Cyclone V Device Handbook

Volume 4: Device Basics

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

CV-5V5-1.1

Document last updated for Altera Complete Design Suite version: 11.1
Document publication date: November 2011

Contents

Chapter Revision Dates

The chapters in this document, Cyclone V Device Handbook, 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 Cyclone V Devices
Revised: *November 2011*
Part Number: *CV-55001-1.1*

Chapter 2. Transceiver Basics for Cyclone V Devices
Revised: *October 2011*
Part Number: *CV-55002-1.0*

1. Device Interfaces and Integration Basics for Cyclone V Devices

CV-55001-1.1

This chapter contains basic information of specific feature in the Cyclone® V device interfaces and integration. This chapter serves as a supplementary reading to *Volume 2: Device Interfaces and Integration* of the *Cyclone V Device Handbook*.

This chapter 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 Cyclone® V devices.

The information in this section should be used in conjunction with the *Logic Array Blocks and Adaptive Logic Modules in Cyclone V Devices* chapter.

Logic Array Blocks

The LABs are configurable logic blocks that consist of a group of logic resources.

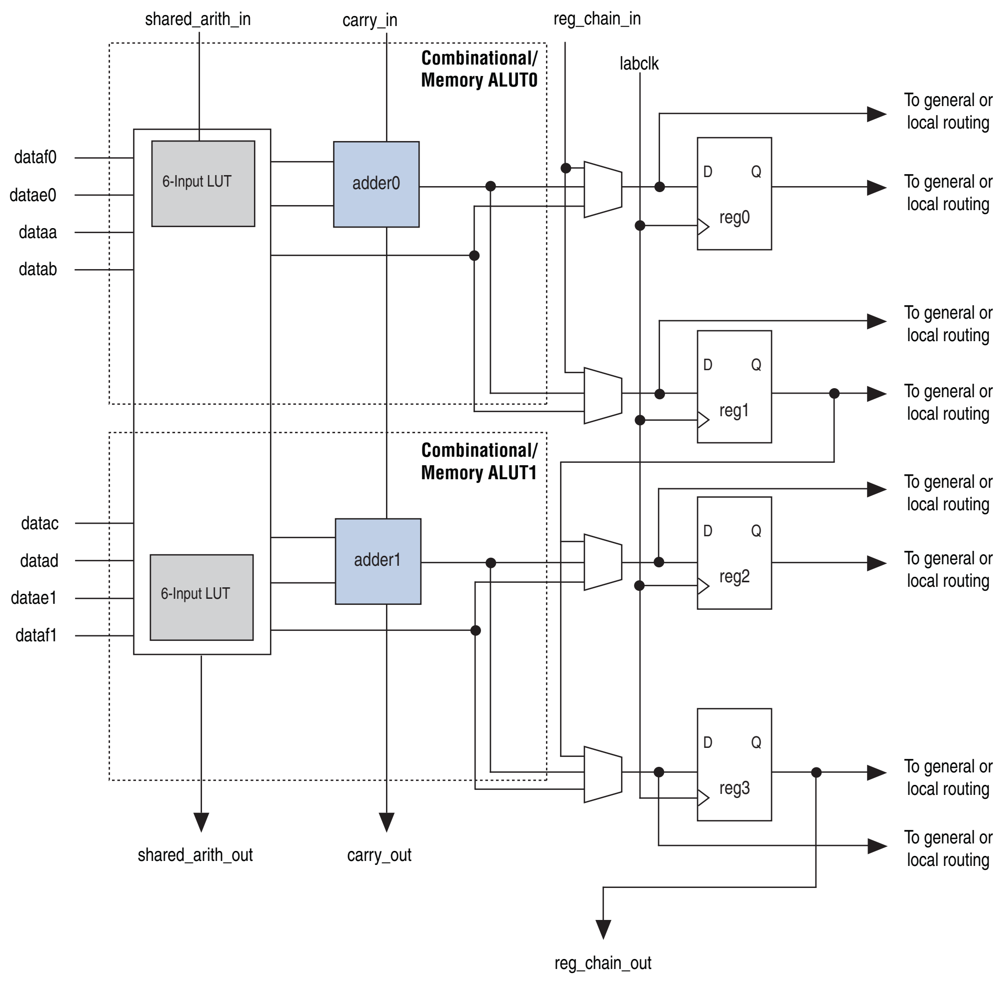
You can use a quarter of the available LABs in Cyclone V devices as a memory LAB (MLAB). The MLAB supports a maximum of 640 bits of simple dual-port SRAM.

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. Cyclone 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 Cyclone 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 Cyclone V ALM.

Figure 1–1. High-Level Block Diagram of the Cyclone 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 Cyclone V ALM operates in any of the following modes:

- "Normal Mode" on page 1–3
- "Extended LUT Mode" on page 1–3
- "Arithmetic Mode" on page 1–3
- "Shared Arithmetic Mode" on page 1–3

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 a Cyclone 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 Cyclone 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

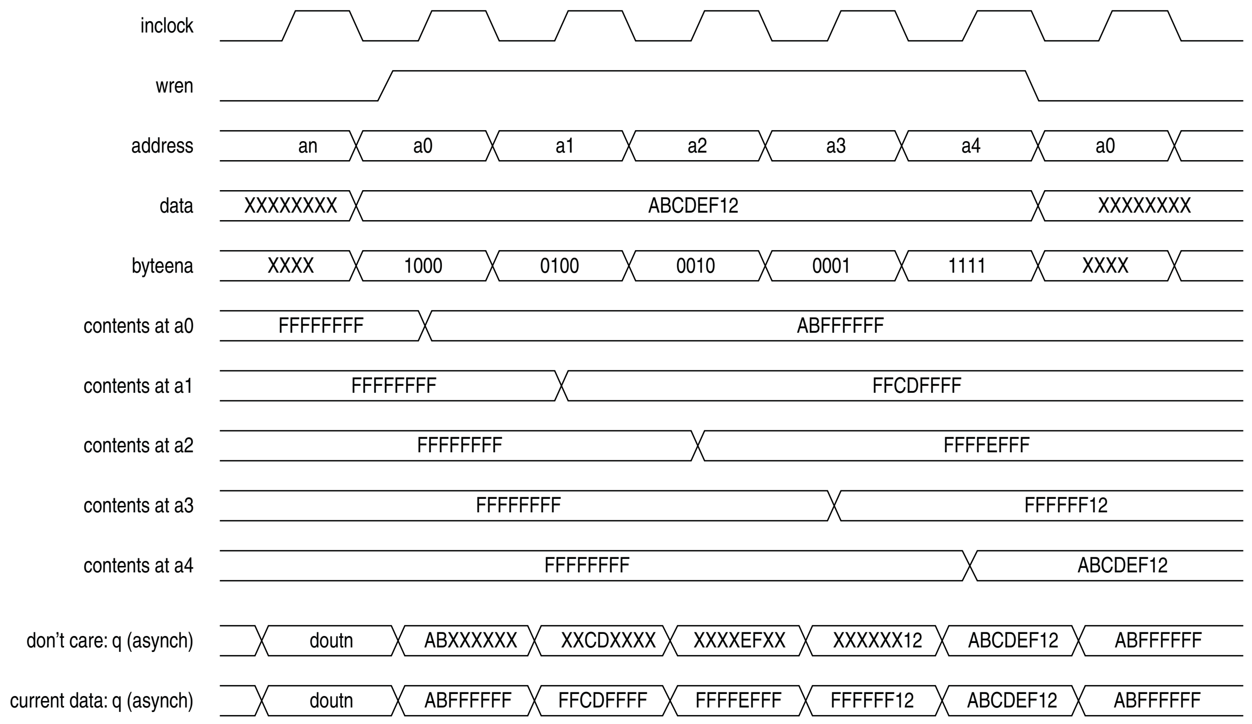
byteena[3:0]	Data Bits Written			
1111 (default)	`[39:30]`	`[29:20]`	`[19:10]`	`[9:0]`
1000	`[39:30]`	—	—	—
0100	—	`[29:20]`	—	—
0010	—	—	`[19:10]`	—
0001	—	—	—	`[9:0]`

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" on page 1–6
- "Conflict Resolution" on page 1–6
- "Read-During-Write Behavior" on page 1–6
- "Power-Up Conditions and Memory Initialization" on page 1–10
- "Power Management" on page 1–10

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

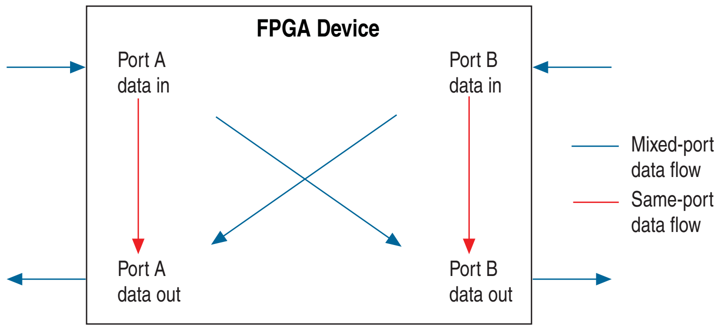
If you use the memory blocks in true dual-port mode, you can perform two write operations to the same memory location (address). Because conflict resolution circuitry is not available in the memory blocks, you must implement the conflict resolution logic external to the memory block to avoid unknown data from being written to the address.

Read-During-Write Behavior

You can customize the read-during-write behavior of the Cyclone 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 Cyclone V Devices



Same-Port Read-During-Write Mode

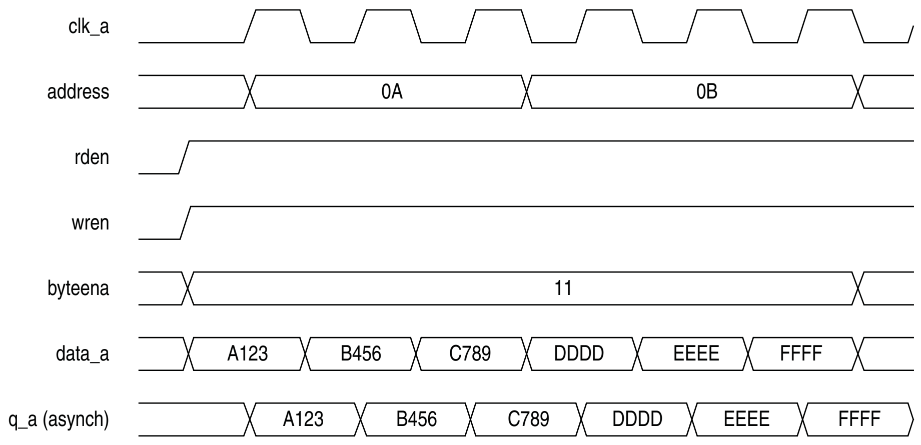
The same-port read-during-write mode applies to either a single-port RAM or the same port of a true dual-port RAM. Table 1–3 lists the available output modes if you select the M10K or MLAB in the same-port read-during-write mode.

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

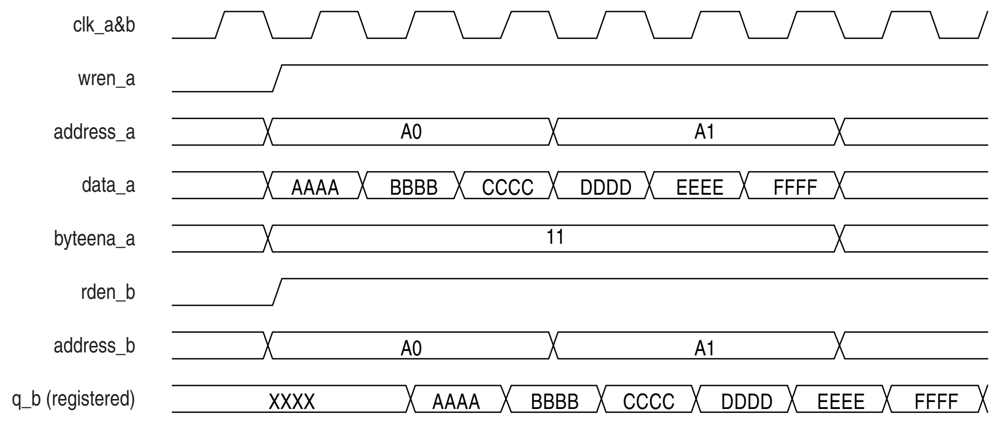


Figure 1–6 shows a sample functional waveform of mixed-port read-during-write behavior for the "old data" mode.

Figure 1–6. Mixed-Port Read-During-Write: Old Data Mode

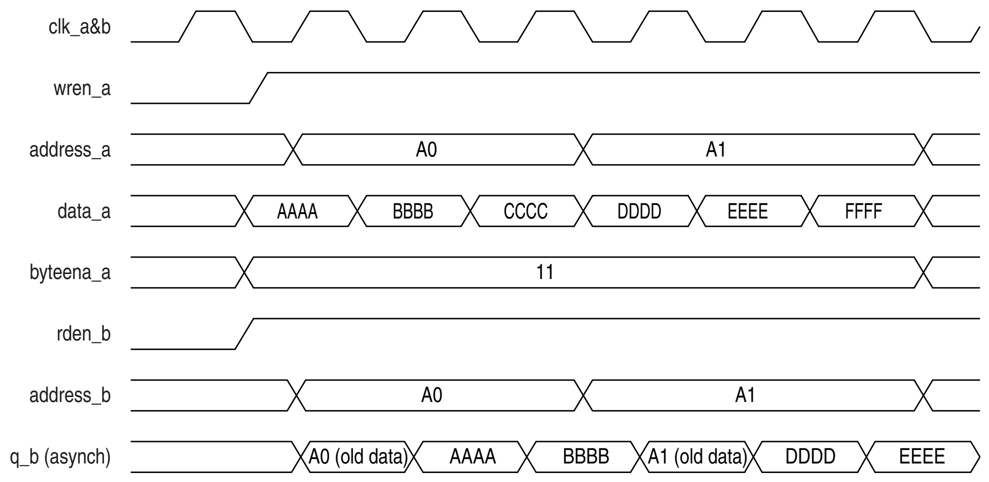
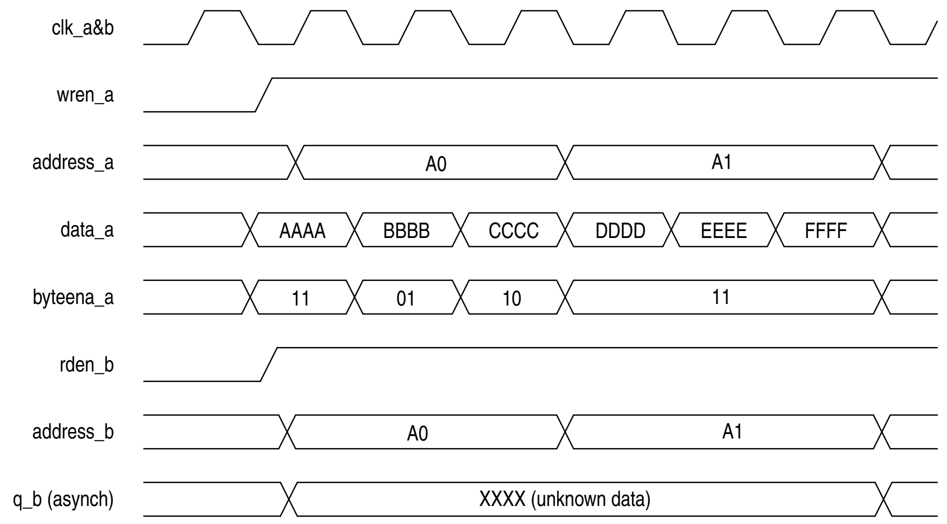


Figure 1–7 shows a sample functional waveform of mixed-port read-during-write behavior for the "don't care" or "constrained don't care" mode.

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 Cyclone 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 Cyclone 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 Cyclone V devices.

Use the information in this section in conjunction with the *Variable-Precision DSP Blocks in Cyclone V Devices* chapter.

Independent Multiplier Mode

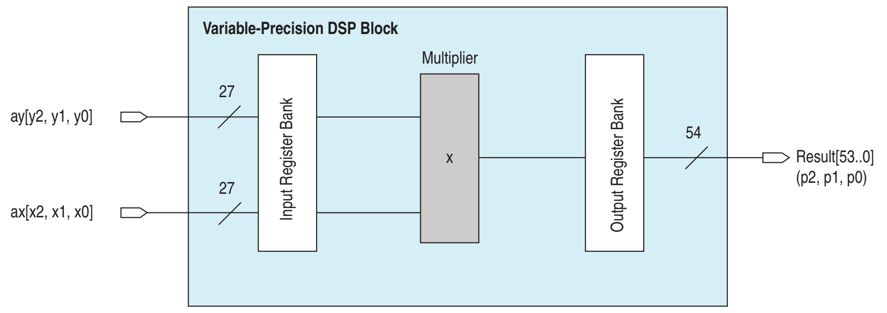
In independent input and output multiplier mode, the variable-precision DSP blocks perform individual multiplication operations for general purpose multipliers.

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 1–12 show the variable-precision DSP block in independent multiplier operation mode.

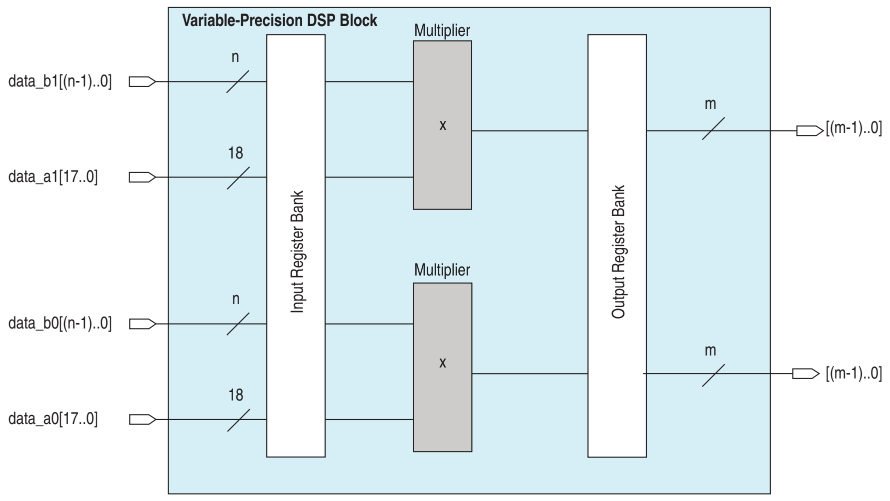
Figure 1–8. Three 9 x 9 Independent Multiplier Mode with One Variable-Precision DSP Block *(1)*



Note to Figure 1–8:

(1) Three pairs of data are packed into the `ax` and `ay` ports; `result` contains three 18-bit products.

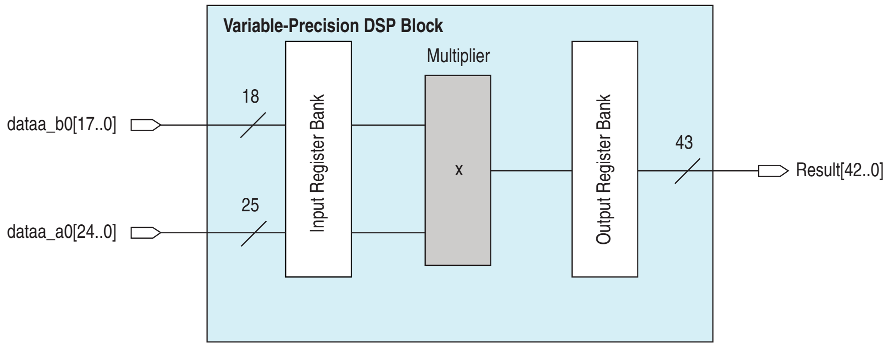
Figure 1–9. Two 18 x 18 or One 18 x 19 Independent Multiplier Mode with One Variable-Precision DSP Block *(1), (2)*



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.

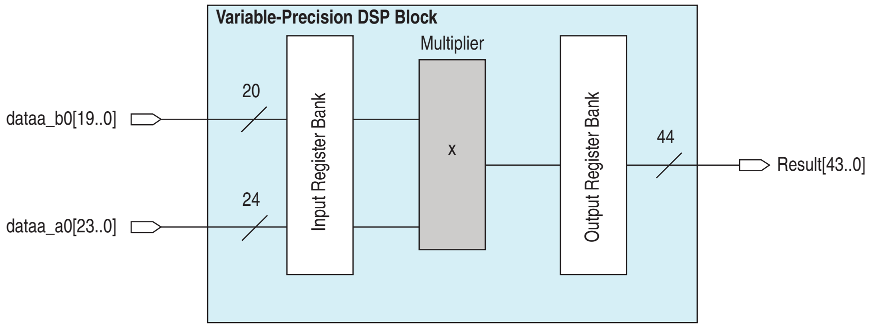
Figure 1–10. One 18 x 25 Independent Multiplier Mode with One Variable-Precision DSP Block *(1)*



Note to Figure 1–10:

(1) The result can be up to 52 bits when combined with a chainout adder or accumulator.

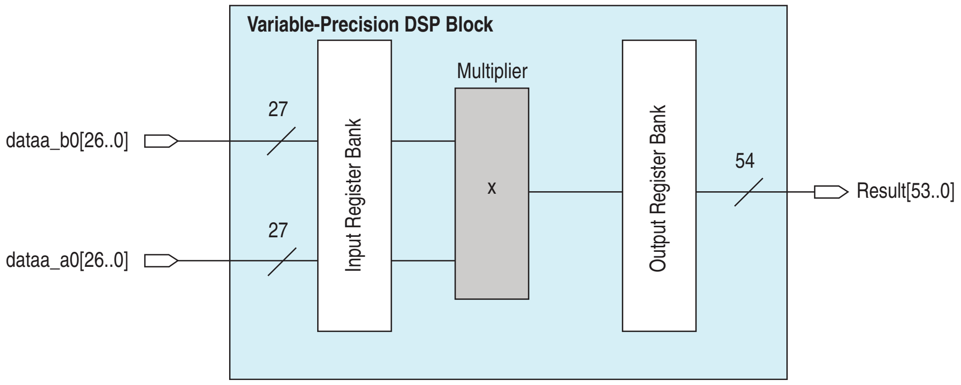
Figure 1–11. One 20 x 24 Independent Multiplier Mode with One Variable-Precision DSP Block *(1)*



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 Cyclone 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

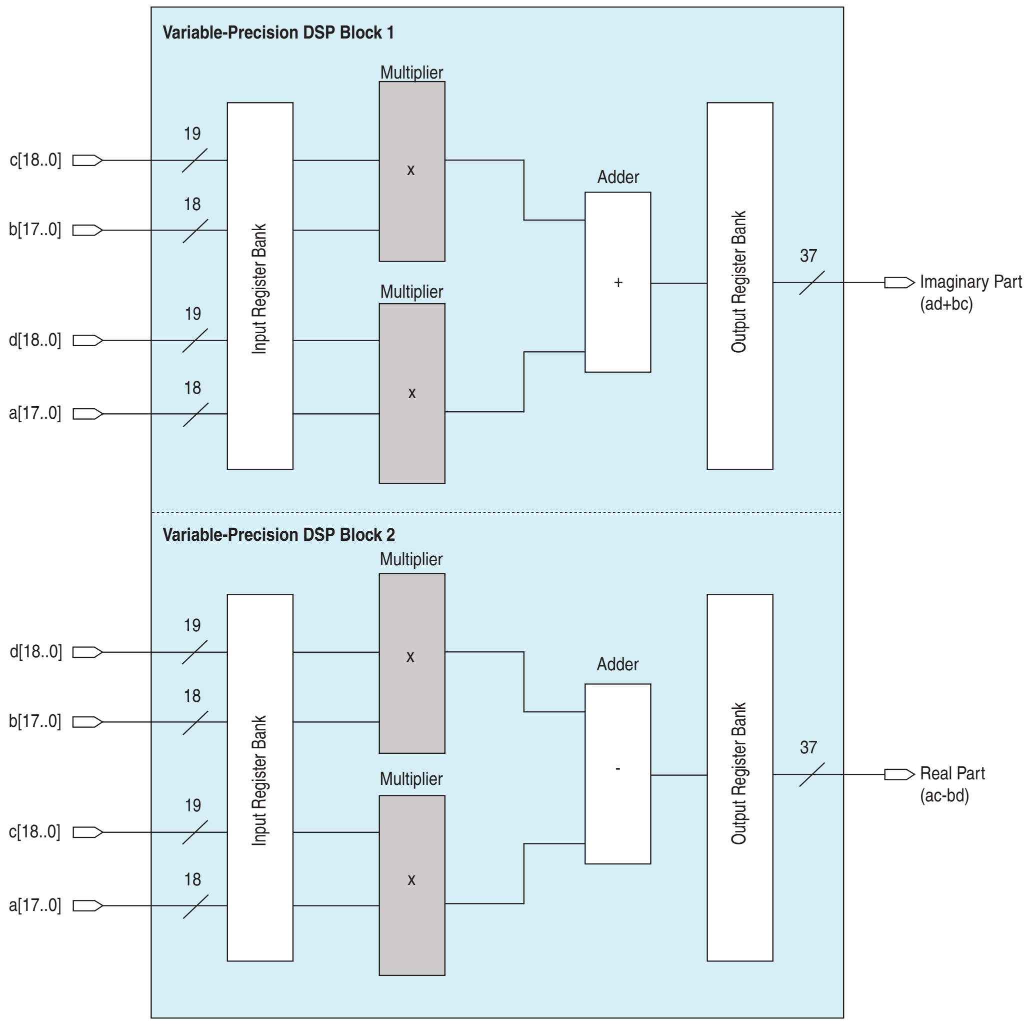
$$(a + jb) \times (c + jd) \ = \ [(a \times c) - (b \times d)] + j[(a \times d) + (b \times c)]$$

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

Cyclone 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. Cyclone 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

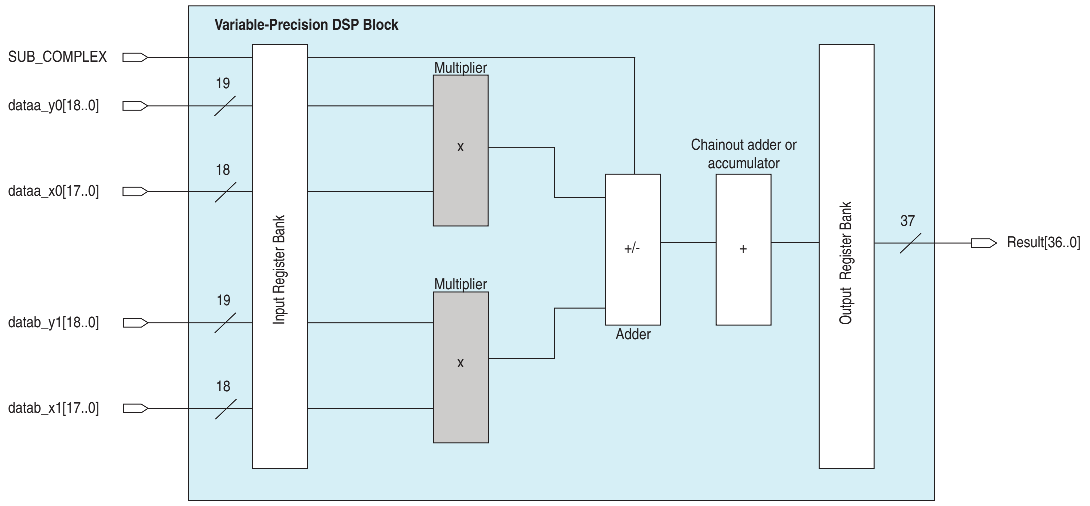


Figure 1–15. One Sum of Two 27 x 27 Multipliers with Two Variable-Precision DSP Blocks

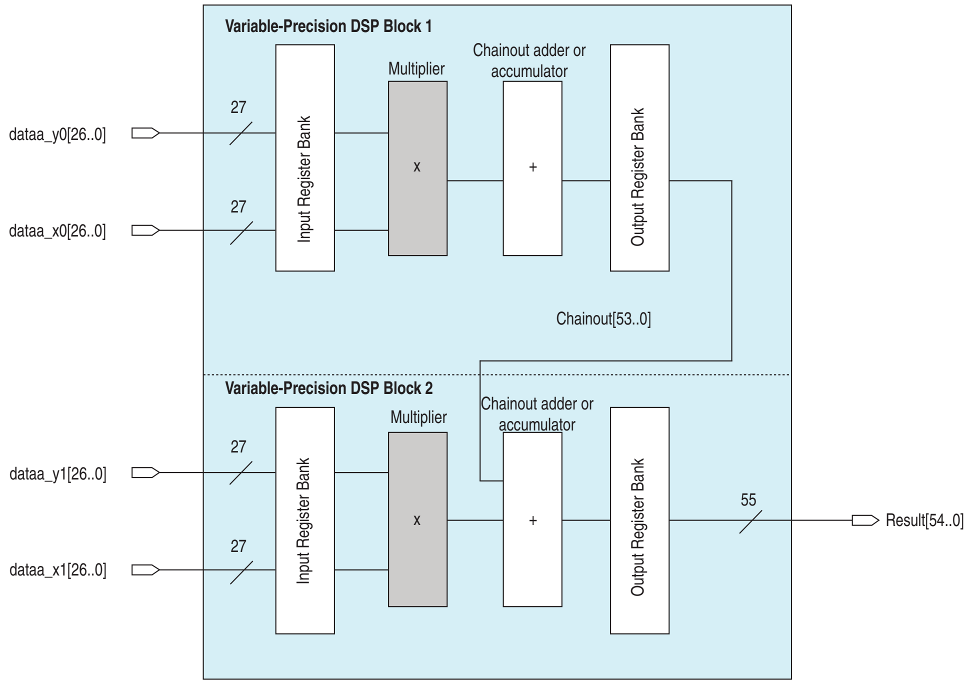
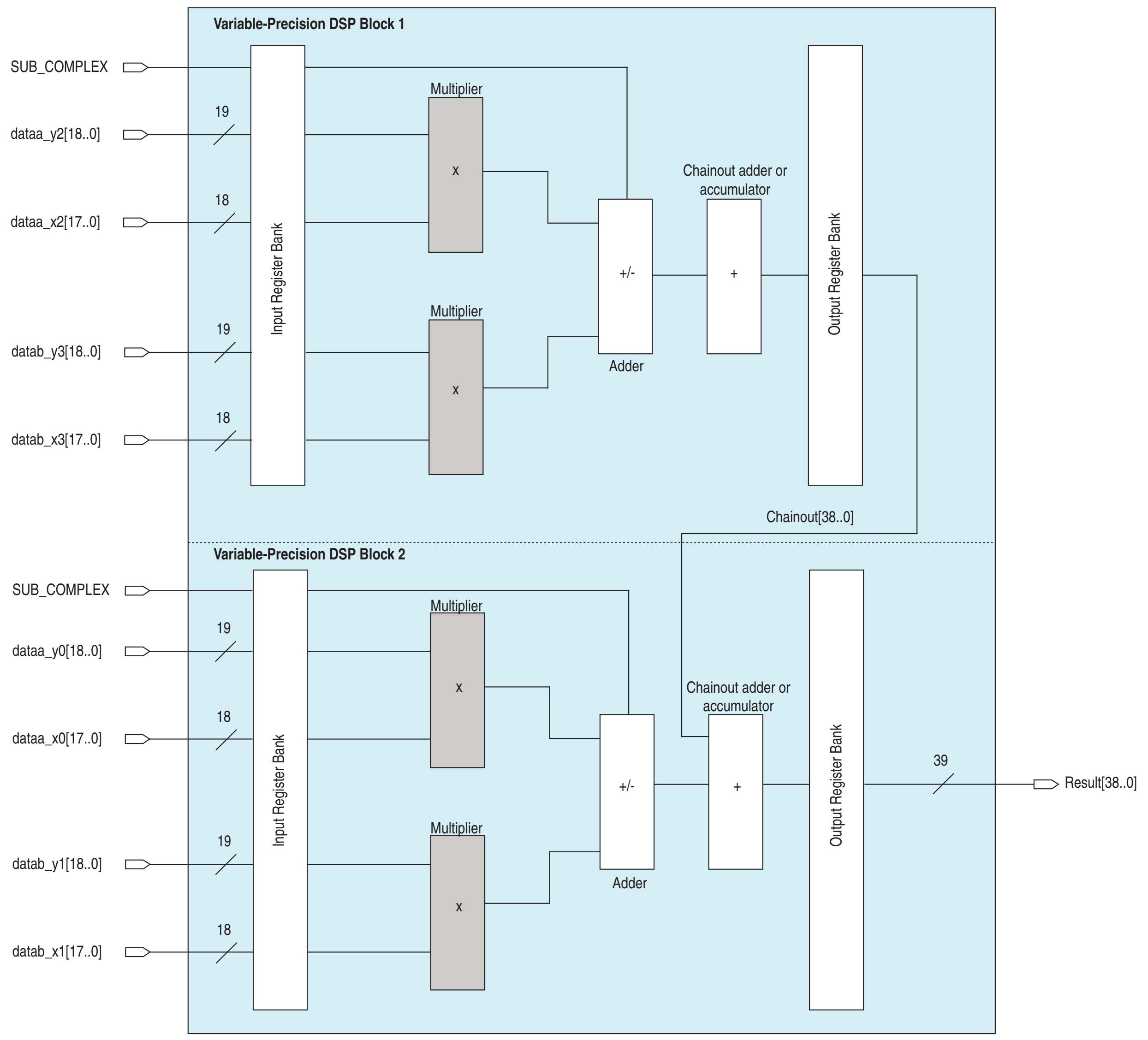


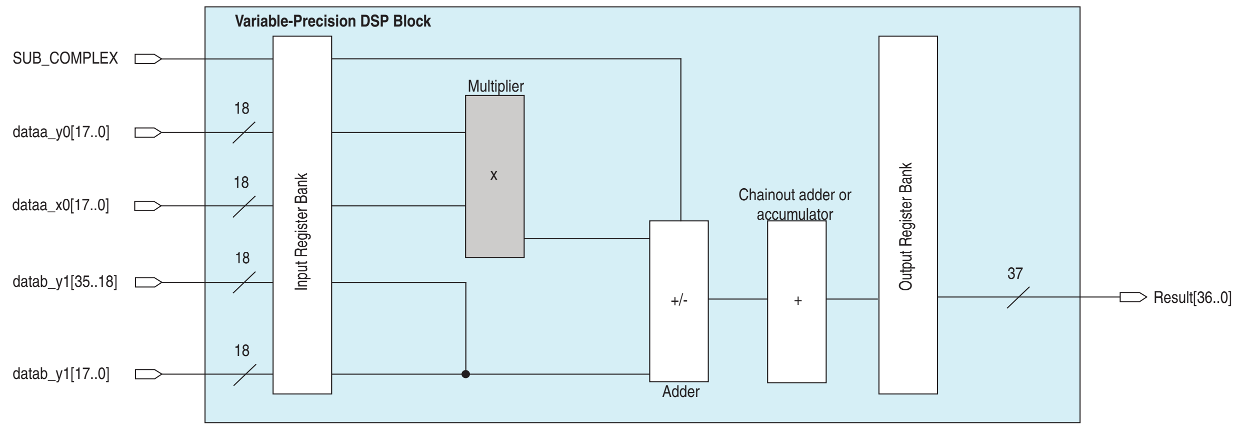
Figure 1–16. One Sum of Four 18 x 19 Multipliers with Two Variable-Precision DSP Blocks



18 x 18 Multiplication Summed with 36-Bit Input Mode

Cyclone 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



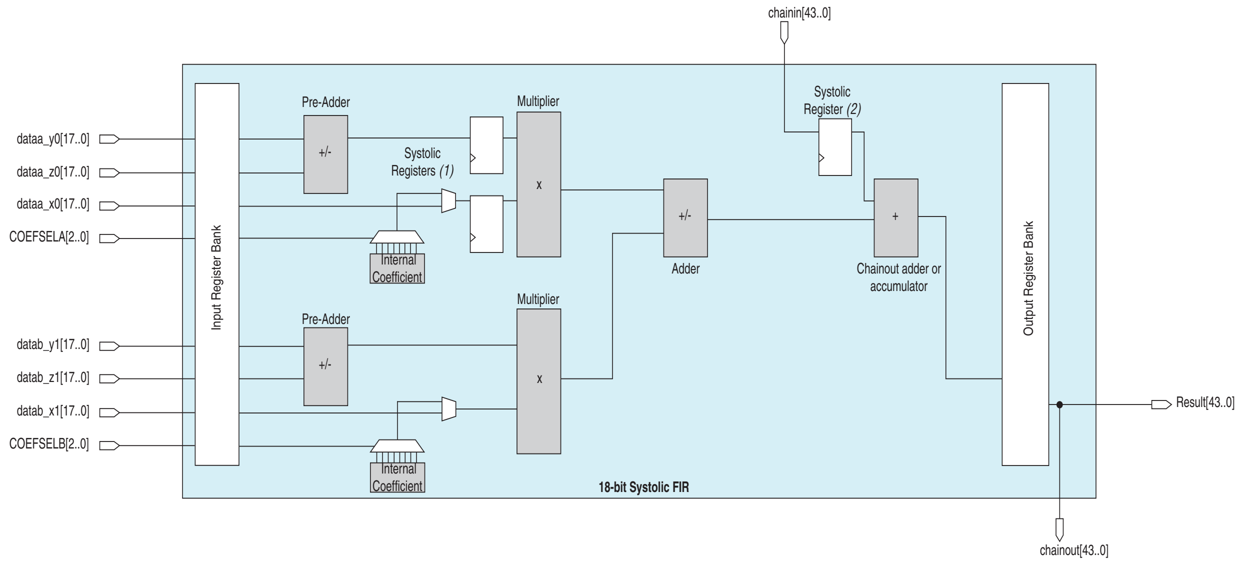
Systolic FIR Mode

Cyclone 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



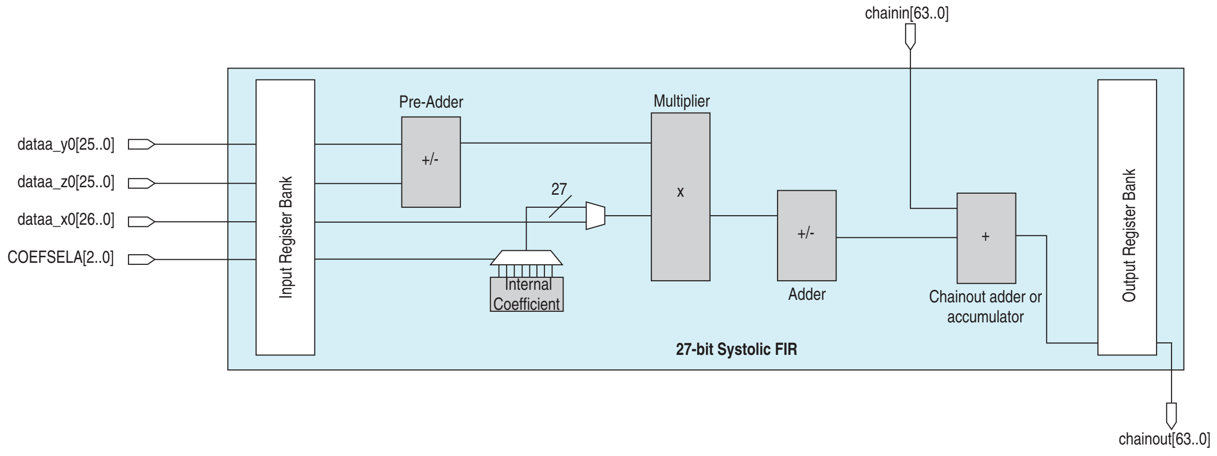
Notes to Figure 1–18:

(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 Cyclone V devices.

Use the information in this section in conjunction with the *Clock Networks and PLLs in Cyclone V Devices* chapter.

Clock Regions

This section describes how you can form different types of clock regions in Cyclone V devices and the best options for each type.

Cyclone 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

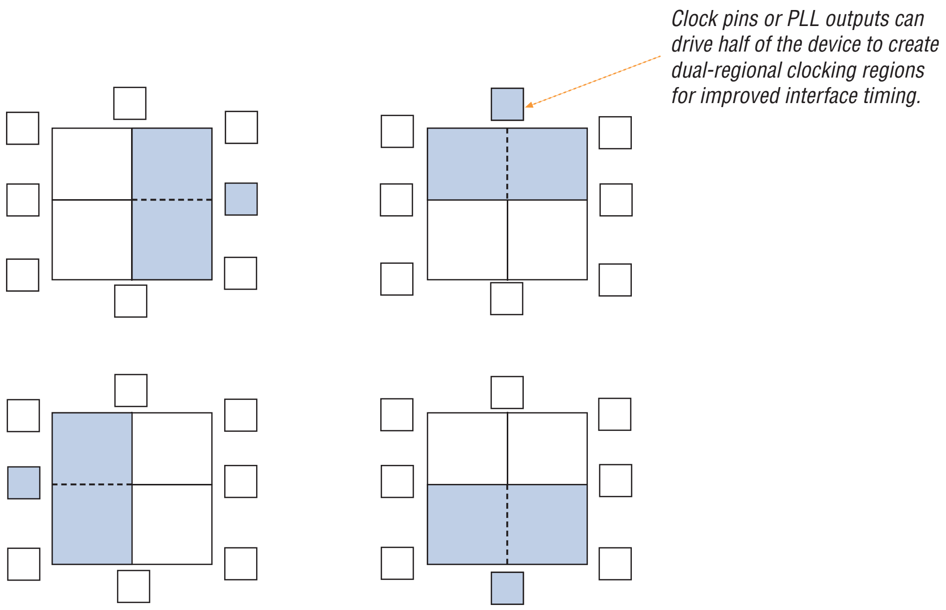
To form a regional clock region, a source drives a signal RCLK network that you can route throughout one quadrant of the device. This clock region provides the lowest skew in a quadrant. It is a good option if all the destinations are in a single quadrant.

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.

Figure 1–20 shows the dual-regional clock region.

Figure 1–20. Dual-Regional Clock Region for Cyclone V Devices


Clock pins or PLL outputs can drive half of the device to create dual-regional clocking regions for improved interface timing.

Clock Control Block

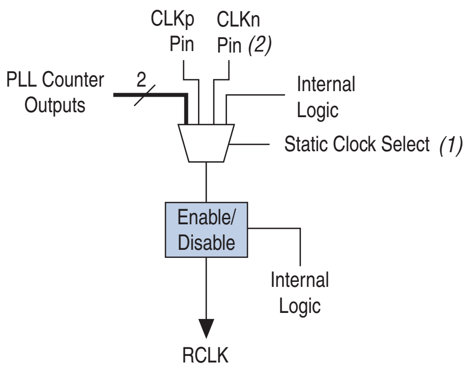
This section describes how you can control the clock source selection for the RCLK and PCLK select block.

RCLK Control Block

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

Figure 1–21 shows the RCLK control blocks.

Figure 1–21. RCLK Control Block



Notes to Figure 1–21:

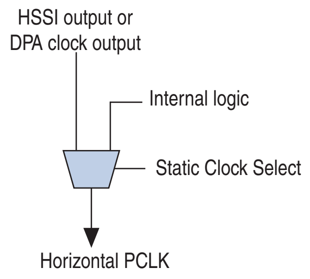
(1) When the device is in user mode, you can set the clock select signals only through the **.sof** or **.pof** because the signals cannot be controlled dynamically.

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

PCLK Control Block

This section describes how you can control the clock source selection for the RCLK select block. 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.

Figure 1–22. Horizontal PCLK Control Block

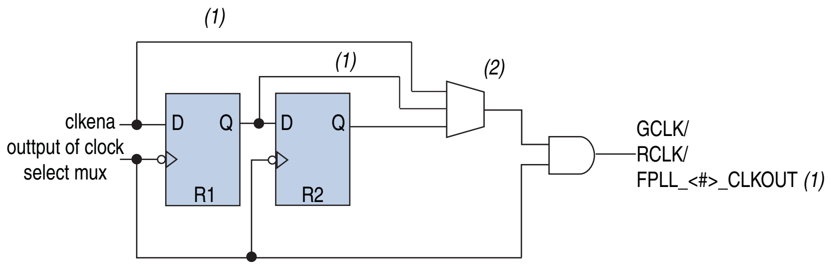


You can power down the Cyclone V GCLK and RCLK clock networks using static and dynamic approaches. When a clock network is powered down, all the logic fed by the clock network is in off-state, thereby reducing the overall power consumption of the device. The unused GCLK, RCLK, and PCLK networks are automatically powered down through configuration bit settings in **.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.

Clock Enable Signals

Figure 1–23 shows how the clock enable and disable circuit of the clock control block is implemented in Cyclone V devices.

Figure 1–23. clkena Implementation



Notes to Figure 1–23:

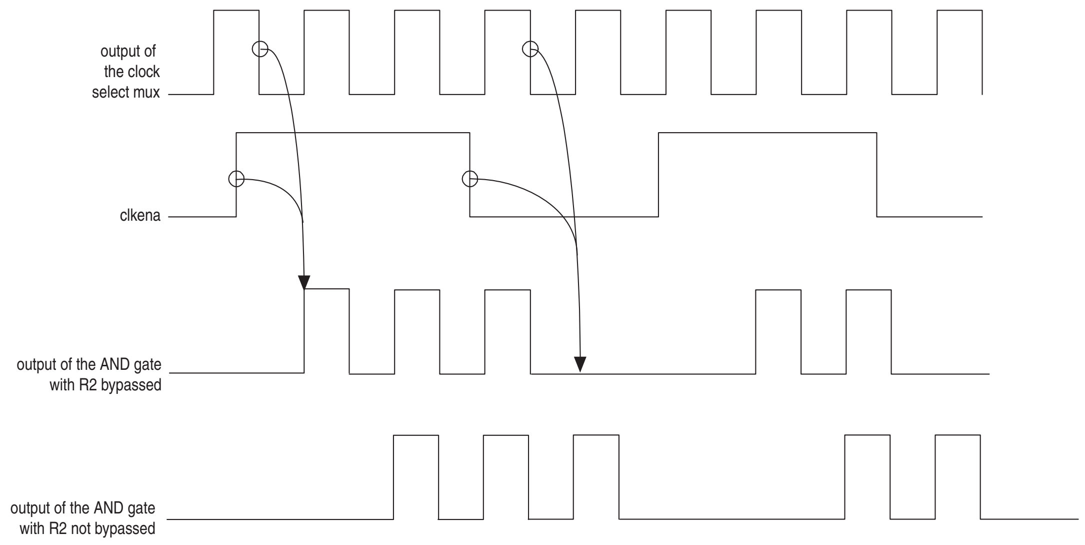
(1) The R1 and R2 bypass paths are not available for the PLL external clock outputs.

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

In Cyclone 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 1–24 shows a waveform example for a clock output enable. The `clkena` signal is synchronous to the falling edge of the clock output.

Cyclone 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 1–24. clkena Signals (1)


output of the clock select mux
clkena
output of the AND gate with R2 bypassed
output of the AND gate with R2 not bypassed

Note to Figure 1–24:

(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

Cyclone V PLLs support different clock feedback modes. Each mode allows clock multiplication and division, phase shifting, and programmable duty cycle.

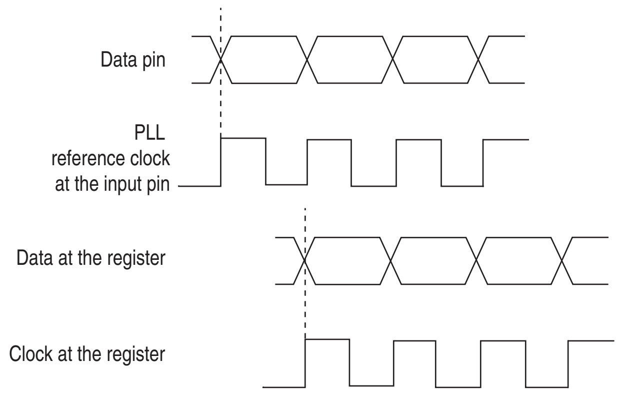
The input and output delays are fully compensated by a PLL only when using the dedicated clock input pins associated with a given PLL as the clock source. When a RCLK or GCLK network drives the PLL or the PLL is driven by a dedicated clock pin that is not associated with the PLL, the input and output 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 "Cyclone V PLLs" section in the *Clock Networks and PLLs in Cyclone 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 1–25 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 1–25. 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 Cyclone 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.

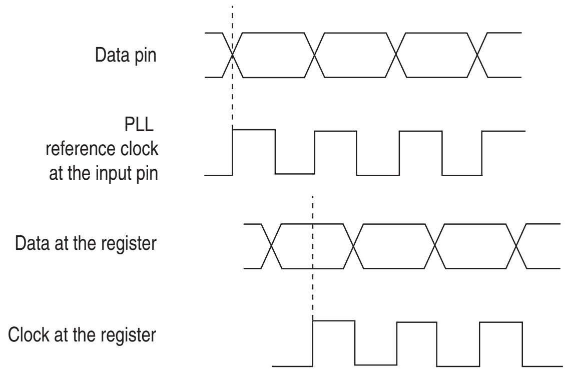
LVDS Compensation

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 1–26 shows a waveform example of the clock and data in LVDS mode.

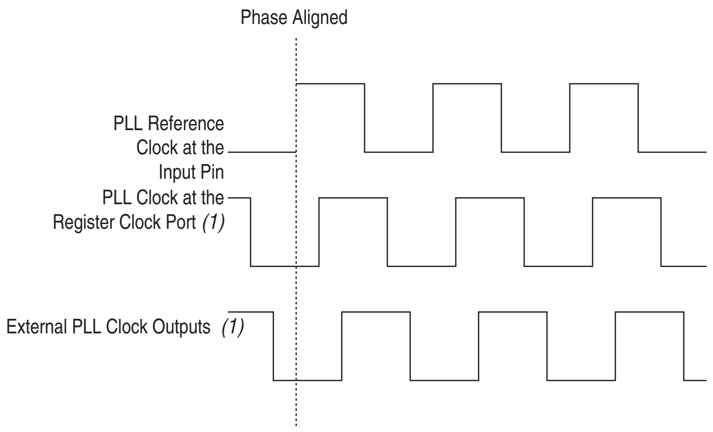
Figure 1–26. Phase Relationship Between the Clock and Data in LVDS Mode



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 1–27 shows a waveform example of the PLL clocks' phase relationship in direct compensation mode.

Figure 1–27. Phase Relationship Between the PLL Clocks in Direct Compensation Mode


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

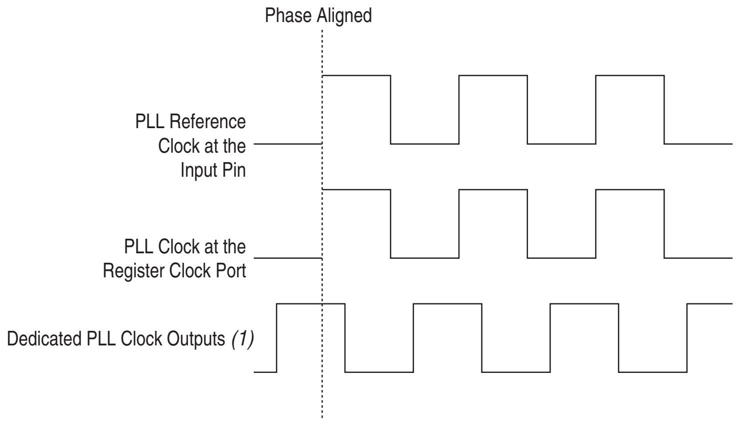
Note to Figure 1–27:

(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 1–28 shows a waveform example of the PLL clocks' phase relationship in normal mode.

Figure 1–28. Phase Relationship Between the PLL Clocks in Normal Mode


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

Note to Figure 1–28:

(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 Cyclone V PLLs.

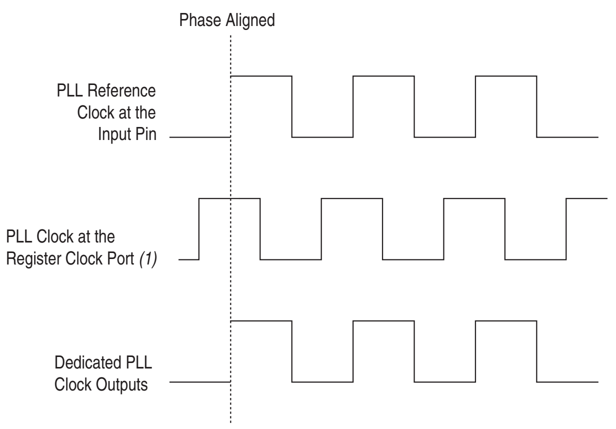
To ensure phase alignment between the `clk` pin and the external clock output (`CLKOUT`) pin when you use Cyclone V PLLs in ZDB mode, along with single-ended I/O standards, instantiate a bidirectional I/O pin in the design to serve as the feedback path connecting the `fbout` and `fbin` ports of the PLL. 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 1–29 shows a waveform example of the PLL clocks' phase relationship in ZDB mode.

Figure 1–29. Phase Relationship Between the PLL Clocks in ZDB Mode


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

Note to Figure 1–29:

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

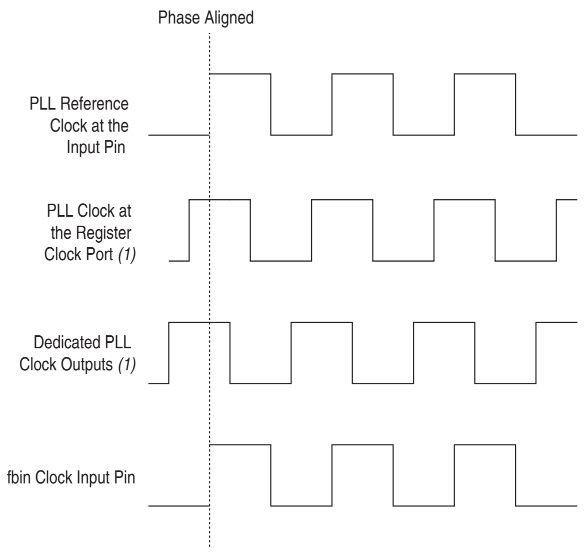
External Feedback Mode

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 1–30 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)*


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

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 Cyclone V PLLs, except on the corner fractional PLLs.

Clock Multiplication and Division

Each Cyclone 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 Cyclone 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 Cyclone V Devices* chapter.

I/O Element Features

The following sections describe the I/O element (IOE) features in Cyclone 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 Cyclone 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 Cyclone V Devices* chapter.

Open-Drain Output

Cyclone 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 Cyclone V Devices* chapter.

Differential Transmitter

The Cyclone 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 *I/O Features in Cyclone V Devices* chapter.

Differential Output Voltage

The Cyclone 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. Cyclone 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).

Cyclone 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.

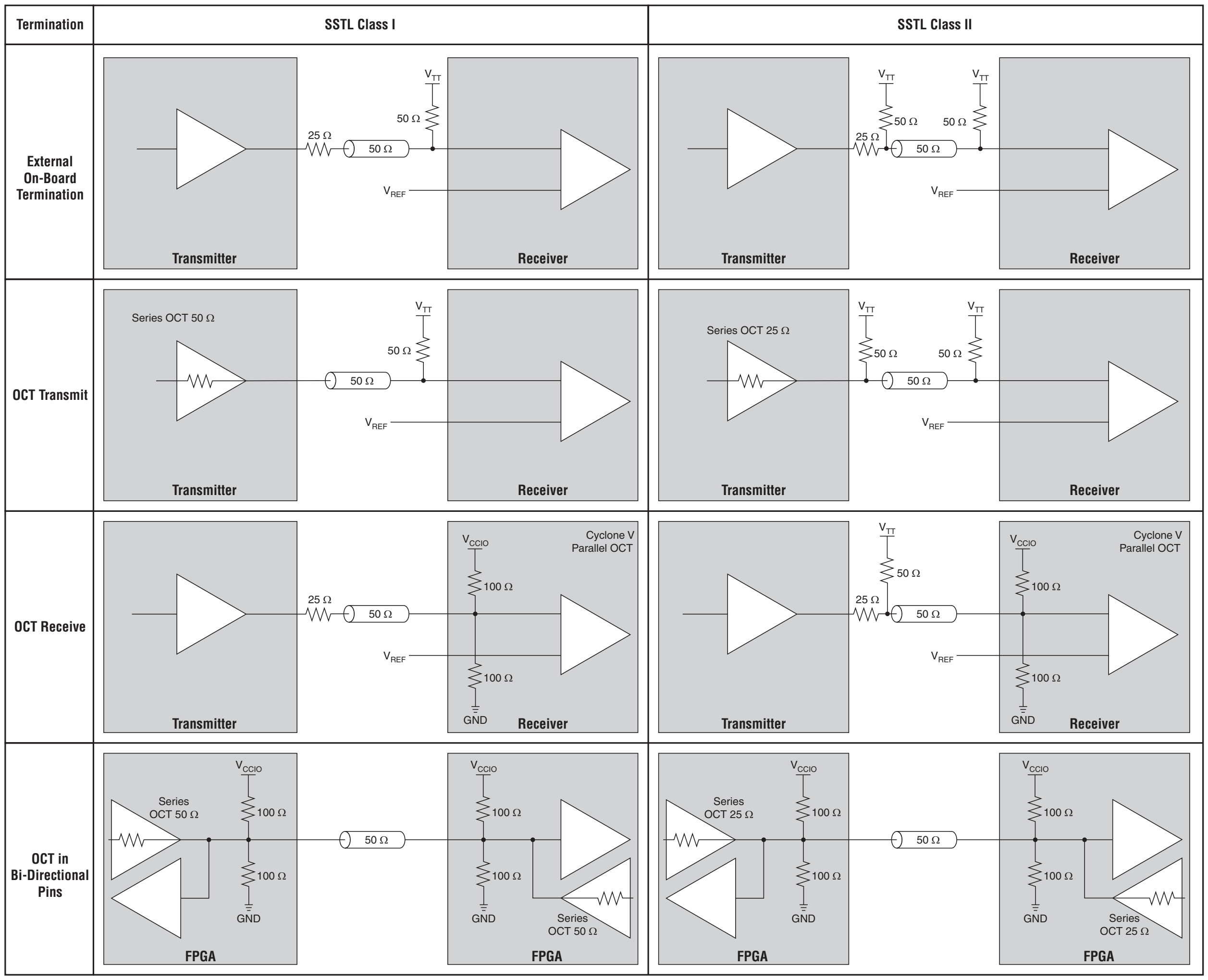
Termination Schemes for I/O Standards

Cyclone 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 Cyclone V devices.

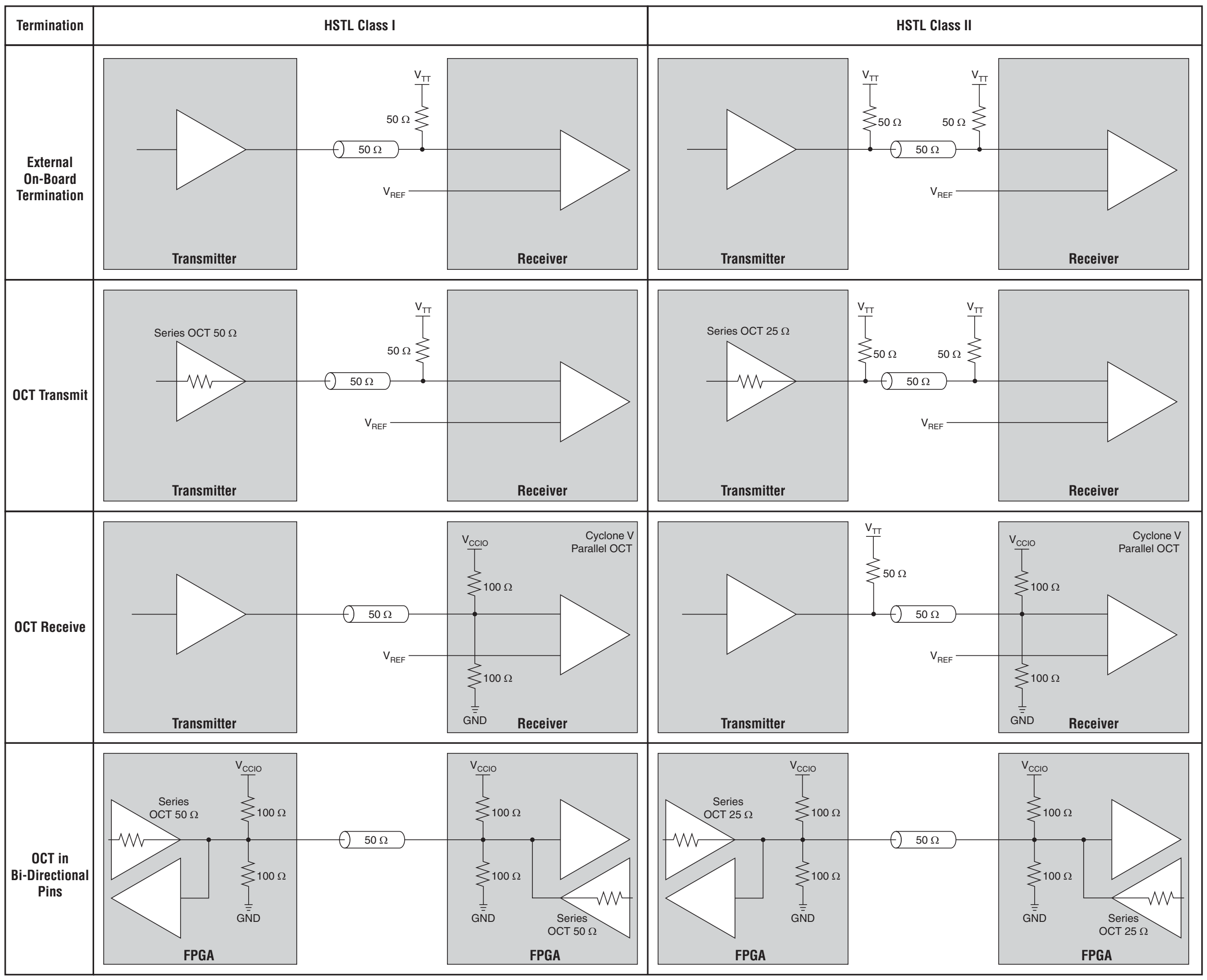
Figure 1–31. SSTL I/O Standard Termination



HSTL I/O Standard Termination

Figure 1–32 shows the details of **HSTL** I/O termination on Cyclone V devices.

Figure 1–32. HSTL I/O Standard Termination

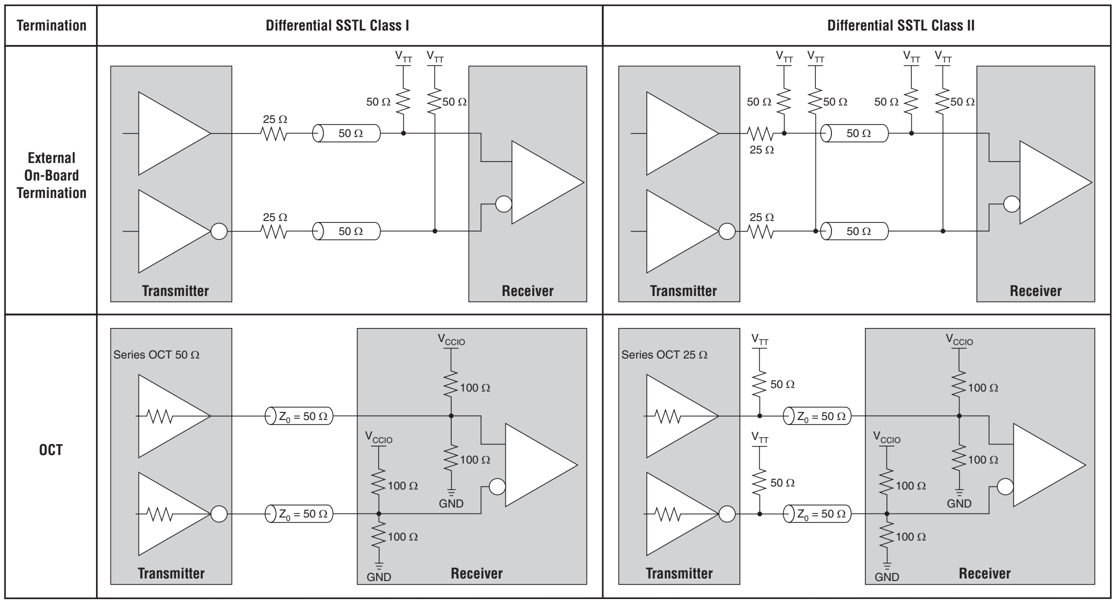


You cannot use R_S and R_T OCT simultaneously. For more information, refer to the *I/O Features in Cyclone V Devices* chapter.

Differential SSTL I/O Standard Termination

Figure 1–33 shows the details of **Differential SSTL** I/O termination on Cyclone V devices.

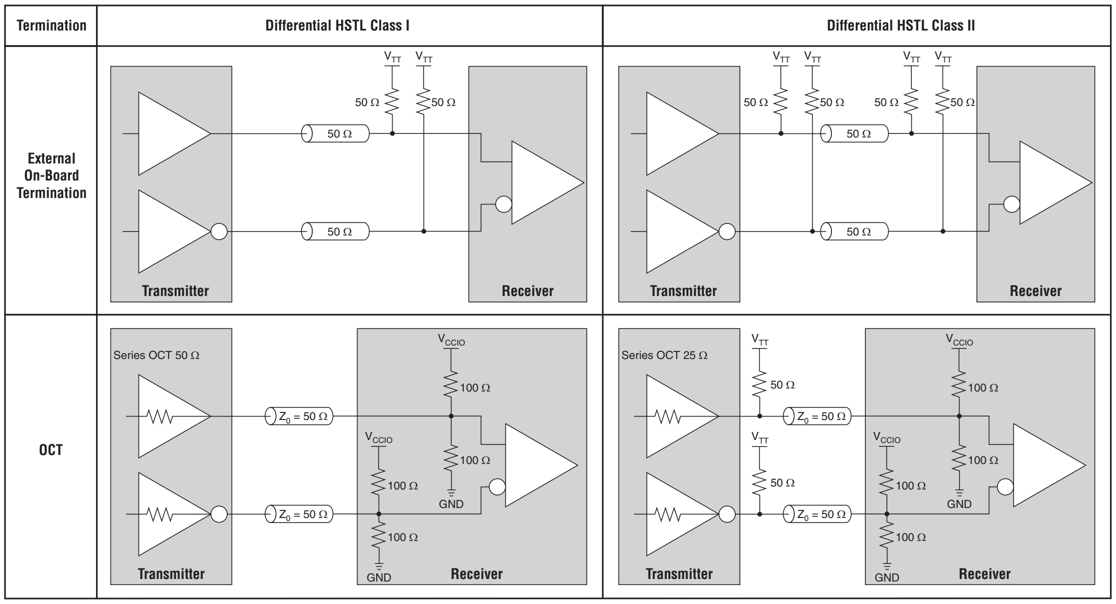
Figure 1–33. Differential SSTL I/O Standard Termination



Differential HSTL I/O Standard Termination

Figure 1–34 shows the details of **Differential HSTL** I/O standard termination on Cyclone V devices.

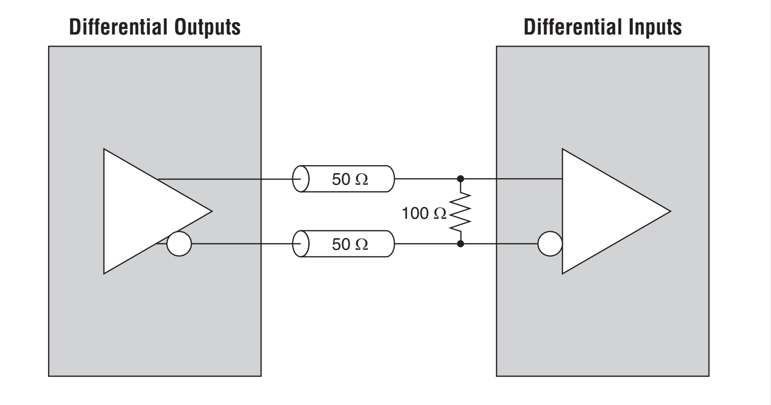
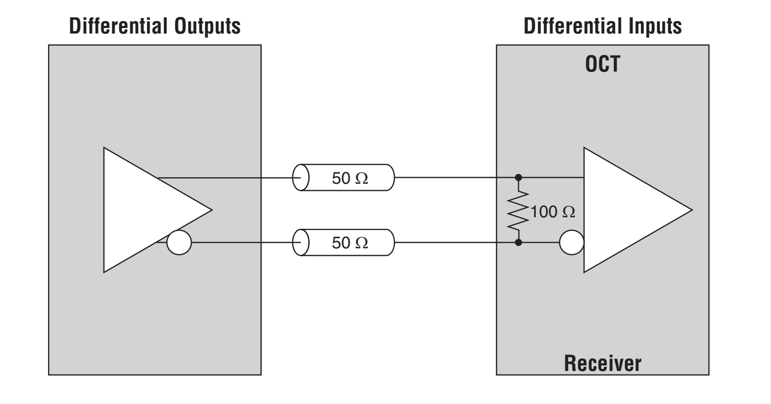
Figure 1–34. Differential HSTL I/O Standard Termination



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.

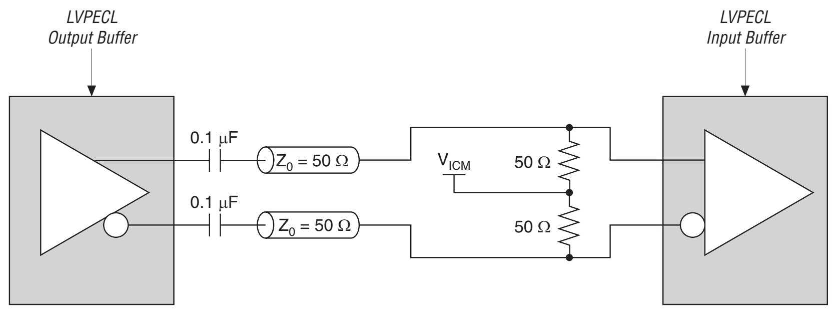
Figure 1–35. LVDS I/O Standard Termination

Termination	LVDS
External On-Board Termination	 Differential Outputs Differential Inputs 50 Ω 50 Ω 100 Ω
OCT Receiver (True LVDS Output)	 Differential Outputs Differential Inputs OCT 50 Ω 50 Ω 100 Ω Receiver

LVPECL I/O Standard Termination

Cyclone 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)*

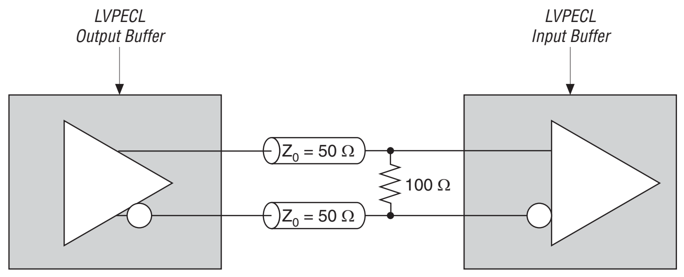


Note to Figure 1–36:

(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 Cyclone V **LVPECL** input buffer specification, as shown in Figure 1–37.

Figure 1–37. LVPECL DC-Coupled Termination *(1)*



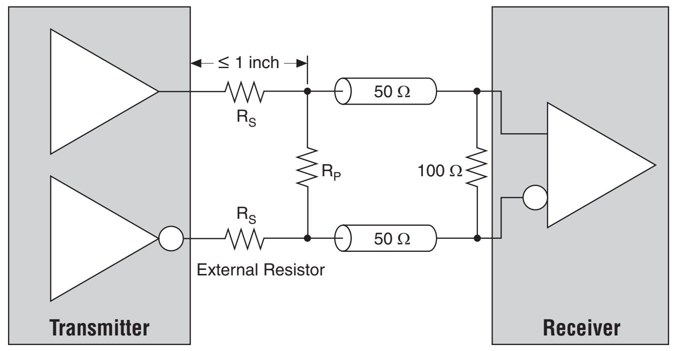
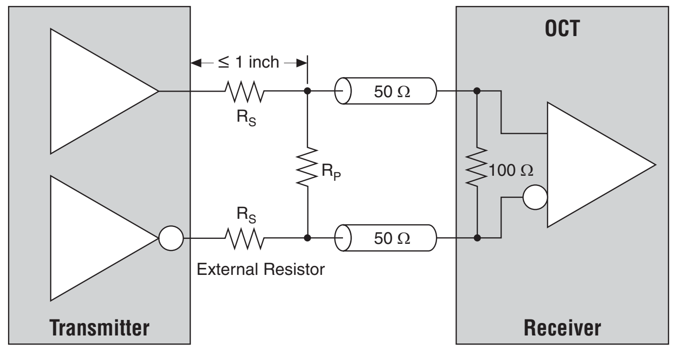
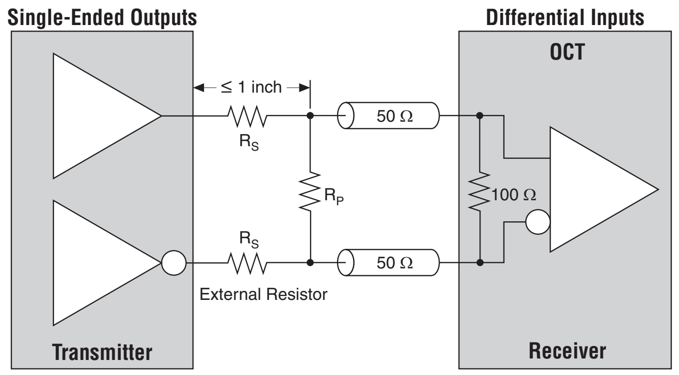
Note to Figure 1–37:

(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)

Termination	Three-Resistor Network (LVDS/RSDS_E_3R/mini-LVDS_E_3R)
External On-Board Termination	 ≤ 1 inch R_S R_P 50 Ω 100 Ω External Resistor Transmitter Receiver
OCT	 OCT ≤ 1 inch R_S R_P 50 Ω 100 Ω External Resistor Transmitter Receiver
OCT Receive (Single-Ended Output with Three-Resistor Network, LVDS_E_3R)	 Single-Ended Outputs Differential Inputs OCT ≤ 1 inch R_S R_P 50 Ω 100 Ω External Resistor Transmitter Receiver

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 Cyclone 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 Cyclone 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 Cyclone 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 Cyclone 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 a Cyclone 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 Cyclone 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. The VCCPD grouping on Cyclone V are as follows. Each line item is a separate group:

- BANK 3A
- BANK3B + BANK4A
- BANK5A
- BANK5B
- BANK6A
- BANK7A + BANK8A

For example, 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}. This would also require each I/O bank in the same group to also use a 3.0-V V_{CCIO}.

If one I/O bank in a group uses a 2.5-V V_{CCPD}, other I/O banks in the same group must also use 2.5-V V_{CCPD}. However, each I/O bank can use different V_{CCIO} voltages provided they are 1.2, 1.25, 1.35, 1.5, 1.8, or 2.5 V.

High-Speed Differential I/O Interfaces

This section provides basic information about the high-speed differential I/O interfaces of Cyclone V devices.

The information in this section should be used in conjunction with the *I/O Features in Cyclone V Devices* chapter.

Differential Pin Placement Guidelines

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.

When you use LVDS channels, adhere to the guidelines in the following sections.

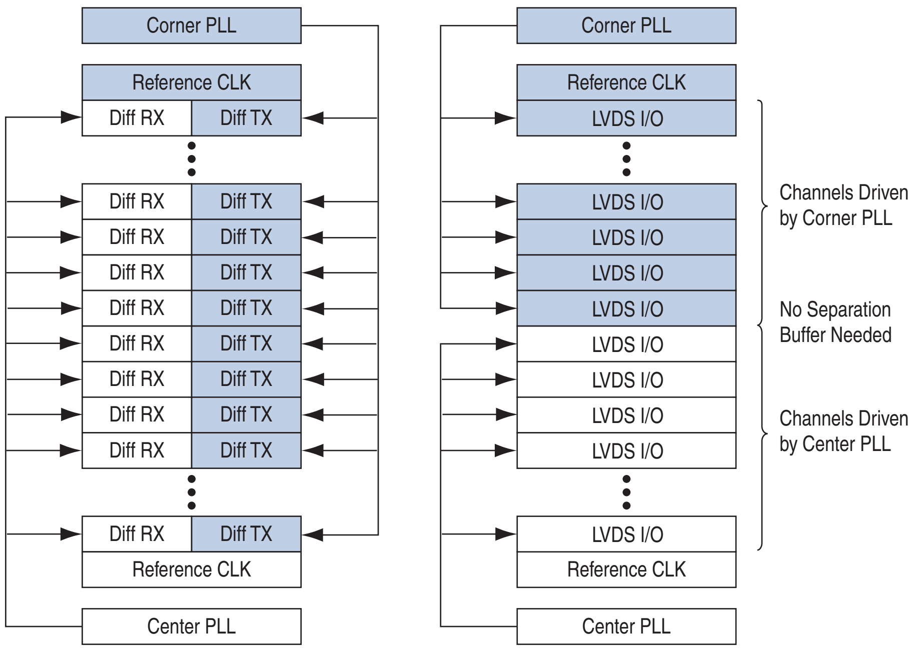
LVDS Channel Driving Distance

Each PLL can drive all the LVDS channels in the entire quadrant.

Using Corner and Center PLLs

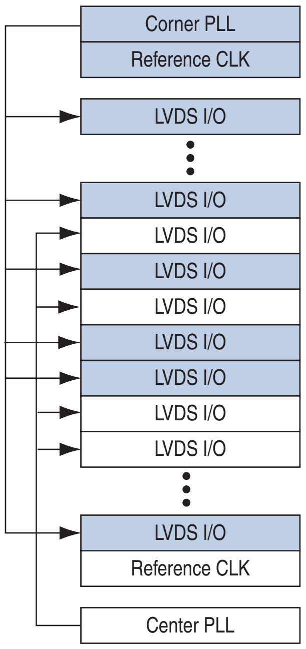
You can use a corner PLL to drive all transmitter channels and a center PLL to drive all LVDS receiver channels in the same I/O bank. You can drive a transmitter channel and a receiver channel in the same LAB row by two different PLLs, as shown in Figure 1–39.

Figure 1–39. Corner and Center PLLs Driving LVDS Differential I/Os in the Same Quadrant



A corner PLL and a center PLL can drive duplex channels in the same I/O quadrant 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 1–40.

Figure 1–40. Invalid Placement of LVDS I/Os Due to Interleaving of Channels Driven by the Corner and Center PLLs



External Memory Interfaces

This section describes the basic information of memory interface pin, delay-locked loop (DLL), and DQS logic block of Cyclone V devices.

Use the information in this section in conjunction with the *External Memory Interfaces in Cyclone V Devices* chapter.

Memory Interface Pin

A typical memory interface requires data (DQ), data strobe (DQS and DQSn), address, command, and clock (CK and CK#) pins. Some memory interfaces use data mask (DM) pins to enable write masking.

For the Cyclone V pin tables for a particular Cyclone V device, refer to the Cyclone V Device Pin-Out Files page of the Altera website.

DQ pins are bidirectional signals, as in DDR3 and DDR2 SDRAM common I/O interfaces. You must assign the write clocks to the DQS/DQSn pins associated to this write DQ/DQS group.

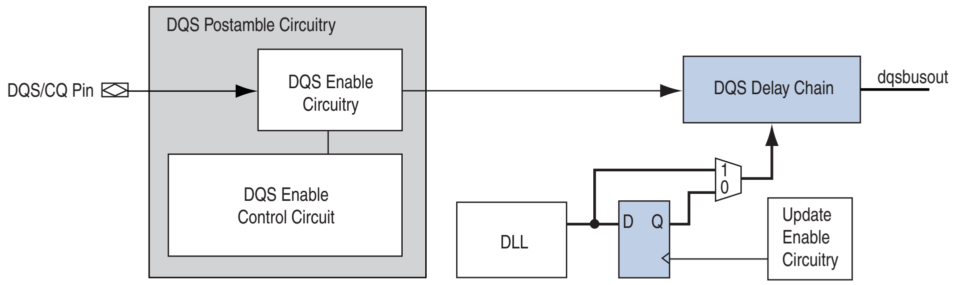
Delay-Locked Loop

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 1–41 shows a simple block diagram of the DQS logic block in the Cyclone V devices.

Figure 1–41. 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

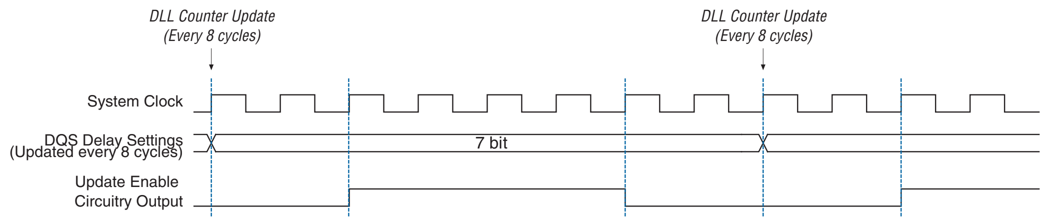
DQS delay chains consist of a set of variable delay elements to allow the input DQS/CQ signals to be shifted by the amount specified by the DQS phase-shift circuitry or the logic array. There are two delay elements in the DQS delay chain. The DQS/CQ pin is shifted by the DQS delay settings.

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 1–42 shows an example waveform of the update enable circuitry output.

Figure 1–42. DQS Update Enable Waveform



Configuration, Design Security, and Remote System Upgrades

This section describes the basic information of supported configuration schemes for Cyclone 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 Cyclone 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 Cyclone 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 Cyclone V Devices* chapter.

Reset

After power-up, the Cyclone 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 Cyclone 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 Cyclone 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 Cyclone V device on the falling edge of `DCLK`.

After the Cyclone 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 the MSEL pins, refer to the *Configuration, Design Security, and Remote System Upgrades in Cyclone V Devices* chapter.

Configuration Error

If an error occurs during configuration, Cyclone 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 Cyclone 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 the `tSTATUS` and `tCFG` timing parameters, refer to the *Device Datasheet for Cyclone 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 Cyclone 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 Cyclone 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 Cyclone 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 Cyclone V devices.

Table 1–6. Initialization Clock Source Option and the Maximum Frequency for Cyclone 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, Cyclone 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 Cyclone V Devices* chapter.

User Mode

The Cyclone 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 Cyclone V devices. FPP is supported in two data widths—8 bits and 16 bits. You can perform an FPP configuration of Cyclone 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 Cyclone 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 Cyclone 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) (Part 1 of 2)

Configuration Scheme	Decompression	Design Security	DCLK-to-DATA[] Ratio
FPP x8	Disabled	Disabled	1
	Disabled	Enabled	1
	Enabled	Disabled	2
	Enabled	Enabled	2

Table 1–7. DCLK-to-DATA[] Ratio [(1)](#) **(Part 2 of 2)**

Configuration Scheme	Decompression	Design Security	DCLK-to-DATA[] Ratio
FPP x16	Disabled	Disabled	1
	Disabled	Enabled	2
	Enabled	Disabled	4
	Enabled	Enabled	4

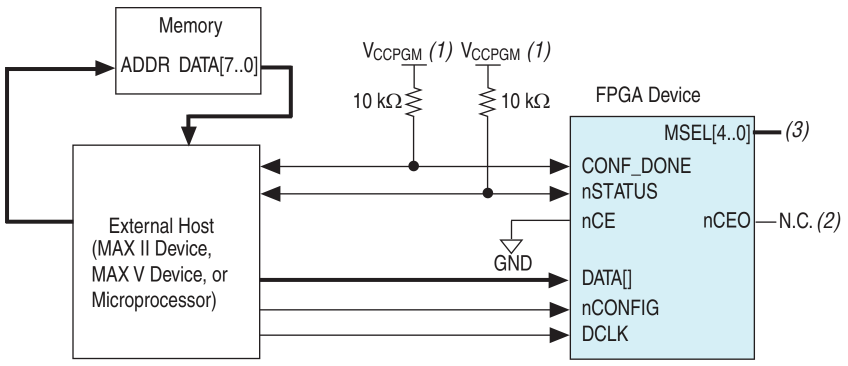
Note to Table 1–7:

(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. Cyclone 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 Cyclone V device.

Figure 1–43 shows the configuration interface connections between the Cyclone V device and a MAX II or MAX V device for a single device configuration.

Figure 1–43. Single Device FPP Configuration Using an External Host



Notes to Figure 1–43:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Cyclone 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

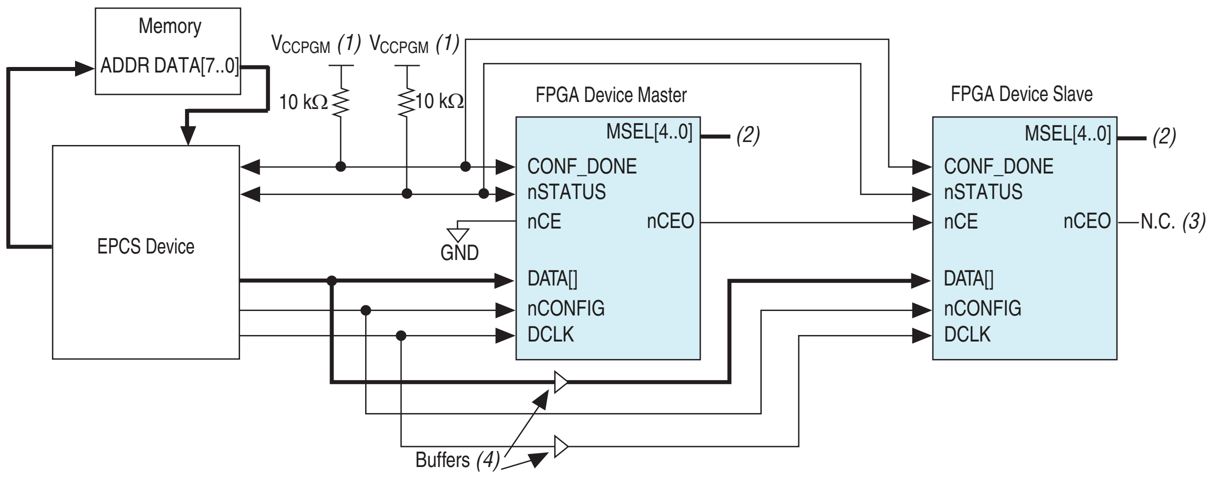
For an FPP multi-device configuration, you can configure all the devices with different sets of configuration data (multiple SRAM object files [**.sof**s]) or with the same configuration data (single **.sof**). In both cases, the `nCONFIG`, `nSTATUS`, `DCLK`, `DATA[]`, and `CONF_DONE` pins are connected 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 Cyclone V device with other FPGA devices that support the FPP x8 configuration scheme.

Figure 1–44 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 1–44. Multi-Device FPP Configuration Using an External Host When Both Devices Receive a Different Set of Configuration Data



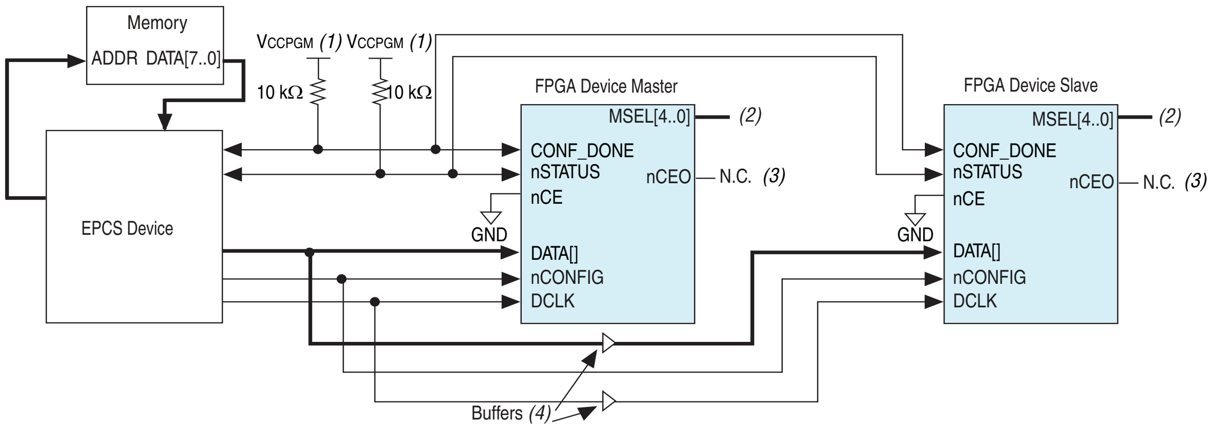
Notes to Figure 1–44:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Cyclone 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 Cyclone V master and slave device for DATA[] and DCLK for every fourth device.

In Figure 1–44, 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 1–45 shows the FPP configuration setup for multiple devices when both Cyclone V devices receive the same configuration data (single **.sof**).

Figure 1–45. Multiple Device FPP Configuration Using an External Host When Both Devices Receive the Same Data



Notes to Figure 1–45:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Cyclone 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 Cyclone V master and slave device for `DATA[]` and `DCLK` for every fourth device. .

In Figure 1–45, 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 Cyclone 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 Cyclone 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 Cyclone 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, Cyclone 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, Cyclone 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.

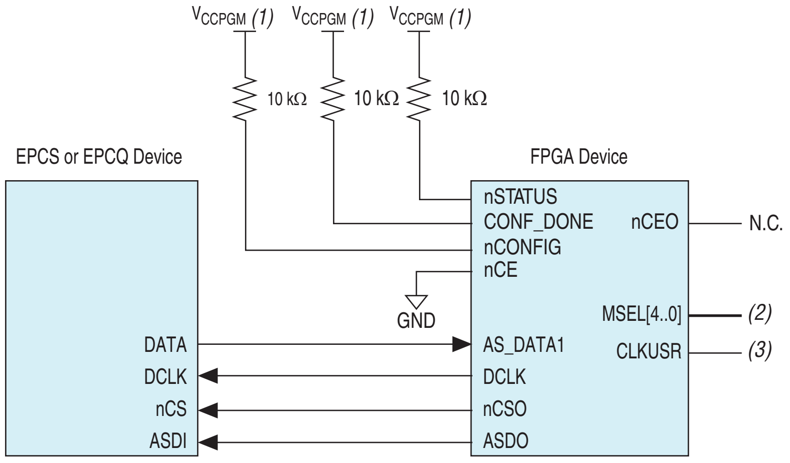
During device configuration, Cyclone V devices read the configuration data using the serial interface, decompress the data if necessary, and configure their SRAM cells. In the AS configuration scheme, the Cyclone V device controls the configuration interface. In the PS configuration scheme, the external 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, Cyclone V devices power-up and begin configuration as an AS x1 mode. After reading the option bits from the programming file, Cyclone 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 1–46 shows the single-device configuration setup for AS x1 mode.

Figure 1–46. Single Device AS x1 Mode Configuration

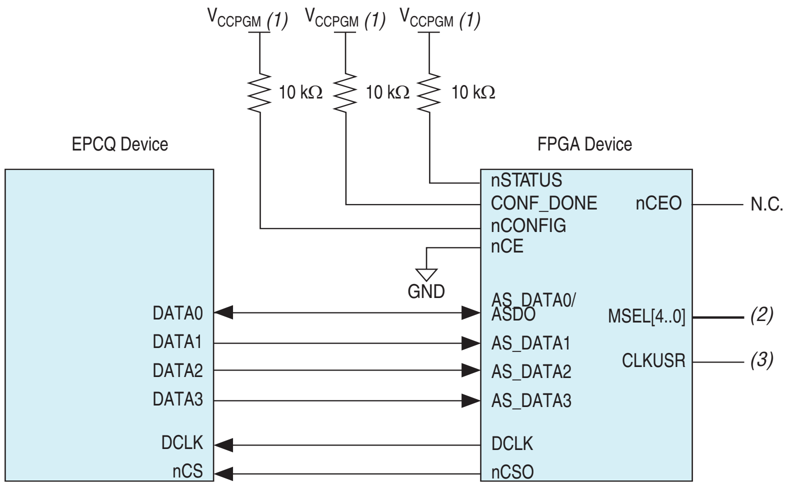


Notes to Figure 1–46:

(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 1–47 shows the single-device configuration setup for AS x4 mode.

Figure 1–47. Single Device AS x4 Mode Configuration



Notes to Figure 1–47:

(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 Cyclone 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, Cyclone 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, Cyclone 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. Cyclone 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 Cyclone 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.

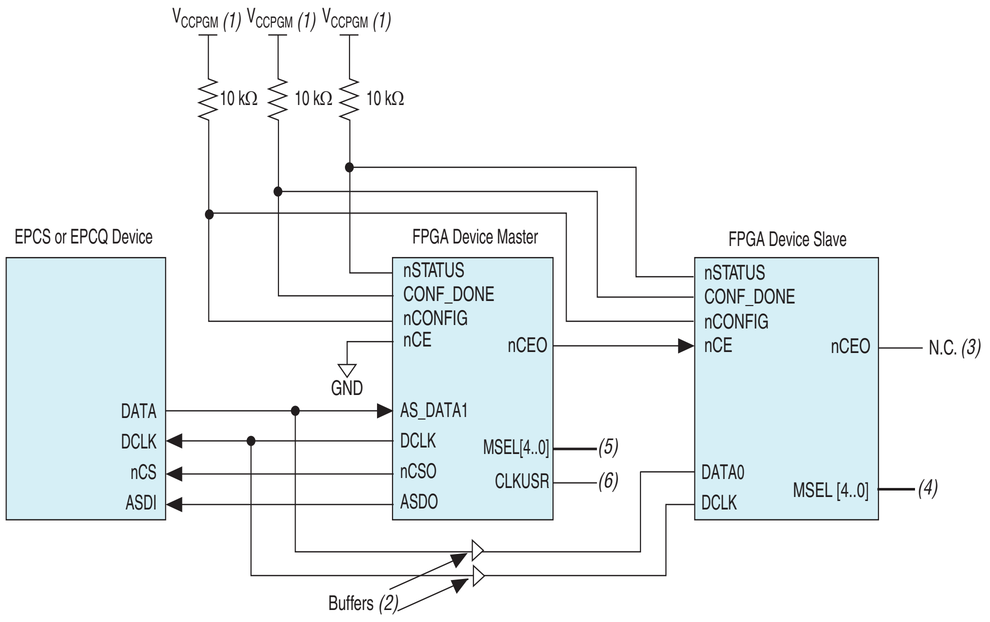
☞ The AS multi-device configuration scheme does not support `DCLK` frequency of 100 MHz.

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 Cyclone 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 Cyclone 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 1–48 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 1–48. AS Multi-Device Configuration When Both Devices in the Chain Receive Different Sets of Configuration Data *(1)*



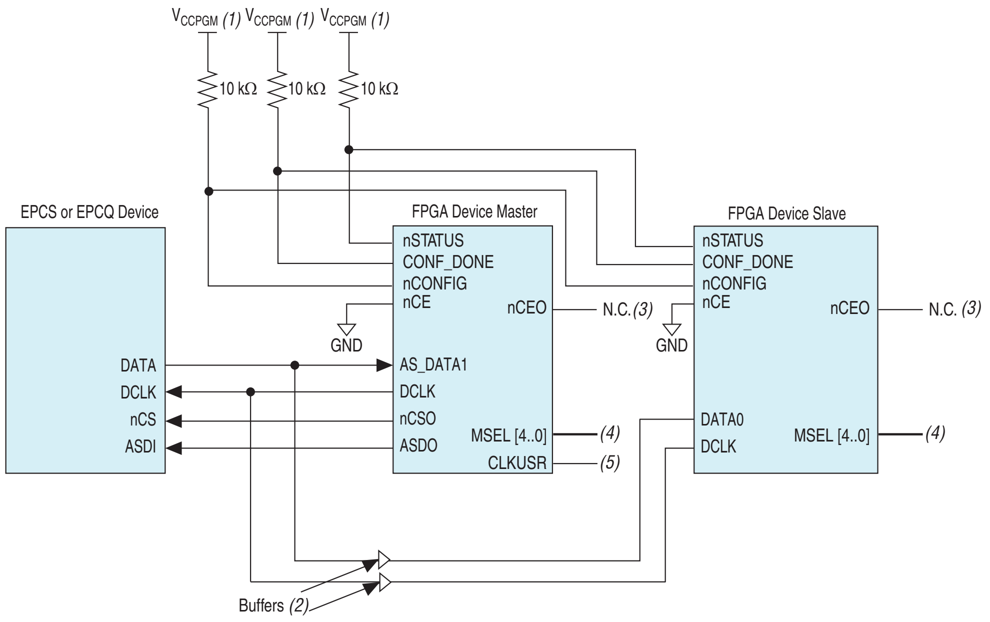
Notes to Figure 1–48:

(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 Cyclone 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 1–48, 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 1–49 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 1–49. AS Multi-Device Configuration When the Devices Receive the Same Data Using a Single .sof [(1)]



Notes to Figure 1–49:

(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 Cyclone 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 Cyclone V Devices

For single- and multi-device AS configurations, the board trace length and loading between the supported EPCS and Cyclone 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 Cyclone V Devices

Cyclone V Device AS Pins	Maximum Board Trace Length from the Cyclone V Device to the Serial Configuration Device for a 12.5/25/50 MHz Operation (Inches)	Maximum Board Trace Length from the Cyclone 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 Cyclone 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 Cyclone V device. This serial interface is clocked by the Cyclone 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 Cyclone 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.

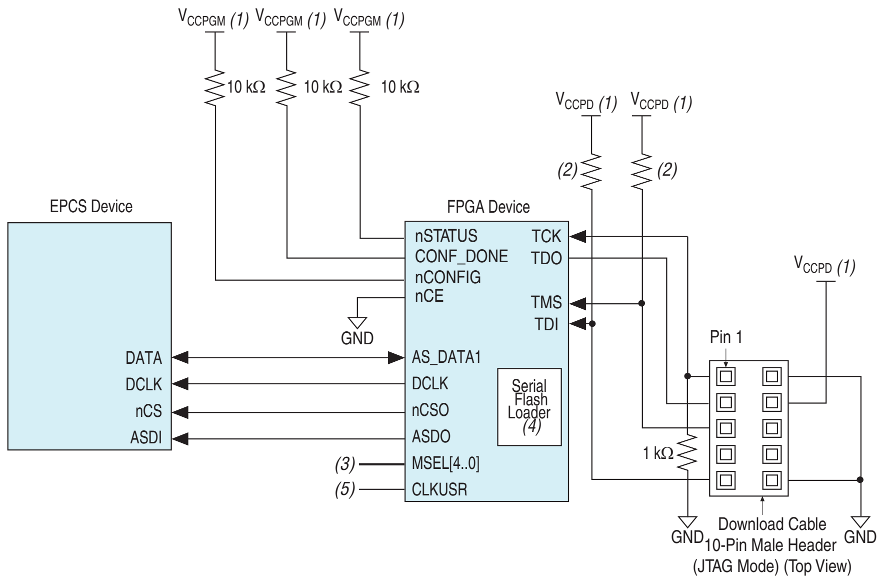
For more information about the SRunner software driver, refer to *AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming*.

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 Cyclone 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 1–50 shows the connection setup when programming the EPCS using the JTAG interface.

Figure 1–50. Connection Setup for Programming the EPCS Using the JTAG Interface



Notes to Figure 1–50:

(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 Cyclone 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.

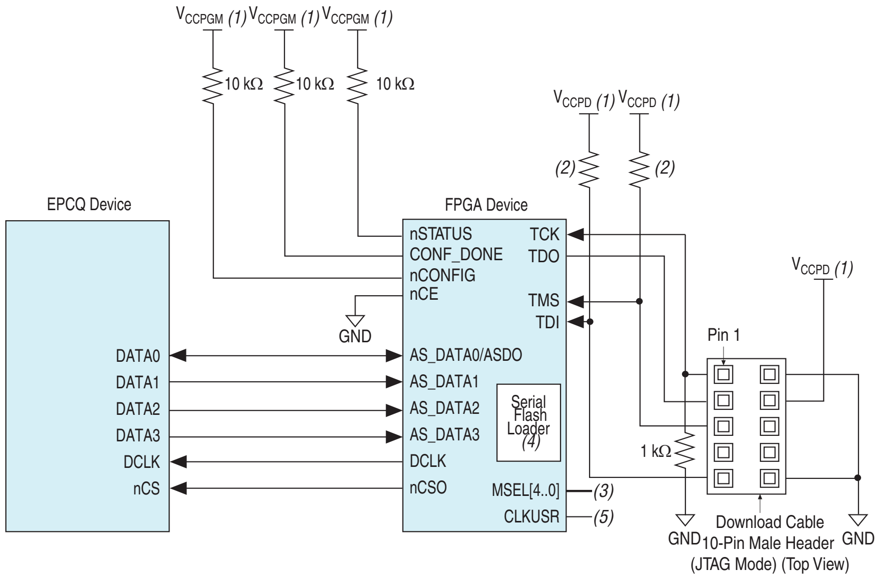
(3) The `MSEL` pin settings vary for different configuration voltage standards and POR delays.

(4) Instantiate SFL in your design to form a bridge between the EPCS and the Cyclone V device.

(5) Use the `CLKUSR` pin to supply the external clock source to drive `DCLK` during configuration.

Figure 1–51 shows the connection setup when programming the EPCQ using the JTAG interface.

Figure 1–51. Connection Setup for Programming the EPCQ Using the JTAG Interface



Notes to Figure 1–51:

(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 Cyclone 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.

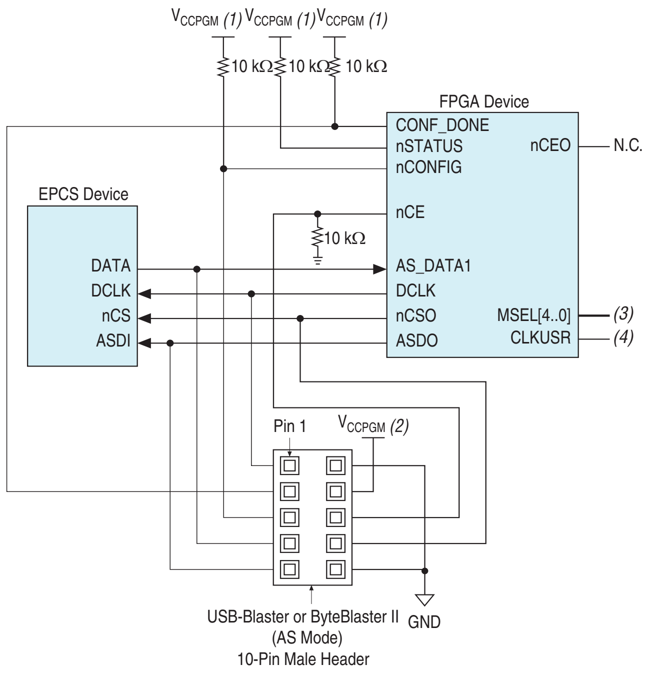
(3) The `MSEL` pin settings vary for different configuration voltage standards and POR delays.

(4) Instantiate SFL in your design to form a bridge between the EPCS and the Cyclone V device.

(5) Use the `CLKUSR` pin to supply the external clock source to drive `DCLK` during configuration.

Figure 1–52 shows the connection setup when programming the EPCS using the AS interface.

Figure 1–52. Connection Setup for Programming the EPCS Using the AS Interface

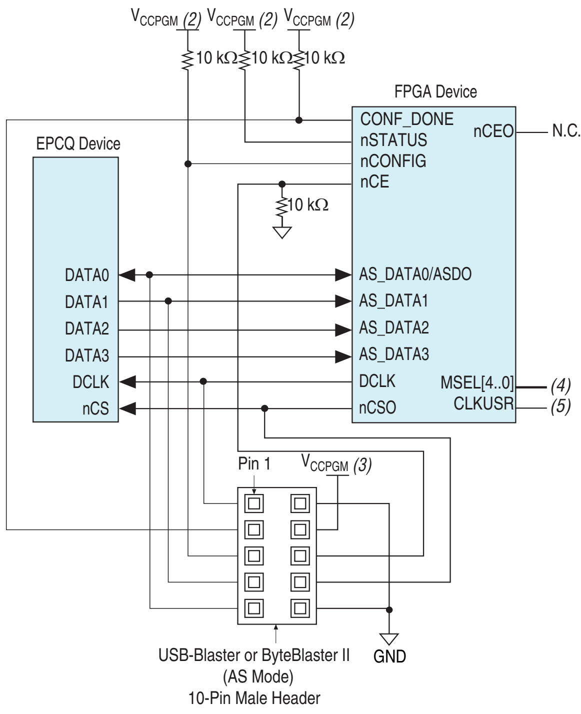


Notes to Figure 1–52:

(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 1–53 shows the connection setup when programming the EPCQ using the AS interface.

Figure 1–53. Connection Setup for Programming the EPCQ Using the AS Interface *(1)*



Notes to Figure 1–53:

(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 Cyclone 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 Cyclone 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 Cyclone V device.

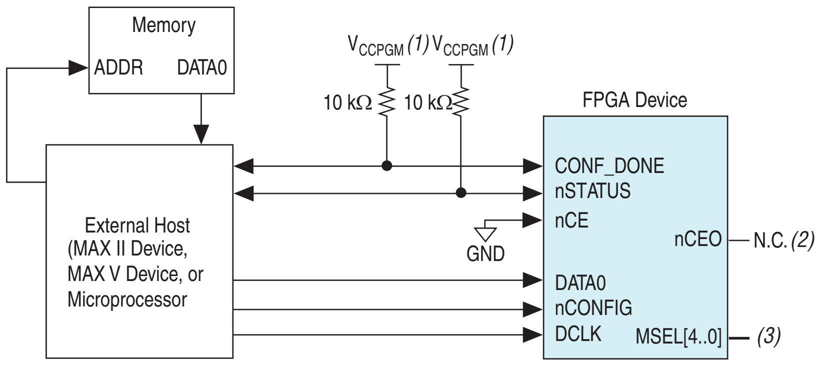
For more information about the PFL megafunction, refer to the *Parallel Flash Loader Megafunction User Guide*.

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 Cyclone 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 1–54 shows the configuration interface connections between a Cyclone V device and a MAX II or MAX V device for single device configuration.

Figure 1–54. Single Device PS Configuration Using an External Host



Notes to Figure 1–54:

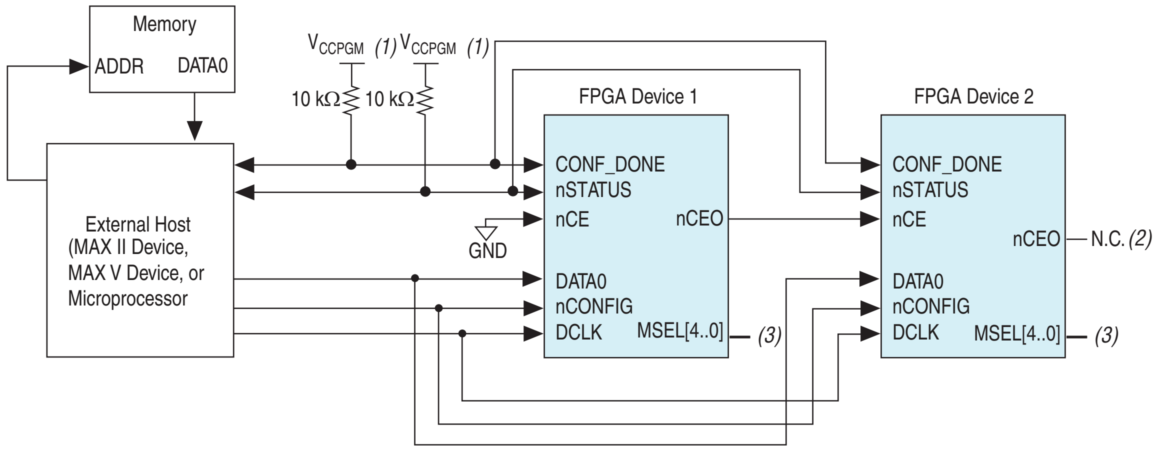
(1) Connect the resistor to a supply that provides an acceptable input signal for the Cyclone V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all the configuration system I/Os with V_{CCPGM}.

(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 1–55 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 1–55. PS Multi-Device Configuration when Both Devices Receive Different Sets of Configuration Data



Notes to Figure 1–55:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Cyclone 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}.

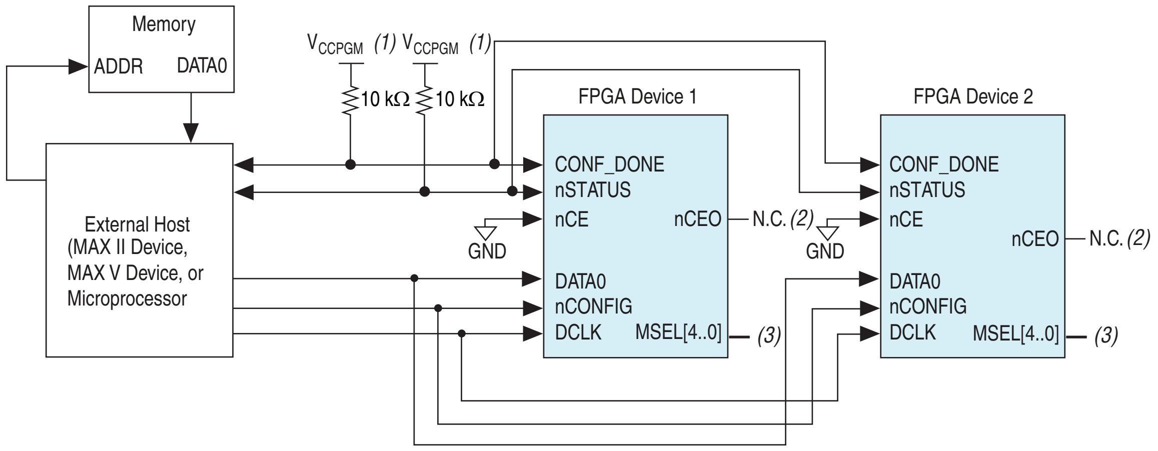
(2) You can leave the `nCEO` pin unconnected or use it as a user I/O pin when it does not feed another device's `nCE` pin.

(3) The `MSEL` pin settings vary for different configuration voltage standards and POR delays.

In Figure 1–55, 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 external host.

Figure 1–56 shows the PS multi-device configuration when all devices receive the same set of configuration data (single **.sof**).

Figure 1–56. PS Multi-Device Configuration When Both Devices Receive the Same Set of Configuration Data



Notes to Figure 1–56:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Cyclone 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 1–56, 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 Cyclone 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 Cyclone V Devices* chapter.

PS Configuration Using a Download Cable

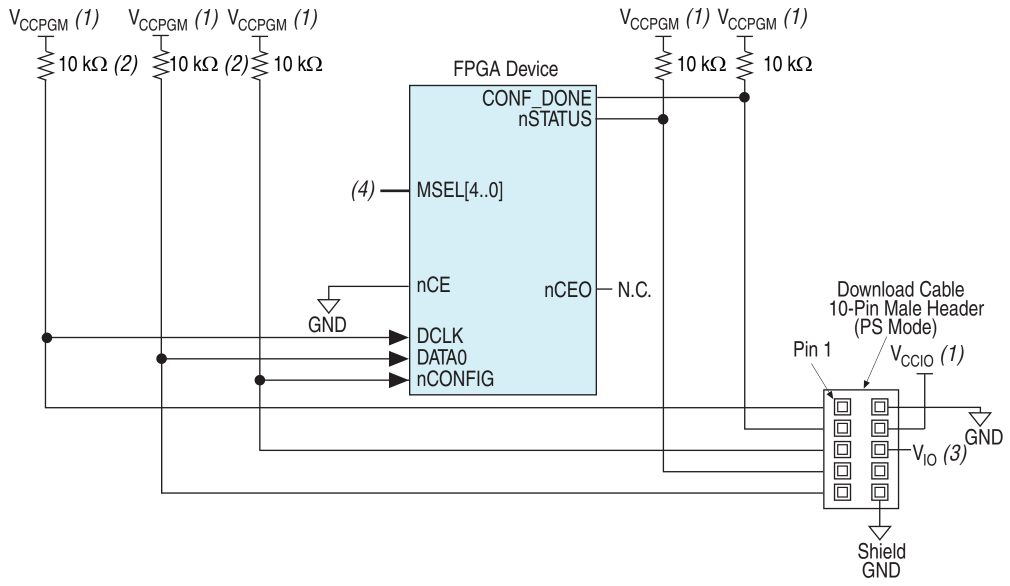
In this section, the generic term “download cable” includes the Altera USB-Blaster USB port download cable, ByteBlaster II parallel port download cable, EthernetBlaster download cable, and EthernetBlaster II download cable.

In a PS configuration with a download cable, a PC acts as a host to transfer data from a storage device to the Cyclone 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 1–57 shows a PS configuration for Cyclone V devices using an Altera download cable.

Figure 1–57. PS Configuration Using an Altera Download Cable



Notes to Figure 1–57:

(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`.

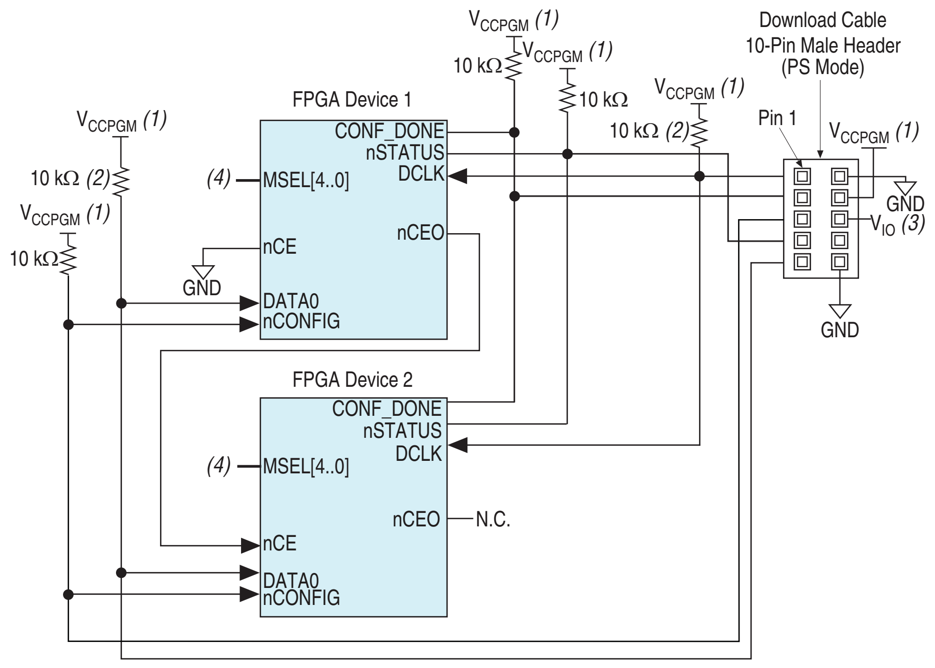
(3) In the USB-Blaster and ByteBlaster II cables, this pin is connected to `nCE` when you use it for AS programming; otherwise, it is a no connect.

(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 Cyclone V devices. Figure 1–58 shows the multi-device PS configuration using an Altera download cable.

Figure 1–58. Multi-Device PS Configuration Using an Altera Download Cable



Notes to Figure 1–58:

(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 1–58, 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 1–81.

For more information about JTAG BST and the commands available using Cyclone V devices, refer to the following documents:

- *JTAG Boundary-Scan Testing in Cyclone V Devices* chapter
- Programming Support for Jam STAPL Language

In Cyclone 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 Cyclone 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 Cyclone V decompression or design security features if you are configuring your Cyclone V device using the JTAG-based configuration.

For more information about `TDI`, `TDO`, `TMS`, and `TCK`, refer to "Device Configuration Pins" on page 1–67.

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 Cyclone 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.

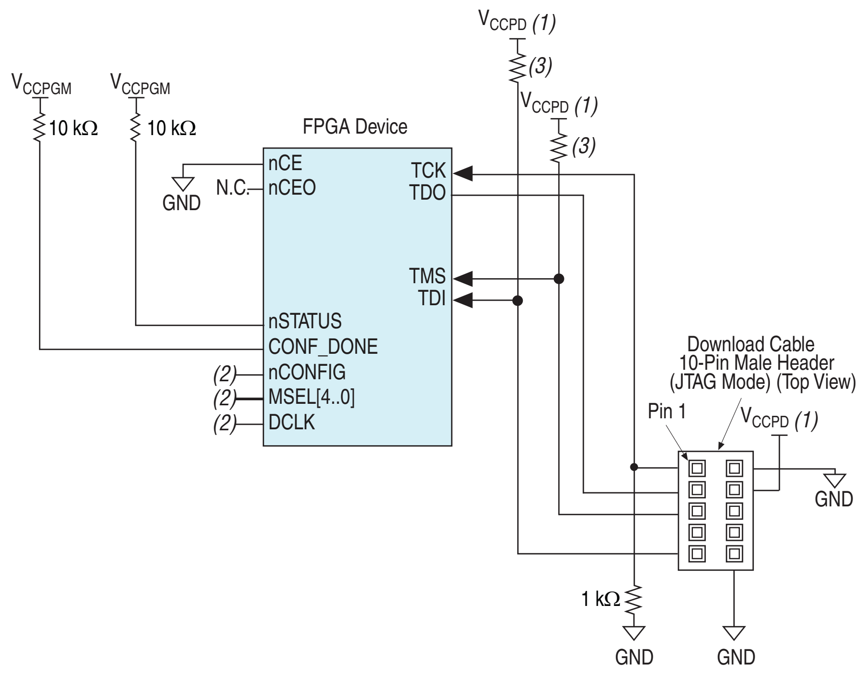
If `CONF_DONE` is low, the Quartus II software indicates that configuration has failed. If `CONF_DONE` is high, the software indicates that configuration was successful. After the configuration data is transmitted serially using the JTAG `TDI` port, the `TCK` port is clocked an additional 1,222 cycles to perform device initialization.

The chip-wide reset (`DEV_CLRn`) and chip-wide output enable (`DEV_OE`) pins on Cyclone 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 1–59 shows the JTAG configuration of a single Cyclone V device.

Figure 1–59. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 1–59:

(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 Cyclone 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.

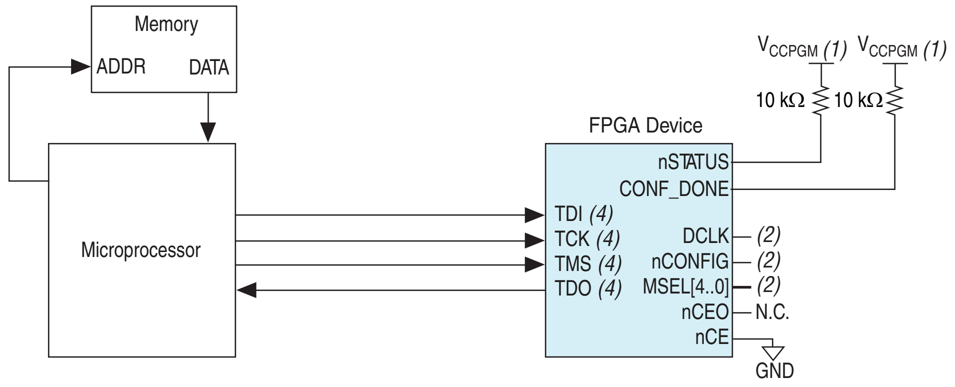
(4) You must connect `nCE` to GND or drive it low for successful JTAG configuration.

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 1–60 shows a JTAG configuration of a Cyclone V device using a microprocessor.

Figure 1–60. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 1–60:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all Cyclone 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 Cyclone 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.

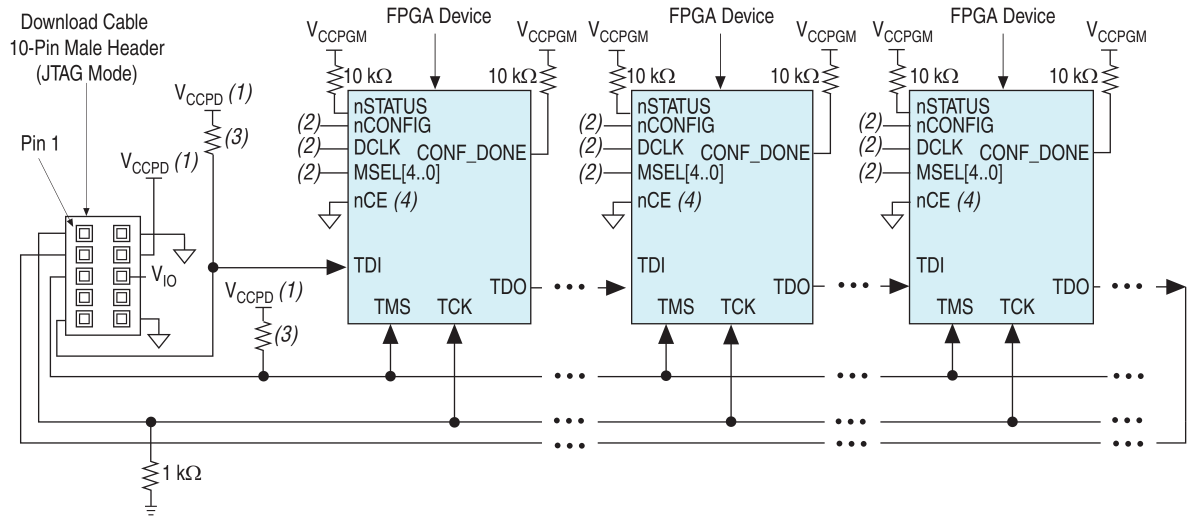
All JTAG instructions (except `BYPASS`, `IDCODE`, and `SAMPLE`) can be issued by first interrupting the configuration and reprogramming the I/O pins using the `CONFIG_IO` instruction.

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 1–61 shows a multi-device JTAG configuration.

Figure 1–61. JTAG Configuration of Multiple Devices Using a Download Cable



Notes to Figure 1–61:

(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 Cyclone 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 1–60, 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 Cyclone 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 Cyclone V device configuration scheme. For more information about the appropriate connections, refer to the *Configuration, Design Security, and Remote System Upgrades in Cyclone 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 Cyclone 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 Cyclone 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

Cyclone 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 Cyclone V devices. During configuration, the Cyclone 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 1–44.

There are two ways to enable compression for Cyclone 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 Cyclone V device, open the **Device and Pin Options** dialog box.
3. In the **Configuration settings** panel, turn on the **Generate compressed bitstreams** option.

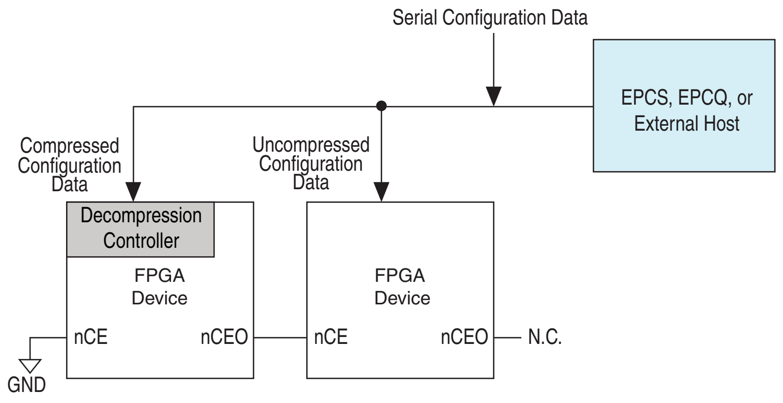
To enable compression when creating programming files from the **Convert Programming Files** window, follow these steps:

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 a Cyclone 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 1–62 shows a chain of two Cyclone V devices. The first Cyclone V device has compression enabled and therefore receives compressed data from the external host. The second Cyclone 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 1–62. Compressed and Uncompressed Serial Configuration Data in the Same Configuration File *(1)*


Serial Configuration Data
EPCS, EPCQ, or External Host
Compressed Configuration Data
Uncompressed Configuration Data
Decompression Controller
FPGA Device
FPGA Device
nCE
nCEO
nCE
nCEO
N.C.
GND

Note to Figure 1–62:

(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. Cyclone 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.

Cyclone V devices feature dedicated remote system upgrade circuitry. A soft logic (either the Nios® II embedded processor or user logic) implemented in a Cyclone 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 Cyclone 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 Cyclone V devices, follow these steps:

1. A Nios II processor (or user logic) implemented in the Cyclone 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 1–63 shows these remote system upgrade steps.

Figure 1–63. Functional Diagram of the Cyclone V Remote System Upgrade Process

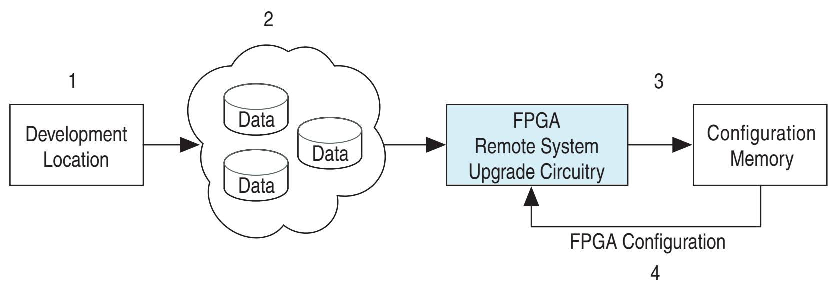
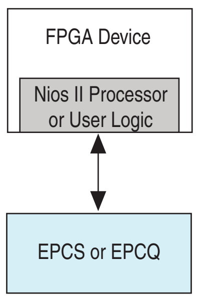


Figure 1–64 shows a block diagram for implementing a remote system upgrade with the Cyclone V AS configuration scheme.

Figure 1–64. Remote System Upgrade Block Diagram for the Cyclone V Device AS Configuration Scheme *(1)*



Note to Figure 1–64:

(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, Cyclone 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 Cyclone 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 Cyclone V device. You may also include the default application image functionality in the factory image. Each Cyclone V device in your system requires one factory image and one or more application images.

Remote Update Mode

Cyclone V remote system upgrade circuitry only supports remote update mode. In remote update mode, Cyclone 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 Cyclone 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.

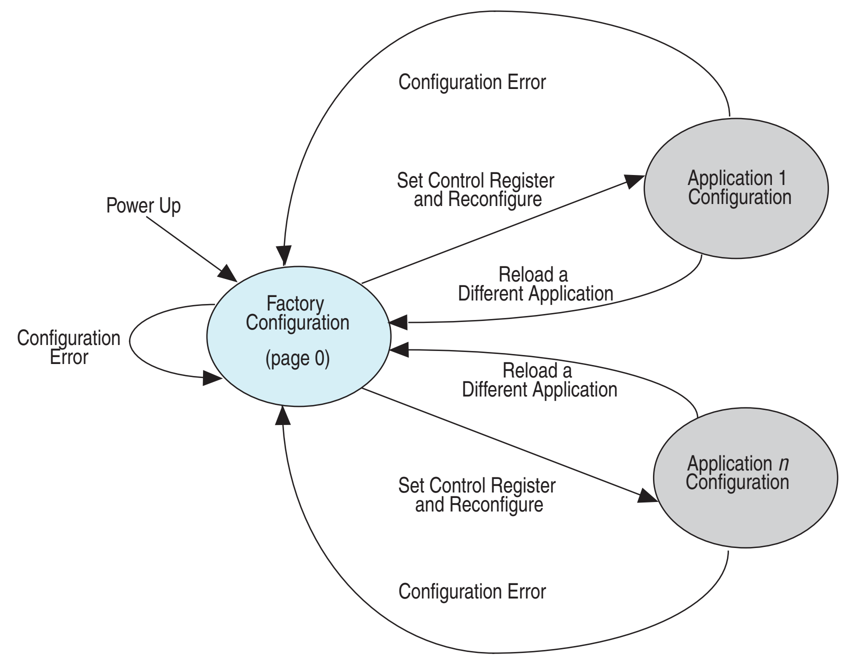
☞ The application image start address can be at any EPCS or EPCQ sector boundary. Altera recommends using different sectors in the EPCS for two images.

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 Cyclone 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 1–65 shows the transitions between the factory and application configurations in remote update mode.

Figure 1–65. Transitions Between Configurations in Remote Update Mode


Configuration Error
Set Control Register and Reconfigure
Application 1 Configuration
Power Up
Reload a Different Application
Factory Configuration
(page 0)
Configuration Error
Reload a Different Application
Application *n* Configuration
Set Control Register and Reconfigure
Configuration Error

After power up or a configuration error, the factory configuration 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 Cyclone V Devices* chapter.

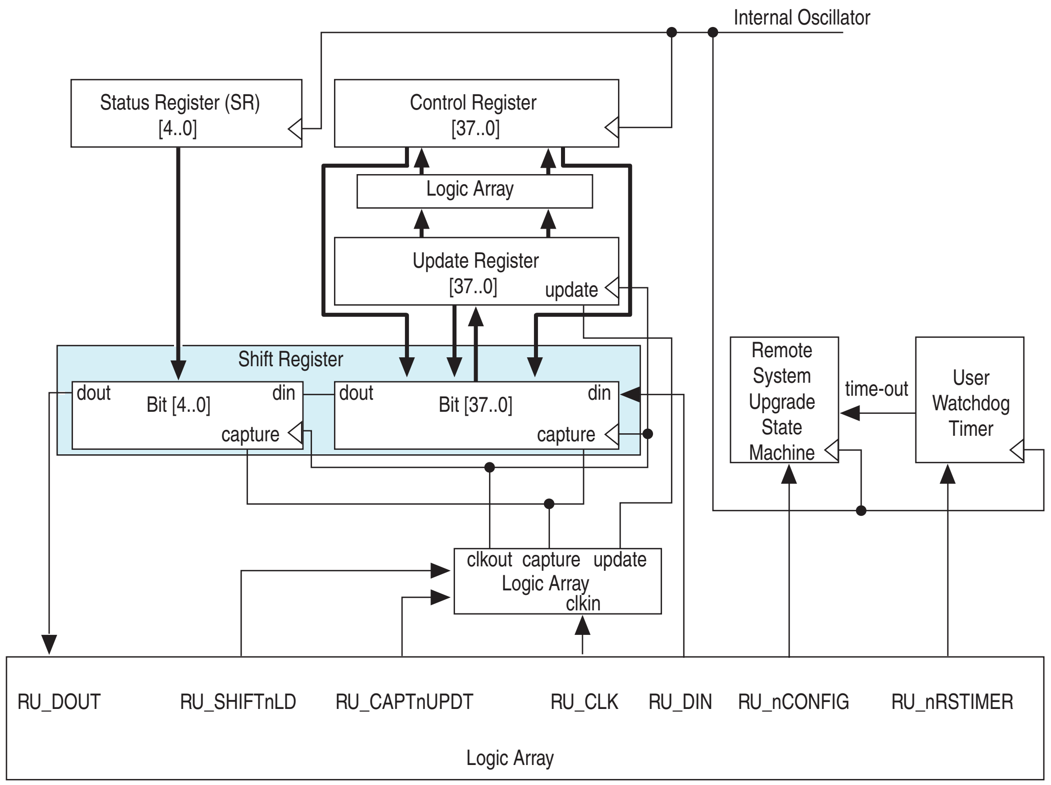
If you are not using the Quartus II software or SRunner software for EPCQ 256 programming, put your EPCQ 256 device into four-byte addressing mode before you program and configure your device.

Dedicated Remote System Upgrade Circuitry

This section describes the implementation of the Cyclone 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 Cyclone 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 1–66 shows the remote system upgrade circuitry.

Figure 1–66. Remote System Upgrade Circuitry *(1)*



Note to Figure 1–66:

(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 Cyclone 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 (Part 1 of 2)

Register	Description
Shift register	This register is accessible by the logic array and allows the update, status, and control registers to be written and sampled by user logic.
Control register	This register contains the current page address, user watchdog timer settings, and one bit specifying whether the current configuration is a factory configuration or an application configuration. During a read operation in an application configuration, this register is read into the shift register. When a reconfiguration cycle is initiated, the contents of the update register are written into the control register.

Table 1–10. Remote System Upgrade Registers (Part 2 of 2)

Register	Description
Update register	This register contains data similar to that in the control register. However, it can only be updated by the factory configuration by shifting data into the shift register and issuing an update operation. When a reconfiguration cycle is triggered by the factory configuration, the control register is updated with the contents of the update register. During a capture in a factory configuration, this register is read into the shift register.
Status register	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.

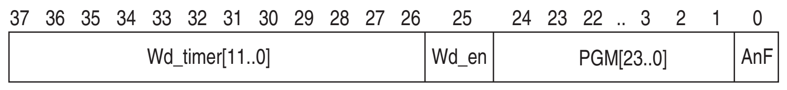
The remote system upgrade control and status registers are clocked by the 10-MHz internal oscillator (the same oscillator that controls the user watchdog timer). However, the remote system upgrade shift and update registers are clocked by the user clock input (`RU_CLK`).

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 1–67 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 1–67. Remote System Upgrade Control Register



The application-not-factory (`AnF`) bit indicates whether the current configuration loaded in the Cyclone 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 (Part 1 of 2)

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

Table 1–11. Remote System Upgrade Control Register Contents (Part 2 of 2)

Control Register Bit	Value (1)	Definition
`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.

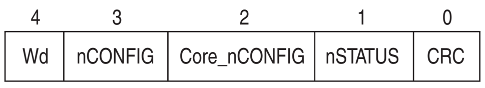
Status Register

The status register specifies the reconfiguration trigger condition. Figure 1–68 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—Cyclone 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 1–68 shows the contents of the status register. The numbers in the figure show the bit positions in a 5-bit register.

Figure 1–68. Remote System Upgrade Status Register *(1)*



Note to Figure 1–68:

(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 1–76, 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 Cyclone 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 Cyclone 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 Cyclone 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 Cyclone 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 Cyclone V devices using the AES. It also describes the security modes available in Cyclone V devices that allow you to use these new features in your designs.

As Cyclone 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. Cyclone 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 Cyclone V design security feature provides the following security protection for your designs:

- Security against copying—the security key is securely stored in the Cyclone V device and cannot be read out through any interface. In addition, as configuration file read-back is not supported in Cyclone 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 Cyclone V configuration file formats are proprietary and the file contains millions of bits that require specific decryption. In addition, the Cyclone V devices are manufactured on the most advanced 28-nm process technology, making this process very difficult.
- Security against tampering—Cyclone 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 Cyclone 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 Cyclone V device first decrypts and then decompresses the configuration file.

When you use design security with Cyclone 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 1–44.

JTAG Secure Mode

When you enable the tamper-protection bit, Cyclone V devices are in JTAG secure mode after power-up. During JTAG secure mode, many JTAG instructions are disabled. Cyclone 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 the JTAG binary instruction code related to the `LOCK` and `UNLOCK` instructions, refer to the *JTAG Boundary-Scan Testing in Cyclone V Devices* chapter.

Security Key Types

Cyclone 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 Cyclone V Devices* chapter.

For more information about the V_{CCBAT} pin connection recommendations, refer to the *Cyclone V Device Family Pin Connection Guidelines*.

Security Modes

Table 1–13 lists the security modes available in Cyclone V devices.

Table 1–13. Supported Security Modes (Part 1 of 2)

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

Table 1–13. Supported Security Modes (Part 2 of 2)

Security Mode	Tamper protection Bit Setting	Device Accepts Unencrypted File	Device Accepts Encrypted File	Security Level
Non-volatile Key with Tamper Protection Bit Set	Set *(2)*	No	Yes	Secure with tamper resistant

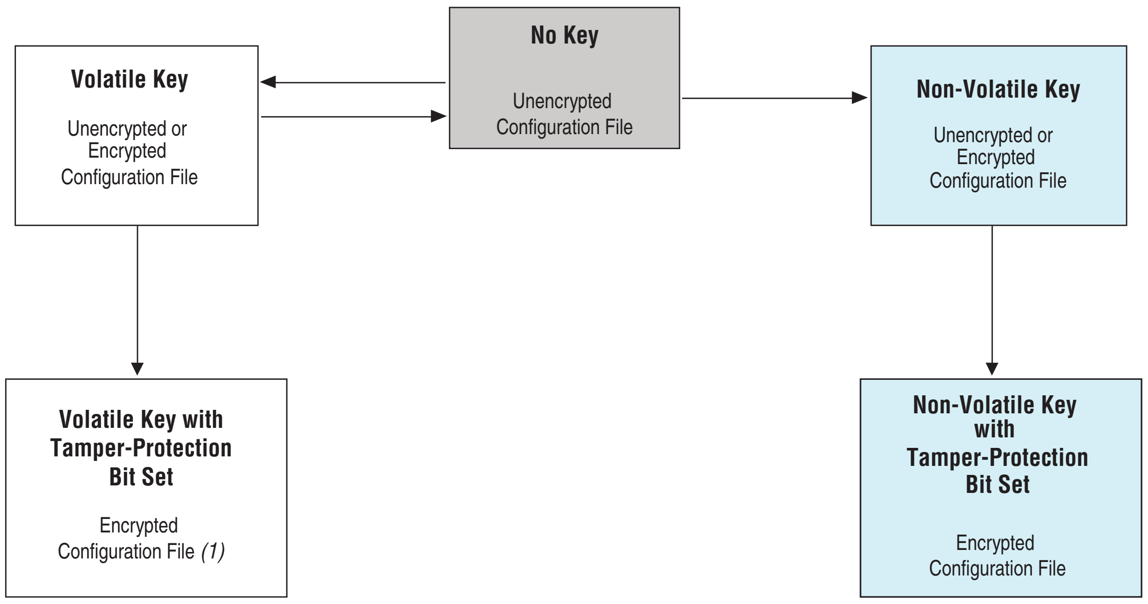
Notes to Table 1–13:

(1) Use the unencrypted configuration bitstream support only for board-level testing.

(2) Enabling the tamper protection bit disables test mode in Cyclone 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 1–69 shows the sequence of the security modes available in Cyclone V devices.

Figure 1–69. Cyclone V Security Modes—Sequence and Restrictions


No Key
Unencrypted Configuration File
Volatile Key
Unencrypted or Encrypted Configuration File
Non-Volatile Key
Unencrypted or Encrypted Configuration File
Volatile Key with Tamper-Protection Bit Set
Encrypted Configuration File (1)
Non-Volatile Key with Tamper-Protection Bit Set
Encrypted Configuration File

Note to Figure 1–69:

(1) Cyclone 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 Cyclone V device is erased.

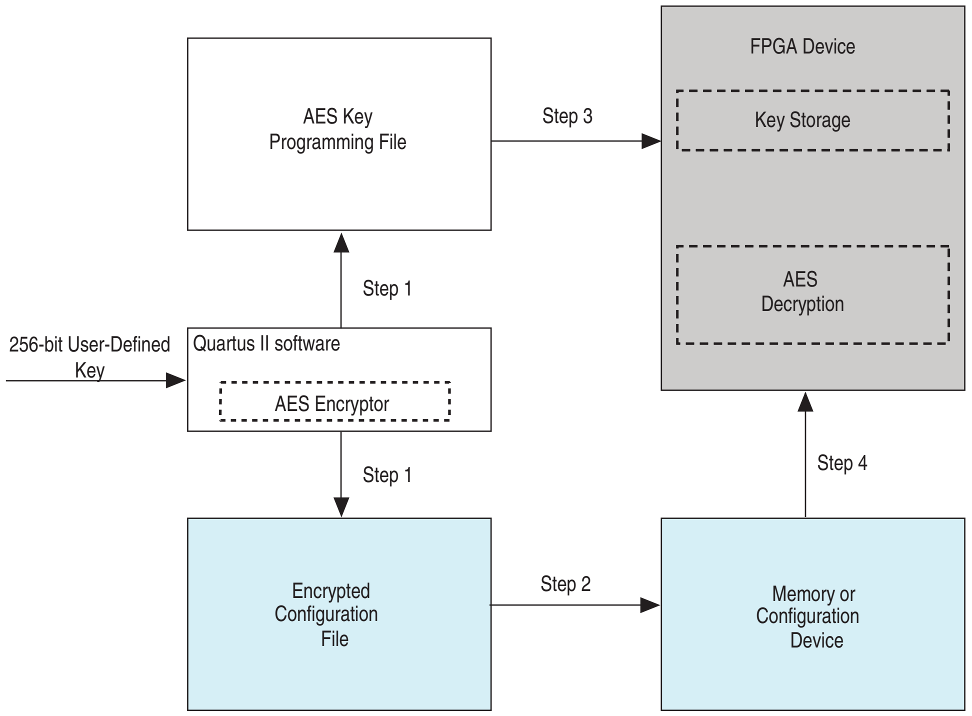
Design Security Implementation Steps

Cyclone V devices are SRAM-based devices. To provide design security, Cyclone V devices require a 256-bit security key for configuration bitstream design security. To carry out secure configuration, follow these steps (Figure 1–70):

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 Cyclone V device through a JTAG interface.

4. Configure the Cyclone V device. At system power-up, the external memory device sends the encrypted configuration file to the Cyclone V device.

Figure 1–70. 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 Cyclone Vdevice.

Use the information in this section in conjunction with the *SEU Mitigation in Cyclone 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 Cyclone 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 Cyclone 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

Cyclone 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 Cyclone V devices, eliminating the need for external logic. Cyclone 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 Cyclone 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.

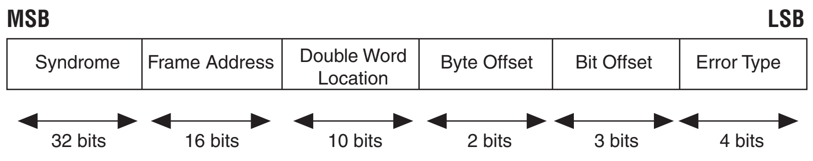
Table 1–14 lists the instruction code for the `SHIFT_EDERROR_REG` JTAG instruction.

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 1–71 shows the content of the EMR.

Figure 1–71. 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 Cyclone 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 Cyclone 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 Cyclone 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 Cyclone 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 Cyclone 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 Cyclone 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 1–89.

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 Cyclone 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 Cyclone 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 1–72 shows the error detection circuitry, syndrome registers, and error injection block.

Figure 1–72. Error Detection Circuitry, Syndrome Register, and Error Injection Block

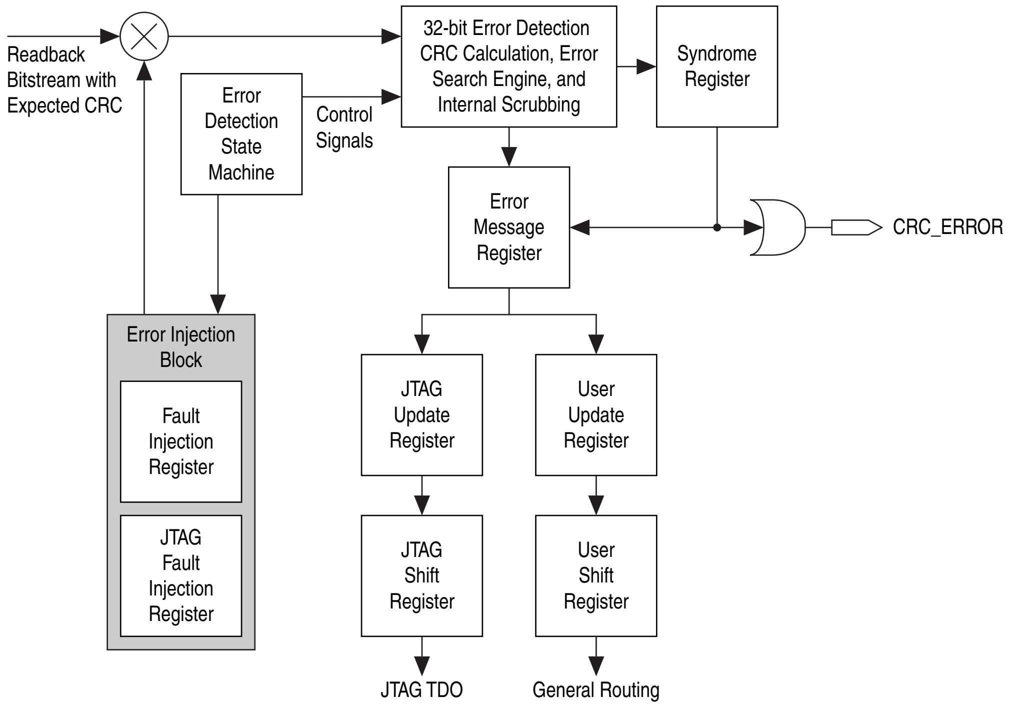


Table 1–20 lists the registers shown in Figure 1–72.

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.
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.

Table 1–20. Error Detection Registers (Part 2 of 2)

Register	Description
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 Cyclone 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 a Cyclone 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 Cyclone 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 Cyclone V devices.

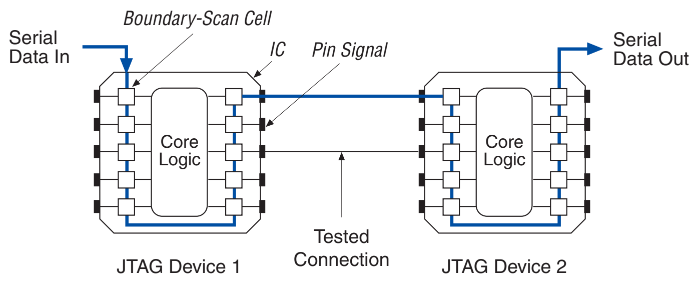
Use the information in this section in conjunction with the *JTAG Boundary-Scan Testing in Cyclone V Devices* chapter.

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 1–73 shows the BST architecture.

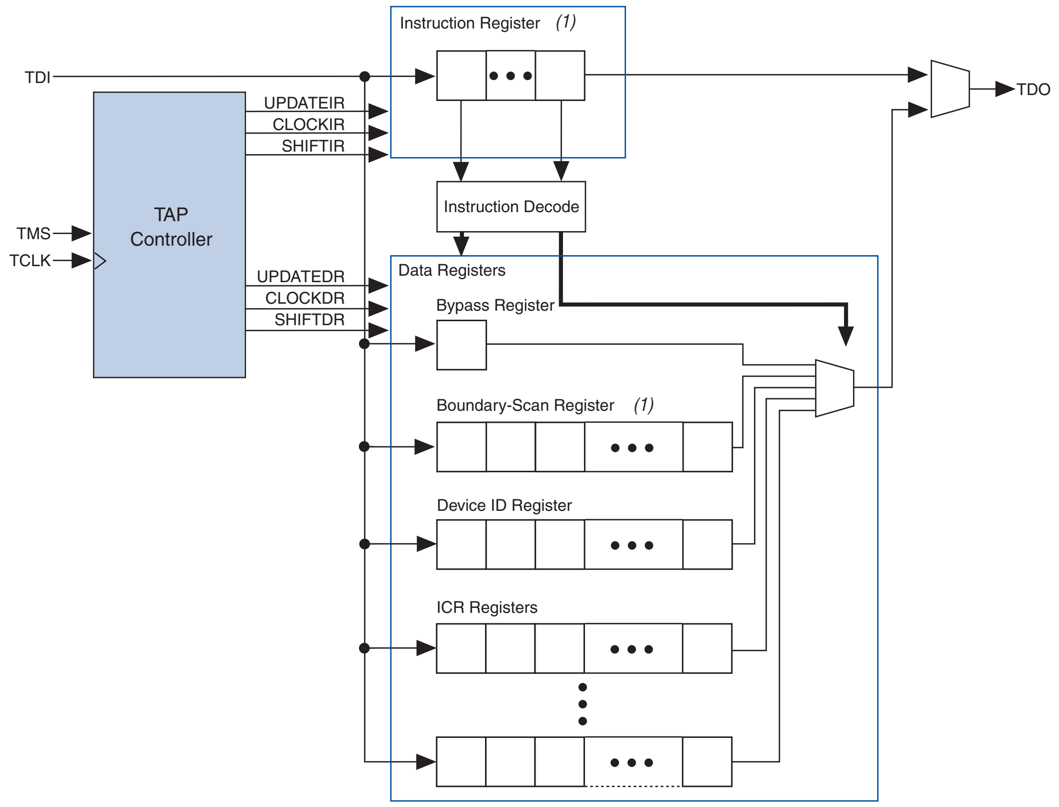
Figure 1–73. IEEE Std. 1149.1 Boundary-Scan Testing



For information about configuring Cyclone V devices through the IEEE Std. 1149.1 circuitry, refer to “JTAG Configuration” on page 1–62.

Figure 1–74 shows a functional model of the IEEE Std. 1149.1 circuitry.

Figure 1–74. 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 1–96. 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

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.
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.

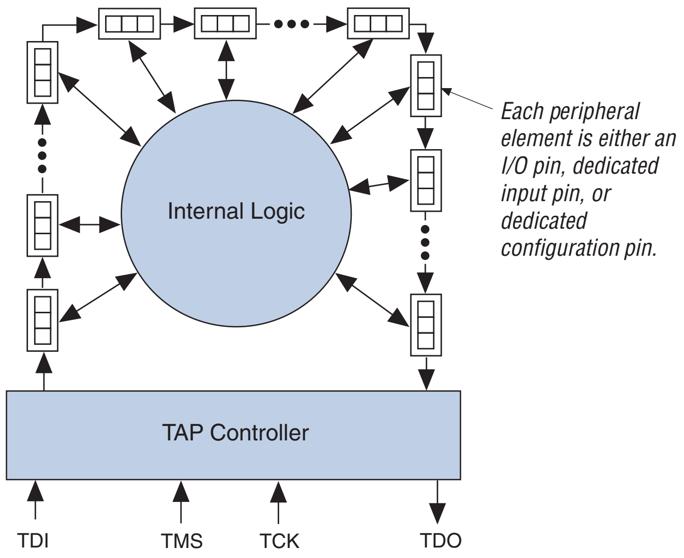
Boundary-scan register—a shift register composed of all the boundary-scan cells (BSCs) of the device.

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 Cyclone V I/O pins. You can use the boundary-scan register to test external pin connections or to capture internal data.

Figure 1–75 shows how test data is serially shifted around the periphery of the IEEE Std. 1149.1 device.

Figure 1–75. Boundary-Scan Register


Internal Logic
TAP Controller
TDI
TMS
TCK
TDO
Each peripheral element is either an I/O pin, dedicated input pin, or dedicated configuration pin.

Boundary-Scan Cells of a Cyclone V Device I/O Pin

The Cyclone 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 1–76 shows the user I/O BSC for Cyclone V devices.

Figure 1–76. Cyclone V Device's User I/O BSC with IEEE Std. 1149.1 BST Circuitry for Cyclone V Devices

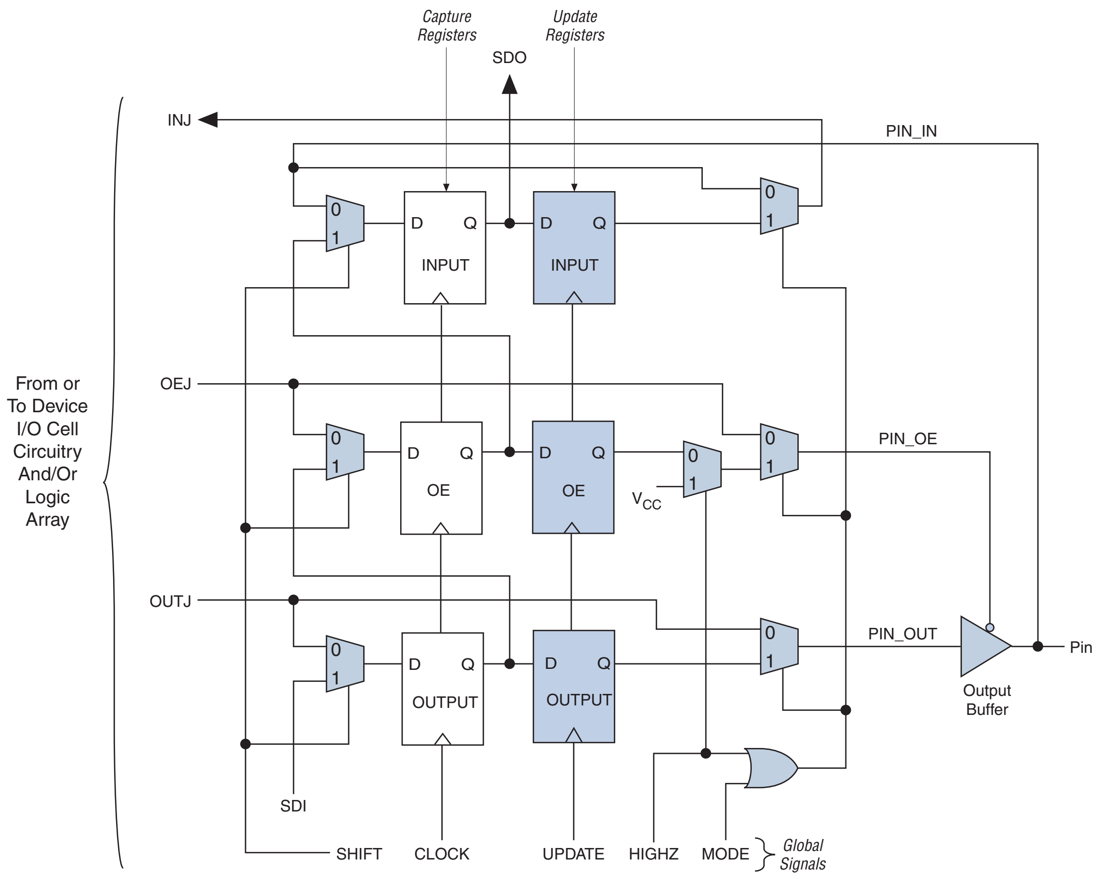


Table 1–22 lists the capture and update register capabilities of all BSCs within Cyclone V devices.

Table 1–22. Boundary Scan Cell Descriptions for Cyclone 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

Table 1–22. Boundary Scan Cell Descriptions for Cyclone 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**	
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 V_{CC} and GND pin types, VREF, and TEMP_DIODE pins do not have BSCs.

(2) No Connect (N.C.).

(3) This includes the PLL_ENA, nCONFIG, MSEL0, MSEL1, MSEL2, MSEL3, MSEL4nCE, VCCSEL, PORSEL, and nIO_PULLUP pins., 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

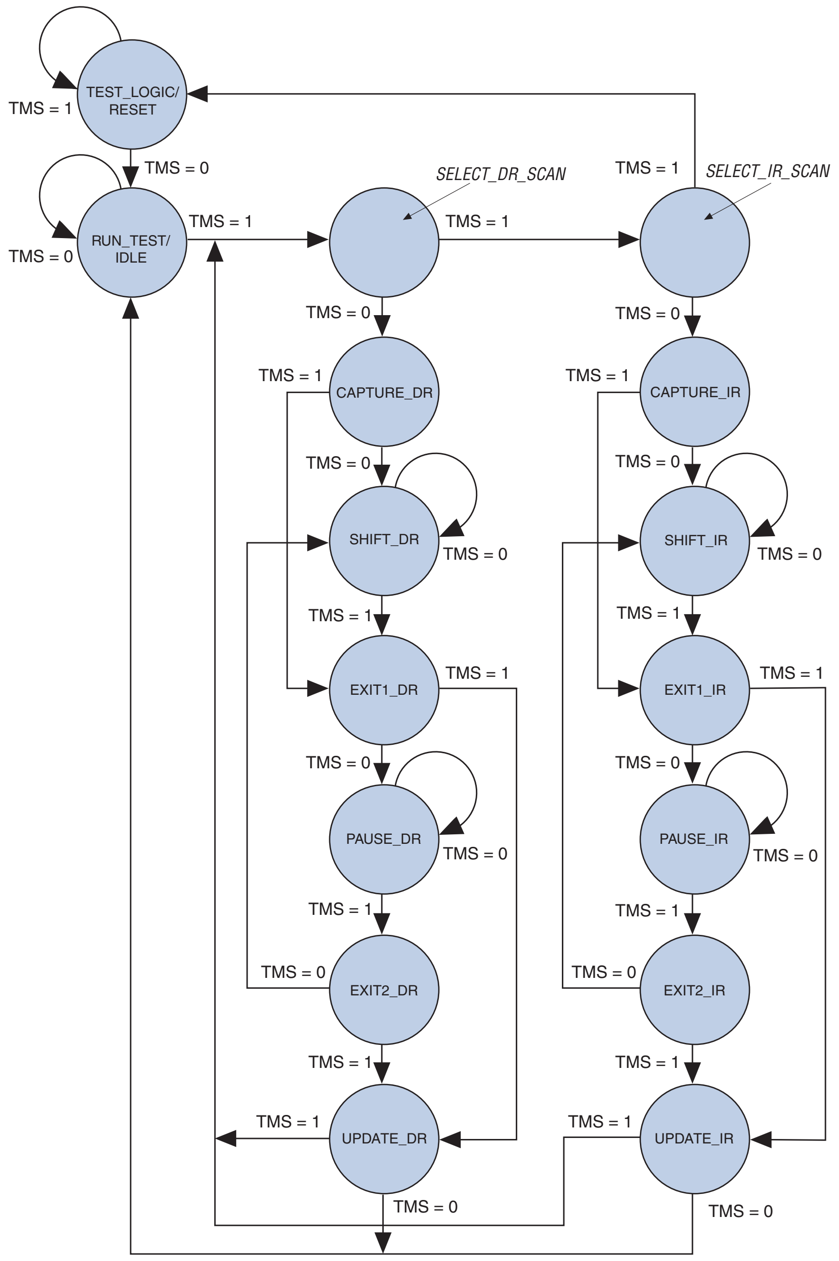
Cyclone V devices support several of the IEEE Std. 1149.1 BST instructions.

For more information about JTAG instruction code, refer to the *JTAG Boundary-Scan Testing in Cyclone 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 1–77 shows the TAP controller state machine.

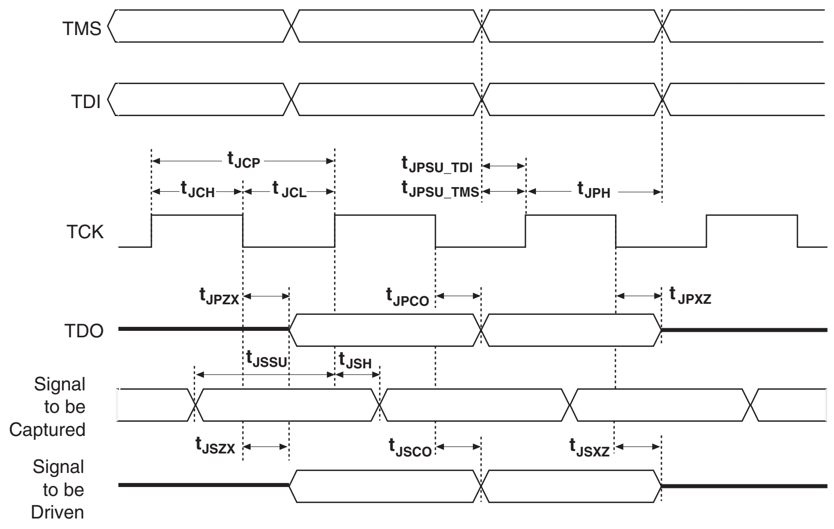
Figure 1–77. IEEE Std. 1149.1 TAP Controller State Machine



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 1–78 shows the timing requirements for the IEEE Std. 1149.1 signals.

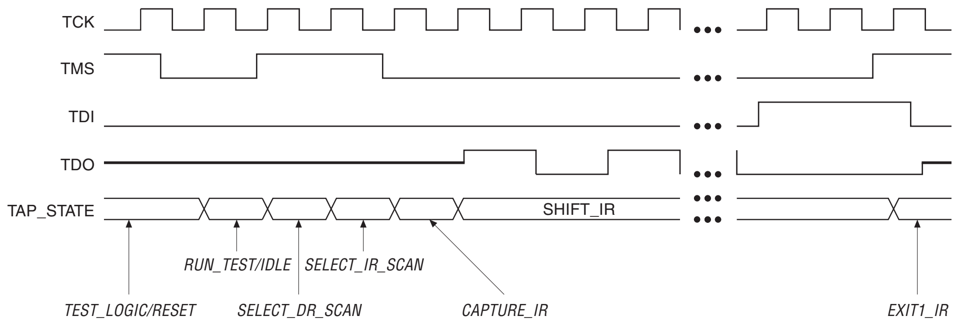
Figure 1–78. IEEE Std. 1149.1 Timing Waveforms



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 1–79 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 1–79. 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:

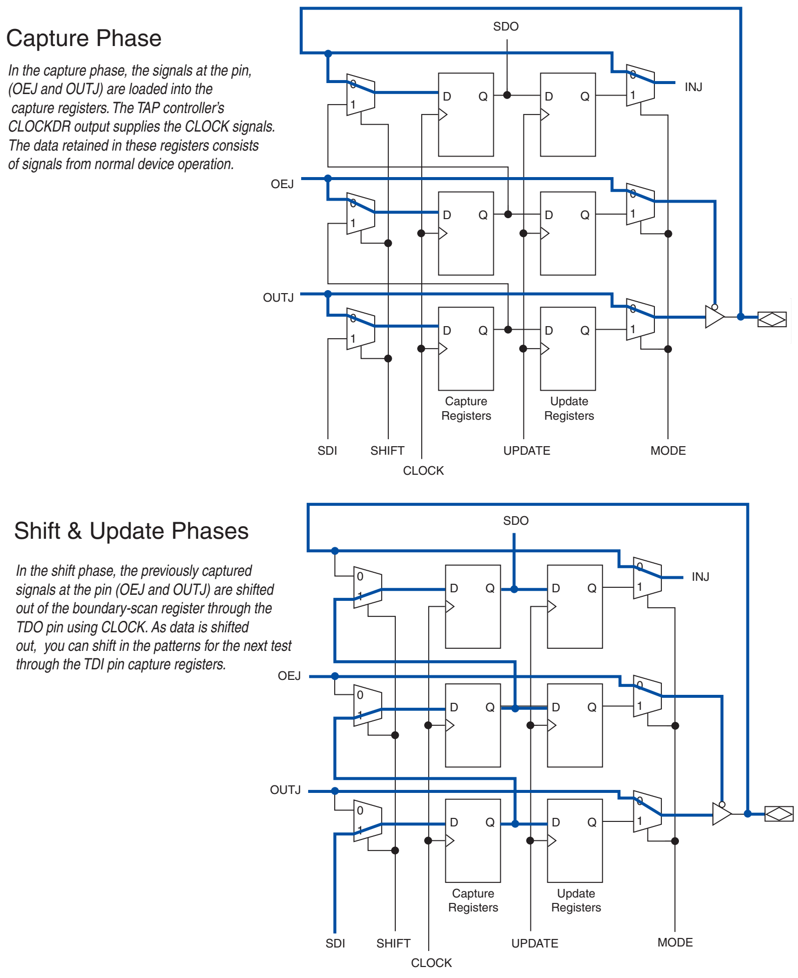
- "SAMPLE/PRELOAD Instruction Mode" on page 1–99
- "EXTEST Instruction Mode" on page 1–100
- "BYPASS Instruction Mode" on page 1–102

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 1–80 shows the capture, shift, and update phases of `SAMPLE/PRELOAD` mode.

Figure 1–80. IEEE Std. 1149.1 BST SAMPLE/PRELOAD Mode

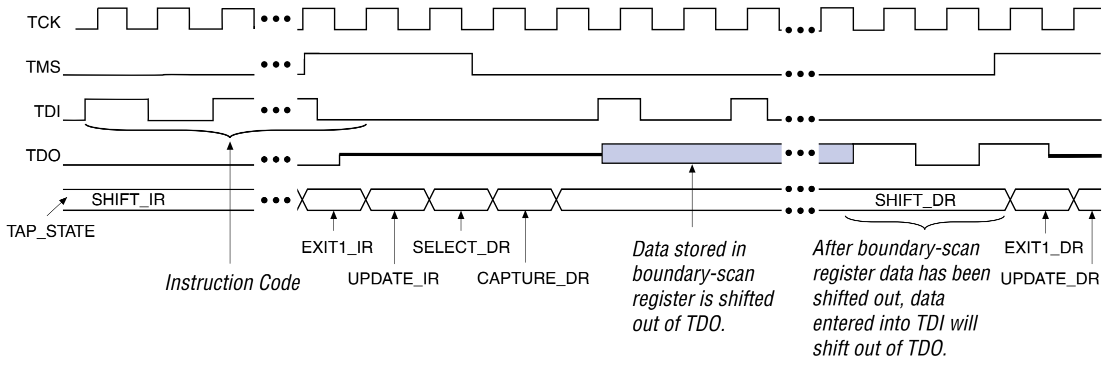

Capture Phase
In the capture phase, the signals at the pin, (OEJ and OUTJ) are loaded into the capture registers. The TAP controller's CLOCKDR output supplies the CLOCK signals. The data retained in these registers consists of signals from normal device operation.
SDO
INJ
OEJ
OUTJ
D Q
Capture Registers
Update Registers
SDI
SHIFT
CLOCK
UPDATE
MODE
Shift & Update Phases
In the shift phase, the previously captured signals at the pin (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 capture registers.
SDO
INJ
OEJ
OUTJ
D Q
Capture Registers
Update Registers
SDI
SHIFT
CLOCK
UPDATE
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 1–100.

Figure 1–81 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 1–81 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 1–81. SAMPLE/PRELOAD Shift Data Register Waveforms


TCK
TMS
TDI
TDO
SHIFT_IR
TAP_STATE
SHIFT_DR
EXIT1_IR
SELECT_DR
UPDATE_IR
CAPTURE_DR
Instruction Code
Data stored in boundary-scan register is shifted out of TDO.
After boundary-scan register data has been shifted out, data entered into TDI will shift out of TDO.
EXIT1_DR
UPDATE_DR

EXTEST Instruction Mode

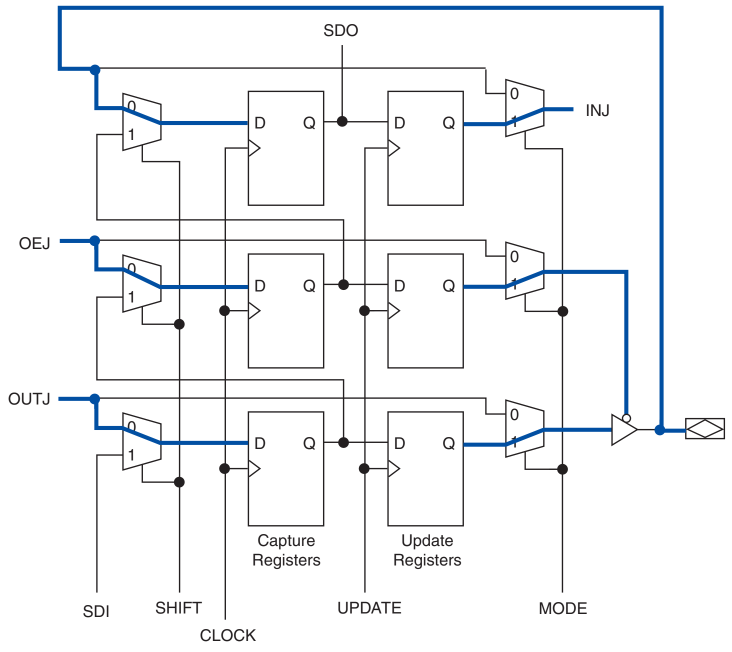
Use `EXTEST` instruction mode primarily to check external pin connections between devices. Unlike `SAMPLE/PRELOAD` mode, `EXTEST` allows test data to be forced onto the pin signals. By forcing known logic high and low levels on output pins, you can detect opens and shorts at the pins of any device in the scan chain.

Figure 1–82 shows the capture, shift, and update phases of `EXTEST` mode.

Figure 1–82. IEEE Std. 1149.1 BST EXTEST Mode

Capture Phase

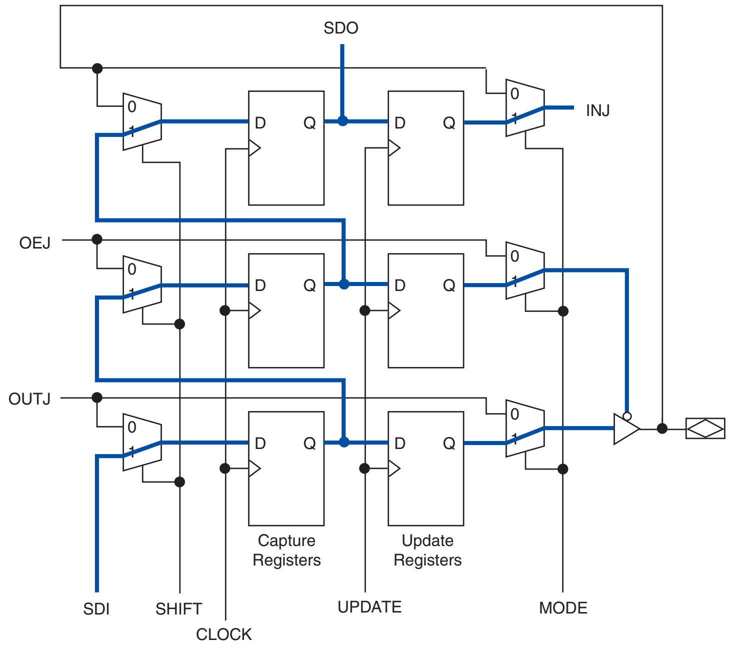
In the capture phase, the signals at the pin (OEJ and OUTJ) are loaded into the capture registers. The TAP controller's CLOCKDR output supplies the CLOCK signals. Previously retained data in the update registers drive PIN_IN and INJ, and allows the I/O pin to tri-state or drive a signal out.



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.

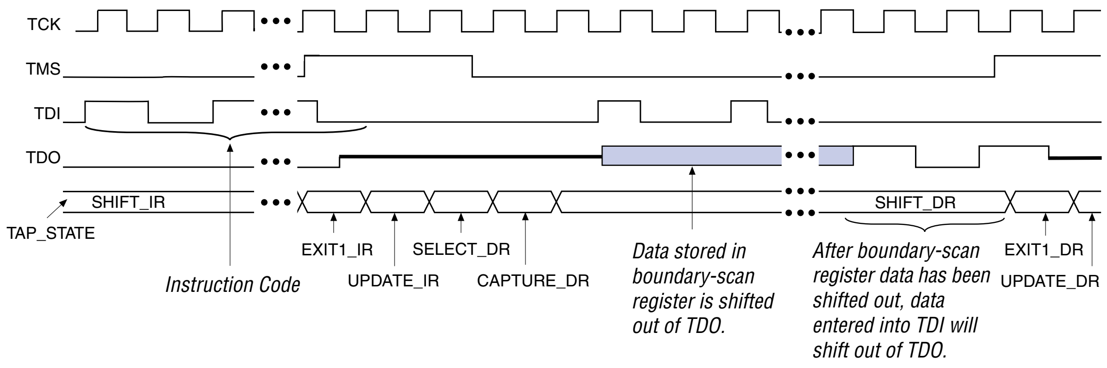
In the update phase, data is transferred from the capture registers to the update registers using the UPDATE clock. The update registers then drive PIN_IN and INJ, and allow the I/O pin to tri-state or drive a signal out.



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 1–83 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.

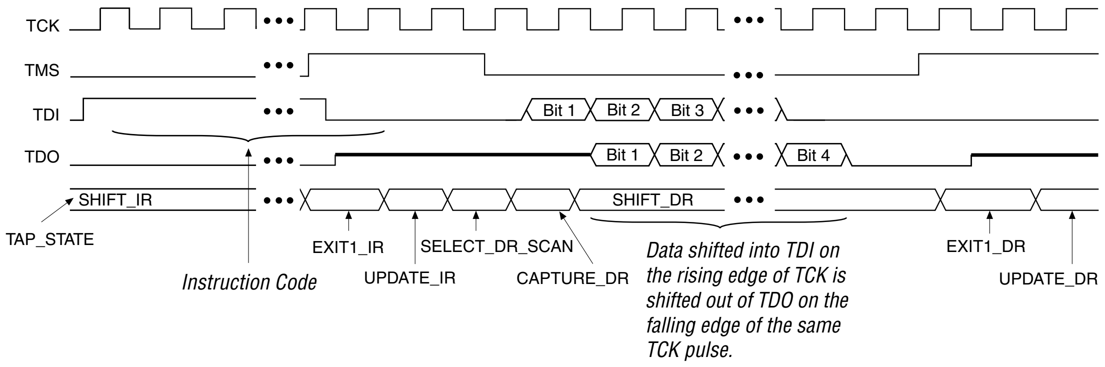
Figure 1–83. EXTEST Shift Data Register Waveforms


TCK
TMS
TDI
TDO
SHIFT_IR
TAP_STATE
Instruction Code
EXIT1_IR
UPDATE_IR
SELECT_DR
CAPTURE_DR
Data stored in boundary-scan register is shifted out of TDO.
After boundary-scan register data has been shifted out, data entered into TDI will shift out of TDO.
SHIFT_DR
EXIT1_DR
UPDATE_DR

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 1–84 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 1–84. 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 Cyclone V devices, refer to the *JTAG Boundary-Scan Testing in Cyclone 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

Cyclone 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 Cyclone V devices before, after, and during configuration. Cyclone 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 Cyclone 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 Cyclone 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 Cyclone 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 “JTAG Configuration” on page 1–62.

Disabling IEEE Std. 1149.1 BST Circuitry

The IEEE Std. 1149.1 BST circuitry for Cyclone 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 1–103.

If you are not using the IEEE Std. 1149.1 circuitry in Cyclone 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 Cyclone 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 Cyclone 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 V_{CC}, 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 Cyclone V devices.

Use the information in this section in conjunction with the *Power Management in Cyclone 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. Cyclone V devices minimize static and dynamic power using advanced process optimizations. This technology allows Cyclone V designs to meet specific performance requirements with the lowest possible power.

Hot-Socketing Specifications

The advantages of hot-socketing support in Cyclone V devices include the following:

- "Cyclone V Devices Can Be Driven Before Power Up"
- "I/O Pins Remain Tri-Stated During Power Up"
- "Insertion or Removal of the Cyclone V Device from a Powered-Up System"

Cyclone V Devices Can Be Driven Before Power Up

In Cyclone 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}, and V_{CC} power supplies through internal paths within the device. To simplify the system level design, Cyclone 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 Cyclone V output buffers are tri-stated during system power up or power down. Also, the Cyclone V device does not drive signal out until the device is configured and working within the recommended operating conditions.

Insertion or Removal of the Cyclone 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.

CAUTION This irregular power up can damage both the driving and driven devices and can disrupt card power up.

You can insert a Cyclone 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, Cyclone 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.

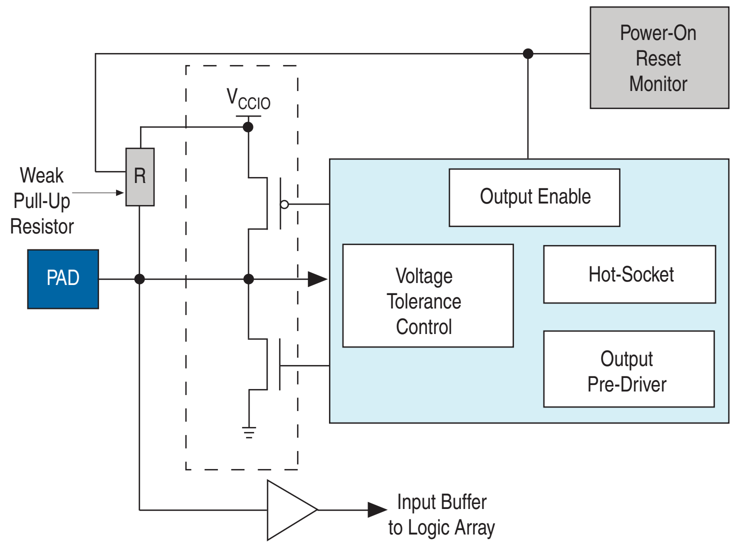
Hot-Socketing Implementation

The hot-socketing feature turns off the output buffer during power up and power down of the V_{CCIO}, V_{CCPD}, and V_{CC} 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 1–85 shows the I/O pin circuitry for Cyclone V devices.

Figure 1–85. Hot-Socketing Circuitry for Cyclone V Devices



POR circuitry monitors the voltage level of the power supplies (V_{CC}, 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 Cyclone 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_{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

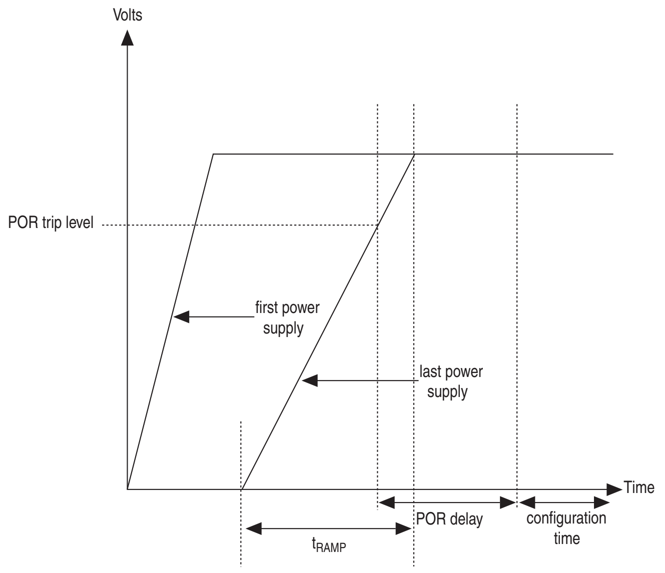
POR circuitry keeps the devices in the reset state until the power supply outputs are within operating range.

When you apply power to the Cyclone 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 Cyclone V devices is 100 ms; the minimum t_{RAMP} is 200 µs.

For more information about the fast and standard POR delay specification, refer to the *Device Datasheet for Cyclone V Devices* chapter.

Figure 1–86 shows the relationship between t_{RAMP} and POR delay.

Figure 1–86. Relationship Between t_{RAMP} and POR Delay

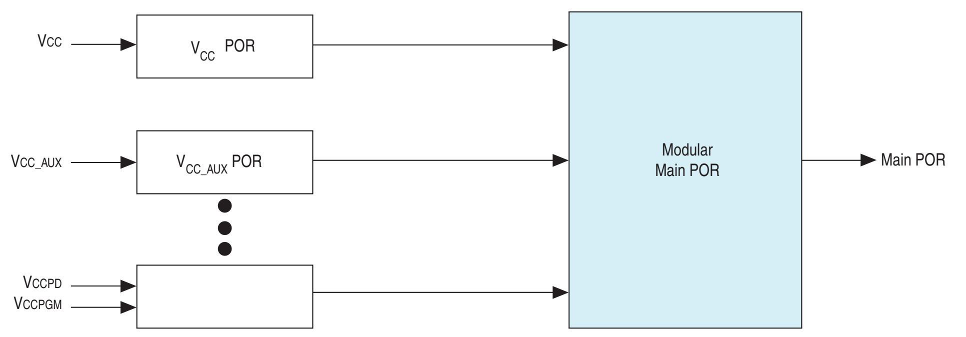


The Cyclone V POR circuitry uses an individual detecting circuit to monitor each of the configuration-related power supplies independently. The main POR circuitry is gated by the outputs of all the individual detectors. The main POR signal is asserted when the power starts to ramp up and is released 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 1–87 shows a simplified POR diagram for Cyclone V devices.

Figure 1–87. Simplified POR Diagram for Cyclone V Devices



Document Revision History

Table 1–24 lists the revision history for this document.

Table 1–24. Document Revision History

Date	Version	Changes
November 2011	1.1	■ Updated "Adaptive Logic Modules" on page 1–2, "Independent Complex Multiplier Mode" on page 1–13, and "Differential Pin Placement Guidelines" on page 1–38. ■ Updated Figure 1–35, Figure 1–38, Figure 1–44, Figure 1–45, Figure 1–48, and Figure 1–49. ■ Updated Table 1–9 and Table 1–22. ■ Minor text edits.
October 2011	1.0	Initial release.

2. Transceiver Basics for Cyclone V Devices

CV-55002-1.0

This chapter contains basic technical details pertaining to specific features in Cyclone® V device transceivers. This chapter serves as supplementary reading to *Volume 3: Transceivers of the Cyclone V Device Handbook* and covers the following topic:

- "Transceiver Architecture"

Transceiver Architecture

This section provides a detailed description of the architecture and operations of the Cyclone V transceiver channel datapaths.

Use the information in this section in conjunction with the *Transceiver Architecture in Cyclone 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 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 the swapped signal 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 serializer, which has the same effect as swapping the positive and negative signals of the serial differential link.

The 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.

Subscribe

Bit Reversal

To reverse the transmission bit order to achieve MSB-to-LSB ordering, use the bit reversal feature at the transmitter. Table 2–1 lists the bit reversal serialization factors for Cyclone V devices.

Table 2–1. Bit Reversal Serialization Factors

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]`

Transmitter Buffer

Table 2–2 lists the features provided by the Pseudo Current Mode Logic (**PCML**) output buffer to the integrated circuitry.

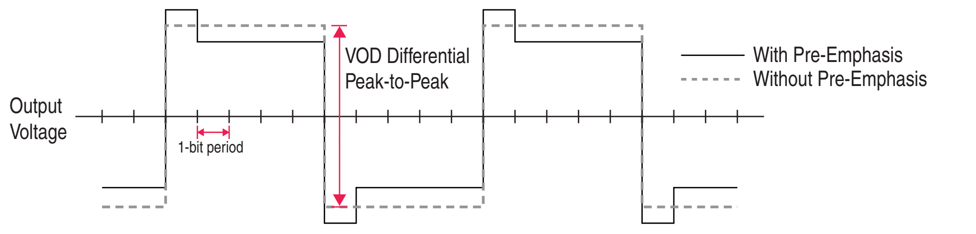
Table 2–2. Description of the Transmitter Buffer Features (Part 1 of 2)

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. Use the pre-emphasis feature to maximize 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 3.125 Gbps signal with an alternating data pattern of five 1s and five 0s.
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 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 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 you use external termination.

Table 2–2. Description of the Transmitter Buffer Features (Part 2 of 2)

Category	Features	Description
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. This 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 1.1 for Gen1 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 PCIe Link Training and Status State Machine (LTSSM) states. The circuit detects if there is a receiver downstream by changing the transmitter common-mode voltage 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 1.1 for Gen1 signaling rates. The circuit is clocked using `fixedclk` and requires an enabled transmitter OCT with the output tri-stated.

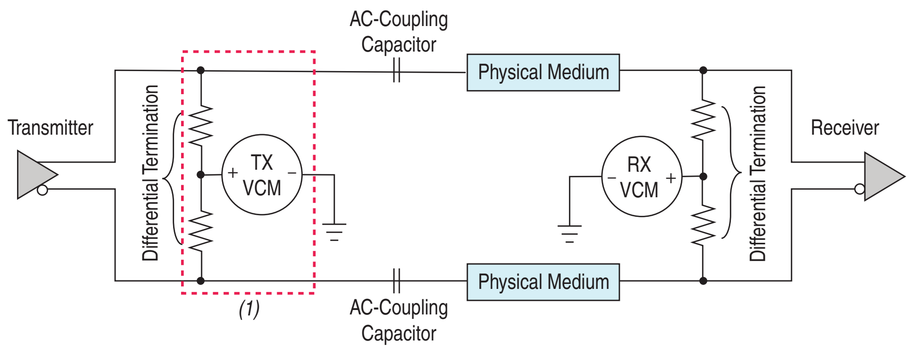
Figure 2–1. Example of the Pre-Emphasis Effect on Signal Transmission at the Transmitter Output



You can AC-couple the transmitter to a receiver. In an AC-coupled link, the AC-coupling capacitor blocks the transmitter common-mode voltage. At the receiver end, the termination and biasing circuitry restores the common-mode voltage level that is required by the receiver.

Figure 2–2 shows an AC-coupled link with a Cyclone V transmitter.

Figure 2–2. AC-Coupled Link with a Cyclone 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 PLL configured for 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.

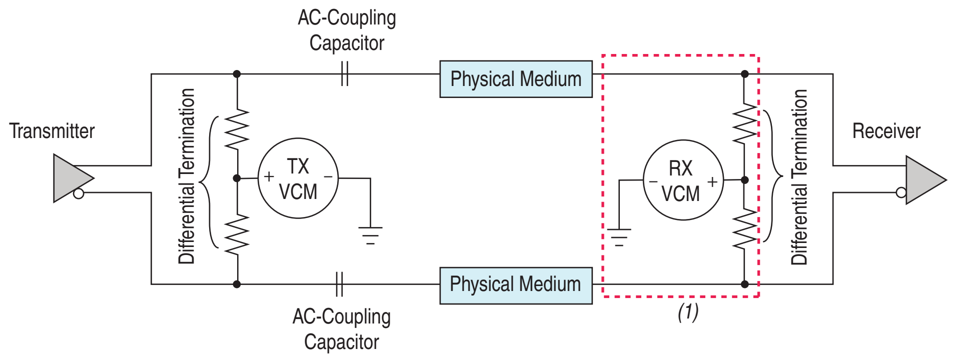
Table 2–3. Receiver Buffer Features

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 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 you use external termination.
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 1.1 for Gen1 signaling rates.

You can AC-couple the receiver to a transmitter. In an AC-coupled link, the AC-coupling capacitor blocks the transmitter common-mode voltage. At the receiver end, the termination and biasing circuitry restores the common-mode voltage level that is required by the receiver.

Figure 2–3 shows an AC-coupled link with a Cyclone V receiver.

Figure 2–3. AC-Coupled Link with a Cyclone 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.

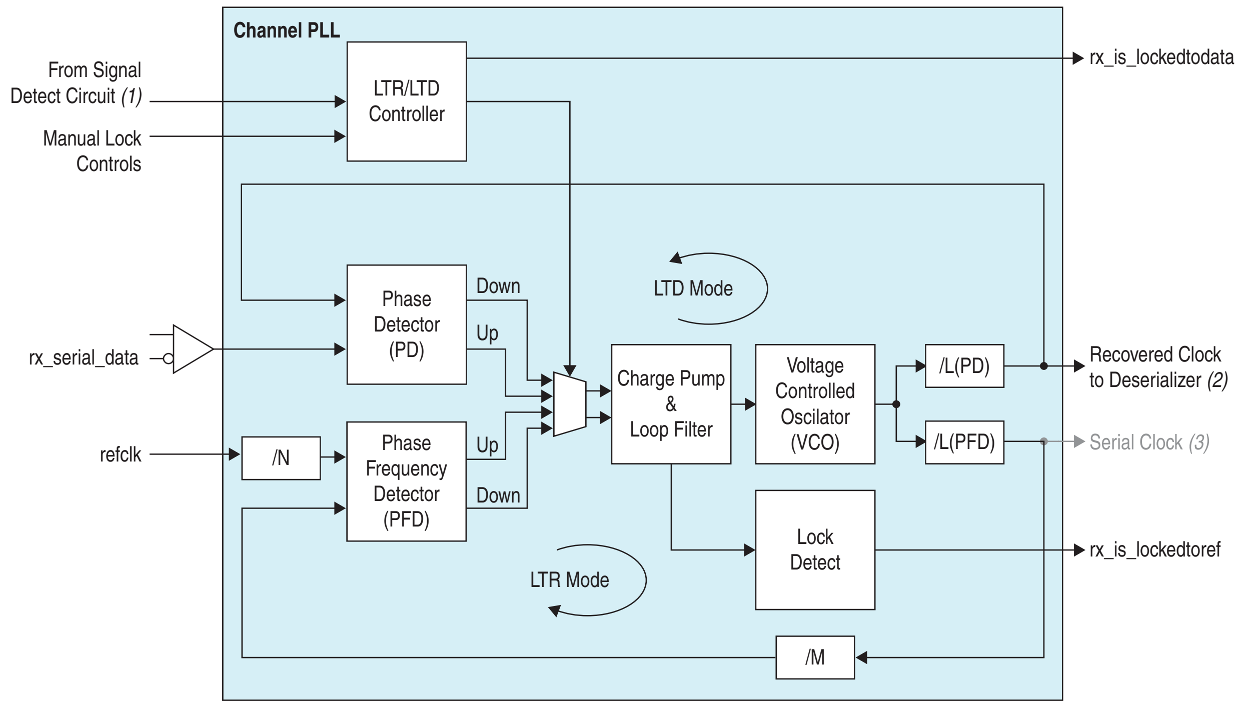
If you configure the channel PLL as a CDR PLL, the channel PLL recovers the clock and data from the serial data stream. If you do not use the channel PLL as a CDR PLL, you can configure the channel PLL as a CMU PLL for clocking the transceivers.

For more information about the channel PLL when configured as a CMU PLL, refer to the *CMU PLL* section in the *Transceiver Architecture in Cyclone V Devices* chapter.

Channel PLL Architecture

Figure 2–4 shows the major components in a channel PLL. The channel PLL supports operation in either LTR or LTD mode.

Figure 2–4. Channel PLL Block Diagram



Notes to Figure 2–4:

(1) Applicable in a PCIe® configuration only.
(2) Applicable when configured as a CDR PLL.
(3) Applicable when configured as a CMU PLL.

In LTR mode, the channel PLL tracks the input reference clock. The 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 the channel PLL is configured as a CDR PLL.

Table 2–4 lists the acceptable values for the counters 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 N factor	1, 2, 4, 8
M	Feedback loop counter to multiply the VCO frequency above the input reference frequency to the PFD by 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 LTR/LTD controller directs the CDR PLL transition between the LTR and LTD modes. The controller supports operation in automatic lock mode and manual lock mode.

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.

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 ppm difference between the receiver input reference clock and the upstream transmitter reference clock. You must hold the receiver PCS in reset until the CDR PLL locks to data and produces a stable recovered clock.

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 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 1.1.

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 1.1.

When 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 and bypassing ppm detection as required by certain applications that include, but are 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 SSC downspread of –0.5%, where the SSC modulation results in a ppm difference of up to 5,000.
- Link that requires faster CDR PLL transition to LTD, 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 Cyclone 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.

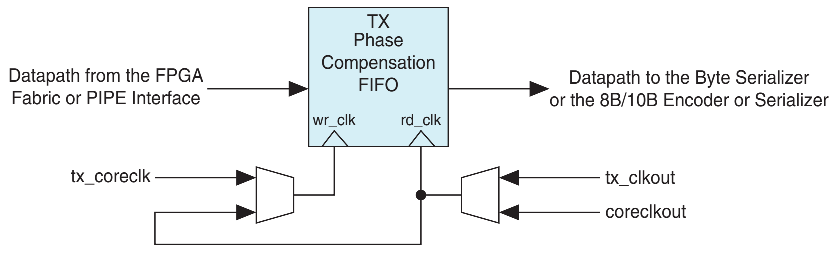
Transmitter PCS Datapath

This section describes the transmitter phase compensation FIFO, byte serializer, 8B/10B encoder, and transmitter bit-slip blocks in the transmitter PCS datapath.

Transmitter Phase Compensation FIFO

The transmitter phase compensation FIFO is four words deep and interfaces with the transmitter channel PCS and the FPGA fabric or PCIe interface. The transmitter phase compensation FIFO compensates for the phase difference between the low-speed parallel clock and the FPGA fabric interface clock. Figure 2–5 shows the datapath and clocking of the transmitter phase compensation FIFO.

Figure 2–5. Transmitter Phase Compensation FIFO


TX Phase Compensation FIFO
wr_clk
rd_clk
Datapath from the FPGA Fabric or PIPE Interface
Datapath to the Byte Serializer or the 8B/10B Encoder or Serializer
tx_coreclk
tx_clkout
coreclkout

The transmitter phase compensation FIFO supports two 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 for 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, while 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 FIFO supports various clocking modes on the read and write clocks depending on the transceiver configuration.

For more information about the transmitter datapath interface clocking modes when using the transmitter phase compensation FIFO, refer to the *Transceiver Clocking in Cyclone V Devices* chapter.

Byte Serializer

The byte serializer divides the input datapath by two to run the transceiver channel at higher data rates while keeping the FPGA fabric interface frequency within the maximum limit. In single-width mode, the byte serializer converts the two-byte wide datapath to a 1-byte wide datapath. In double-width mode, the byte serializer converts the 4-byte wide datapath to a 2-byte wide datapath. The byte serializer is optional in configurations that do not exceed the FPGA fabric-transceiver interface maximum frequency limit.

You must use the byte serializer in configurations that exceed the maximum frequency limit of the FPGA fabric–transceiver interface.

Byte Serializer in Single-Width Mode

The byte serializer forwards the LSByte first, followed by the MSByte. The input data width to the byte serializer depends on the channel width option. For example, in single-width mode with a channel width of 20 bits, the byte serializer sends out the least significant word `tx_parallel_data[9:0]` of the parallel data from the FPGA fabric, followed by `tx_parallel_data[19:10]`. Table 2–5 lists the input and output data widths of the byte serializer in single-width mode.

Table 2–5. Input and Output Data Width of the Byte Serializer in Single-Width Mode for Cyclone V Devices

Mode	Input Data Width to the Byte Serializer	Output Data Width from the Byte Serializer	Byte Serializer Output Ordering
Single-width	16	8	Least significant 8 bits of the 16-bit output first
	20	10	Least significant 10 bits of the 20-bit output first

Byte Serializer in Double-Width Mode

The operation in double-width mode is similar to that of single-width mode. For example, in double-width mode with a channel width of 32 bits, the byte serializer forwards `tx_parallel_data[15:0]` first, followed by `tx_parallel_data[31:16]`. Table 2–6 lists the input and output data widths of the byte serializer in double-width mode.

Table 2–6. Input and Output Data Width of the Byte Serializer in Double-Width Mode for Cyclone V Devices

Mode	Input Data Width to the Byte Serializer	Output Data Width from the Byte Serializer	Byte Serializer Output Ordering
Double-width	32	16	Least significant 8 bits of the 16-bit output first
	40	20	Least significant 10 bits of the 20-bit output first

If you select the **8B/10B Encoder** option, the 8B/10B encoder uses the output from the byte serializer. Otherwise, the byte serializer output is forwarded to the serializer.

8B/10B Encoder

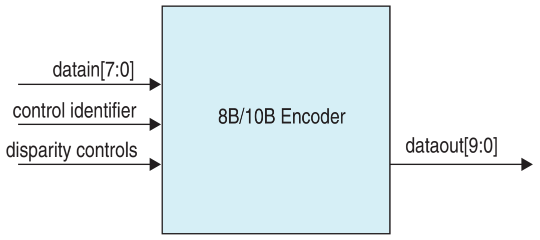
The 8B/10B encoder is available only in PCIe configurations. The encoder generates 10-bit code groups from the 8-bit data and 1-bit control identifier and supports operations in single- and double-width modes with the running disparity control feature.

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 the 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 2–6 shows a simplified diagram of the 8B/10B encoder in single-width mode.

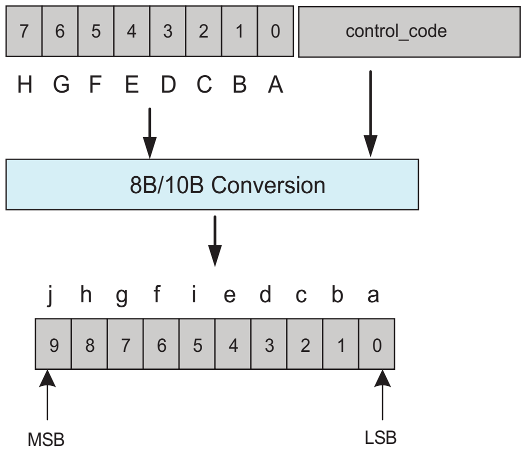
Figure 2–6. 8B/10B Encoder in Single-Width Mode



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.

In single-width mode, the 8B/10B encoder translates the 8-bit data to a 10-bit code group (control word or data word) with proper disparity. If the `tx_datak` input is high, the 8B/10B encoder translates the input `data[7:0]` to a 10-bit control word. If the `tx_datak` input is low, the 8B/10B encoder translates the input `data[7:0]` to a 10-bit data word. Figure 2–7 shows the conversion format. The LSB is transmitted first.

Figure 2–7. 8B/10B Conversion Format

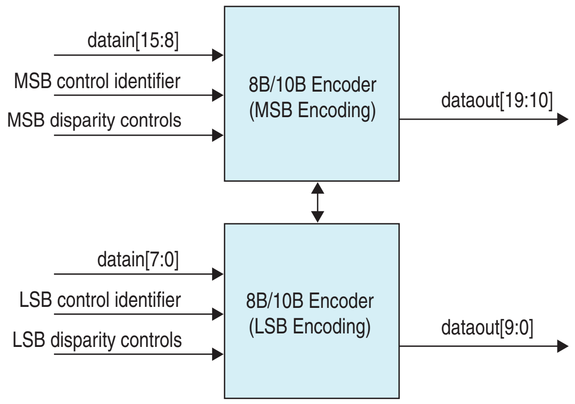


8B/10B Encoder in Double-Width Mode

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 2–8 shows a simplified diagram of the 8B/10B encoder in double-width mode.

Figure 2–8. 8B/10B Encoder in Double-Width Mode



Running Disparity Control

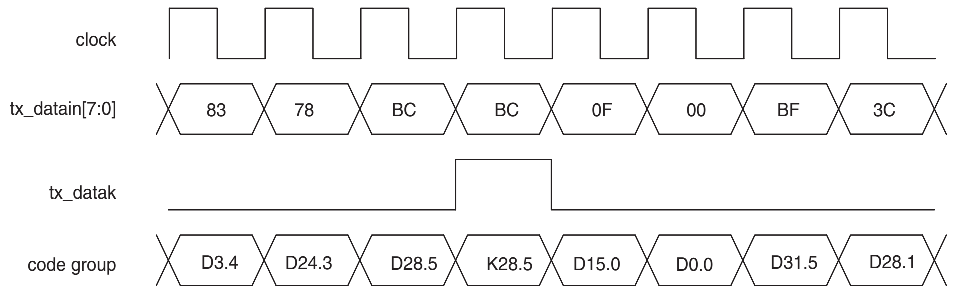
The 8B/10B encoder automatically performs calculations that meet the running disparity rules when generating the 10-bit code groups. The running disparity control feature provides user-controlled signals (`tx_dispval` and `tx_forcedisp`) to manually force encoding into a positive or negative current running disparity code group. When you enable running disparity control, the control overwrites the current running disparity value in the encoder based on the user-controlled signals, regardless of the internally-computed current running disparity in that cycle.

Using running disparity control may temporarily cause a running disparity error at the receiver.

Control Code Encoding

The 8B/10B block provides the `tx_datak` signal to indicate whether the 8-bit data at the `tx_parallel_data` signal should be encoded as a control word (Kx.y) or a data word (Dx.y). When `tx_datak` is low, the 8B/10B encoder block encodes the byte at the `tx_parallel_data` signal as data (Dx.y). When `tx_datak` is high, the 8B/10B encoder encodes the input data as a Kx.y code group. Figure 2–9 shows the second 0xBC encoded as a control word (K28.5). The rest of the `tx_parallel_data` bytes are encoded as a data word (Dx.y).

Figure 2–9. Control Word and Data Word Transmission



CAUTION The IEEE802.3 8B/10B encoder specification identifies only a set of 8-bit characters for which you must assert `tx_datak`. If you assert `tx_datak` for any other set of bytes, the 8B/10B encoder might encode the output 10-bit code as an invalid code (it does not map to a valid Dx.y or Kx.y code), or unintended valid Dx.y code, depending on the value entered. It is possible for a downstream 8B/10B decoder to decode an invalid control word into a valid Dx.y code without asserting code error flags.

Reset Condition

The `reset_tx_digital` signal resets the 8B/10B encoder. During reset, the running disparity and data registers are cleared. Also, the 8B/10B encoder outputs a K28.5 pattern from the RD– column continuously until `reset_tx_digital` is deasserted. The input data and control code from the FPGA fabric is ignored during the reset state. After reset, the 8B/10B encoder starts with a negative disparity (RD–) and transmits three K28.5 code groups for synchronization before it starts encoding and transmitting the data on its output.

While `reset_tx_digital` is asserted, the downstream 8B/10B decoder that receives the data might observe synchronization or disparity errors.

Encoder Output During Reset Sequence

Figure 2–10 shows the 8B/10B encoder output during and after reset conditions in both single- and double-width modes. When in reset (`reset_tx_digital` is high), a K28.5- (K28.5 10-bit code group from the RD– column) is sent continuously until `reset_tx_digital` is low. Because of some pipelining of the transmitter channel PCS, some "don't cares" (10'hxxx) are sent before the three synchronizing K28.5 code groups. User data follows the third K28.5 code group.

Figure 2–10. 8B/10B Encoder Output During and After Reset Conditions

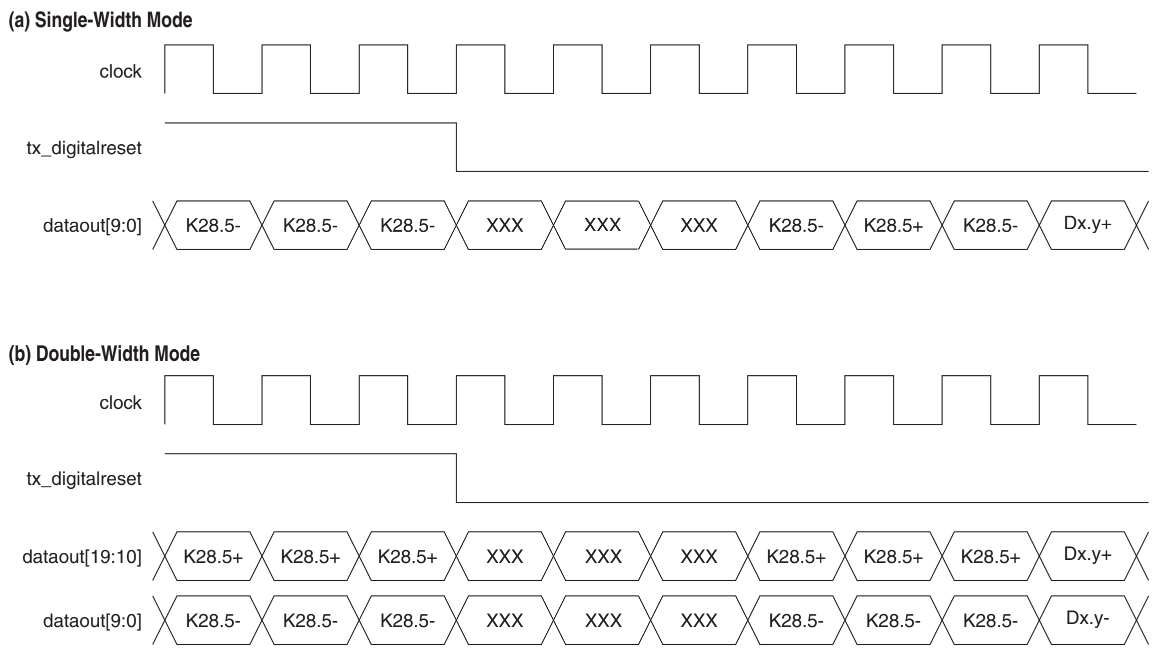

(a) Single-Width Mode
clock
tx_digitalreset
dataout[9:0]
K28.5- K28.5- K28.5- XXX XXX XXX K28.5- K28.5+ K28.5- Dx.y+
(b) Double-Width Mode
clock
tx_digitalreset
dataout[19:10]
K28.5+ K28.5+ K28.5+ XXX XXX XXX K28.5+ K28.5+ K28.5+ Dx.y+
dataout[9:0]
K28.5- K28.5- K28.5- XXX XXX XXX K28.5- K28.5- K28.5- Dx.y-

Table 2–7 lists the 8B/10B encoder output during and after reset conditions.

Table 2–7. 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" are seen 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 the LSByte and the /K28.5/ code from the RD+ column on the MSByte	Some "don't cares" are seen 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 allows you to compensate for the channel-to-channel skew between multiple transmitter channels by slipping the data sent to the PMA. The maximum number of bits slipped is controlled from the FPGA fabric and is equal to the width of the PMA-PCS minus 1. Transmitter bit-slip is supported only in 10G custom configurations and operates in single- and double-width modes. Table 2–8 lists the number of bits allowed to be slipped with the `tx_bitslipboundaryselect` signal.

Table 2–8. Bits Slip Allowed with the `tx_bitslipboundaryselect` signal

Operation Mode	Maximum Bit-Slip Setting
Single 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 datapath.

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 receives parallel data from the deserializer and restores the word boundary based on a pre-defined alignment pattern that must be received during link synchronization.

The word aligner searches for a pre-defined alignment pattern in the deserialized data to identify the correct boundary and restores the word boundary during link synchronization. The alignment pattern is pre-defined for standard serial protocols according to the respective protocol specifications for achieving synchronization. For proprietary protocol implementations, you can specify a custom word alignment pattern specific to your application.

In addition to restoring the word boundary, the word aligner implements the following features:

- Synchronization state machine
- Programmable run length violation detection (for all transceiver configurations)
- Receiver polarity inversion (for all transceiver configurations except PCIe)
- Receiver bit reversal (for custom single- and double-width configurations only)
- Receiver byte reversal (for custom double-width configuration only)

The word aligner operates in one of the following three modes:

- Manual alignment
- Automatic synchronization state machine
- Bit-slip

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.

The operation mode and alignment pattern length support varies depending on the word aligner configurations. Table 2–9 lists the available word aligner options.

Table 2–9. Word Aligner Options

PMA-PCS Interface Width	Word Alignment Mode	Word Alignment Pattern Length	Word Alignment Behavior
8 bits	Manual Alignment	16 bits	User-controlled signal starts the alignment process. Alignment happens once unless the signal is reasserted.
	Bit-Slip	16 bits	User-controlled signal shifts data one bit at a time.
10 bits	Manual Alignment	7 and 10 bits	User-controlled signal starts the alignment process. Alignment happens once unless the signal is reasserted.
	Bit-Slip	7 and 10 bits	User-controlled signal shifts data one bit at a time.
	Automatic Synchronized State Machine	7 and 10 bits	Data is required to be 8B/10B encoded. Aligns to selected word aligner pattern when pre-defined conditions are satisfied.
16 bits	Manual Alignment	8, 16, and 32 bits	Alignment happens automatically after RX PCS reset. User-controlled signal starts the alignment process thereafter. Alignment happens once unless the signal is reasserted.
	Bit-Slip	8, 16, and 32 bits	User-controlled signal shifts data one bit at a time.
20 bits	Manual Alignment	7, 10, and 20 bits	Alignment happens automatically after RX PCS reset. User-controlled signal starts the alignment process thereafter. Alignment happens once unless the signal is reasserted.
	Bit-Slip	7, 10, and 20 bits	User-controlled signal shifts data one bit at a time.
	Automatic Synchronized State Machine	7 and 10 bits	Data is required to be 8B/10B encoded. Aligns to selected word aligner pattern when pre-defined conditions are satisfied.

Word Aligner in Manual Alignment Mode

In manual alignment mode, word alignment 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 on 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 in 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.
	10 bits	1. After the `rx_digitalreset` signal deasserts, setting the `rx_enapatternalign` register to 1 triggers the word aligner to look for the predefined word alignment pattern, or its complement in the received data stream, and automatically synchronize 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 in the `rx_enapatternalign` register.
	20 bits	

Word Aligner in Bit-Slip Mode with an 8-Bit PMA-PCS Interface Configuration

You can configure the word aligner in bit-slip mode with a custom single-width configuration with an 8-bit PMA-PCS interface width. In bit-slip mode, the word aligner is controlled by the `rx_bitslip` bit of the `pcs8g_rx_wa_control` register. At every 0-1 transition of the `rx_bitslip` bit of the `pcs8g_rx_wa_control` register, the bit-slip circuitry slips one bit into the received data stream, effectively shifting the word boundary by one bit. Also in bit-slip mode, the word aligner `pcs8g_rx_wa_status` register bit for `rx_patterndetect` is driven high for one parallel clock cycle when the received data after bit-slipping matches the 16-bit word alignment pattern programmed.

To achieve word alignment, you can implement a bit-slip controller in the FPGA fabric that monitors the `rx_parallel_data` signal, the `rx_patterndetect` signal, or both, and controls them with the `rx_bitslip` signal. Table 2–11 lists the word aligner operations in bit-slip mode.

Table 2–11. Word Aligner 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 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 the moment that 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 so they are 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. Word Aligner in 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 with 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 with 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 Automatic Synchronization State Machine Mode with a 10-Bit PMA-PCS Interface Configuration

Protocols such as PCIe require the receiver PCS logic to implement a synchronization state machine to provide hysteresis during link synchronization. Each of these protocols defines a specific number of synchronization code groups that the link must receive to acquire synchronization and a specific number of erroneous code groups that it must receive to fall out of synchronization.

In PCIe configurations, the word aligner in automatic synchronization state machine mode automatically selects the word alignment pattern length and pattern as specified by each protocol. Table 2–14 lists the synchronization state machine modes. The synchronization state machine parameters are fixed for PCIe configurations as specified by the respective protocol.

Table 2–14. Word Aligner in Synchronization State Machine Modes for a PCIe Configuration

Mode	PCIe
Number of valid synchronization code groups or ordered sets received to achieve synchronization	4
Number of erroneous code groups received to lose synchronization	17
Number of continuous good code groups received to reduce the error count by one	16

After deassertion of the `reset_rx_digital` signal in automatic synchronization state machine mode, the word aligner starts looking for the word alignment pattern or synchronization code groups in the received data stream. When the programmed number of valid synchronization code groups or ordered sets is received, the `rx_syncstatus` status bit is driven high to indicate that synchronization is acquired.

The `rx_syncstatus` status bit is constantly driven high until the programmed number of erroneous code groups is received without receiving intermediate good groups; after which `rx_syncstatus` is driven low. The word aligner indicates loss of synchronization (`rx_syncstatus` remains low) until the programmed number of valid synchronization code groups are received again.

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 deterministic latency state machine controls the clock-slip process in the deserializer after the word aligner has found the alignment pattern and has 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 2–15 lists the word aligner operations in deterministic latency state machine mode.

Table 2–15. 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 in 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 clock-slip is complete, the deserialized data coming into the receiver PCS is word-aligned and indicated when the `rx_syncstatus` signal asserts.
Double Width	20 bits	

Programmable Run-Length Violation Detection

The programmable run-length violation detection circuit resides in the word aligner block and detects if consecutive 1s or 0s in the received data exceed the user-specified threshold. If the data stream exceeds the preset maximum number of consecutive 1s or 0s, the violation is signified by the assertion of the `rx_rlv status` bit. Table 2–16 lists the detection capabilities of the circuit.

The programmable run length violation circuit resides in the word aligner block and detects consecutive 1s or 0s in the data. If the data stream exceeds the preset maximum number of consecutive 1s or 0s, the violation is signified by the assertion of the `rx_rlv` signal.

For more information about the programmable run length violation circuit, refer to the *Altera Transceiver PHY IP Core User Guide*.

For more information about the `rx_rlv` signal, refer to the *Transceiver Architecture in Stratix IV Devices* chapter in the *Stratix IV Device Handbook*.

Table 2–16. 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 erroneously be swapped during board layout. Solutions such as board re-spin or major updates to the PLD logic can be expensive. The polarity inversion feature at the receiver corrects the swapped signal 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. When you enable the polarity inversion feature, initial disparity errors may occur at the receiver with the 8B/10B-coded data. The receiver must be able to tolerate these disparity errors.

CAUTION If you enable polarity inversion midway through a word, the word will be corrupted.

Bit Reversal

By default, the receiver assumes a LSB-to-MSB transmission. If the transmission order is MSB-to-LSB, the receiver forwards the bit-flipped version of the parallel data to the FPGA fabric on `rx_parallel_data`. To reverse the bit order at the output of the word aligner to receive a MSB-to-LSB transmission, use the bit reversal feature at the receiver. Table 2–17 lists the bit reversal feature options.

Table 2–17. 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 to use 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, 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 order.

Receiver Byte Reversal

In double-width mode, two symbols of incoming data at the receiver 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 corrects the swapped signal error by swapping the two symbols in double-width mode at the word aligner output, as listed in Table 2–18.

Table 2–18. 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]`

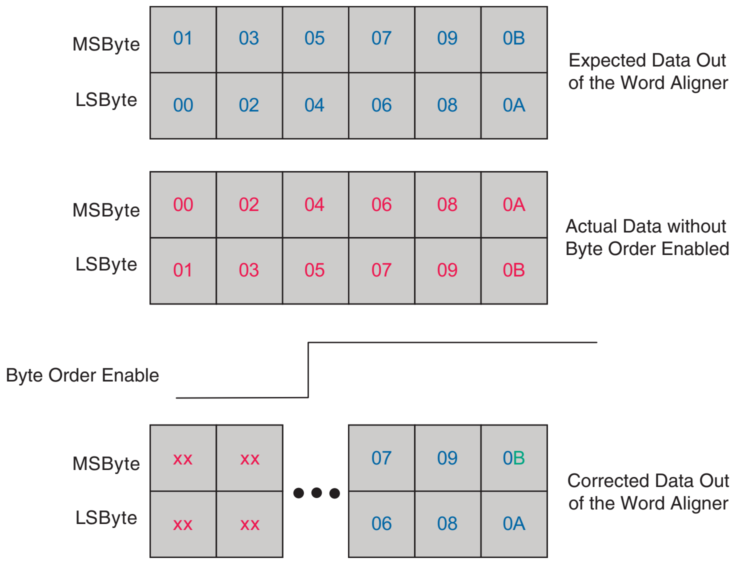
The reversal is controlled dynamically using the `rx_bytereversal_enable` register, and when you enable the receiver byte reversal option, this may cause initial disparity errors at the receiver with 8B/10B-coded data. The receiver must be able to tolerate these disparity errors.

When receiving swapped symbols, the word alignment pattern must be byte-reversed accordingly to match the incoming byte-reversed data.

Receiver Byte Reversal in Custom Double-Width Configurations

The MSByte and LSByte of the input data to the transmitter may be erroneously swapped. The receiver byte reversal feature is available to correct this situation. Figure 2–11 shows the receiver byte reversal feature.

Figure 2–11. Receiver Byte Reversal Feature in Cyclone V Devices



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.

The deskew FIFO module is available only when you use the XAUI protocol. Use deskew FIFO to align four channels to meet the maximum skew requirement of 40 UI (12.8 ns) as seen at the receiver of the four lanes. The deskew operation is compliant with the PCS deskew state machine diagram specified in clause 48 of the IEEE P802.3ae specification.

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 requires that the transceiver channel is in duplex configuration (both transmit and receive functions) and has a predefined 20-bit pattern (that consists 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 rate match 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 rate match 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-width mode. The 20-bit pattern can be user-defined for custom configurations. For protocol configurations, the rate match 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 1.1 for Gen1 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

For more information about the rate match FIFO in custom and protocol-specific configurations, refer to the *Transceiver Custom Configurations in Cyclone V Devices* and the *Transceiver Protocol Configurations in Cyclone V Devices* chapters, respectively.

In asynchronous systems, use independent reference clocks to clock the upstream transmitter and local receiver. Frequency differences in the order of a few hundred ppm can corrupt the data when latching from the recovered clock domain (the same clock domain as the upstream transmitter reference clock) to the local receiver reference clock domain.

The rate match FIFO deletes `SKP` symbols or ordered sets when the upstream transmitter reference clock frequency is higher than the local receiver reference clock frequency and inserts `SKP` symbols or ordered sets when the local receiver reference clock frequency is higher than the upstream transmitter reference clock frequency.

8B/10B Decoder

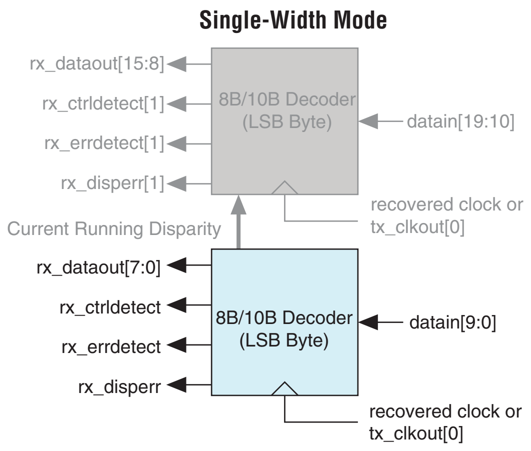
The receiver channel PCS datapath implements the 8B/10B decoder after the rate match FIFO. In configurations with the rate match FIFO enabled, the 8B/10B decoder receives data from the rate match FIFO. In configurations with the rate match FIFO disabled, the 8B/10B decoder receives data from the word aligner. The 8B/10B decoder supports operation in single- and double-width modes.

PCIe mode requires the serial data sent over the link to be 8B/10B encoded to maintain the DC balance in the transmitted serial data. This protocol requires the receiver PCS logic to implement an 8B/10B decoder to decode the data before forwarding it to the upper layers for packet processing.

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. The decoded data is fed to the byte deserializer or the receiver phase compensation FIFO (if the byte deserializer is disabled). Figure 2–12 shows the 8B/10B decoder in single-width mode.

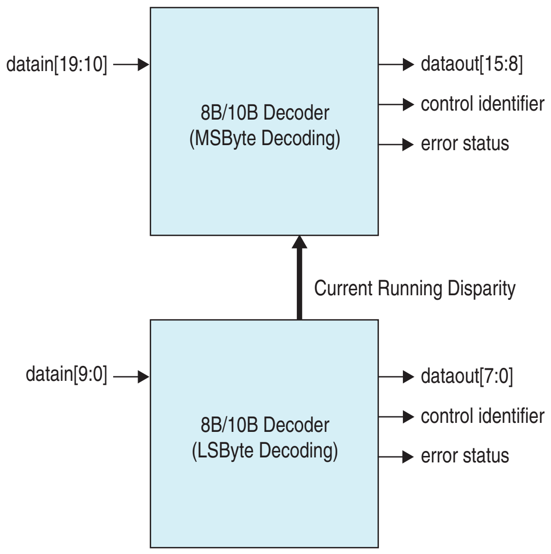
Figure 2–12. 8B/10B Decoder in Single-Width Mode



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 2–13 shows a simplified 8B/10B decoder block diagram in double-width mode.

Figure 2–13. 8B/10B Decoder in Double-Width Mode



Control Code Group Detection

The 8B/10B decoder indicates whether the decoded 8-bit code group is a data or a control code group on the `rx_datak` signal. If the received 10-bit code group is one of the 12 control code groups (/Kx.y/) specified in the IEEE802.3 specification, the `rx_datak` signal is driven high. If the received 10-bit code group is a data code group (/Dx.y/), the `rx_datak` signal is driven low.

Byte Deserializer

The FPGA fabric-transceiver interface frequency has an upper limit. In configurations that have a receiver PCS frequency greater than the upper limit stated, the parallel received data and status signals cannot be forwarded directly to the FPGA fabric because it violates this upper limit for the FPGA fabric-transceiver interface frequency. In such configurations, the byte deserializer is required to reduce the FPGA fabric-transceiver interface frequency to half while doubling the parallel data width.

☞ The byte deserializer is required in configurations that exceed the FPGA fabric-transceiver interface clock upper frequency limit. It is optional in configurations that do not exceed the FPGA fabric-transceiver interface clock upper frequency limit.

The 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. 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 2–19 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.

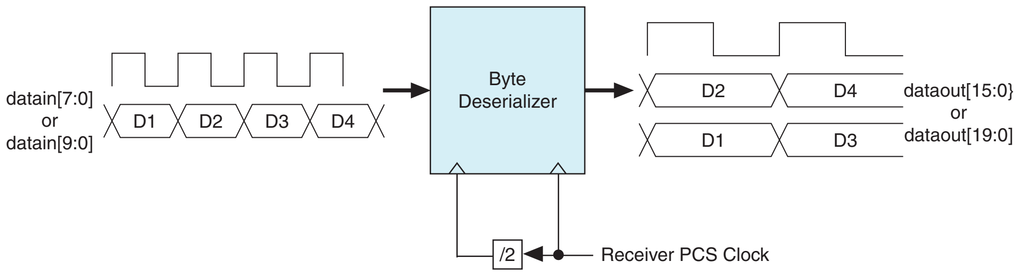
Table 2–19. Byte Deserializer Input Datapath Width Conversion

Mode	Byte Deserializer Input Datapath Width	Receiver Output Datapath Width
Single Width	8	16
	10	20
Double Width	16	32
	20	40

Byte Deserializer in Single-Width Mode

In single-width mode, the byte deserializer receives 8-bit wide data from the 8B/10B decoder or 10-bit wide data from the word aligner (if the 8B/10B decoder is disabled) and deserializes it into 16- or 20-bit wide data at half the speed. Figure 2–14 shows the byte deserializer in single-width mode.

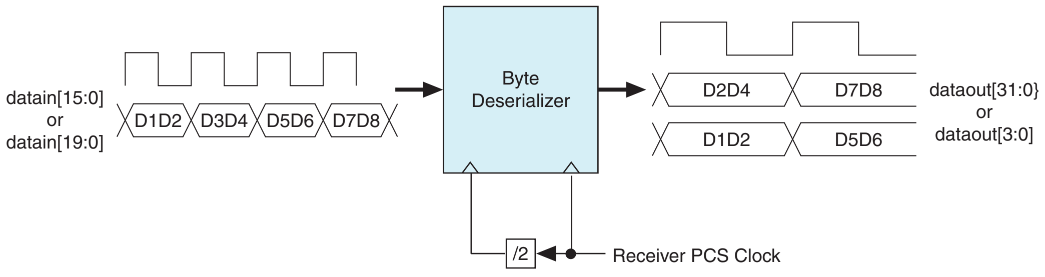
Figure 2–14. Byte Deserializer in Single-Width Mode



Byte Deserializer in Double-Width Mode

In double-width mode, the byte deserializer receives 16-bit wide data from the 8B/10B decoder or 20-bit wide data from the word aligner (if the 8B/10B decoder is disabled) and deserializes it into 32- or 40-bit wide data at half the speed. Figure 2–15 shows the byte deserializer in double-width mode.

Figure 2–15. Byte Deserializer in Double-Width Mode



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 restores 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:

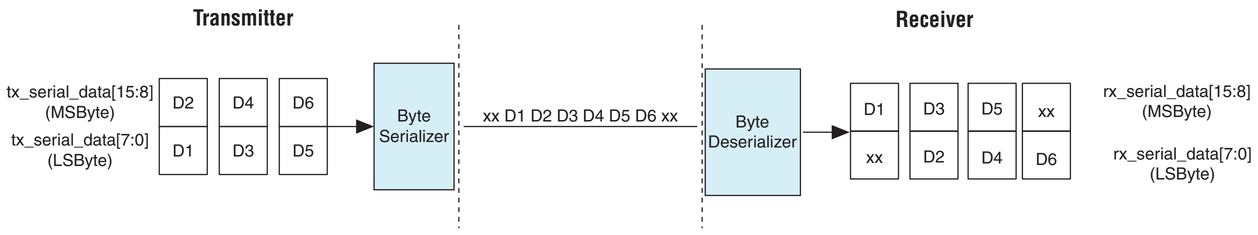
- A receiver with the byte deserializer enabled
- A predefined byte ordering pattern that must be ordered at the LSByte position of the data
- A predefined pad pattern

Byte Ordering in Single-Width Mode

In single-width mode with a 16- or 20-bit FPGA fabric-transceiver interface, the byte deserializer receives one data byte (8 or 10 bits) and deserializes it into two data bytes (16 or 20 bits). Depending on when the receiver PCS logic comes out of reset, the byte ordering at the output of the byte deserializer may or may not match the original byte ordering of the transmitted data. The byte misalignment resulting from byte deserialization is unpredictable because it depends on which byte is being received by the byte deserializer when it comes out of reset.

Figure 2–16 shows a scenario in which the MSByte and LSByte of the two-byte transmitter data appears straddled across two word boundaries after being byte deserialized at the receiver.

Figure 2–16. MSByte and LSByte of the Two-Bit Transmitter Data Straddled Across Two Word Boundaries

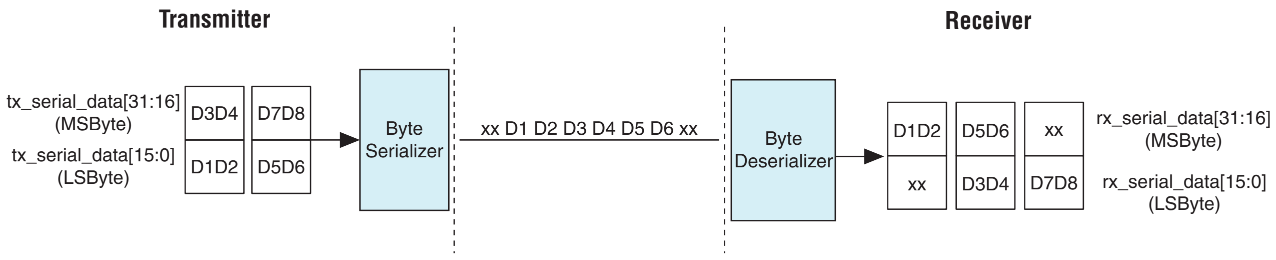

Transmitter
Receiver
tx_serial_data[15:8] (MSByte)
tx_serial_data[7:0] (LSByte)
D2
D4
D6
D1
D3
D5
Byte Serializer
xx D1 D2 D3 D4 D5 D6 xx
Byte Deserializer
D1
D3
D5
xx
xx
D2
D4
D6
rx_serial_data[15:8] (MSByte)
rx_serial_data[7:0] (LSByte)

Byte Ordering in Double-Width Mode

In double-width mode with a 32-bit FPGA fabric-transceiver interface, the byte deserializer receives two data bytes (16 bits) and deserializes it into four data bytes (32 bits).

Figure 2–17 shows a scenario in which the two MSBytes and LSBytes of the four-byte transmitter data appears straddled across two word boundaries after being byte deserialized at the receiver.

Figure 2–17. MSByte and LSByte of the Four-Bit Transmitter Data Straddled Across Two Word boundaries


Transmitter
Receiver
tx_serial_data[31:16] (MSByte)
tx_serial_data[15:0] (LSByte)
D3D4
D7D8
D1D2
D5D6
Byte Serializer
xx D1 D2 D3 D4 D5 D6 xx
Byte Deserializer
D1D2
D5D6
xx
xx
D3D4
D7D8
rx_serial_data[31:16] (MSByte)
rx_serial_data[15:0] (LSByte)

The transceivers have an optional byte ordering block in the receiver datapath that restores proper byte ordering before forwarding the data to the FPGA fabric. The byte ordering block looks for the user-programmed byte ordering pattern in the byte-deserialized data. You must select a byte ordering pattern that you know appears at the LSBytes position of the parallel transmitter data. If the byte ordering block finds the programmed byte ordering pattern in the MSBytes position of the byte-deserialized data, the byte ordering block inserts the appropriate number of user-programmed PAD bytes to push the byte ordering pattern to the LSBytes position, thereby restoring proper byte ordering.

Byte Ordering in a Custom Single-Width Mode

In a custom single-width configuration, you can program a custom byte ordering pattern and byte ordering PAD pattern. Table 2–20 lists the byte ordering pattern length allowed in a custom single-width configuration.

Table 2–20. Byte Ordering Pattern Length in a Custom Single-Width Configuration for Cyclone V Devices

Configuration	Byte Ordering Pattern Length	Byte Ordering PAD Pattern Length
Custom single-width configuration with: ■ 16-bit FPGA fabric-transceiver interface ■ No 8B/10B decoder ■ Word aligner in manual alignment mode	8 bits	8 bits

Byte Ordering Block in a Custom Double-Width Mode

In a custom double-width configuration, you can program a custom byte ordering pattern and byte ordering PAD pattern in the ALT PHY IP megafunction. Table 2–21 lists the byte ordering pattern length allowed in a custom double-width configuration.

Table 2–21. Byte Ordering Pattern Length in a Custom Double-Width Configuration for Cyclone V Devices

Configuration	Byte Ordering Pattern Length	Byte Ordering PAD Pattern Length
Custom double-width configuration with: ■ 32-bit FPGA fabric-transceiver interface ■ No 8B/10B decoder (16-bit PMA-PCS interface) ■ Word aligner in manual alignment mode	16 bits, 8 bits	8 bits

Receiver Phase Compensation FIFO

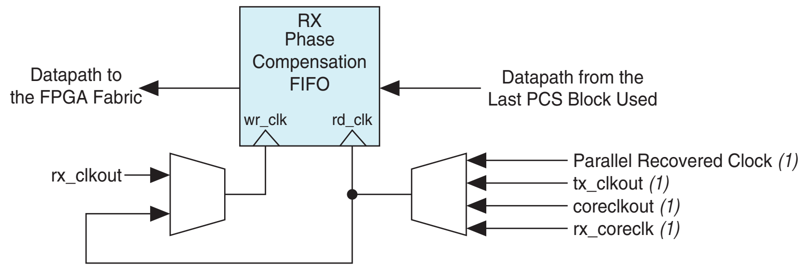
The receiver phase compensation FIFO is four words deep and interfaces the status and data signals between the receiver PCS clock (FIFO write clock) and the FPGA fabric clock (FIFO read clock) or the PCIe hard IP block. The low-speed parallel clock feeds the write clock, while the FPGA fabric interface clock feeds the read clock. The clocks must have 0 ppm difference in frequency or a receiver phase compensation FIFO underrun or overflow condition may result.

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

Figure 2–18 shows the receiver phase compensation FIFO.

Figure 2–18. Receiver Phase Compensation FIFO


RX Phase Compensation FIFO
wr_clk
rd_clk
Datapath to the FPGA Fabric
Datapath from the Last PCS Block Used
rx_clkout
Parallel Recovered Clock (1)
tx_clkout (1)
coreclkout (1)
rx_coreclk (1)

Note to Figure 2–18:

(1) These clocks may have been divided by 2 if you used a byte deserializer.

For a detailed description of the receiver datapath interface clocking modes when you use the receiver phase compensation FIFO, refer to the *Transceiver Clocking in Cyclone V Devices* chapter.

Document Revision History

Table 2–22 lists the revision history for this chapter.

Table 2–22. Document Revision History

Date	Version	Changes
October 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.
?	The question mark directs you to a software help system with related information.
👣	The feet direct you to another document or website with related information.
🎥	The multimedia icon directs you to a related multimedia presentation.
⚠ CAUTION	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
⚠ WARNING	A warning calls attention to a condition or possible situation that can cause you injury.
✉	The 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.