-bonded configuration—Only the serial clock from the transmit PLL is routed to the transmitter channel. The clock divider of each channel generates the local parallel clock. The x1 clock line is used for non-bonded configurations. This configuration is available for both the 6-Gbps and 10-Gbps transceivers.
- Bonded configuration—Both the serial clock and parallel clock are routed from the central clock divider in channel 1 or 4 to the bonded transmitter channels. The x6 and xN clock lines are used for bonded configurations. This configuration is only available for 6-Gbps transceivers.

The Quartus II software performs the clock routing related to the transmitter clock network based on the transceiver configuration that you select.

Figure 2–8 shows the x1 clock lines used by a 6-Gbps transceiver bank.

Figure 2–8. x1 Clock Lines Used for Non-Bonded Configuration in a 6-Gbps Transceiver Bank



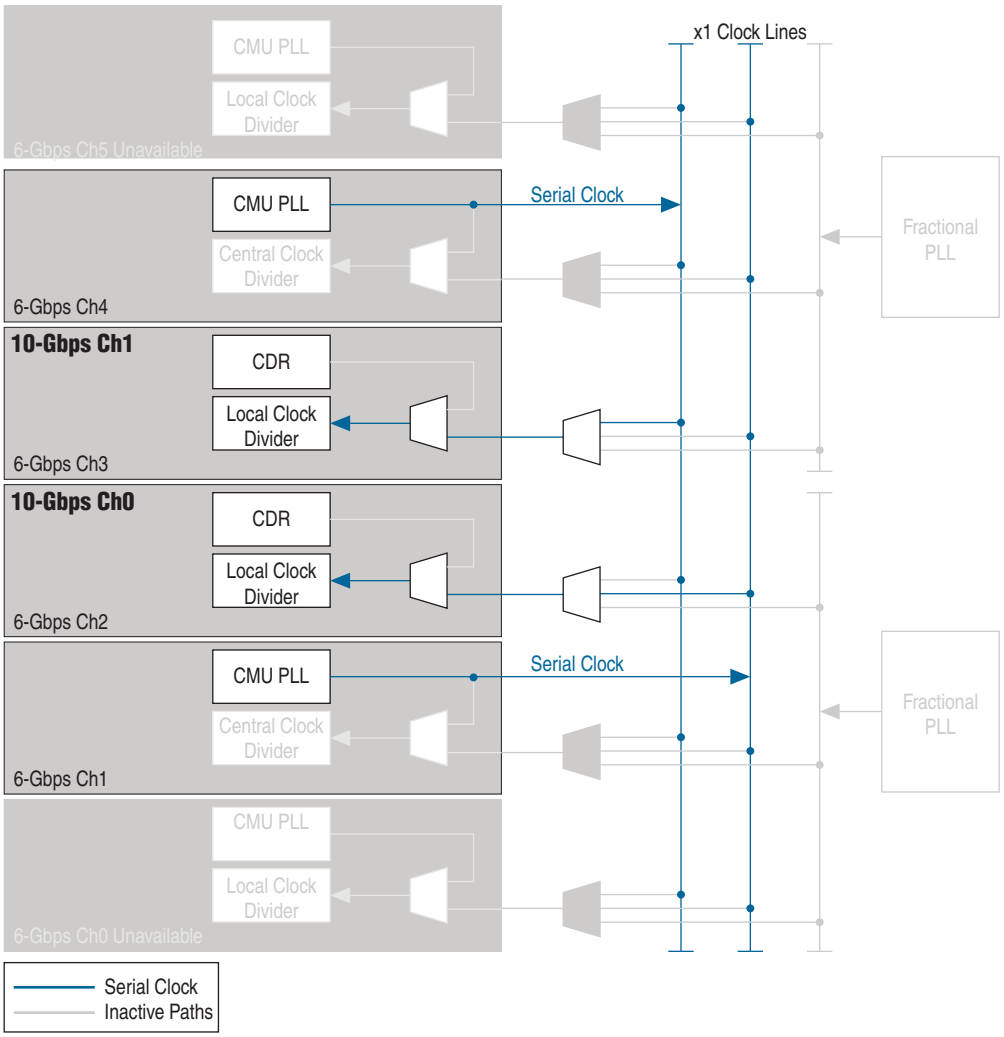
Notes to Figure 2–8:

(1) You can use the central clock divider as a local clock divider.

(2) One fractional PLL provides a clock to the x1 clock line of only three channels.

Figure 2–9 shows the x1 clock lines for a 10-Gbps transceiver bank.

Figure 2–9. 10-Gbps Transceiver Bank Operation and x1 Clock Lines Used for a Non-Bonded Configuration *(1)*



Note to Figure 2–9:

(1) This figure is applicable to the Arria V 5AGTD7GF31, 5AGTD7HF35, and 5AGTD7KF40 devices.

The x1 clock lines are driven by the following resources in a transceiver bank:

- 6-Gbps transceivers—the channel PLLs of channels 1 and 4 that are configured as CMU PLLs, or the fractional PLLs
- 10-Gbps transceivers—the channel PLLs of the 6-Gbps channels 1 and 4 that are configured as CMU PLLs

The x1 clock lines can drive the clock divider of each channel within a transceiver bank. For the 10-Gbps transceiver operation, the x1 clock lines can drive only the local dividers of the 10-Gbps channel 0 (6-Gbps channel 2) and the 10-Gbps channel 1 (6-Gbps channel 3) within the transceiver bank.

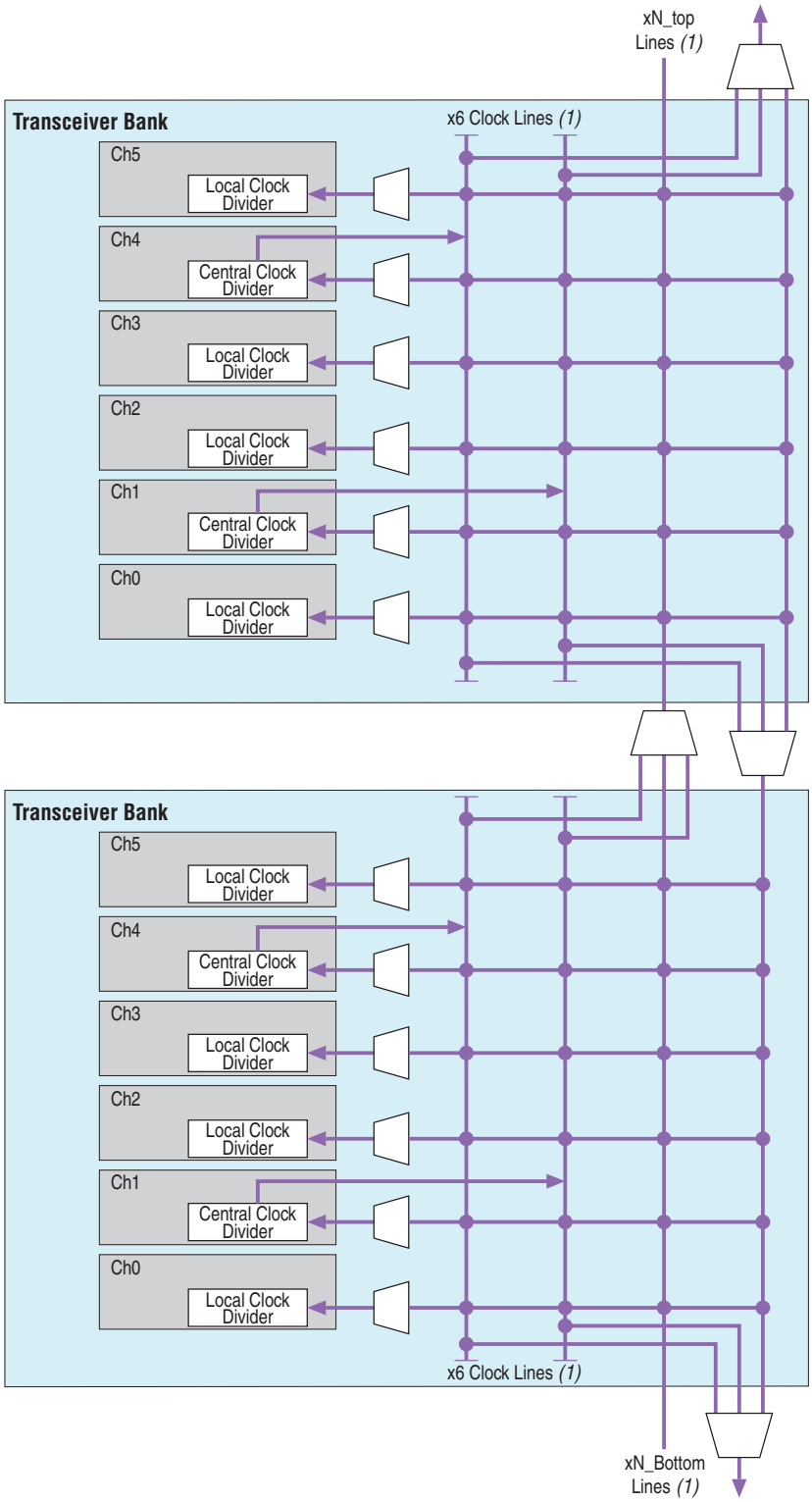
Bonded configurations use the x6 and xN clock lines. These clock lines route the serial and parallel clock from the central clock dividers of channel 1 or 4 when using 6-Gbps transceiver channels.

To conserve the number of transmit PLLs in your design, you can use the xN clock lines, which are driven by the x6 clock lines, to route the serial clock from the central clock dividers in a transceiver bank to the transceiver channels in another transceiver bank for non-bonded configurations.

Figure 2–10 shows the x6 and xN clock lines for 6-Gbps transceiver banks. You can apply the following conditions to the clock lines:

- Drive the x6 clock lines with the central clock divider of channels 1 and 4 in a transceiver bank.
- Use the x6 clock lines to drive the xN clock lines.
- Use the x6 clock lines to drive any channel within a transceiver bank.
- Use the xN clock lines to drive any channel on the respective side, because the xN clock lines span the entire side of the device.

Figure 2–10. x6 and xN Clock Lines in 6-Gbps Transceivers



Note to Figure 2–10:

(1) The clock lines carry both the serial and parallel clocks.

Table 2–1 lists the span and data rates supported by the clock sources and networks in Arria V devices.

Table 2–1. Data Rates and Spans Supported Using Arria V Clock Sources and Networks

Transceiver	Clock Network	Clock Source	Maximum Data Rate (1)	Bonding	Span
6-Gbps	x1	Ch1 or Ch4 CMU PLL in a transceiver bank	6.375 Gbps	No	Within transceiver bank
		Fractional PLLs in a transceiver bank	3.125 Gbps		
	x6	Central clock dividers in a transceiver bank (in Ch1 or Ch4 only)	6.375 Gbps	Yes	Transceiver bank
	xN	Central clock dividers in a transceiver bank through x6 clock lines (in Ch1 or Ch4 only)	3.125 Gbps	Yes	Side wide
10-Gbps	x1	Inactive 6-Gbps Ch1 and Ch4 used as a dedicated CMU PLL in the transceiver bank	10.3125 Gbps	No	Specific to 10-Gbps Channel

Note to Table 2–1:

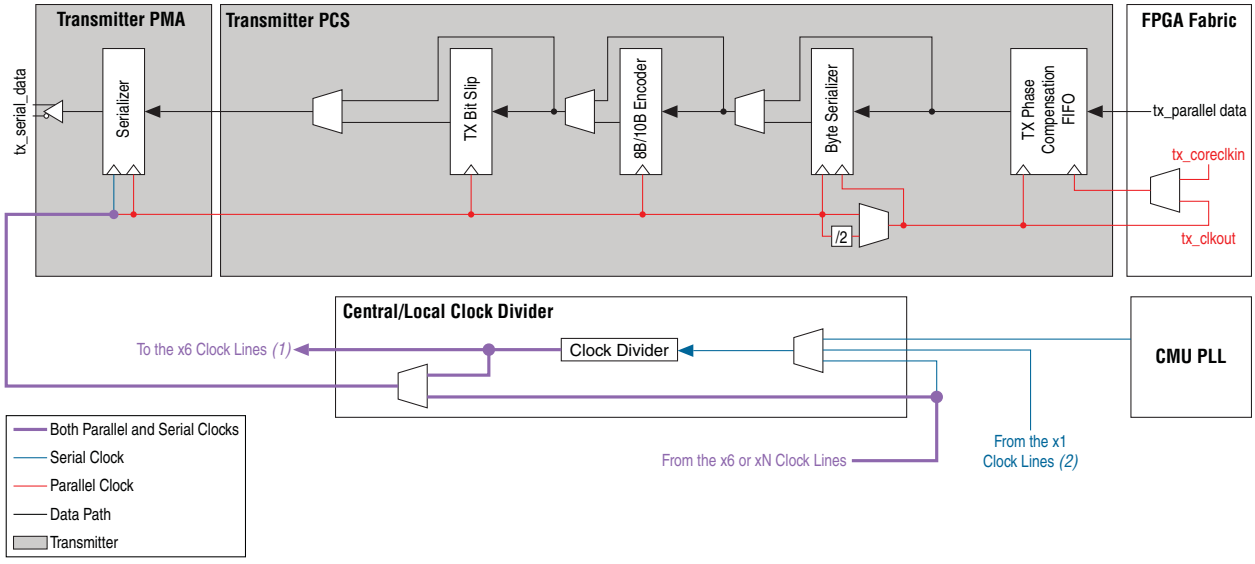
(1) Applies only to the fastest speed grade. For other speed grades, refer to the *Device Datasheet for Arria V Devices* chapter.

Transmitter Clocking

Transmitter clocking refers to the clocking architecture that is internal to the transmitter channel of a transceiver.

Figure 2–11 shows the transmitter PCS and physical medium attachment (PMA) clocking for 6-Gbps transceivers.

Figure 2–11. Transmitter PCS Clocking for 6-Gbps Transceiver



Notes to Figure 2–11:

(1) Only available in the central clock dividers of channel 1 and channel 4 in a transceiver bank.

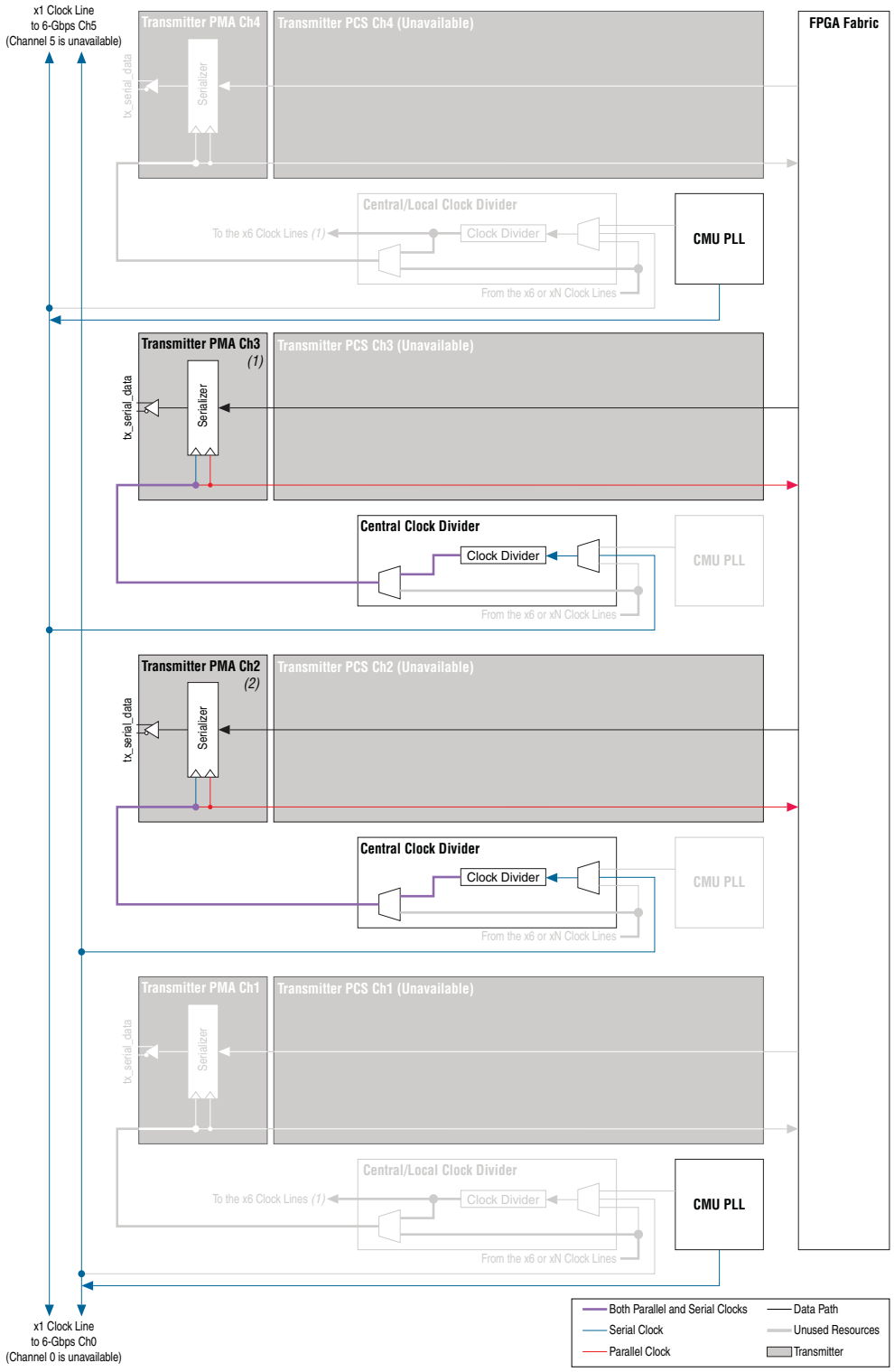
(2) You can drive x1 clock lines of the 6-Gbps transceiver with either a CMU PLL or fractional PLL.

As shown in Figure 2–11, the clock divider block provides the serial and parallel clock to the serializer of the transmitter PMA, and the parallel clock to the transmitter PCS. The parallel clock clocks all the blocks up to the read side of the transmitter (TX) phase compensation FIFO in all configurations that do not use the byte serializer block.

For configurations that use the byte serializer block, the clock is divided by a factor of two for the byte serializer and the read side of the TX phase compensation FIFO. The clock that clocks the read side of the TX phase compensation FIFO is also forwarded to the FPGA fabric to interface the FPGA fabric with the transceiver.

Figure 2–12 shows transmitter PMA clocking for 10-Gbps transceivers.

Figure 2–12. Transmitter PMA Direct Clocking for 10-Gbps Transceiver



Notes to Figure 2–12:

(1) The 10-Gbps transceiver channel 2 is the 6-Gbps transceiver channel 3.

(2) The 10-Gbps transceiver channel 1 is the 6-Gbps transceiver channel 2.

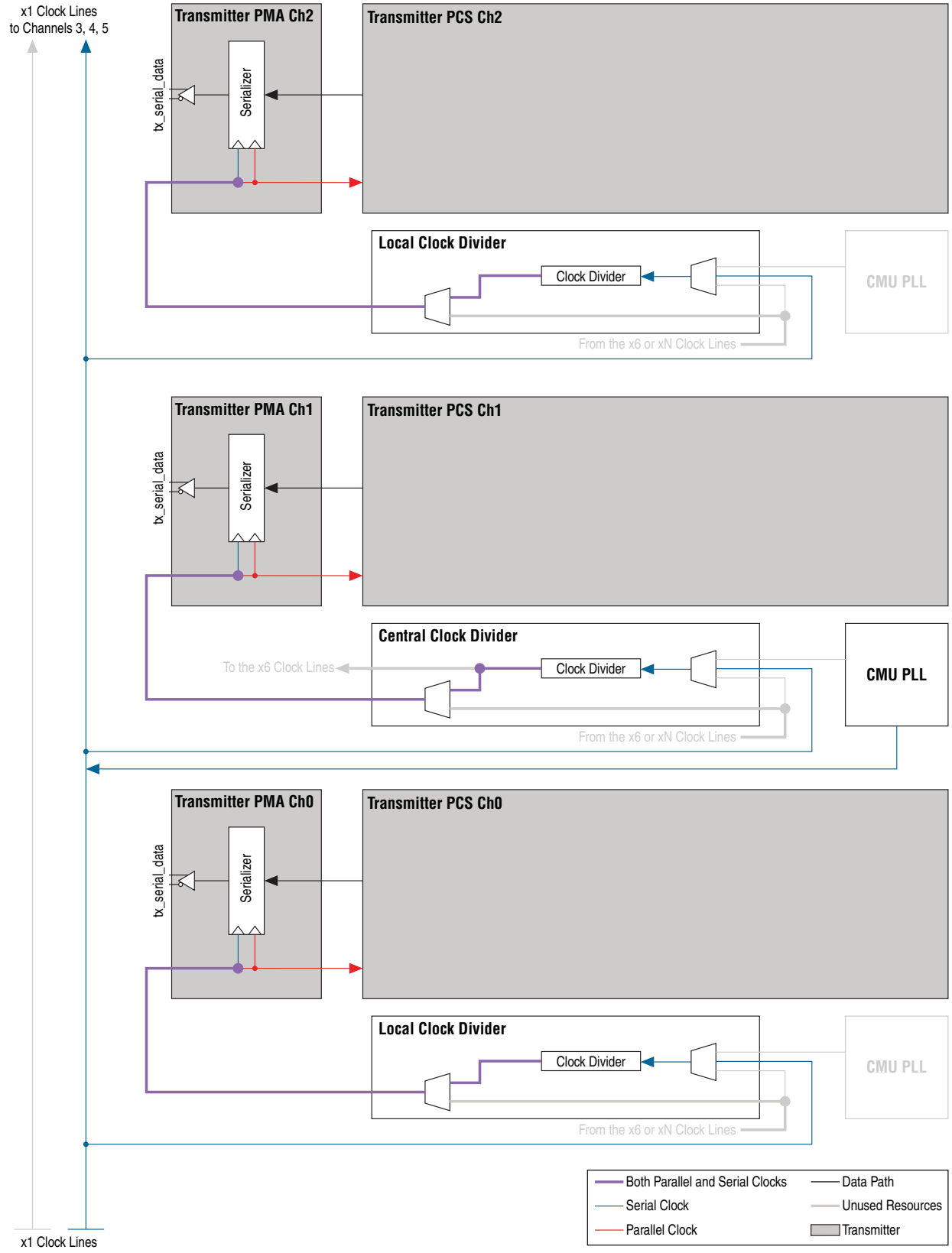
As shown in Figure 2–12, the clock divider block provides the serial and parallel clock to the serializer of the transmitter PMA. The parallel clock is driven directly to the FPGA fabric because the PCS is unavailable in 10-Gbps transceivers. All PCS functions, such as encoding and bit slipping, must be implemented in the core IP.

For more about clocking schemes used in different configurations, refer to the *Transceiver Protocol Configurations in Arria V Devices* and *Custom Transceiver Configuration Datapath in Arria V Devices* chapters.

Non-Bonded Channel Configurations

In non-bonded configurations for 6-Gbps transceivers, the clock divider of individual channels generates the parallel clock. Figure 2–13 shows three transmitter channels in a non-bonded configuration driven by the CMU PLL of channel 1 and driving the x1 clock line. The clock divider block of each channel generates its own parallel clock by dividing the serial clock from the x1 clock line.

Figure 2–13. Three 6-Gbps Transmitter Channels in Non-Bonded Configuration



In non-bonded configurations for 10-Gbps transceivers, the local clock divider of the individual channel generates the parallel clock for transmitter functionality. In Figure 2–12 on page 2–14, two 10-Gbps transmit channels function independently and are driven by their respective CMU PLL from the adjacent inactive 6-Gbps channels. The clock divider of each 10-Gbps channel generates its own parallel clock by dividing the serial clock coming from the CMU PLL of the adjacent channel.

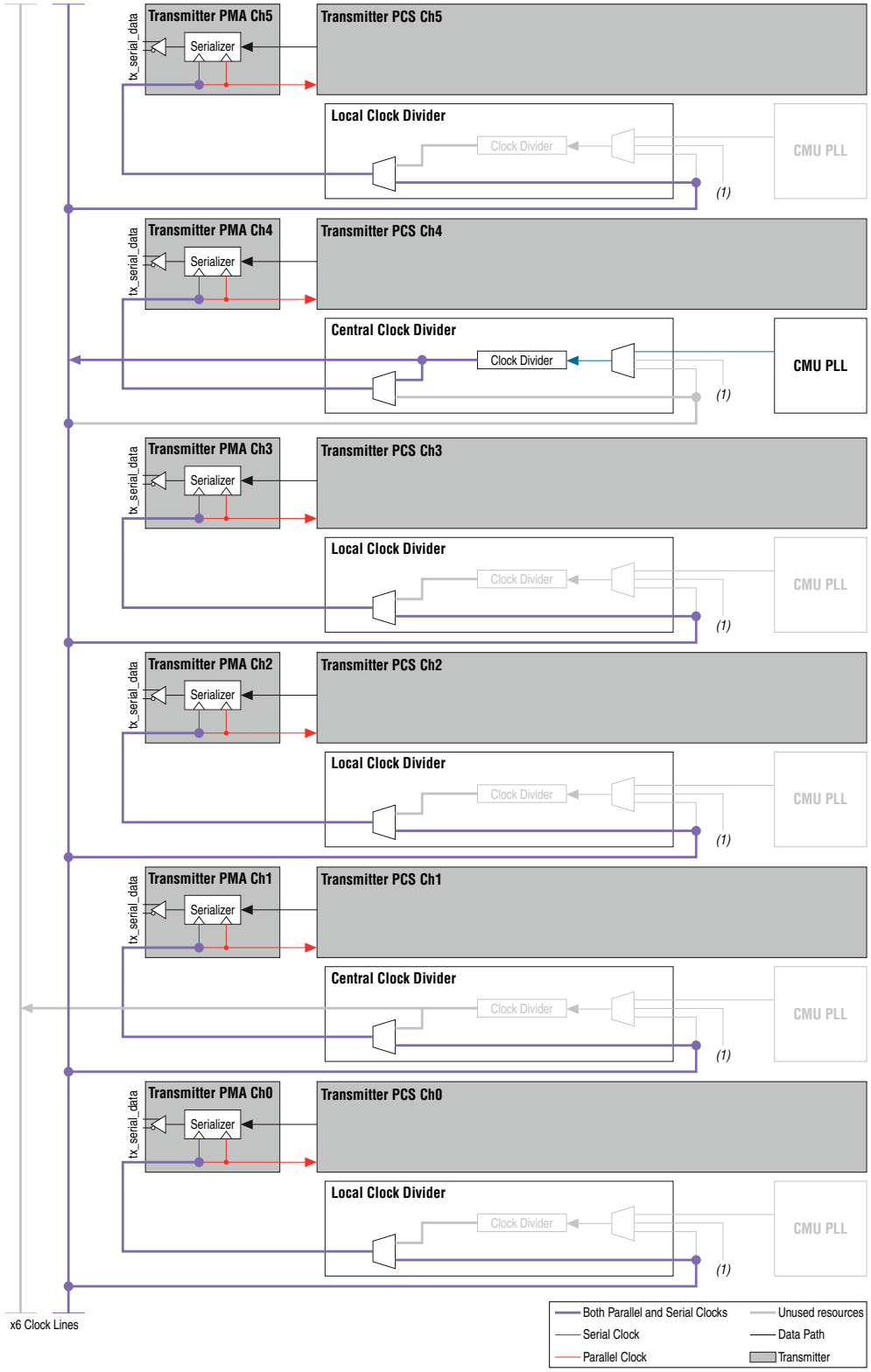
Bonded Channel Configurations

In bonded configurations, both the parallel clock and serial clock are sourced from the x6 or xN clock line. The central clock dividers use the serial clock from a transmit PLL from the same transceiver bank using the x1 clock line. The central clock divider generates the parallel clock and drives both the serial clock and parallel clock on the x6 clock line.

The bonded configuration is applicable to 6-Gbps transceivers only.

Figure 2–14 shows six transmit-only channels configured in a bonded configuration driven by the channel PLL of channel 4, which is configured as a CMU PLL. The central clock divider of channel 4 generates a parallel clock and drives both the serial clock and parallel clock on the x6 clock line. All bonded channels source both serial and parallel clocks from the x6 clock line.

Figure 2–14. Six Transmitter Channels Configured in Bonded Configuration



Note to Figure 2–14:

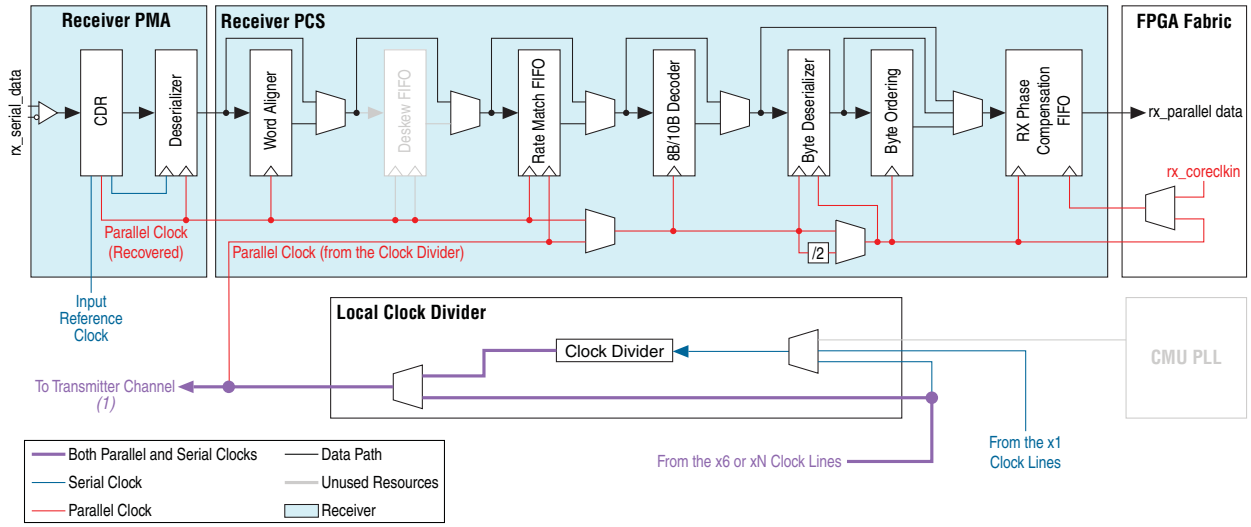
(1) Serial clock from the x1 clock lines. The x1 clock lines are inactive because all 6 transceiver channels in the bank are bonded.

For an example of using the xN clock lines, refer to the PCIe® x8 configuration in the *Transceiver Protocol Configurations in Arria V Devices* and *Custom Transceiver Configuration Datapath in Arria V Devices* chapters.

Receiver Clocking

Receiver clocking refers to the clocking architecture internal to the receiver channel of a transceiver. Figure 2–15 shows 6-Gbps receiver PCS and PMA clocking.

Figure 2–15. Receiver PCS Clocking for 6-Gbps Transceivers



Note to Figure 2–15:

(1) Only available in the central clock dividers of channel 1 and channel 4 in a transceiver bank.

The CDR in the PMA of each channel recovers the serial clock from the incoming data and generates the parallel clock (recovered) by dividing the serial clock (recovered). The deserializer uses both clocks. The receiver PCS can use the following clocks depending on the configuration of the receiver channel:

- Parallel clock (recovered) from the CDR in the PMA
- Parallel clock from the clock divider that is used by the channel's transmitter PCS

Table 2–2 lists the different clock sources available for each block in the receiver PCS.

Table 2–2. Clock Sources for All Receiver PCS Blocks (Part 1 of 2)

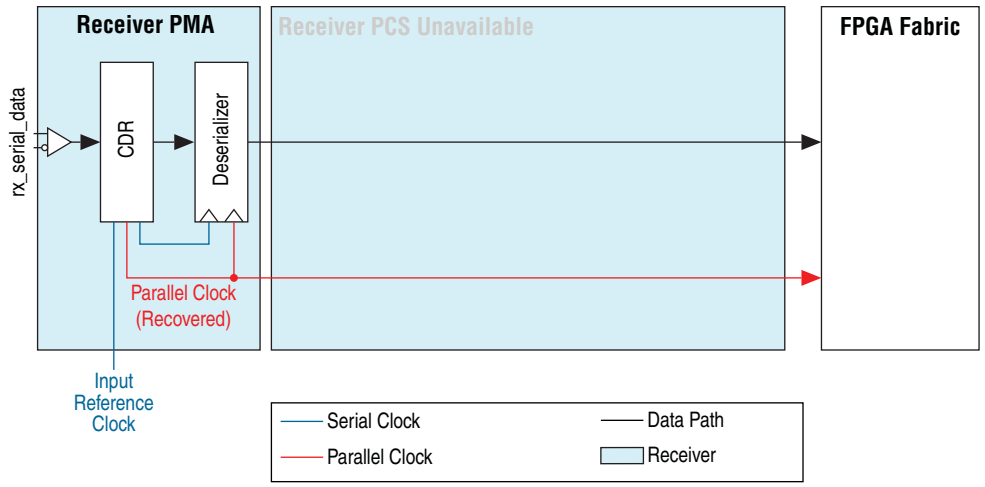
Block	Side	Clock Source
Word aligner	—	Parallel clock (recovered)
Rate match FIFO	Write	Parallel clock (recovered)
	Read	Parallel clock from the clock divider
8B/10B decoder	—	■ Rate match FIFO is not used—Parallel clock (recovered) ■ Rate match FIFO is used—Parallel clock from the clock divider

Table 2–2. Clock Sources for All Receiver PCS Blocks (Part 2 of 2)

Block	Side	Clock Source
Byte deserializer	Write	■ Rate match FIFO is not used—Parallel clock (recovered) ■ Rate match FIFO is used—Parallel clock from the clock divider
	Read	Divided down version of the write side clock depending on the deserialization factor of 1 or 2, also called the parallel clock (divided)
Byte ordering	—	Parallel clock (divided)
Receiver (RX) phase compensation FIFO	Write	Parallel clock (divided). This clock is also forwarded to the FPGA fabric.
	Read	Clock sourced from the FPGA fabric

Figure 2–16 shows clocking for the receiver PMA in a 10-Gbps transceiver.

Figure 2–16. Receiver PMA Direct Clocking for a 10-Gbps Transceiver



As shown in Figure 2–16, the parallel recovered clock from the CDR and deserializer is directly connected to the FPGA fabric because the PCS is unavailable in 10-Gbps transceivers. All PCS functions, such as word alignment, rate matching, decoding, and byte ordering, must be implemented in the core IP.

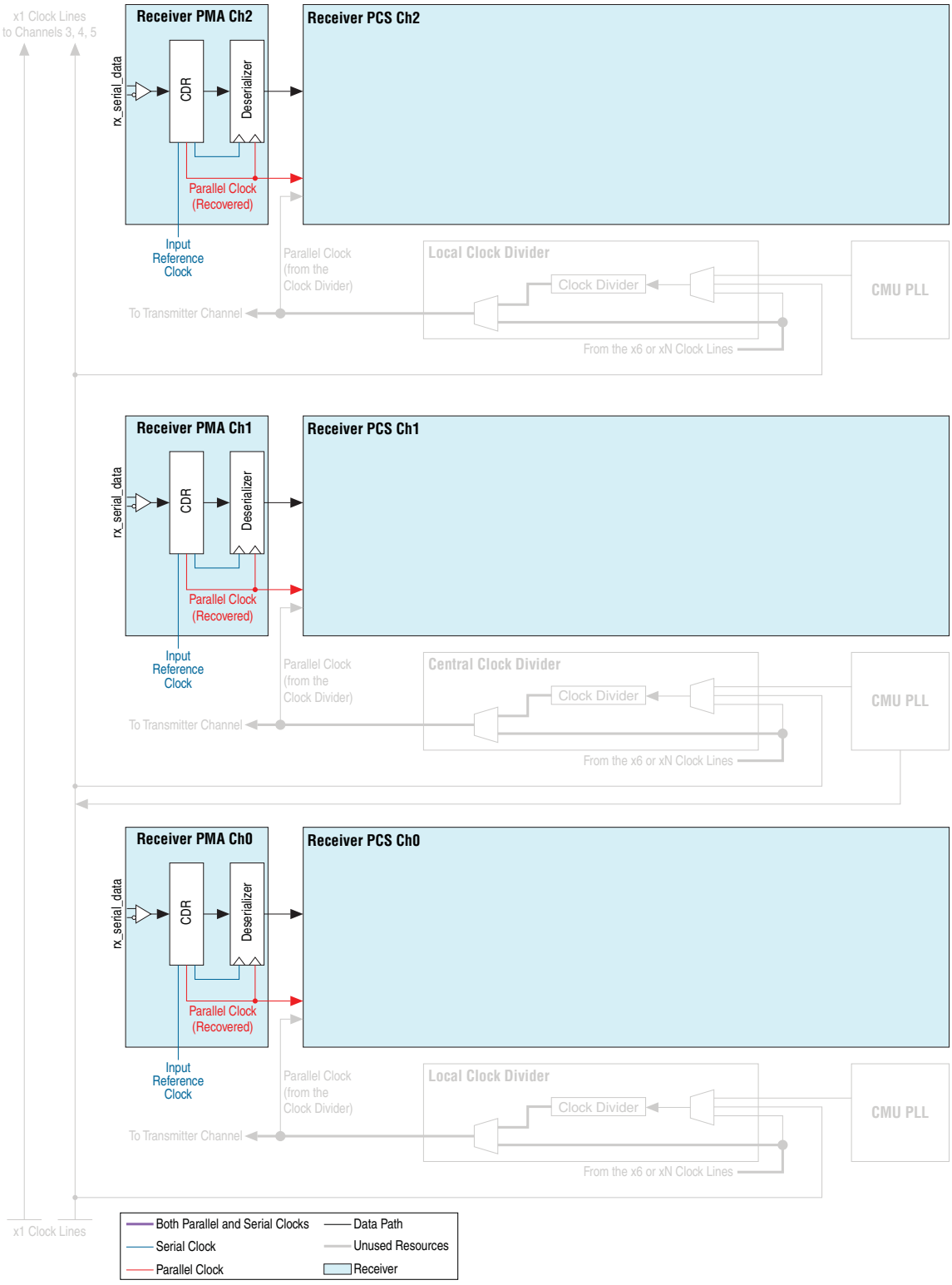
Non-Bonded Channel Configurations

In non-bonded configurations, the 6-Gbps receiver PCS requires the parallel clock (recovered) and depending on the configuration you use, may also require the parallel clock from the clock divider used by the transmitter.

For 10-Gbps transceivers, the FPGA fabric requires the parallel clock (recovered) and depending on the configuration you use, may also require the parallel clock from the clock divider used by the transmitter.

Figure 2–17 shows three 6-Gbps receiver channels in a non-bonded configuration when the rate match FIFO is not used.

Figure 2–17. Three 6-Gbps Transceiver Channels in a Non-Bonded Configuration



Bonded Channel Configurations

Bonded channel configurations are supported in 6-Gbps transceiver channels only. In bonded configurations, the receiver PCS requires the parallel clock (recovered) and, depending on your configuration, may require the parallel clock from the central clock divider in channel 1 or 4.

Figure 2–18 shows five bonded channels in a transceiver bank. Because the channel PLL in channel 4 is configured as a CMU PLL, you cannot use the receiver CDR. Therefore, you can only use channel 4 as a transmitter.

In Figure 2–19, all six channels in the transceiver bank are in a bonded configuration, as opposed to a maximum of five in Figure 2–18. This configuration is possible because the fractional PLL is used as a transmit PLL instead of a channel PLL in the transceiver bank. Using the fractional PLL enables you to configure the channel PLLs of both channels 1 and 4 as CDRs to perform receiver operations.

For more information about the clocking scheme used in different configurations, refer to the *Transceiver Protocol Configurations in Arria V Devices* and *Custom Transceiver Configuration Datapath in Arria V Devices* chapters.

Figure 2–18. Five Channels Configured in Bonded Configuration

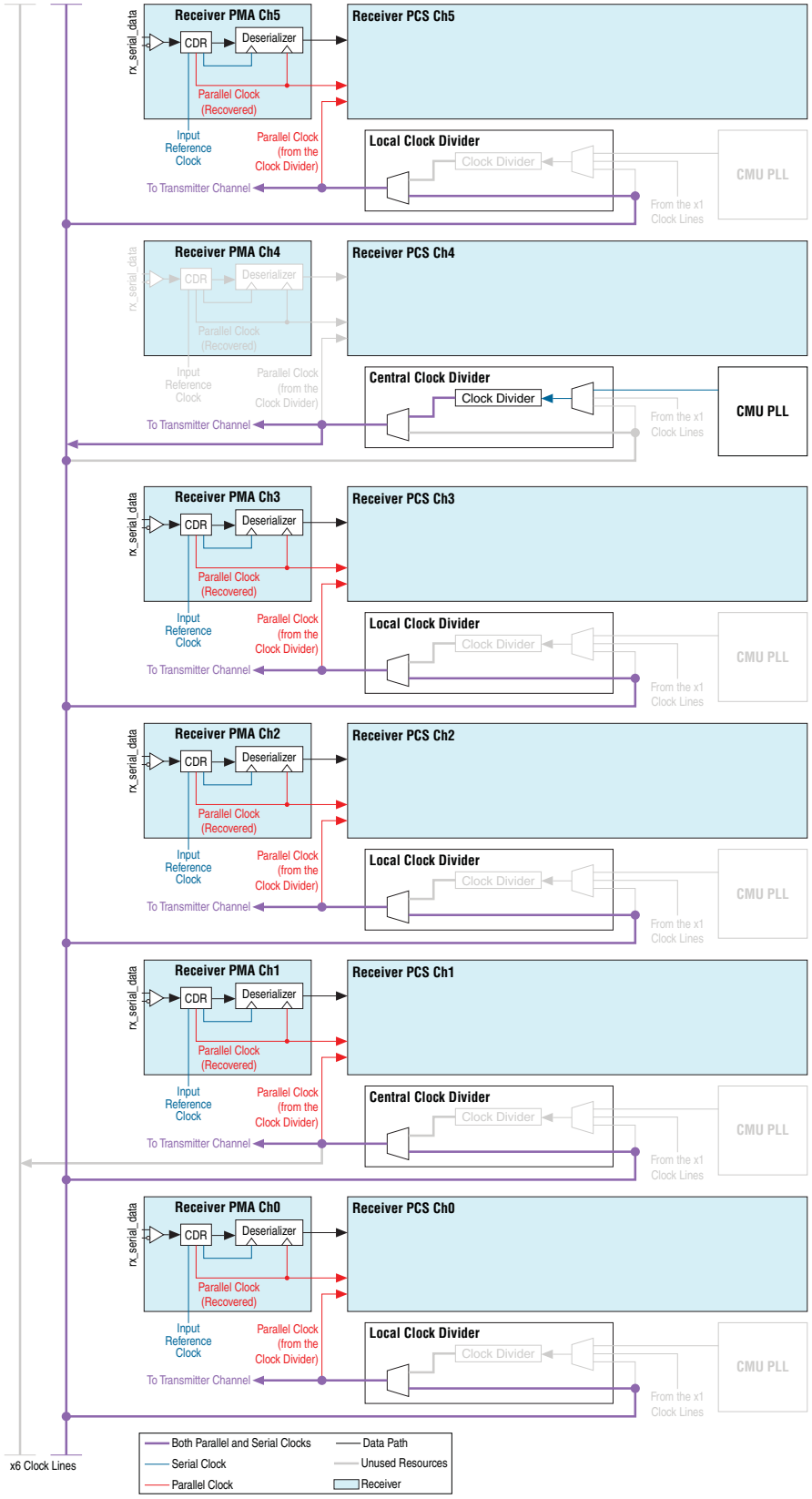
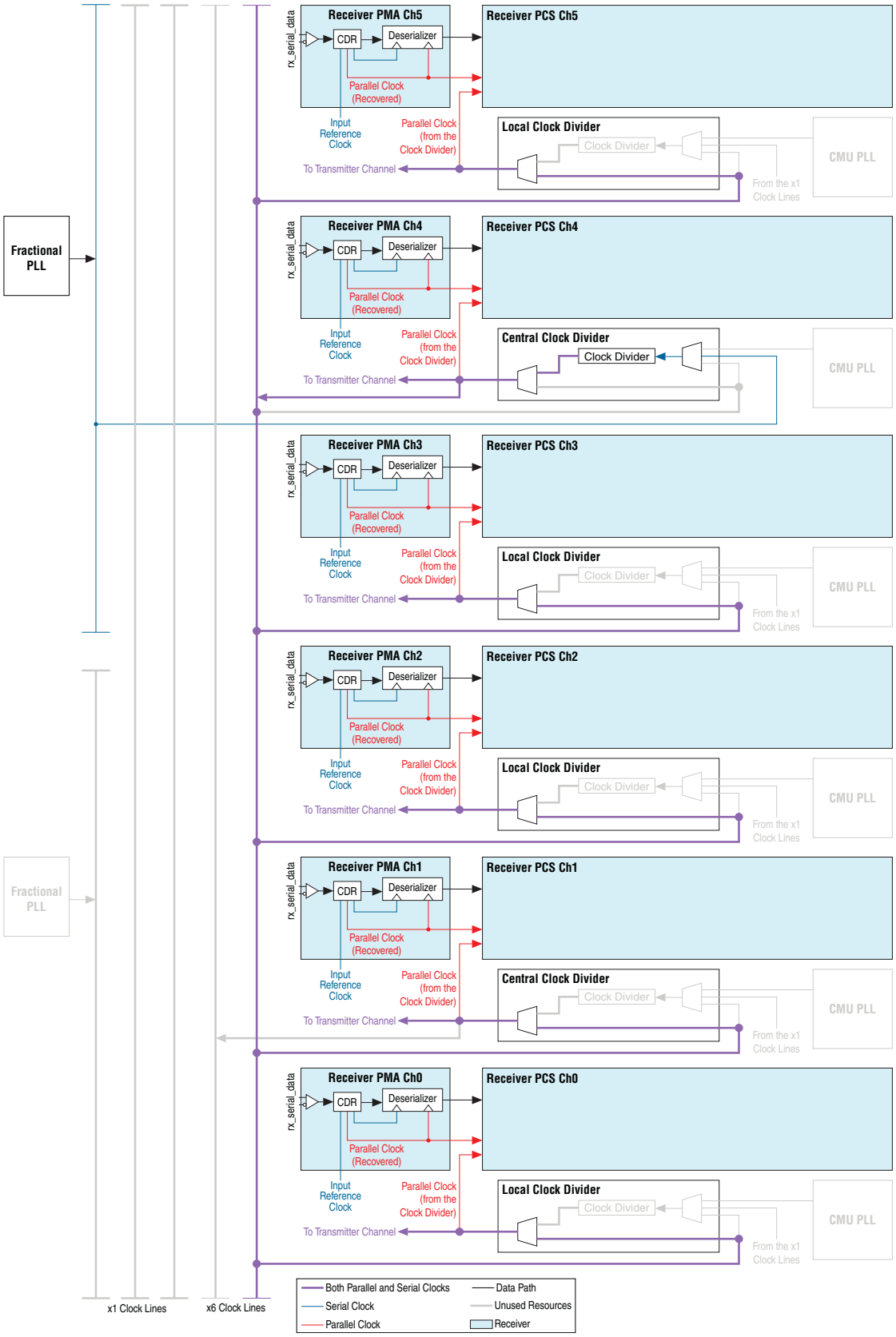


Figure 2–19. Six Channels Configured in Bonded Configuration Using Fractional PLL



FPGA Fabric–Transceiver Interface Clocking

The FPGA fabric–transceiver interface clocks consist of clock signals from the FPGA fabric to the transceiver blocks and clock signals from the transceiver blocks to the FPGA fabric. These clock resources use the clock networks in the FPGA core, including the global (GCLK), regional (RCLK), and periphery (PCLK) clock networks.

There are three categories of FPGA fabric–transceiver interface clocks:

- Input reference clocks
- Transceiver datapath interface clocks
- Other transceiver clocks

For information about the input reference clock sources, refer to "Input Reference Clocking" on page 2–1.

For more information about the clocking options available when the transceiver interfaces with the FPGA fabric, refer to the "Transceiver Clocking" section in the *Transceiver Basics for Arria V Devices* chapter.

Document Revision History

Table 2–3 lists the revision history for this chapter.

Table 2–3. Document Revision History

Date	Version	Changes
November 2011	1.1	Updated chapter for clarity and Quartus II 11.1 update.
August 2011	1.0	Initial release

3. Transceiver Reset Control and Power-Down in Arria V Devices

AV53003-1.1

This chapter provides information about the transceiver reset control and transceiver power-down support in Arria® V devices.

You can implement the transceiver reset using the embedded reset controller in the PHY IP MegaCore® function. Use the transceiver reset controller to initialize the physical coding sublayer (PCS) and physical medium attachment (PMA) blocks. Use the reset sequence recommended in this chapter to ensure a reliable link initialization after the initial power-up and for reestablishing the link.

This chapter contains the following sections:

Transceiver Reset Controller

The embedded reset controller in the PHY IP MegaCore function provides an option to implement an automatic reset sequence, simplifying your transceiver-based design. The embedded reset controller requires only one control input to initiate the automatic reset sequence. There is only one embedded reset controller for all the channels in a PHY IP instance.

Figure 3–1 shows the embedded reset controller in the PHY IP MegaCore function.

Figure 3–1. Embedded Reset Controller in Arria V Devices

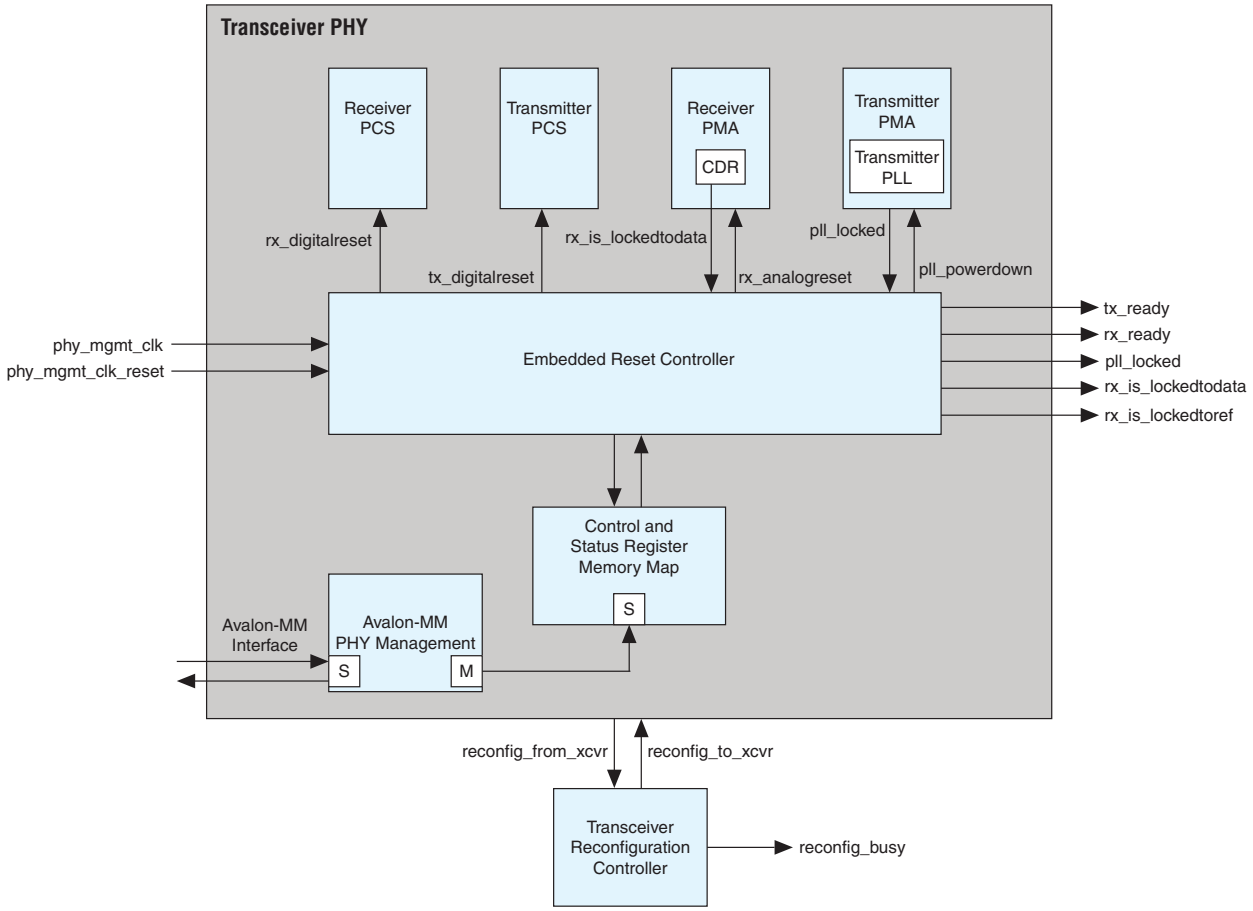


Table 3–1 lists the control inputs, status outputs, and internal signals.

Table 3–1. Transceiver Reset Control and Status Signals (Part 1 of 2)

Signal Name	Signal Type	Description
phy_mgmt_clk	Control Input	Clock for the embedded reset controller.
phy_mgmt_clk_reset	Control Input	A high-to-low transition of this asynchronous reset signal initiates the automatic reset sequence control.
reconfig_busy	Status Output	An output from the ALT_XCVR_RECONFIG block indicates the status of the dynamic reconfiguration controller. In the first reconfig_clk clock cycle after power-up, reconfig_busy remains low. This signal is asserted from the second reconfig_clk clock cycle to indicate that the offset cancellation process is active on clock data recovery (CDR). When the offset cancellation process is completed, the reconfig_busy signal is deasserted. This signal is also routed to the embedded reset controller by the Quartus® II software by embedding the signal with the reconfig_to_xcvr bus between the PHY IP and the ALT_XCVR_RECONFIG block.
tx_ready	Status Output	A low-to-high transition of this signal indicates that the transmitter (TX) channel is out of reset and is ready for data transmission.

Table 3–1. Transceiver Reset Control and Status Signals (Part 2 of 2)

Signal Name	Signal Type	Description
rx_ready	Status Output	A low-to-high transition of this signal indicates that the receiver (RX) channel is out of reset and is ready for data reception.
pll_powerdown	Internal Control	When this signal is asserted, the TX phase-locked loop (PLL) and all blocks in the TX PMA are reset. The pll_powerdown signal can only be controlled by using the embedded reset controller.
tx_digitalreset	Internal Control	When this signal is asserted, all blocks in the TX PCS are reset.
rx_analogreset	Internal Control	When this signal is asserted, the RX CDR and all blocks in the RX PMA are reset. After reconfig_busy is low, this signal can be deasserted.
rx_digitalreset	Internal Control	When this signal is asserted, all blocks in the RX PCS are reset.
pll_locked	Internal Status	This signal is asserted to indicate that the TX PLL achieves lock to the input reference clock. When this signal is asserted high, the embedded reset controller deasserts the tx_digitalreset signal.
	Status Output	
rx_is_lockedtodata	Internal Status	When this signal is asserted, the embedded reset controller deasserts the rx_digitalreset signal. When this signal is deasserted, the embedded reset controller asserts the rx_digitalreset signal.
	Status Output	This signal is an optional output status port. When asserted, this signal indicates that the CDR is locked to the RX data and the CDR has changed from lock-to-reference (LTR) to lock-to-data (LTD) mode.
rx_is_lockedtoref	Status Output	This is an optional output status port. When asserted, this signal indicates that the CDR is locked to the reference clock.

You cannot control the pll_powerdown signal using memory map registers.

Table 3–2 lists the memory map registers for CDR lock mode and channel reset.

Table 3–2. Transceiver Manual Reset Control Using Memory Map Registers

Register Name	Description
pma_rx_set_locktodata	This register is for CDR manual lock mode. When you set the register to high, the RX CDR PLL locks to the incoming data.
pma_rx_set_locktoref	This register is for CDR manual lock mode. When you set the register to high, the RX CDR PLL locks to the reference clock.
reset_tx_digital	When you set this register to high, the tx_digitalreset signal is asserted in every channel that is enabled for reset control. To deassert the tx_digitalreset signal, set the reset_tx_digital register to low.
reset_rx_analog	When you set this register to high, the rx_analogreset signal is asserted in every channel that is enabled for reset control. To deassert the rx_analogreset signal, set the reset_rx_analog register to low.
reset_rx_digital	When you set this register to high, the rx_digitalreset signal is asserted in every channel that is enabled for reset control. To deassert the rx_digitalreset signal, set the reset_rx_digital register to low.

If you set the pma_rx_set_locktodata or pma_rx_set_locktoref register to high, the CDR is placed in manual lock mode.

The reset_ch_bitmask registers provide an option to enable or disable certain channels in a PHY IP instance for reset control. By default, all channels in a PHY IP instance are enabled for reset control.

Transceiver Reset Sequence

After device power-up and `phy_mgmt_clk_reset` is set to low, the embedded reset controller automatically performs the reset sequence for the transmitter and receiver.

If `phy_mgmt_clk_reset` is set to high after device power-up, the embedded reset controller holds the transmitter and receiver in reset and the calibration process is on hold. After `phy_mgmt_clk_reset` is deasserted, the embedded reset controller performs an automatic reset sequence for the transmitter and receiver and the calibration process starts.

During device operation, you can perform the reset sequence with the following options:

- Automatic reset sequence using the embedded reset controller for both the transmitter and receiver.
- Manual reset sequence using the memory map registers for the receiver. This option allows you to perform the reset sequence for the receiver only while in duplex mode.

The transceiver reset control can use the CDR in automatic or manual lock mode. The mode is determined by the memory map registers `pma_rx_set_locktoref` and `pma_rx_set_locktodata`. By default, CDR is in automatic lock mode.

After the transceiver dynamic reconfiguration completes, as indicated by the deassertion of the `reconfig_busy` signal, you must repeat the entire reset sequence.

Each reset sequence described in this section is the full reset sequence for a duplex channel unless otherwise stated. For a transmitter- or receiver-only channel, only the signals related to the transmitter or receiver are used.

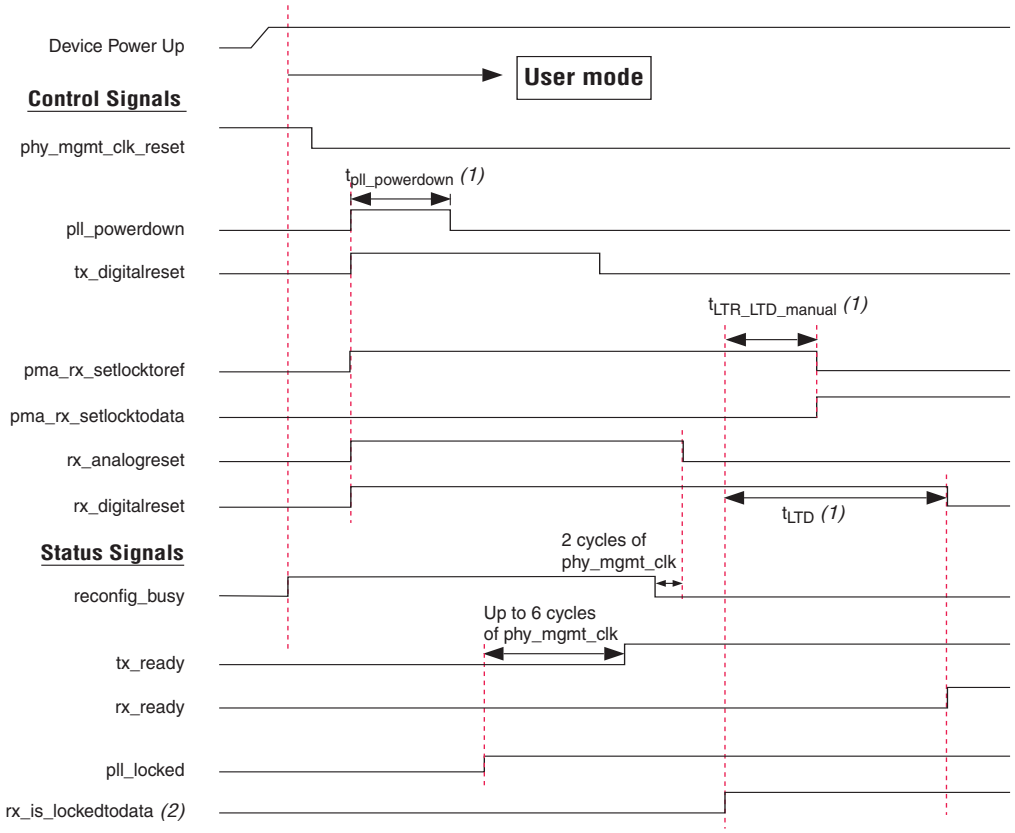
Automatic Reset Sequence Using the Embedded Reset Controller

The recommended transceiver reset sequence is different for non-PCI Express® (PCIe®) and PCIe configurations.

Reset Sequence in Non-PCIe Configurations

Figure 3–2 shows the transceiver automatic reset sequence timing diagram for non-PCIe configurations using the embedded reset controller with CDR manual lock mode.

Figure 3–2. Timing Diagram for Transceiver Automatic Reset Sequence in Non-PCIe Configurations with CDR Manual Lock Mode



Notes to Figure 3–2:

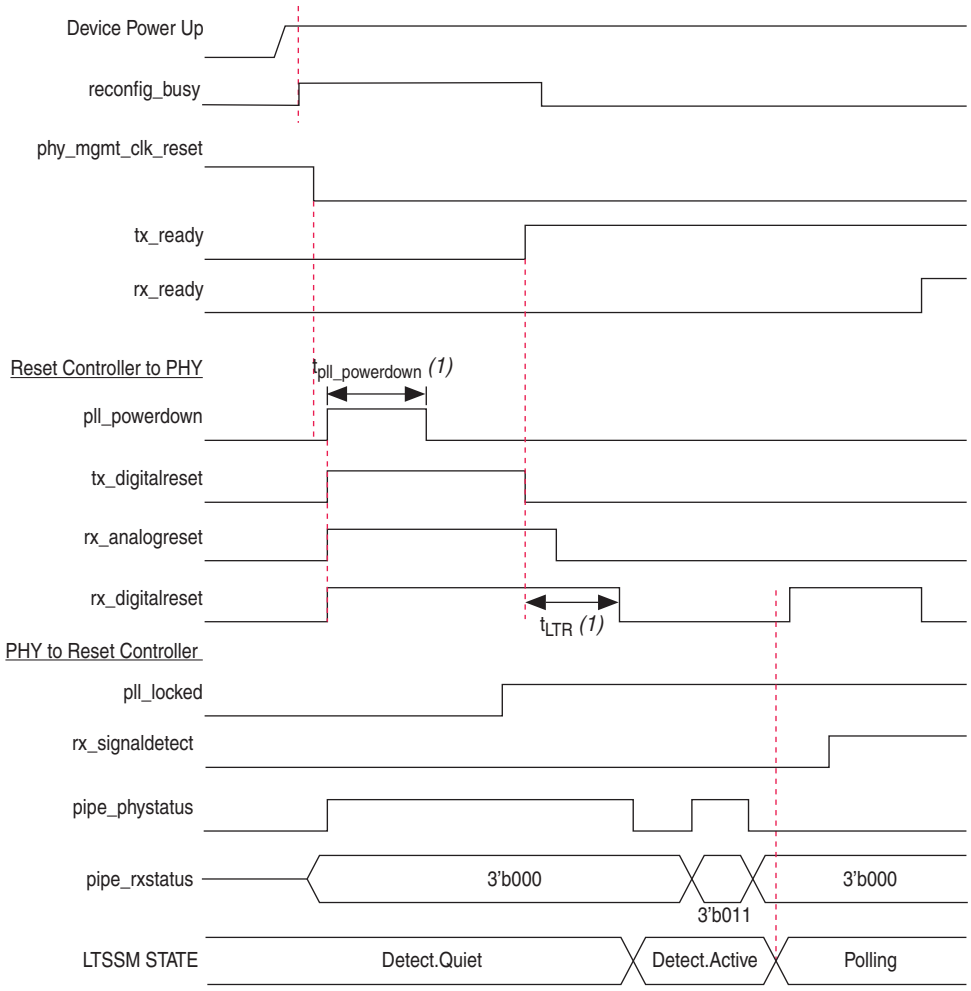
(1) $t_{pll\_powerdown}$, $t_{LTR\_LTD\_manual}$, and t_{LTD} are pending characterization.
(2) In bonded mode configurations, the `rx_is_lockedtodata` signal shown in this figure is the logical AND of the `rx_is_lockedtodata` signals from all channels.

In Figure 3–2, $t_{LTR\_LTD\_manual}$ is the timing requirement when using CDR manual lock mode. t_{LTD} is the timing requirement when using CDR automatic lock mode.

Reset Sequence in PCIe Configuration

Figure 3–3 shows the transceiver reset sequence timing diagram for PCIe configurations using the embedded reset controller in the PHY IP MegaCore function.

Figure 3–3. Timing Diagram for Transceiver Reset Sequence in PCIe Configurations



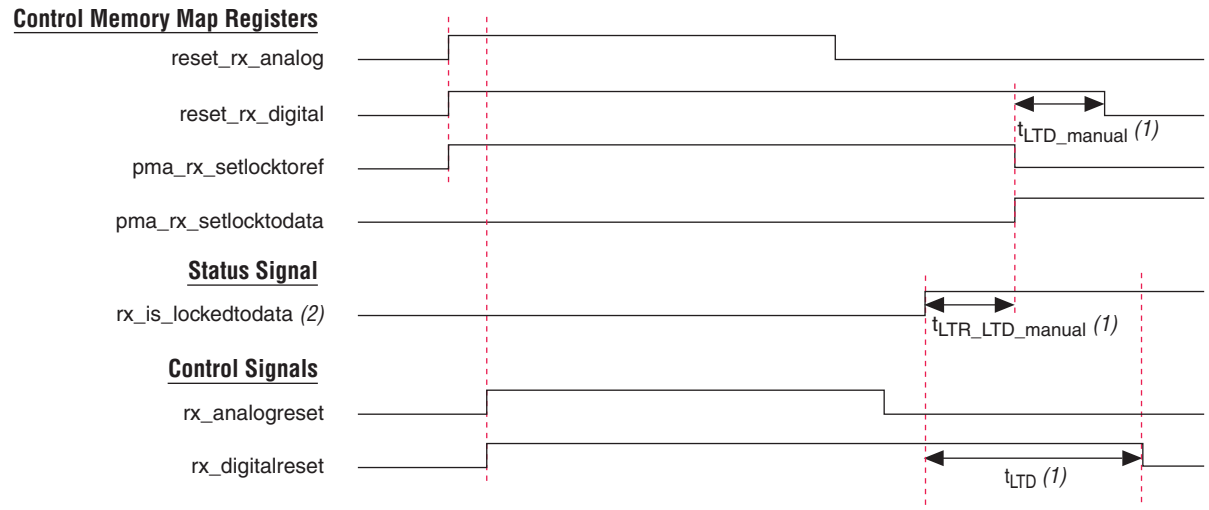
Note to Figure 3–3:

(1) $t_{pll\_powerdown}$ and t_{LTR} are pending characterization.

Manual Reset Sequence

Figure 3–4 shows the timing diagram for the receiver manual reset sequence during the device operation for non-PCIe configurations with CDR manual lock mode. This option allows you to perform the reset sequence for only the receiver in a duplex channel configuration.

Figure 3–4. Timing Diagram for Receiver Manual Reset Sequence During Device Operation for Non-PCIe Configurations with CDR Manual Lock Mode



Notes to Figure 3–4:

(1) $t_{LTR\_LTD\_manual}$, t_{LTD_manual}, and t_{LTD} are pending characterization.
(2) In bonded mode configurations, the `rx_is_lockedtodata` signal shown in this figure is the logical AND of the `rx_is_lockedtodata` signals from all channels.

In Figure 3–4, $t_{LTR\_LTD\_manual}$ and t_{LTD_manual} are the timing requirements when using CDR manual lock mode. t_{LTD} is the timing requirement when using CDR automatic lock mode.

Transceiver Power-Down

To maximize power savings, enable PMA hard power-down across all channels on a side of the device where you do not use the transceivers.

The power granularity control of the transceiver PMA is per side. To enable PMA hard power-down on the left or right side of the device, ground the transceiver power supply of the respective side.

To power up the PMA on the device side, supply power to the side and perform the initialization according to the recommended reset sequence, as shown in Figure 3–2 on page 3–5 and Figure 3–3 on page 3–6.

When you configure the Arria V device, the Quartus II software automatically selects the power-down channel feature. All unused transceiver channels and blocks are powered down to reduce overall power consumption.

For information about the transceiver power supply operating conditions of the left and right side of Arria V devices, refer to the *Device Datasheet for Arria V Devices* chapter.

Document Revision History

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

Table 3–3. Document Revision History

Date	Version	Changes
November 2011	1.1	■ Updated all figures and tables. ■ Reorganized and updated the "Transceiver Reset Sequence" section.
August 2011	1.0	Initial release.

4. Transceiver Protocol Configurations in Arria V Devices

AV53004-1.1

This chapter provides the transceiver channel datapath, clocking guidelines, channel placement guidelines, and a brief description of protocol features supported in each transceiver configuration for Arria® V devices.

This chapter contains the following sections:

Transceiver PCS Features

Arria V devices have dedicated transceiver physical coding sublayer (PCS) and physical medium attachment (PMA) circuitry to support the communication protocols listed in Table 4–1.

Table 4–1. Transceiver PCS Features for Arria V Devices *(1)*

PCS Support	Data Rates (Gbps)	Transmitter Datapath	Receiver Datapath
PCI Express® (PCIe®) Gen1 (x1, x2, x4, and x8)	2.5	The same as custom single- and double-width modes, plus the PHY interface for PCI Express (PIPE) 2.0 interface to the core logic	The same as custom single- and double-width modes, plus the rate match FIFO and PIPE 2.0 interface to the core logic
Gbps Ethernet (GbE)	1.25	The same as custom single- and double-width modes	The same as custom single- and double-width modes, plus the rate match FIFO
Common Public Radio Interface (CPRI)	0.6144–6.144, 9.8304 *(2)*	The same as custom single- and double-width modes, plus the transmitter (TX) deterministic latency	The same as custom single- and double-width modes, plus the receiver (RX) deterministic latency
OBSAI	0.768–6.144	The same as custom single- and double-width modes, plus the TX deterministic latency	The same as custom single- and double-width modes, plus the RX deterministic latency

Notes to Table 4–1:

(1) Not yet supported by the Quartus II® software version 11.1.

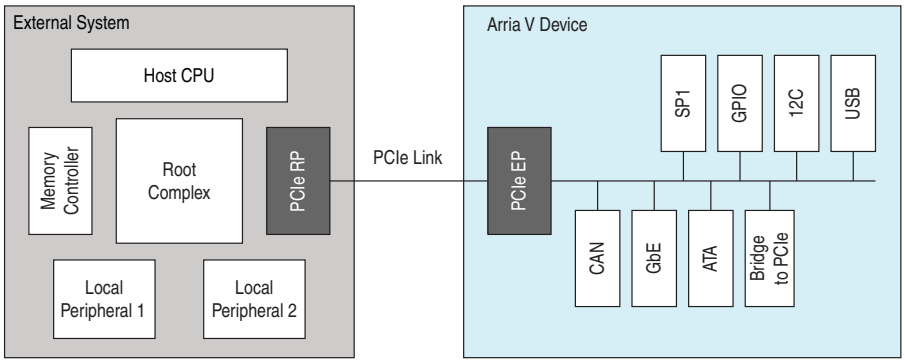
(2) For the 9.8304 Gbps CPRI implementation, there is no hard PCS. To interface with the PMA in a PMA Direct configuration, soft PCS is required.

For a complete list of transceiver protocols supported by Arria V devices, refer to the *Upcoming Arria V Device Features* document. For a complete list of serial protocols supported by Arria V devices, refer to the *Overview for Arria V Device Family* chapter in volume 1 of the *Arria V Device Handbook*.

PCI Express

The Arria V devices have PCIe hard IP—consisting of the media access control (MAC) lane, data link, and transaction layers—that is designed for performance, ease-of-use, and increased functionality. The PCIe hard IP supports the PCIe Gen1 end point and root port up to x8 lane configurations. The PCIe endpoint support includes multifunction support for up to eight functions, as shown in Figure 4–1.

Figure 4–1. PCIe Multifunction for Arria V Devices



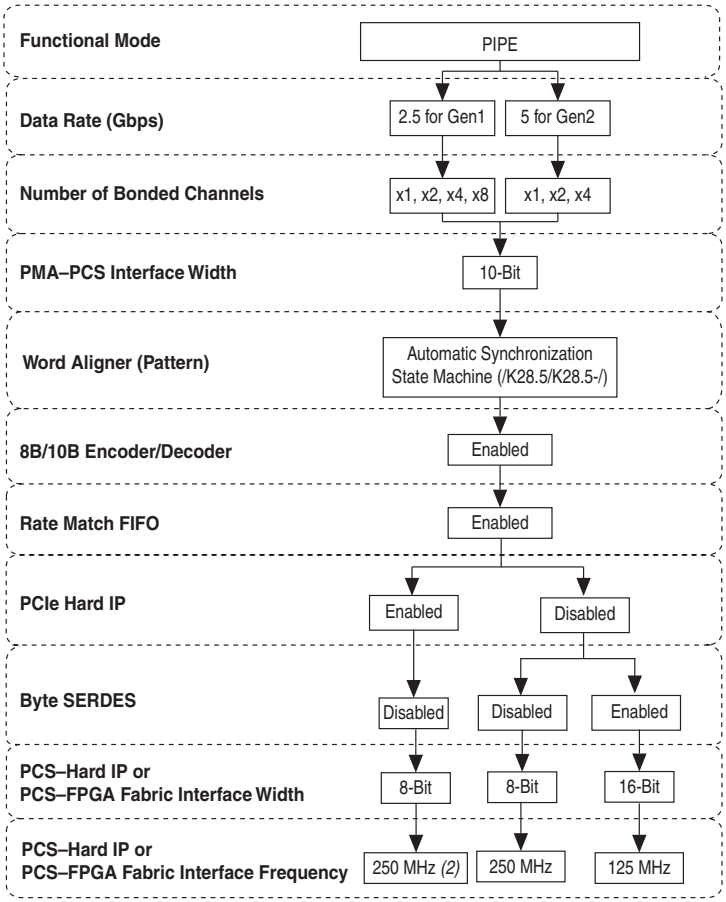
The Arria V PCIe hard IP operates independently from the core logic, which allows the PCIe link to wake up and complete link training in less than 100 ms while the Arria V device completes loading the programming file for the rest of the device.

In addition, the Arria V device PCIe hard IP has improved end-to-end datapath protection using error correction code (ECC).

Transceiver Datapath

Figure 4–2 shows the transceiver configurations allowed in a PIPE configuration.

Figure 4–2. Arria V Transceivers in a PIPE Configuration *(1)*



Notes to Figure 4–2:

(1) The PCIe Gen2 (up to x4) is supported only through the PCS–hard IP interface.

(2) Applies to the PCS–hard IP interface. The PCS–FPGA fabric interface frequency is limited to 159.3 MHz.

Table 4–2 lists the transceiver datapath clock frequencies in a PIPE configuration.

Table 4–2. Arria V Transceiver Datapath Clock Frequencies in a PIPE Configurations

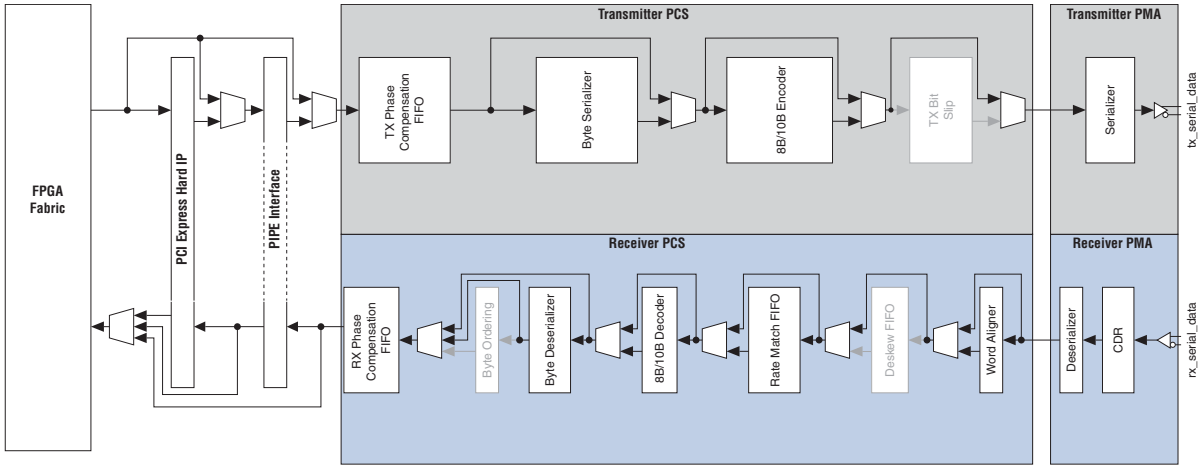
Functional Configuration	Data Rate	Serial Clock Frequency	Parallel Clock Frequency	FPGA Fabric–Transceiver Interface Clock Frequency	
				Without Byte SERDES (8 Bit Wide)	With Byte SERDES (16 Bit Wide)
PIPE (Gen1) x1, x2, x4, and x8	2.5 Gbps	1.25 GHz	250 MHz	250 MHz	125 MHz
PIPE (Gen2) x1, x2, and x4	5.0 Gbps	1.25 GHz	250 MHz	250 MHz	125 MHz

The transceiver datapath clocking varies between non-bonded (x1) and bonded (x2, x4, and x8) configurations. For more information about transceiver datapath clocking in different PIPE configurations, refer to "Transceiver Clocking" on page 4–10.

Transceiver Channel Datapath

Figure 4–3 shows the Arria V transmitter and receiver channel datapath in a PIPE configuration.

Figure 4–3. Arria V Transmitter Channel Datapath in a PIPE Configuration



For more information about the blocks in the transmitter datapath, refer to the *Transceiver Architecture in Arria V Devices* chapter.

Supported Features

The PIPE configuration for the 2.5 Gbps (Gen1) data rate supports these features:

- PCIe-compliant synchronization state machine
- ±300 parts per million (ppm)—total 600 ppm—clock rate compensation
- 8-bit FPGA fabric–transceiver interface
- 16-bit FPGA fabric–transceiver interface
- Transmitter buffer electrical idle
- Receiver detection
- 8B/10B encoder disparity control when transmitting compliance pattern
- Power state management
- Receiver status encoding

PIPE Interface

In a PIPE configuration, each channel has a PIPE interface block that transfers data, control, and status signals between the PHY-MAC layer and the transceiver channel PCS and PMA blocks. The PIPE interface block complies with version 2.0 of the PIPE specification. If you use the PIPE hard IP block, the PHY-MAC layer is implemented in the hard IP block. Otherwise, you can implement the PHY-MAC layer using soft IP in the FPGA fabric.

The PIPE interface block is only used in a PIPE configuration and cannot be bypassed.

In addition to transferring data, control, and status signals between the PHY-MAC layer and the transceiver, the PIPE interface block implements the following functions that are required in a PCIe-compliant physical layer device:

- Forces the transmitter buffer to an electrical idle state
- Initiates the receiver detect sequence
- Controls the 8B/10B encoder disparity when transmitting a compliance pattern
- Manages the PCIe power states
- Indicates the completion of various PHY functions, such as receiver detection and power state transitions on the `pipe_phystatus` signal
- Encodes the receiver status and error conditions on the `pipe_rxstatus[2:0]` signal, as specified in the PCIe specification

Transmitter Electrical Idle Generation

The PIPE interface block in Arria V devices places the channel transmitter buffer in an electrical idle state when the electrical idle input signal is asserted. During electrical idle, the transmitter buffer differential and common configuration output voltage levels are compliant to the PCIe Base Specification 2.0 for the PCIe Gen1 data rate.

The PCIe specification requires that the transmitter buffer be placed in electrical idle in certain power states. For more information about input signal levels required in different power states, refer to the "Power State Management" section.

Power State Management

The PCIe specification defines four power states—P0, P0s, P1, and P2—that the physical layer device must support to minimize power consumption:

- P0 is the normal operating state during which packet data is transferred on the PCIe link.
- P0s, P1, and P2 are low-power states into which the physical layer must transition as directed by the PHY-MAC layer to minimize power consumption.

The PIPE interface in Arria V transceivers provides an input port for each transceiver channel configured in a PIPE configuration.

When transitioning from the P0 power state to lower power states (P0s, P1, and P2), the PCIe specification requires that the physical layer device implements power saving measures. The Arria V transceivers do not implement these power saving measures except to place the transmitter buffer in electrical idle in the lower power states.

8B/10B Encoder Usage for Compliance Pattern Transmission Support

The PCIe transmitter transmits a compliance pattern when the LTSSM state machine enters a polling compliance substate. The polling compliance substate assesses if the transmitter is electrically compliant with the PCIe voltage and timing specifications.

Receiver Electrical Idle Inference

The PCIe protocol allows inferring the electrical idle condition at the receiver instead of detecting the electrical idle condition using analog circuitry.

In all PIPE configurations, each receiver channel PCS has an optional Electrical Idle Inference module designed to implement the electrical idle inference conditions specified in the PCIe Base Specification 2.0.

Receiver Status

The PCIe specification requires that the PHY encode the receiver status on a 3-bit status signal (`pipe_rxstatus[2:0]`). This status signal is used by the PHY-MAC layer for its operation. The PIPE interface block receives the status signals from the transceiver channel PCS and PMA blocks, and encodes the status on the `pipe_rxstatus[2:0]` signal to the FPGA fabric. The encoding of the status signals on the `pipe_rxstatus[2:0]` signal is compliant with the PCIe specification.

Receiver Detection

The PIPE interface block in Arria V transceivers provides an input signal (`pipe_txdetectrx_loopback`) for the receiver detect operation that is required by the PCIe protocol during the detect substate of the LTSSM.

When the `pipe_txdetectrx_loopback` signal is asserted in the P1 power state, the PCIe interface block sends a command signal to the transmitter buffer in that channel to initiate a receiver detect sequence. In the P1 power state, the transmitter buffer must always be in the electrical idle state.

After receiving this command signal, the receiver detect circuitry creates a step voltage at the output of the transmitter buffer. If an active receiver that complies with the PCIe input impedance requirements is present at the far end, the time constant of the step voltage on the trace is higher than if the receiver is not present. The receiver detect circuitry monitors the time constant of the step signal that is seen on the trace to determine if a receiver was detected. The receiver detect circuitry monitor requires a 125-MHz clock for operation that you must drive on the `fixedclk` port.

For the receiver detect circuitry to function reliably, the AC-coupling capacitor on the serial link and the receiver termination values used in your system must be compliant with the PCIe Base Specification 2.0.

The PCI Express PHY (PIPE) IP core provides a 1-bit PHY status (`pipe_phystatus`) and a 3-bit receiver status signal (`pipe_rxstatus[2:0]`) to indicate whether a receiver was detected or not, in accordance to the PIPE 2.0 specifications.

Clock Rate Compensation Up to ±300 ppm

In compliance with the PCIe protocol, the Arria V receiver channels are equipped with a rate match FIFO to compensate for the small clock frequency differences of up to ±300 ppm between the upstream transmitter and local receiver clocks.

For more information about the rate match FIFO, refer to the "Receiver PCS Datapath" section in the *Transceiver Basics for Arria V Devices* chapter.

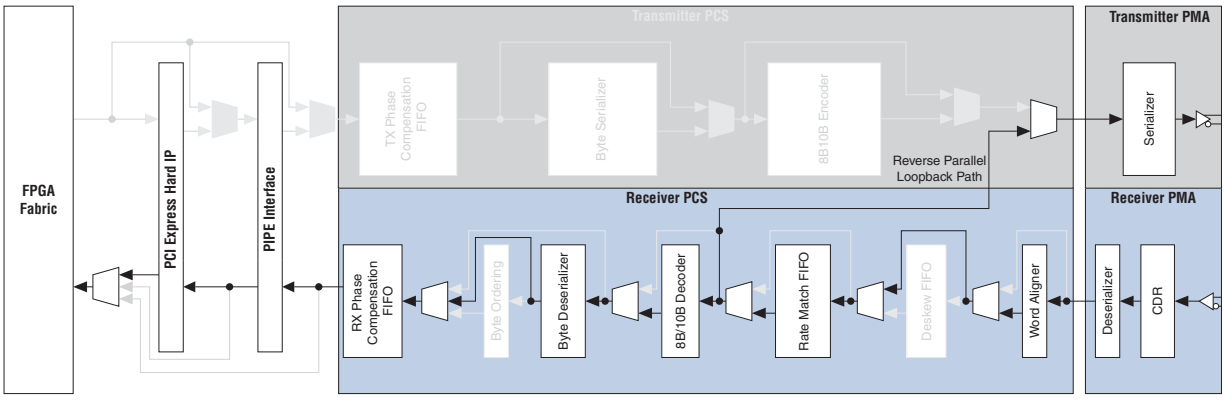
PCIe Reverse Parallel Loopback

The PCIe reverse parallel loopback is only available in the PCIe functional configuration for the Gen1 data rate. The received serial data passes through the receiver CDR, deserializer, word aligner, and rate matching FIFO buffer, as shown in Figure 4–4. It is then looped back to the transmitter serializer and transmitted out through the transmitter buffer. The received data is also available to the FPGA fabric through the port. This loopback mode is compliant with the PCIe specification 2.0.

Arria V devices provide the `pipe_txdetectrx_loopback` input signal to enable this loopback mode. If the `pipe_txdetectrx_loopback` signal is asserted in the P1 power state, receiver detection is performed. If the signal is asserted in the P0 power state, reverse parallel loopback is performed.

The PCIe reverse parallel loopback is the only loopback option that is supported in PIPE configurations.

Figure 4–4. PIPE Reverse Parallel Loopback Mode Datapath *(1)*



Note to Figure 4–4:

(1) The grayed out blocks are inactive.

Transceiver Channel Placement Guidelines

Table 4–3 lists the physical placement of PIPE channels in x1, x2, x4, and x8 bonding configurations.

Table 4–3. PIPE Channel Placement for PCIe Gen1

Configuration	Data Channel Placement	Minimum Channel Utilization *(1)*
x1	Any channel	2 (1 data channel, 1 clock channel)
x2	2 contiguous channels	3 (2 data channels, 1 clock channel)
x4	4 contiguous channels	5 (4 data channels, 1 clock channel)
x8	8 contiguous channels	9 (8 data channels, 1 clock channel)

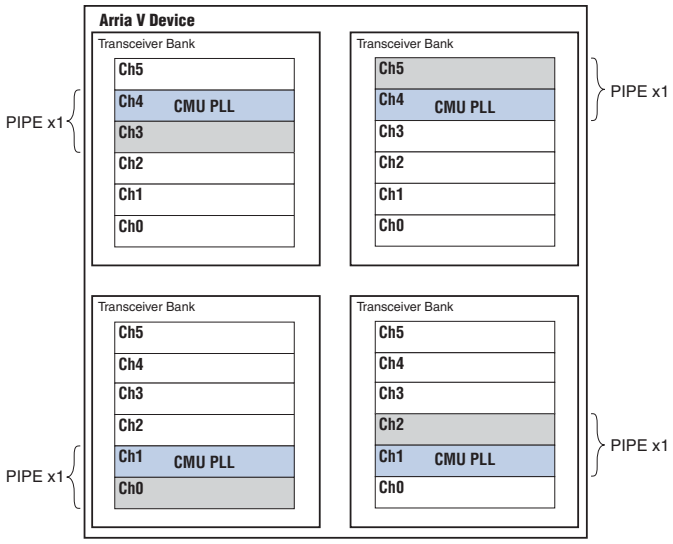
Notes to Table 4–3:

(1) Placement by the Quartus II software may vary with design—resulting in higher channel utilization.

(2) For more information about the hard IP implementation of PCIe and restrictions, refer to the "*Transceiver Banks*" section of the *Transceiver Architecture in Arria V Devices*.

Figure 4–5, Figure 4–6, and Figure 4–7 show examples of channel placement for PIPE x1, x2, x4, and x8 configurations.

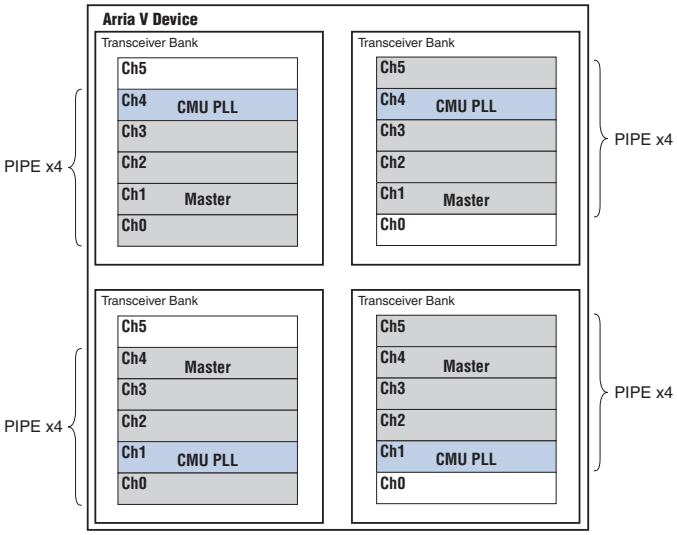
Figure 4–5. Example of PIPE x1 Channel Placement (1), (2), (3)



Notes to Figure 4–5:

(1) Channels shaded in blue provide the high-speed serial clock.
(2) Channels shaded in gray are data channels.
(3) You can place the PIPE data channels in any available channel in the transceiver bank.

Figure 4–6. Example of PIPE x2 and x4 Channel Placement (1), (2), (3), (4)



Notes to Figure 4–6:

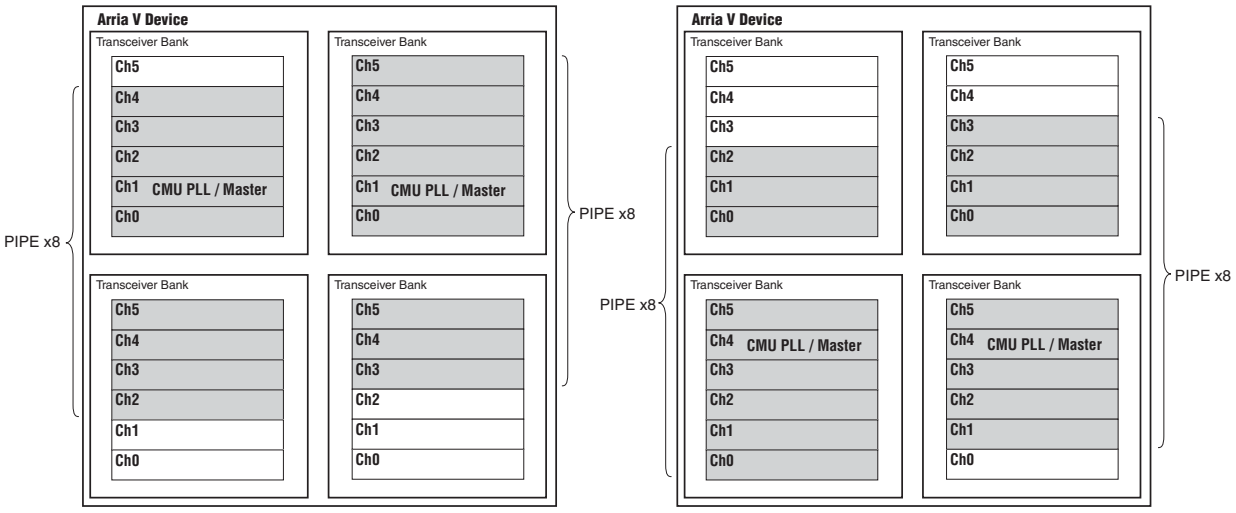
(1) Channels shaded in blue provide the high-speed serial clock.

(2) Channels shaded in gray are data channels.

(3) The Quartus II software automatically places the clock generator and master in either channels 1 or 4 within a transceiver bank. The master is referring to the transceiver channel with the central clock divider.

(4) The x2 channel placement for the clock generator and master channel is the same as the x4 channel placement. The other channel for the x2 configuration can be placed in any of the channels shaded in gray.

Figure 4–7. Example of PIPE x8 Channel Placement (1), (2)



Notes to Figure 4–7:

(1) Channels shaded in blue provide the high-speed serial clock.

(2) Channels shaded in gray are data channels.

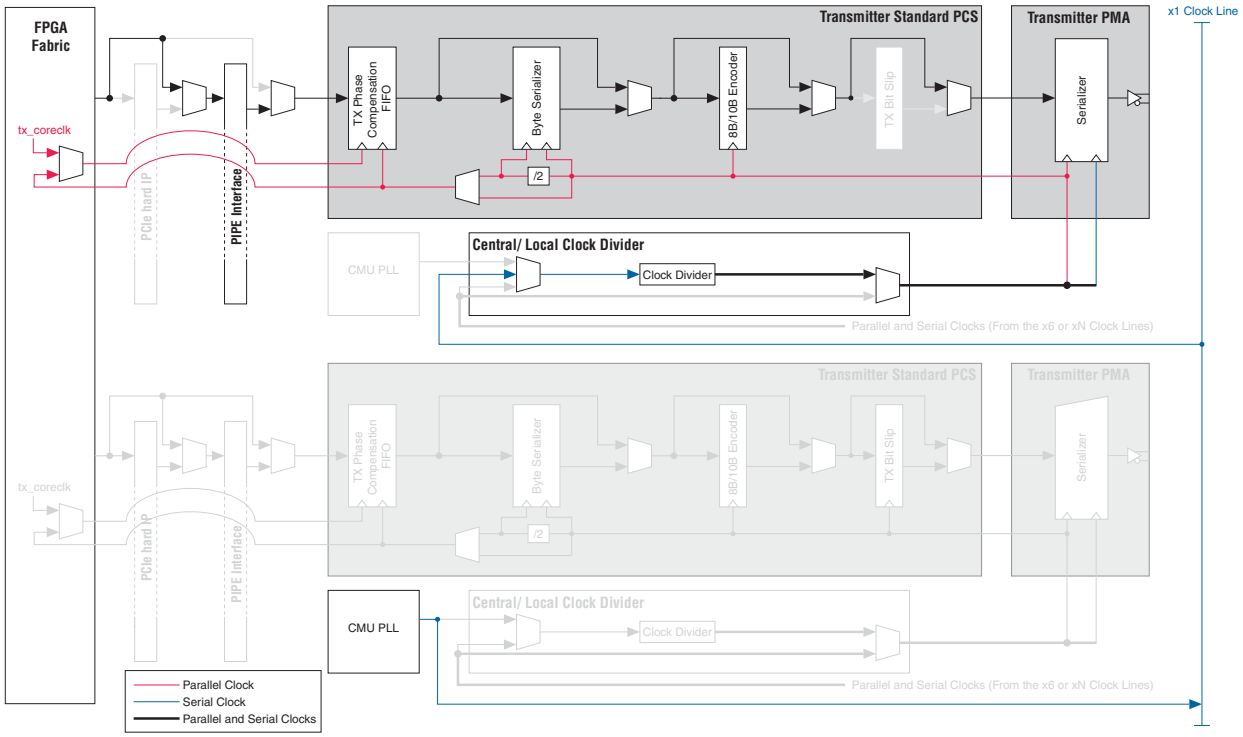
Transceiver Clocking

This section describes the transceiver clocking configurations for PIPE.

PIPE x1 Configuration

Figure 4–8 shows the transceiver clocking configuration in a PIPE x1 configuration. The serial clock is provided by the CMU PLL in a channel different from that of the data channel. The local clock divider block in the data channel generates a parallel clock from this high-speed clock and distributes both clocks to the PMA and PCS of the data channel.

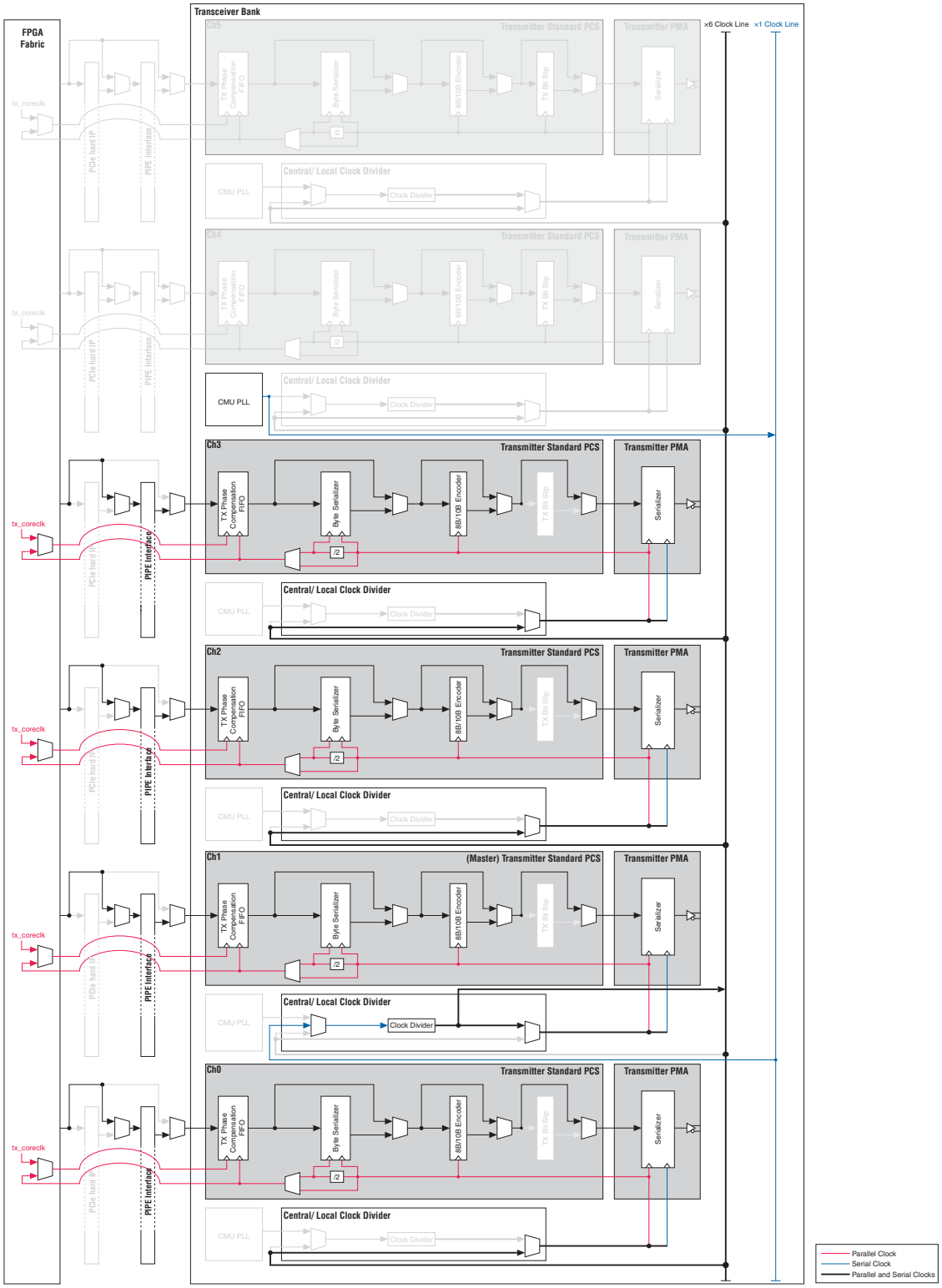
Figure 4–8. Transceiver Clocking Configuration in a PIPE x1 Configuration



PIPE x4 Configuration

Figure 4–9 shows the transmitter clocking for a PIPE x4 bonded configuration. Clocking is independent for the receiver channels. The clocking and control signals are bonded only for the transmitter channels.

Figure 4–9. Transceiver Clocking Configuration in a PIPE x4 Configuration (1)



Note to Figure 4–9:

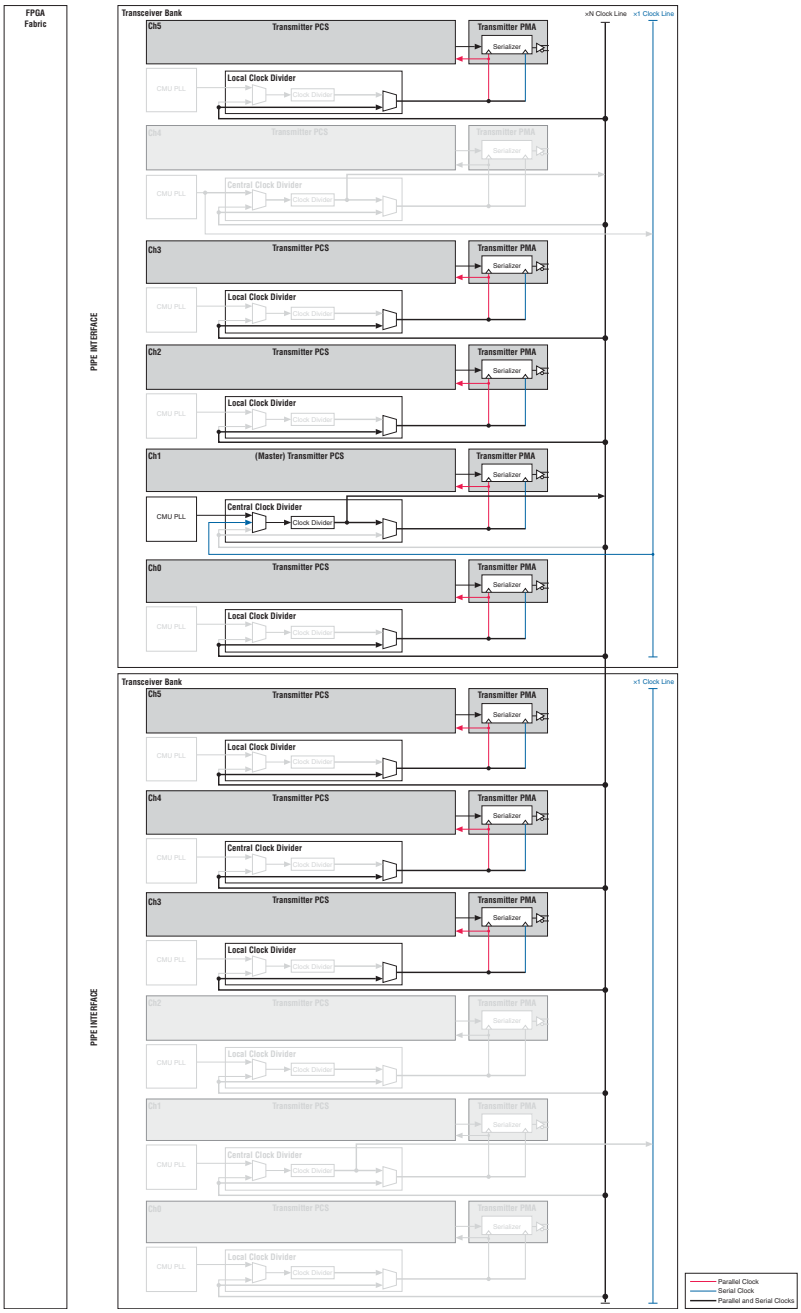
(1) The x2 channel placement for the clock generator (CMU PLL) and master channel is the same as the x4 channel placement. The other channel for the x2 configuration can be placed in either of the channels shaded in gray.

PIPE x8 Configuration

Figure 4–10 shows the clocking for the PMA and PCS blocks in the PIPE x8 bonded configuration. The clocking is independent for the receiver channels. The clocking and control signals are bonded only for the transmitter channels.

For more information about clocking in Arria V devices, refer to the *Transceiver Clocking in Arria V Devices* chapter.

Figure 4–10. Transceiver Clocking Configuration in a PIPE x8 Configuration



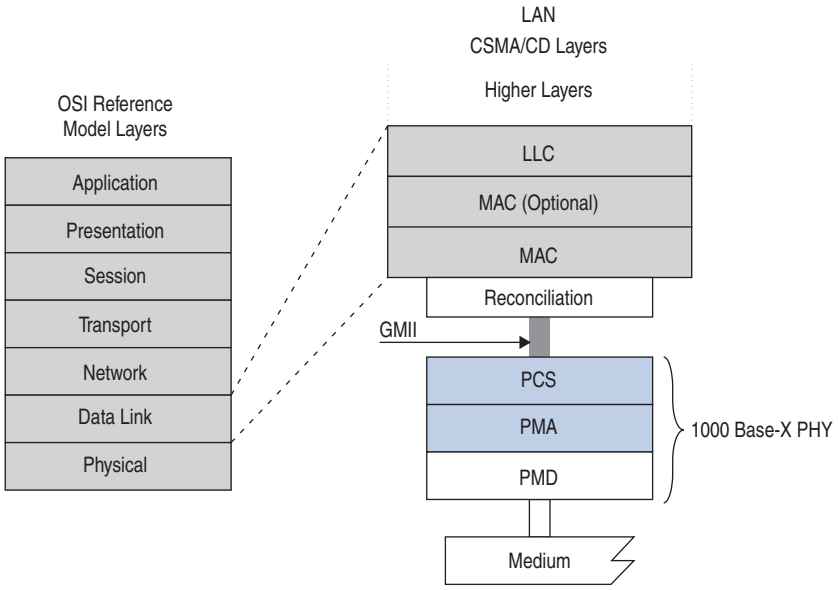
For more information about PCI Express PHY IP implementation, refer to the *"PCI Express PHY IP Core"* chapter in the *Altera Transceiver PHY IP Core User Guide*.

Gigabit Ethernet

The IEEE 802.3 specification defines the 1000BASE-X PHY as an intermediate, or transition layer that interfaces various physical media with the MAC in a gigabit ethernet (GbE) system, shielding the MAC layer from the specific nature of the underlying medium. The 1000BASE-X PHY is divided into the PCS, PMA, and PMD sublayers.

The PCS sublayer interfaces with the MAC through the gigabit media independent interface (GMII). The 1000BASE-X PHY defines a physical interface data rate of 1 Gbps. Figure 4–11 shows the 1000BASE-X PHY position in a GbE Open Systems Interconnection (OSI) reference model.

Figure 4–11. 1000BASE-X PHY in a GbE OSI Reference Model



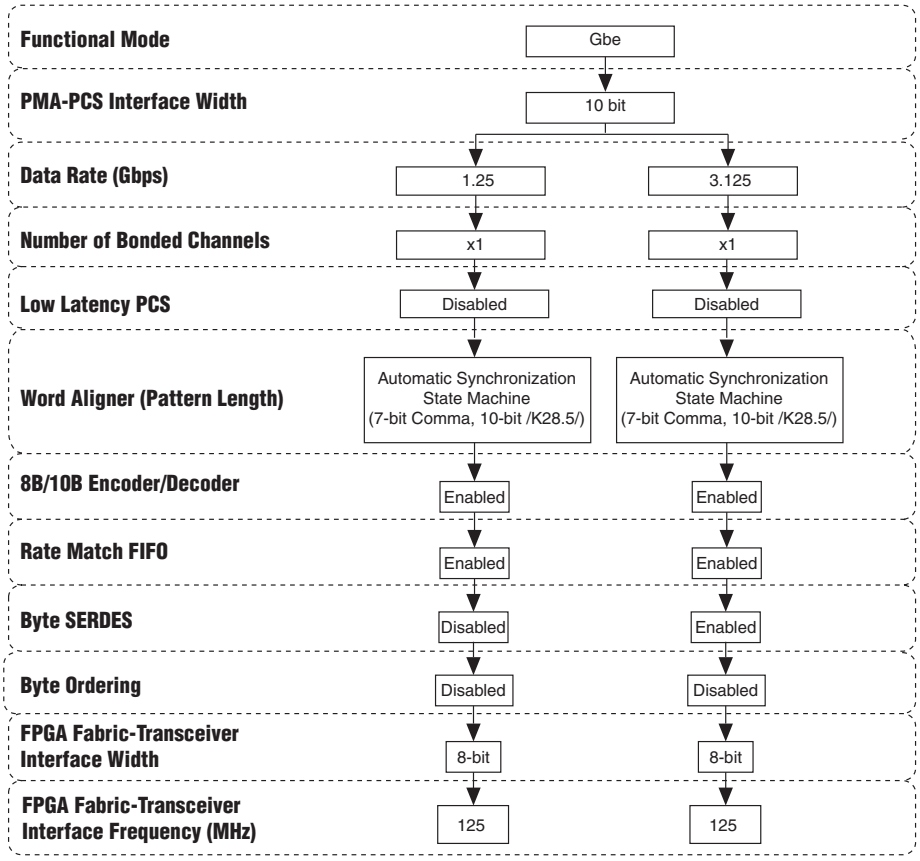
Arria V transceivers, when configured in GbE functional mode, have built-in circuitry to support the following PCS and PMA functions, as defined in the IEEE 802.3 specification:

- 8B/10B encoding and decoding
- Synchronization
- Upstream transmitter and local receiver clock frequency compensation (rate matching)
- Clock recovery from the encoded data forwarded by the receiver PMD
- Serialization and deserialization

Arria V transceivers do not have built-in support for other PCS functions, such as the autonegotiation state machine, collision-detect, and carrier-sense functions. If you require these functions, implement them in a programmable logic device (PLD) logic array or in external circuits.

Figure 4–12 shows the transceiver blocks that are enabled in a GbE configuration.

Figure 4–12. Arria V GX GbE Configuration



Transceiver Datapath

Figure 4–13 shows the transceiver datapath when configured in the GbE functional mode.

Figure 4–13. GbE Mode Datapath

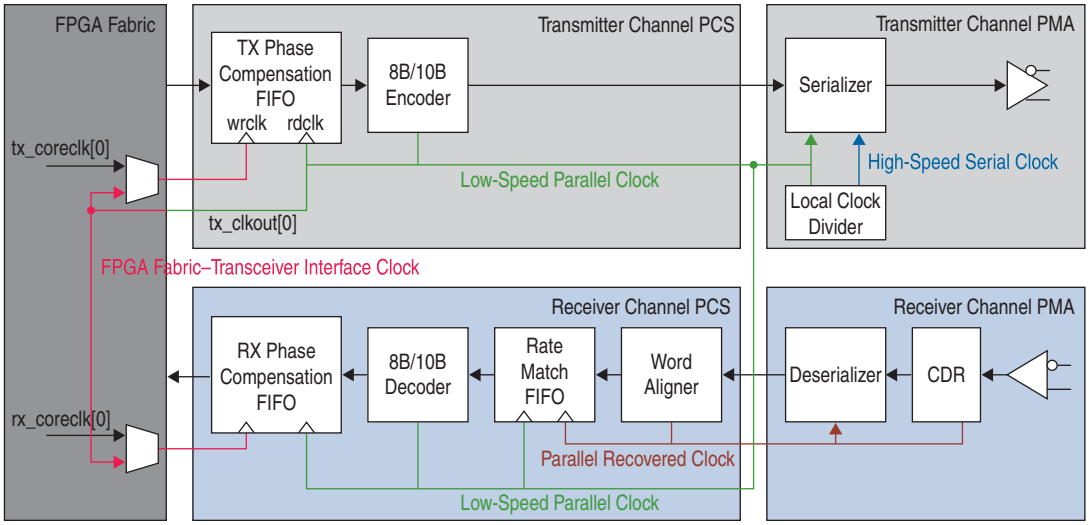


Table 4–4 lists the transceiver datapath clock frequencies in GbE functional mode.

Table 4–4. Transceiver Datapath Clock Frequencies in GbE Mode

Functional Mode	Data Rate	High-Speed Serial Clock Frequency	Parallel Recovered Clock and Low-Speed Parallel Clock Frequency	FPGA Fabric–Transceiver Interface Clock Frequency
GbE	1.25 Gbps	625 MHz	125 MHz	125 MHz
GbE	3.125 Gbps	625 MHz	125 MHz	125 MHz

8B/10B Encoder

In GbE mode, the 8B/10B encoder clocks in 8-bit data and 1-bit control identifiers from the transmitter phase compensation FIFO and generates 10-bit encoded data. The 10-bit encoded data is fed to the serializer.

For more information about the 8B/10B encoder functionality, refer to the "Transmitter PCS Datapath" section in the *Transceiver Basics for Arria V Devices* chapter.

Rate Match FIFO

In GbE mode, the rate match FIFO is capable of compensating for up to ±100 ppm (200 ppm total) difference between the upstream transmitter and the local receiver reference clock. The GbE protocol requires that the transmitter send idle ordered sets /I1/ (/K28.5/D5.6/) and /I2/ (/K28.5/D16.2/) during interpacket gaps, adhering to the rules listed in the IEEE 802.3 specification.

The rate match operation begins after the synchronization state machine in the word aligner indicates that the synchronization is acquired—by driving the rx_syncstatus signal high. The rate matcher deletes or inserts both symbols (/K28.5/ and /D16.2/) of the /I2/ ordered sets, even if it requires deleting only one symbol to prevent the rate match FIFO from overflowing or underrunning. The rate matcher can insert or delete as many /I2/ ordered sets as necessary to perform the rate match operation.

Two flags are forwarded to the FPGA fabric:

- rx_rmfifodatadeleted—Asserted for two clock cycles for each deleted /I2/ ordered set to indicate the rate match FIFO deletion event
- rx_rmfifodatainserted—Asserted for two clock cycles for each inserted /I2/ ordered set to indicate the rate match FIFO insertion event

For more information about the rate match FIFO, refer to the "Receiver PCS Datapath" section in the *Transceiver Basics for Arria V Devices* chapter.

GbE Protocol-Ordered Sets and Special Code Groups

Table 4–5 lists the ordered sets and special code groups, as specified in the IEEE 802.3-2008 specification.

Table 4–5. GIGE Ordered Sets

Code	Ordered Set	Number of Code Groups	Encoding
/C/	**Configuration**	—	Alternating /C1/ and /C2/
/C1/	Configuration 1	4	/K28.5/D21.5/Config_Reg (1)
/C2/	Configuration 2	4	/K28.5/D2.2/Config_Reg (1)
/I/	**IDLE**	—	Correcting /I1/, Preserving /I2/
/I1/	IDLE 1	2	/K28.5/D5.6/
/I2/	IDLE 2	2	/K28.5/D16.2/
—	**Encapsulation**	—	—
/R/	Carrier_Extend	1	/K23.7/
/S/	Start_of_Packet	1	/K27.7/
/T/	End_of_Packet	1	/K29.7/
/V/	Error_Propagation	1	/K30.7/

Note to Table 4–5:

(1) Two data code groups representing the Config_Reg value.

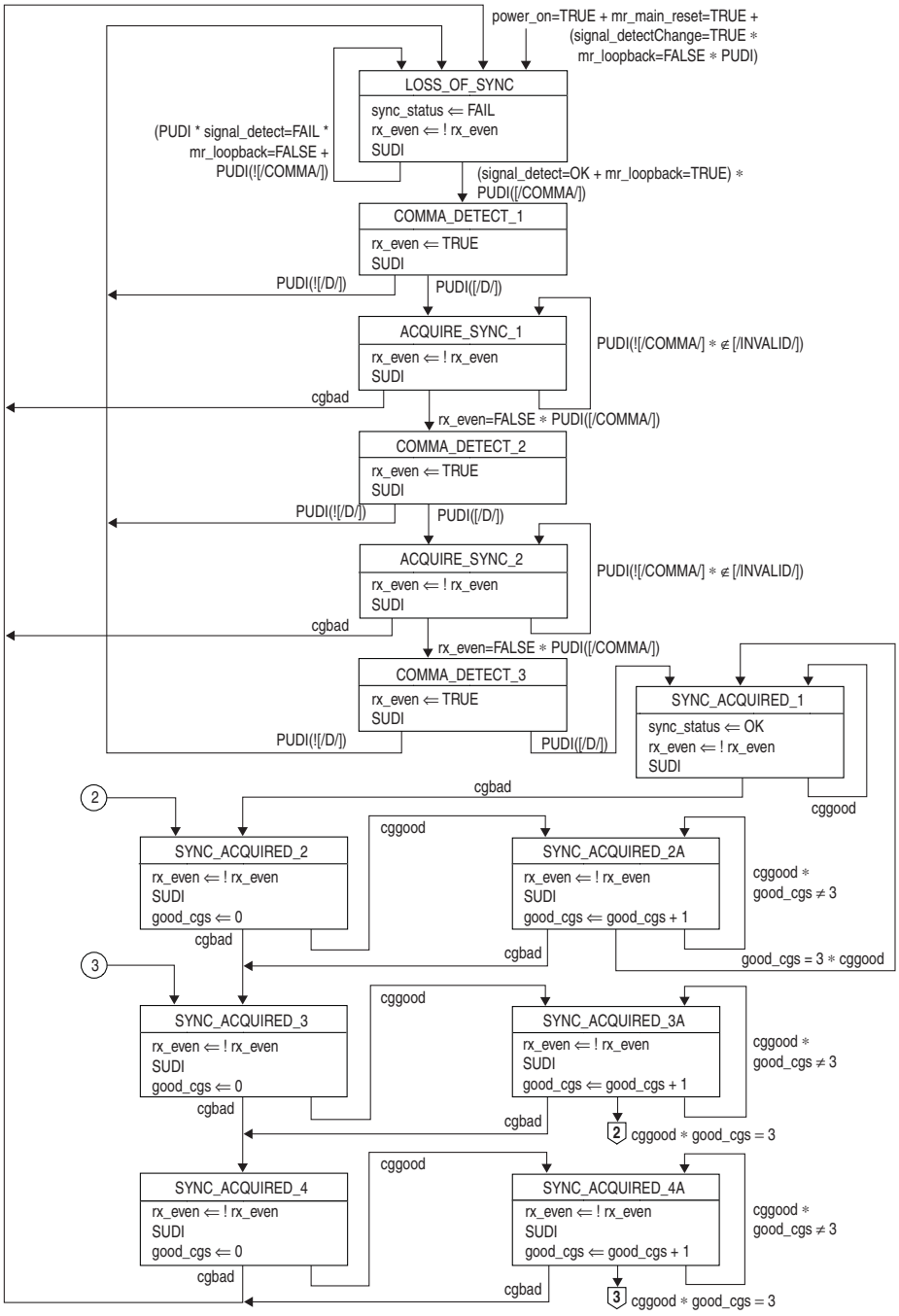
Table 4–6 lists the synchronization state machine parameters in GbE functional mode.

Table 4–6. Synchronization State Machine Parameters in GbE Mode

Synchronization State Machine Parameters	Setting
Number of valid {/K28.5/, /Dx,y/} ordered sets received to achieve synchronization	3
Number of errors received to lose synchronization	4
Number of continuous good code groups received to reduce the error count by 1	4

Figure 4–14 shows the synchronization state machine implemented in GbE mode.

Figure 4–14. Synchronization State Machine in GbE Mode *(1)*



Note to Figure 4–14:

(1) This figure is from "Figure 36–9" in the IEEE 802.3-2008 specification. For more details about the 1000BASE-X implementation, refer to Clause 36 of the IEEE 802.3-2008 specification.

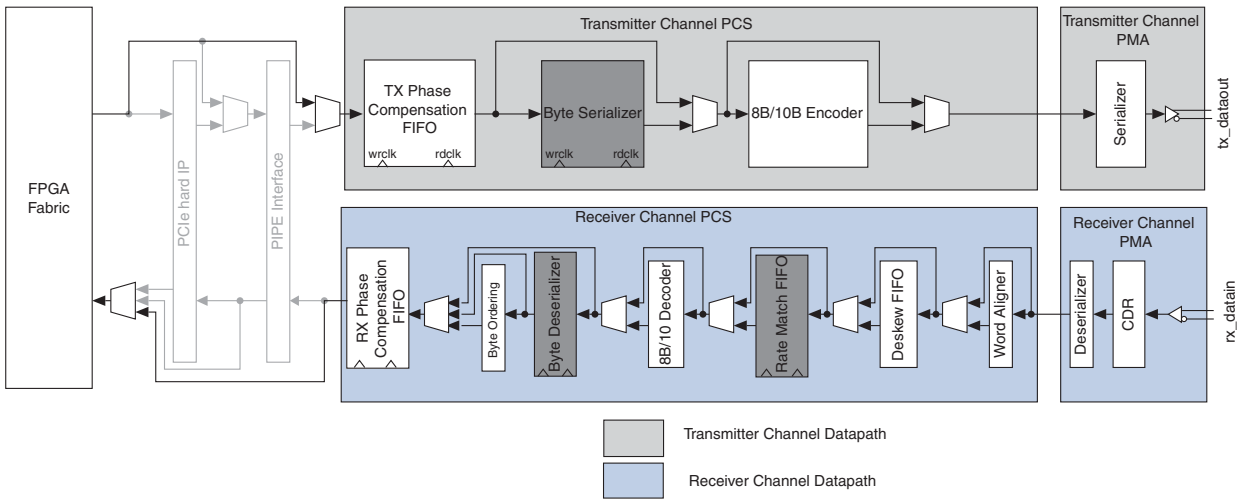
For more information about Gigabit Ethernet implementation in a Custom PHY IP, refer to the *Custom PHY IP Core* chapter in the *Altera Transceiver PHY IP Core User Guide*.

Deterministic Latency Protocols—CPRI and OBSAI

The Arria V device has a deterministic latency option available for use in high-speed serial interfaces such as the Common Public Radio Interface (CPRI) and OBSAI Reference Point 3 (OBSAI RP3). Both CPRI and OBSAI RP3 protocols place stringent requirements on the amount of latency variation that is permissible through a link that implements these protocols.

Figure 4–15 shows the transceiver datapath when using deterministic latency mode.

Figure 4–15. Transceiver Datapath in Deterministic Latency Mode



Receiver Phase Compensation FIFO in Register Mode

To remove the latency uncertainty through the receiver's phase compensation FIFO, the receiver and transmitter phase compensation FIFOs are always set to register mode. In register mode, the phase compensation FIFO acts as a register and thereby removes the uncertainty in latency. The latency through the phase compensation FIFO in register mode is one clock cycle.

The following options are available:

- Single-width mode with 8-bit channel width and 8B/10B Encoder enabled or 10-bit channel width with 8B/10B disabled
- Double-width mode with 16-bit channel width and 8B/10B encoder enabled or 20-bit channel width with 8B/10B disabled

Transmitter Phase Compensation FIFO in Register Mode

In register mode, the phase compensation FIFO acts as a register and thereby removes the uncertainty in latency. The latency through the transmitter and receiver phase compensation FIFO in register mode is one clock cycle.

Channel PLL Feedback

To implement the deterministic latency functional mode, the phase relationship between the low-speed parallel clock and channel PLL input reference clock must be deterministic. A feedback path is enabled to ensure a deterministic relationship between the low-speed parallel clock and channel PLL input reference clock.

To achieve deterministic latency through the transceiver, the reference clock to the channel PLL must be the same as the low-speed parallel clock. For example, if you need to implement a data rate of 1.2288 Gbps for the CPRI protocol, which places stringent requirements on the amount of latency variation, you must choose a reference clock of 122.88 MHz to allow the usage of a feedback path from the channel PLL. This feedback path reduces the variations in latency.

When you select this option, provide an input reference clock to the channel PLL that is of the same frequency as the low-speed parallel clock.

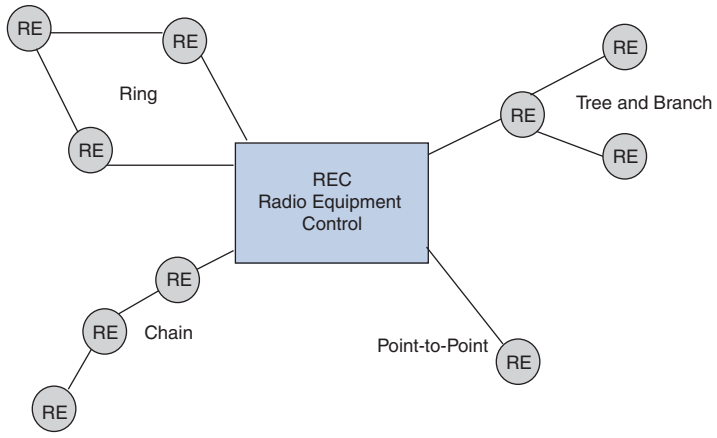
CPRI and OBSAI

You can use the deterministic latency functional mode to implement protocols such as CPRI and OBSAI.

The CPRI interface defines a digital point-to-point interface between the Radio Equipment Control (REC) and the Radio Equipment (RE), allowing flexibility in either co-locating the REC and the RE, or a remote location of the RE.

Figure 4–16 shows various CPRI topologies. In most cases, CPRI links are between REC and RE modules or between two RE modules in a chain configuration.

Figure 4–16. CPRI Topologies



If the destination for the high-speed serial data that leaves the REC is the first RE, it is a single-hop connection. If the serial data from the REC must traverse through multiple REs before reaching the destination RE, it is a multi-hop connection.

Remotely locating the RF transceiver from the main base station introduces a complexity with overall system delay. The CPRI specification requires that the accuracy of measurement of roundtrip delay on single-hop and multi-hop connections be within ±16.276 ns to properly estimate the cable delay.

For a single-hop system, this allows a variation in roundtrip delay of up to ±16.276 ns. However, for multi-hop systems, the allowed delay variation is divided among the number of hops in the connection—typically, equal to ±16.276 ns/(the number of hops) but not always equally divided among the hops.

Deterministic latency on a CPRI link also enables highly accurate triangulation of the location of the caller.

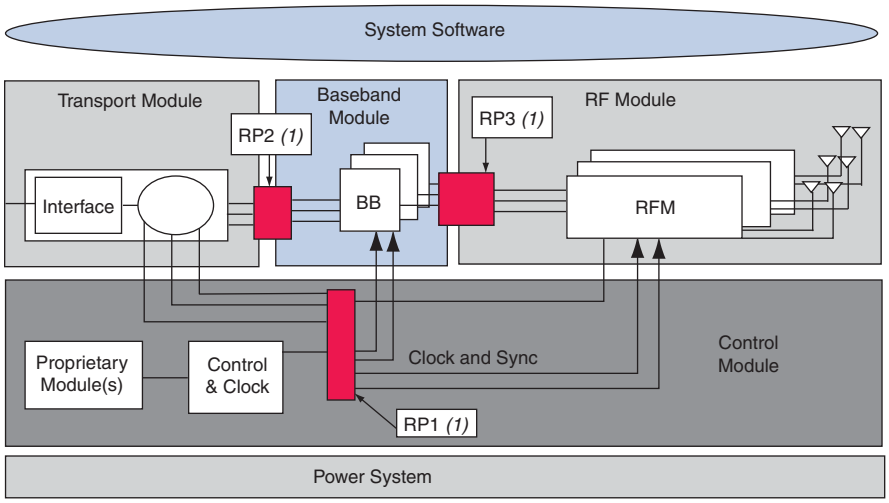
OBSAI was established by several OEMs to develop a set of specifications that can be used for configuring and connecting common modules into base transceiver stations (BTS).

The BTS has four main modules:

- Radio frequency (RF)
- Baseband
- Control
- Transport

Figure 4–17 shows a typical BTS. The radio frequency module (RFM) receives signals using portable devices and converts the signals to digital data. The baseband module processes the encoded signal and brings it back to the baseband before transmitting it to the terrestrial network using the transport module. A control module maintains the coordination between these three functions.

Figure 4–17. Example of the OBSAI BTS Architecture



Note to Figure 4–17:

(1) "RP" means Reference Point.

Using the deterministic latency option, you can implement the CPRI data rates in the following modes:

- Single-width mode—with 8/10-bit channel width
- Double-width mode—with 16/20-bit channel width

Table 4–4 lists sample channel width options, and CPRI and OBSAI data rates in the deterministic latency mode.

Table 4–7. Sample Channel Width Options for Supported Serial Data Rates

Serial Data Rate (Mbps)	Channel Width (FPGA-PCS Fabric)			
	Single Width		Double-Width	
	8-Bit	16-Bit	16-Bit	32-Bit
614.4	✓	✓	—	—
1228.8	✓	✓	✓	✓
2457.6	—	✓	✓	✓
3072	—	✓ (1)	✓ (1)	✓
4915.2	—	—	—	✓ (1)
6144	—	—	—	✓ (1)

Note to Table 4–4:

(1) Applicable to C4, C5, and I5 speed grades only.

CPRI Enhancements in Arria V Devices

Enhancements in Arria V transceivers over Stratix IV transceivers support the deterministic latency requirements in the CPRI specification. The following list describes the enhancements:

- New feature for deterministic by-2 deserialization, in which the byte deserializer ensures that the word alignment pattern is always placed in the least significant bit (LSB) with a fixed latency of three cycles.
- Deterministic latency state machine in the word aligner reduces the known delay variation from the word alignment process and automatically synchronizes and aligns the word boundary by slipping a clock cycle in the deserializer. Incoming data to the word aligner is aligned to the boundary of the word alignment pattern (K28.5). User logic is not required to manipulate the TX bit slipper for constant round-trip delay. In manual mode, the TX bit slipper is able to compensate one unit interval (UI). Figure 4–18 shows the deterministic latency state machine.

Figure 4–18. Deterministic Latency State Machine in the Word Aligner

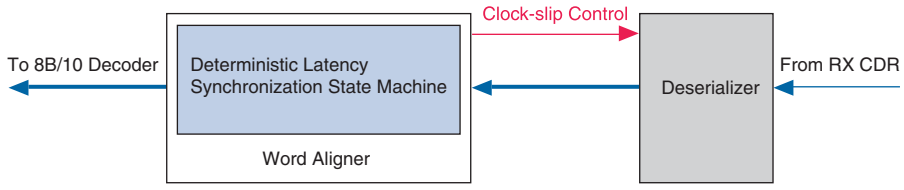


Table 4–8 lists the existing and new methods to achieve the deterministic latency mode in Arria V devices.

Table 4–8. Methods to Achieve Deterministic Latency Mode in Arria V Devices

Existing Feature (1)		New Feature (2)	
Description	**Requirement**	**Description**	**Requirement**
Manual alignment with bit position indicator provides deterministic latency. Delay variation up to 1 parallel clock cycle.	Extra user logic to manipulate the TX bit slipper with a bit position indicator from the word aligner for constant total round-trip delay.	Deterministic latency state machine alignment reduces the known delay variation in word alignment operation.	None.
K28.5 position varies in byte deserialized data. Delay variation up to ½ parallel clock cycle.	Extra user logic to manually check the K28.5 position in byte deserialized data for actual latency.	K28.5 is always placed in the least significant byte position in byte deserialized data. Fixed latency.	None.

Notes to Table 4–8:

(1) Backward compatible with CPRI designs in Arria II devices.

(2) New deterministic latency feature in Arria V devices.

For more information about deterministic latency PHY IP implementation, refer to the *Deterministic Latency PHY IP Core* chapter in the *Altera Transceiver PHY IP Core User Guide*.

Document Revision History

Table 4–9 lists the revision history for this document.

Table 4–9. Document Revision History

Date	Version	Changes
November 2011	1.1	Updated for the Quartus II software version 11.1.
August 2011	1.0	Initial release.

5. Transceiver Custom Configurations in Arria V Devices

AV53005-1.1

This chapter describes the custom transceiver datapath configuration for customized and proprietary high-speed serial interface (HSSI) implementations in Arria® V devices.

For integration with the FPGA fabric, the full-duplex transceiver channel supports custom configuration with physical medium attachment (PMA), physical coding sublayer (PCS), and low latency custom configurations with PMA and low latency PCS.

The 10-Gbps transceiver channel in Arria V GT devices supports PMA Direct configurations for data rates above 6.5536 gigabits per second (Gbps).

This chapter contains the following sections:

PCS Datapath Latency

Table 5–1 compares the latency incurred in each available PCS block between custom and low latency custom configurations.

Table 5–1. PCS Datapath Latency *(1)*

Datapath	PCS Block	Custom Configuration		Low Latency Custom Configuration	
		Enabled	Disabled	Enabled	Disabled
Transmitter (TX)	TX Phase Compensation FIFO	3–4	1 *(1)*	3–4	—
	Byte Serializer	1–2	1	1–2	0
	8B/10B Encoder	1	1	—	1
	Transmitter Bit-Slip	0–1	0	—	1

Table 5–1. PCS Datapath Latency *(1)*

Datapath	PCS Block	Custom Configuration		Low Latency Custom Configuration	
		Enabled	Disabled	Enabled	Disabled
Receiver (RX)	Word Aligner	4–8	—	—	1
	Rate Match FIFO	11–14	0	—	0
	8B/10B Decoder	1	1	—	0
	Byte Deserializer	1–2	1	1–2	0
	Byte Ordering	1–3	1	—	0
	RX Phase Compensation FIFO	2–3	1 *(1)*	2–3	—

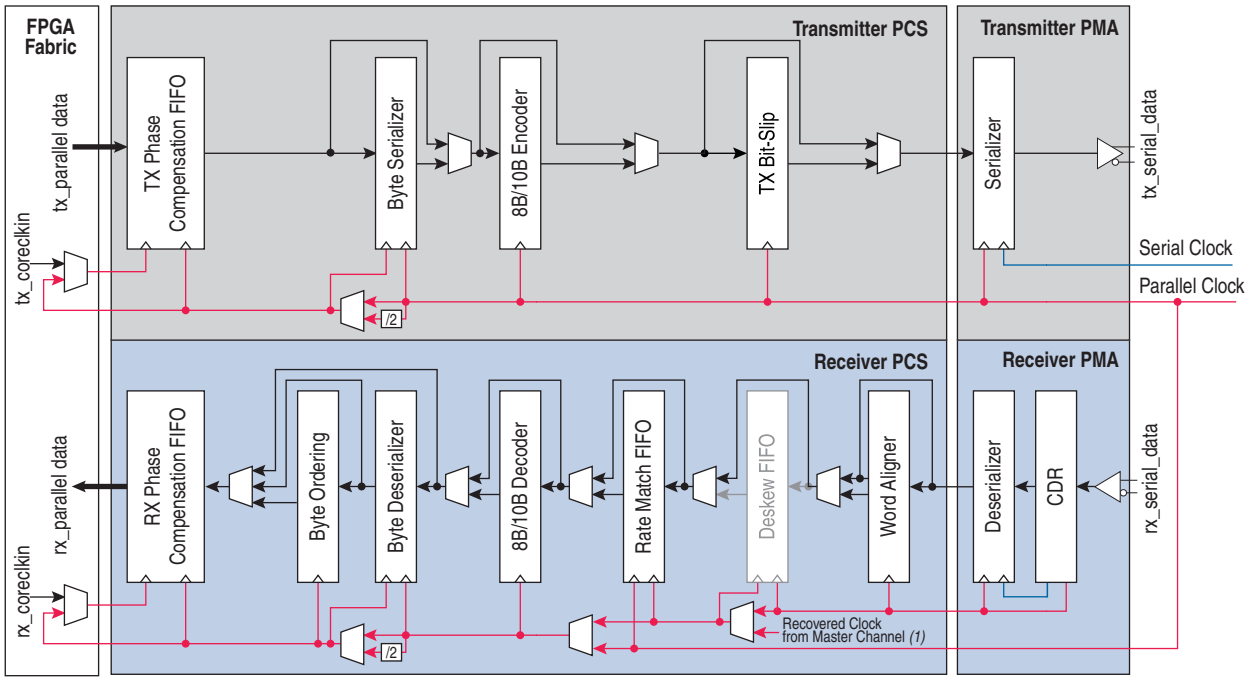
Note to Table 5–1:

(1) The registered mode FIFO is supported in deterministic latency configurations only.

Custom Configuration

In a custom configuration, you can customize the transceiver channel to include a PMA and PCS with functions that your application requires. The transceiver channel interfaces with the FPGA fabric through the PCS. Based on your application requirements, you can enable, modify, or disable the blocks, except the deskew FIFO block, as shown in Figure 5–1.

Figure 5–1. Compete Datapath in a Custom Configuration



Note to Figure 5–1:

(1) Used only for XAUI configurations. For more information, refer to the *Transceiver Protocol Configurations in Arria V Devices* chapter.

The maximum supported data rate varies depending on the customization. Table 5–2 lists an example of the maximum supported data rate in various custom configurations with the PMA and PCS using the fastest speed grade device.

Table 5–2. Maximum Supported Data Rate for Fastest Speed Grade Device (–4 Commercial) in Custom Configuration

Datapath Configuration	PMA–PCS Interface Width	PCS–FPGA Fabric Interface Width (1)		Maximum Data Rate (Mbps) (2), (3)
		8B/10B Enabled	8B/10B Disabled	
Single-width	8	—	8, 16	2500
	10	8, 16	10, 20	3125
Double-width	16	—	16, 32	5242.88
	20	16, 32	20, 40	6553.6

Note to Table 5–2:

(1) The supported interface width varies depending on the usage of the byte serializer/deserializer (SERDES), and the 8B/10B encoder or decoder.

(2) The byte serializer or deserializer is assumed to be enabled. Otherwise, the maximum data rate supported is half of the specified value.

(3) Data rates over 6553.6 Mbps are supported in PMA Direct mode. Refer to "PMA Direct Configuration" on page 5–13 for more information.

In all the supported configuration options of the channel, the transmitter bit-slip function is optional, as shown in Figure 5–2 through Figure 5–5.

Figure 5–2. Configuration Options for Custom Single-Width Mode (8-bit PMA–PCS Interface Width)

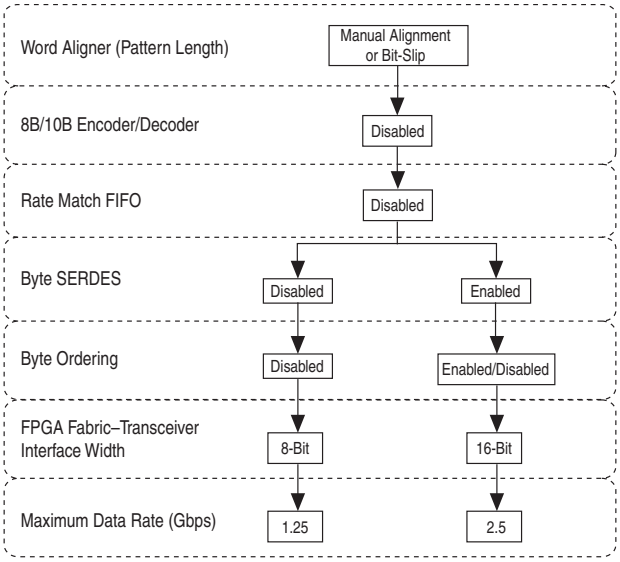


Figure 5–3. Configuration Options for Custom Single-Width Mode (10-bit PMA–PCS Interface Width)

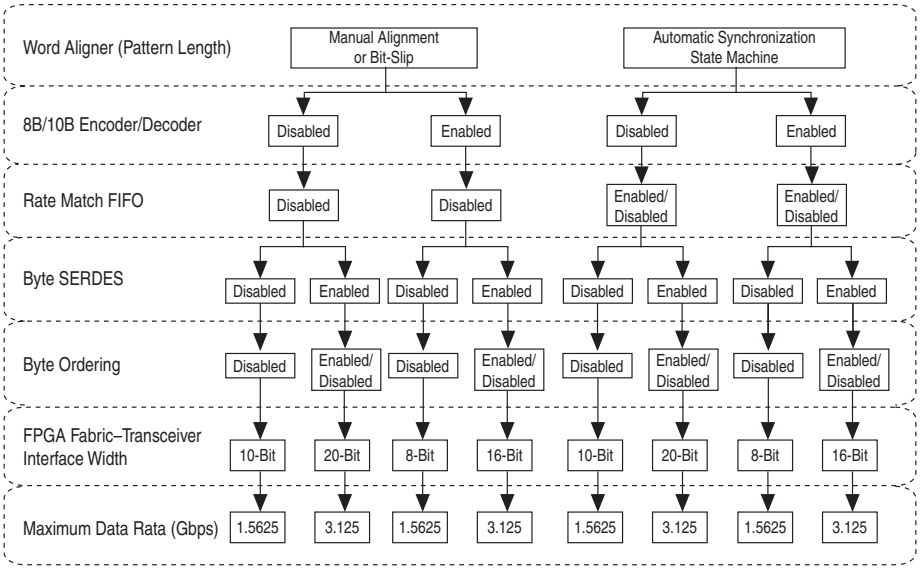


Figure 5–4. Configuration Options for Custom Double-Width Mode (16-bit PMA–PCS Interface Width)

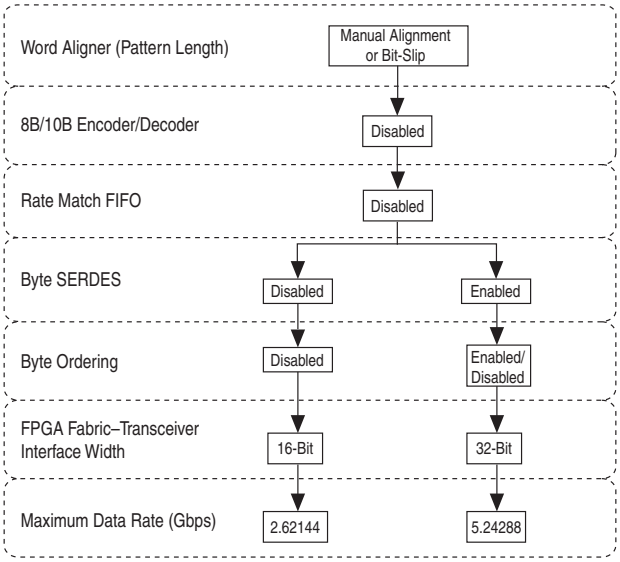
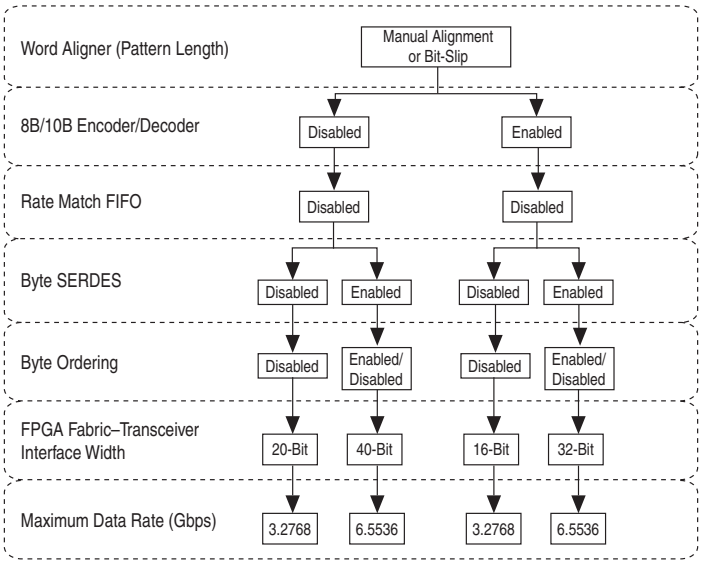


Figure 5–5. Configuration Options for Custom Double-Width Mode (20-bit PMA–PCS Interface Width)



For information about the operation of the blocks available in a custom configuration, excluding the rate match FIFO, refer to the *Transceiver Architecture in Arria V Devices* chapter.

In a custom configuration, the 20-bit pattern for the rate match FIFO is user-defined. The FIFO operates by looking for the 10-bit control pattern followed by the 10-bit skip pattern in the data, after the word aligner restores the word boundary. After finding the pattern, the FIFO performs a skip pattern insertion or deletion to ensure that the FIFO does not underflow or overflow a given part per million (ppm) difference between the clocks.

Table 5–3 lists the rate match FIFO behavior in a custom single-width mode configuration.

Table 5–3. Rate Match FIFO Behaviors in Custom Single-Width Mode (10-bit PMA–PCS Interface Width)

Operation	Behavior
Symbol Insertion	Inserts a maximum of four skip patterns in a cluster, only if there are no more than five skip patterns in the cluster after the symbol insertion.
Symbol Deletion	Deletes a maximum of four skip patterns in a cluster, only if there is one skip pattern left in the cluster after the symbol deletion.
Full Condition	Deletes the data byte that causes the FIFO to go full.
Empty Condition	Inserts a /K30.7/ (9'h1FE) after the data byte that caused the FIFO to go empty.

Table 5–4 lists the rate match FIFO behaviors in custom a double-width mode configuration.

Table 5–4. Rate Match FIFO Behaviors in Custom Double-Width Mode (20-bit PMA–PCS Interface Width) *(1)*

Operation	Behavior
Symbol Insertion	Inserts as many pairs (10-bit skip patterns at the LSByte and MSByte of the 20-bit word at the same clock cycle) of skip patterns as needed.
Symbol Deletion	Deletes as many pairs (10-bit skip patterns at the LSByte and MSByte of the 20-bit word at the same clock cycle) of skip patterns as needed.
Full Condition	Deletes the pair (20-bit word) of data bytes that causes the FIFO to go full.
Empty Condition	Inserts a pair of /K30.7/ ({9'h1FE, 9'h1FE}) after the data byte that causes the FIFO to go empty.

Note to Table 5–4:

(1) The rate match FIFO operation requires 8B/10B-coded data.

Figure 5–6 shows an example of rate match FIFO deletion in the case where three skip patterns are required to be deleted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern followed by two /K28.0/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by four /K28.0/ skip patterns. The rate match FIFO deletes only one /K28.0/ skip pattern from the first skip cluster to maintain at least one skip pattern in the cluster after deletion. Two /K28.0/ skip patterns are deleted from the second cluster for the three skip pattern deletion requirement.

Figure 5–6. Rate Match Deletion in Custom Single-Width Mode

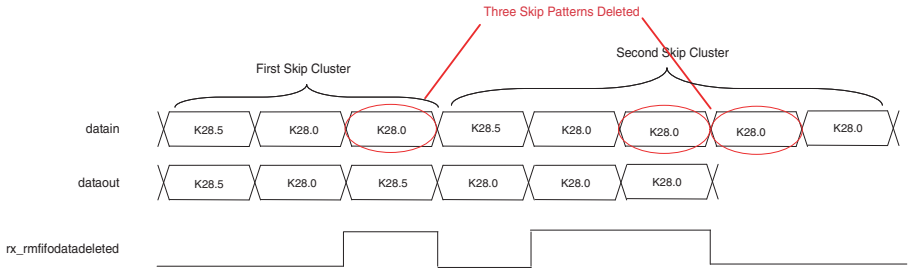


Figure 5–7 shows an example of rate match FIFO insertion in the case where three skip patterns are required to be inserted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern followed by three /K28.0/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by one /K28.0/ skip pattern. The rate match FIFO inserts only two /K28.0/ skip patterns into the first skip cluster to maintain a maximum of five skip patterns in the cluster after insertion. One /K28.0/ skip pattern is inserted into the second cluster for a total of three skip patterns to meet the insertion requirement.

Figure 5–7. Rate Match Insertion in Custom Single-Width Mode

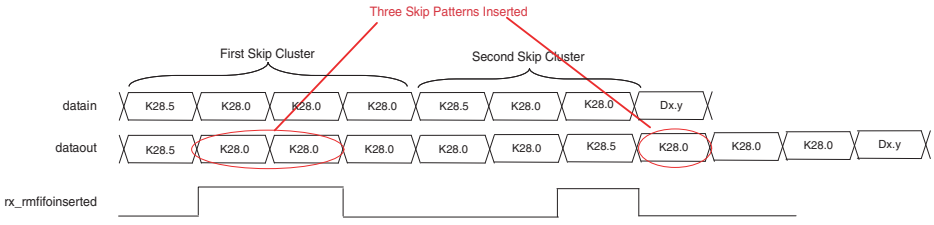


Figure 5–8 shows the rate match FIFO full condition in custom single-width mode. The rate match FIFO becomes full after receiving data byte D4.

Figure 5–8. Rate Match FIFO Full Condition in Custom Single-Width Mode

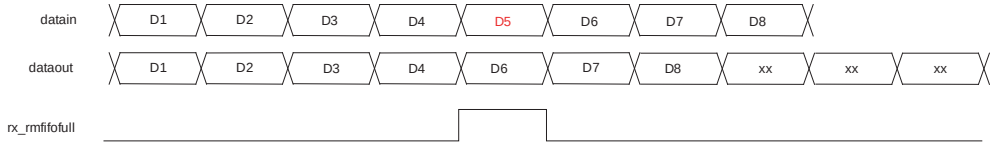


Figure 5–9 shows the rate match FIFO empty condition in custom single-width mode. The rate match FIFO becomes empty after reading out data byte D3.

Figure 5–9. Rate Match FIFO Empty Condition in Custom Single-Width Mode

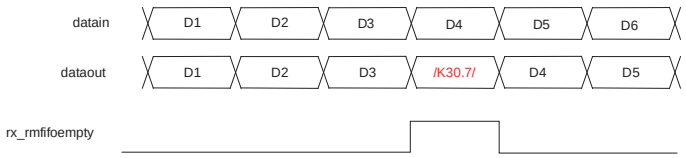


Figure 5–10 shows an example of rate match FIFO deletion in the case where three skip patterns are required to be deleted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern in the LSByte and /K28.0/ skip pattern in the MSByte of a clock cycle followed by one /K28.0/ skip pattern in the LSByte of the next clock cycle. The rate match FIFO cannot delete the two skip patterns in this skip cluster because they do not appear in the same clock cycle. The second skip cluster has a /K28.5/ control pattern in the MSByte of a clock cycle followed by two pairs of /K28.0/ skip patterns in the next two cycles. The rate match FIFO deletes both pairs of /K28.0/ skip patterns (for a total of four skip patterns deleted) from the second skip cluster to meet the three skip pattern deletion requirement.

Figure 5–10. Rate Match Deletion in Custom Double-Width Mode

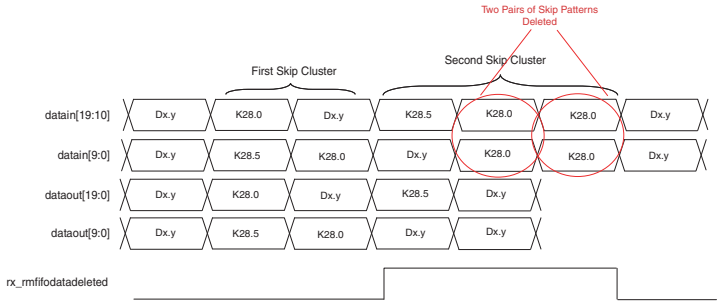


Figure 5–11 shows an example of rate match FIFO insertion in the case where three skip patterns are required to be inserted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern in the LSByte and /K28.0/ skip pattern in the MSByte of a clock cycle followed by one /K28.0/ skip pattern in the LSByte of the next clock cycle. The rate match FIFO inserts pairs of skip patterns in this skip cluster to meet the three skip pattern insertion requirement.

Figure 5–11. Rate Match Insertion in Custom Double-Width Mode

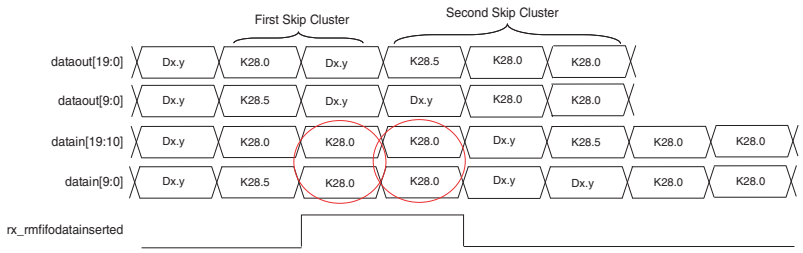


Figure 5–12 shows the rate match FIFO full condition in custom double-width mode. The rate match FIFO becomes full after receiving the 20-bit word D5D6.

Figure 5–12. Rate Match FIFO Full Condition in Custom Double-Width Mode

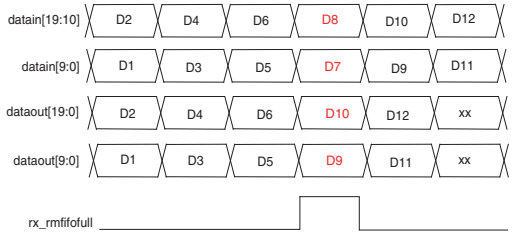
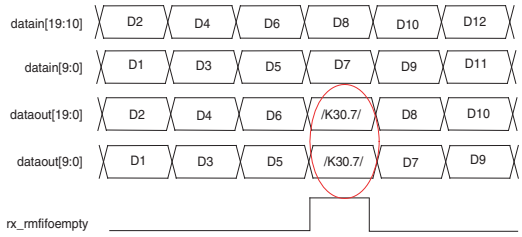


Figure 5–13 shows the rate match FIFO empty condition in custom double-width mode. The rate match FIFO becomes empty after reading out the 20-bit word D5D6.

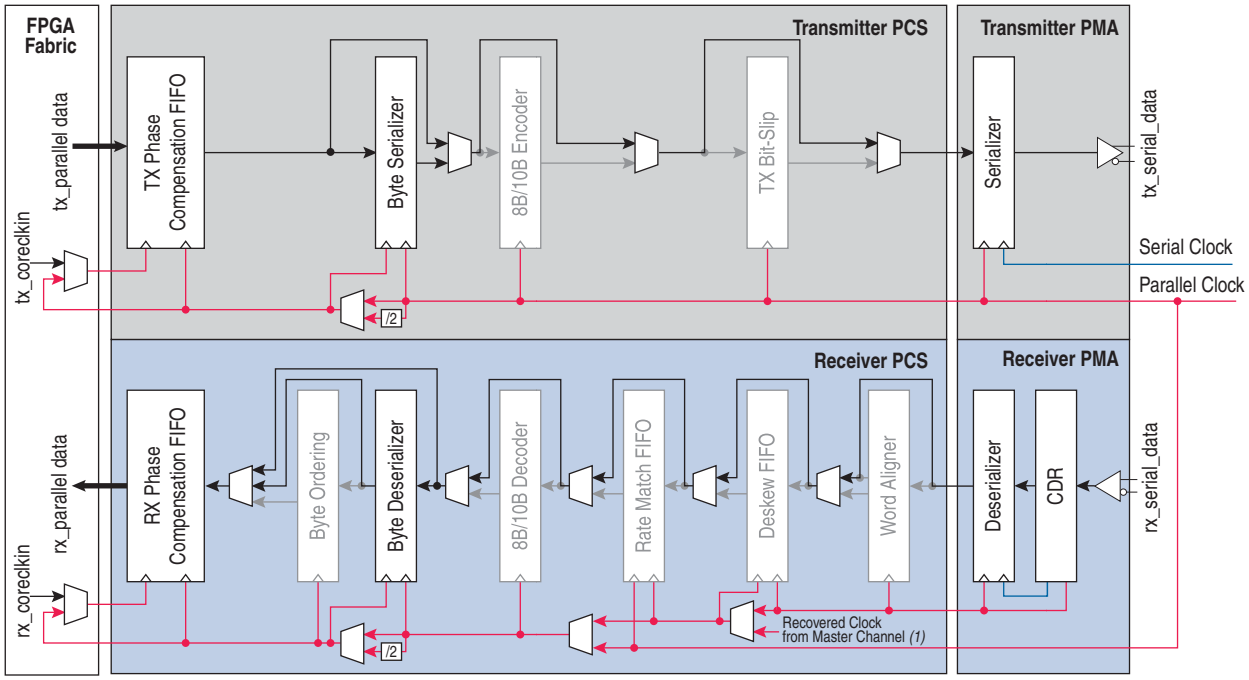
Figure 5–13. Rate Match FIFO Empty Condition in Custom Double-Width Mode



Low Latency Custom Configuration

In a low latency custom configuration, you can customize the transceiver channel to include a PMA and PCS that bypasses most of the PCS logical functionality for a low latency datapath. To provide a low latency datapath, the PCS includes only the phase compensation FIFO in phase compensation mode, and optionally, the byte serializer and byte deserializer blocks, as shown in Figure 5–14. The transceiver channel interfaces with the FPGA fabric through the PCS.

Figure 5–14. Complete Datapath in Low Latency Custom Configuration



Note to Figure 5–14:

(1) Used for XAUI configurations only. For more information, refer to the *Transceiver Protocol Configurations in Arria V Devices* chapter.

The maximum supported data rate varies depending on the customization and is identical to the custom configuration except that the 8B/10B block is disabled, as listed in Table 5–2 on page 5–3.

The supported configuration options of the channel are shown in Figure 5–15 through Figure 5–18, where:

- The blocks shown as "Disabled" are not used but incur latency.
- The blocks shown as "Bypassed" are not used and do not incur any latency.
- The transmitter bit-slip is disabled.

Figure 5–15. Configuration Options for Low Latency Custom Single-Width Mode (8-bit PMA–PCS Interface Width)

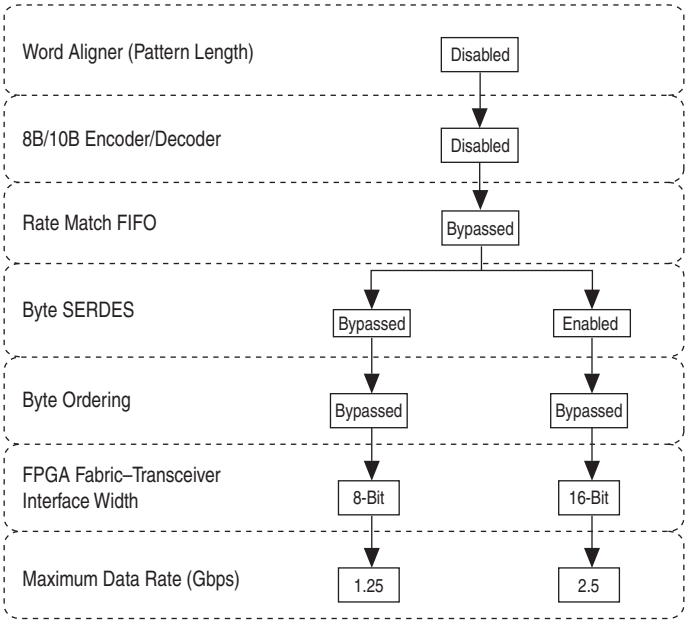


Figure 5–16. Configuration Options for Low Latency Custom Single-Width Mode (10-bit PMA–PCS Interface Width)

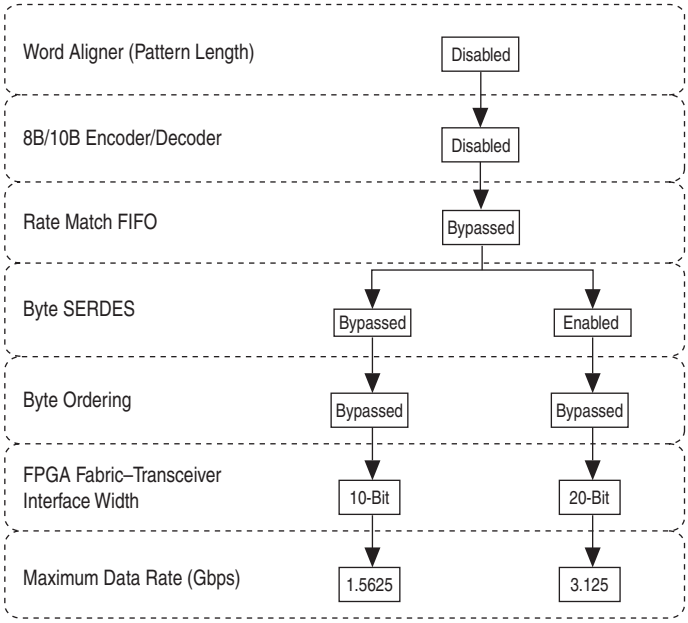


Figure 5–17. Configuration Options for Low Latency Custom Double-Width Mode (16-bit PMA–PCS Interface Width)

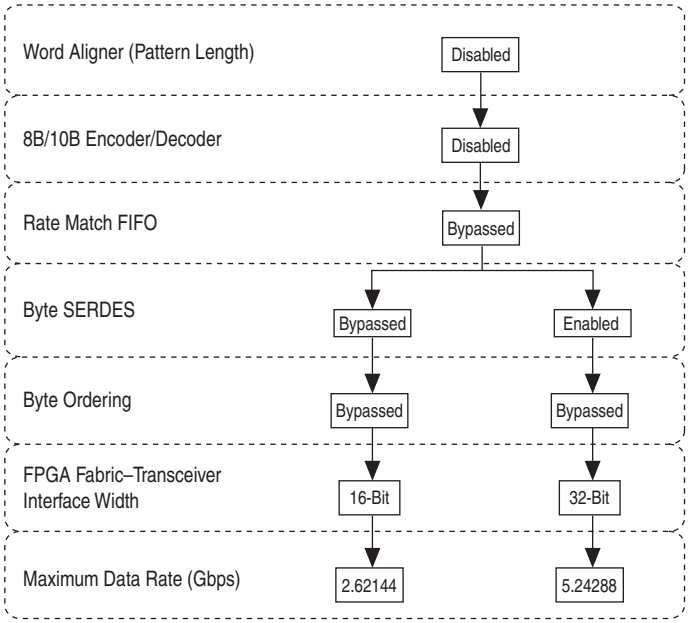
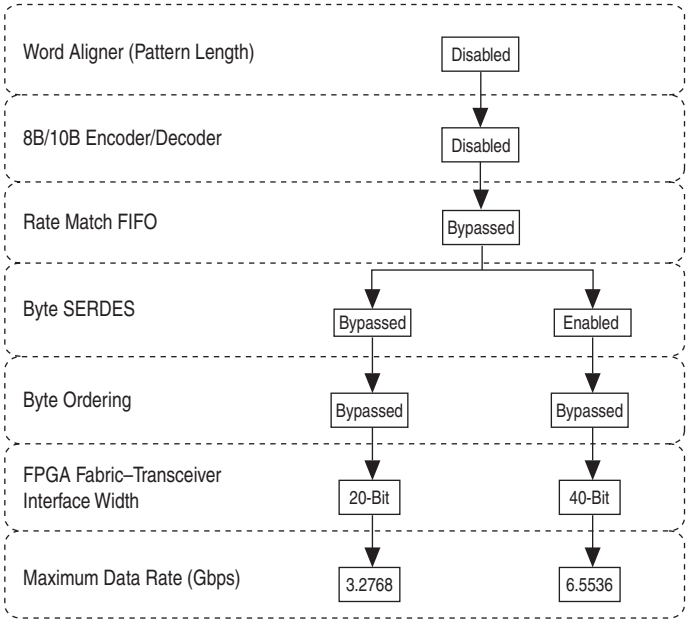


Figure 5–18. Configuration Options for Low Latency Custom Double-Width Mode (20-bit PMA–PCS Interface Width)



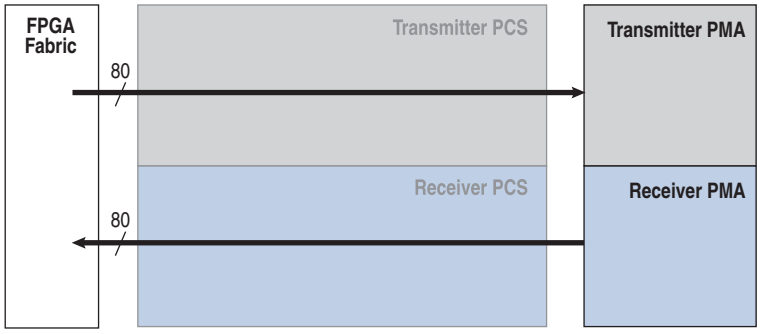
For information about the operation of the blocks available in a low latency custom configuration, refer to the *Transceiver Architecture in Arria V Devices* chapter.

PMA Direct Configuration

In a PMA Direct configuration, you can customize the transceiver channel to include only the PMA. In this configuration, the serializer and deserializer support an 80-bit deserialization factor and interfaces directly to the FPGA fabric, bypassing the PCS. You must implement the PCS functions in the FPGA fabric.

Figure 5–19 shows the transceiver datapath in a PMA Direct configuration.

Figure 5–19. Transceiver Datapath in a PMA Direct Configuration



The PMA Direct configuration is supported only for the 10-Gbps channels of Arria V GT devices at serial data rates above 6.5536 Gbps.

Document Revision History

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

Table 5–5. Document Revision History

Date	Version	Changes
November 2011	1.1	Updated for the Quartus II software release version 11.1.
August 2011	1.0	Initial release.

6. Transceiver Loopback Support in Arria V Devices

AV53006-1.1

This chapter describes the loopback options available for Arria® V devices, which allow you to verify how different functional blocks work in the transceiver.

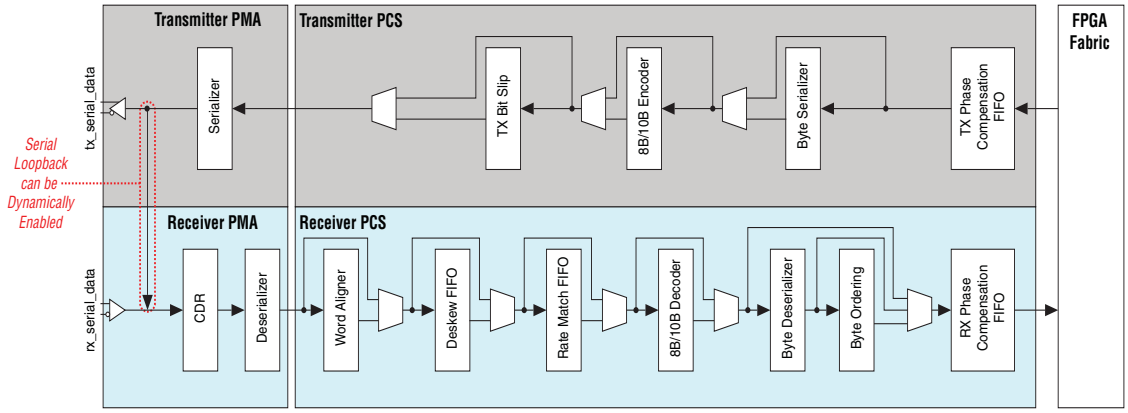
The available loopback options are:

Serial Loopback

Serial loopback is available for all transceiver configurations except the PIPE mode. You can use serial loopback as a debugging aid to ensure that the enabled physical coding sublayer (PCS) and physical media attachment (PMA) blocks in the transmitter and receiver channels are functioning correctly. Furthermore, you can dynamically enable serial loopback on a channel-by-channel basis.

Figure 6–1 shows the datapath for serial loopback. The data from the FPGA fabric passes through the transmitter channel and is looped back to the receiver channel, bypassing the receiver buffer. The received data is available to the FPGA logic for verification.

Figure 6–1. Serial Loopback Datapath



Subscribe

When you enable serial loopback, the transmitter channel sends data to both the `tx_serial_data` output port and to the receiver channel. The differential output voltage on the `tx_serial_data` port is based on the selected differential output voltage (V_{OD}) settings.

The looped-back data is forwarded to the receiver clock data recovery (CDR). You must provide an alignment pattern for the word aligner to enable the receiver channel to retrieve the byte boundary.

If the device is not in the serial loopback configuration and is receiving data from a remote device, the recovered clock from the receiver CDR is locked to the data from the remote source.

If the device is placed in the serial loopback configuration, the data source to the receiver changes from the remote device to the local transmitter channel—prompting the receiver CDR to start tracking the phase of the new data source. During this time, the recovered clock from the receiver CDR may be unstable. Because the receiver PCS is running off of this recovered clock, you must place the receiver PCS under reset by asserting the `rx_digitalreset` signal during this period.

When moving into or out of serial loopback, you must assert the `rx_digitalreset` signal for a minimum of two parallel clock cycles.

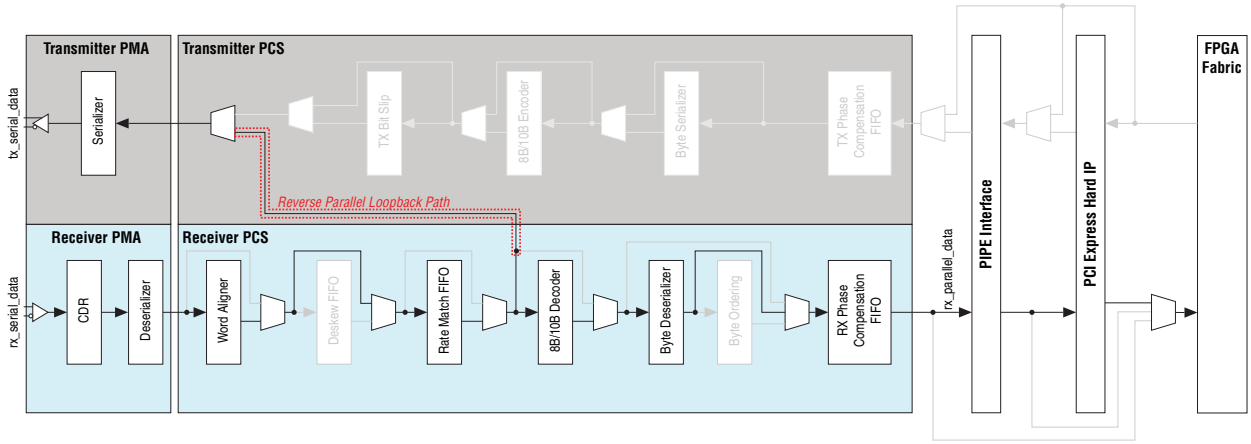
PIPE Reverse Parallel Loopback

PIPE reverse parallel loopback is only available in the PCIe® configuration for Gen1 and Gen2 data rates. Figure 6–2 shows the received serial data passing through the receiver CDR, deserializer, word aligner, and rate match FIFO buffer. The parallel data from the rate match FIFO is then looped back to the transmitter serializer and transmitted out through the `tx_serial_data` port. The received data is also available to the FPGA fabric through the `rx_parallel_data` signal.

PIPE reverse parallel loopback is compliant with the PCIe 2.0 specification. To enable this loopback configuration, assert the `pipe_txdetectrx_loopback` signal.

PIPE reverse parallel loopback is the only loopback option supported in the PCIe configuration.

Figure 6–2. PIPE Reverse Parallel Loopback Configuration Datapath *(1)*



Note to Figure 6–2:

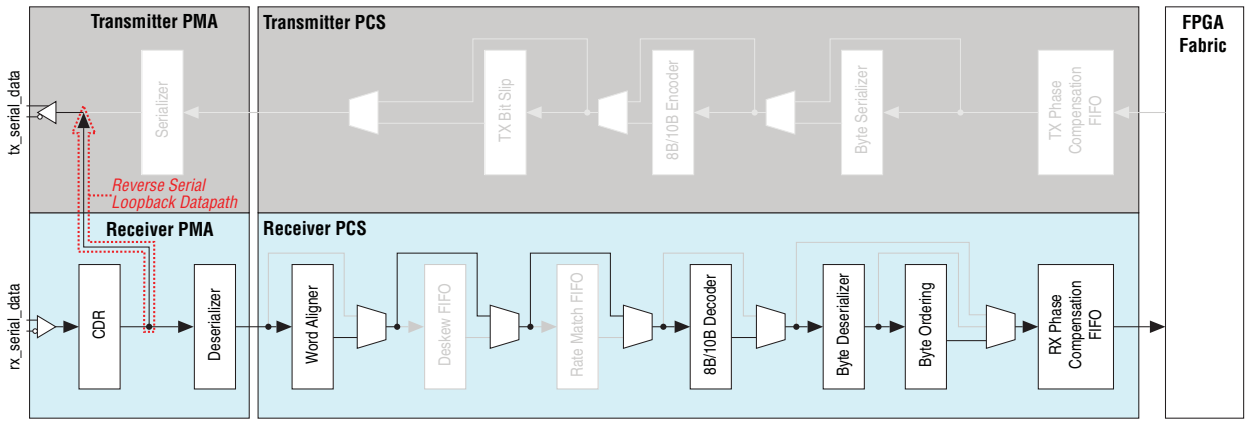
(1) Grayed-out blocks are not active when the PIPE reverse parallel loopback is enabled.

Reverse Serial Loopback

Reverse serial loopback is available as a subprotocol under custom configuration. In reverse serial loopback, the data is received through the `rx_serial_data` port, retimed through the receiver CDR, and sent to the `tx_serial_data` port. The received data is also available to the FPGA logic. No dynamic pin control is available to select or deselect reverse serial loopback. Figure 6–3 shows the transceiver channel datapath for reverse serial loopback mode.

The transmitter buffer is the only active block in the transmitter channel. You can change the VOD and the pre-emphasis first post tap values on the transmitter buffer through the ALTGX MegaWizard™ Plug-In Manager or through the dynamic reconfiguration controller. Reverse serial loopback is often implemented when using a bit error rate tester (BERT) on the upstream transmitter.

Figure 6–3. Reverse Serial Loopback Datapath *(1)*



Note to Figure 6–3:

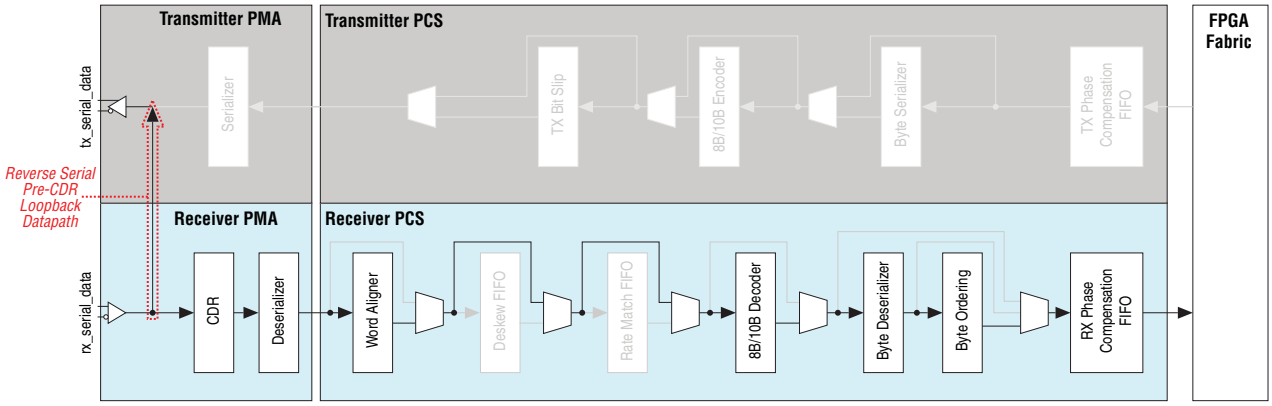
(1) Grayed-out blocks are not active when the reverse serial loopback is enabled.

Reverse Serial Pre-CDR Loopback

Reverse serial pre-CDR loopback is available as a subprotocol under custom configuration. In reverse serial pre-CDR loopback, the data received through the `rx_serial_data` port is looped back to the `tx_serial_data` port before the receiver CDR. The received data is also available to the FPGA logic. Figure 6–4 shows the transceiver channel datapath for reverse serial pre-CDR loopback mode.

The transmitter buffer is the only active block in the transmitter channel. You can change the VOD on the transmitter buffer through the ALTGX MegaWizard Plug-In Manager. The pre-emphasis settings for the transmitter buffer cannot be changed in this configuration.

Figure 6–4. Reverse Serial Pre-CDR Loopback Datapath *(1)*



Note to Figure 6–3:

(1) Grayed-out blocks are not active when the reverse serial pre-CDR loopback is enabled.

Document Revision History

Table 6–1 lists the revision history for this chapter.

Table 6–1. Document Revision History

Date	Version	Changes
November 2011	1.1	■ Added the following two loopback options: ■ "Reverse Serial Loopback" ■ "Reverse Serial Pre-CDR Loopback"
August 2011	1.0	Initial release.

7. Dynamic Reconfiguration in Arria V Devices

AV53007-1.1

This chapter describes the dynamic reconfiguration features available in Arria® V transceivers. Table 7–1 shows the reconfiguration applications, the blocks that are being reconfigured, and the reconfiguration modes that are supported by the Arria V transceivers.

Table 7–1. Arria V Reconfiguration Applications

Reconfiguration Application	Affected Blocks	Reconfiguration Mode
Offset Cancellation—Counter offset variations due to process operation for the analog circuit. This feature is mandatory if you use receivers.	CDR	Transceiver Calibration Functions
Analog Controls Reconfiguration—Fine-tune the signal integrity by adjusting the transmitter (TX) and receiver (RX) buffer settings while bringing up a link	Analog circuit of TX and RX buffer	PMA Analog Settings Reconfiguration Mode
Loopback Modes—Enable or disable Pre-CDR Reverse Serial Loopback or Post-CDR Reverse Serial Loopback dynamically	PMA	PMA Analog Settings Reconfiguration Mode

The transceiver reconfiguration controller offers several different reconfiguration modes. You can choose the appropriate reconfiguration modes that best suit your application needs. All the reconfiguration applications are done through the transceiver reconfiguration PHY IP.

For more information about the transceiver reconfiguration PHY IP, refer to the *Altera Transceiver PHY IP Core User Guide*.

This chapter contains the following sections:

Offset Cancellation

Every transceiver channel in Arria V devices has an offset cancellation circuitry to compensate for the offset variations that are caused by process operations.

The offset cancellation circuitry is controlled by the offset cancellation control logic IP within the transceiver reconfiguration controller. This IP is active only during device power up and it automatically performs offset cancellation. When the offset cancellation is completed, a `reconfig_busy` status signal from the reconfiguration controller is deasserted to indicate the completion of the process.

In Arria V devices the embedded reset controller automatically performs a reset sequence on the transceiver's channels after the offset cancellation process is complete. After the reset sequence is completed, the transceiver reconfiguration controller is ready for user-controlled operations.

PMA Analog Settings Reconfiguration

After offset cancellation is complete and the required transceiver reset sequence has been performed automatically by the embedded reset controller, you can dynamically reconfigure the analog controls setting. You can continue with the subsequent reconfigurations of the analog controls when the `reconfig_busy` status signal is low. A high on the `reconfig_busy` signal indicates that the reconfiguration operation is in progress.

You can reconfigure the following transceiver analog controls:

- Transmitter pre-emphasis
- Differential output voltage (V_{OD})
- Receiver equalizer control
- Direct-current (DC) gain settings

To reconfigure the analog control settings, perform `read` and `write` operations to the PMA analog settings reconfiguration control IP within the controller. For more information about the `read` and `write` operations, refer to the *Altera Transceiver PHY IP Core User Guide*.

Enabling and Disabling Loopback Modes

The following loopback paths are available:

- **Serial loopback path**—This mode takes the output at the serializer and sends the data to both the CDR's own receiver channel and the TX output port. Enabling or disabling the serial loopback mode is done through the PHY IP.
- **Pre- and Post-CDR reverse serial loopback path**—The RX captures the input data and feeds it into the CDR. The recovered data from the CDR output feeds into the TX driver and sends to the TX pins through the TX driver. For this path, the RX and CDR can be tested. For this path, the TX driver can be programmed to use either the main tap only or the main tap and the pre-emphasis first post-tap. Enabling or disabling the pre- and post-CDR reverse serial loopback modes is done through the PMA Analog Reconfiguration IP in the Transceiver Reconfiguration PHY IP.
- **Reverse serial diagnostic loopback path**—The RX captures the input data and feeds it back to the TX driver through a buffer. With this path, you can perform a quick check for the quality of the RX and TX buffers. Enabling or disabling the reverse serial diagnostic loopback mode is done through the analog IP in the Transceiver Reconfiguration PHY IP.

For implementation details about the different loopback modes, refer to "Loopback Modes" in the *Transceiver Reconfiguration Controller* chapter of the *Altera Transceiver PHY IP Core User Guide*.

Document Revision History

Table 7–2 lists the revision history for this chapter.

Table 7–2. Document Revision History

Date	Version	Changes
November 2011	1.1	■ Added Table 7–1. ■ Deleted "Transceiver Reconfiguration Controller" and "Channel and PLL Reconfiguration" sections. ■ Updated "Offset Cancellation" and "PMA Analog Settings Reconfiguration" sections ■ Added "Enabling and Disabling Loopback Modes" section. ■ Minor text edits.
August 2011	1.0	Initial release.

This chapter provides additional information about the document and Altera.

How to Contact Altera

To locate the most up-to-date information about Altera products, refer to the following table.

Contact *(1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note to Table:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, **\qdesigns** directory, **D:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicate document titles. For example, *Stratix IV Design Guidelines*.
italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>***.pof** file.
Initial Capital Letters	Indicate keyboard keys and menu names. For example, the Delete key and the Options menu.
"Subheading Title"	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, "Typographic Conventions."

Visual Cue	Meaning
`Courier type`	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. The suffix `n` denotes an active-low signal. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
↵	An angled arrow instructs you to press the Enter key.
1., 2., 3., and a., b., c., and so on	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■ ■	Bullets indicate a list of items when the sequence of the items is not important.
	The hand points to information that requires special attention.
	A question mark directs you to a software help system with related information.
	The feet direct you to another document or website with related information.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The envelope links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.

Arria V Device Handbook

Volume 4: Device Basics

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

AV-5V4-1.2

Document last updated for Altera Complete Design Suite version: 11.1
Document publication date: December 2011

Contents

Chapter 2. Transceiver Basics for Arria V Devices

Additional Information

Chapter Revision Dates

The chapters in this document, Arria V Device Handbook Volume 4: Device Basics, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1. Device Interfaces and Integration Basics for Arria V Devices
Revised: *December 2011*
Part Number: *AV-54001-1.2*

Chapter 2. Transceiver Basics for Arria V Devices
Revised: *November 2011*
Part Number: *AV-54002-1.1*

1. Device Interfaces and Integration Basics for Arria V Devices

AV-54001-1.2

This chapter contains basic technical details pertaining to specific features in the Arria® V device interfaces and integration. This chapter serves as a supplementary reading to the *Volume 2: Device Interfaces and Integration* of the *Arria V Device Handbook* and covers the following topics:

Logic Array Blocks and Adaptive Logic Modules

This section describes the basic information of the logic array block (LAB) and adaptive logic modules (ALMs) in Arria® V devices.

Use the information in this section in conjunction with the *Logic Array Blocks and Adaptive Logic Modules in Arria V Devices* chapter.

Logic Array Blocks

The LABs are configurable logic blocks that consist of a group of logic resources.

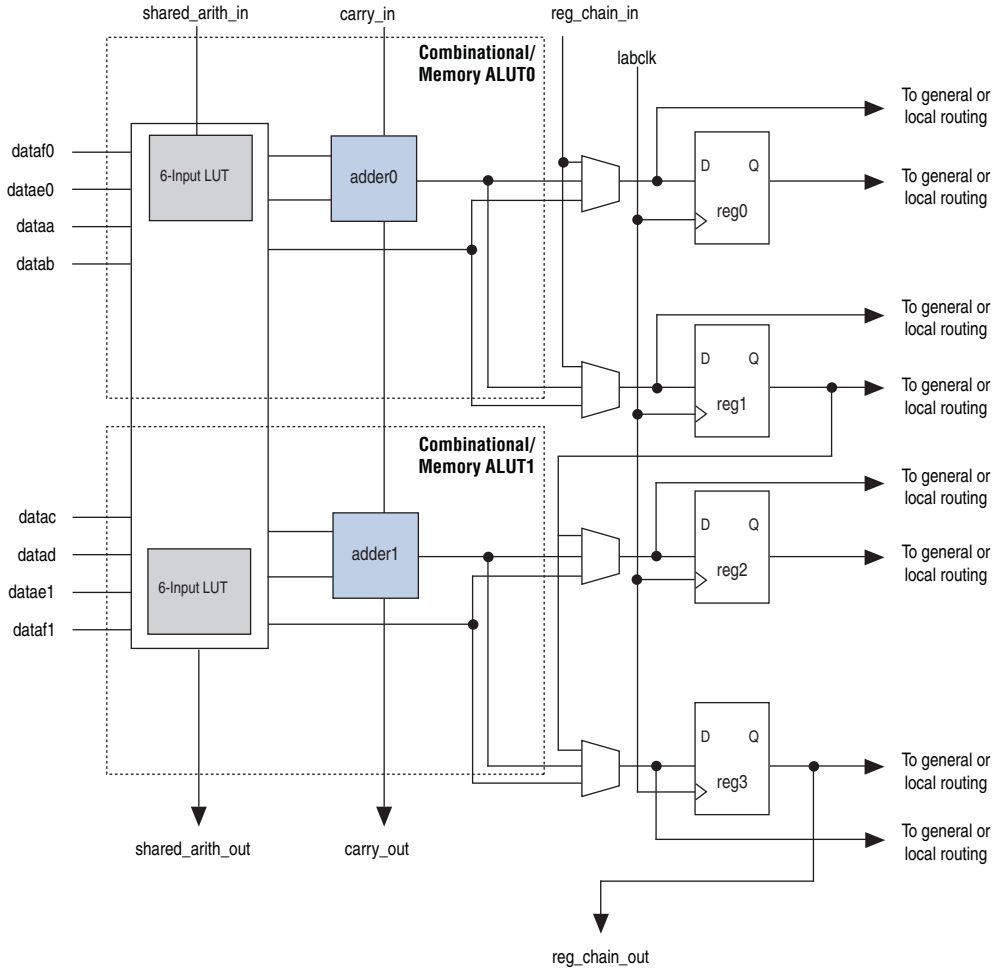
You can use a quarter of the available LABs in Arria V devices as a memory LAB (MLAB). The MLAB supports a maximum of 640 bits of simple dual-port SRAM.

Adaptive Logic Modules

Each ALM has general routing outputs and register chain outputs. For interconnects, there are two dedicated paths between ALMs—Carry chain and Shared Arithmetic chain. Arria V devices include an enhanced interconnect structure in LABs for routing shared arithmetic chains and carry chains for efficient arithmetic functions.

The ALM is the basic building block of logic in the Arria V device architecture, providing advanced features with efficient logic utilization. One ALM can implement any function of up to 6-input and certain 7-input functions. Each ALM drives all types of interconnects—local, row, column, carry chain, shared arithmetic chain, and direct link interconnects. Figure 1–1 shows a high-level block diagram of the Arria V ALM.

Figure 1–1. High-Level Block Diagram of the Arria V ALM



Register packing improves device utilization by allowing unrelated register and combinational logic to be packed into a single ALM. Another mechanism to improve fitting is to allow the register output to feed back into the look-up table (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 V ALM operates in any of the following modes:

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

Normal Mode

Normal mode is suitable for general logic applications and combinational functions.

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

Extended LUT Mode

Use extended LUT mode to implement a specific set of 7-input functions. The set must be a 2-to-1 multiplexer fed by two arbitrary 5-input functions sharing four inputs.

Arithmetic Mode

Arithmetic mode is ideal for implementing adders, counters, accumulators, wide parity functions, and comparators.

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 pair in half the ALM. The Quartus II software automatically places any registers that are not used by the counter into other LABs.

Shared Arithmetic Mode

A shared arithmetic chain can significantly improve the performance of an adder tree by reducing the number of summation stages required to implement the adder tree.

Memory Blocks

The embedded memory of an Arria V device consists of the following blocks:

- M10K blocks—dedicated memory resources.
- MLABs—configured from dual-purpose blocks that you can also configure as regular logic array blocks (LABs).

Each MLAB block is made up of ten adaptive logic modules (ALMs). You can configure each ALM in an MLAB as a 32 x 2 block, resulting in a 32 x 20 simple dual-port SRAM block in a single MLAB.

Use the information in this section in conjunction with the *Memory Blocks in Arria V Devices* chapter.

Parity Bit Support

The M10K memory supports one parity bit for each 4-data bits if the data width is 5, 10, 20, or 40. The parity bits for inputs and outputs are bits 4, 9, 14, 19, 24, 29, 34, and 39. Parity function is not performed on these bits. These bits are skipped during read or write operations with non-parity data widths.

In MLABs, the ninth bit associated with each byte can store a parity bit or serve as an additional data bit. Parity function is not performed on the ninth bit.

Byte Enable Support

All of the embedded memory blocks support byte enable controls:

- The byte enable controls mask the input data so that only specific bytes of data are written. The unwritten bytes retain the values written previously.
- The write enable (`wren`) signal, together with the byte enable (`byteena`) signal, control the write operations on the RAM blocks. By default, the `byteena` signal is high (enabled) and only the `wren` signal controls the writing.
- The byte enable registers do not have a `clear` port.
- If you are using parity bits, on the M10K blocks, the byte enable function controls 8 data bits and 2 parity bits; on the MLABs, the byte enable function controls all 10 bits in the widest mode.
- The MSB and LSB of the `byteena` signal correspond to the MSB and LSB of the data bus, respectively.
- The byte enables are active high.

Table 1–1 lists the `byteena` controls in the x20 data widths.

Table 1–1. byteena Controls in x20 Data Width

byteena[1:0]	Data Bits Written	
11 (default)	[19:10]	[9:0]
10	[19:10]	—
01	—	[9:0]

Table 1–2 lists the byteena controls in the x40 data widths.

Table 1–2. byteena Controls in x40 Data Width

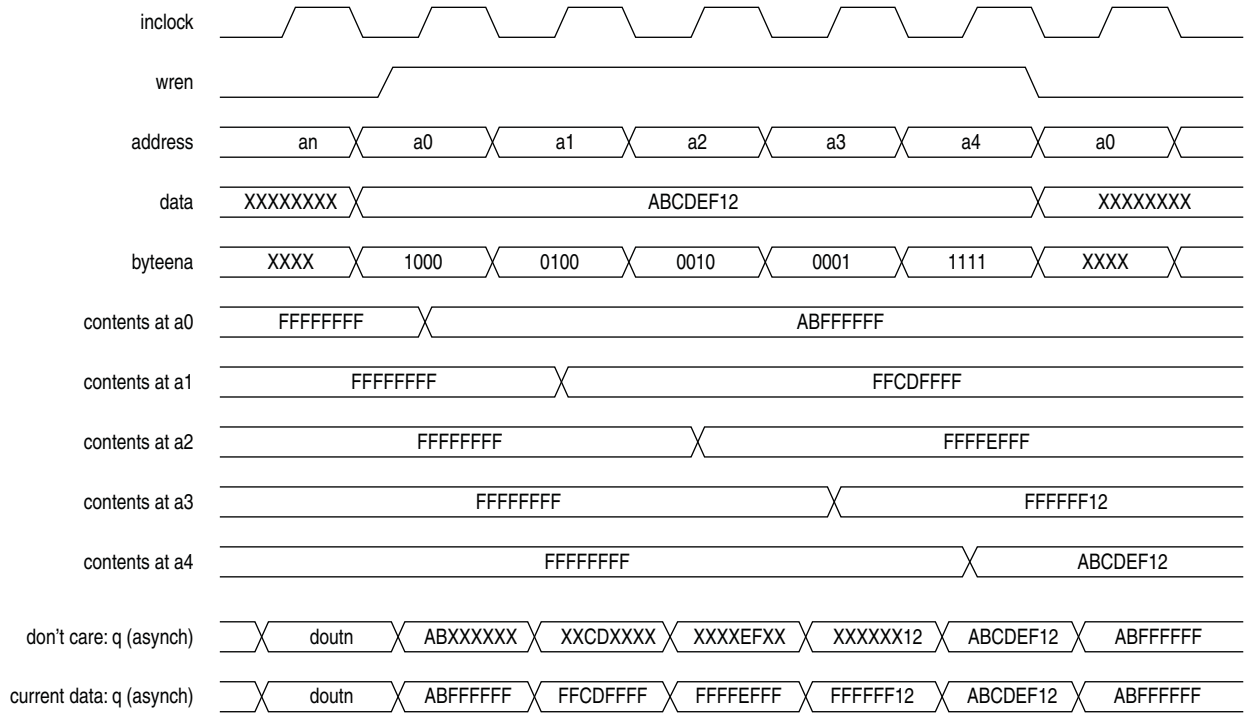
byteena[3:0]	Data Bits Written			
1111 (default)	[39:30]	[29:20]	[19:10]	[9:0]
1000	[39:30]	—	—	—
0100	—	[29:20]	—	—
0010	—	—	[19:10]	—
0001	—	—	—	[9:0]

In MLABs, when a byte-enable bit is deasserted during a write cycle, the corresponding data byte output appears as either a "don't care" value or the current data at that location. You can control the output value for the masked byte in MLABs using the Quartus II software.

In M10K blocks, the corresponding masked data byte output appears as a "don't care" value.

Figure 1–2 shows how the wren and byteena signals control the operations of the RAM blocks.

Figure 1–2. Byte Enable Functional Waveform *(1)*



Note to Table 1–2:

(1) For the M10K blocks, the write-masked data byte output appears as a "don't care" value because the "current data" value is not supported.

Design Considerations

To ensure the success of your designs, take into consideration the following guidelines when you are designing with the memory blocks:

Selecting Embedded Memory Blocks

Based on the speed and size constraints placed on your design, the Quartus II software automatically partitions the user-defined memory into the embedded memory blocks. For example, the Quartus II software may spread out the memory across multiple available memory blocks to increase the performance of the design. Use the RAM megafunction in the MegaWizard™ Plug-In Manager to manually assign the memory to a specific block size.

For the MLABs, you can implement single-port SRAM through emulation using the Quartus II software. Emulation results in minimal additional use of logic resources. Because of the dual-purpose architecture of the MLAB, only data input registers and output registers are available in the block. The MLABs gain read address registers from the ALMs. However, the write address and read data registers are internal to the MLABs.

Conflict Resolution

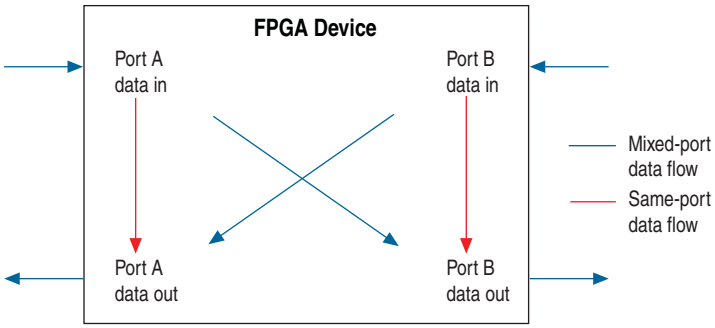
If you use the memory blocks in true dual-port mode, you can perform two write operations to the same memory location (address). Because conflict resolution circuitry is not available in the memory blocks, you must implement the conflict resolution logic external to the memory block to avoid unknown data from being written to the address.

Read-During-Write Behavior

You can customize the read-during-write behavior of the Arria V embedded memory blocks to fit your design requirements. There are two types of read-during-write operations—same port and mixed port.

Figure 1–3 shows the difference between the two types of read-during-write operations.

Figure 1–3. Read-During-Write Data Flow for Arria V Devices



Same-Port Read-During-Write Mode

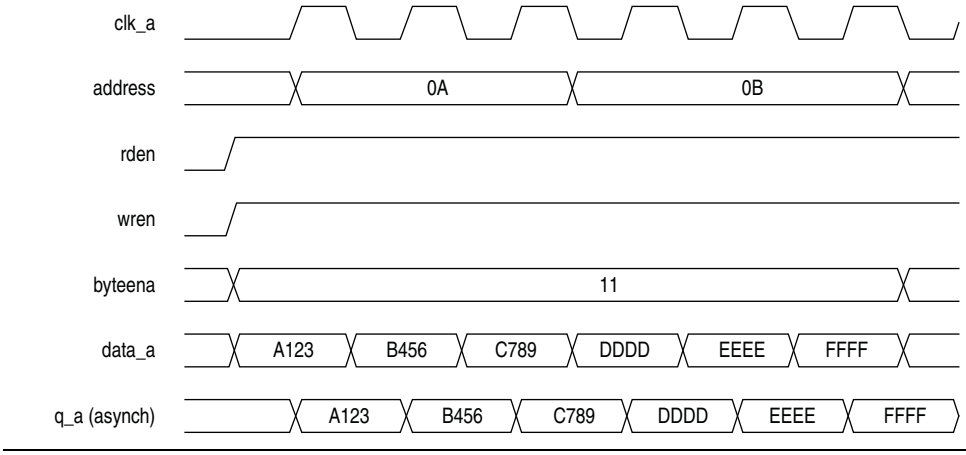
The same-port read-during-write mode applies to either a single-port RAM or the same port of a true dual-port RAM. Table 1–3 lists the available output modes if you select the M10K or MLAB in the same-port read-during-write mode.

Table 1–3. Output Modes for M10K or MLAB in Same-Port Read-During-Write Mode

Output Mode	Memory Type	Description
"new data" (flow-through)	M10K	The new data is available on the rising edge of the same clock cycle on which the new data is written.
"don't care"	M10K, MLAB	The RAM outputs "don't care" values for a read-during-write operation.

Figure 1–4 shows sample functional waveforms of same-port read-during-write behavior in new data mode.

Figure 1–4. Same-Port Read-During-Write: New Data Mode



Mixed-Port Read-During-Write Mode

The mixed-port read-during-write mode applies to RAM, in simple or true dual-port mode, where two ports perform read and write operations on the same memory address using the same clock—one port reading from the address, and the other port writing to it. Table 1–4 lists the available output modes in the mixed-port read-during-write mode.

Table 1–4. Output Modes for RAM in Mixed-Port Read-During-Write Mode

Output Mode	Memory Type	Description
"new data"	MLAB	A read-during-write operation to different ports causes the MLAB registered output to reflect the "new data" on the next rising edge after the data is written to the MLAB memory. This mode is available only if the output is registered.
"old data"	M10K, MLAB	A read-during-write operation to different ports causes the RAM output to reflect the "old data" value at the particular address. For MLAB, this mode is available only if the output is registered.
"don't care"	M10K, MLAB	The RAM outputs "don't care" or "unknown" value. For MLAB, the Quartus II software does not analyze the timing between write and read operations. To prevent metastability issue at the MLAB output, never write and read the same address at the same time. For M10K memory, the Quartus II software analyzes the timing between write and read operations.
"constrained don't care"	MLAB	The RAM outputs "don't care" or "unknown" value. The Quartus II software analyzes the timing between write and read operations in the MLAB.

Figure 1–5 shows a sample functional waveform of mixed-port read-during-write behavior for the "new data" mode.

Figure 1–5. Mixed-Port Read-During-Write: New Data Mode

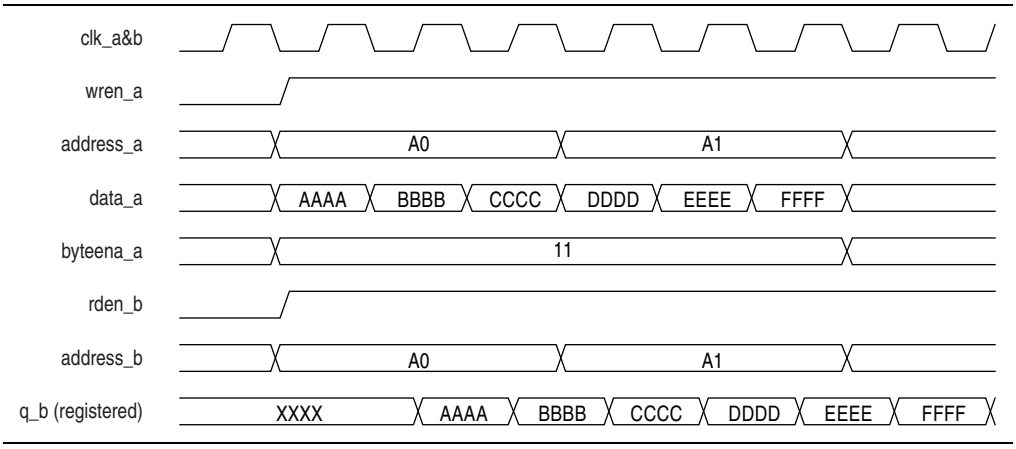


Figure 1–6 shows a sample functional waveform of mixed-port read-during-write behavior for the "old data" mode.

Figure 1–6. Mixed-Port Read-During-Write: Old Data Mode

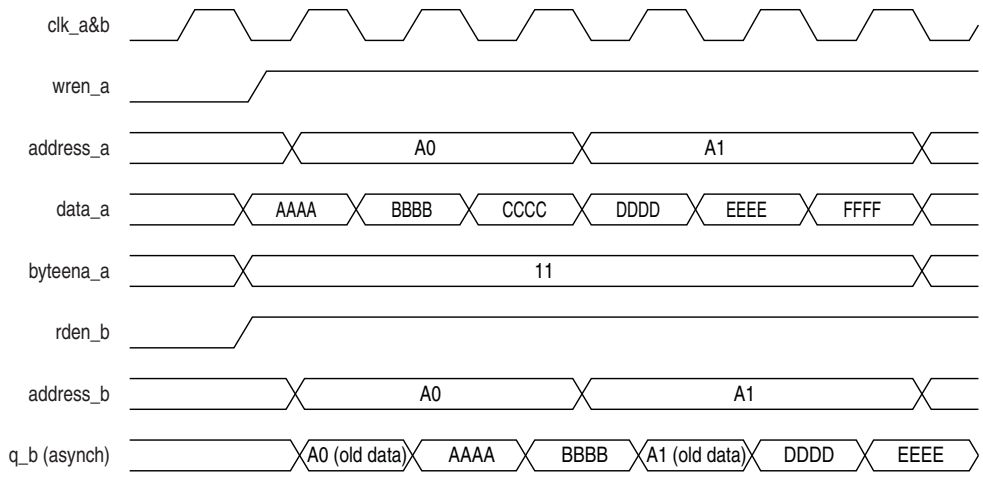
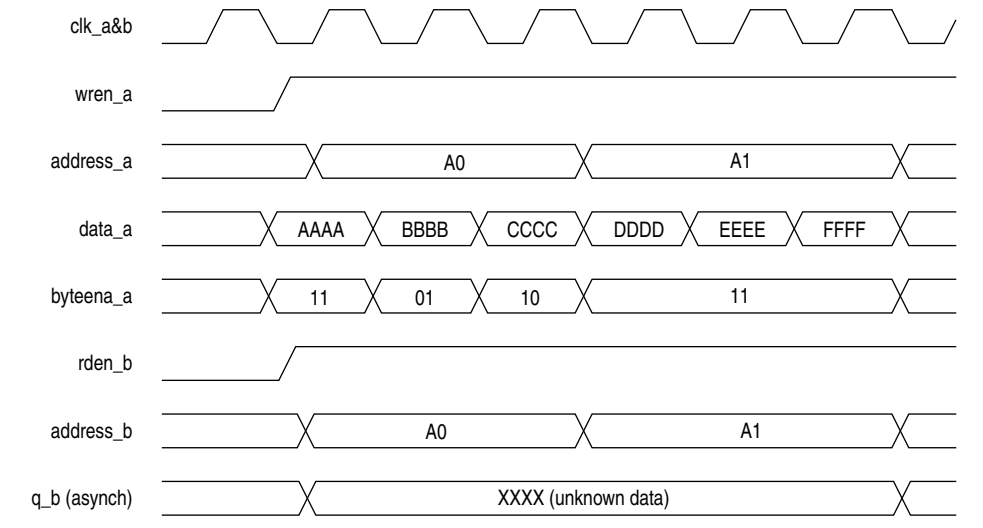


Figure 1–7 shows a sample functional waveform of mixed-port read-during-write behavior for the "don't care" or "constrained don't care" mode.

Figure 1–7. Mixed-Port Read-During-Write: Don't Care or Constrained Don't Care Mode



The mixed-port read-during-write operation is not supported if you use two different clocks in a dual-port RAM and the output value during the operation is "unknown."

For more information about the RAM megafunction, which controls the read-during-write behavior, refer to the *Internal Memory (RAM and ROM) User Guide*.

Power-Up Conditions and Memory Initialization

If you are designing logic that evaluates the initial power-up values of the MLAB memory block, consider the power-up condition of the different memory types, as listed in Table 1–5.

Table 1–5. Initial Power-Up Values of M10K and MLAB Blocks

Memory Type	Output Registers	Power Up Value
M10K	Used	Zero (cleared)
	Bypassed	Zero (cleared)
MLAB	Used	Zero (cleared)
	Bypassed	Read memory contents

By default, the Quartus II software initializes the RAM cells in Arria V devices to zero unless you specify a **.mif**.

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

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

Power Management

The clock-enables of the Arria V memory block allow you to control the clocking of each memory block to reduce AC power consumption:

- Use the read-enable signal to ensure that read operations occur only when required. 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 during the period when no memory operations occur.
- Use the Quartus II software to automatically place any unused memory blocks in low-power mode to reduce static power.

Variable-Precision DSP Blocks

This section describes the basic information of variable-precision digital signal processing (DSP) blocks in Arria V devices.

Use the information in this section in conjunction with the *Variable-Precision DSP Blocks in Arria V Devices* chapter.

Independent Multiplier Mode

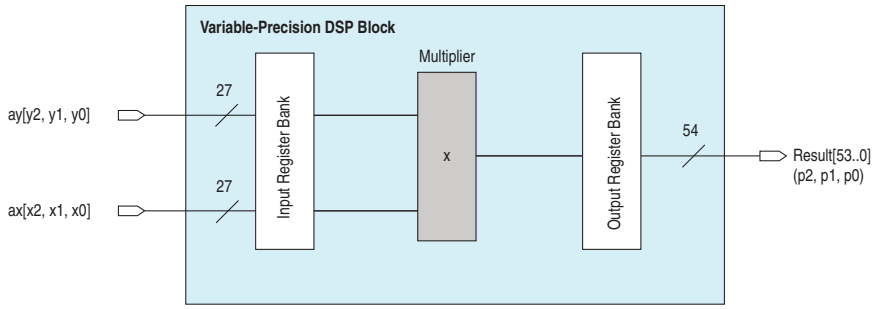
In independent input and output multiplier mode, the variable-precision DSP blocks perform individual multiplication operations for general purpose multipliers.

9 x 9, 18 x 18, 18 x 19, 18 x 25, 20 x 24, and 27 x 27 Multipliers

You can configure each variable-precision DSP block multiplier for 9 x 9, 18 (signed or unsigned) x 18 (unsigned), 18 (signed or unsigned) x 19 (signed), 18 x 25, 20 x 24, or 27 x 27 multiplication. A variable-precision DSP block supports up to three individual 9 x 9 multipliers, two individual 18 x 19 multipliers, two individual 18 x 18 multipliers, one individual 18 x 25 multiplier, one individual 20 x 24 multiplier, or one individual 27 x 27 multiplier.

Figure 1–8 through Figure 1–12 on page 1–13 show the variable-precision DSP block in independent multiplier operation mode.

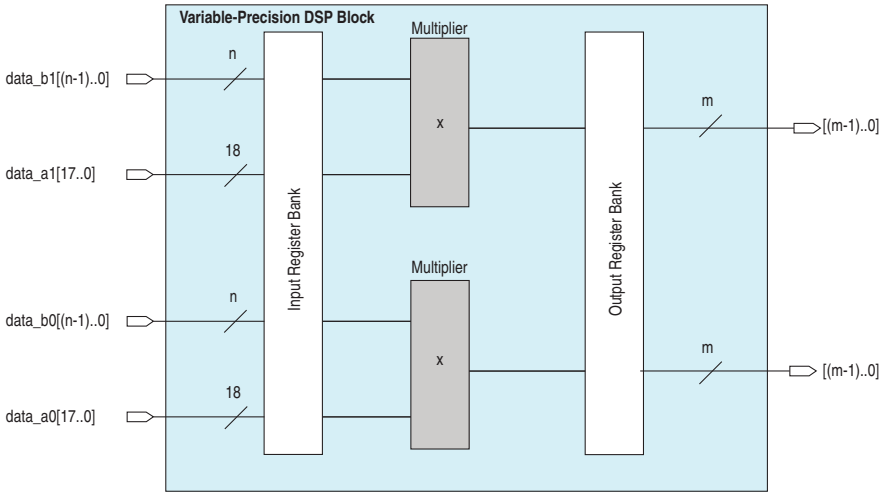
Figure 1–8. Three 9 x 9 Independent Multiplier Mode with One Variable-Precision DSP Block *(1)*



Note to Figure 1–8:

(1) Three pairs of data are packed into the ax and ay ports; `result` contains three 18-bit products.

Figure 1–9. Two 18 x 18 or One 18 x 19 Independent Multiplier Mode with One Variable-Precision DSP Block *(1), (2)*

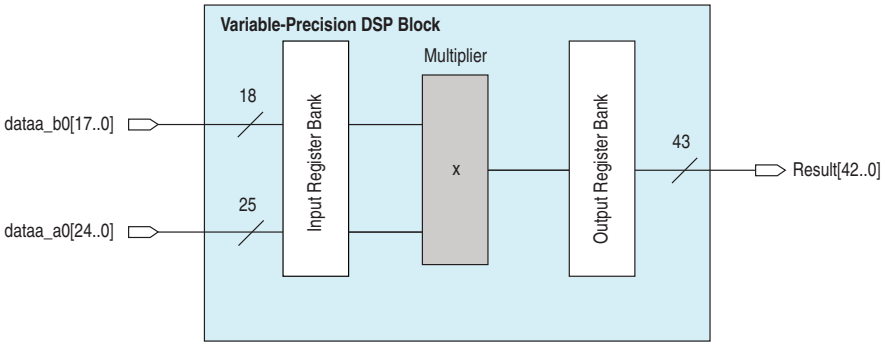


Notes to Figure 1–9:

(1) n = 19 and m = 37 for 18 x 19 mode.

(2) n = 18 and m = 36 for 18 x 18 mode.

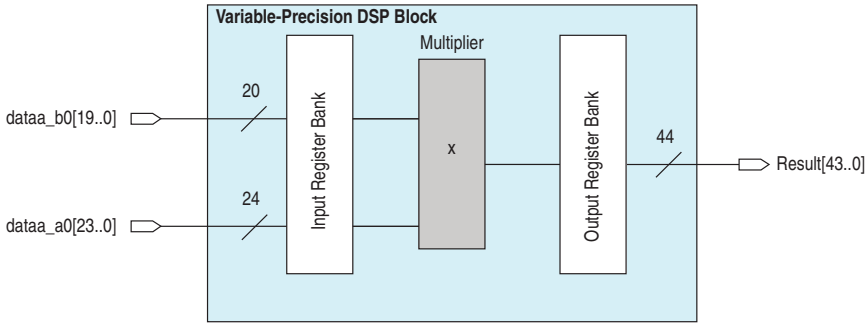
Figure 1–10. One 18 x 25 Independent Multiplier Mode with One Variable-Precision DSP Block *(1)*



Note to Figure 1–10:

(1) The result can be up to 52 bits when combined with a chainout adder or accumulator.

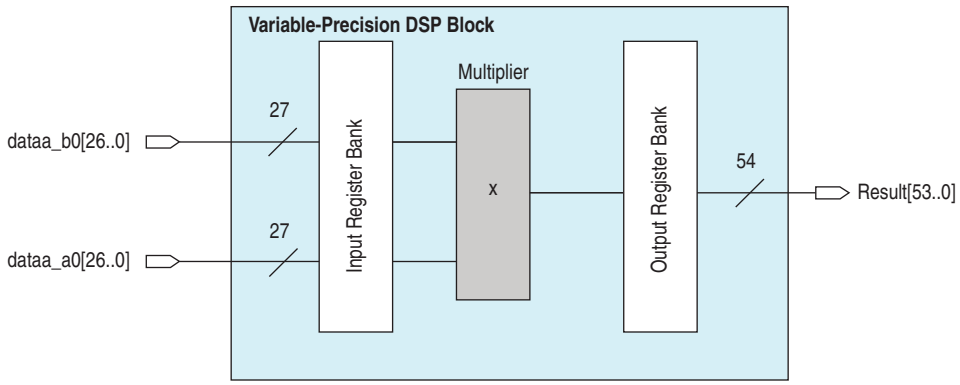
Figure 1–11. One 20 x 24 Independent Multiplier Mode with One Variable-Precision DSP Block *(1)*



Note to Figure 1–11:

(1) The result can be up to 52 bits when combined with a chainout adder or accumulator.

Figure 1–12. One 27 x 27 Independent Multiplier Mode with One Variable-Precision DSP Block *(1)*



Note to Figure 1–12:

(1) The result can be up to 64 bits when combined with a chainout adder or accumulator.

Independent Complex Multiplier Mode

The Arria V variable-precision DSP block provides the means for complex multiplication and supports an 18 x 19 complex multiplier. Equation 1–1 shows a sample complex multiplication equation that you can write.

Equation 1–1. Complex Multiplication Equation

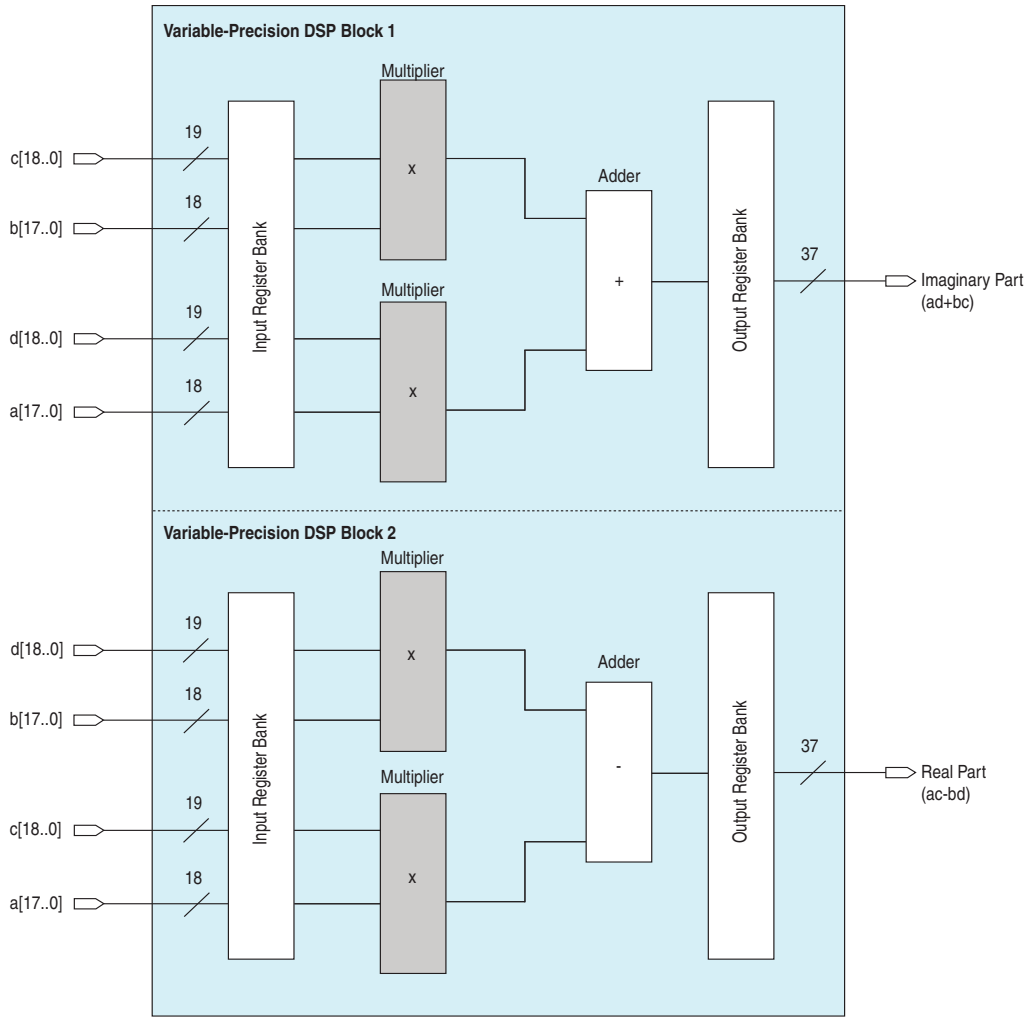
$$(a + jb) \times (c + jd) = [(a \times c) - (b \times d)] + j[(a \times d) + (b \times c)]$$

18 x 19 Complex Multiplier

Two variable-precision DSP blocks are required to perform this multiplication mode. The imaginary part [(a x d) + (b x c)] is implemented in the first variable-precision DSP block, while the real part [(a x c) - (b x d)] is implemented in the second variable-precision DSP block.

Figure 1–13 shows an 18 x 19 complex multiplication.

Figure 1–13. An 18 x 19 Complex Multiplier with Two Variable-Precision DSP Blocks



Multiplier Adder Sum Mode

Arria V devices support two-multiplier adder sum mode and four-multiplier adder sum mode. For a two-multiplier adder configuration, the variable-precision DSP blocks support 18 x 19 multipliers and 27 x 27 multipliers.

To implement a 27 x 27 multiplier adder sum mode, two variable-precision DSP blocks are required. Arria V devices support one sum of four 18 x 19 multipliers with two variable-precision DSP blocks. Figure 1–14, Figure 1–15, and Figure 1–16 show the variable-precision DSP blocks in multiplier adder sum mode.

Figure 1–14. One Sum of Two 18 x 19 Multipliers with One Variable-Precision DSP Block

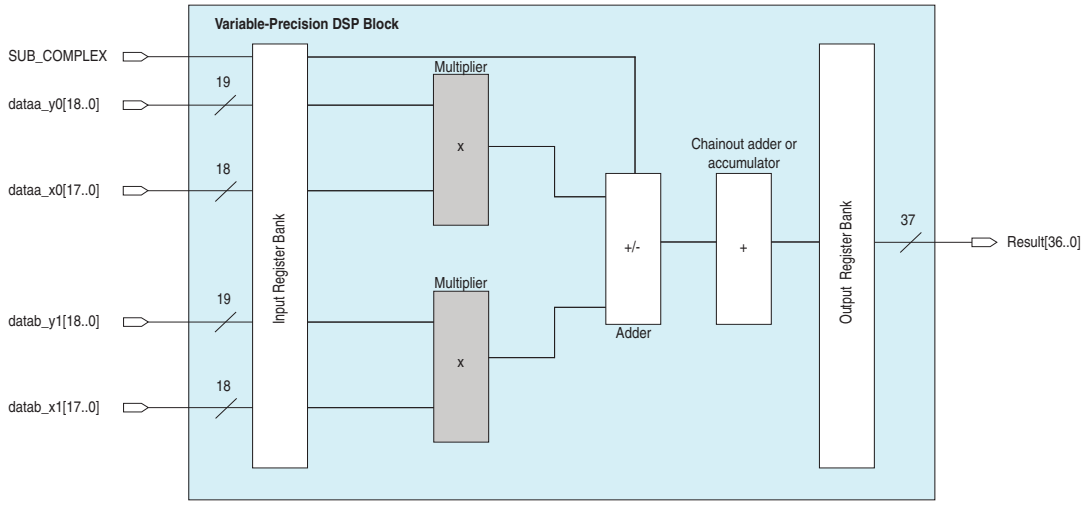


Figure 1–15. One Sum of Two 27 x 27 Multipliers with Two Variable-Precision DSP Blocks

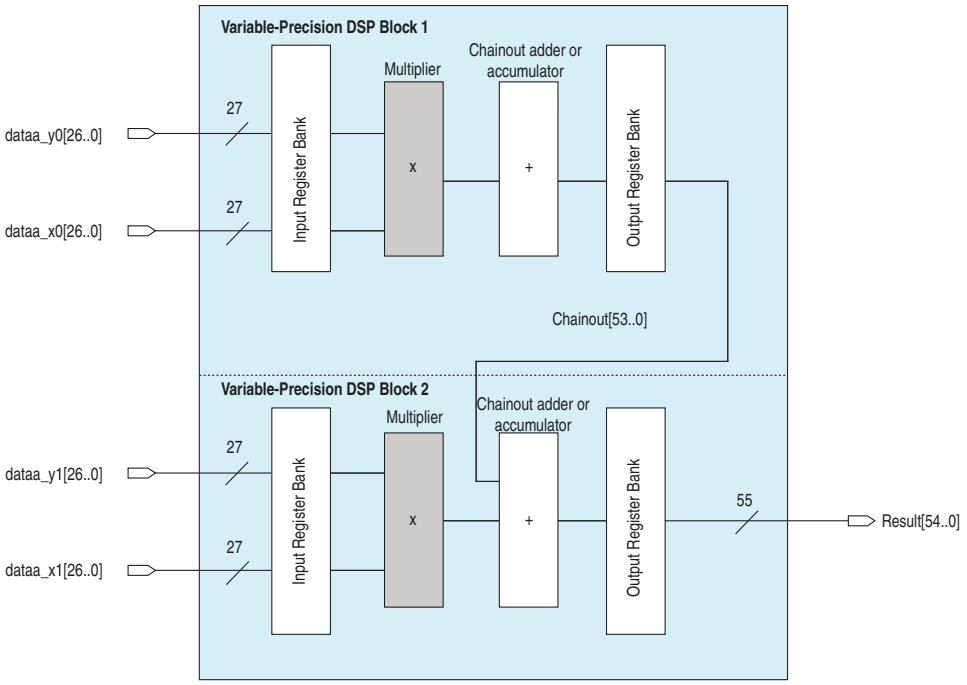
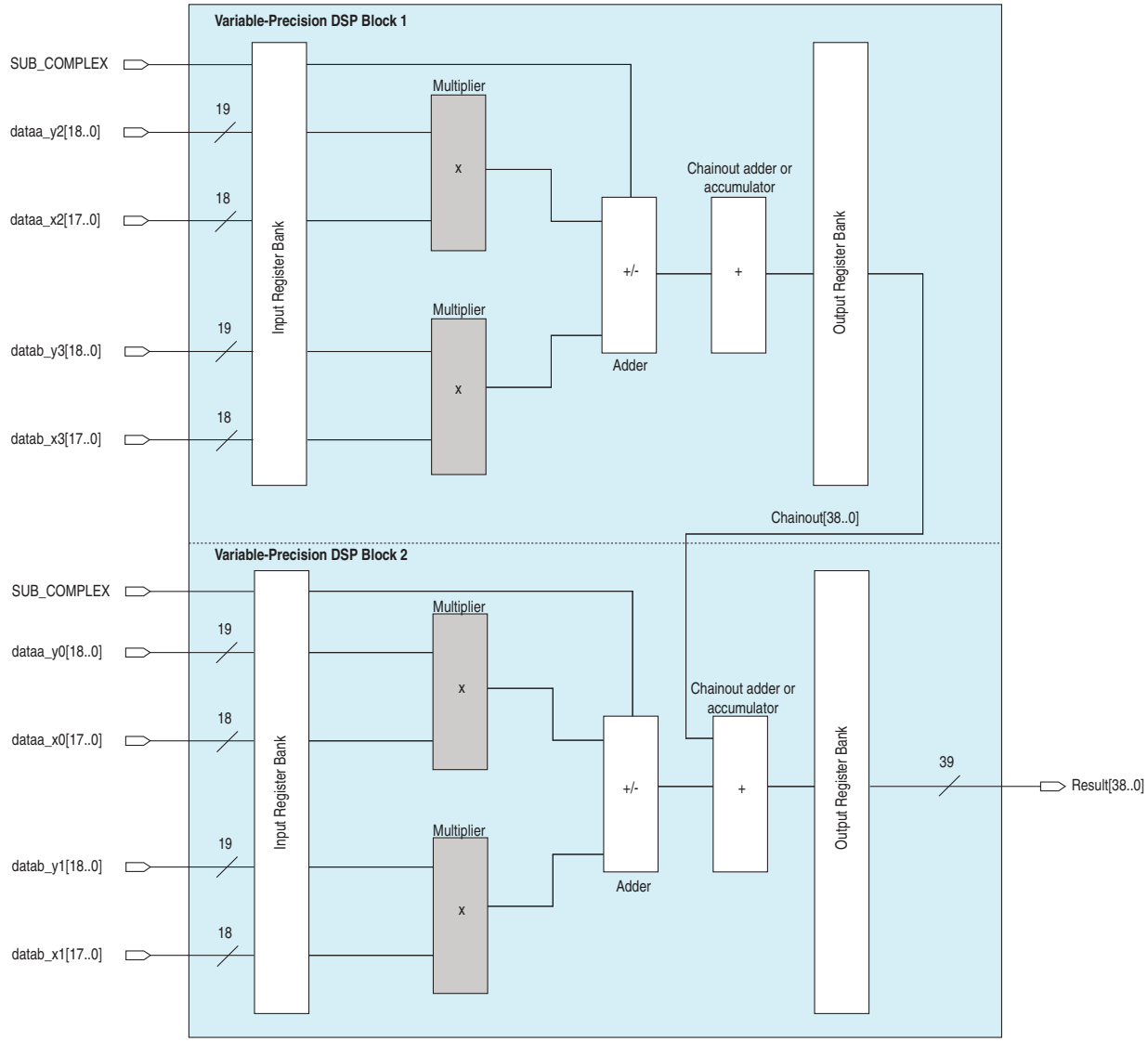


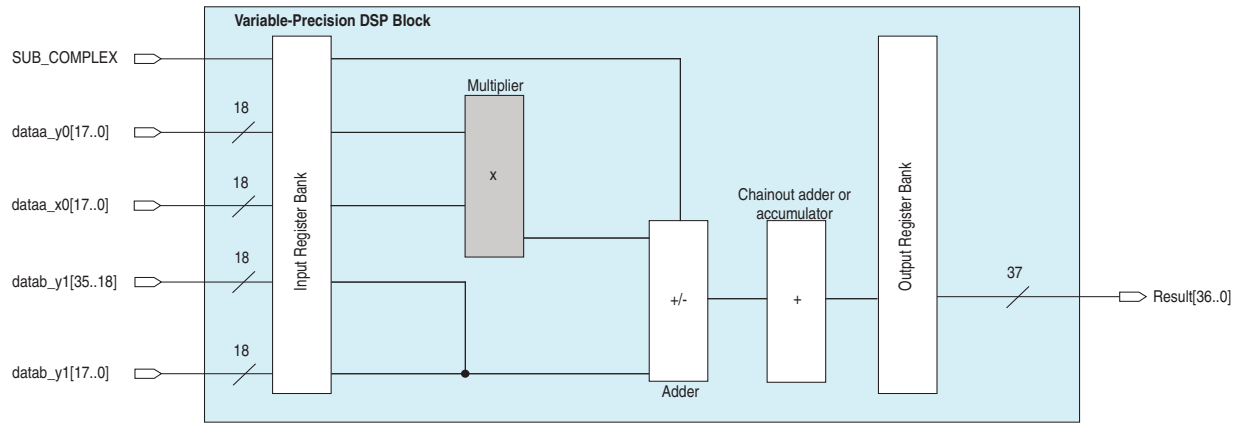
Figure 1–16. One Sum of Four 18 x 19 Multipliers with Two Variable-Precision DSP Blocks



18 x 18 Multiplication Summed with 36-Bit Input Mode

Arria V variable-precision DSP blocks support one 18 x 18 multiplication summed to a 36-bit input. Use the upper multiplier to provide the input for an 18 x 18 multiplication, while the bottom multiplier is bypassed. The `datab_y1[17..0]` and `datab_y1[35..18]` signals are concatenated to produce a 36-bit input. Figure 1–17 shows the 18 x 18 multiplication summed with the 36-bit input mode in a variable-precision DSP block.

Figure 1–17. One 18 x 18 Multiplication Summed with 36-Bit Input Mode



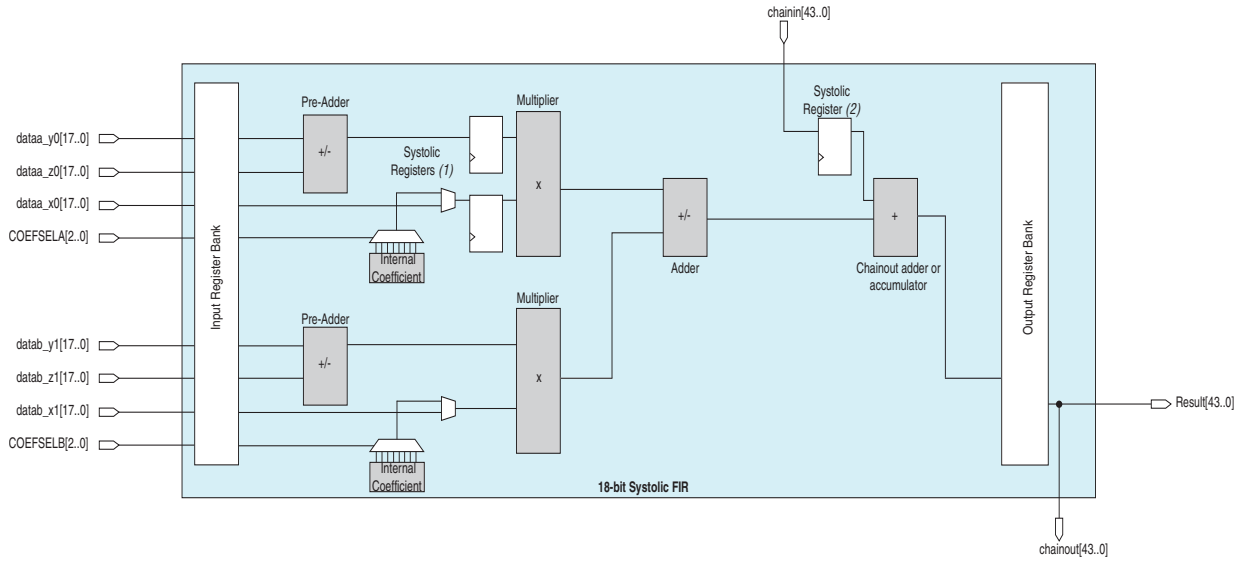
Systolic FIR Mode

Arria V variable-precision DSP blocks support 18-bit and 27-bit systolic finite impulse response (FIR) structures. In systolic FIR mode, the input of the multiplier can come from four different sets of sources:

- two dynamic inputs
- one dynamic input and one coefficient input
- one coefficient input and one pre-adder output
- one dynamic input and one pre-adder output

Figure 1–18 shows 18-bit systolic FIR mode. In this mode, the adders are configured as dual 44-bit adders, thereby giving 8 bits of overhead when using an 18-bit operation (36-bit products). This allows a total of 256 multiplier products.

Figure 1–18. 18-Bit Systolic FIR Mode

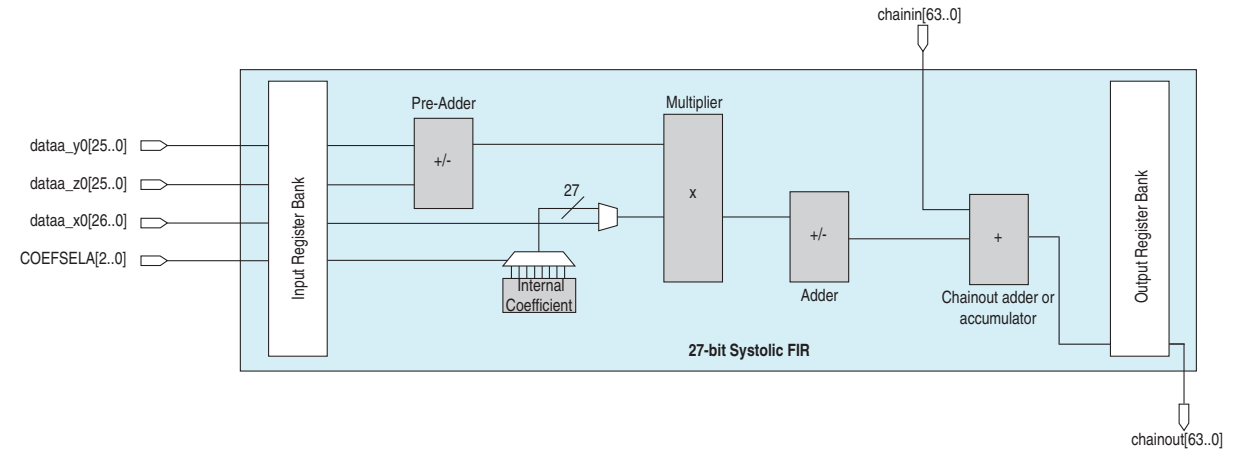


Notes to Figure 1–18:

(1) The systolic registers have the same clock source as the multiplier inputs.
(2) The systolic registers have the same clock source as the output register bank.

Figure 1–19 shows 27-bit systolic FIR mode. This mode allows the implementation of one stage systolic filter per DSP block. The chainout adder or accumulator is configured for a 64-bit operation, providing 10 bits of overhead when using a 27-bit data (54-bit products). This allows a total of 1,024 multiplier products.

Figure 1–19. 27-Bit Systolic FIR Mode



Clock Networks and PLLs

This section describes the basic information of the clock networks and PLLs in Arria V devices.

Use the information in this section in conjunction with the *Clock Networks and PLLs in Arria V Devices* chapter.

Clock Regions

This section describes how you can form different types of clock regions in Arria V devices and the best options for each type.

Arria V devices provide the following types of clock regions:

- "Entire Device Clock Region"
- "Regional Clock Region"
- "Dual-Regional Clock Region"

Entire Device Clock Region

To form the entire device clock region, a source drives a GCLK network that can be routed through the entire device. The source is not necessarily a clock signal. This clock region has the maximum insertion delay when compared with other clock regions, but allows the signal to reach every destination in the device. It is a good option for routing global reset and clear signals or routing clocks throughout the device.

Regional Clock Region

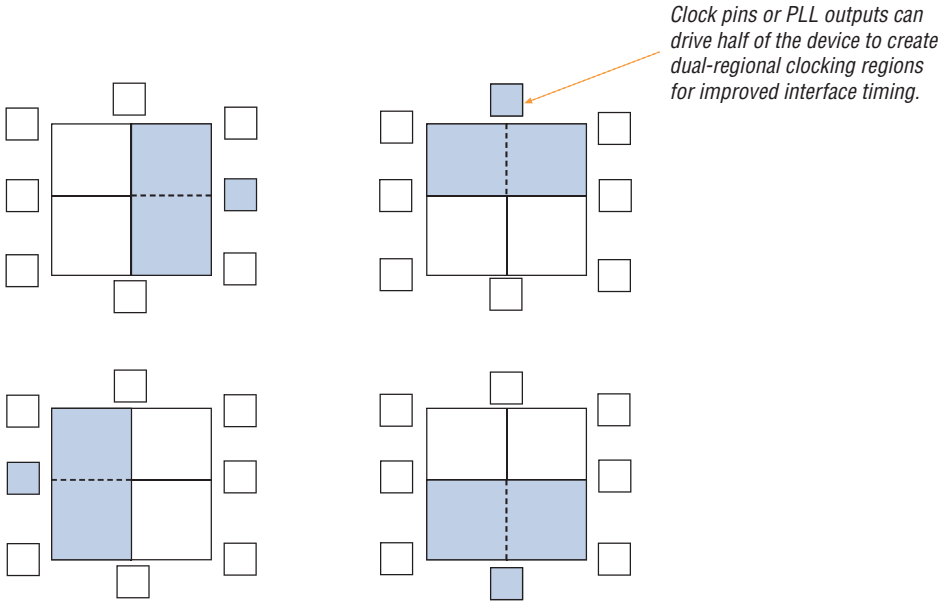
To form a regional clock region, a source drives a signal RCLK network that you can route throughout one quadrant of the device. This clock region provides the lowest skew in a quadrant. It is a good option if all the destinations are in a single quadrant.

Dual-Regional Clock Region

To form a dual-regional clock region, a single source (a clock pin or PLL output) generates a dual-regional clock by driving two RCLK networks (one from each quadrant). This technique allows destinations across two adjacent device quadrants to use the same low-skew clock. The routing of this signal on an entire side has approximately the same delay as a RCLK region. Internal logic can also drive a dual-regional clock network. Corner PLL outputs span only one quadrant because the outputs cannot generate a dual-regional clock network.

Figure 1–20 shows the dual-regional clock region.

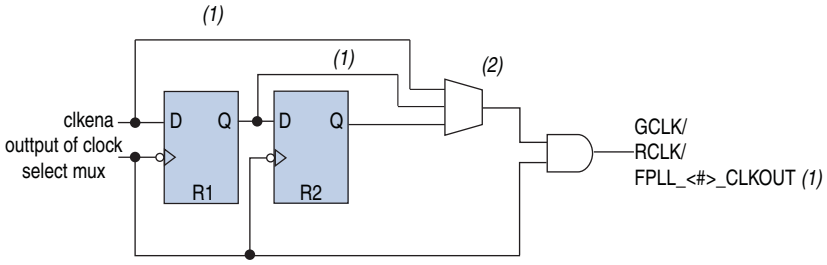
Figure 1–20. Dual-Regional Clock Region for Arria V Devices



Clock Enable Signals

Figure 1–21 shows how the clock enable and disable circuit of the clock control block is implemented in Arria V devices.

Figure 1–21. clkena Implementation



Notes to Figure 1–21:

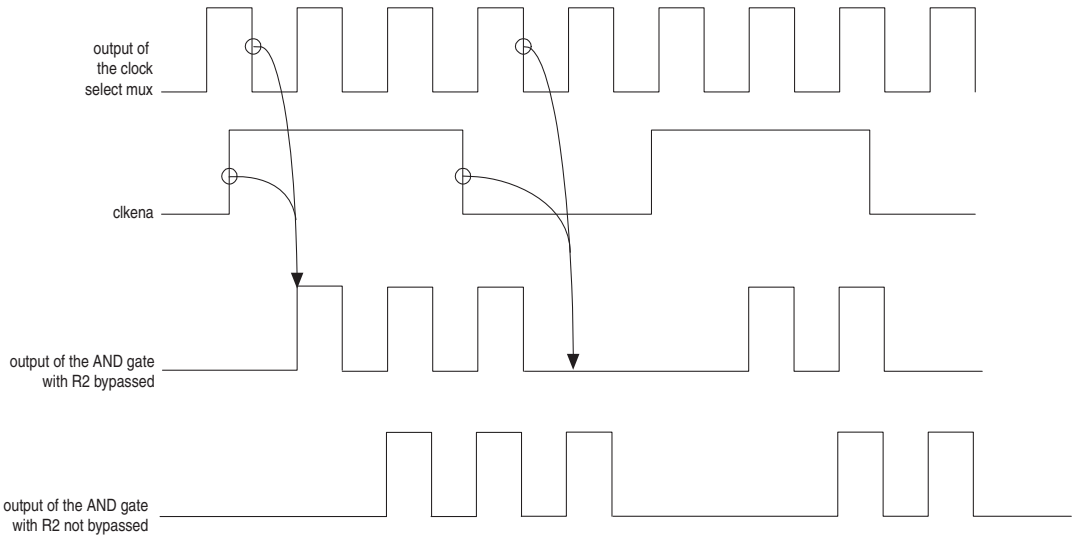
(1) The R1 and R2 bypass paths are not available for the PLL external clock outputs.

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

In Arria V 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 you are not using a PLL. You can also use the clkena signals to control the dedicated external clocks from the PLLs. Figure 1–22 shows a waveform example for a clock output enable. The clkena signal is synchronous to the falling edge of the clock output.

Arria V devices have an additional metastability register that aids in asynchronous enable and disable of the GCLK and RCLK networks. You can optionally bypass this register in the Quartus II software.

Figure 1–22. clkena Signals *(1)*



Note to Figure 1–22:

(1) Use the clkena signals to enable or disable the GCLK and RCLK networks or the FPLL_<#>_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 overshoot during resynchronization.

Clock Feedback Modes

Arria V PLLs support different clock feedback modes. Each mode allows clock multiplication and division, phase shifting, and programmable duty cycle.

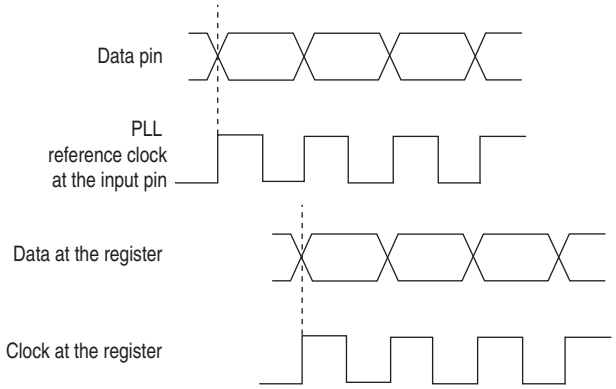
The input and output delays are fully compensated by a PLL only when using the dedicated clock input pins associated with a given PLL as the clock source. When a RCLK or GCLK network drives the PLL or the PLL is driven by a dedicated clock pin that is not associated with the PLL, the input and output delays may not be fully compensated in the Quartus II software. For example, when you configure a PLL in zero-delay buffer (ZDB) mode and the PLL input is driven by an associated dedicated clock input pin. In this configuration, a fully compensated clock path results in zero delay between the clock input and one of the clock outputs from the PLL. However, if the PLL input is fed by a non-dedicated input (using the GCLK network), the clock output may not be perfectly aligned with the input clock.

For a mapping of dedicated clock pins to their associated PLLs, refer to the "Arria V PLLs" section in the *Clock Networks and PLLs in Arria V PLLs* chapter.

Source Synchronous Mode

If the 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 1–23 shows a waveform example of the clock and data in this mode. Altera recommends source synchronous mode 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 1–23. Phase Relationship Between Clock and Data in Source Synchronous Mode



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

The Arria V PLL can compensate multiple pad-to-input-register paths, such as a data bus when it is set to use source-synchronous compensation mode. 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. To compensate for the clock delay properly, all of the input pins must be on the same side of the device. 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 make the "PLL Compensation" assignment, the Quartus II software automatically selects all of the pins driven by the compensated output of the PLL as the compensation target.

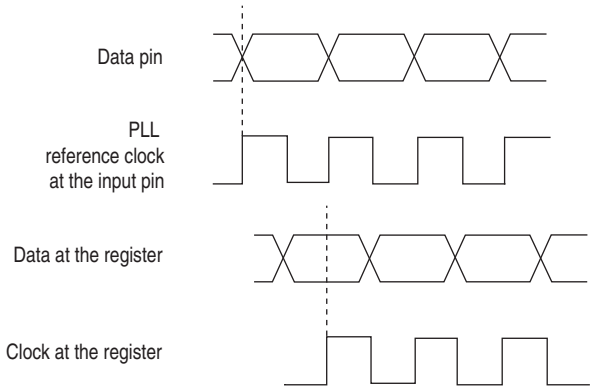
LVDS Compensation

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

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

Figure 1–24 shows a waveform example of the clock and data in LVDS mode.

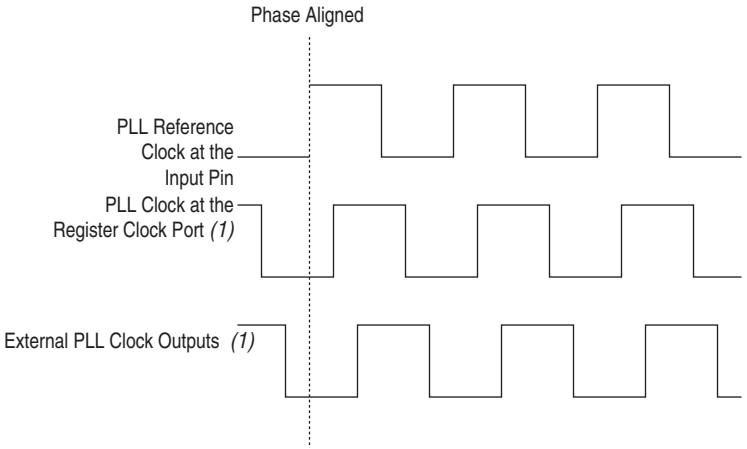
Figure 1–24. Phase Relationship Between the Clock and Data in LVDS Mode



Direct Compensation Mode

In direct compensation mode, the PLL does not compensate for any clock networks. This mode provides better jitter performance because the clock feedback into the PFD passes through less circuitry. Both the PLL internal- and external-clock outputs are phase-shifted with respect to the PLL clock input. Figure 1–25 shows a waveform example of the PLL clocks' phase relationship in direct compensation mode.

Figure 1–25. Phase Relationship Between the PLL Clocks in Direct Compensation Mode



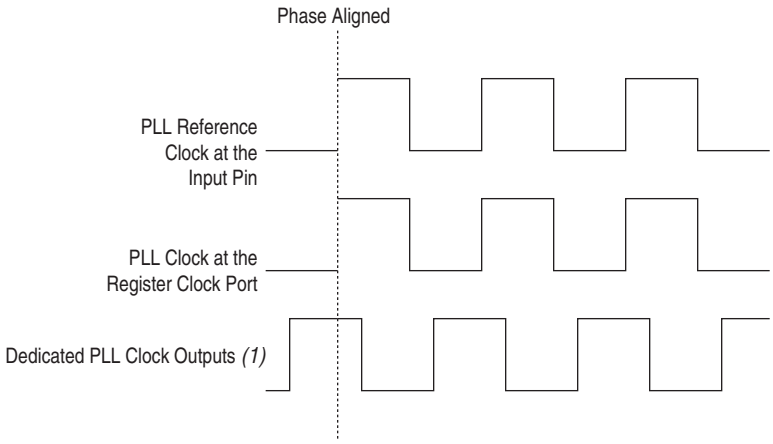
Note to Figure 1–25:

(1) The PLL clock outputs lag the PLL input clocks depending on routing 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 1–26 shows a waveform example of the PLL clocks' phase relationship in normal mode.

Figure 1–26. Phase Relationship Between the PLL Clocks in Normal Mode



Note to Figure 1–26:

(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. When using this mode, you must use the same I/O standard on the input clocks and clock outputs to guarantee clock alignment at the input and output pins. ZDB mode is supported on all Arria V PLLs.

To ensure phase alignment between the `clk` pin and the external clock output (`CLKOUT`) pin when you use Arria V PLLs in ZDB mode, along with single-ended I/O standards, instantiate a bidirectional I/O pin in the design to serve as the feedback path connecting the `fbout` and `fbin` ports of the PLL. 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.

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

Figure 1–27 shows ZDB mode in Arria V PLLs. When using ZDB mode, you cannot use differential I/O standards on the PLL clock input or output pins.

In Arria V devices, ZDB mode can support up to four single-ended clock outputs. For more information, refer to the "PLL External I/O Clock Pins" section in the *Clock Networks and PLLs in Arria V Devices* chapter.

Figure 1–27. ZDB Mode in Arria V PLLs

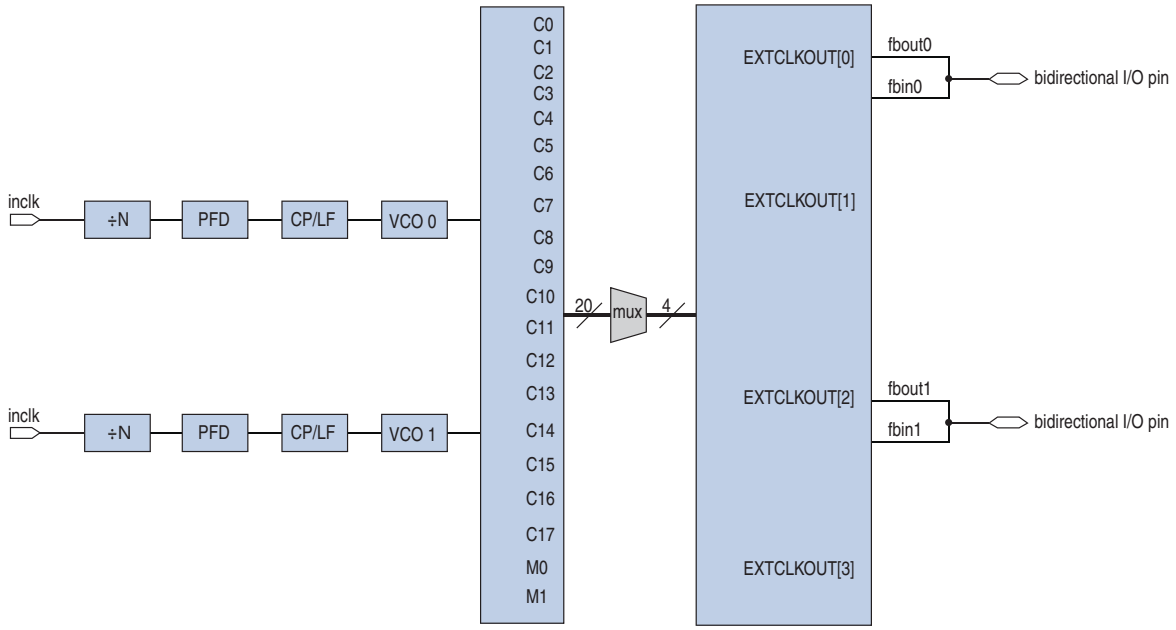
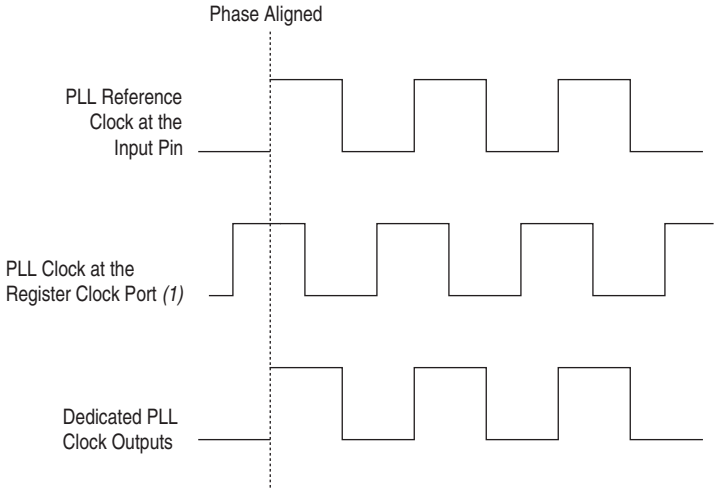


Figure 1–28 shows a waveform example of the PLL clocks' phase relationship in ZDB mode.

Figure 1–28. Phase Relationship Between the PLL Clocks in ZDB Mode



Note to Figure 1–28:

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

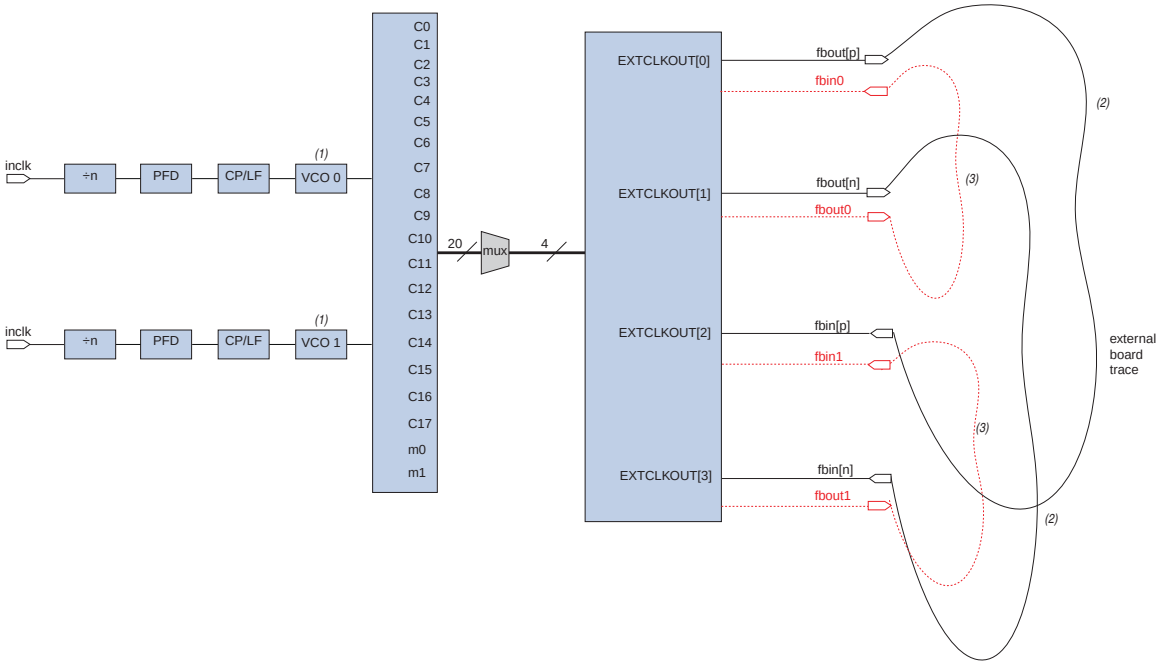
External Feedback Mode

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

When using EFB mode, you must use the same I/O standard on the input clock, feedback input, and clock outputs.

Figure 1–29 shows the EFB mode in Arria V devices.

Figure 1–29. EFB Mode in Arria V Devices *(1)*

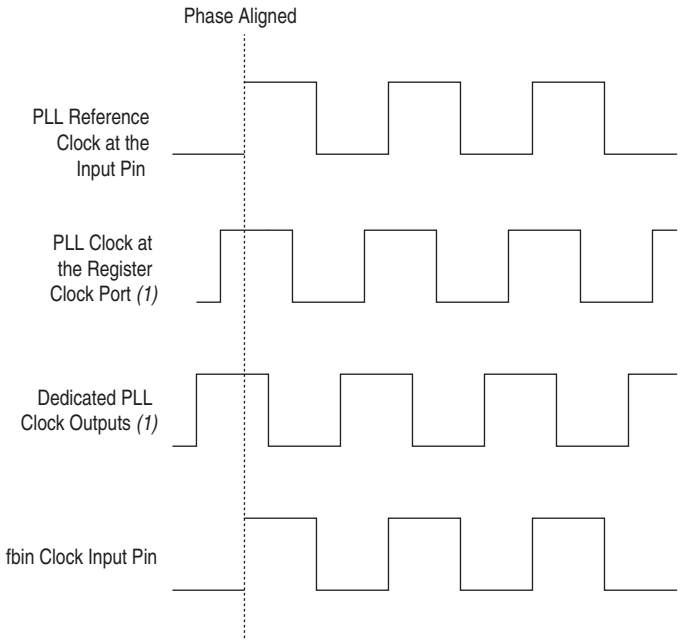


Notes to Figure 1–29:

(1) Only one of the two VCOs can support differential EFB mode at one time while you can use the other VCO for general purpose clocking.
(2) External board connection for one differential clock output and one differential feedback input for differential EFB support.
(3) External board connection for two single-ended clock outputs and two single-ended feedback inputs for single-ended EFB support.

Figure 1–30 shows a waveform example of the phase relationship between the PLL clocks in EFB mode.

Figure 1–30. Phase Relationship Between the PLL Clocks in EFB Mode *(1)*



Note to Figure 1–30:

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

In EFB mode, the external feedback input pin (`fbin`) is phase-aligned with the clock input pin. Aligning these clocks allows you to remove clock delay and skew between devices. This mode is supported on all Arria V PLLs, except on the corner fractional PLLs.

Clock Multiplication and Division

Each Arria V PLL provides clock synthesis for PLL output ports using the $K.M/(N \times \textit{post-scale counter})$ scaling factors. The input clock is divided by a pre-scale factor, `N`, and is then multiplied by the `K` and `M` feedback factors. The control loop drives the VCO to match f_{in} ($K.M/N$).

A post-scale counter, `K`, is inserted after the VCO. When you enable the VCO post-scale counter, the counter divides the VCO frequency by two. When the `K` counter is bypassed, the VCO frequency goes to the output port without being divided by two. Each output port has a unique post-scale counter that divides down the output from the `K` counter. 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 the 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 within the VCO range). Then the post-scale `C` counters scale down the VCO frequency for each output port.

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. The post-scale counters range from 1 to 512 with a 50% duty cycle setting. The high- and low-count values for each counter range from 1 to 256. The sum of the high- and low-count values chosen for a design selects the divide value for a given counter.

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

Programmable Duty Cycle

The programmable duty cycle allows PLLs to generate clock outputs with a variable duty cycle. This feature is supported on the PLL post-scale counters. The duty-cycle setting is achieved by a low and high time-count setting for the post-scale counters. To determine the duty cycle choices, the Quartus II software uses the frequency input and the required multiply or divide rate. The post-scale counter value determines the precision of the duty cycle. The precision is defined as 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 from 5% to 90%.

If the PLL is in external feedback mode, set the duty cycle for the counter driving the fbin pin to 50%. Combining the programmable duty cycle with programmable phase shift allows the generation of precise non-overlapping clocks.

I/O Features

This section describes the basic I/O capabilities in the Arria V device that allow you to work in compliance with current and emerging I/O standards and requirements.

Use the information in this section in conjunction with the *I/O Features in Arria V Devices* chapter.

I/O Element Features

The following sections describe the I/O element (IOE) features in Arria V devices.

Slew-Rate Control

The output buffer for each regular- and dual-function I/O pin contains a programmable output slew-rate control that you can configure for low-noise or high-speed performance:

- Fast slew rate—provides high-speed transitions for high-performance systems.
- Slow slew rate—reduces system noise and crosstalk but adds a nominal delay to the rising and falling edges.

You can specify the slew rate on a pin-by-pin basis because each I/O pin contains a slew-rate control.

Altera recommends that you perform IBIS or SPICE simulations to determine the best slew rate setting for your specific application.

I/O Delay

The following sections describe the programmable IOE delay and the programmable output buffer delay.

Programmable IOE Delay

The IOE includes programmable delays that you can activate to ensure zero hold times, minimize setup times, or increase clock-to-output times.

Each pin can have a different input delay from pin-to-input register or a delay from output register-to-output pin values. Use this feature to ensure that the signals within a bus have the same delay going into or out of the device.

This feature helps read and write timing margins because it minimizes the uncertainties between signals in the bus.

For more information about programmable IOE delay specifications, refer to the *Device Datasheet for Arria V Devices* chapter.

Programmable Output Buffer Delay

The device supports delay chains built inside the single-ended output buffer. There are four levels of output buffer delay settings. By default, there is no delay.

The delay chains can independently control the rising and falling edge delays of the output buffer.

This feature provides you with the following abilities:

- Adjust the output-buffer duty cycle.
- Compensate channel-to-channel skew.
- Reduce simultaneous switching output (SSO) noise by deliberately introducing channel-to-channel skew.
- Improve high-speed memory-interface timing margins.

For more information about programmable output buffer delay specifications, refer to the *Device Datasheet for Arria V Devices* chapter.

Open-Drain Output

Arria V devices provide an optional open-drain output—equivalent to an open collector output—for each I/O pin. When configured as an open drain, the logic value of the output is either high-Z or logic low. You require an external resistor to pull the signal to a logic high.

Bus-Hold

Each I/O pin provides an optional bus-hold feature. If you enable the bus-hold feature, you cannot use the programmable pull-up option. To configure the I/O pin for differential signals, disable the bus-hold feature.

The bus-hold circuitry uses a resistor with a nominal resistance (R_{BH}), approximately 7 kΩ, to weakly pull the signal level to the last-driven state of the pin. The bus-hold circuitry holds this pin state until the next input signal is present. Because of this, you do not require an external pull-up or pull-down resistor to hold a signal level when the bus is tri-stated.

For each I/O pin, you can individually specify that the bus-hold circuitry pulls non-driven pins away from the input threshold voltage—where noise can cause unintended high-frequency switching. To prevent over-driving signals, the bus-hold circuitry drives the voltage level of the I/O pin lower than the V_{CCIO} level.

The bus-hold circuit is active only after configuration. When the device enters user mode, the bus-hold circuit captures the value that is present on the pin by the end of the configuration.

Pull-Up Resistor

Each 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.

The device supports programmable weak pull-up resistors only on user I/O pins but not on dedicated configuration pins, JTAG pins, or dedicated clock pins. If you enable this option, you cannot use the bus-hold feature.

For the weak pull-up resistor value, refer to the *Device Datasheet for Arria V Devices* chapter.

Differential Transmitter

The Arria V transmitter features dedicated circuitry to provide support for **LVDS** signaling. The dedicated circuitry consists of a true differential buffer, a serializer, and fractional PLLs that you can share between the transmitter and receiver. The differential buffer can drive out **LVDS**, **mini-LVDS**, and **RSDS** signaling levels. The serializer takes up to 10 bits wide parallel data from the FPGA fabric, clocks it into the load registers, and serializes it using shift registers that are clocked by the fractional PLL before sending the data to the differential buffer. The MSB of the parallel data is transmitted first.

Programmable Pre-Emphasis

The V_{OD} setting and the output impedance of the driver set the output current limit of a high-speed transmission signal. At a high frequency, the slew rate may not be fast enough to reach the full V_{OD} level before the next edge, producing pattern-dependent jitter.

With pre-emphasis, the output current is boosted momentarily during change of state 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. The amount of pre-emphasis required depends on the attenuation of the high-frequency component along the transmission line.

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

Differential Output Voltage

The Arria V **LVDS** transmitters support programmable V_{OD}. The programmable V_{OD} settings allow you to adjust the output eye opening to optimize the trace length and power consumption. A higher V_{OD} swing improves voltage margins at the receiver end, and a smaller V_{OD} swing reduces power consumption.

OCT Support

On-chip termination (OCT) maintains signal quality, saves board space, and reduces external component costs. Arria V devices support on-chip series termination (R_S OCT), on-chip parallel termination (R_T OCT), and on-chip differential termination (R_D OCT).

Arria V 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 signal reflections on PCB traces.

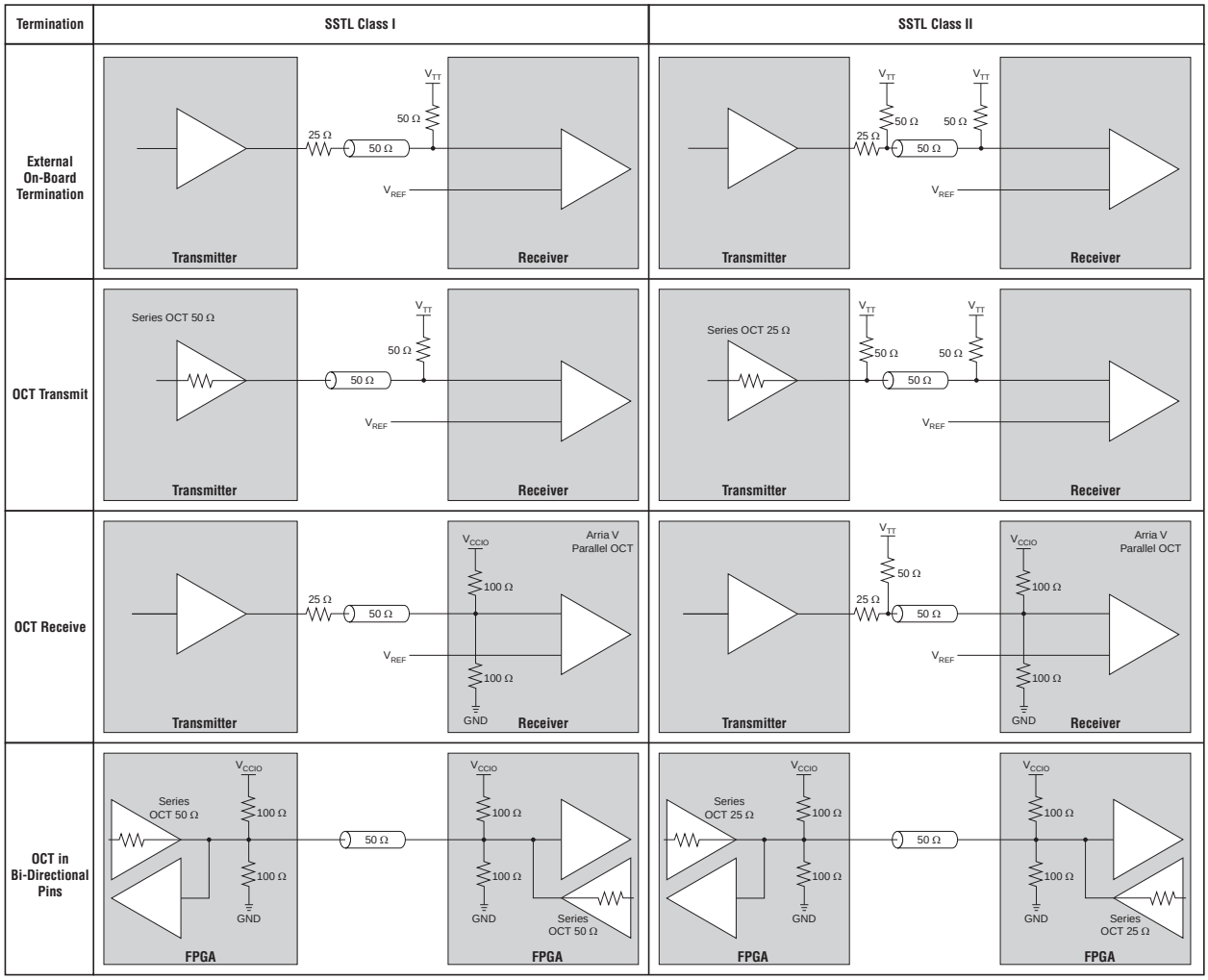
Termination Schemes for I/O Standards

Arria V devices support several termination schemes for different I/O standards.

SSTL I/O Standard Termination

Figure 1–31 shows the details of **SSTL** I/O termination on Arria V devices.

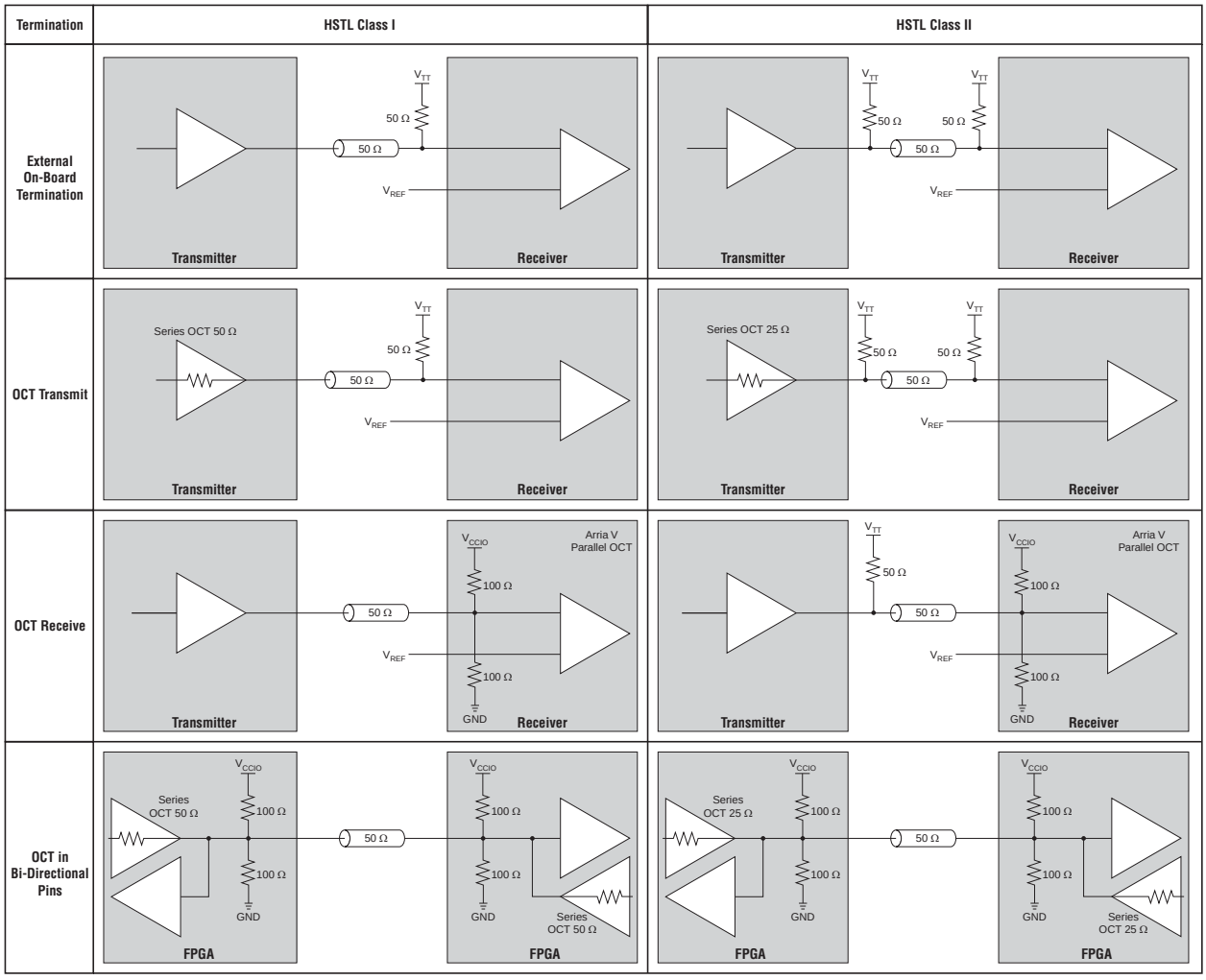
Figure 1–31. SSTL I/O Standard Termination



HSTL I/O Standard Termination

Figure 1–32 shows the details of **HSTL** I/O termination on Arria V devices.

Figure 1–32. HSTL I/O Standard Termination

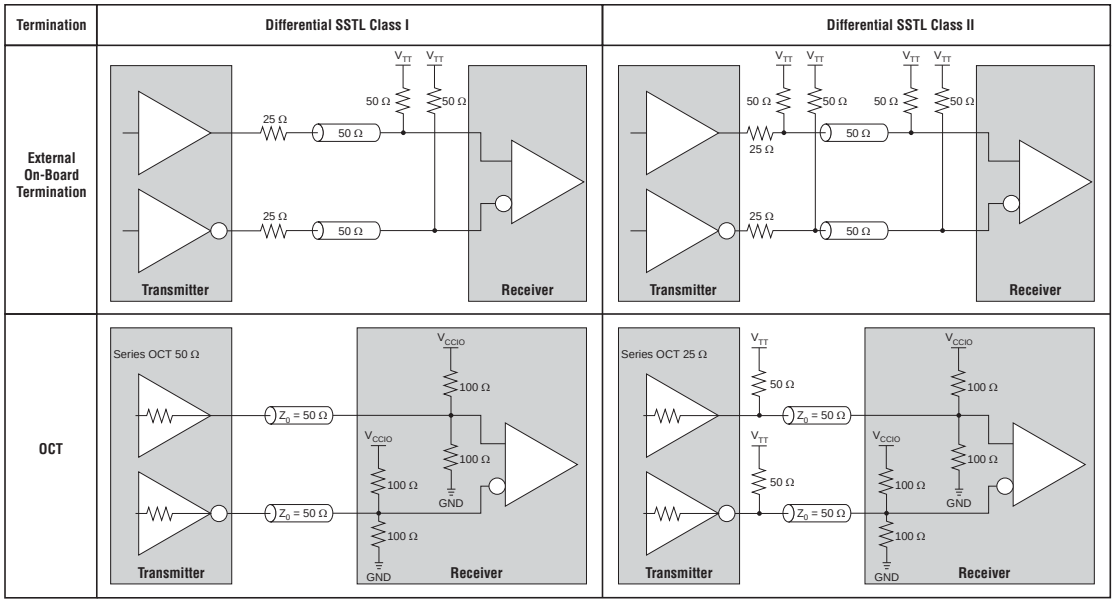


You cannot use R_S and R_T OCT simultaneously. For more information, refer to the *I/O Features in Arria V Devices* chapter.

Differential SSTL I/O Standard Termination

Figure 1–33 shows the details of **Differential SSTL** I/O termination on Arria V devices.

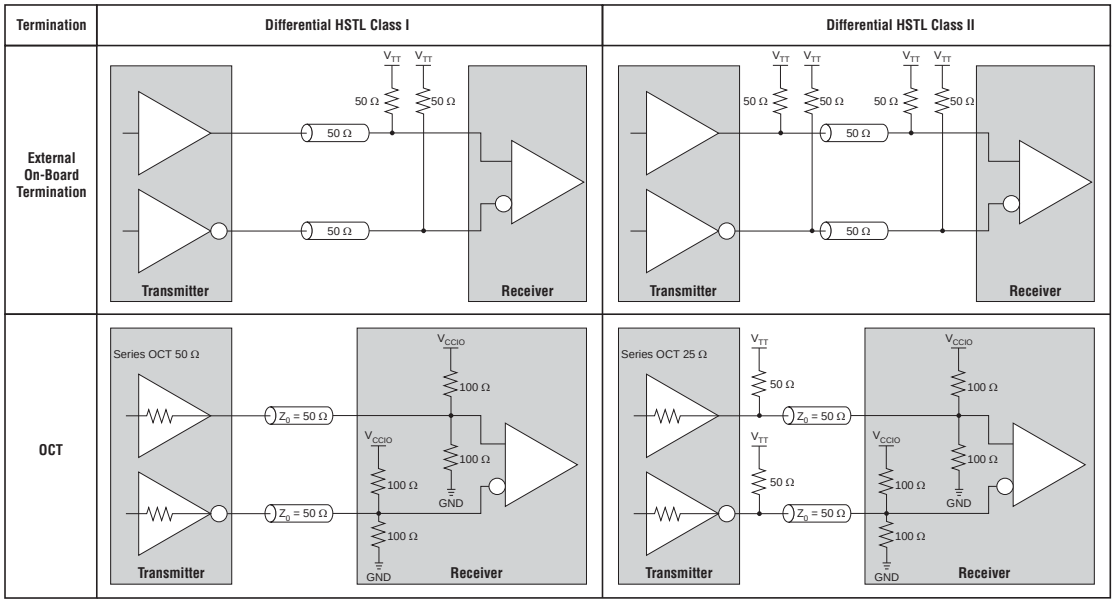
Figure 1–33. Differential SSTL I/O Standard Termination



Differential HSTL I/O Standard Termination

Figure 1–34 shows the details of **Differential HSTL** I/O standard termination on Arria V devices.

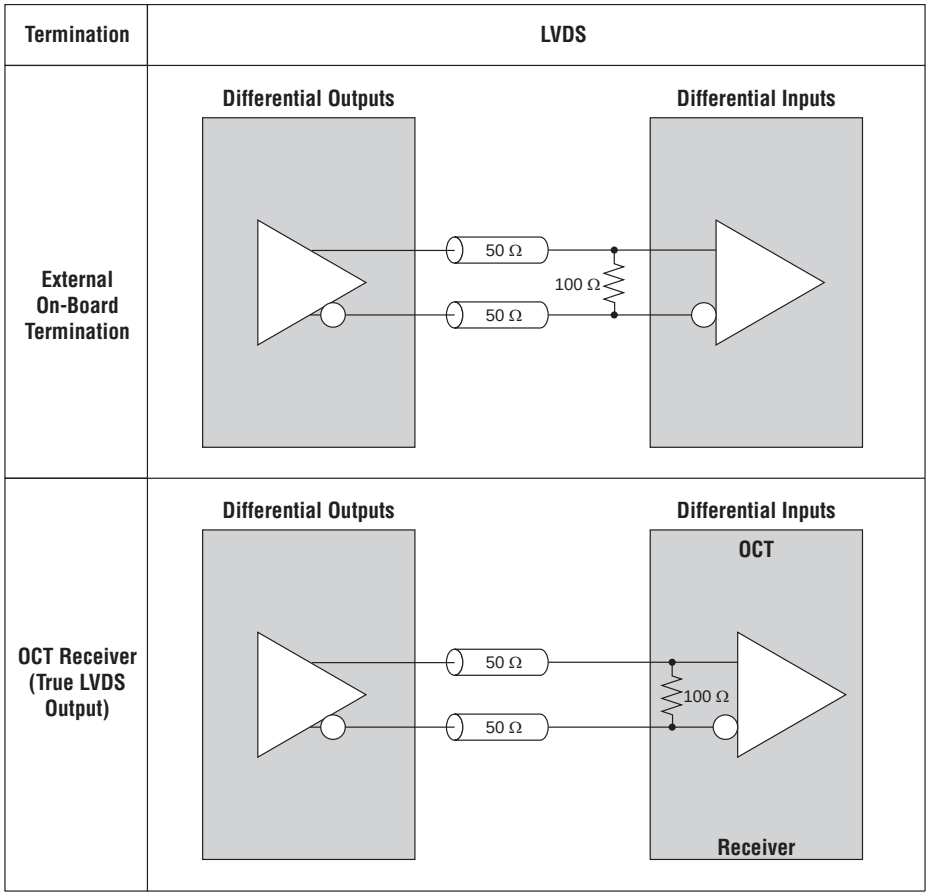
Figure 1–34. Differential HSTL I/O Standard Termination



LVDS, RSDS, and Mini-LVDS I/O Standard Termination

Figure 1–35 shows the **LVDS** I/O standard termination. The on-chip differential resistor is available in all I/O banks.

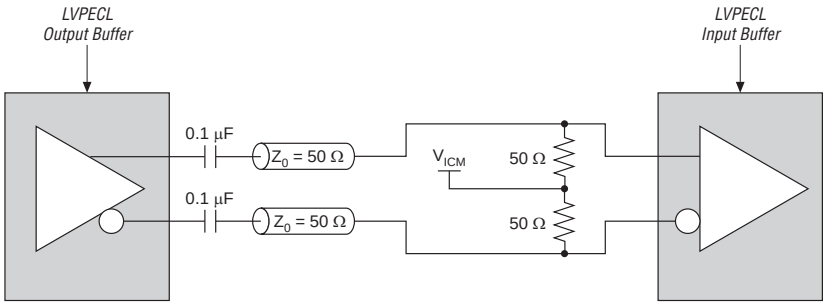
Figure 1–35. LVDS I/O Standard Termination



LVPECL I/O Standard Termination

Arria V devices support the **LVPECL** I/O standard on input clock pins only. **LVPECL** output operation is not supported. Use **LVDS** input buffers to support the **LVPECL** input operation. You require AC coupling when the **LVPECL** common-mode voltage of the output buffer does not match the **LVPECL** input common-mode voltage. Figure 1–36 shows the AC-coupled termination scheme.

Figure 1–36. LVPECL AC-Coupled Termination *(1)*

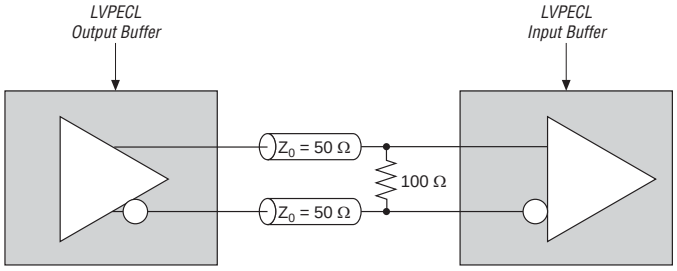


Note to Figure 1–36:

(1) The **LVPECL** AC/DC-coupled termination is applicable only when you use an Altera FPGA transmitter.

Support for DC-coupled **LVPECL** is available if the **LVPECL** output common mode voltage is within the Arria V **LVPECL** input buffer specification, as shown in Figure 1–37.

Figure 1–37. LVPECL DC-Coupled Termination *(1)*



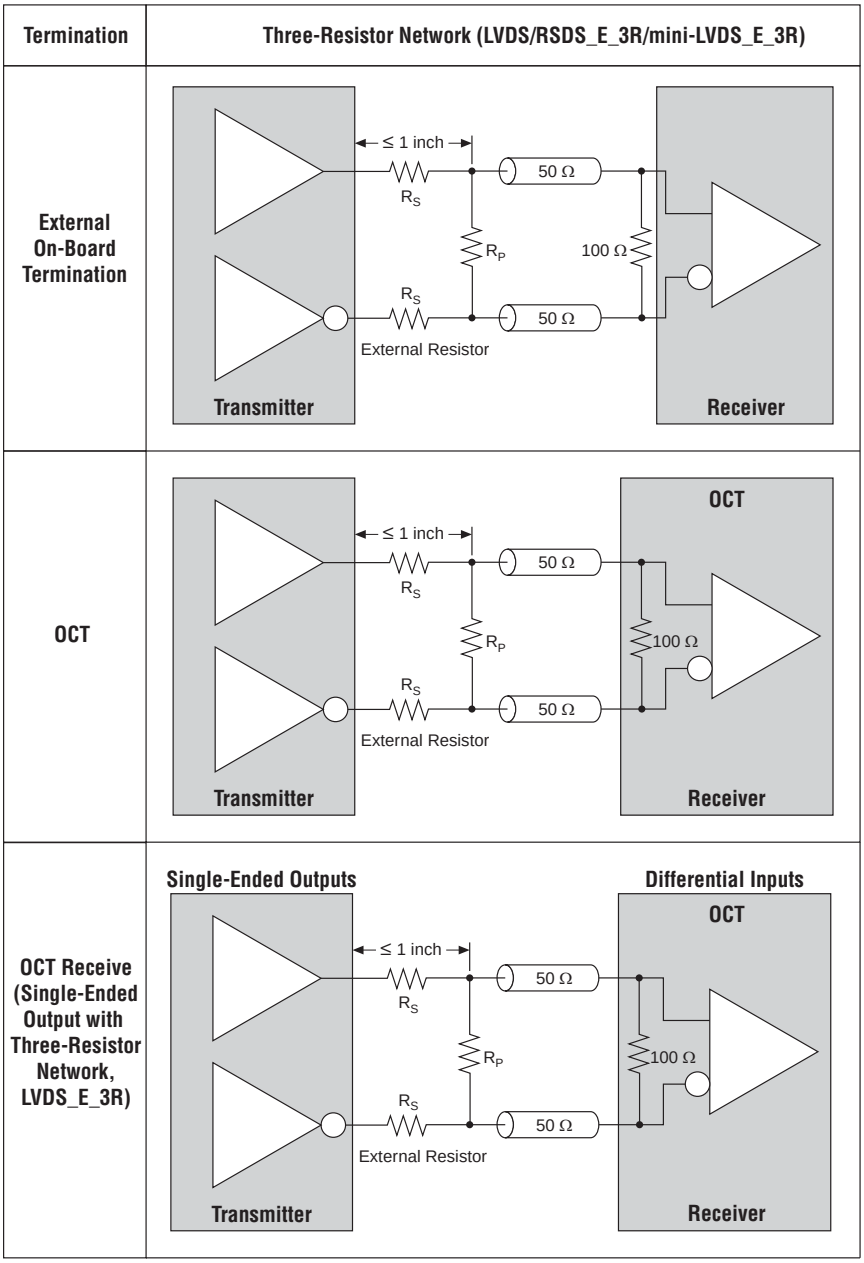
Note to Figure 1–37:

(1) The **LVPECL** AC/DC-coupled termination is applicable only when you use an Altera FPGA transmitter.

Emulated LVDS, RSDS, and Mini-LVDS I/O Standard Termination

Emulated **RSDS** and **mini-LVDS** output buffers use two single-ended output buffers with external three-resistor networks, and can be tri-stated. The output buffers are available in all I/O banks, as shown in Figure 1–38.

Figure 1–38. Emulated LVDS, RSDS, or Mini-LVDS I/O Standard Termination *(1)*



Note to Figure 1–38:

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

To meet the **RSDS** or **mini-LVDS** specifications, you require a resistor network to attenuate the output-voltage swing. You can modify the three-resistor network values to reduce power or improve the noise margin. Choose resistor values that satisfy Equation 1–2.

Equation 1–2. Resistor Network Calculation

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

Altera recommends that you perform additional simulations with IBIS or SPICE models to validate that the custom resistor values meet the **RSDS** or **mini-LVDS** I/O standard requirements.

For more information about the **RSDS** I/O standard, refer to the *RSDS Specification* document available on the National Semiconductor web site (www.national.com).

I/O Interface Design Considerations

There are several considerations that require your attention to ensure the success of your designs.

3.3-V I/O Interface

To ensure device reliability and proper operation when you use the Arria V device for 3.3-V I/O interfacing, do not violate the absolute maximum ratings of the device.

Altera recommends that you perform IBIS or SPICE simulations to determine that the overshoot and undershoot voltages are within the specifications.

When you use the Arria V device I/O as a transmitter, use slow slew rate and series termination to limit the overshoot and undershoot at the I/O pins. Limit the overshoot by reducing the V_{CCIO} of the bank to 3.0 V.

When you use the Arria V device I/O as a receiver, Altera recommends that you use the on-chip clamp diode to limit the overshoot or undershoot voltage at I/O pins.

For more information about absolute maximum rating and maximum allowed overshoot during transitions, refer to the *Device Datasheet for Arria V Devices* chapter.

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 the devices.

Non-Voltage-Referenced Standards

Each I/O bank of an Arria V device has its own `VCCIO` pins and supports only one V_{CCIO} (1.2, 1.25, 1.35, 1.5, 1.8, 2.5, 3.0, or 3.3 V). An I/O bank can simultaneously support any number of input signals with different I/O standard assignments.

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 I/O bank of the Arria V device contains a dedicated VREF pin. Each bank can have only a single V_{CCIO} voltage level and a single V_{REF} voltage level.

An I/O bank featuring single-ended or differential standards can support voltage-referenced standards if all voltage-referenced standards in that I/O bank use the same V_{REF} setting. Voltage-referenced bidirectional and output signals must be the same as the V_{CCIO} voltage of the I/O bank.

For example, you can place only **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 voltage-referenced and non-voltage-referenced pins by applying each of the rule sets individually. For example, an I/O bank can support **SSTL-18** inputs and outputs, 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 **1.5-V HSTL** I/O standards with a 1.5-V V_{CCIO} and 0.75-V V_{REF}.

V_{CCPD} Restriction

One VCCPD pin is shared in a group of I/O banks. For example, the I/O banks with the same number—8A, 8B, 8C, and 8D—form a group and these I/O banks share the same VCCPD pin. This sharing is applicable to all I/O banks except I/O banks 4A and 7A—each of these I/O banks has their own individual VCCPD pin.

If one I/O bank in a group uses 3.0-V V_{CCPD}, other I/O banks in the same group must also use 3.0-V V_{CCPD}.

High-Speed Differential I/O Interfaces

This section provides basic information about the high-speed differential I/O interfaces of Arria V devices.

Use the information in this section in conjunction with the *I/O Features in Arria V Devices* chapter.

Differential Pin Placement Guidelines

This section describes the pin placement guidelines with and without DPA usage. DPA usage adds some constraints on the placement of high-speed differential channels.

Differential pin placement guidelines have been established to ensure proper high-speed operation. The Quartus II compiler automatically checks the design and issues an error message if the guidelines are not followed.

DPA-enabled differential channels refer to DPA mode or soft-CDR mode; DPA disabled channels refer to non-DPA mode.

DPA-Enabled Channels, DPA-Disabled Channels, and Single-Ended I/Os

When you enable a DPA channel in a bank, you can use both single-ended I/Os and differential I/O standards in the bank.

You can place double data rate I/O (DDIO) output pins within I/O modules that have the same pad group number as a SERDES differential channel. However, you cannot place SDR I/O output pins within I/O modules that have the same pad group number as a receiver SERDES differential channel. You must implement the input register within the FPGA fabric logic.

Guidelines for DPA-Enabled Differential Channels

The Arria V device family has differential receivers and transmitters in all I/O blocks. Each receiver has a dedicated DPA circuit to align the phase of the clock to the data phase of its associated channel. When you use DPA-enabled channels in differential banks, 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.

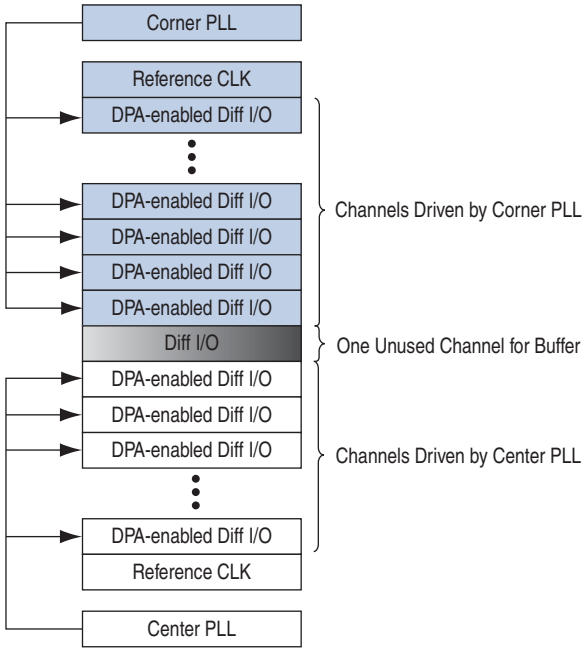
Using Center and Corner PLLs

If two PLLs drive the DPA-enabled channels in a bank—the corner and center PLL drive one group each—there must be at least one row of separation between the two groups of DPA-enabled channels, as shown in Figure 1–39. 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 1–39 to Figure 1–47 show guidelines for usage of corner and center PLLs in the DPA or non-DPA LVDS channels in the high-speed **LVDS** I/O banks. These figures do not represent the exact locations of the high-speed **LVDS** I/O banks.

For information about high-speed **LVDS** I/O banks locations, refer to the *High-Speed Differential I/O Interfaces with DPA in Arria V Devices* chapter.

Figure 1–39. Center and Corner PLLs Driving DPA-enabled Differential I/Os in the Same Bank

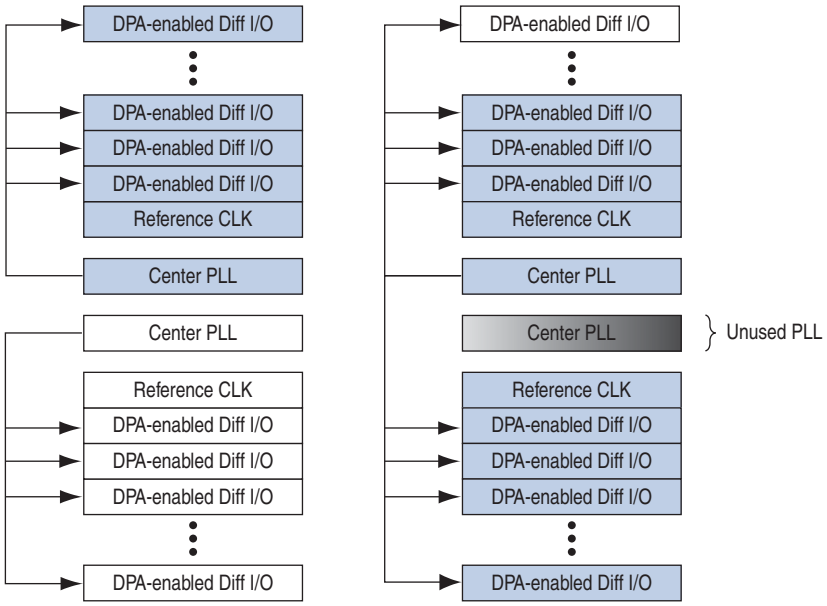


Using Both Center PLLs

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 1–39.

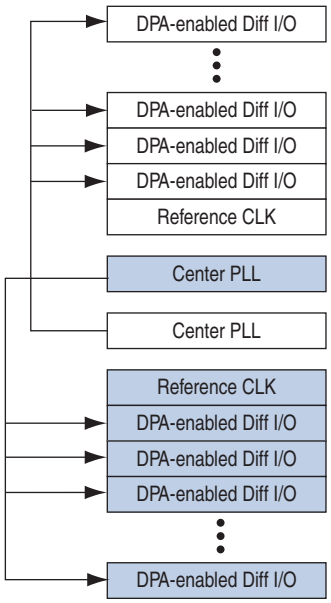
If one of the center PLLs drives the DPA-enabled channels in the upper and lower I/O banks, you cannot use the other center PLL for DPA-enabled channels, as shown in Figure 1–40.

Figure 1–40. Center PLLs Driving DPA-enabled Differential I/Os



If the upper center PLL drives the DPA-enabled channels in the lower I/O bank, the lower center PLL cannot drive the DPA-enabled channels in the upper I/O bank, and vice versa. The center PLLs cannot drive cross-banks simultaneously, as shown in Figure 1–41.

Figure 1–41. Invalid Placement of DPA-enabled Differential I/Os Driven by Both Center PLLs



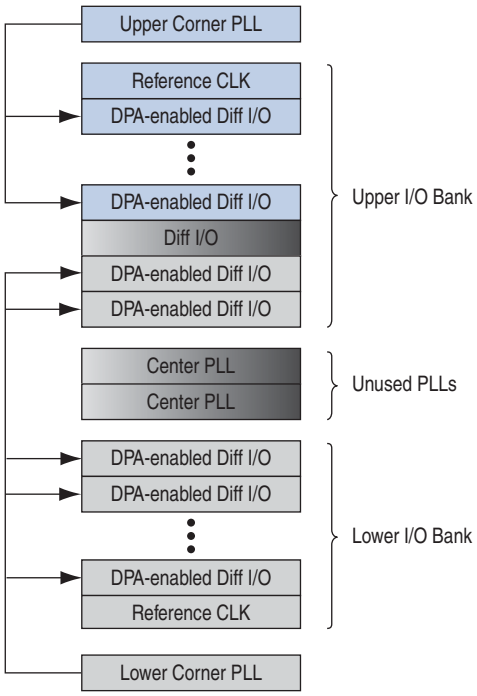
Using Both Corner PLLs

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 1–42.

Figure 1–42. Corner PLLs Driving DPA-enabled Differential I/Os



Guidelines for DPA-Disabled Differential Channels

When you use DPA-disabled channels, adhere to the guidelines in the following sections.

DPA-Disabled Channel Driving Distance

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

Using Corner and Center PLLs

You can use a corner PLL to drive all transmitter channels and a center PLL to drive all DPA-disabled receiver channels in the same I/O bank. You can drive a transmitter channel and a receiver channel in the same LAB row by two different PLLs, as shown in Figure 1–43.

Figure 1–43. Corner and Center PLLs Driving DPA-Disabled Differential I/Os in the Same Bank

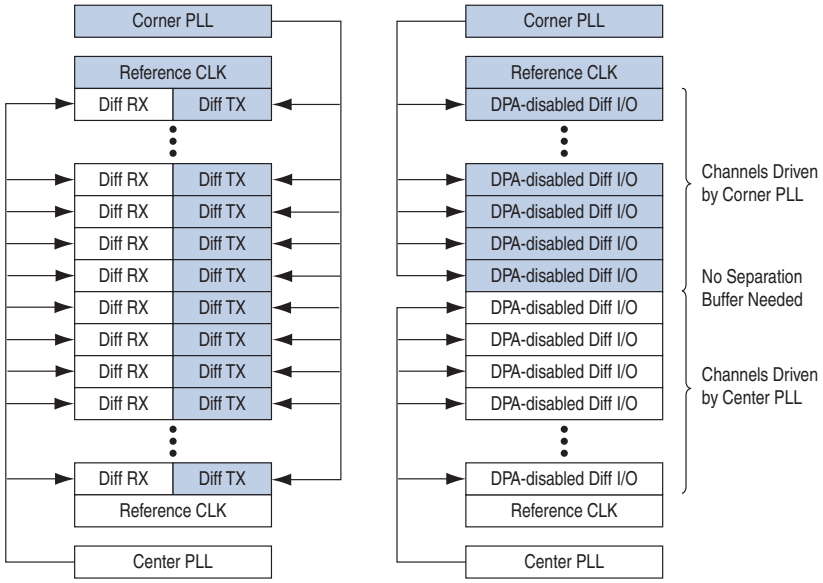
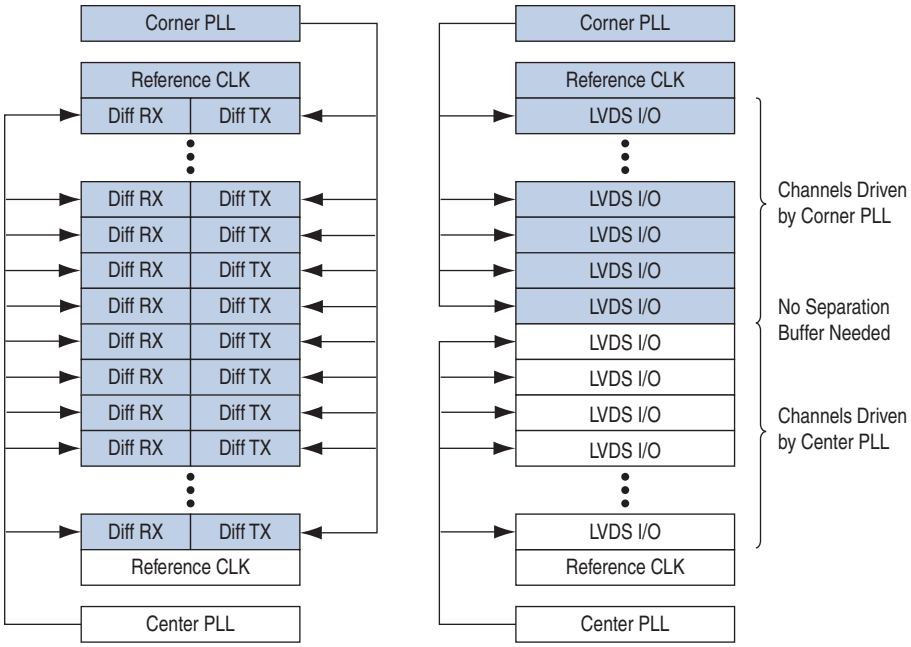


Figure 1–44. Corner and Center PLLs DrivingDPA-disabled Differential I/Os in the Same Bank



A corner PLL and a center PLL can drive duplex channels in the same I/O bank if the channels that are driven by each PLL are not interleaved. You do not require separation between the group of channels that are driven by the corner and center, left and right PLLs. Refer to Figure 1–43 and Figure 1–45.

Figure 1–45. Invalid Placement of DPA-disabled Differential I/Os Due to Interleaving of Channels Driven by the Corner and Center PLLs

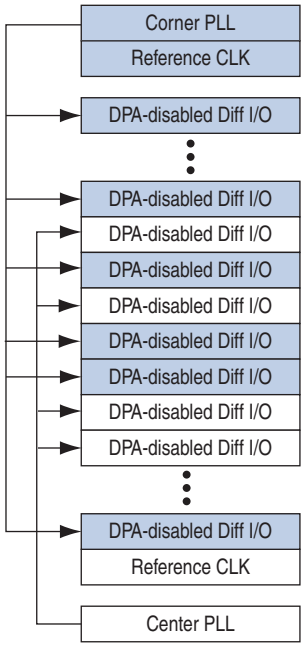
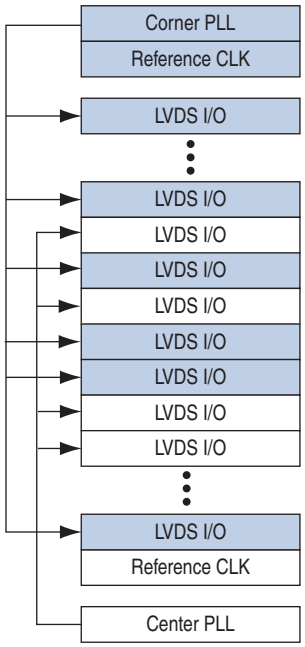


Figure 1–46. Invalid Placement of LVDS I/Os Due to Interleaving of Channels Driven by the Corner and Center PLLs

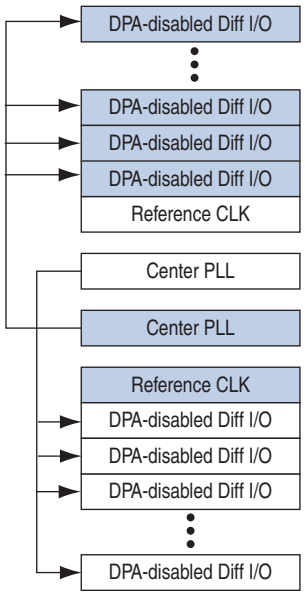


Using Both Center PLLs

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 1–47.

The center PLLs are available in Arria V GX devices on the right I/O banks, and in Arria V GZ devices on the right and left I/O banks.

Figure 1–47. Both Center PLLs Driving Cross-Bank DPA-Disabled Channels Simultaneously



Using Both Corner PLLs

You can use both corner PLLs to drive DPA-disabled channels simultaneously. You can use a corner PLL to drive all the transmitter channels and the other corner PLL to drive all the 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 that are driven by each PLL are not interleaved. You do not require separation between the groups of channels that are driven by both corner PLLs.

External Memory Interfaces

This section describes the basic information of memory interface pin, delay-locked loop (DLL), and DQS logic block of Arria V devices.

Use the information in this section in conjunction with the *External Memory Interfaces in Arria V Devices* chapter.

Memory Interface Pin

A typical memory interface requires data (D, Q, or DQ), data strobe (DQS/CQ/QK and DQSn/CQn/QK#), address, command, and clock (DK and DK#) pins. Some memory interfaces use data mask (DM, BWSn, or NWSn) pins to enable write masking and QVLD pins to indicate that the read data is ready for capture.

For the Arria V pin tables for a particular Arria V device, refer to the Arria V Device Pin-Out Files page of the Altera website.

DQ pins are bidirectional signals, as in DDR3 and DDR2 SDRAM, and RLDRAM II common I/O interfaces, or unidirectional signals, QDR II+ and QDR II SRAM, and RLDRAM II separate I/O devices. Connect the unidirectional read-data signals to Arria V DQ pins and the unidirectional write-data signals to a different DQ/DQS group than the read DQ/DQS group. You must assign the write clocks to the DQS/DQSn pins associated to this write DQ/DQS group.

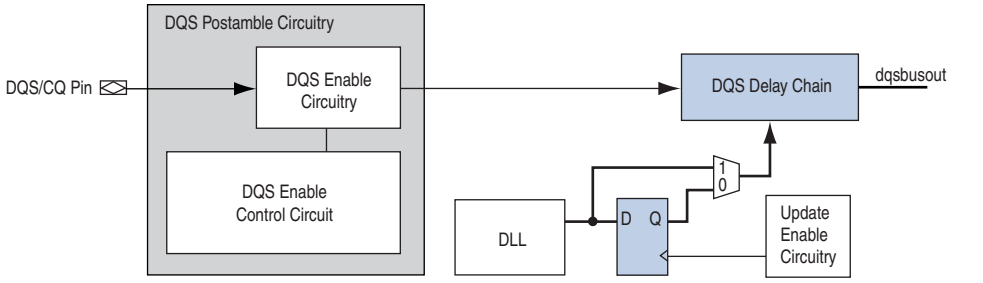
Delay-Locked Loop

The DLL uses a frequency reference to dynamically generate control signals for the delay chains in each of the DQS/CQ pins, allowing it to compensate for process, voltage, and temperature (PVT) variations. The DQS delay settings are Gray-coded to reduce jitter if the DLL updates the settings.

DQS Logic Block

Figure 1–48 shows a simple block diagram of the DQS logic block in the Arria V devices.

Figure 1–48. Simplified Diagram of the DQS Logic Block



DQS Postamble Circuitry

For external memory interfaces that use a bidirectional read strobe (DDR3 and DDR2 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 and DDR2 SDRAM. The DQS postamble circuitry ensures that data is not lost if there is noise on the DQS line during the end of a read operation that occurs while DQS is in a postamble state.

DQS Delay Chain

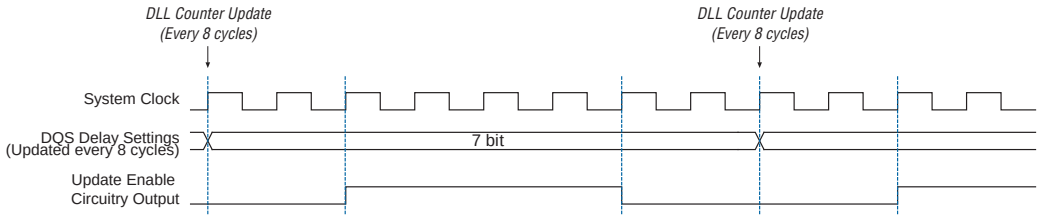
DQS delay chains consist of a set of variable delay elements to allow the input DQS/CQ signals to be shifted by the amount specified by the DQS phase-shift circuitry or the logic array. There are two delay elements in the DQS delay chain. The DQS/CQ pin is shifted by the DQS delay settings.

Update Enable Circuitry

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. The circuitry uses the input reference clock or a user clock from the core to generate the update enable output. The UniPHY IP uses this circuit by default.

Figure 1–49 shows an example waveform of the update enable circuitry output.

Figure 1–49. DQS Update Enable Waveform



Configuration, Design Security, and Remote System Upgrades

This section describes the basic information of supported configuration schemes for Arria V devices, instructions for executing the required configuration schemes, and all the necessary option pin settings.

Use the information in this section in conjunction with the *Configuration, Design Security, and Remote System Upgrades in Arria V Devices* chapter.

Configuration Sequence

The following sections describe the general configuration process for power up, reset, configuration, configuration error, initialization, and user mode.

Power Up

To begin the configuration process, you must fully power-up all the power supplies monitored by the power-on reset (POR) circuitry to the appropriate voltage levels.

All power supplies, including V_{CCPGM} and V_{CCPD}, must ramp-up from 0 V to the desired voltage level within the ramp-up time specification. If these supplies are not ramped up within this specified time, your Arria V device will not configure successfully. If your system cannot ramp-up the power supplies within the required ramp-up time specification, you must hold nCONFIG low until all the power supplies are stable.

For more information about the ramp-up time specifications, refer to the *Power Management in Arria V Devices* chapter.

Reset

After power-up, the Arria V 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 the user I/O pins. After the device successfully exits POR, all the user I/O pins remain tri-stated until the device is configured.

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.

For more information about the POR delay specification, refer to the *Device Datasheet for Arria V Devices* chapter.

Configuration

Both nCONFIG and nSTATUS must be de-asserted at a logic-high level in order for the configuration stage to begin. For the fast passive parallel (FPP) and passive serial (PS) configuration schemes, the device receives configuration data on its DATA pins and the clock source on the DCLK pin. Configuration data is latched into the Arria V device on the rising edge of DCLK. For the active serial (AS) configuration scheme, the device receives configuration data on its AS_DATA[] pins and drives the clock source on the DCLK pin. Configuration data is latched into the Arria V device on the falling edge of DCLK.

After the Arria V device 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 has completed and initialization of the device can begin. For the FPP and PS schemes, DCLK must not be left floating at the end of configuration. You must drive it either high or low, whichever is convenient on your board.

For the FPP and PS schemes, there is no maximum DCLK period, which means you can stop the configuration by holding the DCLK low for an indefinite amount of time. To resume configuration, the external host must provide data on the DATA[] pins before sending the first DCLK rising edge.

To connect MSEL, refer to the *Configuration, Design Security, and Remote System Upgrades in Arria V Devices* chapter.

Configuration Error

If an error occurs during configuration, Arria V devices assert the nSTATUS signal low to indicate the data frame error and the CONF_DONE signal remains low.

For the PS or FPP configuration schemes, if the **Auto-restart configuration after error** option is turned on, the Arria V device releases the nSTATUS pin high after the t_{STATUS} period and retries the configuration. If this option is turned off, the system must monitor nSTATUS for errors and sends a low-to-high signal on nCONFIG with at least a duration of t_{CFG} to restart the configuration.

For more information about t_{STATUS} and t_{CFG} timing parameters, refer to the *Device Datasheet for Arria V Devices* chapter.

The **Auto-restart configuration after error** option is available in the Quartus II software from the **General** panel of the **Device and Pin Options** dialog box.

Initialization

In Arria V devices, initialization begins after CONF_DONE goes high. For the FPP and PS configuration schemes, two DCLK falling edges are required after the last configuration byte is sent to the Arria V device to begin the initialization of the device for both uncompressed and compressed configuration data.

The initialization clock source is from the internal oscillator, CLKUSR, or the DCLK pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Arria V device provides itself with enough clock cycles for proper initialization.

Table 1–6 lists the initialization clock source option, the applicable configuration schemes, and the maximum frequency for Arria V devices.

Table 1–6. Initialization Clock Source Option and the Maximum Frequency for Arria V Devices

Initialization Clock Source	Configuration Schemes	Maximum Frequency (MHz)	Minimum Number of Clock Cycles (1)
Internal Oscillator	AS, PS, and FPP	12.5	T_{init}
CLKUSR	AS, PS, and FPP (2)	125	

Notes to Table 1–6:

(1) The minimum number of clock cycles required for device initialization.

(2) To enable CLKUSR as the initialization clock source, turn on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software from the **General** panel of the **Device and Pin Options** dialog box.

If you use the optional CLKUSR pin as the initialization clock source and nCONFIG is pulled low to restart configuration during device initialization, ensure that CLKUSR or DCLK continues toggling until nSTATUS goes low and goes high again.

CLKUSR provides you with the flexibility to synchronize initialization of multiple devices or to delay initialization. Supplying a clock on the CLKUSR pin during initialization does not affect configuration. After CONF_DONE goes high, CLKUSR or DCLK is enabled after the time specified by t_{CD2CU}. When this time period elapses, Arria V devices require a minimum number of clock cycles as specified by T_{init} to initialize properly and enter user mode as specified by the t_{CD2UMC} parameter.

For more information about the t_{CD2CU}, T_{init}, and t_{CD2UMC} timing parameters, refer to the *Device Datasheet for Arria V Devices* chapter.

User Mode

The Arria V device enters user mode when initialization is complete. You can monitor the end of the initialization stage by enabling the optional INIT_DONE pin. If you enable the INIT_DONE pin, the low-to-high transition of INIT_DONE indicates the device has completed initialization and has entered user mode. In this mode, your design is executed. The user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

At any time during the configuration stage or user mode operation, you can initiate a reconfiguration by setting a low pulse on the nCONFIG pin. The pulse must meet the minimum t_{CFG} low-pulse width. When nCONFIG is pulled low, the nSTATUS and CONF_DONE pins are pulled low and all I/O pins are tri-stated. Configuration begins when the nCONFIG and nSTATUS pins return to a logic-high level.

Fast Passive Parallel Configuration

The FPP configuration using an external host provides the fastest method to configure Arria V devices. FPP is supported in two data widths—8 bits and 16 bits. You can perform an FPP configuration of Arria V devices using an external host such as a MAX® II device, MAX V device, or microprocessor. The external host controls the transfer of configuration data from a storage device, such as flash memory, to the

target Arria V device. You can store configuration data in raw binary file (**.rbf**), hexadecimal file (**.hex**), or tabular text file (**.ttf**) formats. Therefore, the design that controls the configuration stages, such as fetching the data from flash memory and sending it to the device, must be stored in the MAX II device, MAX V device, or microprocessor.

The parallel flash loader (PFL) megafunction in MAX II or MAX V devices provides an efficient method to program common flash interface (CFI) flash memory devices through the JTAG interface. The PFL megafunction also acts as a controller to read configuration data from the flash memory device and configures the Arria V device. The PFL megafunction supports both the PS and FPP configuration schemes.

For more information about the PFL megafunction, refer to the *Parallel Flash Loader Megafunction User Guide*.

Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device for both uncompressed and compressed configuration data in an FPP configuration.

DCLK-to-DATA[] Ratio for the FPP Configuration

The FPP configuration requires a different DCLK-to-DATA[] ratio when you enable the design security, decompression, or both features. Table 1–7 lists the DCLK-to-DATA[] ratio for each combination.

Table 1–7. DCLK-to-DATA[] Ratio *(1)*

Configuration Scheme	Decompression	Design Security	DCLK-to-DATA[] Ratio
FPP x8	Disabled	Disabled	1
	Disabled	Enabled	1
	Enabled	Disabled	2
	Enabled	Enabled	2
FPP x16	Disabled	Disabled	1
	Disabled	Enabled	2
	Enabled	Disabled	4
	Enabled	Enabled	4

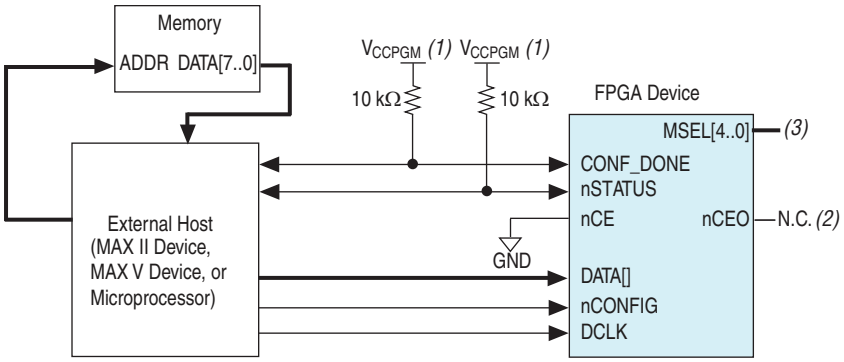
Note to Table 1–7:

(1) Depending on the DCLK-to-DATA[] ratio, the host must send a DCLK frequency that is r times the data rate in bps, or words per second (Wps). For example, in FPP x16 when the DCLK-to-DATA[] ratio is 2, the DCLK frequency must be 2 times the data rate in Wps. Arria V devices use the additional clock cycles to decrypt and decompress the configuration data.

If the DCLK-to-DATA[] ratio is greater than 1, at the end of configuration, you can only stop the DCLK (DCLK-to-DATA[] ratio – 1) clock cycles after the last data is latched into the Arria V device.

Figure 1–50 shows the configuration interface connections between the Arria V device and a MAX II or MAX V device for a single device configuration.

Figure 1–50. Single Device FPP Configuration Using an External Host



Notes to Figure 1–50:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Arria V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all configuration system I/Os with V_{CCPGM}.

(2) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.

(3) The MSEL pin settings vary for different data widths, configuration voltage standards, and POR delays.

FPP Multi-Device Configuration

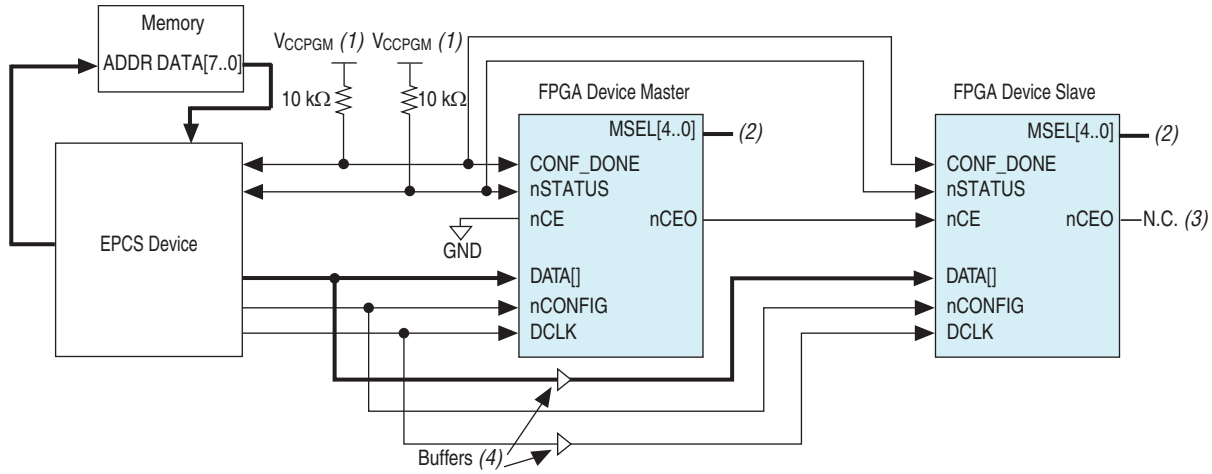
For an FPP multi-device configuration, you can configure all the devices with different sets of configuration data (multiple SRAM object files [**.sof**s]) or with the same configuration data (single **.sof**). In both cases, the nCONFIG, nSTATUS, DCLK, DATA[], and CONF_DONE pins are connected to every device in the chain. Ensure that the DCLK and data line are buffered for every fourth device. This ensures signal integrity and prevents clock skew problems.

In an FPP multi-device configuration, the CONF_DONE and nSTATUS pins are tied together. Therefore, all devices initialize and enter user mode at the same time. If any device detects an error, configuration stops for the entire chain and you must reconfigure all the devices. 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.

For an FPP multi-device configuration, all devices in the chain must have the same data width. If you are using the FPP x16 configuration scheme, all devices in the chain must use the FPP x16 configuration scheme. If you are using the FPP x8 configuration scheme, use the Arria V device with other FPGA devices that support the FPP x8 configuration scheme.

Figure 1–51 shows how to configure multiple devices using a MAX II or MAX V device when both devices receive a different set of configuration data (multiple **.sof**s).

Figure 1–51. Multi-Device FPP Configuration Using an External Host When Both Devices Receive a Different Set of Configuration Data



Notes to Figure 1–51:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Arria V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all configuration system I/Os with V_{CCPGM}.

(2) The MSEL pin settings vary for different data widths and POR delays.

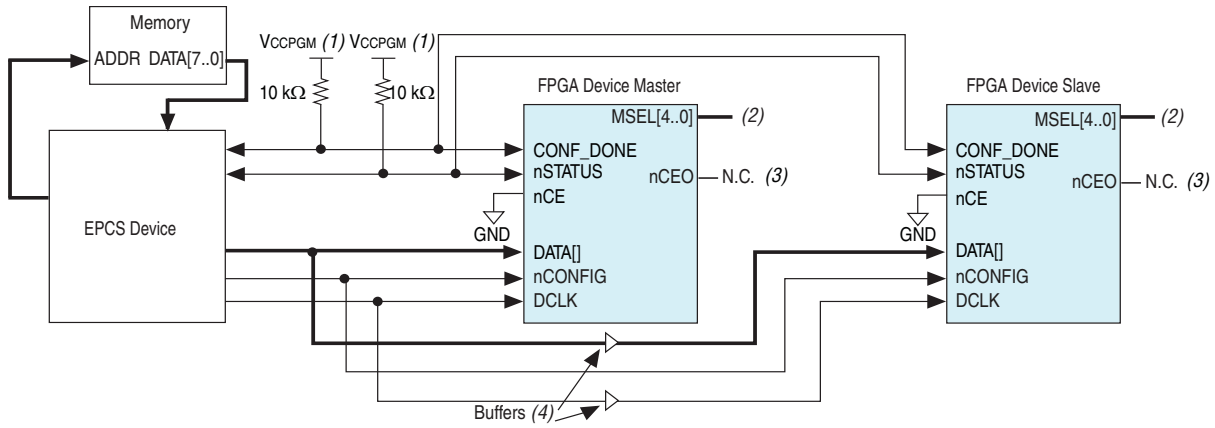
(3) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.

(4) Connect the repeater buffers between the Arria V master and slave device for DATA[] and DCLK for every fourth device.

In Figure 1–51, 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 to the second device is transparent to the MAX II device, MAX V device, or microprocessor.

Figure 1–52 shows the FPP configuration setup for multiple devices when both Arria V devices receive the same configuration data (single **.sof**).

Figure 1–52. Multiple Device FPP Configuration Using an External Host When Both Devices Receive the Same Data



Notes to Figure 1–52:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Arria V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all configuration system I/Os with V_{CCPGM}.

(2) The MSEL pin settings vary for different data widths and POR delays.

(3) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.

(4) Connect the repeater buffers between the Arria V master and slave device for DATA[] and DCLK for every fourth device. .

In Figure 1–52, because both nCE pins are tied to GND, both devices in the chain begin and complete the configuration and enter user mode at the same time.

To configure an FPP multi-device configuration with a single **.sof**, all the Arria V devices in the chain must be in the same package and density.

FPP Configuration Timing

For more information about FPP timing parameters, refer to the *Device Datasheet for Arria V Devices* chapter.

Active Serial Configuration

The AS configuration scheme supports AS x1 (1-bit data width) and AS x4 (4-bit data width) modes. Serial configuration device (EPCS) supports AS x1 mode and quad-serial configuration device (EPCQ) supports AS x1 and AS x4 modes. The AS x4 mode provides four times faster configuration time than the AS x1 mode.

EPCS and EPCQ are low-cost devices with non-volatile memory that feature a simple four-pin interface or six-pin interface, respectively, and a small form factor. These features make the AS configuration scheme an ideal low-cost configuration solution.

If you wish to gain control of the EPCS 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.

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

For more information about EPCQ, refer to the *Quad-Serial Configuration (EPCQ) Devices Datasheet* chapter in volume 2 of the *Configuration Handbook*.

AS mode supports a DCLK frequency up to 100 MHz. You can choose CLKUSR or internal oscillator as the configuration clock source that drives DCLK. If you use the internal oscillator as the configuration clock source, you can choose a 12.5, 25, 50, or 100 MHz clock from the **Configuration** panel in the **Device and Pins Option** settings.

For more information about the DCLK frequency specification in the AS configuration scheme, refer to the *Device Datasheet for Arria V Devices* chapter.

You can choose the internal oscillator or CLKUSR as the DCLK clock source by selecting the option under **Device and Pins Option** settings, in the **Configuration** panel of the Quartus II software. This sets a specific option in the programming file. By default, in the AS scheme, Arria V devices power-up and begin configuration with a 12.5 MHz internal oscillator as the DCLK clock source. After reading the option bits from the programming file, Arria V devices continue using the internal oscillator at the 12.5 MHz frequency, switch to a higher internal oscillator clock frequency, or switch to the CLKUSR pin.

If you choose CLKUSR as the configuration clock source, the maximum frequency allowed is 100 MHz.

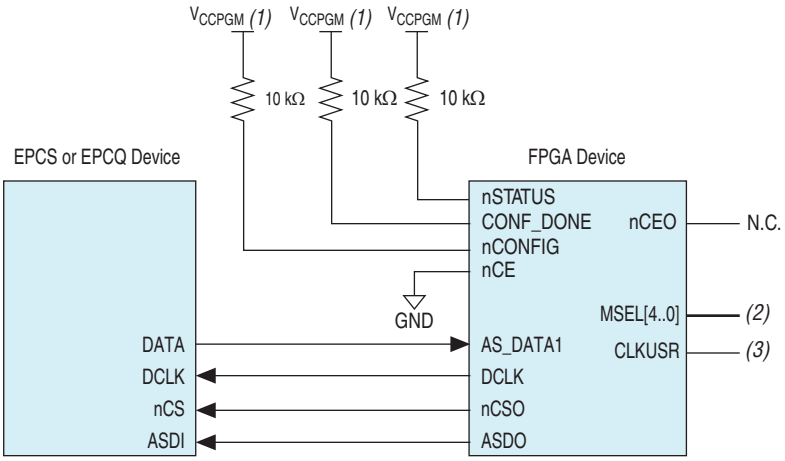
During device configuration, Arria V devices read the configuration data using the serial interface, decompress the data if necessary, and configure their SRAM cells. In the AS configuration scheme, the Arria V device controls the configuration interface. In the PS configuration scheme, the external host (a MAX II device, MAX V device, or microprocessor) controls the interface.

You can select between the AS x1 and AS x4 settings by selecting the option under **Device and Pins Option** settings, in the **Configuration** panel of the Quartus II software. This sets a specific option bit in the programming file. By default, in the AS scheme, Arria V devices power-up and begin configuration as an AS x1 mode. After reading the option bits from the programming file, Arria V devices either stay as AS x1 mode or switch to AS x4 mode for the rest of the configuration.

AS Single-Device Configuration

Figure 1–53 shows the single-device configuration setup for AS x1 mode.

Figure 1–53. Single Device AS x1 Mode Configuration

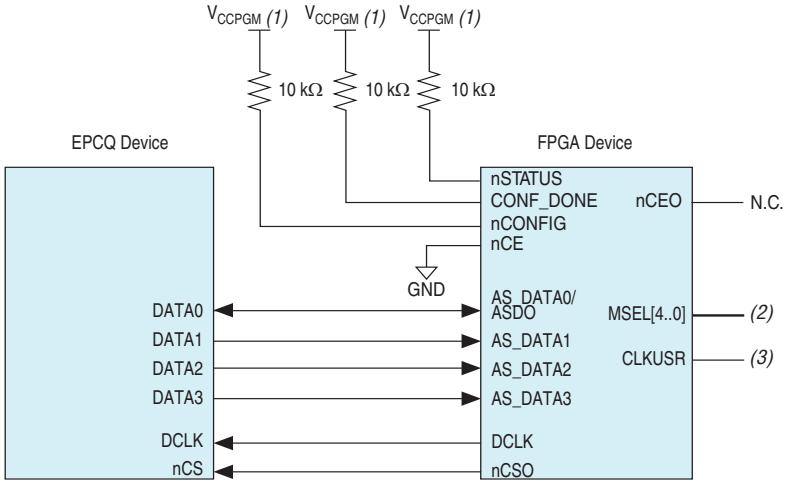


Notes to Figure 1–53:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0- or 3.3-V power supply.
(2) The MSEL pin settings vary for different configuration voltage standards and POR delays.
(3) Use the CLKUSR pin to supply the external clock source to drive DCLK during configuration.

Figure 1–54 shows the single-device configuration setup for AS x4 mode.

Figure 1–54. Single Device AS x4 Mode Configuration



Notes to Figure 1–54:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0- or 3.3-V power supply.
(2) The MSEL pin settings vary for different configuration voltage standards and POR delays.
(3) Use the CLKUSR pin to supply the external clock source to drive DCLK during configuration.

The Arria V device generates the serial clock (DCLK). The DCLK controls the entire configuration cycle and provides timing for the serial interface. In the AS configuration scheme, Arria V devices drive out control signals on the falling edge of DCLK and latch in the data on the following falling edge of DCLK.

During configuration, Arria V devices enable the EPCS or EPCQ by driving the nCSO output pin low, which connects to the chip select (nCS) pin of the EPCS or EPCQ. Arria V devices use the DCLK and serial data output (ASDO) pins to send operation commands and read address signals to the EPCS or EPCQ. The EPCS or EPCQ provides data on its serial data output (DATA[]) pin, which connects to the AS_DATA[] input of the Arria V devices.

AS Multi-Device Configuration

For the AS multi-device configuration scheme, you can configure all the devices with different sets of configuration data (different **.sof**s) or with the same configuration data (same **.sof**). In both cases, the nCONFIG, nSTATUS, DCLK, data line (AS_DATA1 on the master device and DATA0 on the slave device), and CONF_DONE pins are connected to every device in the chain. Ensure that the DCLK pin and data line are buffered for every fourth device.

The AS configuration scheme supports multi-device in AS x1 mode. AS x4 mode does not support the multi-device configuration setup.

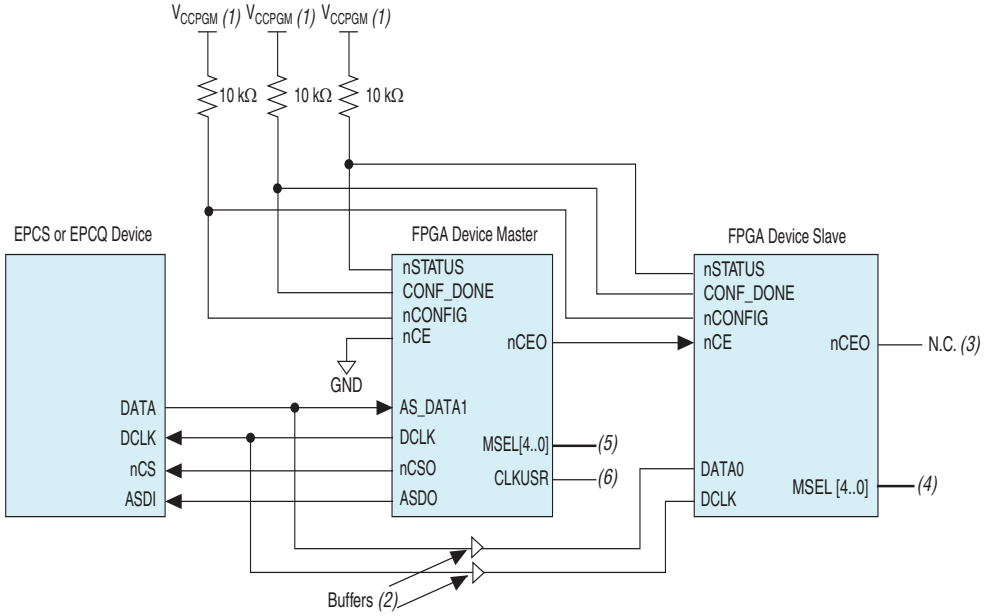
The AS multi-device configuration scheme does not support DCLK frequency of 100 MHz.

In the AS multi-device configuration, the nSTATUS, nCONFIG, and CONF_DONE pins are tied together. Therefore, all devices initialize and enter user mode at the same time. If any device detects an error, configuration stops for the entire chain and you must reconfigure all the devices. 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 this configuration scheme, the first Arria V device in the chain is the configuration master and controls the configuration of the entire chain. You must connect its MSEL pins to select the AS configuration scheme. The remaining Arria V devices are configuration slaves. You must connect their MSEL pins to select the PS configuration scheme. Any other Altera® device that supports a PS configuration can also be part of the chain as the configuration slave.

Figure 1–55 shows the multi-device configuration setup for AS x1 mode when both devices in the chain receive different sets of configuration data (multiple **.sof**s).

Figure 1–55. AS Multi-Device Configuration When Both Devices in the Chain Receive Different Sets of Configuration Data *(1)*



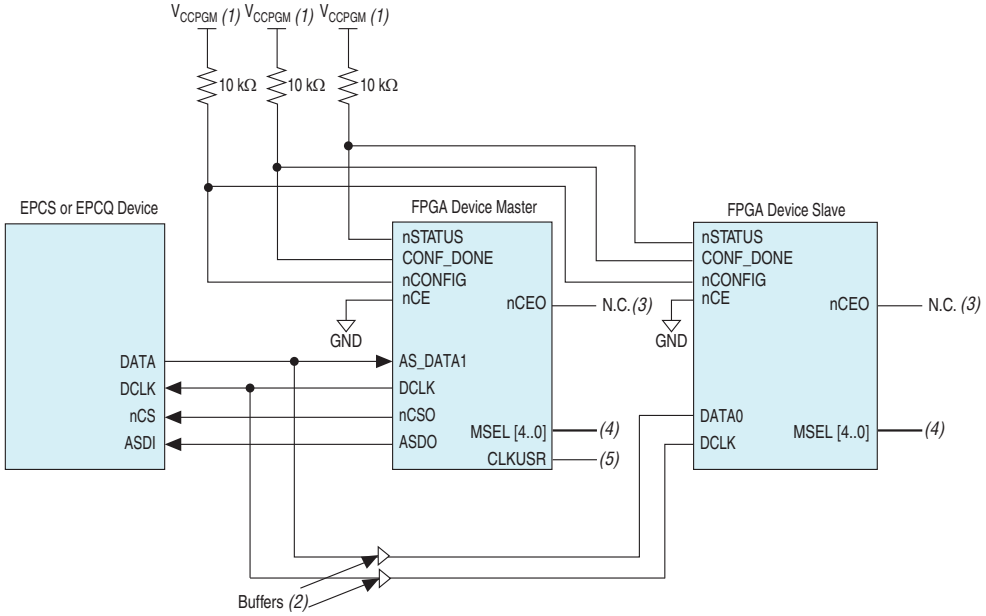
Notes to Figure 1–55:

(1) Connect the pull-up resistors to V_{CCPGM} at a 1.8-, 3.0-, or 3.3-V power supply.
(2) Connect the repeater buffers between the Arria V master and slave device for AS_DATA1 or DATA0 and DCLK for every fourth device.
(3) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.
(4) For the appropriate MSEL settings based on POR delay settings, set the slave device MSEL setting to the PS scheme.
(5) The MSEL pin settings vary for different configuration voltage standards and POR delays.
(6) Use the CLKUSR pin to supply the external clock source to drive DCLK during configuration.
(7) For 50 MHz frequency, delay DCLK in reference to DATA0 by a minimum of 5 ns and a maximum of 10 ns.

In Figure 1–55, 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 to the second device is transparent to the first device in the chain.

Figure 1–56 shows the multi-device configuration setup for AS x1 mode when all devices in the chain receive the same set of configuration data (single **.sof**).

Figure 1–56. AS Multi-Device Configuration When the Devices Receive the Same Data Using a Single .sof *(1)*



Notes to Figure 1–56:

(1) Connect the pull-up resistors to V_{CCPGM} at a 1.8-, 3.0-, or 3.3-V power supply.
(2) Connect the repeater buffers between the Arria V master and slave device for AS_DATA1 or DATA0 and DCLK.
(3) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.
(4) The MSEL pin settings vary for different configuration voltage standards and POR delays.
(5) Use the CLKUSR pin to supply the external clock source to drive DCLK during configuration.
(6) For 50 MHz frequency, delay DCLK in reference to DATA0 by a minimum of 5 ns and a maximum of 10 ns.

AS Interface Connection Guidelines for Connecting the EPCS and EPCQ to Arria V Devices

For single- and multi-device AS configurations, the board trace length and loading between the supported EPCS and Arria V device must follow the recommendations provided. Table 1–8 lists the maximum trace length and loading for AS x1 and AS x4 configuration modes.

Table 1–8. Maximum Trace Length and Loading for AS x1 and x4 Configurations for Arria V Devices

Arria V Device AS Pins	Maximum Board Trace Length from the Arria V Device to the Serial Configuration Device for a 12.5/25/50 MHz Operation (Inches)	Maximum Board Trace Length from the Arria V Device to the Serial Configuration Device for a 100 MHz Operation (Inches)	Maximum Board Load (pF)
DCLK	10	6	15
DATA[3..0]	10	6	30
nCSO	10	6	30

AS Configuration Timing

For more information about the AS timing parameters, refer to the *Device Datasheet for Arria V Devices* chapter.

Estimating the AS Configuration Time

The AS configuration time is dominated by the time it takes to transfer data from the EPCS to the Arria V device. This serial interface is clocked by the Arria V DCLK.

You can estimate the minimum AS x1 mode configuration time by using the following equation:

.rbf Size × (minimum DCLK period / 1 bit per DCLK cycle) = estimated minimum configuration time.

You can estimate the minimum AS x4 mode configuration time by using the following equation:

.rbf Size × (minimum DCLK period / 4 bits per DCLK cycle) = estimated minimum configuration time.

Enabling compression reduces the amount of configuration data that is transmitted to the Arria V device, which also reduces the configuration time. Your configuration time is reduced based on the compression ratio. The compression ratio varies based on the design.

Programming the EPCS and EPCQ

EPCS and EPCQ are non-volatile, flash-memory-based devices. You can program these devices in-system using a USB-Blaster™, EthernetBlaster, EthernetBlaster II, or ByteBlaster™ II download cable. Alternatively, you can program the EPCS or EPCQ using a microprocessor with the SRunner software driver.

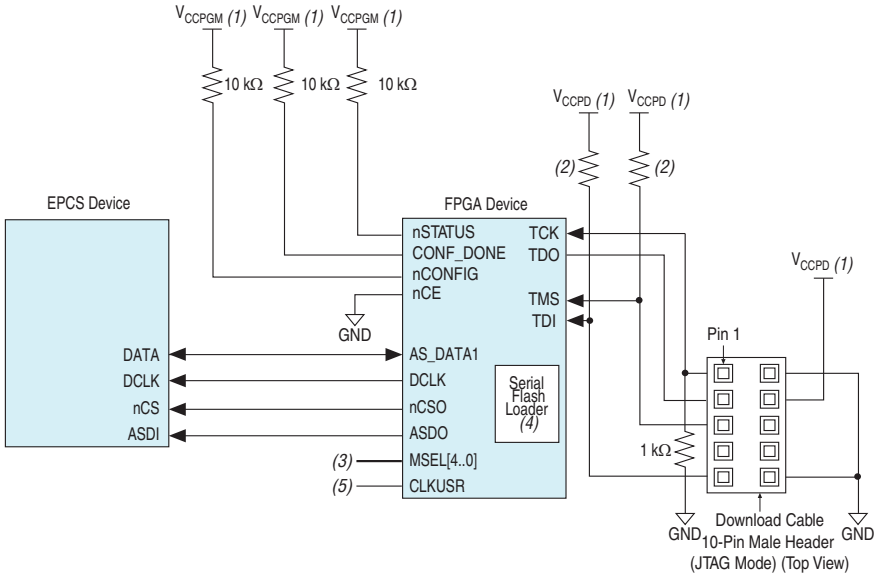
For more information about the SRunner software driver, refer to *AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming*.

In-system programming (ISP) offers you the option to program the EPCS or EPCQ either using an AS programming interface or a JTAG interface. Using the AS programming interface, the configuration data is programmed into the EPCS by the Quartus II software or any supported third-party software. Using the JTAG interface, an Altera IP called the serial flash loader (SFL) must be downloaded into the Arria V device to form a bridge between the JTAG interface and the EPCS or EPCQ. This allows the EPCS or EPCQ to be programmed directly using the JTAG interface.

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

Figure 1–57 shows the connection setup when programming the EPCS using the JTAG interface.

Figure 1–57. Connection Setup for Programming the EPCS Using the JTAG Interface



Notes to Figure 1–57:

(1) Connect the pull-up resistors to V_{CCPGM} at a 1.8-, 3.0-, or 3.3-V power supply. For more information about how to connect to V_{CCPD}, refer to the *Configuration, Design Security, and Remote System Upgrades in Arria V Devices* chapter.

(2) The resistor value can vary from 1 kΩ to 10 kΩ. Perform signal integrity analysis to select the resistor value for your setup.

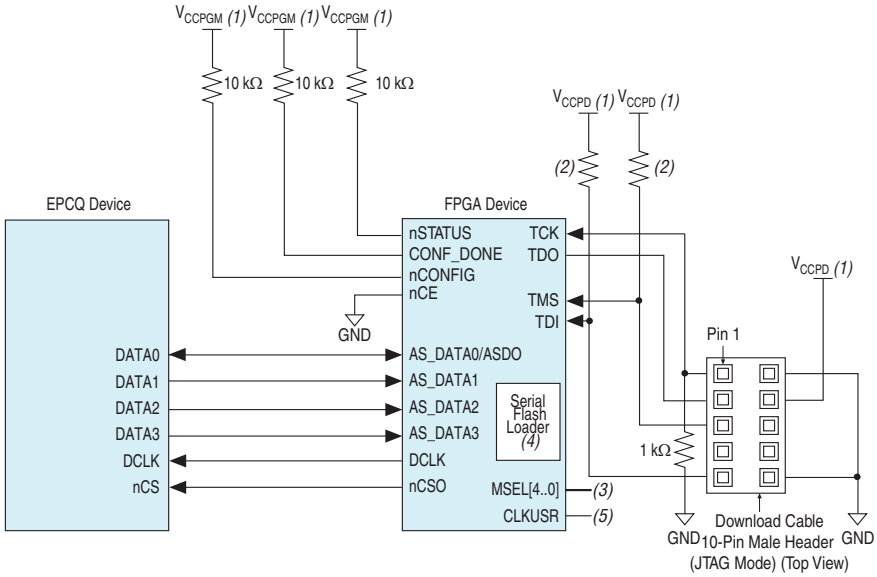
(3) The MSEL pin settings vary for different configuration voltage standards and POR delays.

(4) Instantiate SFL in your design to form a bridge between the EPCS and the Arria V device.

(5) Use the CLKUSR pin to supply the external clock source to drive DCLK during configuration.

Figure 1–58 shows the connection setup when programming the EPCQ using the JTAG interface.

Figure 1–58. Connection Setup for Programming the EPCQ Using the JTAG Interface



Notes to Figure 1–58:

(1) Connect the pull-up resistors to V_{CCPGM} at a 1.8-, 3.0-, or 3.3-V power supply. For more information about how to connect to V_{CCPD}, refer to the *Configuration, Design Security, and Remote System Upgrades in Arria V Devices* chapter.

(2) The resistor value can vary from 1 kΩ to 10 kΩ. Perform signal integrity analysis to select the resistor value for your setup.

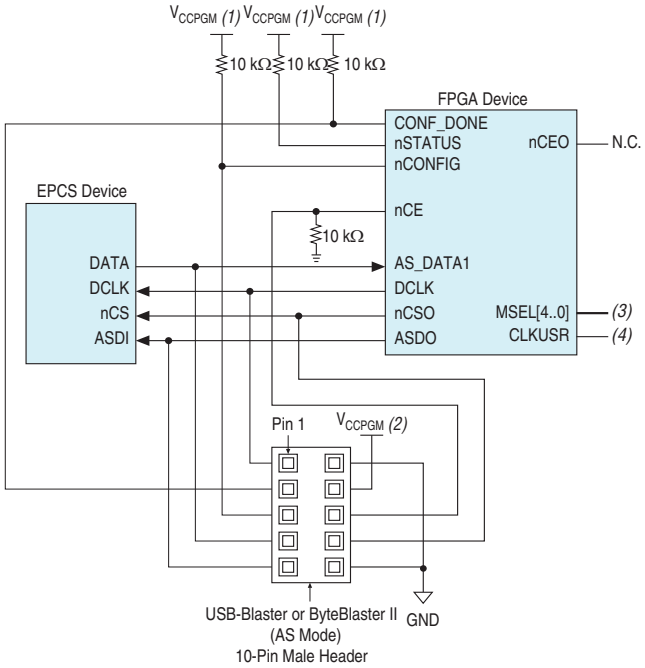
(3) The MSEL pin settings vary for different configuration voltage standards and POR delays.

(4) Instantiate SFL in your design to form a bridge between the EPCS and the Arria V device.

(5) Use the CLKUSR pin to supply the external clock source to drive DCLK during configuration.

Figure 1–59 shows the connection setup when programming the EPCS using the AS interface.

Figure 1–59. Connection Setup for Programming the EPCS Using the AS Interface



Notes to Figure 1–59:

(1) Connect the pull-up resistors to V_{CCPGM} at a 1.8-, 3.0-, or 3.3-V power supply.

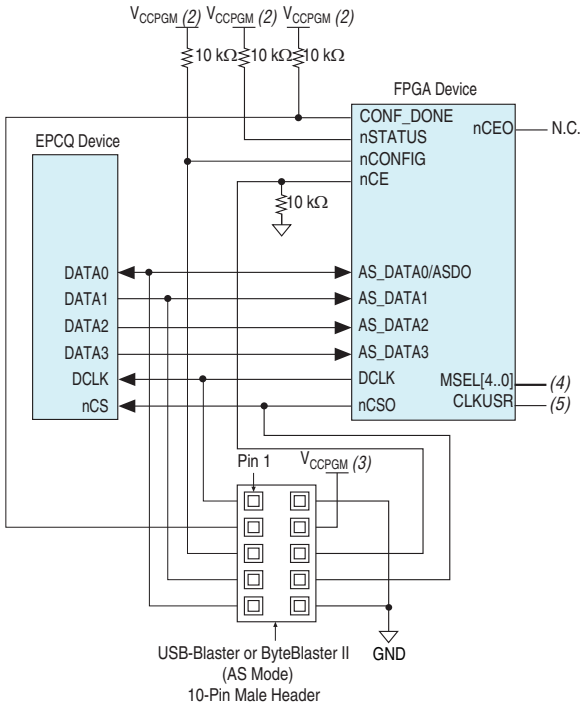
(2) Power up the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II cable's $V_{CC(TRGT)}$ to V_{CCPGM}.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delays.

(4) Use the CLKUSR pin to supply the external clock source to drive DCLK during configuration.

Figure 1–60 shows the connection setup when programming the EPCQ using the AS interface.

Figure 1–60. Connection Setup for Programming the EPCQ Using the AS Interface *(1)*



Notes to Figure 1–60:

(1) Using the AS header, the programmer transmits the operation commands and the configuration bits to the EPCQ serially on DATA0. This is equivalent to the programming operation for the EPCS.
(2) Connect the pull-up resistors to V_{CCPGM} at a 1.8-, 3.0-, or 3.3-V power supply.
(3) Power up the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II cable's $V_{CC(TRGT)}$ to V_{CCPGM}.
(4) The MSEL pin settings vary for different configuration voltage standards and POR delays.
(5) Use the CLKUSR pin to supply the external clock source to drive DCLK during configuration.

During the EPCS and EPCQ programming, the download cable disables the device access to the AS interface by driving the nCE pin high. The nCONFIG line is also pulled low to hold the Arria V device in the reset stage. After programming completes, the download cable releases nCE and nCONFIG, allowing the pull-down and pull-up resistors to drive the pin to GND and V_{CCPGM}, respectively.

During the EPCQ programming using the download cable, DATA0 carries the programming data, operation command, and address information from the download cable into the EPCQ. During the EPCQ verification using the download cable, DATA1 carries the programming data back to the download cable.

Passive Serial Configuration

You can perform PS configuration of Arria V devices using an external host such as a MAX II device, MAX V device, microprocessor, or a host PC. Therefore, the design that controls the configuration stages, such as fetching the data from flash memory and sending it to the device, must be stored in the external host.

The PFL megafunction in MAX II or MAX V devices provide an efficient method to program CFI flash memory devices through the JTAG interface. The PFL megafunction also acts as a controller to read configuration data from the flash memory device and configures the Arria V device.

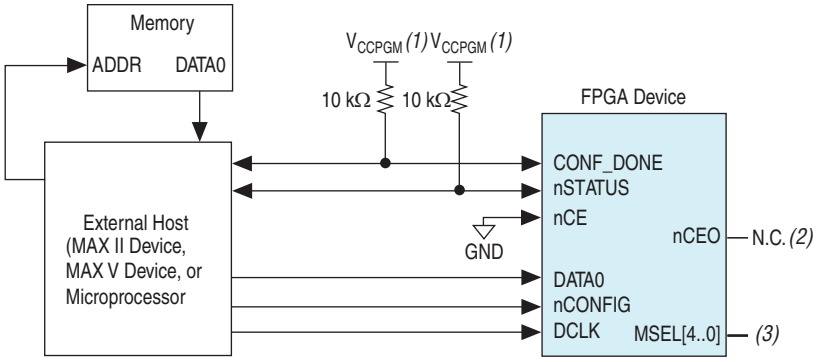
For more information about the PFL megafunction, refer to the *Parallel Flash Loader Megafunction User Guide*.

PS Configuration Using a MAX II Device, MAX V Device, or Microprocessor

The external host (a MAX II device, MAX V device, or microprocessor) reads configuration data from the storage devices, such as flash memory, and transfers it to the Arria V devices. You can store the configuration data in programmer object file (**.pof**), **.rbf**, **.hex**, or **.ttf** format. If you are using configuration data in **.rbf**, **.hex**, or **.ttf** format, you must send the LSB of each data byte first. For example, if the **.rbf** file contains the byte sequence
02 1B EE 01 FA, the serial data transmitted to the device must be 0100-0000 1101-1000 0111-0111 1000-0000 0101-1111.

Figure 1–61 shows the configuration interface connections between an Arria V device and a MAX II or MAX V device for single device configuration.

Figure 1–61. Single Device PS Configuration Using an External Host



Notes to Figure 1–61:

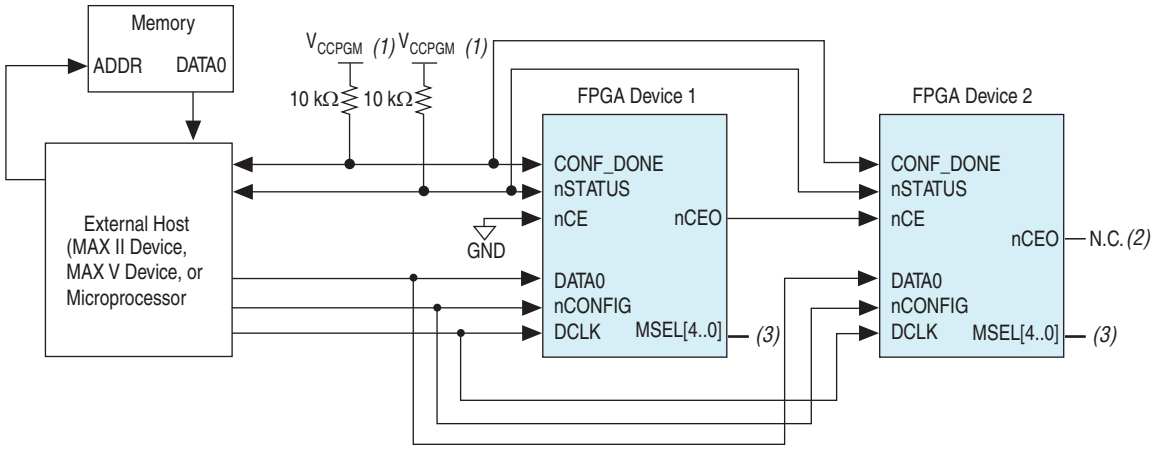
(1) Connect the resistor to a supply that provides an acceptable input signal for the Arria V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all the configuration system I/Os with V_{CCPGM}.

(2) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delays.

Figure 1–62 shows the PS multi-device configuration using an external host when all devices in the chain receive different sets of configuration data (multiple **.sof**s).

Figure 1–62. PS Multi-Device Configuration when Both Devices Receive Different Sets of Configuration Data



Notes to Figure 1–62:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Arria V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all the configuration system I/Os with V_{CCPGM}.

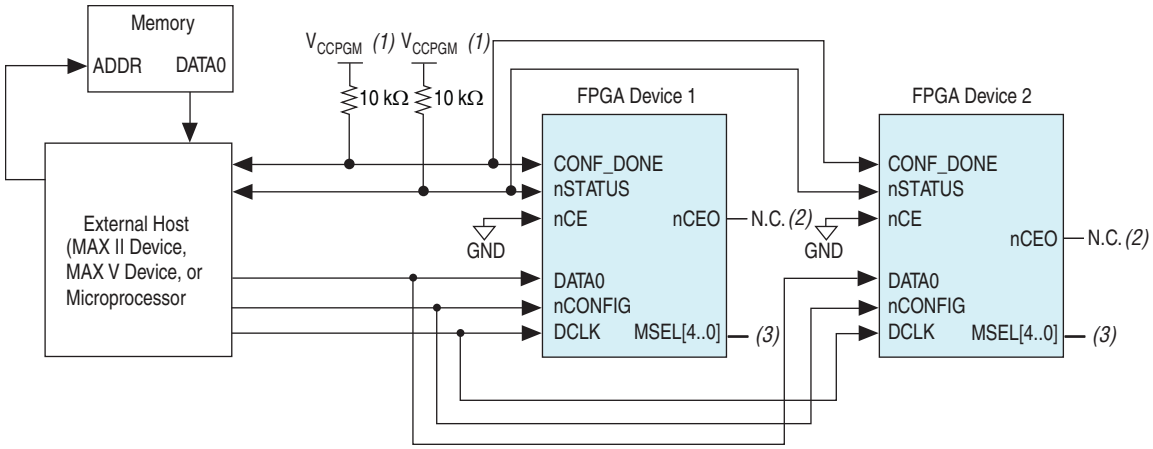
(2) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delays.

In Figure 1–62, 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 to the second device is transparent to the external host.

Figure 1–63 shows the PS multi-device configuration when all devices receive the same set of configuration data (single **.sof**).

Figure 1–63. PS Multi-Device Configuration When Both Devices Receive the Same Set of Configuration Data



Notes to Figure 1–63:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Arria V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all the configuration system I/Os with V_{CCPGM}.

(2) You can leave the nCEO pin unconnected or use it as a user I/O pin.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delays.

In Figure 1–63, because both nCE pins are tied to GND, both devices in the chain begin and complete the configuration and enter user mode at the same time.

To configure the PS multi-device with a single **.sof**, all Arria V devices in the chain must be in the same package and density.

PS Configuration Timing

For more information about the POR delay specifications, refer to the *Device Datasheet for Arria V Devices* chapter.

PS Configuration Using a Download Cable

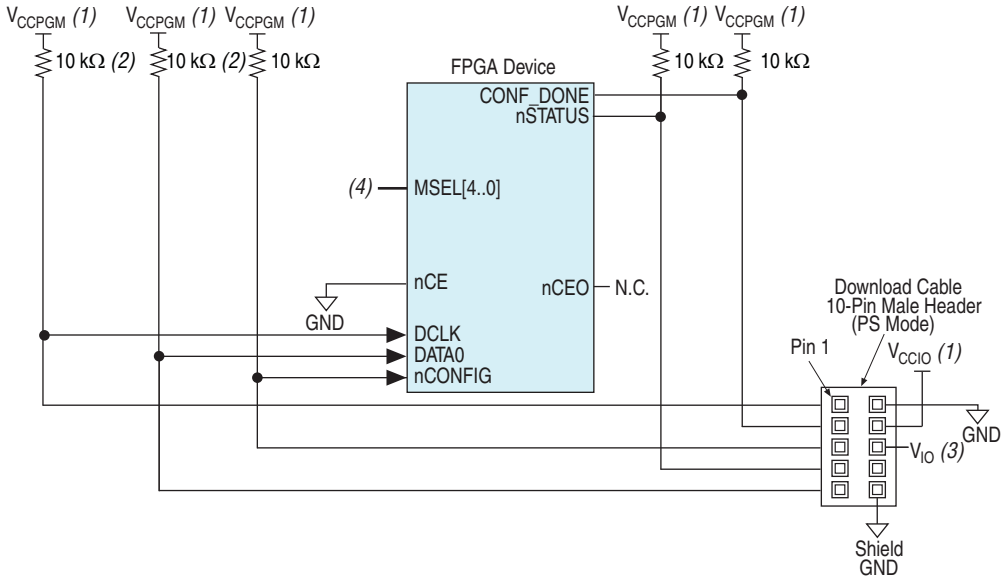
In this section, the generic term "download cable" includes the Altera USB-Blaster USB port download cable, ByteBlaster II parallel port download cable, EthernetBlaster download cable, and EthernetBlaster II download cable.

In a PS configuration with a download cable, a PC acts as a host to transfer data from a storage device to the Arria V 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.

If you turn on the CLKUSR option during PS configuration using a download cable and the Quartus II programmer, you do not have to provide a clock on the CLKUSR pin to initialize your device.

Figure 1–64 shows a PS configuration for Arria V devices using an Altera download cable.

Figure 1–64. PS Configuration Using an Altera Download Cable



Notes to Figure 1–64:

(1) Connect the pull-up resistor to the same supply voltage (V_{CCIO}) as the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster 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 MAX II device, MAX V device, or microprocessor, you do not need the pull-up resistors on DATA0 and DCLK.

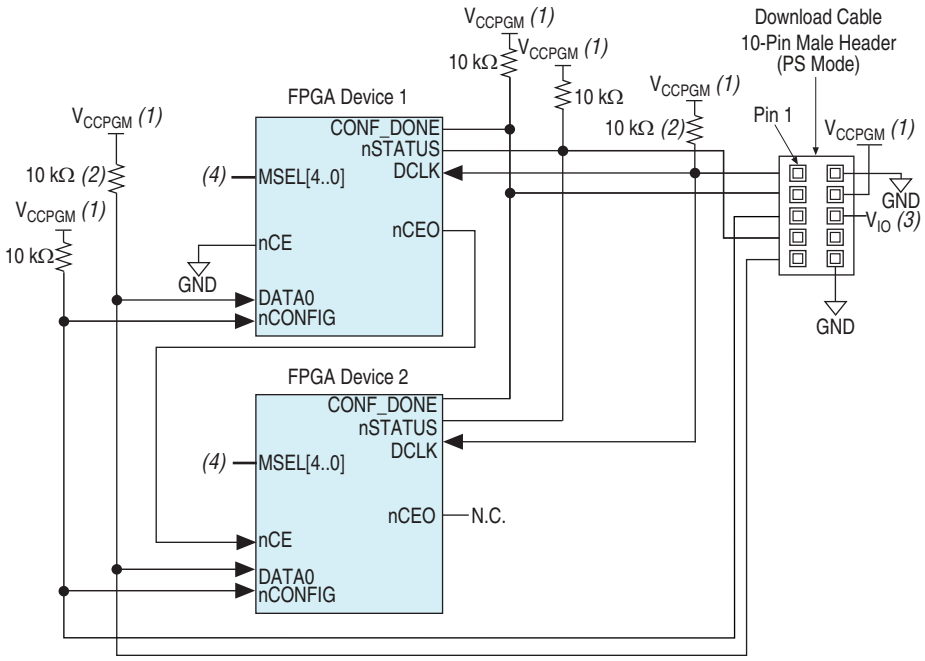
(3) In the USB-Blaster and ByteBlaster II cables, this pin is connected to nCE when you use it for AS programming; otherwise, it is a no connect.

(4) The MSEL pin settings vary for different configuration voltage standards and POR delays.

Multi-Device PS Configuration Using a Download Cable

Use a download cable to configure multiple Arria V devices. Figure 1–65 shows the multi-device PS configuration using an Altera download cable.

Figure 1–65. Multi-Device PS Configuration Using an Altera Download Cable



Notes to Figure 1–65:

(1) Connect the pull-up resistor to the same supply voltage (V_{CCIO}) as the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster 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) In the USB-Blaster and ByteBlaster II cables, this pin is connected to nCE when you use it for AS programming; otherwise, it is a no connect.

(4) The MSEL pin settings vary for different configuration voltage standards and POR delays.

In Figure 1–65, after the first device completes configuration, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE pins are connected to every device in the chain. As all the CONF_DONE and nSTATUS pins are tied together, all devices initialize and enter user mode at the same time. If any device detects an error, configuration stops for the entire chain and you must reconfigure all the devices. 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.

JTAG Configuration

Use the same JTAG interface specifically developed for boundary-scan testing (BST) to shift the configuration data into the device. The Quartus II software automatically generates a **.sof** that you can use for JTAG configuration with a download cable in the Quartus II software programmer.

For more information, refer to "JTAG Secure Mode" on page 1–91.

For more information about JTAG BST and the commands available using Arria V devices, refer to the following documents:

- *JTAG Boundary-Scan Testing in Arria V Devices* chapter
- Programming Support for Jam STAPL Language

In Arria V devices, JTAG instructions have precedence over any device configuration modes. JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt a JTAG configuration of Arria V devices during a PS configuration, the PS configuration is terminated and the JTAG configuration begins. All user I/O pins are tri-stated during the JTAG configuration.

You cannot use the Arria V decompression or design security features if you are configuring your Arria V device using the JTAG-based configuration.

For more information about `TDI`, `TDO`, `TMS`, and `TCK`, refer to "Device Configuration Pins" on page 1–76.

For more information about instructions to connect a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary-Scan Testing in Arria V Devices* chapter.

To configure a single device in a JTAG chain, the programming software places all the 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 by checking the state of `CONF_DONE` through the JTAG port.

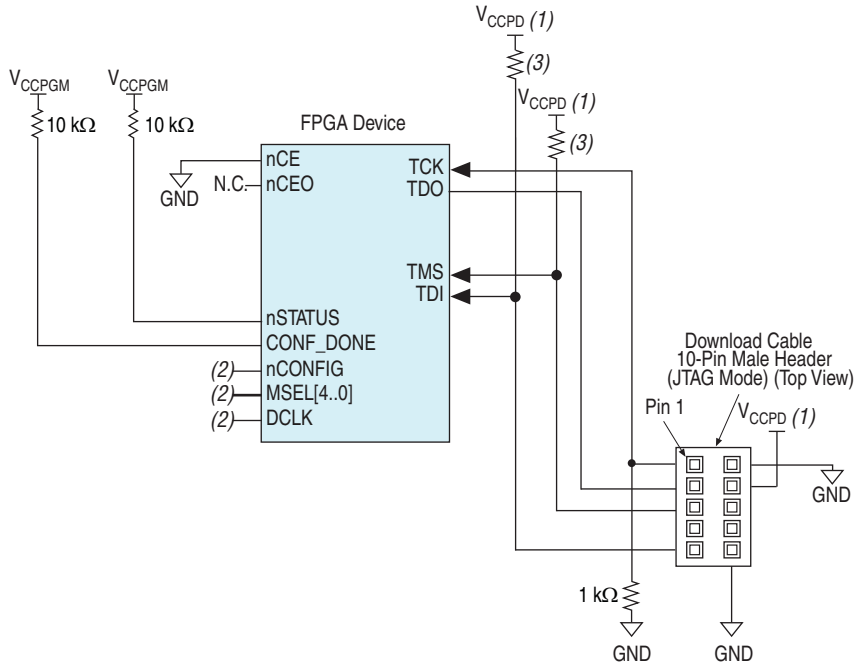
If `CONF_DONE` is low, the Quartus II software indicates that configuration has failed. If `CONF_DONE` is high, the software indicates that configuration was successful. After the configuration data is transmitted serially using the JTAG `TDI` port, the `TCK` port is clocked an additional 1,222 cycles to perform device initialization.

The chip-wide reset (`DEV_CLRn`) and chip-wide output enable (`DEV_OE`) pins on Arria V devices do not affect the JTAG boundary-scan or programming operation.

You can generate a JAM™ standard test and programming language (STAPL) format file (**.jam**) or JAM byte code file (**.jbc**) to use it with other third-party programmer tools. Alternatively, you can use JRunner with **.rbf** to program your device.

Figure 1–66 shows the JTAG configuration of a single Arria V device.

Figure 1–66. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 1–66:

(1) Connect the pull-up resistor V_{CCPD}. For more information about the V_{CCPD} requirement, refer to the *Configuration, Design Security, and Remote System Upgrades in Arria V Devices* chapter.

(2) If you only use the JTAG configuration, connect `nCONFIG` to V_{CCPGM} and `MSEL[4..0]` to GND. Pull `DCLK` either high or low, whichever is convenient on your board. If you are using JTAG in conjunction with another configuration scheme, connect `MSEL[4..0]`, `nCONFIG`, and `DCLK` based on the selected configuration scheme.

(3) The resistor value can vary from 1 kΩ to 10 kΩ. Perform signal integrity analysis to select the resistor value for your setup.

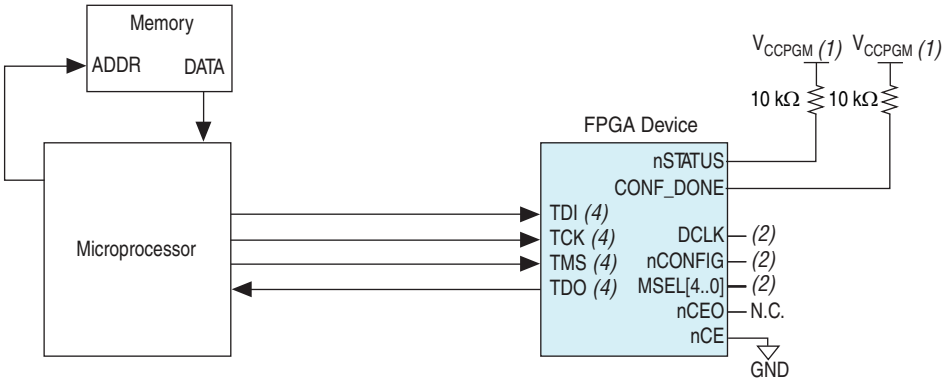
(4) You must connect `nCE` to GND or drive it low for successful JTAG configuration.

Alternatively, you can use a microprocessor to program the device through the JTAG interface. You can use JRunner as your software driver.

For more information about JRunner, refer to *AN 414: The JRunner Software Driver: An Embedded Solution for PLD JTAG Configuration*.

Figure 1–67 shows a JTAG configuration of an Arria V device using a microprocessor.

Figure 1–67. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 1–67:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all Arria V devices in the chain. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device.

(2) If you only use the JTAG configuration, connect `nCONFIG` to V_{CCPGM} and `MSEL[4..0]` to GND. Pull `DCLK` either high or low, whichever is convenient on your board. If you are using JTAG in conjunction with another configuration scheme, set `MSEL[4..0]` and tie `nCONFIG` and `DCLK` based on the selected configuration scheme.

(3) Connect `nCE` to GND or drive it low for successful JTAG configuration.

(4) The microprocessor must use the same I/O standard as V_{CCPD} to drive the JTAG pins.

CONFIG_IO Instruction

The `CONFIG_IO` instruction allows you to configure the I/O buffers using the JTAG port and when issued, interrupts configuration. This instruction allows you to perform board-level testing before configuring the Arria V 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 the JTAG interface or if you support an FPP, PS, or AS configuration scheme on your board, you can reconfigure the device by externally pulsing `nCONFIG` low. Alternatively, you can pulse `nCONFIG` low through the same JTAG interface using the `PULSE_NCONFIG` JTAG instruction.

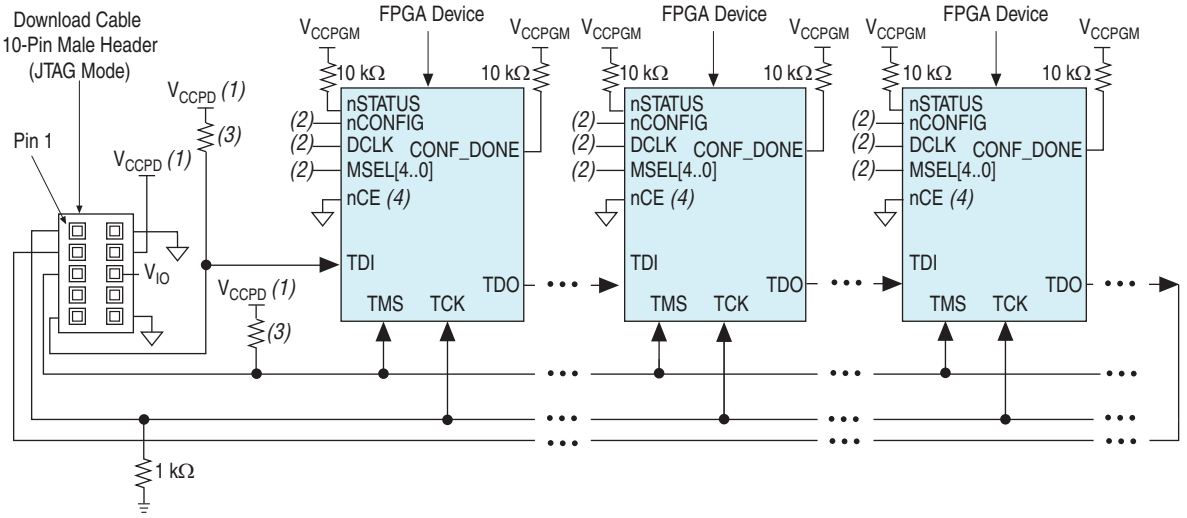
All JTAG instructions (except `BYPASS`, `IDCODE`, and `SAMPLE`) can be issued by first interrupting the configuration and reprogramming the I/O pins using the `CONFIG_IO` instruction.

Multi-Device JTAG Configuration

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. You can place other Altera devices that have JTAG support in the same JTAG chain for device programming.

JTAG-chain device programming is ideal when the system contains multiple devices or when testing your system using JTAG BST circuitry. Figure 1–68 shows a multi-device JTAG configuration.

Figure 1–68. JTAG Configuration of Multiple Devices Using a Download Cable



Notes to Figure 1–68:

(1) Connect the pull-up resistor V_{CCPD}. For more information about the V_{CCPD} requirement, refer to the *Configuration, Design Security, and Remote System Upgrades in Arria V Devices* chapter.

(2) If you only use the JTAG configuration, connect `nCONFIG` to V_{CCPGM} and `MSEL[4..0]` to GND. Pull `DCLK` either high or low, whichever is convenient on your board. If you are using JTAG in conjunction with another configuration scheme, connect `MSEL[4..0]`, `nCONFIG`, and `DCLK` based on the selected configuration scheme.

(3) The resistor value can vary from 1kΩ to 10 kΩ. Perform signal integrity analysis to select the resistor value for your setup.

(4) You must connect `nCE` to GND or drive it low for successful JTAG configuration.

If you want to use the JTAG multi-device configuration in conjunction with other configuration schemes, such as FPP, PS, or AS, tie `CONF_DONE`, `nSTATUS`, and `nCONFIG` together as recommended in the FPP, PS, or AS multi-device configuration schemes. Ensure that the JTAG chain is the same order as the multi-device FPP, PS, or AS configuration chain.

If you only use the JTAG configuration, Altera recommends connecting the circuitry as shown in Figure 1–67, where each of the `CONF_DONE` and `nSTATUS` signals are isolated to enable each device to enter user mode individually.

For more information about combining the JTAG configuration with other configuration schemes, refer to the *Combining Different Configuration Schemes* chapter in volume 2 of the *Configuration Handbook*.

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.

For more information about how to use the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II cables, refer to the following user guides:

- *USB-Blaster Download Cable User Guide*
- *ByteBlaster II Download Cable User Guide*
- *EthernetBlaster Communications Cable User Guide*
- *EthernetBlaster II Communications Cable User Guide*

Device Configuration Pins

Table 1–9 lists the configuration pin descriptions.

Table 1–9. Configuration Pins Description (Part 1 of 3)

Pin Name	Description
TDI (1)	Dedicated 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. This pin has an internal 25-kΩ pull-up that is always active.
TMS (1)	Dedicated test mode select. Input pin that provides the control signal to determine the transitions of the test access port (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. This pin has an internal 25-kΩ pull-up that is always active.
TCK (1)	Dedicated test clock input. Clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. It is expected that the clock input waveform have a nominal 50% duty cycle. This pin has an internal 25-kΩ pull-down that is always active.
TDO (1)	Dedicated 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. This pin is tri-stated if the data is not being shifted out of the device.
CLKUSR	Optional user-supplied clock input. It synchronizes the initialization of one or more devices. Enable this pin by turning on the **Enable user-supplied start-up clock (CLKUSR)** option under **Device and Pins Option**, in the **Configuration** panel of the Quartus II software.
CRC_ERROR	Optional output pin. Signals that the device have detected a cyclic redundancy check (CRC) error during user mode operation. This pin is an open-drain output pin by default and requires a 10 kΩ pull-up resistor. To use this pin as regular output, turn-off the **Enable Open-drain on CRC_ERROR pin** in **Device and Pins Option**, in the **Error Detection CRC** panel of the Quartus II software. The target device drives this pin low if there is no CRC error in the user mode operation. As an open-drain output, if a CRC error occurs, the device releases the pin which is then pulled high by the external pull-up resistor. Enable this pin by turning on the **Enable CRC error detection on CRC_ERROR pin** option in the Quartus II software. For more information about the CRC_ERROR pin, refer to the *SEU Mitigation in Arria V Devices* chapter.

Table 1–9. Configuration Pins Description (Part 2 of 3)

Pin Name	Description
CONF_DONE	Dedicated open-drain bidirectional pin. The target device drives the CONF_DONE pin low before and during configuration. After all the configuration data is received without error and the initialization cycle starts, the target device releases the CONF_DONE pin, which is then pulled high by the external pull-up resistor. The target device then reads the CONF_DONE pin status to ensure that the CONF_DONE pin is at logic high. After it is sensed high, the target device initializes and enters user mode. Driving CONF_DONE pin low after initialization completes does not affect the configured device.
DATA[15..5] *(3)*	Dual-purpose data input pins. For FPP x16, all pins are required for configuration. For FPP x8, only a subset of this pin is required for configuration. This pin is not required for PS or AS configuration. You can use the pins that are not required for configuration as regular I/Os. During configuration, byte-wide or word-wide data is received on these pins. The data received on DATA[15..5] are synchronized to DCLK.
DCLK	Dedicated bidirectional clock pin. In PS and FPP configuration schemes, 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. After configuration completes, drive DCLK high or low, whichever is convenient. In AS mode, DCLK is an output clock to clock the EPCS or EPCQ. Data is latched into the device on the falling edge of DCLK. After AS configuration completes, this pin is tri-stated with a weak pull-up resistor. Toggling this pin after configuration does not affect the configured device.
DEV_OE *(2)*	Optional input pin that allows you to override all tri-states on the device. When this pin is driven low, all the I/O pins are tri-stated. When this pin is driven high, all the 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 *(2)*	Optional input pin that allows you to override all clears on all the device registers. When this pin is driven low, all the registers are cleared. When this pin is driven high, all the registers behave as programmed. Enable this pin by turning on the **Enable device-wide reset (DEV_CLRn)** option in the Quartus II software.
INIT_DONE *(2)*	Optional output pin. Signals when the device has initialized and is in user mode. During the reset stage, after the device exits POR, and during the beginning of the configuration, the INIT_DONE pin is tri-stated and pulled high because of an external pull-up resistor. After you program the option bit to enable the INIT_DONE pin in the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization completes, 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.
MSEL[4..0]	Dedicated input pins. Five-bit configuration input that sets the Arria V device configuration scheme. For more information about the appropriate connections, refer to the *Configuration, Design Security, and Remote System Upgrades in Arria V Devices* chapter. The MSEL[4..0] pins have internal 25-kΩ pull-down resistors that are always active.

Table 1–9. Configuration Pins Description (Part 3 of 3)

Pin Name	Description
nSTATUS	Dedicated open-drain bidirectional pin. The device drives the nSTATUS pin low immediately after power-up and releases it after the device exits POR. During user mode and regular configuration, this pin is pulled high by an external 10-kΩ resistor. During configuration, the device drives this pin low to indicate an error during configuration. If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. You use this mechanism during multi-device configuration setup. If one of the devices in the chain has an error and pulls its nSTATUS pin low, it resets the entire chain. Driving the nSTATUS pin low after configuration and initialization completes does not affect the configured device.
nCE	Dedicated active-low chip enable input pin. Driving this pin low allows configuration. Drives the nCE pin low during configuration, initialization, and user mode for all single-device configurations. For a multi-device configuration, connect the nCE pin to GND or to nCEO of the previous device in the chain based on the recommendation in the respective configuration setup diagram.
nCEO (3)	Dual-purpose open-drain output pin. This pin drives low when device configuration completes. To use this pin to feed the next device's nCE pin in a multi-device chain, turn on the **Enable INIT_DONE output** option under **Device and Pins Option** in the **General** panel of the Quartus II software. In a single-device configuration, use this pin as a regular I/O. In a multi-device configuration, if this pin is not feeding nCE of the next device, you can use it as a regular I/O.
nCONFIG	Dedicated input pin. A low pulse on this pin during configuration and user mode causes the device to enter a reset state and tri-states all the I/O pins. A low-to-high logic starts a reconfiguration. During JTAG programming, the nCONFIG status is ignored.
AS_DATA0/ASDO/ DATA0	In a PS or FPP configuration, DATA0 is a dedicated input data pin. The data received on DATA0 is synchronized to DCLK. In AS x1 and AS x4 configuration schemes, AS_DATA0 and ASDO are dedicated bidirectional data pins. Use ASDO to send the operation command and addresses to the EPCS or EPCQ. During an AS x4 configuration, the data is received on AS_DATA0 and is synchronized to DCLK. This pin is tri-stated if AS configuration scheme is not selected. After the AS configuration completes, this pin is tri-stated with a weak pull-up resistor.
AS_DATA[3..1]/ DATA[3..1]	In an AS configuration, AS_DATA[3..1] are dedicated bidirectional data pins. During an AS configuration, the data are received on these pins and are synchronized to DCLK. In an FPP x8 or x16 configuration, the data received on DATA[3..1] are synchronized to DCLK. This pin is tri-stated if AS configuration scheme is not selected. After the AS configuration completes, this pin is tri-stated with a weak pull-up resistor.
nCSO/DATA4	In an AS configuration, nCSO is a dedicated output pin. nCSO drives the control signal from the Arria V device to the EPCS and EPCQ in AS mode. In an FPP configuration, the data received on DATA4 is synchronized to DCLK. This pin is tri-stated if AS configuration scheme is not selected. After the AS configuration completes, this pin is tri-stated with a weak pull-up resistor.

Notes to Table 1–9:

(1) If the JTAG interface is not required on the board, you can disable the JTAG circuitry by connecting this pin to logic high. For more information about instructions to connect a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary-Scan Testing in Arria V Devices* chapter. .

(2) This is a dual-purpose pin. This pin is available as an I/O if the associated option that enables this pin is turned off from the **Configuration** panel in the **Device and Pins Option** settings. For example, DEV_OE is available as a user I/O if the **Enable device-wide output enable** option is turned off.

(3) This is a dual-purpose pin. The state of this pin in the user mode depends on the **Dual-purpose Pins** settings in the **Device and Pins Option** settings.

Configuration Data Decompression

Arria V devices support configuration data decompression, which saves configuration memory space and may shorten configuration time. This feature allows you to store compressed configuration data in the configuration or other memory devices and transmit this compressed data to the Arria V devices. During configuration, the Arria V device decompresses the data in real time and programs its SRAM cells. The data decompression is done on-the-fly during configuration and does not require an additional processing time.

Preliminary data indicates that compression typically reduces the configuration data size by 30 to 55% based on the designs used. This reduces the storage requirement capacity for the flash memory. The decompression feature is supported in all configuration schemes except JTAG.

In the FPP configuration scheme, enabling the decompression feature requires a different DCLK-to-DATA[] ratio. For more information, refer to "Fast Passive Parallel Configuration" on page 1–52.

There are two ways to enable compression for Arria V data—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** to bring up the **Settings** dialog box.
2. After selecting your Arria V device, open the **Device and Pin Options** dialog box.
3. In the **Configuration settings** panel, turn on the **Generate compressed bitstreams** option.

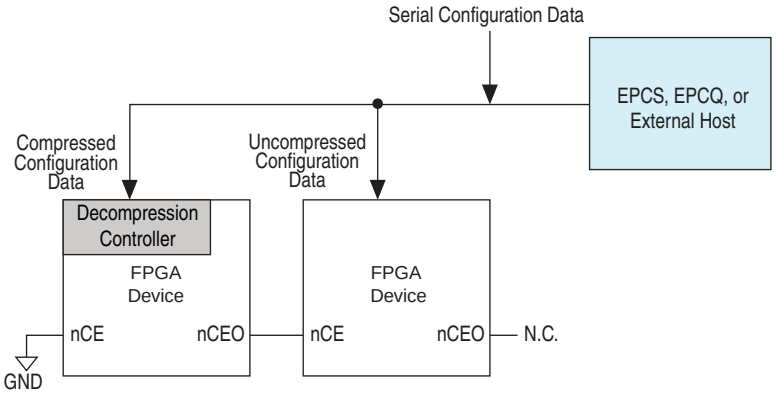
To enable compression when creating programming files from the **Convert Programming Files** window, follow these steps:

1. On the File menu, click **Convert Programming Files**.
2. Select the programming file type (**.pof**, **.sram**, **.hex**, **.hexout**, **.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 V 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.

If you are using a serial configuration scheme, AS x1 or PS, for a multi-device configuration, you can selectively enable the compression feature for each device in the chain. Figure 1–69 shows a chain of two Arria V devices. The first Arria V device has compression enabled and therefore receives compressed data from the external host. The second Arria V device has the compression feature disabled and receives uncompressed data.

For FPP configuration schemes, a combination of compressed and uncompressed configuration in the same multi-device chain is not allowed due to the difference of the DCLK-to-DATA[] ratio.

Figure 1–69. Compressed and Uncompressed Serial Configuration Data in the Same Configuration File *(1)*



Note to Figure 1–69:

(1) You can generate the configuration for this setup from the Convert Programming Files 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 V 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 V devices feature dedicated remote system upgrade circuitry. A soft logic (either the Nios® II embedded processor or user logic) implemented in an Arria V 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.

AS configuration schemes with the EPCS and EPCQ support remote system upgrades. You can also implement remote system upgrades in conjunction with advanced Arria V 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 EPCS and EPCQ currently supports 128 Mbits and 256 Mbits configuration data, respectively.

Remote system upgrades are supported only in single-device configurations.

To perform remote system upgrade in Arria V devices, follow these steps:

1. A Nios II processor (or user logic) implemented in the Arria V 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 errors that might occur during or after the reconfiguration cycle and provides error status information to your design.

Figure 1–70 shows these remote system upgrade steps.

Figure 1–70. Functional Diagram of the Arria V Remote System Upgrade Process

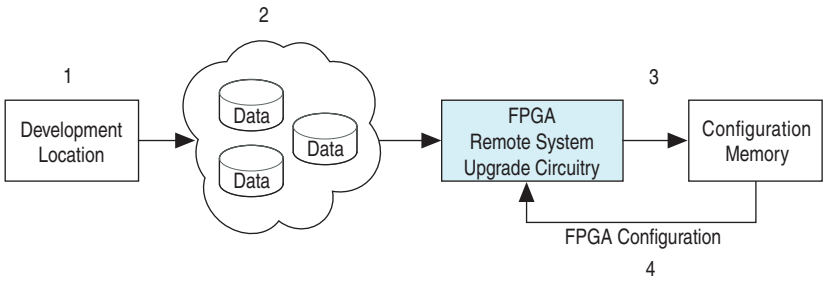
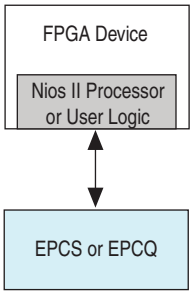


Figure 1–71 shows a block diagram for implementing a remote system upgrade with the Arria V AS configuration scheme.

Figure 1–71. Remote System Upgrade Block Diagram for the Arria V Device AS Configuration Scheme *(1)*



Note to Figure 1–71:

(1) You must set the mode select pins (`MSEL[4..0]`) to **AS mode** to use remote system upgrade in your system. The `MSEL` pin settings vary for different POR delays

Configuration Image Types

When performing a remote system upgrade, Arria V device configuration data are classified as factory configuration images or application configuration images. An image, also referred to as a configuration image, is a design loaded into the Arria V device that performs certain user-defined functions.

The factory image is a user-defined fall-back, or safe configuration, and initiates the reconfiguration to a new image with dedicated circuitry. Application images implement user-defined functionality in the target Arria V device. You may also include the default application image functionality in the factory image. Each Arria V device in your system requires one factory image and one or more application images.

Remote Update Mode

Arria V remote system upgrade circuitry only supports remote update mode. In remote update mode, Arria V 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.

When the Arria V device is first powered up in remote update mode, it loads the factory configuration located at the start address of `PGM[23..0]` = 24'h000000 in the EPCS and EPCQ. You must store your factory configuration image at this start address.

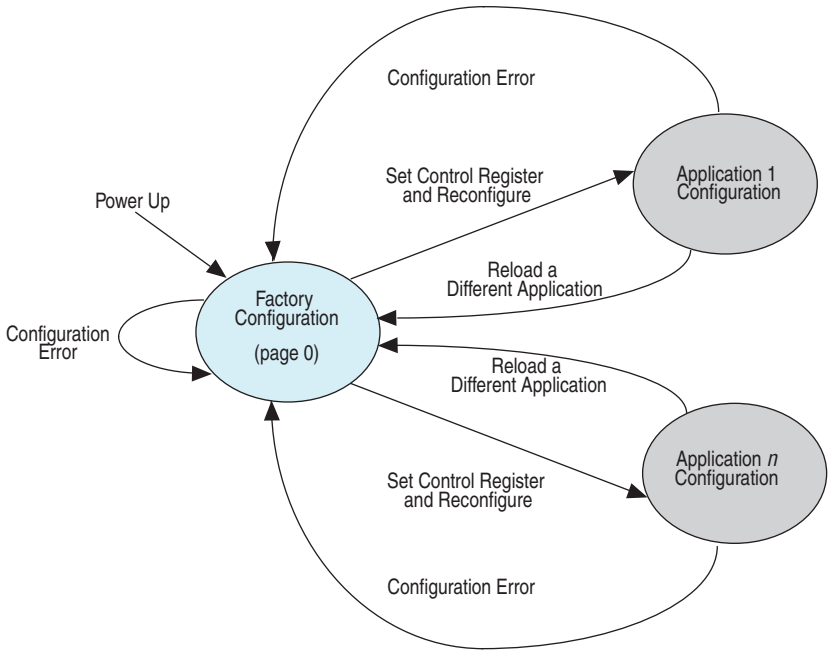
The application image start address can be at any EPCS or EPCQ sector boundary. Altera recommends using different sectors in the EPCS for two images.

The factory image is user-designed and contains soft logic to perform the following:

- 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 V device
- Enable or disable the user watchdog timer and load its time-out value
- Instruct the dedicated remote system upgrade circuitry to start a reconfiguration cycle

Figure 1–72 shows the transitions between the factory and application configurations in remote update mode.

Figure 1–72. Transitions Between Configurations in Remote Update Mode



After power up or a configuration error, the factory configuration image is loaded automatically. The system then decides to switch to the application configuration image or to stay in the factory configuration image. After the system decides to switch to an application configuration image, a reconfiguration is initiated through the remote system upgrade circuitry. In the application configuration image, the system may revert back to the factory configuration image after the following reconfiguration trigger conditions are met:

- nSTATUS driven low externally
- Configuration CRC error
- User watchdog timer time-out
- Core nCONFIG signal assertion
- External nCONFIG signal assertion

After the factory configuration image is reloaded, the 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.

Whenever the application configuration image is successfully loaded, the soft logic (Nios II processor or state machine and the remote communication interface) determine when the remote system update is arriving. When this occurs, 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.

Remote System Upgrade Using EPCQ 256

When you are using EPCQ 256, ensure that the application image address granularity is 32'h00000100. The **.rbf** size for your application image is 76,500 bytes longer than the numbers listed in the configuration **.rbf** size. You must take this extra space requirement into consideration when you try to fit multiple application images in the EPCQ 256 device.

For more information about the configuration **.rbf** size, refer to the *Configuration, Design Security, and Remote System Upgrades in Arria V Devices* chapter.

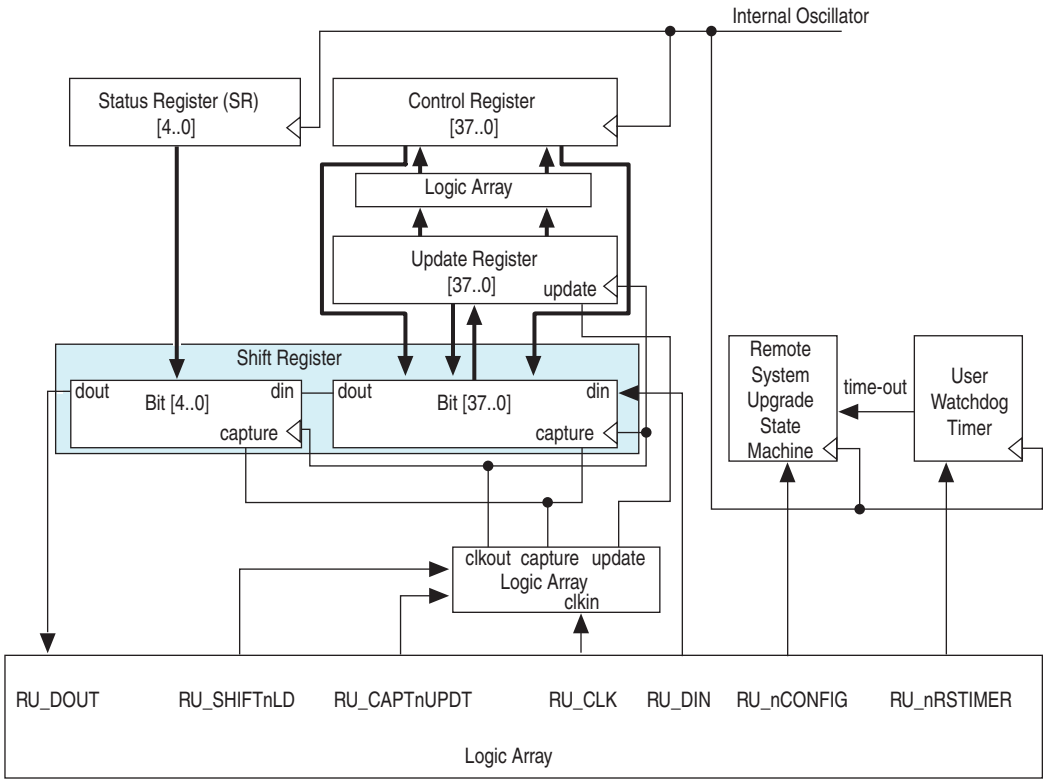
If you are not using the Quartus II software or SRunner software for EPCQ 256 programming, put your EPCQ 256 device into four-byte addressing mode before you program and configure your device.

Dedicated Remote System Upgrade Circuitry

This section describes the implementation of the Arria V dedicated remote system upgrade circuitry. The remote system upgrade circuitry is implemented in hard logic. This dedicated circuitry interfaces with the user-defined factory and application configurations implemented in the Arria V 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 1–73 shows the remote system upgrade circuitry.

Figure 1–73. Remote System Upgrade Circuitry *(1)*



Note to Figure 1–73:

(1) If you are using the ALTREMOTE_UPDATE megafunction, the megafunction controls the `RU_DOUT`, `RU_SHIFTnLD`, `RU_CAPTnUPDT`, `RU_CLK`, `RU_DIN`, `RU_nCONFIG`, and `RU_nRSTIMER` signals internally to perform all the related remote system upgrade operations.

For more information about remote system upgrade circuitry timing specifications, refer to the *Device Datasheet for Arria V Devices* chapter.

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 1–10 lists these registers.

Table 1–10. Remote System Upgrade Registers

Register	Description
Shift register	This register is accessible by the logic array and allows the update, status, and control registers to be written and sampled by user logic.
Control register	This register contains the current page address, user watchdog timer settings, and one bit specifying whether the current configuration is a factory configuration or an application configuration. During a read operation in an application configuration, this register is read into the shift register. When a reconfiguration cycle is initiated, the contents of the update register are written into the control register.
Update register	This register contains data similar to that in the control register. However, it can only be updated by the factory configuration by shifting data into the shift register and issuing an update operation. When a reconfiguration cycle is triggered by the factory configuration, the control register is updated with the contents of the update register. During a capture in a factory configuration, this register is read into the shift register.
Status register	The remote system upgrade circuitry writes this register on every reconfiguration to record the cause of the reconfiguration. The factory configuration uses this information to determine the appropriate action following a reconfiguration. During a capture cycle, this register is read into the shift register.

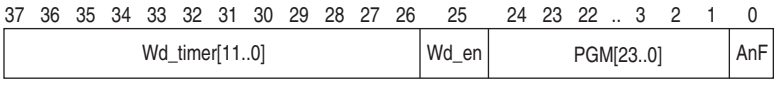
The remote system upgrade control and status registers are clocked by the 10-MHz internal oscillator (the same oscillator that controls the user watchdog timer). However, the remote system upgrade shift and update registers are clocked by the user clock input (RU_CLK).

Control Register

The 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. A factory configuration in remote update mode has write access to this register.

Figure 1–74 shows the control register bit positions. Table 1–11 lists the control register bits. In the figure, the numbers show the bit position of a setting in a register. For example, bit number 25 is the enable bit for the watchdog timer.

Figure 1–74. Remote System Upgrade Control Register



The application-not-factory (AnF) bit indicates whether the current configuration loaded in the Arria V 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 restricts the access to only read operations and enables the watchdog timer. The factory configuration design must set this bit high (1'b1) when updating the contents of the update register with the application page address and watchdog timer settings.

Table 1–11 lists the remote system upgrade control register contents.

Table 1–11. Remote System Upgrade Control Register Contents

Control Register Bit	Value (1)	Definition
AnF (2)	1'b0	Application not factory
PGM[23..0]	24'b0x000000	AS configuration start address (StAdd[23..0])
Wd_en	1'b0	User watchdog timer enable bit
Wd_timer[11..0]	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 1–11:

(1) This is the default value of the control register bit after the device exits POR and during reconfiguration back to the factory configuration image after reconfiguration trigger conditions.

(2) Factory configuration designs must set the AnF bit to **1'b1** before triggering the reconfiguration-to-application configuration image.

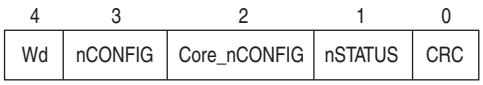
Status Register

The status register specifies the reconfiguration trigger condition. Figure 1–75 shows the status register content. The following list defines each bit:

- Bit 0—CRC error during application configuration
- Bit 1—nSTATUS assertion by an external device due to an error
- Bit 2—Arria V device logic array triggered a reconfiguration cycle, possibly after downloading a new application configuration image
- Bit 3—external configuration reset (nCONFIG) assertion
- Bit 4—user watchdog timer time-out

Figure 1–75 shows the contents of the status register. The numbers in the figure show the bit positions in a 5-bit register.

Figure 1–75. Remote System Upgrade Status Register (1)



Note to Figure 1–75:

(1) After the device exits POR and powers-up, the status register content is 5'b00000.

Remote System Upgrade State Machine

After power-up, the shift register, control register, and update registers are reset to the values listed in Table 1–10 on page 1–86, also known as POR reset values before the factory configuration image is loaded. In the factory configuration image, the user logic writes the AnF bit, page address, and watchdog timer settings for the next application configuration image 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 triggers a reconfiguration to the new application configuration image.

If there is an error during reconfiguration to the new application configuration image, the remote system upgrade state machine directs the system to reload a factory configuration image. The control and update registers are reset to POR reset values and the status register is updated with the error information. For example, if there is a CRC error during application configuration image configuration, the status register is updated with 5'b00001.

If there is no error during reconfiguration and the application configuration image is successfully loaded, the system stays in the application configuration image until another reconfiguration trigger condition occurs. This trigger condition can be a core nCONFIG assertion, external nCONFIG assertion, or the watchdog timer time-out error. If this happens, the control register and update registers are reset to POR reset values and the status register is updated with the error information. Consequently, the system proceeds to load the factory configuration image. Based on the status register content, the user logic in the factory configuration image then decides to stay in the factory configuration image or reload a new application reconfiguration image.

Read operations during factory configuration access the contents of the update register. The factory configuration image user logic uses this feature to verify that the page address and watchdog timer settings are written correctly. Read operations in application configurations access the contents of the control register. The user logic in the application configuration uses this information.

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 V device. This feature is automatically disabled in the factory configuration image and enabled in the application configuration image. Functional errors must not exist in the factory configuration because they are stored and validated during production and must never be updated remotely.

The user watchdog timer feature is automatically enabled in the application configuration image. If you do not wish to use this feature, disable it during the factory configuration image operation before triggering the reconfiguration to the application configuration image.

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 user watchdog timer internal oscillator.

For more information about the operating range of the user watchdog internal oscillator's frequency, refer to the *Device Datasheet for Arria V Devices* chapter.

The user watchdog timer begins counting after the application configuration enters device user mode. This timer must be periodically 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. This causes the device to reload the factory configuration image and update the status register to reflect the watchdog timer time-out error.

Enabling the Remote System Update Feature

You can enable remote update for Arria V 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 in the Quartus II software, follow these steps:

1. On the Assignments 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** panel.
4. From the Configuration scheme list, select **Active Serial x1** (you can also use **Configuration Device**).
5. From the Configuration mode list, select **Remote**.
6. Click **OK**.
7. In the **Settings** dialog box, click **OK**.

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 V 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. Using the megafunction block instead of creating your own logic saves design time and offers more efficient logic synthesis and device implementation.

For more information about the ALTREMOTE_UPDATE megafunction, refer to the *Remote System Upgrade (ALTREMOTE_UPDATE) Megafunction User Guide*.

Design Security

This section provides an overview of the design security features and their implementation in Arria V devices using the AES. It also describes the security modes available in Arria V devices that allow you to use these new features in your designs.

As Arria V 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 V design security supports the following features:

- Enhanced built-in AES decryption block to support 256-bit key industry-standard design security algorithm (FIPS-197 Certified)
- Volatile and non-volatile key programming support
- Secure operation mode for both volatile and non-volatile key through tamper protection bit setting
- Limited accessible JTAG instruction during power-up
- Supports board-level testing
- Supports in-socket key programming for non-volatile key
- Available in all configuration schemes except JTAG
- Supports both remote system upgrades and decompression features

You can use the design security feature with or without the remote system upgrades or decompression features.

The Arria V design security feature provides the following security protection for your designs:

- Security against copying—the security key is securely stored in the Arria V device and cannot be read out through any interface. In addition, as configuration file read-back is not supported in Arria V 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 V configuration file formats are proprietary and the file contains millions of bits that require specific decryption. In addition, the Arria V devices are manufactured on the most advanced 28-nm process technology, making this process very difficult.
- Security against tampering—Arria V devices always power-up with limited accessible JTAG instructions. This disables tamper attempts through the JTAG interface. You can enhance this security feature with the tamper protection bit setting. After the tamper protection bit is set, the Arria V device can only accept configuration files encrypted with the same key. Additionally, programming through the JTAG interface is blocked. This prevents any attempts to tamper with the device from both the JTAG interface and the configuration interface.

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 V device first decrypts and then decompresses the configuration file.

When you use design security with Arria V devices in an FPP configuration scheme, it requires a different DCLK-to-DATA[] ratio. For more information, refer to "Fast Passive Parallel Configuration" on page 1–52.

JTAG Secure Mode

When you enable the tamper-protection bit, Arria V devices are in JTAG secure mode after power-up. During JTAG secure mode, many JTAG instructions are disabled. Arria V devices only allow mandatory JTAG 1149.1 instructions to be exercised. These instructions are SAMPLE/PRELOAD, BYPASS, EXTEST, and optional instructions such as IDCODE and SHIFT_EDERROR_REG.

To enable the access of other JTAG instructions such as USERCODE, HIGHZ, CLAMP, PULSE_NCONFIG, and CONFIG_IO, you must issue the UNLOCK instruction to deactivate the JTAG secure mode. You can issue the LOCK instruction to put the device back into JTAG secure mode. Both the LOCK and UNLOCK instructions can only be issued during user mode.

For more information about JTAG binary instruction code related to the LOCK and UNLOCK instructions, refer to the *JTAG Boundary-Scan Testing in Arria V Devices* chapter.

Security Key Types

Arria V devices offer two types of keys—volatile and non-volatile. Table 1–12 lists the differences between the volatile key and non-volatile key.

Table 1–12. Security Key Types

Key Types	Key Programmability	Power Supply for Key Storage	Programming Method
Volatile Key	■ Reprogrammable ■ Erasable	Required external battery, V_{CCBAT} (1)	On-board
Non-volatile Key	One-time programming	Does not require an external battery	On-board and in-socket programming (2)

Notes to Table 1–12:

(1) 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_{CCPGM}. V_{CCBAT} continuously supplies power to the volatile register regardless of the on-chip supply condition.

(2) In-socket programming is offered through third-party vendors.

Both non-volatile and volatile key programming offers protection from reverse engineering and copying. If you set the tamper-protection bit, the design is also protected from tampering.

Perform key programming through the JTAG interface. Also, ensure that the nSTATUS pin is released high before any key-programming attempts.

For more information about battery specifications, refer to the *Device Datasheet for Arria V Devices* chapter.

For more information about the V_{CCBAT} pin connection recommendations, refer to the *Arria V Device Family Pin Connection Guidelines*.

Security Modes

Table 1–13 lists the security modes available in Arria V devices.

Table 1–13. Supported Security Modes

Security Mode	Tamper protection Bit Setting	Device Accepts Unencrypted File	Device Accepts Encrypted File	Security Level
No-key	—	Yes	No	—
Volatile Key	—	Yes (1)	Yes	Secure
Volatile Key with Tamper Protection Bit Set	Set (2)	No	Yes	Secure with tamper resistant
Non-volatile Key	—	Yes (1)	Yes	Secure
Non-volatile Key with Tamper Protection Bit Set	Set (2)	No	Yes	Secure with tamper resistant

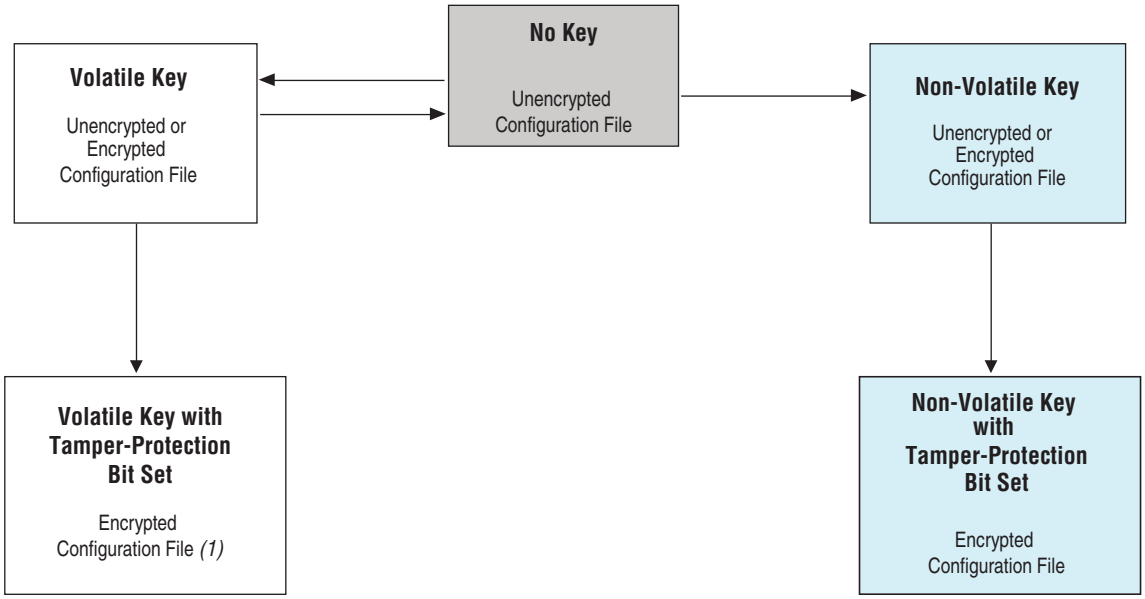
Notes to Table 1–13:

(1) Use the unencrypted configuration bitstream support only for board-level testing.

(2) Enabling the tamper protection bit disables test mode in Arria V devices and disables programming through the JTAG interface. This process is irreversible and prevents Altera from carry-out failure analysis. Contact Altera Technical Support to enable the tamper protection bit.

Figure 1–76 shows the sequence of the security modes available in Arria V devices.

Figure 1–76. Arria V Security Modes—Sequence and Restrictions



Note to Figure 1–76:

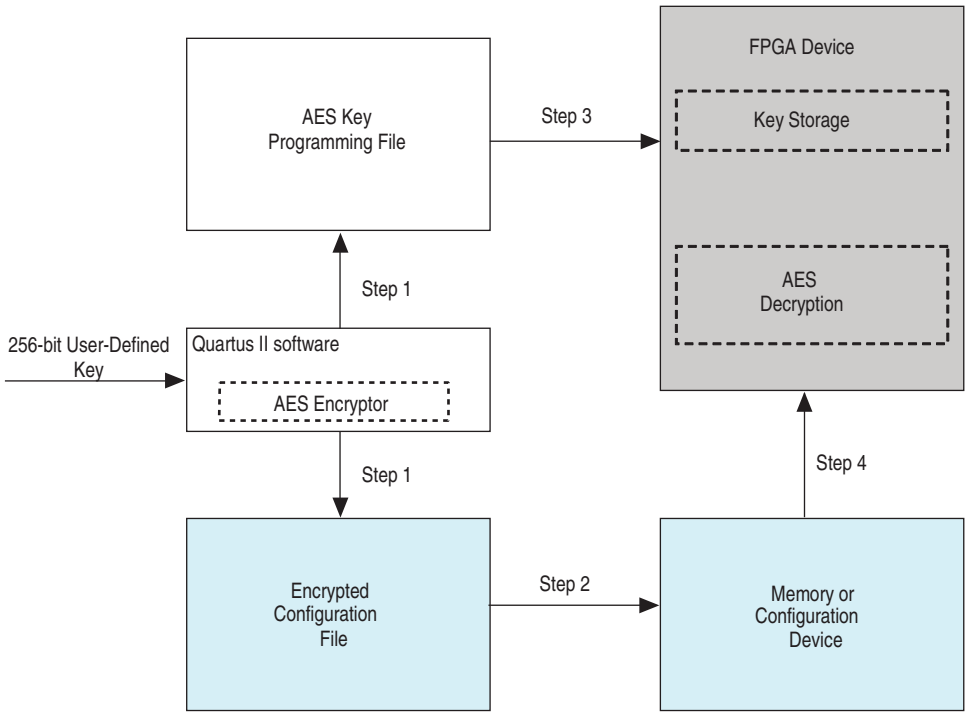
(1) Arria V devices do not accept the encrypted configuration file if the volatile key is erased. You must use the volatile key without tamper-protection bit set to reprogram the key if the volatile key in Arria V device is erased.

Design Security Implementation Steps

Arria V devices are SRAM-based devices. To provide design security, Arria V devices require a 256-bit security key for configuration bitstream design security. To carry out secure configuration, follow these steps (Figure 1–77):

1. The Quartus II software generates the design security key programming file and encrypts the configuration data using the user-defined 256-bit key.
2. Store the encrypted configuration file in the external memory.
3. Program the AES key programming file into the Arria V device through a JTAG interface.
4. Configure the Arria V device. At system power-up, the external memory device sends the encrypted configuration file to the Arria V device.

Figure 1–77. Design Security Implementation Steps



SEU Mitigation

This section describes the basic information of how to activate and use the error detection cyclic redundancy check (CRC) feature in the Arria Vdevice.

Use the information in this section in conjunction with the *SEU Mitigation in Arria V Devices* chapter.

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 V 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 16-bit configuration CRC is stored in the configuration data and contains the CRC value for each data frame.

During configuration, the Arria V device calculates the 16-bit configuration CRC value based on the frame of data that is received and compares it against the pre-calculated 16-bit configuration CRC value in the data stream. Configuration continues until the device detects an error or the configuration is complete.

User Mode Error Detection and Correction

Arria V devices offer on-chip circuitry for automated single event upset (SEU) detection. Some applications require the device to operate error-free in high-neutron flux environments that require periodic checks to ensure continued data integrity. The error detection CRC feature ensures the data reliability and is one of the best options for mitigating SEU.

You can implement the error detection CRC feature using the existing circuitry in Arria V devices, eliminating the need for external logic. Arria V devices have built-in error detection circuitry to detect data corruption by soft errors in the configuration RAM (CRAM) cells. This feature allows all CRAM contents to be read and verified to match a configuration-computed 32-bit error detection CRC value. Soft errors are changes in a CRAM's bit state due to an ionizing particle.

To enable the error detection process when the device transitions into user mode, turn on the **Enable Error Detection CRC_ERROR pin** option on the **Error Detection CRC** page of the **Device and Pin Options** dialog box in the Quartus II software.

The error detection capability continuously calculates the 32-bit error detection CRC value of the configured CRAM bits and compares it with the configuration-computed 32-bit error detection CRC value. The 32-bit error detection CRC value is computed during the configuration stage. The error detection circuitry generates 32 CRC check bits per frame and then stores them in the CRAM. If the 32-bit error detection CRC values match, there is no error in the current configuration CRAM bits. The process of error detection continues until the device is reset by setting the nCONFIG signal low.

A single 32-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 32-bit signature is 0x00000000. If the error detection circuitry detects no CRAM bit errors in a frame, the CRC_ERROR output signal is set to low. If the circuitry detects a CRAM bit error in a frame, the resulting signature is non-zero and the error detection circuitry starts searching for the error bit location.

The error detection circuitry in Arria V 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-, double-, triple-, quadruple-, and quintuple-bit errors. The probability of more than five CRAM bits being flipped by an SEU is very low. In general, the probability of detection for all error patterns is 99.9999%.

The error detection circuitry reports the bit location and determines the type of error for single-bit errors or 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 JTAG and the core interface. Before the error detection circuitry detects the next error in another frame, you must shift out the erroneous bits from the error message register (EMR) with the SHIFT_EDERROR_REG JTAG instruction or with 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.

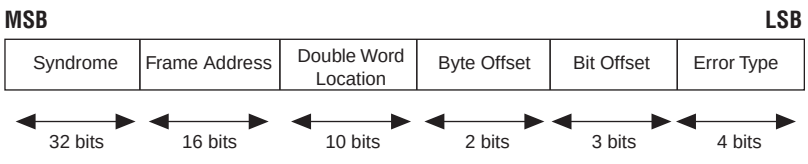
Table 1–14 lists the instruction code for the SHIFT_EDERROR_REG JTAG instruction.

Table 1–14. SHIFT_EDERROR_REG JTAG Instruction

JTAG Instruction	Instruction Code	Description
SHIFT_EDERROR_REG	00 0001 0111	The JTAG instruction connects the EMR to the JTAG pin in the error detection block between the TDI and TDO pins.

Figure 1–78 shows the content of the EMR.

Figure 1–78. Error Message Register



The type of error is identified in the first four bits of the EMR. Table 1–15 lists the error types represented in the EMR.

Table 1–15. Error Type in Error Message Register

Error Type				Description
Bit 3	Bit 2	Bit 1	Bit 0	
0	0	0	0	No CRC error.
0	0	0	1	Location of a single-bit error is identified.
0	0	1	0	Location of a double-adjacent error is identified.
1	1	1	1	There is more than one error.
Others				Reserved.

For more information about the timing requirement to shift out error information from the EMR, refer to "Error Detection Timing" in the *SEU Mitigation in Arria V Devices* chapter.

The error detection circuitry continues to calculate the 32-bit error detection CRC value and 32-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 appropriate actions if a CRC error occurs.

The error detection circuitry in Arria V devices uses a 32-bit CRC-ANSI standard (32-bit polynomial) as the CRC generator. The computed 32-bit CRC signature for each frame is stored in the CRAM. The total storage size is 32 (number of bits per frame) x the number of frames.

The Arria V device error detection CRC feature does not check memory blocks and I/O buffers. Thus, the CRC_ERROR signal may stay solid high or low, depending on the error status of the previously checked CRAM frame. The I/O buffers are not verified during error detection because these bits use flipflops as storage elements that are more resistant to soft errors when compared with CRAM cells. Memory logic array blocks (MLABs) and M10K memory blocks support parity bits that are used to check the contents of memory blocks for any error.

In Arria V devices, in addition to the error detection capability, the error detection circuitry also supports error correction or internal scrubbing, which internally corrects soft errors that have been detected. This is done on a per-frame basis. If you enable internal scrubbing, the device corrects single-bit errors or double-adjacent errors in the CRAM bits while the device is still running.

To test the capability of the error detection block, use the EDERROR_INJECT JTAG instruction. This instruction can change the content of the 47-bit JTAG fault injection register that is used for error injection in the Arria V devices.

You can execute the EDERROR_INJECT JTAG instruction only when the device is in user mode.

Table 1–16 lists the EDERROR_INJECT JTAG instruction.

Table 1–16. EDERROR_INJECT JTAG Instruction

JTAG Instruction	Instruction Code	Description
EDERROR_INJECT	00 0001 0101	This instruction controls the 47-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 reconfiguring the device. You can then switch to the CRC circuit to check for real errors induced by an SEU.

You can introduce a single error or double errors adjacent to each other to the configuration memory. This provides an extra way to facilitate design verification and system fault tolerance characterization. Use the JTAG fault injection register with the EDERROR_INJECT JTAG instruction to flip the readback bits. The Arria V device is then forced into the error test mode. Altera recommends reconfiguring the device after the test is completed.

You can introduce error injection only in the first data frame. However, 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 1–99.

Table 1–17 lists the implementation of the fault injection register and describes the error injection.

Table 1–17. Fault Injection Register

Description	Bit[46..43]					Bit[42..32]	Bit[31..0]
	Error Type				Error Injection Type	Byte Location of the Injected Error	Error Byte Value
	Bit[46]	Bit[45]	Bit[44]	Bit[43]			
Content	0	0	0	0	No 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.
	0	0	0	1	Single error injection		
	0	0	1	0	Double-adjacent error injection		
	Others				Reserved		

Error Detection Pin Description

Table 1–18 describes the CRC_ERROR pin.

Table 1–18. CRC_ERROR Pin Description

Pin Name	Pin Type	Description
CRC_ERROR	I/O, output, or output open-drain	This is an active high signal indicating that the error detection circuit has detected errors in the configuration CRAM bits. This is an optional pin and is used if you enable the error detection CRC circuit. If you disable the error detection CRC circuit, it becomes a user I/O pin. When using the WYSIWYG function, you can route the crcerror port from the WYSIWYG atom to the dedicated CRC_ERROR pin or any user I/O. To route the crcerror port to a user I/O, insert a D-type flipflop in between the crcerror port and the I/O.

Error Detection Block

The error detection block contains the logic necessary to calculate the 32-bit error detection CRC signature for the configuration CRAM bits in the Arria V 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 1–19 lists the two types of CRC detection that check the configuration bits.

Table 1–19. Two Types of CRC Detection

User Mode CRC Error Detection	Configuration CRC Error Detection
■ This is the CRAM error checking ability (32-bit error detection CRC) during user mode for use by the CRC_ERROR pin. ■ For each frame of data, the pre-calculated 32-bit error detection CRC enters the CRC circuit at the end of the frame data and determines whether there is an error or not. ■ If an error occurs, the search engine finds the location of the error. ■ The error messages can be shifted out through the JTAG instruction or core interface logics while the error detection block continues running. ■ The JTAG interface reads out the 32-bit error detection CRC result for the first frame and also shifts the 32-bit error detection CRC bits to the 32-bit error detection CRC storage registers for test purposes. ■ You can deliberately introduce a single error or double-adjacent errors to the configuration memory for testing and design verification.	■ This is the 16-bit configuration CRC that is embedded in every configuration data frame. ■ During configuration, after a frame of data is loaded into the Arria V device, the pre-computed configuration CRC is shifted into the CRC circuitry. ■ At the same time, the 16-bit 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 as 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.

Error Detection Registers

This section focuses on the user mode CRC error detection.

There is one set of 32-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 1–79 shows the error detection circuitry, syndrome registers, and error injection block.

Figure 1–79. Error Detection Circuitry, Syndrome Register, and Error Injection Block

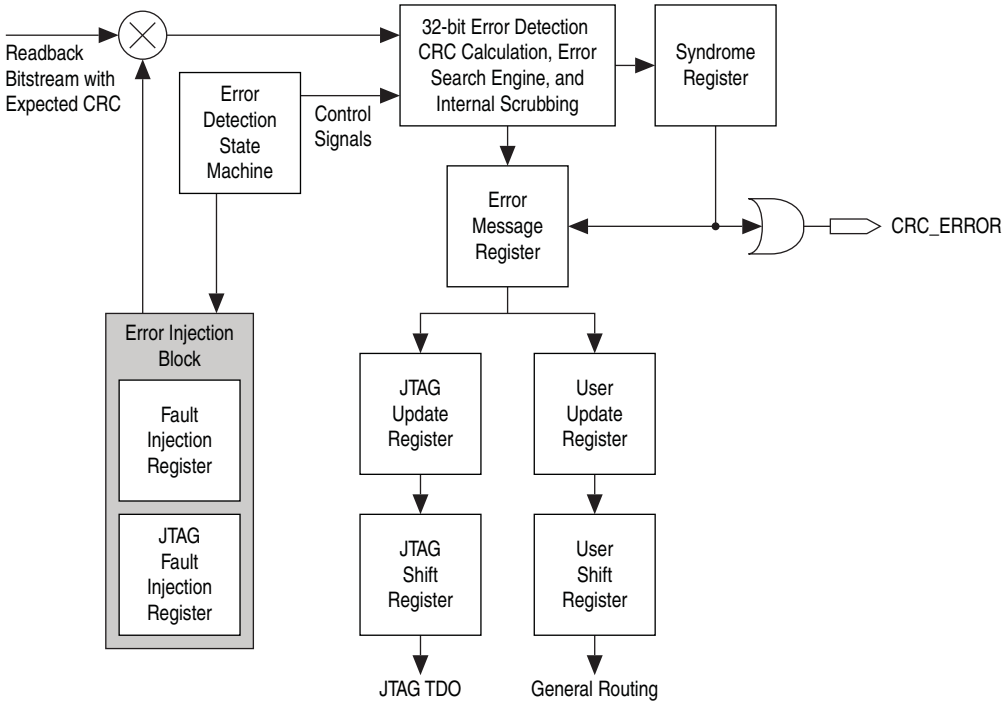


Table 1–20 lists the registers shown in Figure 1–79.

Table 1–20. Error Detection Registers (Part 1 of 2)

Register	Description
Syndrome Register	This 32-bit 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 67-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 67-bit register is automatically updated with the contents of the EMR one cycle after this 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.

Table 1–20. Error Detection Registers (Part 2 of 2)

Register	Description
User Update Register	This 67-bit register is automatically updated with the contents of the EMR one cycle after this 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 exactly the same time that the user shift register is reading its contents.
JTAG Shift Register	This 67-bit 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 67-bit 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 47-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 47-bit register when it is updated.

Software Support

The Quartus II software, starting with version 11.1, supports the error detection CRC feature for Arria V devices. Enable this feature to generate the CRC_ERROR output signal to the optional, dual-purpose CRC_ERROR pin.

To enable the error detection feature using CRC, follow these steps:

1. Open the Quartus II software and load a project that uses an Arria V device.
2. On the Assignments menu, click **Device**.
3. In the **Device** dialog box, click **Device and Pin Options**.
4. In the **Category** list, click **Error Detection CRC**.
5. Turn on **Enable Error Detection CRC_ERROR pin**.
6. To set the CRC_ERROR pin as output open-drain, turn on **Enable open drain on CRC_ERROR pin**. To set this pin as an output, turn this option off.
7. To enable or disable the on-chip error correction feature, turn on or off **Enable internal scrubbing**.
8. In the **Divide error check frequency by** list, select a valid divisor.
9. Click **OK**.

Recovering From CRC Errors

Although soft errors are uncommon in Altera devices, some high-reliability applications may require that your designs account for these errors.

The system that host the Arria V device must control device reconfiguration. After an error is detected on the CRC_ERROR pin, strobe the nCONFIG signal low to direct the system to perform reconfiguration at a safe time. After the device reconfiguration rewrites the data bit with the correct value, the device functions correctly.

JTAG Boundary-Scan Testing

This section describes the basic information of the JTAG boundary-scan testing in Arria V devices.

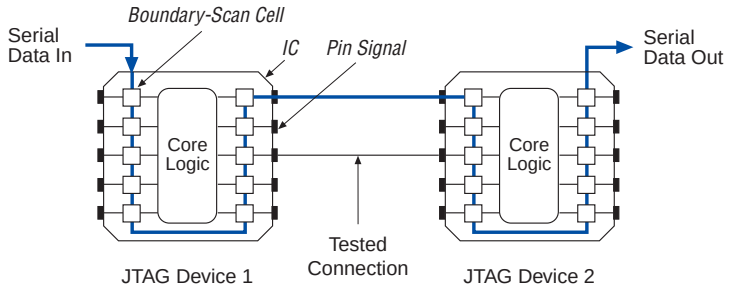
Use the information in this section in conjunction with the *JTAG Boundary-Scan Testing in Arria V Devices* chapter.

IEEE Std. 1149.1 BST Architecture

This boundary-scan test (BST) architecture tests pin connections without using physical test probes and captures functional data while a device is operating normally. Boundary-scan cells (BSCs) in a device can force signals onto pins or capture data from pins or logic array signals. Forced test data is serially shifted into the BSCs. Captured data is serially shifted out of and externally compared with the expected results.

Figure 1–80 shows the BST architecture.

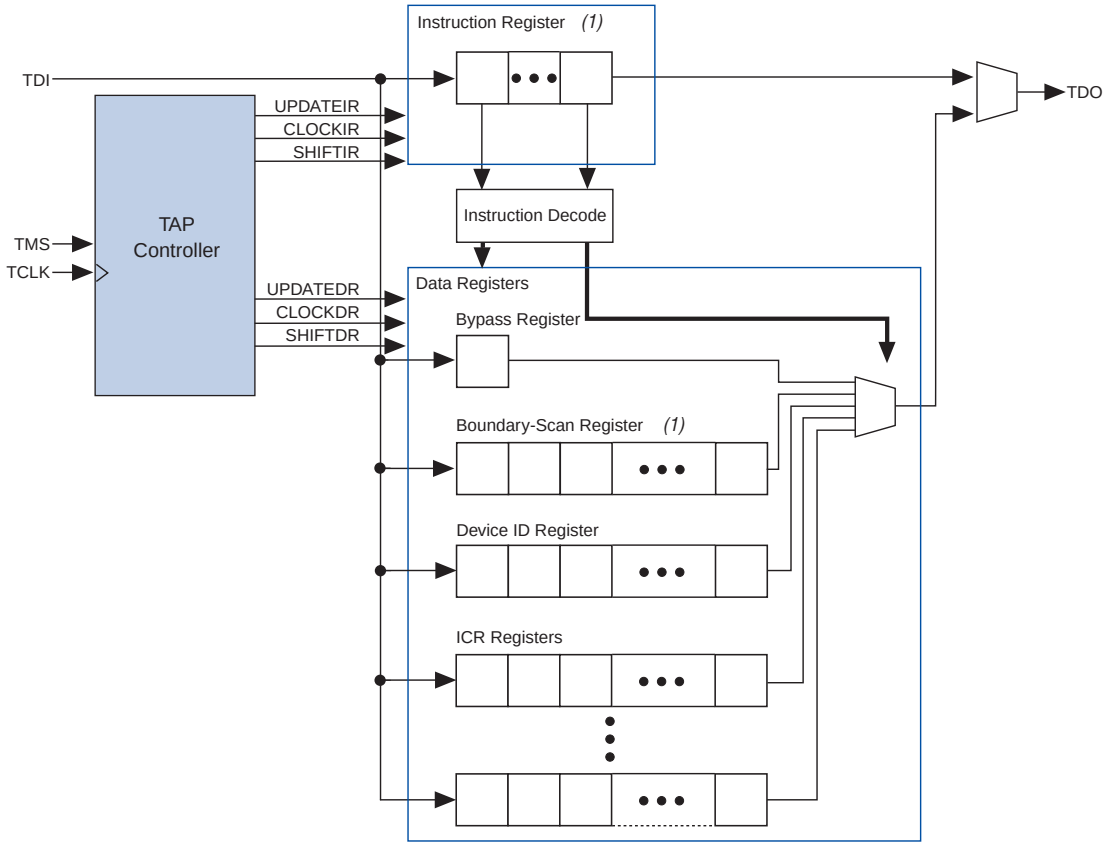
Figure 1–80. IEEE Std. 1149.1 Boundary-Scan Testing



For information about configuring Arria V devices through the IEEE Std. 1149.1 circuitry, refer to the *JTAG Configuration* section.

Figure 1–81 shows a functional model of the IEEE Std. 1149.1 circuitry.

Figure 1–81. IEEE Std. 1149.1 Circuitry



The test access port (TAP) controller controls the IEEE Std. 1149.1 boundary-scan testing. For more information about the TAP controller, refer to "IEEE Std. 1149.1 BST Operation Control" on page 1–105. The TMS and TCK pins operate the TAP controller, while the TDI and TDO pins provide the serial path for the data registers. The TDI pin also provides data to the instruction register, and generates the control logic for the data registers.

Table 1–21 lists the functions of each of these pins.

Table 1–21. IEEE Std. 1149.1 Pin Descriptions (Part 1 of 2)

Pin	Description	Function
TDI	Test data input	Serial input pin for instructions as well as testing and programming data. Data is shifted in on the rising edge of TCK.
TDO	Test data output	Serial data output pin for instructions as well as testing 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.

Table 1–21. IEEE Std. 1149.1 Pin Descriptions (Part 2 of 2)

Pin	Description	Function
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the test access port (TAP) controller state machine. Transitions within the state machine occur at the rising edge of TCK. Therefore, you must set up TMS before the rising edge of TCK. TMS is evaluated on the rising edge of TCK.
TCK	Test clock input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge.

The IEEE Std. 1149.1 BST circuitry requires the following registers:

- Instruction register—determines the action to be performed and the data register to be accessed.
- Bypass register—a 1-bit-long data register that provides a minimum-length serial path between TDI and TDO.

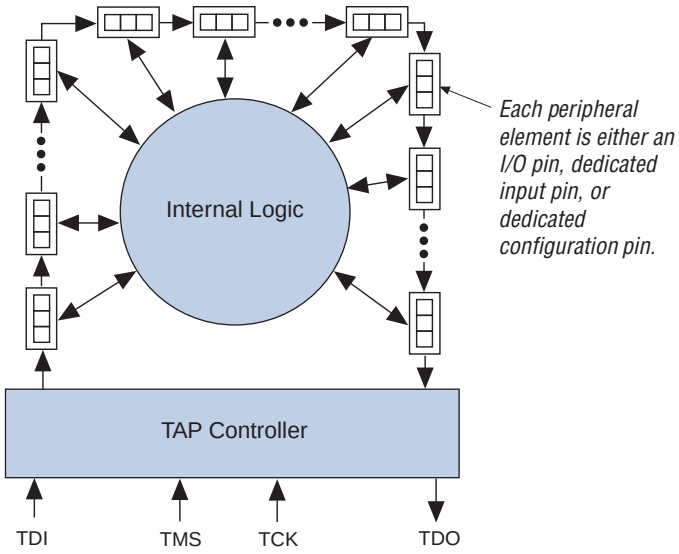
Boundary-scan register—a shift register composed of all the boundary-scan cells (BSCs) of the device.

IEEE Std. 1149.1 Boundary-Scan Register

The boundary-scan register is a large serial shift register that uses the TDI pin as an input and the TDO pin as an output. The boundary-scan register consists of 3-bit peripheral elements that are associated with Arria V I/O pins. You can use the boundary-scan register to test external pin connections or to capture internal data.

Figure 1–82 shows how test data is serially shifted around the periphery of the IEEE Std. 1149.1 device.

Figure 1–82. Boundary-Scan Register



Boundary-Scan Cells of an Arria V Device I/O Pin

The Arria V device 3-bit BSC consists of a set of capture registers and a set of update registers. The capture registers connect to internal device data through the OUTJ, OEJ, and PIN_IN signals, while the update registers connect to external data through the PIN_OUT and PIN_OE signals. The TAP controller generates the global control signals for the IEEE Std. 1149.1 BST registers (shift, clock, and update) internally. A decode of the instruction register generates the MODE signal. The data signal path for the boundary-scan register runs from the serial data in (SDI) signal to the serial data out (SDO) signal. The scan register begins at the TDI pin and ends at the TDO pin of the device.

Figure 1–83 shows the user I/O BSC for Arria V devices.

Figure 1–83. Arria V Device's User I/O BSC with IEEE Std. 1149.1 BST Circuitry for Arria V Devices

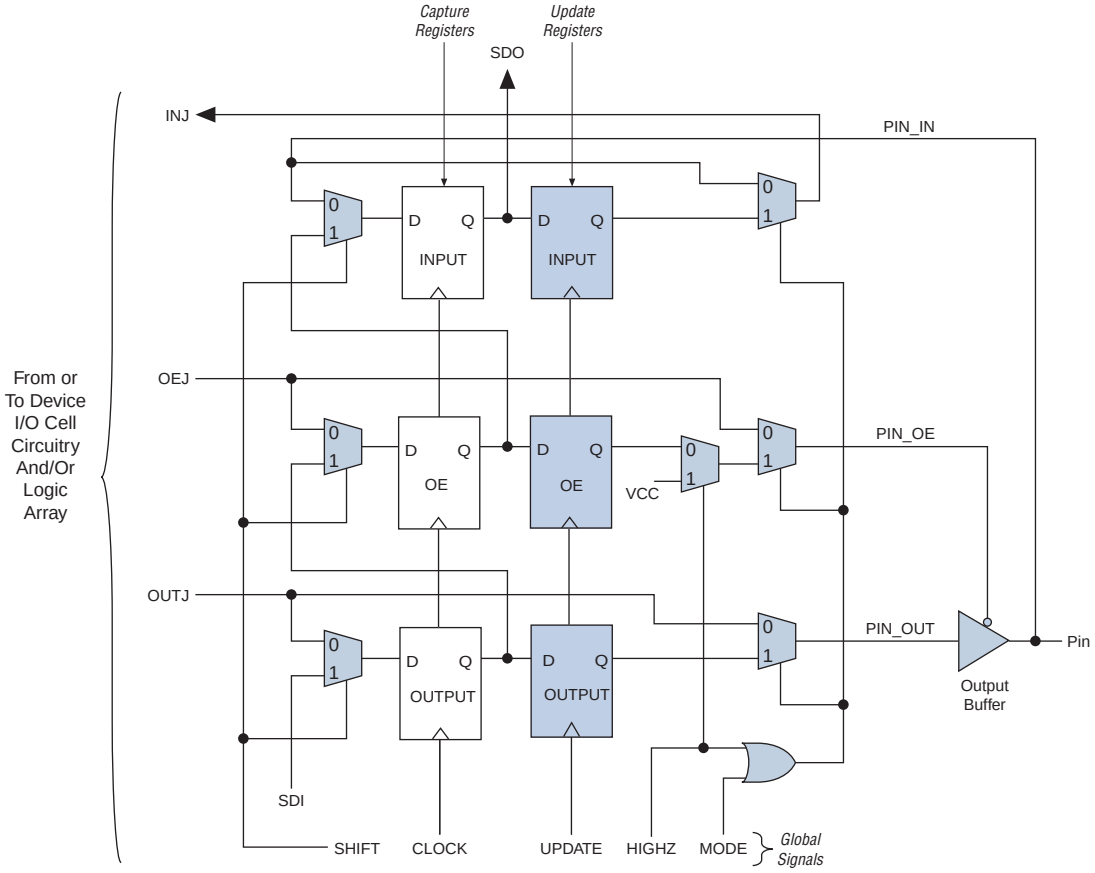


Table 1–22 lists the capture and update register capabilities of all BSCs within Arria V devices.

Table 1–22. Boundary Scan Cell Descriptions for Arria V Devices *(1)*

Pin Type	Captures			Drives			Comments
	Output Capture Register	OE Capture Register	Input Capture Register	Output Update Register	OE Update Register	Input Update Register	
User I/O pins	OUTJ	OEJ	PIN_IN	PIN_OUT	PIN_OE	INJ	NA
Dedicated clock input	0	1	PIN_IN	N.C. *(2)*	N.C. *(2)*	N.C. *(2)*	PIN_IN drives to the clock network or logic array
Dedicated input *(3)*	0	1	PIN_IN	N.C. *(2)*	N.C. *(2)*	N.C. *(2)*	PIN_IN drives to the control logic
Dedicated bidirectional (open drain) *(4)*	0	OEJ	PIN_IN	N.C. *(2)*	N.C. *(2)*	N.C. *(2)*	PIN_IN drives to the configuration control
Dedicated bidirectional *(5)*	OUTJ	OEJ	PIN_IN	N.C. *(2)*	N.C. *(2)*	N.C. *(2)*	PIN_IN drives to the configuration control and OUTJ drives to the output buffer
Dedicated output *(6)*	OUTJ	0	0	N.C. *(2)*	N.C. *(2)*	N.C. *(2)*	OUTJ drives to the output buffer

Notes to Table 1–22:

(1) TDI, TDO, TMS, TCK, all VCC and GND pin types, and VREF pins do not have BSCs.
(2) No Connect (N.C.).
(3) This includes the PLL_ENA, nCONFIG, MSEL0, MSEL1, MSEL2, MSEL3, MSEL4, and nCE pins.
(4) This includes the CONF_DONE and nSTATUS pins.
(5) This includes DCLK pin.
(6) This includes nCEO pin.

IEEE Std. 1149.1 BST Operation Control

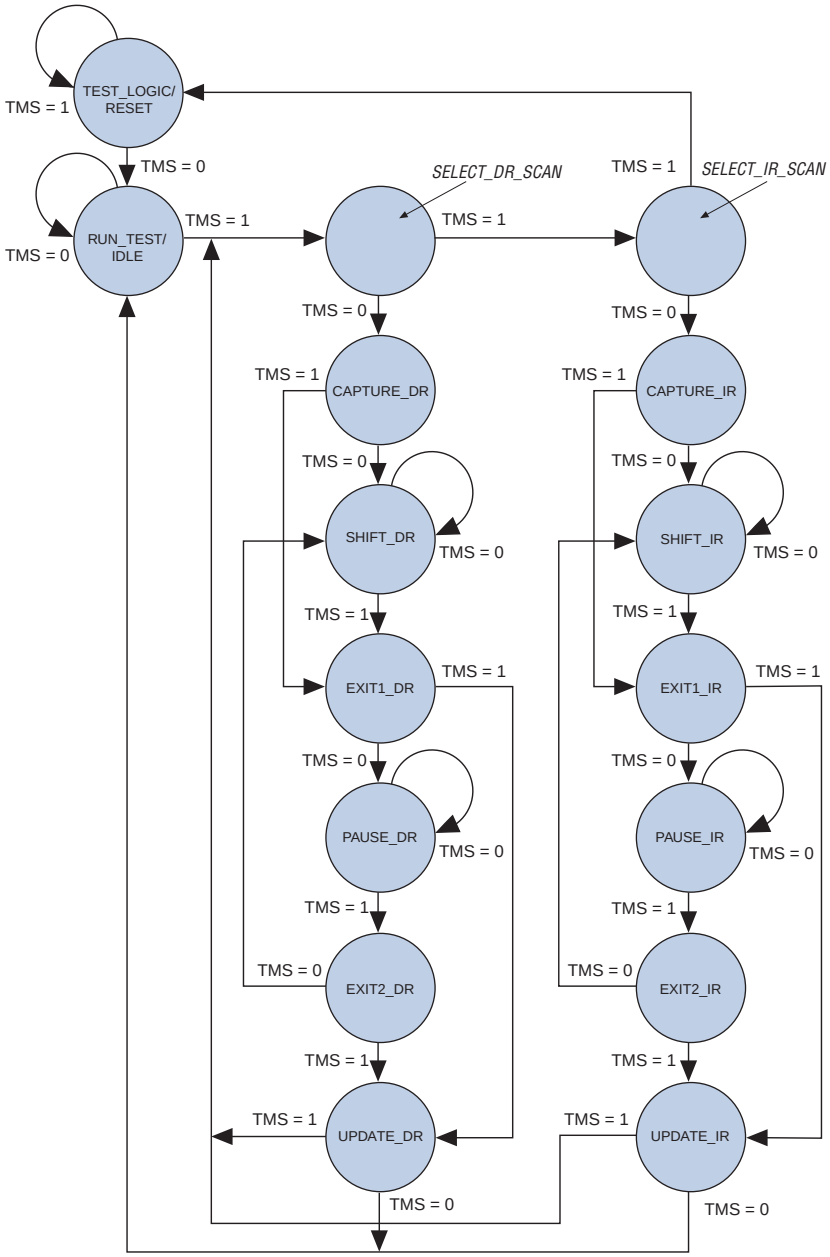
Arria V devices support several of the IEEE Std. 1149.1 BST instructions.

For more information about JTAG instruction code, refer to the *JTAG Boundary-Scan Testing in Arria V Devices* chapter.

The IEEE Std. 1149.1 TAP controller—a 16-state machine clocked on the rising edge of TCK—uses the TMS pin to control IEEE Std. 1149.1 operation in the device.

Figure 1–84 shows the TAP controller state machine.

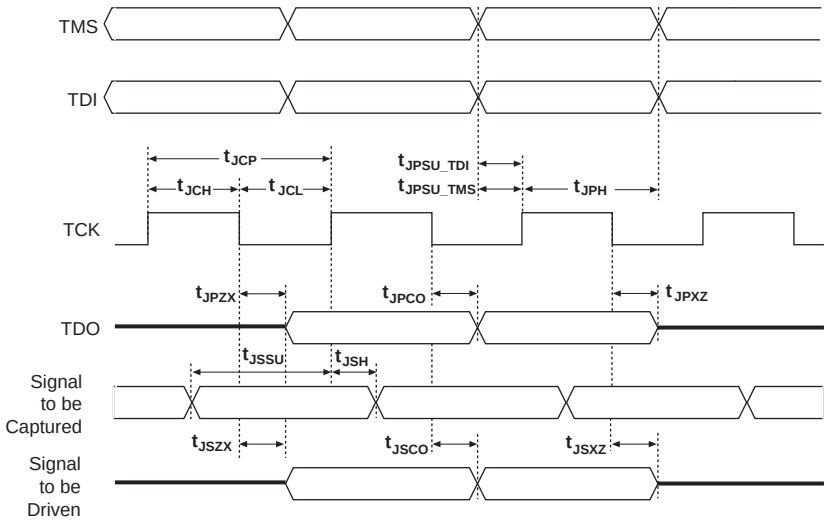
Figure 1–84. IEEE Std. 1149.1 TAP Controller State Machine



When the TAP controller is in the TEST_LOGIC/RESET state, the BST circuitry is disabled, the device is in normal operation, and the instruction register is initialized with IDCODE as the initial instruction. At device power up, the TAP controller starts in the TEST_LOGIC/RESET state. You can also force the TAP controller to the TEST_LOGIC/RESET state by holding TMS high for five TCK clock cycles. After the TAP controller is in the TEST_LOGIC/RESET state, the TAP controller remains in this state as long as TMS is held high (while TCK is clocked).

Figure 1–85 shows the timing requirements for the IEEE Std. 1149.1 signals.

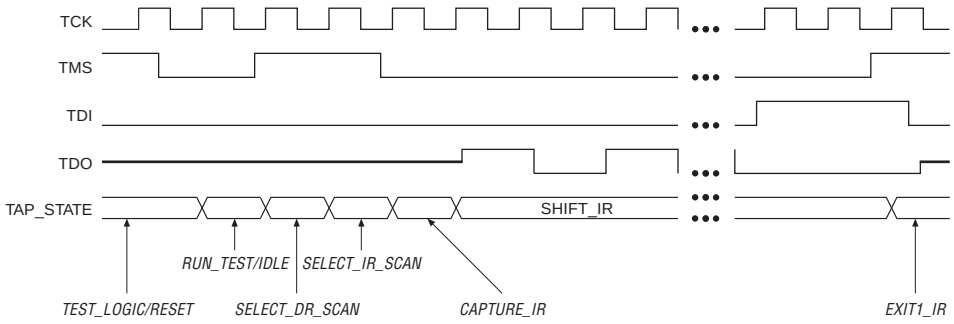
Figure 1–85. IEEE Std. 1149.1 Timing Waveforms



To start an IEEE Std. 1149.1 operation, select an instruction mode by advancing the TAP controller to the shift instruction register (SHIFT_IR) state and shift in the appropriate instruction code on the TDI pin.

Figure 1–86 shows the entry of the instruction code into the instruction register, the values of TCK, TMS, TDI, TDO, and the states of the TAP controller.

Figure 1–86. Selecting the Instruction Mode



From the RESET state, TMS is clocked with the pattern 01100 to advance the TAP controller to the SHIFT_IR state. The TDO pin is tri-stated in all states except in the SHIFT_IR and SHIFT_DR states. The TDO pin is activated at the first falling edge of TCK after entering SHIFT_IR or SHIFT_DR state and is tri-stated at the first falling edge of TCK after leaving SHIFT_IR or SHIFT_DR state.

When the SHIFT_IR state is activated, TDO is no longer tri-stated and the initial state of the instruction register is shifted out on the falling edge of TCK. TDO continues to shift out the contents of the instruction register as long as the SHIFT_IR state is active. The TAP controller remains in the SHIFT_IR state as long as TMS remains low.

During the SHIFT_IR state, you can enter an instruction code by shifting data on the TDI pin on the rising edge of TCK. The last bit of the instruction code is clocked at the same time that the next state, EXIT1_IR, is activated. Set TMS high to activate the EXIT1_IR state. After in the EXIT1_IR state, TDO becomes tri-stated again. TDO is always tri-stated except in the SHIFT_IR and SHIFT_DR states. After an instruction code is entered correctly, the TAP controller advances to shift test data serially in one of the following three modes:

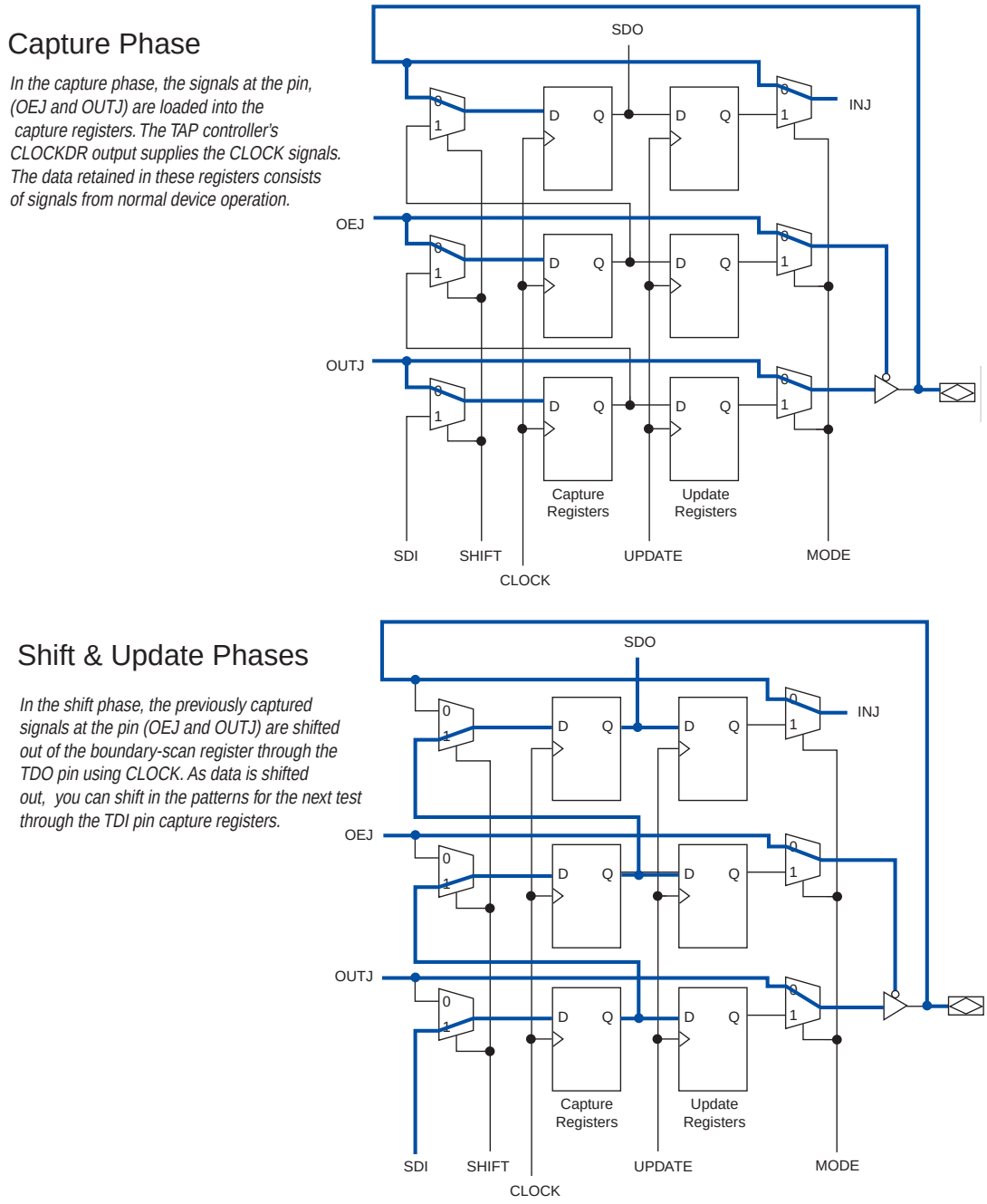
- "SAMPLE/PRELOAD Instruction Mode" on page 1–108
- "EXTEST Instruction Mode" on page 1–110
- "BYPASS Instruction Mode" on page 1–112

SAMPLE/PRELOAD Instruction Mode

SAMPLE/PRELOAD instruction mode allows you to take a snapshot of device data without interrupting normal device operation. However, this instruction is most often used to preload the test data into the update registers prior to loading the EXTEST instruction.

Figure 1–87 shows the capture, shift, and update phases of SAMPLE/PRELOAD mode.

Figure 1–87. IEEE Std. 1149.1 BST SAMPLE/PRELOAD Mode

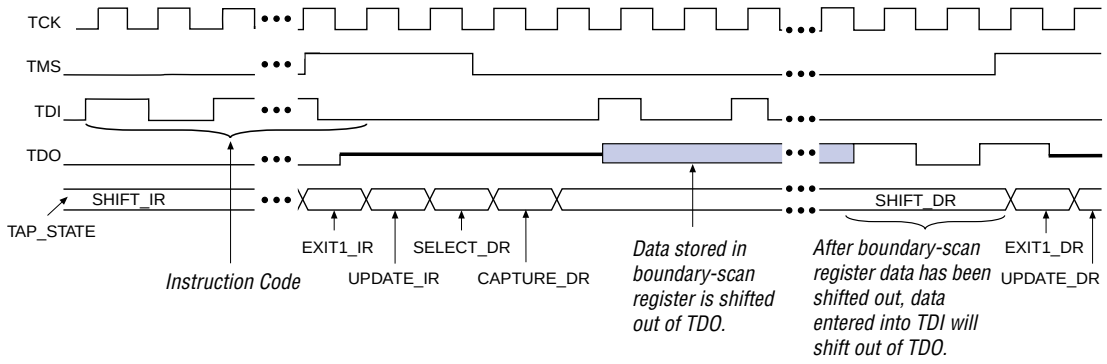


During the capture phase, multiplexers preceding the capture registers select the active device data signals. This data is then clocked into the capture registers. The multiplexers at the outputs of the update registers also select active device data to prevent functional interruptions to the device. During the shift phase, the boundary-scan shift register is formed by clocking data through capture registers around the device periphery and then out of the TDO pin. The device can simultaneously shift new test data into TDI and replace the contents of the capture registers. During the update phase, data in the capture registers are transferred to the update registers. Use this data in EXTEST instruction mode. For more information, refer to "EXTEST Instruction Mode" on page 1–110.

Figure 1–88 shows the SAMPLE/PRELOAD waveforms. The TDI pin shifts in the SAMPLE/PRELOAD instruction code. The TAP controller advances to the CAPTURE_DR state and then to the SHIFT_DR state, where it remains if you hold TMS low. The data that was present in the capture registers after the capture phase is shifted out of the TDO pin. New test data shifted into the TDI pin appears at the TDO pin after being clocked through the entire boundary-scan register.

Figure 1–88 shows that the instruction code at TDI does not appear at the TDO pin until after the capture register data is shifted out. If you hold TMS high on two consecutive TCK clock cycles, the TAP controller advances to the UPDATE_DR state for the update phase.

Figure 1–88. SAMPLE/PRELOAD Shift Data Register Waveforms



EXTEST Instruction Mode

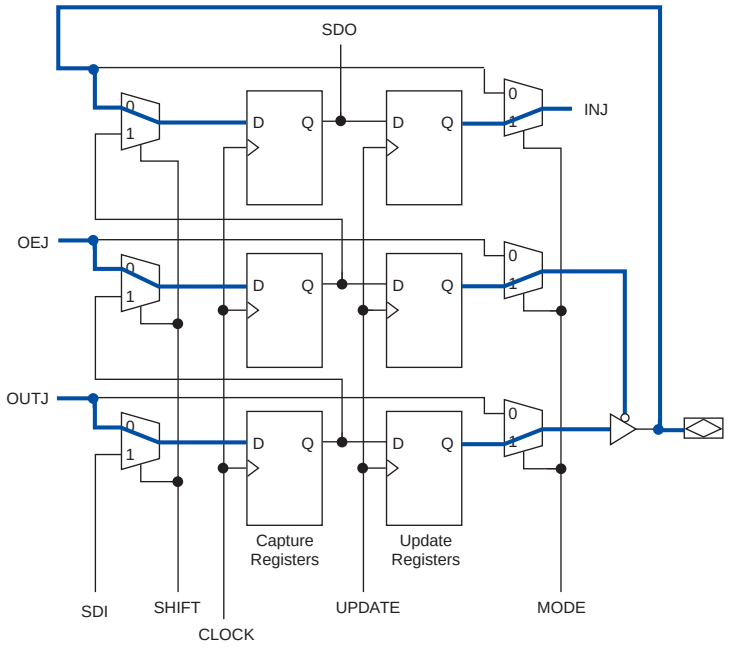
Use EXTEST instruction mode primarily to check external pin connections between devices. Unlike SAMPLE/PRELOAD mode, EXTEST allows test data to be forced onto the pin signals. By forcing known logic high and low levels on output pins, you can detect opens and shorts at the pins of any device in the scan chain.

Figure 1–89 shows the capture, shift, and update phases of EXTEST mode.

Figure 1–89. IEEE Std. 1149.1 BST EXTEST Mode

Capture Phase

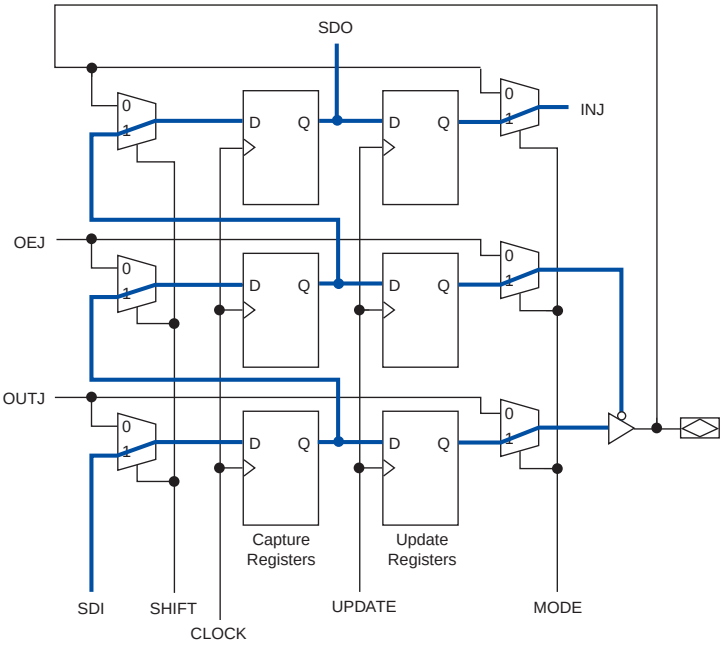
In the capture phase, the signals at the pin (OEJ and OUTJ) are loaded into the capture registers. The TAP controller's CLOCKDR output supplies the CLOCK signals. Previously retained data in the update registers drive PIN_IN and INJ, and allows the I/O pin to tri-state or drive a signal out.



Shift & Update Phases

In the shift phase, the previously captured signals at the pins (OEJ and OUTJ) are shifted out of the boundary-scan register through the TDO pin using CLOCK. As data is shifted out, you can shift in the patterns for the next test through the TDI pin.

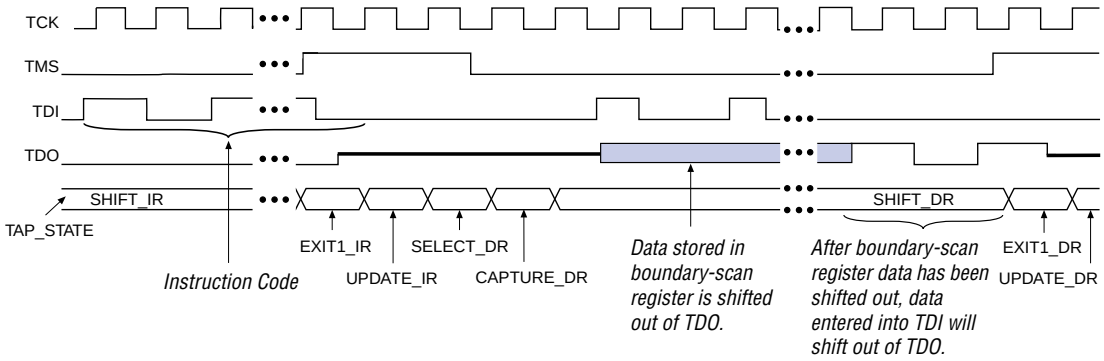
In the update phase, data is transferred from the capture registers to the update registers using the UPDATE clock. The update registers then drive PIN_IN and INJ, and allow the I/O pin to tri-state or drive a signal out.



EXTEST selects data differently than SAMPLE/PRELOAD. EXTEST chooses data from the update registers as the source of the output and output enable signals. After the EXTEST instruction code is entered, the multiplexers select the update register data. Thus, you can force the data stored in these registers from a previous EXTEST or SAMPLE/PRELOAD test onto the pin signals. In the capture phase, the results of this test data is stored in the capture registers and then shifted out of TDO during the shift phase. You can then store new test data in the update registers during the update phase.

The EXTEST waveform diagram in Figure 1–90 resembles the SAMPLE/PRELOAD waveform diagram, except for the instruction code. The data shifted out of TDO consists of the data that was present in the capture registers after the capture phase. New test data shifted into the TDI pin appears at the TDO pin after being clocked through the entire boundary-scan register.

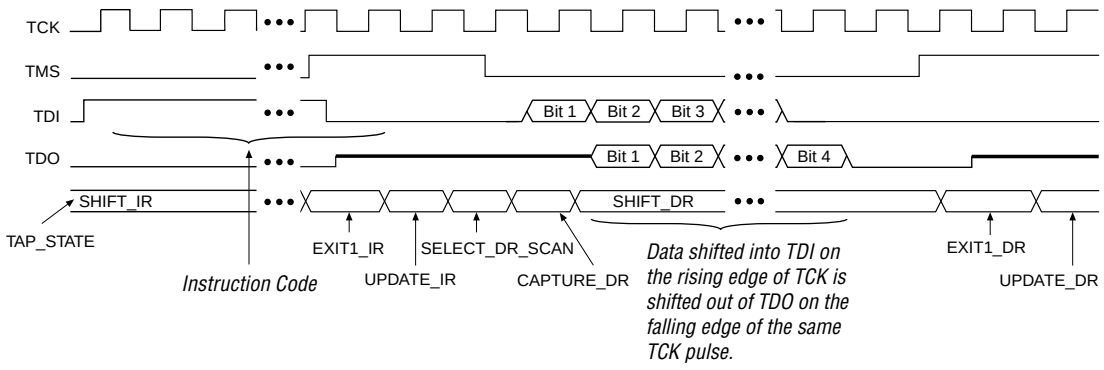
Figure 1–90. EXTEST Shift Data Register Waveforms



BYPASS Instruction Mode

BYPASS mode is activated when an instruction code of all ones is loaded in the instruction register. The waveform diagram in Figure 1–91 show how scan data passes through a device after the TAP controller is in the SHIFT_DR state. In this state, data signals are clocked into the bypass register from TDI on the rising edge of TCK and out of TDO on the falling edge of the same clock pulse.

Figure 1–91. BYPASS Shift Data Register Waveforms



IDCODE Instruction Mode

Use IDCODE instruction mode to identify the devices in an IEEE Std. 1149.1 chain. When you select IDCODE, the device identification register is loaded with the 32-bit vendor-defined identification code. The device ID register is connected between the TDI and TDO ports and the device IDCODE is shifted out.

For more information on IDCODE for Arria V devices, refer to the *JTAG Boundary-Scan Testing in Arria V Devices* chapter.

USERCODE Instruction Mode

Use USERCODE instruction mode to examine the user electronic signature (UES) within the devices along an IEEE Std. 1149.1 chain. When you select this instruction, the device identification register is connected between the TDI and TDO ports. The user-defined UES is shifted into the device ID register in parallel from the 32-bit USERCODE register. The UES is then shifted out through the device ID register.

The UES value is not user defined until after the device is configured. Before configuration, the UES value is set to the default value.

CLAMP Instruction Mode

Use CLAMP instruction mode to allow the boundary-scan register to determine the state of the signals driven from the pins, while the bypass register is selected as the serial path between the TDI and the TDO ports. The data held in the boundary-scan register define the state of all the signals.

If you are testing after configuring the device, the programmable weak pull-up resistor or the bus hold feature overrides the CLAMP value (the value stored in the update register of the BSC) at the pin.

HIGHZ Instruction Mode

HIGHZ instruction mode sets all of the user I/O pins to an inactive drive state. These pins are tri-stated until a new JTAG instruction is executed. When this instruction is loaded into the instruction register, the bypass register is connected between the TDI and the TDO ports.

If you are testing after configuring the device, the programmable weak pull-up resistor or the bus hold feature overrides the HIGHZ value at the pin.

Using IEEE Std. 1149.1 BST Circuitry

Arria V devices have dedicated JTAG pins and the IEEE Std. 1149.1 BST circuitry is enabled after device power up. You can perform BST on Arria V devices before, after, and during configuration. Arria V devices support the BYPASS, IDCODE, and SAMPLE instructions during configuration without interrupting configuration. To send all other JTAG instructions, you must interrupt configuration with the CONFIG_IO instruction.

The CONFIG_IO instruction allows you to configure the I/O buffers through the JTAG port, and when issued, interrupts configuration. This instruction allows you to perform board-level testing prior to configuring Arria V devices or you can wait for the configuration device to complete the configuration. After configuration is interrupted and JTAG BST is complete, you must reconfigure the device through JTAG (PULSE_CONFIG instruction) or by pulsing nCONFIG low.

When you perform JTAG boundary-scan testing before configuration, you must hold the nCONFIG pin low.

The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on Arria V devices do not affect JTAG boundary-scan or configuration operations. Toggling these pins do not disrupt BST operation (other than the expected BST behavior).

When you design a board for JTAG configuration of Arria V devices, you must consider the connections for the dedicated configuration pins.

For more information on using the IEEE Std.1149.1 circuitry for device configuration, refer to the *JTAG Configuration* section.

Disabling IEEE Std. 1149.1 BST Circuitry

The IEEE Std. 1149.1 BST circuitry for Arria V 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 "Using IEEE Std. 1149.1 BST Circuitry" on page 1–114.

If you are not using the IEEE Std. 1149.1 circuitry in Arria V devices, you must permanently disable the circuitry to ensure that you do not inadvertently enable the circuitry when it is not required.

Table 1–23 lists the pin connections necessary for disabling the IEEE Std. 1149.1 circuitry in Arria V devices.

Table 1–23. Disabling IEEE Std. 1149.1 Circuitry

JTAG Pins (1)	Connection for Disabling
TMS	V_{CCPD} supply of Bank 3A
TCK	GND
TDI	V_{CCPD} supply of Bank 3A
TDO	Leave open

Note to Table 1–23:

(1) There is no software option to disable JTAG in Arria V devices. The JTAG pins are dedicated.

Guidelines for IEEE Std. 1149.1 Boundary-Scan Testing

When performing boundary-scan testing with IEEE Std. 1149.1 devices, use the following guidelines:

- If the "10..." pattern does not shift out of the instruction register through the TDO pin during the first clock cycle of the SHIFT_IR state, the TAP controller did not reach the proper state. To solve this problem, try one of the following procedures:
  - Verify that the TAP controller has reached the SHIFT_IR state correctly. To advance the TAP controller to the SHIFT_IR state, return to the RESET state and send the code 01100 to the TMS pin.
  - Check the connections to the VCC, GND, JTAG, and dedicated configuration pins on the device.
- Perform a SAMPLE/PRELOAD test cycle prior to the first EXTEST test cycle to ensure that known data is present at the device pins when you enter EXTEST mode. If the OEJ update register contains a 0, the data in the OUTJ update register is driven out. The state must be known and correct to avoid contention with other devices in the system.
- Do not perform EXTEST testing during in-circuit reconfigurability (ICR). This instruction is supported before or after ICR, but not during ICR. Use the CONFIG_IO instruction to interrupt configuration and then perform testing, or wait for the configuration to complete.
- If performing testing before configuration, hold the nCONFIG pin low.
- After configuration, you cannot test any pins in a differential pin pair. Therefore, performing BST after configuration requires editing of the BSC group definitions that correspond to these differential pin pairs. You must redefine the BSC group as an internal cell.

For more information about editing, refer to the IEEE 1149.1 BSDL Files page on the Altera website.

Power Management

This section describes the basic information of power management, hot-socketing specifications, power-on reset (POR) requirements, and their implementation in Arria V devices.

Use the information in this section in conjunction with the *Power Management in Arria V Devices* chapter.

Power Consumption

The total power of an FPGA includes the following:

- Static power—the power consumed by the FPGA when it is configured but no clocks are operating.
- Dynamic power—the switching power when the FPGA is configured and running. Equation 1–3 shows how to calculate dynamic power.

Equation 1–3. Dynamic Power Equation (1)

$$P = \frac{1}{2}CV^2 \times frequency$$

Note to Equation 1–3:

(1) P = power; C = load capacitance; and V = supply voltage level.

Equation 1–3 shows that power is design-dependent and is determined by the operating frequency of the design. Arria V devices minimize static and dynamic power using advanced process optimizations. This technology allows Arria V designs to meet specific performance requirements with the lowest possible power.

Power consumption also affects thermal management. Arria V devices offer an internal temperature sensing diode (TSD) feature that self-monitors the device junction temperature and can be used with external circuitry for other activities, such as controlling air flow to the Arria V FPGA.

Hot-Socketing Specifications

The advantages of hot-socketing support in Arria V devices include the following:

- "Arria V Devices Can Be Driven Before Power Up"
- "I/O Pins Remain Tri-Stated During Power Up"
- "Insertion or Removal of the Arria V Device from a Powered-Up System"

Arria V Devices Can Be Driven Before Power Up

In Arria V devices, you can drive signals into the I/O pins, dedicated input pins, and dedicated clock pins before or during power up or power down without damaging the device. External input signals to the I/O pins of the device do not internally power the V_{CCIO}, V_{CCPD}, V_{CC}, and V_{CCP} power supplies through internal paths within the device. To simplify the system level design, Arria V devices support power up or power down of the power supplies in any sequence.

I/O Pins Remain Tri-Stated During Power Up

A device that does not support hot-socketing can interrupt the system operation or cause contention by driving out before or during power up. In a hot-socketing situation, the Arria V output buffers are tri-stated during system power up or power down. Also, the Arria V device does not drive signal out until the device is configured and working within the recommended operating conditions.

Insertion or Removal of the Arria V Device from a Powered-Up System

Devices that do not support hot-socketing may sink current through the device signal pins to the device power supplies. This can create a direct connection to GND, causing power supply failures.

This irregular power up can damage both the driving and driven devices and can disrupt card power up.

You can insert an Arria V device into or remove it from a powered-up system board without damaging the system board or interfering with its operation.

When hot-socketing, Arria V devices are immune to latch up. Latch up occurs when a device that does not support hot-socketing feature is hot-socketed into an active system. During hot-socketing, the signal pins can be connected and driven by the active system before the power supply can provide current to the power and ground planes of the device. This condition can lead to latch up and cause a low-impedance path from power to GND within the device. As a result, the device draws a large amount of current, possibly causing electrical damage.

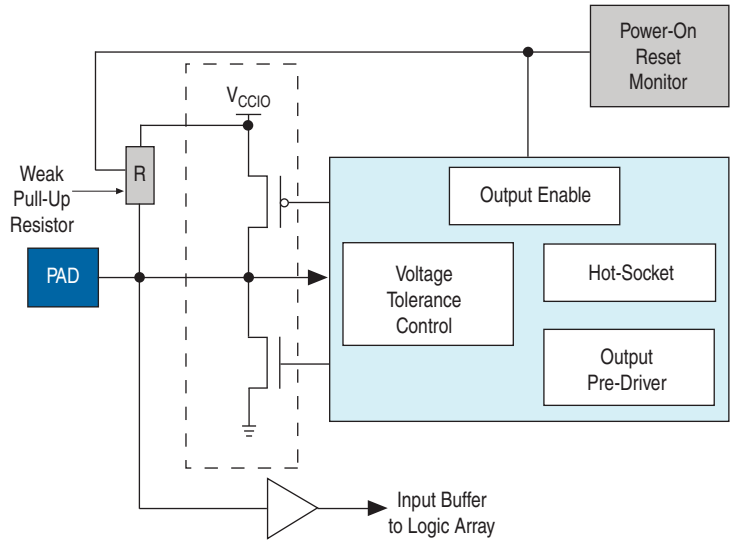
Hot-Socketing Implementation

The hot-socketing feature turns off the output buffer during power up and power down of the V_{CCIO}, V_{CCPD}, V_{CC}, and V_{CCP} power supplies. When these power supplies are below the threshold voltage, the hot-socketing circuitry generates an internal `HOTSCKT` signal.

Hot-socketing circuitry prevents excess I/O leakage during power up. When the voltage ramps up very slowly, I/O leakage is still relatively low, even after the POR signal is released and configuration is complete. In this situation, the `CONF_DONE` and `nSTATUS` pins fail to respond, as the output buffer cannot flip from the state set by the hot-socketing circuitry at this low voltage. Therefore, the hot-socketing circuitry is removed from these configuration pins to ensure that they can operate during configuration. Thus, it is an expected behavior for these pins to drive out during power-up and power-down sequences.

Figure 1–92 shows the I/O pin circuitry for Arria V devices.

Figure 1–92. Hot-Socketing Circuitry for Arria V Devices



POR circuitry monitors the voltage level of the power supplies (V_{CC}, V_{CCP}, V_{CCPGM}, V_{CCPD}, V_{CC_AUX}, and V_{CCBAT}) and keeps the I/O pins tri-stated until the device is in user mode. The weak pull-up resistor (R) in the Arria V input/output element (IOE) keeps the I/O pins from floating. The 3.3-V tolerance control circuit permits 3.3 V to drive the I/O pins before the V_{CC}, V_{CCP}, V_{CCPD}, and V_{CCIO} power supplies are powered and prevents the I/O pins from driving out when the device is not in user mode.

Altera uses GND as a reference for hot-socketing operations and I/O buffer designs. To ensure proper operation, you must connect GND between boards before connecting the power supplies. This prevents GND on your board from being pulled up inadvertently by a path to power through other components on your board. A pulled up GND could otherwise cause an out-of-specification I/O voltage or over current condition in the Altera® device.

Power-On Reset Circuitry

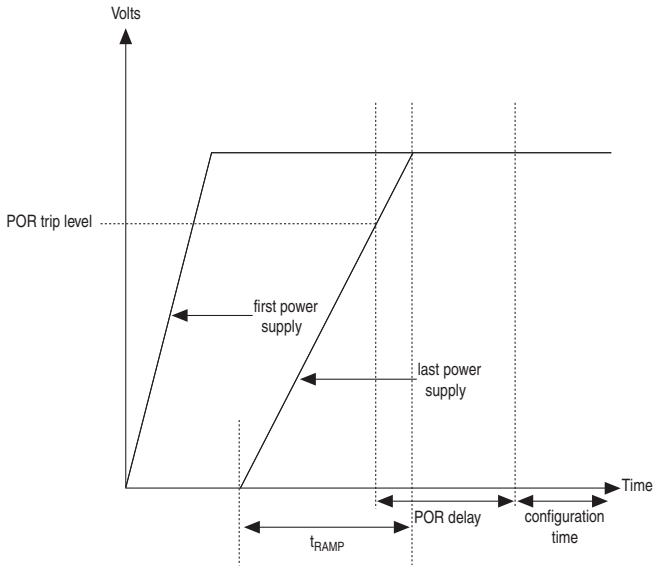
POR circuitry keeps the devices in the reset state until the power supply outputs are within operating range.

When you apply power to the Arria V device, a POR event occurs if the power supply reaches the recommended operating range within the maximum power supply ramp time (t_{RAMP}). If t_{RAMP} is not met, the device I/O pins and programming registers remain tri-stated, during which device configuration could fail. The maximum t_{RAMP} for Arria V devices is 100 ms; the minimum t_{RAMP} is 200 µs.

For more information about fast and standard POR delay specification, refer to the *Device Datasheet for Arria V Devices* chapter.

Figure 1–93 shows the relationship between t_{RAMP} and POR delay.

Figure 1–93. Relationship Between t_{RAMP} and POR Delay

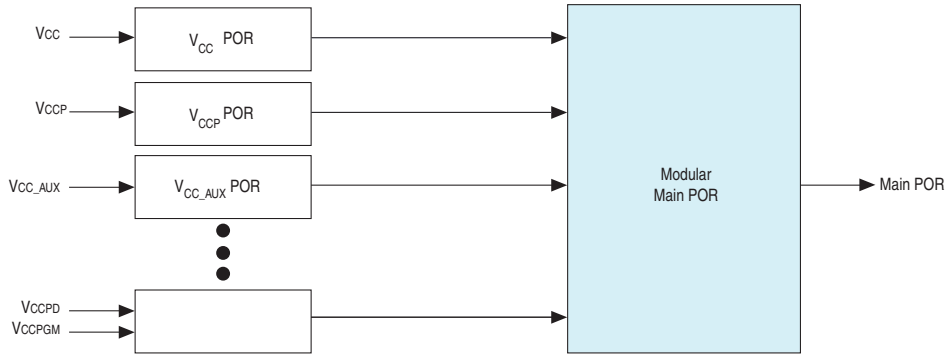


The Arria V POR circuitry uses an individual detecting circuit to monitor each of the configuration-related power supplies independently. The main POR circuitry is gated by the outputs of all the individual detectors. The main POR signal is asserted when the power starts to ramp up and is released after the last ramp-up power reaches its trip level during power up. In user mode, the main POR signal is asserted when any of the monitored power dips down below its trip level and forces the device into the reset state.

The POR also checks the functionality of the I/O level shifters powered by the V_{CCPD} and V_{CCPGM} power supplies during power-up mode. The main POR waits for all the individual PORs to release the POR signal so that the control block can start programming the device.

Figure 1–94 shows a simplified POR diagram for Arria V devices.

Figure 1–94. Simplified POR Diagram for Arria V Devices



Document Revision History

Table 1–24 lists the revision history for this chapter.

Table 1–24. Document Revision History

Date	Version	Changes
December 2011	1.2	■ Updated Figure 1–83 and Figure 1–94. ■ Updated Table 1–22. ■ Updated "Guidelines for IEEE Std. 1149.1 Boundary-Scan Testing" on page 1–115, "Hot-Socketing Specifications" on page 1–116, and "Hot-Socketing Implementation" on page 1–117.
November 2011	1.1	■ Updated "Adaptive Logic Modules" on page 1–1, "Independent Complex Multiplier Mode" on page 1–14, and "Differential Pin Placement Guidelines". ■ Updated Figure 1–35, Figure 1–38, Figure 1–51, Figure 1–52, Figure 1–55, and Figure 1–56. ■ Updated Table 1–9 and Table 1–22. ■ Minor text edits.
August 2011	1.0	Initial release.

2. Transceiver Basics for Arria V Devices

AV-54002-1.1

This chapter contains basic technical details pertaining to specific features in the Arria® V device transceivers. This chapter serves as a supplementary reading to the *Volume 3: Transceivers* of the *Arria V Device Handbook* and covers the following topics:

Transceiver Architecture

This section provides a detailed description of the architecture and operations of the Arria® V transceiver channel datapaths.

For information about using the control and status signals associated with each transceiver feature, refer to the *Altera Transceiver PHY IP Core User Guide*.

Use the information in this section in conjunction with the *Transceiver Architecture in Arria V Devices* chapter.

Transmitter PMA Datapath

This section describes the serializer and transmitter buffer blocks in the transmitter PMA datapath.

Serializer

The serializer provides parallel-to-serial data conversion and sends the data LSB first from the transmitter physical coding sublayer (PCS) to the transmitter buffer. Additionally, the serializer provides the polarity inversion and bit reversal features.

Polarity Inversion

The positive and negative signals of a serial differential link might accidentally be swapped during board layout. The polarity inversion feature of the transmitter corrects this error without requiring a board respin or major updates to the logic in the FPGA fabric. The polarity inversion feature inverts the polarity of every bit at the input to the serializer, which has the same effect as swapping the positive and negative signals of the serial differential link.

Polarity inversion is controlled dynamically with the `tx_invpolarity` register. When you enable the polarity inversion feature, it may cause initial disparity errors at the receiver with 8B/10B-coded data. The downstream system at the receiver must be able to tolerate these disparity errors.

CAUTION Enabling polarity inversion midway through a serialized word corrupts the word.

Bit Reversal

You can reverse the transmission bit order to achieve MSB-to-LSB ordering using the bit reversal feature at the transmitter. Table 2–1 lists the bit reversal serialization factors.

Table 2–1. Bit Reversal Feature

Bit Reversal Option	Transmission Bit Order	
	8- or 10-bit Serialization Factor	16- or 20-bit Serialization Factor
Disabled (default)	LSB to MSB	LSB to MSB
Enabled	MSB to LSB For example: 8-bit—D[7:0] rewired to D[0:7] 10-bit—D[9:0] rewired to D[0:9]	MSB to LSB For example: 16-bit—D[15:0] rewired to D[0:15] 20-bit—D[19:0] rewired to D[0:19]

Table 2–2 lists the features provided by the Pseudo Current Mode Logic (PCML) output buffer to the integrated circuitry.

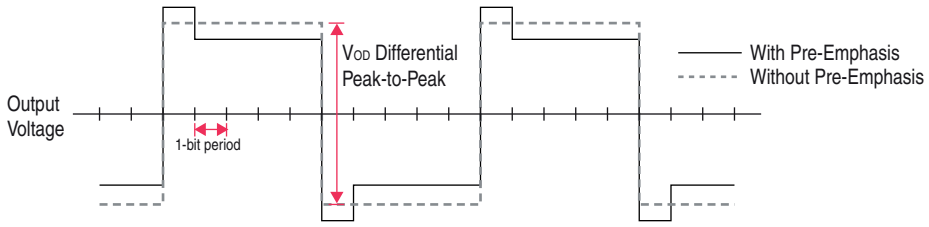
Table 2–2. Description of the Transmitter Buffer Features (Part 1 of 2)

Category	Features	Description
Improve Signal Integrity	Programmable Differential Output Voltage (V_{OD})	Controls the current mode drivers for signal amplitude to handle different trace lengths, various backplanes, and receiver requirements. The actual V_{OD} level is a function of the current setting and the transmitter termination value.
	Programmable Pre-Emphasis	Boosts the high-frequency components of the transmitted signal, which may be attenuated when propagating through the transmission medium. The physical transmission medium can be represented as a low-pass filter in the frequency domain. Variation in the signal frequency response that is caused by attenuation significantly increases the data-dependent jitter and other intersymbol interference (ISI) effects at the receiver end. Using the pre-emphasis feature maximizes the data opening at the far-end receiver. Figure 2–1 shows the signal transmission at the transmitter output with and without applying pre-emphasis post-tap for a 5 Gigabit per second (Gbps) signal with an alternating data pattern of five 1s and five 0s.
	Programmable Slew Rate	Controls the rate of change for the signal transition.
Save Board Space and Cost	On-Chip Biasing	Establishes the required transmitter common-mode voltage (TX V_{CM}) level at the transmitter output. The circuitry is available only if you enable on-chip termination (OCT). When you disable OCT, you must implement off-chip biasing circuitry to establish the required TX V_{CM} level.
	Differential OCT	The termination resistance is adjusted by the calibration circuitry, which compensates for the process, voltage, and temperature variations (PVT). You can disable OCT and use external termination. However, you must implement off-chip biasing circuitry to establish the required TX V_{CM} level. TX V_{CM} is tri-stated when using external termination.

Table 2–2. Description of the Transmitter Buffer Features (Part 2 of 2)

Category	Features	Description
Reduce Power	Programmable V_{CM} Current Strength	Controls the impedance of V_{CM}. A higher impedance setting reduces current consumption from the on-chip biasing circuitry.
Protocol-Specific Function	Transmitter Output Tri-State	Enables the transmitter differential pair voltages to be held constant at the same value determined by the TX V_{CM} level with the transmitter in the high impedance state. The transmitter output tri-state feature is compliant with differential and common-mode voltage levels and operation time requirements for transmitter electrical idle, as specified in the PCI Express Base Specification 2.0 for Gen1 and Gen2 signaling rates.
	Receiver Detect	Provides link partner detection capability at the transmitter end using an analog mechanism for the receiver detection sequence during link initialization in the Detect state of the PCI Express® (PCIe®) Link Training and Status State Machine (LTSSM) states. The circuit detects if there is a receiver downstream by changing the transmitter V_{CM} to create a step voltage and measuring the resulting voltage rise time. For proper functionality, the series capacitor (AC-coupled link) and receiver termination values must comply with the PCI Express Base Specification 2.0. The circuit is clocked using `fixedclk` and requires that the transmitter OCT be enabled with the output tri-stated.

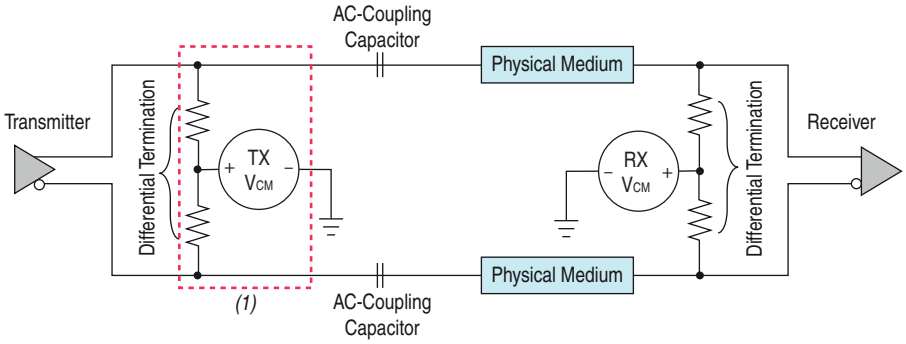
Figure 2–1. Example of Pre-Emphasis Effect on Signal Transmission at Transmitter Output



The receiver can be only AC-coupled to a transmitter. In an AC-coupled link, the AC-coupling capacitor blocks the transmitter V_{CM}. At the receiver end, the termination and biasing circuitry restores the V_{CM} level that is required by the receiver.

Figure 2–2 shows an AC-coupled link with the Arria V transmitter.

Figure 2–2. AC-Coupled Link with Arria V Transmitter



Note to Figure 2–2:

(1) When you disable OCT, you must implement external termination and off-chip biasing circuitry to establish the required TX V_{CM} level.

Receiver PMA Datapath

This section describes the receiver buffer, channel phase-locked loop (PLL) configured for clock data recovery (CDR) operation, and deserializer blocks in the receiver PMA datapath.

Receiver Buffer

Table 2–3 lists the features provided by the receiver buffer to the integrated circuitry.

Table 2–3. Description of the Receiver Buffer Features (Part 1 of 2)

Category	Features	Description
Improve Signal Integrity	Programmable Equalization	Boosts the high-frequency components of the received signal, which may be attenuated when propagating through the transmission medium. The physical transmission medium can be represented as a low-pass filter in the frequency domain. Variation in the signal frequency response that is caused by attenuation leads to data-dependent jitter and other ISI effects, causing incorrect sampling on the input data at the receiver. The amount of the high-frequency boost required at the receiver to overcome signal attenuation depends on the loss characteristics of the physical medium.
	Programmable DC Gain	Provides equal boost to the received signal across the frequency spectrum.
Save Board Space and Cost	On-Chip Biasing	Establishes the required receiver common-mode voltage (RX V_{CM}) level at the receiver input. The circuitry is available only if you enable OCT. When you disable OCT, you must implement off-chip biasing circuitry to establish the required RX V_{CM} level.
	Differential OCT	The termination resistance is adjusted by the calibration circuitry, which compensates for the PVT. You can disable OCT and use external termination. However, you must implement off-chip biasing circuitry to establish the required RX V_{CM} level. RX V_{CM} is tri-stated when using external termination.

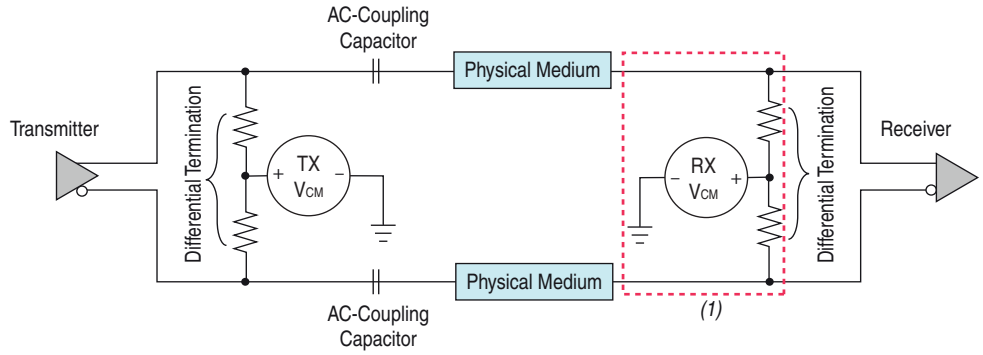
Table 2–3. Description of the Receiver Buffer Features (Part 2 of 2)

Category	Features	Description
Reduce Power	Programmable V_{CM} Current Strength	Controls the impedance of V_{CM}. A higher impedance setting reduces current consumption from the on-chip biasing circuitry.
Protocol-Specific Function	Signal Detect	Senses if the signal level present at the receiver input is above or below the threshold voltage that you specified. The detection circuitry has a hysteresis response that asserts the status signal only when a number of data pulses exceeding the threshold voltage are detected and deasserts the status signal when the signal level below the threshold voltage is detected for a number of recovered parallel clock cycles. The circuitry requires the input data stream to be 8B/10B-coded. Signal detect is compliant to the threshold voltage and detection time requirements for electrical idle detection conditions as specified in the PCI Express Base Specification 2.0 for Gen1 and Gen2 signaling rates.

The receiver can be only AC-coupled to a transmitter. In an AC-coupled link, the AC-coupling capacitor blocks the transmitter V_{CM}. At the receiver end, the termination and biasing circuitry restores the V_{CM} level that is required by the receiver.

Figure 2–3 shows an AC-coupled link with the Arria V receiver.

Figure 2–3. AC-Coupled Link with Arria V Receiver



Note to Figure 2–3:

(1) When you disable OCT, you must implement external termination and off-chip biasing circuitry to establish the required RX V_{CM} level.

Channel PLL

This section describes the architecture and operation of the channel PLL when it is configured as a CDR PLL.

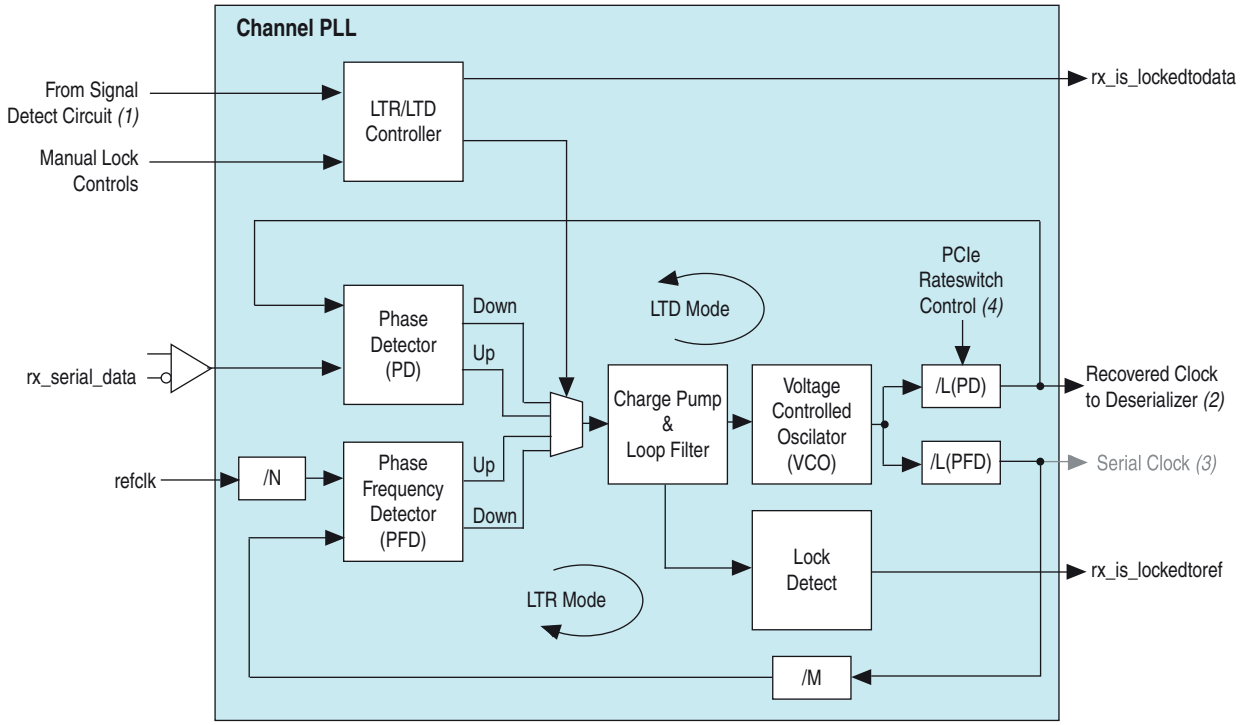
If you configure the channel PLL as a CDR PLL, the channel PLL recovers the clock and data from the serial data stream. If you do not use the channel PLL as a CDR PLL, you can configure it as a clock multiplier unit (CMU) PLL for clocking the transceivers.

For more information about the channel PLL operation when configured as a CMU PLL, refer to the "CMU PLL" section in the *Transceiver Architecture in Arria V Devices* chapter.

Channel PLL Architecture

Figure 2–4 shows the major components in a channel PLL. The channel PLL supports operation in either lock-to-reference (LTR) or lock-to-data (LTD) mode.

Figure 2–4. Channel PLL Block Diagram



Notes to Figure 2–4:

(1) Applicable only in a PCIe configuration.
(2) Applicable when configured as a CDR PLL.
(3) Applicable when configured as a CMU PLL.
(4) The PCIe rateswitch control allows dynamic switching between Gen2 and Gen1 line rates in a PCIe Gen2 design.

In LTR mode, the channel PLL tracks the input reference clock. The phase-frequency detector (PFD) compares the phase and frequency of the voltage controlled oscillator (VCO) output and the input reference clock. The resulting PFD output controls the VCO output frequency to half the data rate with the appropriate counter (M or L) value given an input reference clock frequency. The lock detect determines whether the PLL has achieved lock to the phase and frequency of the input reference clock.

In LTD mode, the channel PLL tracks the incoming serial data. The phase detector compares the phase of the VCO output and the incoming serial data. The resulting phase detector output controls the VCO output to continuously match the phase of the incoming serial data.

Use the LTR/LTD controller only when you configure the channel PLL as a CDR PLL.

Table 2–4 lists the counter values in the channel PLL.

Table 2–4. Counters in the Channel PLL *(1)*

Counter	Description	Values
N	Pre-scale counter to divide the input reference clock frequency to the PFD by the N factor	1, 2, 4, 8
M	Feedback loop counter to multiply the VCO frequency above the input reference frequency to the PFD by the M factor	4, 5, 8, 10, 12, 16, 20, 25
L (PFD)	VCO post-scale counter to divide the VCO output frequency by the L factor in the LTR loop	1, 2, 4, 8
L (phase detector)	VCO post-scale counter to divide the VCO output frequency by the L factor in the LTD loop	1, 2, 4, 8

Note to Table 2–4:

(1) The Quartus® II software automatically selects the appropriate counter values for each transceiver configuration.

CDR PLL Operation

The CDR PLL independently recovers the clock and data from the incoming serial data and sends the clock and data to the deserializer. The CDR PLL supports the full range of data rates.

The CDR PLL requires offset cancellation to correct the analog offset voltages that may exist from process variations between the positive and negative differential signals in the CDR circuitry.

For a detailed description of the offset cancellation process, refer to the *Dynamic Reconfiguration in Arria V Devices* chapter.

The CDR PLL operates either in LTR mode or LTD mode. After power-up or reset of the receiver PMA, the CDR PLL must first operate in LTR mode to keep the VCO output frequency close to the optimum recovered clock rate.

In LTR mode, the phase detector is not active. When the CDR PLL locks to the input reference clock, you can switch the CDR PLL to LTD mode to recover the clock and data from the incoming serial data.

In LTD mode, the PFD output is not valid and may cause the lock detect status indicator to toggle randomly. When there is no transition on the incoming serial data for an extended duration, you must switch the CDR PLL to LTR mode to wait for the real serial data.

The time needed for the CDR PLL to lock to data depends on the transition density and jitter of the incoming serial data and the parts per million (ppm) difference between the receiver input reference clock and the upstream transmitter reference clock. The receiver PCS must be held in reset until the CDR PLL locks to data and produces a stable recovered clock.

The LTR/LTD controller directs the CDR PLL transition between the LTR and LTD modes. The controller supports operation in both automatic lock mode and manual lock mode.

CDR PLL in Automatic Lock Mode

In automatic lock mode, the LTR/LTD controller directs the transition between the LTR and LTD modes when a set of conditions are met to ensure proper CDR PLL operation. The mode transitions are indicated by the `rx_is_lockedtodata` signal.

After power-up or reset of the receiver PMA, the CDR PLL is directed into LTR mode. The controller transitions the CDR PLL from LTR to LTD mode when all the following conditions are met:

- The frequency of the CDR PLL output clock and input reference clock is within the configured ppm frequency threshold setting.
- The phase of the CDR PLL output clock and input reference clock is within approximately 0.08 unit interval (UI) of difference.
- In PCIe configurations only—the signal detect circuitry must also detect the presence of the signal level at the receiver input above the threshold voltage specified in the PCI Express Base Specification 2.0.

The controller transitions the CDR PLL from LTD to LTR mode when either of the following conditions is met:

- The frequency of the CDR PLL output clock and input reference clock exceeds the configured ppm frequency threshold setting.
- In PCIe configurations only—the signal detect circuitry detects the signal level at the receiver input below the threshold voltage specified in the PCI Express Base Specification 2.0.

If there is no transition on the incoming serial data for an extended duration, the CDR output clock may drift to a frequency exceeding the configured ppm threshold when compared with the input reference clock. In such a case, the LTR/LTD controller transitions the CDR PLL from LTD to LTR mode.

CDR PLL in Manual Lock Mode

In manual lock mode, the LTR/LTD controller directs the transition between the LTR and LTD modes based on user-controlled settings in the `pma_rx_set_locktodata` and `pma_rx_set_locktoref` registers. Manual lock mode provides the flexibility to manually control the CDR PLL mode transitions bypassing the ppm detection as required by certain applications that include, but not limited to, the following:

- Link with frequency differences between the upstream transmitter and the local receiver clocks exceeding the CDR PLL ppm threshold detection capability. For example, a system with asynchronous spread-spectrum clocking (SSC) downspread of –0.5% where the SSC modulation results in a ppm difference of up to 5000.
- Link that requires a faster CDR PLL transition to LTD mode, avoiding the duration incurred by the ppm detection in automatic lock mode.

In manual lock mode, your design must include a mechanism—similar to a ppm detector—that ensures the CDR PLL output clock is kept close to the optimum recovered clock rate before recovering the clock and data. Otherwise, the CDR PLL might not achieve locking to data. If the CDR PLL output clock frequency is detected as not close to the optimum recovered clock rate in LTD mode, direct the CDR PLL to LTR mode.

For information about the proper sequence after power-up reset, refer to the *Transceiver Reset Control in Arria V Devices* chapter.

Deserializer

The deserializer provides serial-to-parallel data conversion and assumes the data is received LSB first from the receiver buffer. Additionally, the deserializer provides the clock-slip feature.

Clock-Slip

Word alignment in the PCS may contribute up to one parallel clock cycle of latency uncertainty. The clock-slip feature allows word alignment operation with a reduced latency uncertainty by performing the word alignment function in the deserializer. Use the clock slip feature for applications that require deterministic latency.

The deterministic latency state machine in the word aligner from the PCS automatically controls the clock-slip operation. After completing the clock-slip process, the deserialized data is word-aligned into the receiver PCS.

Transmitter PCS Datapath

This section describes the transmitter phase compensation FIFO, byte serializer, 8B/10B encoder, and transmitter bit-slip blocks in the transmitter PCS datapaths.

Transmitter Phase Compensation FIFO

The transmitter phase compensation FIFO is four words deep and interfaces the control and data signals between the transmitter PCS and FPGA fabric or PCIe hard IP block. The FIFO supports the following operations:

- Phase compensation mode with various clocking modes on the read clock and write clock
- Registered mode with only one clock cycle of datapath latency

Phase Compensation Mode

The transmitter phase compensation FIFO compensates any phase difference between the read and write clocks for the transmitter control and data signals. The low-speed parallel clock feeds the read clock; the FPGA fabric interface clock feeds the write clock. The clocks must have 0 ppm difference in frequency or a FIFO underrun or overflow condition may result.

The transmitter phase compensation FIFO supports various clocking modes on the read and write clocks depending on the transceiver configuration.

For a detailed description of transmitter datapath interface clocking modes when using the transmitter phase compensation FIFO, refer to the *Transceiver Clocking in Arria V Devices* chapter.

Registered Mode

To eliminate the FIFO latency uncertainty for applications with stringent datapath latency uncertainty requirements, bypass the FIFO functionality in registered mode to incur only one clock cycle of datapath latency when interfacing the transmitter channel to the FPGA fabric. Configure the FIFO to registered mode when interfacing the transmitter channel to the PCIe hard IP block to reduce datapath latency. In registered mode, the low-speed parallel clock that is used in the transmitter PCS clocks the FIFO.

Byte Serializer

The byte serializer allows the transmitter channel to operate at higher data rates in a configuration that exceeds the FPGA fabric–transceiver interface frequency limit. The byte serializer supports operation in single- and double-width modes. The datapath clock rate at the output of the byte serializer is twice the FPGA fabric–transmitter interface clock frequency.

You must use the byte serializer in configurations that exceed the maximum frequency limit of the FPGA fabric–transceiver interface.

Table 2–5 lists the byte serializer datapath conversion for the transmitter input datapath width in single- and double-width modes.

Table 2–5. Transmitter Input Datapath Conversion

Mode	Transmitter Input Datapath Width	Byte Serializer Output Datapath Width	Byte Serializer Output Ordering
Single Width	16	8	Least significant 8 bits of the 16-bit input first
	20	10	Least significant 10 bits of the 20-bit input first
Double Width	32	16	Least significant 16 bits of the 32-bit input first
	40	20	Least significant 20 bits of the 40-bit input first

8B/10B Encoder

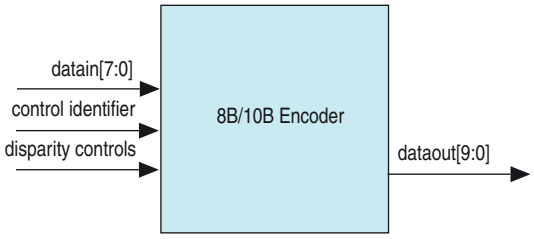
The 8B/10B encoder supports operation in single- and double-width modes with the running disparity control feature.

8B/10B Encoder in Single-Width Mode

In single-width mode, the 8B/10B encoder generates 10-bit code groups from 8-bit data and 1-bit control identifier with proper disparity according to the PCS reference diagram in Clause 36 of the IEEE 802.3 specification. The 10-bit code groups are generated as valid data code-groups (/Dx.y/) or special control code-groups (/Kx.y/), depending on the 1-bit control identifier.

Figure 2–5 shows a simplified diagram of the 8B/10B encoder in single-width mode.

Figure 2–5. 8B/10B Encoder Diagram in Single-Width Mode



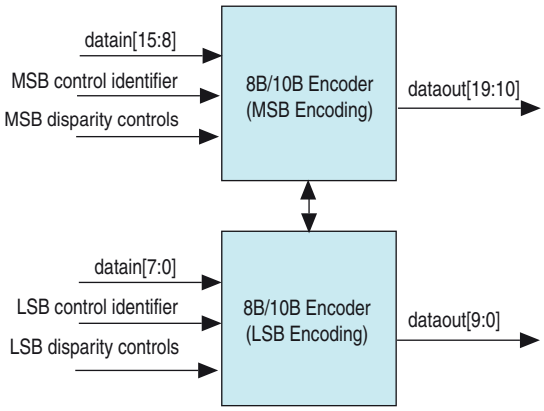
The IEEE 802.3 specification identifies only 12 sets of 8-bit characters as /Kx.y/. If other sets of 8-bit characters are set to encode as special control code-groups, the 8B/10B encoder may encode the output 10-bit code as an invalid code (it does not map to a valid /Dx.y/ or /Kx.y/ code), or unintended valid /Dx.y/ code, depending on the value entered.

8B/10B Encoder in Double-Width Mode

In double-width mode, two 8B/10B encoders are cascaded to generate two sets of 10-bit code groups from 16-bit data and two 1-bit control identifiers. When receiving the 16-bit data, the 8-bit LSByte is encoded first, followed by the 8-bit MSByte.

Figure 2–6 shows a simplified diagram of the 8B/10B encoder in double-width mode.

Figure 2–6. 8B/10B Encoder Diagram in Double-Width Mode



Running Disparity Control

The 8B/10B encoder automatically performs calculations that meet the running disparity rules when generating the 10-bit code groups. The running disparity control feature provides user-controlled signals (`tx_dispval` and `tx_forcedisp`) to manually force encoding into a positive or negative current running disparity code group. When enabled, the control overwrites the current running disparity value in the encoder based on user-controlled signals, regardless of the internally-computed current running disparity in that cycle.

Using the running disparity control may temporarily cause a running disparity error at the receiver.

Encoder Output During Reset Sequence

Figure 2–7 shows the 8B/10B encoder output during and after reset conditions in both single- and double-width modes.

Figure 2–7. 8B/10B Encoder Output During and After Reset Conditions

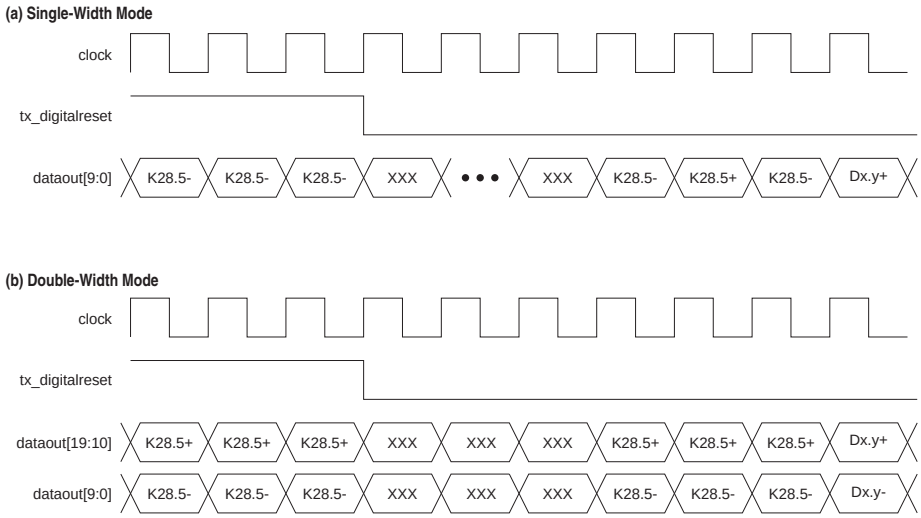


Table 2–6 lists the 8B/10B encoder output during and after reset conditions.

Table 2–6. 8B/10B Encoder Output During and After Reset Conditions

Operation Mode	During 8B/10B Reset	After 8B/10B Reset Release
Single Width	Continuously sends the /K28.5/ code from the RD– column	Some “don’t cares” due to pipelining in the transmitter channel, followed by three /K28.5/ codes with proper disparity—starts with negative disparity—before sending encoded 8-bit data at its input.
Double Width	Continuously sends the /K28.5/ code from the RD– column on LSByte and the /K28.5/ code from the RD+ column on MSByte	Some “don't cares” due to pipelining in the transmitter channel, followed by: ■ Three /K28.5/ codes from the RD– column before sending encoded 8-bit data at its input on LSByte. ■ Three /K28.5/ codes from the RD+ column before sending encoded 8-bit data at its input on MSByte.

Transmitter Bit-Slip

The transmitter bit-slip enables a bit-level delay insertion to the data prior to serialization for the serial transmission. The transmitter bit-slip supports operation in single- and double-width modes. Each bit slipped at the transmitter incurs one serial bit of datapath latency. Table 2–7 lists the number of bits allowed to be slipped with the `tx_bitslipboundaryselect` signal.

Table 2–7. Bits Slip Allowed with the tx_bitslipboundaryselect Signal

Operation Mode	Maximum Bit-Slip Setting
Singe width (8- or 10-bit)	9
Double width (16- or 20-bit)	19

Receiver PCS Datapath

This section describes the word aligner, deskew FIFO, rate match FIFO, 8B/10B decoder, byte deserializer, byte ordering, and receiver phase compensation FIFO blocks in the receiver PCS datapaths.

Word Aligner

Parallel data at the input of the receiver PCS loses the word boundary of the upstream transmitter from the serial-to-parallel conversion in the deserializer. The word aligner provides word boundary restoration during link synchronization with the following four modes:

- Manual alignment—automatic synchronization to new word boundary
- Bit-slip—manual data slip control
- Automatic synchronization state machine—automatic synchronization to a new word boundary with the programmable state machine for hysteresis control
- Deterministic latency state machine—automatic synchronization to a new word boundary for applications with a stringent latency uncertainty requirement

The word aligner searches for a predefined alignment pattern in the deserialized data to identify the correct boundary and restores the word boundary during link synchronization. The alignment pattern is predefined for standard serial protocols according to the respective protocol specifications to achieve synchronization or you can specify the settings with a custom word alignment pattern for proprietary protocol implementations. Except for bit-slip mode, after completing word alignment, the deserialized data is synchronized to have the word alignment pattern at the LSB portion of the aligned data.

In addition to restoring the word boundary, the word aligner supports the optional features listed in Table 2–8.

Table 2–8. Optional Word Aligner Features

Feature	Availability
Programmable Run-Length Violation Detection	All transceiver configurations
Receiver Polarity Inversion	All transceiver configurations except PCIe
Receiver Bit Reversal	Custom single- and double-width configurations only
Receiver Byte Reversal	Custom double-width configuration only

The operation mode and alignment pattern length support varies depending on the word aligner configurations. Table 2–9 lists the operation mode and alignment pattern length support in single- and double-width configurations.

Table 2–9. Word Aligner Operation Mode and Pattern Length Support

PCS Mode	PMA–PCS Interface Width	Word Aligner Mode	Alignment Pattern Length
Singe Width	8 bits	Manual alignment	8 or 16 bits
		Bit-slip	16 bits
	10 bits	Manual alignment	7 or 10 bits
		Bit-slip	7 or 10 bits
		Automatic synchronization state machine	7 or 10 bits
		Deterministic latency state machine	10 bits
Double Width	16 bits	Manual alignment	8, 16, or 32 bits
		Bit-slip	8, 16, or 32 bits
	20 bits	Manual alignment	7, 10, or 20 bits
		Bit-slip	7, 10, or 20 bits
		Deterministic latency state machine	10 or 20 bits

Word Aligner in Manual Alignment Mode

In manual alignment mode, the word alignment operation is manually controlled with the `rx_enapatternalign` register. Depending on the configuration, controlling the `rx_enapatternalign` register enables the word aligner to look for the predefined word alignment pattern in the received data stream and automatically synchronizes to the new word boundary.

Table 2–10 lists the word aligner operations in manual alignment mode.

Table 2–10. Word Aligner Operations in Manual Alignment Mode

PCS Mode	PMA–PCS Interface Width	Word Alignment Operation
Single Width	8 bits	1. After the rx_digitalreset signal deasserts, a 0-to-1 transition to the rx_enapatternalign register triggers the word aligner to look for the predefined word alignment pattern in the received data stream and automatically synchronize to the new word boundary. 2. Any alignment pattern found thereafter in a different word boundary does not cause the word aligner to resynchronize to this new word boundary because there is a lack of a preceding 0-to-1 transition on the rx_enapatternalign register. 3. To resynchronize to the new word boundary, create a 0-to-1 transition to the rx_enapatternalign register. 4. If you set the rx_enapatternalign register to 1 before the deassertion of the rx_digitalreset signal, the word aligner updates the word boundary when the first alignment pattern is found, even though a 0-to-1 transition was not explicitly generated. 5. To resynchronize to the new word boundary, create a 0-to-1 transition to the rx_enapatternalign register.
	10 bits	1. After the rx_digitalreset signal deasserts, setting 1 to the rx_enapatternalign register triggers the word aligner to look for the predefined word alignment pattern, or its complement in the received data stream, and automatically synchronizes to the new word boundary. 2. Any alignment pattern found thereafter in a different word boundary causes the word aligner to resynchronize to this new word boundary if the rx_enapatternalign register remains set to 1. 3. If you set the rx_enapatteralign register to 0, the word aligner maintains the current word boundary even when it finds the alignment pattern in a new word boundary.
Double Width	16 bits	1. After the rx_digitalreset signal deasserts, regardless of the setting in the rx_enapatternalign register, the word aligner synchronizes to the first predefined alignment pattern found. 2. Any alignment pattern found thereafter in a different word boundary does not cause the word aligner to resynchronize to this new word boundary. 3. To resynchronize to the new word boundary, create a 0-to-1 transition to the rx_enapatternalign register.
	20 bits	

Bit-Slip Mode

In bit-slip mode, the word alignment is achieved by manually controlling the data slip with the rx_bitslip signal. Slipping the received data by one bit effectively shifts the word boundary by one bit. You can implement a controller in the FPGA fabric to iteratively control the rx_bitslip signal until the word aligner output matches the predefined word alignment pattern to achieve synchronization. Table 2–11 lists the word aligner operations in bit-slip mode.

Table 2–11. Word Aligner Operations in Bit-Slip Mode

PCS Mode	PMA–PCS Interface Width	Word Alignment Operation
Single Width	8 bits	1. At every rising edge to the rx_bitslip signal, the word aligner slips one bit into the received data. 2. When bit-slipping shifts a complete round of the data bus width, the word boundary is back to the original boundary. 3. Use the rx_patterndetect signal assertion or check the data output to indicate completion of the alignment process—where the word aligner output matches the predefined alignment pattern.
	10 bits	
Double Width	16 bits	
	20 bits	

For every bit slipped in the word aligner, the earliest bit received is lost.

Word Aligner in Automatic Synchronization State Machine Mode

In automatic synchronization state machine mode, a programmable state machine determines when the word aligner has either achieved synchronization or lost synchronization. You can configure the state machine to provide hysteresis control during link synchronization and throughout normal link operation. Depending on your protocol configurations, the state machine parameters are automatically configured to be compliant to the synchronization state machine specified in the respective protocol specification.

Table 2–12 lists the programmable state machine parameters for the word aligner in automatic synchronization state machine mode.

Table 2–12. State Machine Parameters for the Word Aligner in Automatic Synchronization State Machine Mode

Parameter	Values
Number of valid synchronization code groups or ordered sets received to achieve synchronization	1–256
Number of erroneous code groups received to lose synchronization	1–64
Number of continuous good code groups received to reduce the error count by one	1–256

Table 2–13 lists the word aligner operations in automatic synchronization state machine mode.

Table 2–13. Word Aligner Operation in Automatic Synchronization State Machine Mode

PCS Mode	PMA–PCS Interface Width	Word Alignment Operation
Single Width	10 bits	1. After the rx_digitalreset signal deasserts, the word aligner starts looking for the predefined word alignment pattern, or its complement, in the received data stream and automatically aligns to the new word boundary. 2. Synchronization is achieved only after the word aligner receives the programmed number of valid synchronization code groups in the same word boundary and is indicated by the assertion of the rx_syncstatus signal. 3. After assertion and achieving synchronization, the rx_syncstatus signal remains asserted until the word aligner loses synchronization. 4. Loss of synchronization occurs when the word aligner receives the programmed number of erroneous code groups without receiving the intermediate good code groups and is indicated by the deassertion of the rx_syncstatus signal. 5. The word aligner may achieve synchronization again after receiving a new programmed number of valid synchronization code groups in the same word boundary.

Word Aligner in Deterministic Latency State Machine Mode

In deterministic latency state machine mode, word alignment is achieved by performing a clock-slip in the deserializer until the deserialized data coming into the receiver PCS is word-aligned. The state machine controls the clock-slip process in the deserializer after the word aligner has found the alignment pattern and identified the word boundary. Deterministic latency state machine mode offers a reduced latency uncertainty in the word alignment operation for applications that require deterministic latency.

Table 2–14 lists the word aligner operations in deterministic latency state machine mode.

Table 2–14. Word Aligner Operations in Deterministic Latency State Machine Mode

PCS Mode	PMA–PCS Interface Width	Word Alignment Operation
Single Width	10 bits	1. After the rx_digitalreset signal deasserts, a 0-to-1 transition to the rx_enapatternalign register triggers the word aligner to look for the predefined word alignment pattern or its complement in the received data stream. 2. After the pattern is found and the word boundary is identified, the state machine controls the deserializer to clock-slip the boundary-indicated number of serial bits. 3. When the clock-slip is complete, the deserialized data coming into the receiver PCS is word-aligned and is indicated by the assertion of the rx_syncstatus signal.
Double Width	20 bits	

Programmable Run-Length Violation Detection

The programmable run-length violation detection circuit detects if the number of consecutive 1s or 0s in the received data exceeds the user-specified threshold. Table 2–15 lists the detection capabilities of the run-length violation circuit.

Table 2–15. Detection Capabilities of the Run-Length Violation Circuit

PCS Mode	PMA–PCS Interface Width	Run-Length Violation Detector Range	
		Minimum	Maximum
Single Width	8 bits	4	128
	10 bits	5	160
Double Width	16 bits	8	512
	20 bits	10	640

Receiver Polarity Inversion

The positive and negative signals of a serial differential link might accidentally be swapped during the layout of the board. The polarity inversion feature at the receiver can correct this error without requiring board respin or major updates to the logic in the FPGA fabric. The polarity inversion feature inverts the polarity of every bit at the input to the word aligner, which has the same effect as swapping the positive and negative signals of the serial differential link.

Inversion is controlled dynamically with the `rx_invpolarity` register. Enabling the polarity inversion feature may cause initial disparity errors at the receiver with the 8B/10B-coded data. The receiver must be able to tolerate these disparity errors.

If the polarity inversion is enabled midway through a word, the word will be corrupted.

Bit Reversal

You can reverse the bit order at the output of the word aligner for receiving a MSB-to-LSB transmission using the bit reversal feature at the receiver. Table 2–16 lists the bit reversal feature options.

Table 2–16. Bit Reversal Feature

Bit Reversal Option	Received Bit Order	
	Single-Width Mode (8- or 10-bit)	Double-Width Mode (16- or 20-bit)
Disabled (default)	LSB to MSB	LSB to MSB
Enabled	MSB to LSB For example: 8-bit—`D[7:0]` rewired to `D[0:7]` 10-bit—`D[9:0]` rewired to `D[0:9]`	MSB to LSB For example: 16-bit—`D[15:0]` rewired to `D[0:15]` 20-bit—`D[19:0]` rewired to `D[0:19]`

When receiving the MSB-to-LSB transmission, the word aligner receives the data in reverse order. The word alignment pattern must be reversed accordingly to match the MSB-first incoming data ordering.

You can dynamically control the bit reversal feature using the `rx_bitreversal_enable` register with the word aligner in bit-slip mode. When you dynamically enable the bit reversal feature in bit-slip mode, you can ignore the pattern detection function in the word aligner because the word alignment pattern cannot be dynamically reversed to match the MSB-first incoming data ordering.

Receiver Byte Reversal

The two symbols of incoming data at the receiver in double-width mode may be accidentally swapped during transmission. For a 16-bit input data width at the word aligner, the two symbols are `bits[15:8]` and `bits[7:0]`. For a 20-bit input data width at the word aligner, the two symbols are `bits[19:10]` and `bits[9:0]`. The byte reversal feature at the word aligner output can correct this error by swapping the two symbols in double-width mode at the word aligner output, as listed in Table 2–17.

Table 2–17. Byte Reversal Feature

Byte Reversal Option	Word Aligner Output	
	16-bit Data Width	20-bit Data Width
Disabled	`D[15:0]`	`D[19:0]`
Enabled	`D[7:0]`, `D[15:8]`	`D[9:0]`, `D[19:10]`

The reversal is controlled dynamically using the `rx_bytereversal_enable` register. Enabling the byte reversal option may cause initial disparity errors at the receiver with 8B/10B-coded data. The receiver must be able to tolerate these disparity errors.

When receiving swapped symbols, the word alignment pattern must be byte-reversed accordingly to match the incoming byte-reversed data.

Deskew FIFO

The code groups received across multilane links can be misaligned with each other because of the skew in the physical transmission medium or the differences between the independent clock recoveries per lane. The deskew FIFO is 16 words deep, which aligns the code groups between up to four receiver channels bonded for a multilane link configuration. The FIFO can handle up to eight bytes of code group skew between the channels.

FIFO operation is controlled with a state machine that supports lane deskew operation, as required by the XAUI protocol specification. The FIFO supports operation in single-width mode.

For details about FIFO operation for the XAUI protocol, refer to the "PCS deskew state diagram" specified in Clause 48 of the *IEEE Std 802.3™-2008* specification.

For details about deskew FIFO clocking when used for the XAUI protocol, refer to the *Transceiver Protocol Configurations in Arria V Devices* chapter.

Rate Match FIFO

In a link where the upstream transmitter and local receiver can be clocked with independent reference clock sources, the data can be corrupted by any frequency differences (in ppm count) when crossing the data from the recovered clock domain—the same clock domain as the upstream transmitter reference clock—to the local receiver reference clock domain.

The rate match FIFO is 20 words deep, which compensates for the small clock frequency differences of up to ±300 ppm (600 ppm total) between the upstream transmitter and the local receiver clocks by performing symbol insertion or deletion, depending on the ppm difference on the clocks.

The rate match FIFO operation requires that the transceiver channel is in duplex configuration (both transmit and receive functions) and a predefined 20-bit pattern (consisting of a 10-bit control pattern and a 10-bit skip pattern). The 10-bit skip pattern must be chosen from a code group with neutral disparity.

The FIFO operates by looking for the 10-bit control pattern, followed by the 10-bit skip pattern in the data after the word aligner has restored the word boundary. After finding the pattern, the FIFO performs the following operations to ensure the FIFO does not underflow or overflow:

- Inserts the 10-bit skip pattern when the local receiver reference clock frequency is greater than the upstream transmitter reference clock frequency
- Deletes the 10-bit skip pattern when the local receiver reference clock frequency is less than the upstream transmitter reference clock frequency

The rate match FIFO supports operations in single- and double-width modes. You can define the 20-bit pattern for custom configurations. For protocol configurations, the FIFO is automatically configured to support a clock rate compensation function as required by the following specifications:

- The PCIe protocol per clock tolerance compensation requirement, as specified in the PCI Express Base Specification 2.0 for Gen1 and Gen2 signaling rates
- The Gbps Ethernet (GbE) protocol per clock rate compensation requirement using an idle ordered set, as specified in Clause 36 of the IEEE 802.3 specification
- The XAUI protocol per clock rate compensation requirement using the ||R|| ordered set, as specified in Clause 48 of the IEEE 802.3 specification

For more information about the rate match FIFO operation in custom and protocol-specific configurations, refer to the *Transceiver Custom Configurations in Arria V Devices* and the *Transceiver Protocol Configurations in Arria V Devices* chapters, respectively.

8B/10B Decoder

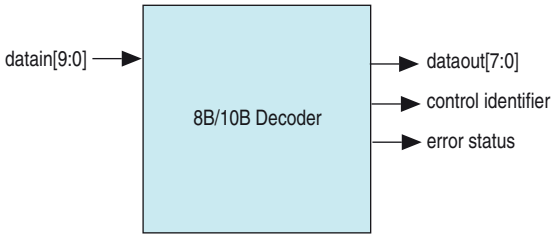
The 8B/10B decoder supports operation in single- and double-width modes.

8B/10B Decoder in Single-Width Mode

In single-width mode, the 8B/10B decoder decodes the received 10-bit code groups into an 8-bit data and a 1-bit control identifier in compliance with Clause 36 in the IEEE 802.3 specification. The 1-bit control identifier indicates if the decoded 8-bit code is a valid data or special control code.

Figure 2–8 shows a simplified 8B/10B decoder block diagram in single-width mode.

Figure 2–8. 8B/10B Decoder Block Diagram in Single-Width Mode

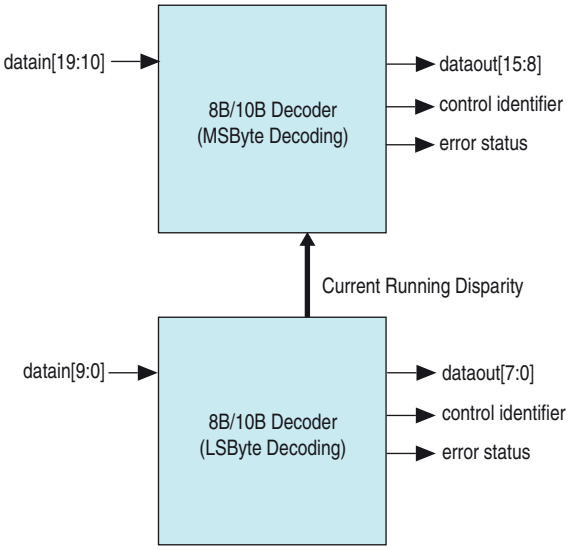


8B/10B Decoder in Double-Width Mode

In double-width mode, two 8B/10B decoders are cascaded to decode the 20-bit code groups into two sets of 8-bit data and two 1-bit control identifiers. When receiving the 20-bit code group, the 10-bit LSByte is decoded first and the ending running disparity is forwarded to the other 8B/10B decoder for decoding the 10-bit MSByte.

Figure 2–9 shows a simplified 8B/10B decoder block diagram in double-width mode.

Figure 2–9. 8B/10B Decoder Block Diagram in Double-Width Mode



Byte Deserializer

The byte deserializer allows the receiver channel to operate at higher data rates in a configuration that exceeds the FPGA fabric–transceiver interface frequency limit. The byte deserializer supports operation in single- and double-width modes. The datapath clock rate at the input of the byte deserializer is twice the FPGA fabric–receiver interface clock frequency.

You must use the byte deserializer in configurations that exceed the maximum frequency limit of the FPGA fabric–transceiver interface.

After byte deserialization, the word alignment pattern may be ordered in the MSByte or LSByte position.

For applications that require deterministic latency, you can configure the byte deserializer to order the word alignment pattern only in the LSByte position after byte deserialization. To achieve this, the data byte prior to the alignment pattern is dropped if the word alignment pattern is found in the MSByte position. In this configuration, there is no latency uncertainty in the byte deserializer operation.

Table 2–18 lists the byte deserializer conversion for the byte deserializer input datapath width in single- and double-width modes. The data is assumed to be received as LSByte first—the least significant 8 or 10 bits in single-width mode or the least significant 16 or 20 bits in double-width mode.

Table 2–18. Byte Deserializer Input Datapath Width Conversion

Mode	Byte Deserializer Input Datapath Width	Receiver Output Datapath Width
Single Width	8	16
	10	20
Double Width	16	32
	20	40

Byte Ordering

When you enable the byte deserializer, the output byte order may not match the originally transmitted ordering. For applications that require a specific pattern to be ordered at the LSByte position of the data, byte ordering can restore the proper byte order of the byte-deserialized data before forwarding it to the FPGA fabric.

Byte ordering operates by inserting a predefined pad pattern to the byte-deserialized data if the predefined byte ordering pattern found is not in the LSByte position.

Byte ordering requires the following:

- A receiver with the byte deserializer enabled
- A predefined byte ordering pattern that must be ordered at the LSByte position of the data
- A predefined pad

Byte ordering supports operation in single- and double-width modes. Both modes support operation in word aligner-based and manual ordering modes.

Byte Ordering in Single-Width Mode

Table 2–19 lists the byte odering operation in single-width mode.

Table 2–19. Byte Ordering Operation in Single-Width Mode (Part 1 of 2) (1)

PMA–PCS Interface Width	FPGA Fabric–Transceiver Interface Width	8B/10B Decoder	Byte Ordering Pattern Length	Pad Pattern Length
8 bits	16 bits	Disabled	8 bits	8 bits

Table 2–19. Byte Ordering Operation in Single-Width Mode (Part 2 of 2) (1)

PMA–PCS Interface Width	FPGA Fabric–Transceiver Interface Width	8B/10B Decoder	Byte Ordering Pattern Length	Pad Pattern Length
10 bits	16 bits	Enabled	9 bits (2)	9 bits (2)
	20 bits	Disabled	10 bits	10 bits

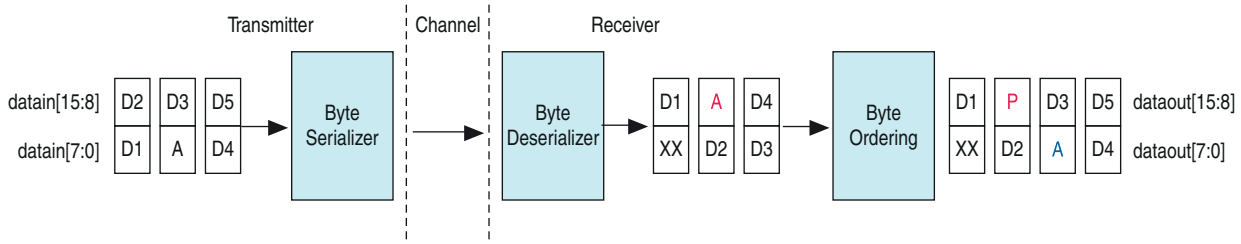
Notes to Table 2–19:

(1) Byte ordering is supported only when you enable the byte deserializer.

(2) The MSB of the 9-bit pattern represents the 1-bit control identifier of the 8B/10B-decoded data. The lower 8 bits represent the 8-bit decoded code.

Figure 2–10 shows an example of a byte ordering operation in single-width mode (8-bit PMA-PCS interface width) where A is the predefined byte ordering pattern and P is the predefined pad pattern.

Figure 2–10. Byte Ordering Operation Example in Single-Width Mode



Byte Ordering in Double-Width Mode

Table 2–20 lists the byte ordering operation in double-width mode.

Table 2–20. Byte Ordering Operation in Double-Width Mode (1)

PMA–PCS Interface Width	FPGA Fabric–Transceiver Interface Width	8B/10B Decoder	Byte Ordering Pattern Length	Pad Pattern Length
16 bits	32 bits	Disabled	8 or 16 bits	8 bits
20 bits	32 bits	Enabled	9 (2) or 18 bits (3)	9 bits (2)
	40 bits	Disabled	10 or 20 bits	10 bits

Notes to Table 2–20:

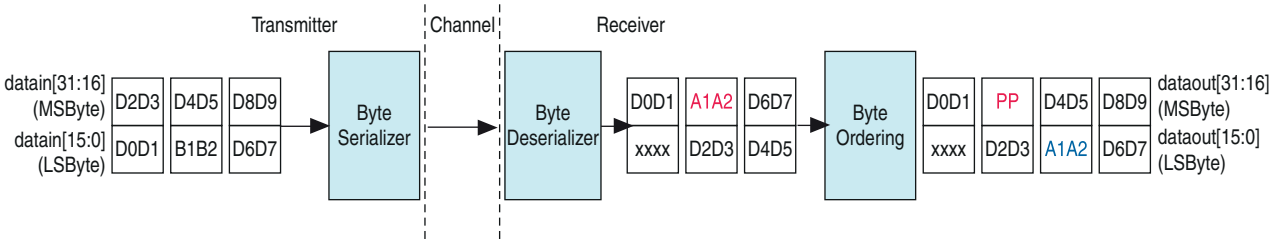
(1) Byte ordering is supported only when you enable the byte deserializer.

(2) The MSB of the 9-bit pattern represents the 1-bit control identifier of the 8B/10B-decoded data. The lower 8 bits represent the 8-bit decoded code.

(3) The 18-bit pattern consists of two sets of 9-bit patterns, individually represented as in Note (2).

Figure 2–11 shows an example of a byte ordering operation in double-width mode (16-bit PMA-PCS interface width) where A1A2 is the predefined byte ordering pattern and P is the predefined pad pattern.

Figure 2–11. Byte Ordering Operation Example in Double-Width Mode



Word Aligner-Based Ordering Mode

In word aligner-based ordering mode, the byte ordering operation is controlled by the word aligner synchronization status signal, `rx_syncstatus`. After a rising edge on the `rx_syncstatus` signal, byte ordering looks for the byte ordering pattern in the byte-deserialized data.

When the first data byte that matches the byte ordering pattern is found, the byte ordering performs the following operations:

- If the pattern is not in the LSByte position—byte ordering inserts the appropriate number of pad patterns to push the byte ordering pattern to the LSByte position and indicates the byte alignment.
- If the pattern is in the LSByte position—byte ordering indicates the byte alignment.

Any byte misalignment found thereafter is ignored unless another rising edge on the `rx_syncstatus` signal, indicating resynchronization, is observed.

Manual Ordering Mode

In manual ordering mode, the byte ordering operation is controlled using the `rx_enabyteord` signal. A rising edge on the `rx_enabyteord` signal triggers byte ordering to look for the byte ordering pattern in the byte-deserialized data.

When the first data byte that matches the byte ordering pattern is found, the byte ordering performs the following operations:

- If the pattern is not in the LSByte position—byte ordering inserts the appropriate number of pad patterns to push the byte ordering pattern to the LSByte position and indicates the byte alignment.
- If the pattern is in the LSByte position—byte ordering indicates the byte alignment.

Any byte misalignment found thereafter is ignored unless another rising edge on the `rx_enabyteord` signal is observed.

Receiver Phase Compensation FIFO

The receiver phase compensation FIFO is four words deep and interfaces the status and data signals between the receiver PCS and the FPGA fabric or the PCIe hard IP block. The FIFO supports the following operations:

- Phase compensation mode with various clocking modes on the read clock and write clock
- Registered mode with only one clock cycle of datapath latency

Phase Compensation Mode

The receiver phase compensation FIFO compensates for any phase difference between the read and write clocks for the receiver status and data signals. The low-speed parallel clock feeds the write clock; the FPGA fabric interface clock feeds the read clock. The clocks must have 0 ppm difference in frequency or a FIFO underrun or overflow condition may result.

The receiver phase compensation FIFO supports various clocking modes on the read and write clocks depending on the transceiver configuration.

For a detailed description of the receiver datapath interface clocking modes when using the receiver phase compensation FIFO, refer to the *Transceiver Clocking in Arria V Devices* chapter.

Registered Mode

To eliminate the FIFO latency uncertainty for applications with stringent datapath latency uncertainty requirements, bypass the FIFO functionality in registered mode to incur only one clock cycle of datapath latency when interfacing the receiver channel to the FPGA fabric. Configure the FIFO to registered mode when interfacing the receiver channel to the PCIe hard IP block to reduce datapath latency. In registered mode, the low-speed parallel clock that is used in the receiver PCS clocks the FIFO.

Transceiver Clocking

This section provides basic information about the Arria V transceiver clocking architecture.

The information in this section should be used in conjunction with the *Transceiver Clocking in Arria V Devices* chapter.

FPGA Fabric–Transceiver Interface Clocking

The FPGA fabric–transceiver interface clocks can be subdivided into the following three categories:

- Input reference clocks—The input reference clock can be an FPGA fabric–transceiver interface clock when it is also forwarded to the FPGA fabric to clock logic in the FPGA fabric.

- Transceiver datapath interface clocks—Used to transfer data, control, and status signals between the FPGA fabric and the transceiver channels. The transceiver channel forwards the tx_clkout signal to the FPGA fabric to clock the data and control signals into the transmitter. The transceiver channel also forwards the recovered rx_clkout clock (in configurations without the rate matcher) or the tx_clkout clock (in configurations with the rate matcher) to the FPGA fabric to clock the data and status signals from the receiver into the FPGA fabric.
- Other transceiver clocks—The following transceiver clocks form a part of the FPGA fabric–transceiver interface clocks:
  - mgmt_clk—Avalon®-MM interface clock used for controlling the transceivers, dynamic reconfiguration, and calibration
  - fixed_clk—the 125 MHz fixed-rate clock used in the PCIe (PIPE) receiver detect circuitry

Table 2–21 shows the FPGA fabric–transceiver interface clocks.

Table 2–21. FPGA Fabric–Transceiver Interface Clocks *(1)*

Clock Name	Clock Description	Interface Direction	FPGA Fabric Clock Resource Utilization
pll_ref_clk	Input reference clock used for clocking logic in the FPGA fabric	Transceiver-to-FPGA fabric	GCLK, RCLK, PCLK
tx_clkout	Clock forwarded by the transceiver for clocking the transceiver datapath interface	Transceiver-to-FPGA fabric	GCLK, RCLK, PCLK
rx_clkout	Clock forwarded by the receiver for clocking the receiver datapath interface	Transceiver-to-FPGA fabric	GCLK, RCLK, PCLK
tx_coreclkin	User-selected clock for clocking the transmitter datapath interface	FPGA fabric-to-transceiver	GCLK, RCLK, PCLK
rx_coreclkin	User-selected clock for clocking the receiver datapath interface	FPGA fabric-to-transceiver	GCLK, RCLK, PCLK
fixed_clk	PCIe receiver detect clock	FPGA fabric-to-transceiver	GCLK, RCLK, PCLK
mgmt_clk *(2)*	Avalon-MM interface management clock	FPGA fabric-to-transceiver	GCLK, RCLK, PCLK

Notes to Table 2–21:

(1) For more information about the GCLK, RCLK, and PCLK resources available in each device, refer to the *Clock Networks and PLLs in Arria V Devices* chapter.

(2) The mgmt_clk is a free-running clock that is not derived from the transceiver blocks.

Table 2–22 lists the port names for tx_clkout and rx_clkout.

Table 2–22. Configuration Specific Port Names for tx_clkout and rx_clkout

Configuration	Port Name for tx_clkout	Port Name for rx_clkout
Custom	tx_clkout	rx_clkout
Interlaken	tx_clkout	rx_clkout
Low Latency	tx_clkout	rx_clkout
PCIe	pipe_pclk	pipe_pclk
XAUI	xgmii_tx_clk	xgmii_rx_clk

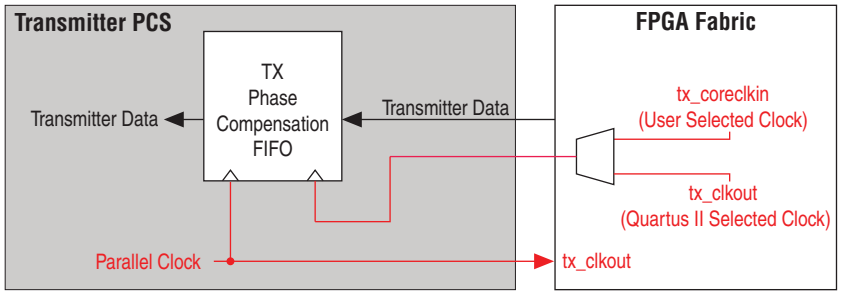
Transmitter Datapath Interface Clocking

For the 6-Gbps transceivers, the write side of the TX phase compensation FIFO makes up the transmitter datapath interface. The transmitter datapath interface clock clocks this interface.

Figure 2–12 shows the 6-Gbps transmitter datapath interface clocking. The transmitter PCS forwards the following clocks to the FPGA fabric:

- `tx_clkout`—for each transmitter channel in a non-bonded configuration
- `tx_clkout[0]`—for all transmitter channels in a bonded configuration

Figure 2–12. Transmitter Datapath Interface Clocking for 6-Gbps Transceivers

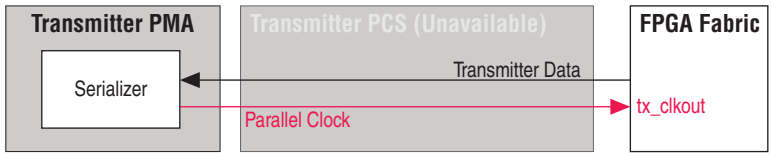


All configurations that uses the PCS channel must have a 0 parts per million (ppm) difference between the transmitter datapath interface clock and the read side clock of the TX phase compensation FIFO.

For the 10-Gbps transceivers, there are no PCS blocks. The only transmit datapath available is a direct connection from the FPGA fabric to the serializer of the transmitter PMA.

Figure 2–13 shows the 10-Gbps transmitter datapath interface clocking. For each transmitter channel in a non-bonded configuration, the FPGA fabric forwards the following `tx_clkout` clock to the transmitter PMA.

Figure 2–13. Transmitter Datapath Interface Clocking for 10-Gbps Transceivers



For more information about interface clocking for each configuration, refer to the *Transceiver Custom Configuration in Arria V Devices* and *Transceiver Protocol Configurations in Arria V Devices* chapters.

You can clock the transmitter datapath interface by one of the following options:

- The Quartus II-selected transmitter datapath interface clock
- The user-selected transmitter datapath interface clock

To reduce GCLK, RCLK, and PCLK resource utilization in your design, you can select the user-selection option to share the transceiver datapath interface clocks.

Quartus II-Selected Transmitter Datapath Interface Clock

The Quartus II software automatically picks the appropriate clock from the FPGA fabric to clock the transmitter datapath interface. Figure 2–14 shows the transmitter datapath interface of two 6-Gbps transceiver non-bonded channels clocked by their respective transmitter PCS clocks, which are forwarded to the FPGA fabric.

Figure 2–14. 6-Gbps Transmitter Datapath Interface Clocking for Non-Bonded Channels

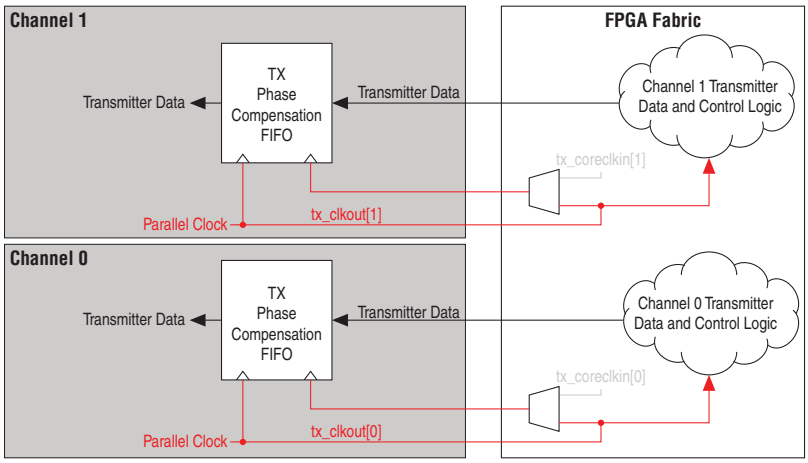


Figure 2–15 shows the transmitter datapath interface of two 10-Gbps transceiver non-bonded channels clocked by their respective transmitter PMA clocks, which are forwarded to the FPGA fabric.

Figure 2–15. 10-Gbps Transmitter Datapath Interface Clocking for Non-Bonded Channels

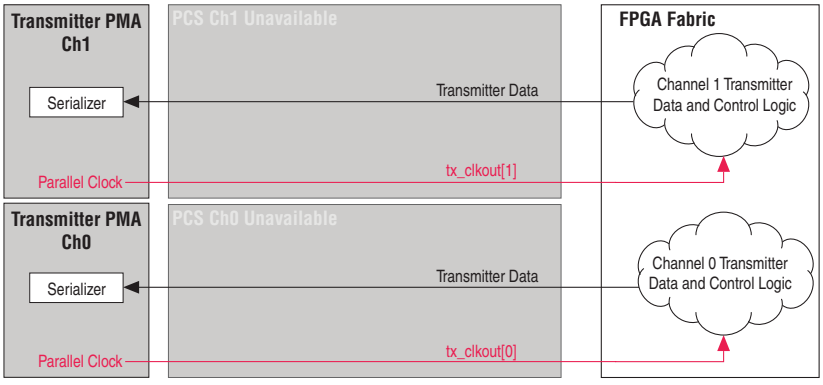
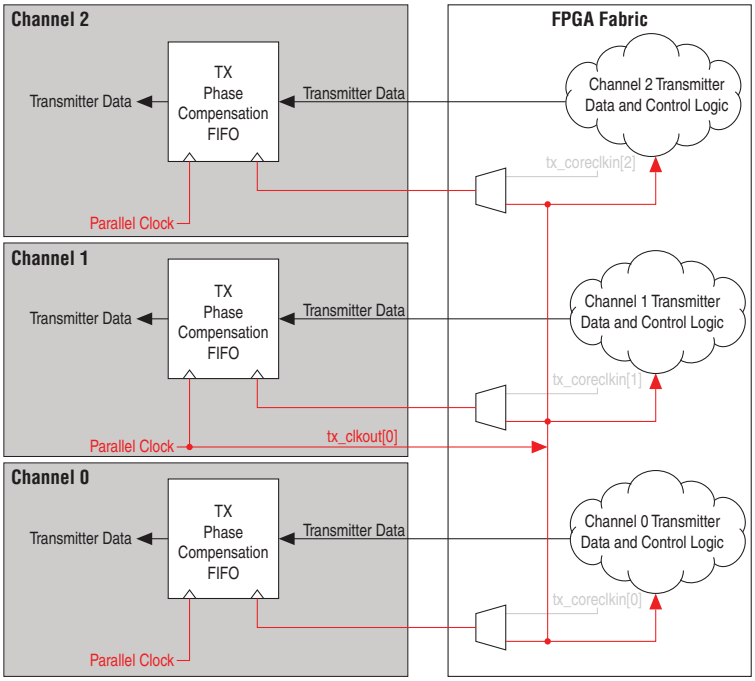


Figure 2–16 shows the 6-Gbps transmitter datapath interface of three bonded channels clocked by the `tx_clkout[0]` clock. The `tx_clkout[0]` clock is derived from the central clock divider of channel 1 or 4 in a transceiver bank.

Figure 2–16. 6-Gbps Transmitter Datapath Interface Clocking for Three Bonded Channels



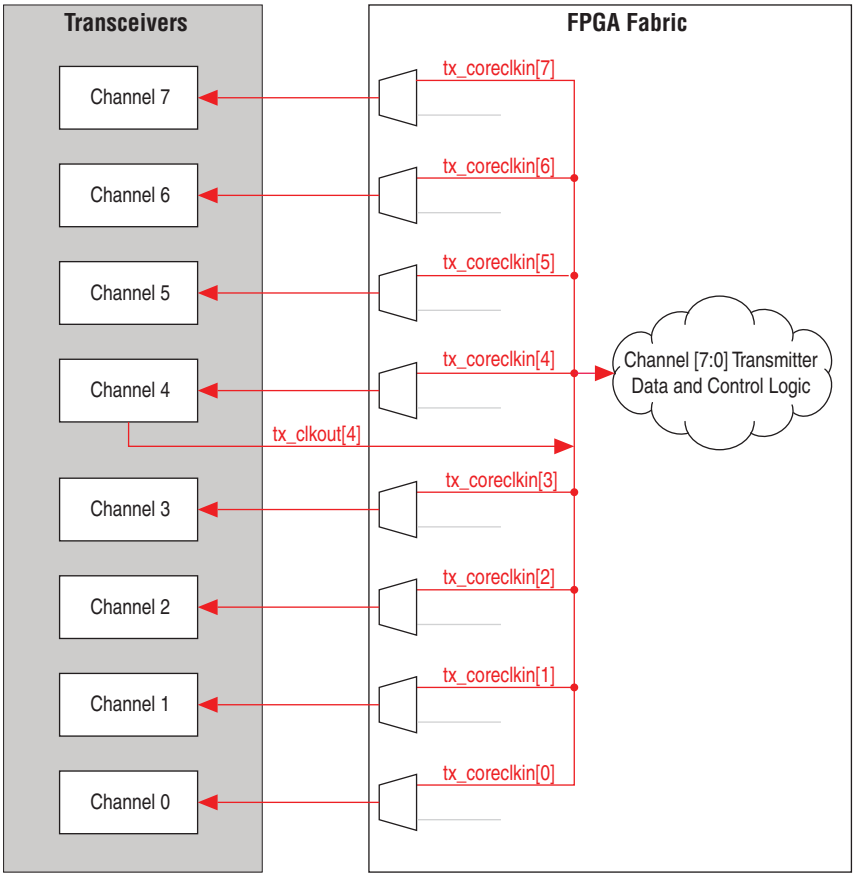
User-Selected Transmitter Datapath Interface Clock

Multiple transmitter channels that are non-bonded lead to high utilization of GCLK, RCLK, and PCLK resources (one clock resource per channel as shown in Figure 2–14). You can significantly reduce GCLK, RCLK, and PCLK resource use for transmitter datapath clocks if the transmitter channels are identical.

Identical transmitter channels are defined as channels that have the same input reference clock source, the same transmit PLL configuration, and the same transmitter PMA and PCS configuration. Identical transmitter channels may have different analog settings, such as transmitter voltage output differential (V_{OD}), transmitter common-mode voltage (V_{CM}), or pre-emphasis setting.

To achieve the clock resource savings, select a common clock driver for the transmitter datapath interface of all identical transmitter channels. Figure 2–17 shows eight identical channels clocked by a single clock (`tx_clkout` of channel 4).

Figure 2–17. Eight Identical Channels with a Single User-Selected Transmitter Interface Clock



To clock eight identical channels with a single clock, perform these steps:

- Instantiate the `tx_coreclkin` port for all the identical transmitter channels (`tx_coreclkin[7:0]`).
- Connect `tx_clkout[4]` to the `tx_coreclkin[7:0]` ports.
- Connect `tx_clkout[4]` to the transmitter data and control logic for all eight channels.

Resetting or powering down channel 4 leads to a loss of the clock for all eight channels.

The common clock must have a 0 ppm difference for the read side of the TX phase compensation FIFO of all the identical channels. A frequency difference causes the FIFO to underrun or overflow, depending on whether the common clock is slower or faster, respectively.

You can drive the 0 ppm common clock by one of the following sources:

- `tx_clkout` of any channel in non-bonded channel configurations
- `tx_clkout[0]` in bonded channel configurations
- Dedicated `refclk` pins

The Quartus II software does not allow gated clocks or clocks that are generated in the FPGA logic to drive the `tx_coreclkin` ports.

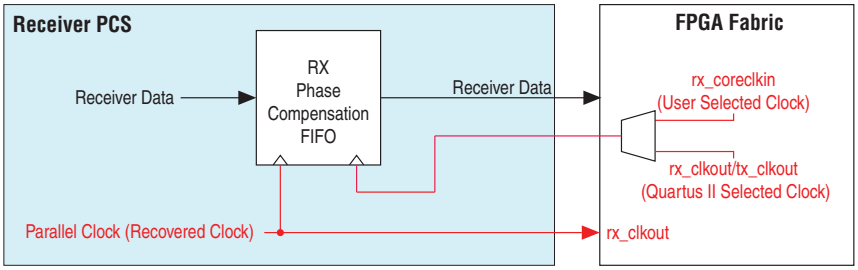
You must ensure a 0 ppm difference. The Quartus II software is unable to ensure a 0 ppm difference because it allows you to use external pins, such as dedicated `refclk` pins.

Receiver Datapath Interface Clock

The read side of the RX phase compensation FIFO makes up the 6-Gbps receiver datapath interface. The receiver datapath interface clock clocks this interface. Figure 2–18 shows the receiver datapath interface clocking. The receiver PCS forwards the following clocks to the FPGA fabric:

- `rx_clkout`—for each receiver channel in a non-bonded configuration when you do not use a rate matcher
- `tx_clkout`—for each receiver channel in a non-bonded configuration when you use a rate matcher
- single `tx_clkout[0]`—for all receiver channels in a bonded configuration

Figure 2–18. 6-Gbps Receiver Datapath Interface Clocking

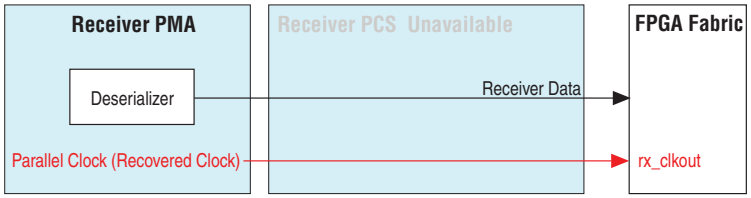


All configurations that use the PCS channel must have a 0 ppm difference between the receiver datapath interface clock and the read side clock of the RX phase compensation FIFO.

For the 10-Gbps transceivers, there are no PCS blocks. The only receive datapath available is a direct connection from the receiver PMA deserializer to the FPGA fabric.

Figure 2–19 shows the 10-Gbps receiver datapath interface clocking. For each receiver channel in a non-bonded configuration, the receiver PMA forwards the rx_clkout clock to the FPGA fabric.

Figure 2–19. 10-Gbps Receiver Datapath Interface Clocking



For more information about interface clocking for each configuration, refer to the *Transceiver Custom Configuration in Arria V Devices* and *Transceiver Protocol Configurations in Arria V Devices* chapters.

You can clock the receiver datapath interface by one of the following options:

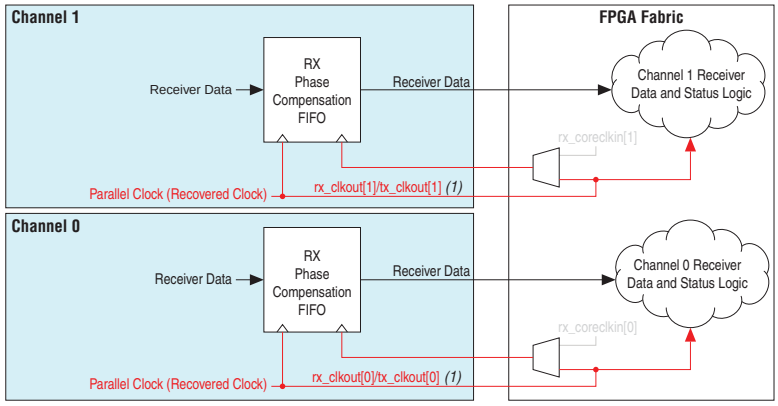
- The Quartus II-selected receiver datapath interface clock
- The user-selected receiver datapath interface clock

To reduce GCLK, RCLK, and PCLK resource utilization in your design, you can select the user-selection option to share the transceiver datapath interface clocks.

Quartus II Software-Selected Receiver Datapath Interface Clock

The Quartus II software automatically picks the appropriate clock from the FPGA fabric to clock the receiver datapath interface. Figure 2–20 shows the receiver datapath interface of two 6-Gbps transceiver non-bonded channels clocked by their respective receiver PCS clocks, which are forwarded to the FPGA fabric.

Figure 2–20. 6-Gbps Receiver Datapath Interface Clocking for Non-Bonded Channels



Note to Figure 2–20:

(1) If you use a rate matcher, the tx_clkout clock is used.

Figure 2–21 shows the receiver datapath interface of two 10-Gbps transceiver non-bonded channels clocked by their respective receiver CDR recovered PMA clocks, which are forwarded to the FPGA fabric.

Figure 2–21. 10-Gbps Receiver Datapath Interface Clocking for Non-Bonded Channels

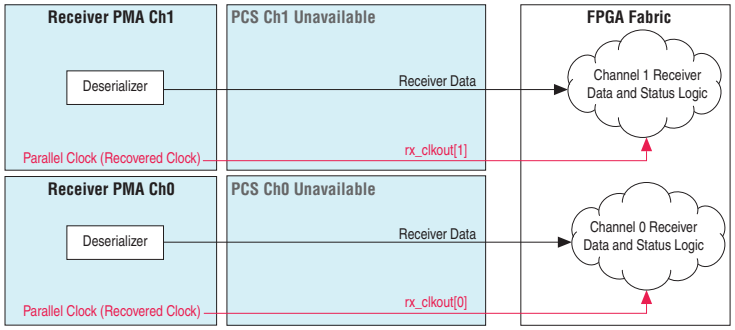
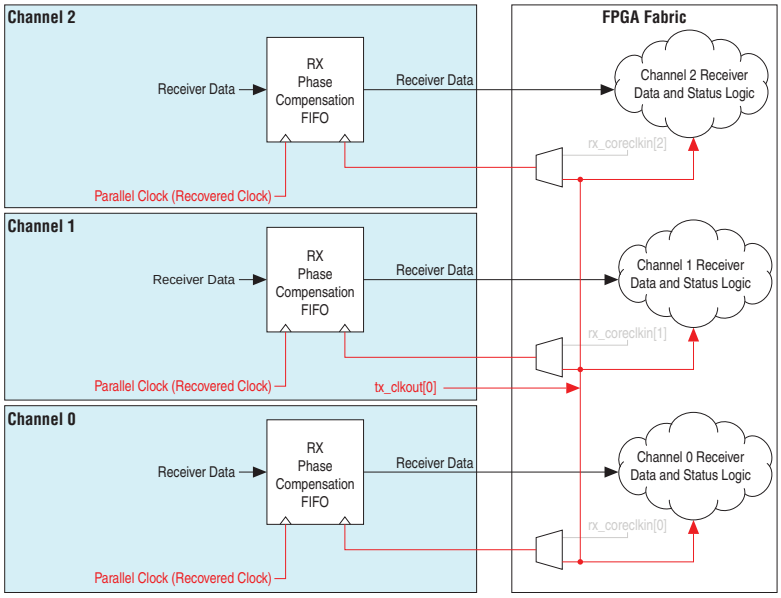


Figure 2–22 shows the 6-Gbps receiver datapath interface of three bonded channels clocked by the `tx_clkout[0]` clock. The `tx_clkout[0]` clock is derived from the central clock divider of channel 1 or 4 in a transceiver bank.

Figure 2–22. 6-Gbps Receiver Datapath Interface Clocking for Three Bonded Channels



User-Selected Receiver Datapath Interface Clock

Non-bonded multiple receiver channels lead to high utilization of GCLK, RCLK, and PCLK resources—one clock resource per channel, as shown in Figure 2–20 on page 2–32. You can significantly reduce GCLK, RCLK, and PCLK resource use for the receiver datapath clocks if the receiver channels are identical.

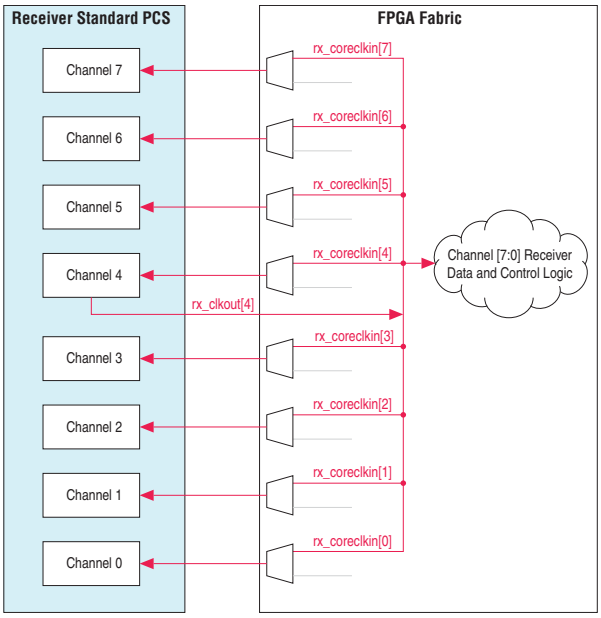
Identical receiver channels are defined as channels that have the same input reference clock source for the CDR and the same receiver PMA and PCS configuration. These channels may have different analog settings, such as receiver common mode voltage (V_{ICM}), equalization, or DC gain setting.

To achieve clock resource savings, select a common clock driver for the receiver datapath interface of all identical receiver channels. To select a common clock driver, perform these steps:

- Instantiate the `rx_coreclkin` port for all the identical receiver channels
- Connect the common clock driver to their receiver datapath interface, and receiver data and control logic.

Figure 2–23 shows eight identical channels that are clocked by a single clock (`rx_clkout` of channel 4).

Figure 2–23. Eight Identical Channels with a Single User-Selected Receiver Interface Clock



To clock eight identical channels with a single clock, perform these steps:

- Instantiate the `rx_coreclkin` port for all the identical receiver channels (`rx_coreclkin[7:0]`).
- Connect `rx_clkout[4]` to the `rx_coreclkin[7:0]` ports.
- Connect `rx_clkout[4]` to the receiver data and control logic for all eight channels.

Resetting or powering down channel 4 leads to a loss of the clock for all eight channels.

The common clock must have a 0 ppm difference for the write side of the RX phase compensation FIFO of all the identical channels. A frequency difference causes the FIFO to underrun or overflow, depending on whether the common clock is faster or slower, respectively.

You can drive the 0 ppm common clock driver by one of the following sources:

- `tx_clkout` of any channel in non-bonded receiver channel configurations with the rate matcher
- `rx_clkout` of any channel in non-bonded receiver channel configurations without the rate matcher
- `tx_clkout[0]` in bonded receiver channel configurations
- Dedicated `refclk` pins

The Quartus II software does not allow gated clocks or clocks generated in the FPGA logic to drive the `rx_coreclkin` ports.

You must ensure a 0 ppm difference. The Quartus II software is unable to ensure a 0 ppm difference because it allows you to use external pins, such as dedicated `refclk` pins.

Document Revision History

Table 2–23 lists the revision history for this chapter.

Table 2–23. Document Revision History

Date	Version	Changes
November 2011	1.1	■ Updated the "Byte Ordering" section. ■ Updated Table 2–1 and Table 2–3. ■ Updated Figure 2–2 and Figure 2–3. ■ Minor text edits.
August 2011	1.0	Initial release.

This chapter provides additional information about the document and Altera.

How to Contact Altera

To locate the most up-to-date information about Altera products, refer to the following table.

Contact (1)	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note to Table:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, **\qdesigns** directory, **D:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicate document titles. For example, *Stratix IV Design Guidelines*.
italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>***.pof** file.
Initial Capital Letters	Indicate keyboard keys and menu names. For example, the Delete key and the Options menu.
"Subheading Title"	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, "Typographic Conventions."

Visual Cue	Meaning
`Courier type`	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. The suffix n denotes an active-low signal. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
↵	An angled arrow instructs you to press the Enter key.
1., 2., 3., and a., b., c., and so on	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■ ■	Bullets indicate a list of items when the sequence of the items is not important.
	The hand points to information that requires special attention.
	A question mark directs you to a software help system with related information.
	The feet direct you to another document or website with related information.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The envelope links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.