Virtex-5 FPGA User Guide

UG190 (v4.5) January 9, 2009

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/14/06	1.0	Initial Xilinx release.
05/12/06	1.1	Minor typographical edits and clarifications. Chapter 1: Revised Figure 1-21. Chapter 2: Revised Figure 2-2 and Figure 2-4. Removed reference to a DCM_PS primitive. Removed outdated clocking wizard section page 79. Chapter 3: Revised Figure 3-1, Figure 3-2, Table 3-2, Table 3-4, Figure 3-9, Equation 3-8, and Figure 3-12. Added "PLL in Virtex-4 FPGA PMCD Legacy Mode" section. Chapter 4: Added a note to Table 4-5, page 122. Clarified the RAMB36 port mapping design rules on page 130. Chapter 5: Added Figure 5-7 and Figure 5-11, revised Figure 5-32 for clarity. Chapter 6: Updated "Simultaneous Switching Output Limits" section. Chapter 7: Revised "ILOGIC Resources," page 316 including Figure 7-1. Revised Table 7-3. Chapter 8: Revised Table 8-1.
7/19/06	1.2	Chapter 1: Revised "Global Clock Buffers," page 23 to clarify single-ended clock pins. Changed the P and N I/O designations in Figure 1-19. Chapter 4: Added "Block RAM SSR in Register Mode," page 131 and "FIFO Architecture: a Top-Level View," page 141. Revised the FIFO operations "Reset," page 143 description. Chapter 6: Minor clarification edits. Changed to N/A from unused in Table 6-36, Table 6-37, and Table 6-38. Chapter 7: Minor edits to clarify IODELAY in this chapter. Chapter 8: Small clarifications in "ISERDES_NODELAY Ports" on page 353.
9/06/06	2.0	Added the LXT platform devices throughout document. Chapter 1: Revised Figure 1-22, page 41. Updated "Clock Capable I/O" on page 36. Chapter 2: Updated "Output Clocks" on page 61. Chapter 4: Clarified the rules regarding FULL and EMPTY flags on page 138. Chapter 5: Revised "Storage Elements" on page 176. Chapter 6: "Differential Termination Attribute" on page 235 is updated for the latest syntax and settings. Replaced the link to the SSO calculator.
10/12/06	2.1	Added System Monitor User Guide reference in the Preface. Added XC5VLX85T to Table 1-5, Table 2-1, and Table 5-2. Chapter 3: Revised Figure 3-1. Chapter 4: Added cascade to Table 4-7, page 124. Revised ADDR in Figure 4-9, page 122. Removed scrub mode in "Built-in Error Correction" section. Chapter 5: Revised Figure 5-22, page 195.
02/02/07	3.0	Added the three SXT devices and the XC5VLX220T to Table 1-5, Table 2-1, and Table 5-2. Chapter 4: Clarified wording in "Synchronous Clocking" on page 117. Chapter 6: Added "DCI Cascading" on page 218. Changed V_{REF} for SSTL18_II_T_DCI to 0.9 in Table 6-39. Chapter 7: Revised OQ in Figure 7-27, page 347. Chapter 8: "Clock Enable Inputs - CE1 and CE2" on page 354.

Date	Version	Revision
09/11/07	3.1	Chapter 1: Added "Clock Gating for Power Savings" on page 22. Revised Figure 1-2, page 26. Revised Figure 1-16, page 33. Chapter 2: Revised DCM reset and locking process in "Reset Input - RST," page 49. Updated DO[2] description in Table 2-4, page 52. Changed the multiply value range on page 54. Revised the description for "FACTORY_JF Attribute," page 57. Revised "Output Clocks," page 61, updated Figure 2-7, page 70, and added a BUFG to Figure 2-10, page 72. Added more steps to Dynamic Reconfiguration (DRPs) when loading new M and D values on page 69. Updated Figure 2-7, page 70. Revised bulleted descriptions under Figure 2-20, page 83. Chapter 3: Updated Figure 3-1, page 86. Add notes to Table 3-2, page 90. Added a note to "Phase Shift," page 92. Added rounding to Equation 3-3 through Equation 3-6. Revised CLKFBIN, CLKFBDCM, CLKFBOUT, RST, LOCKED, and added the REL pin and note 2 to Table 3-3, page 93. Added RESET_ON_LOSS_OF_LOCK attribute to Table 3-4, page 95. Removed general routing discussion from "PLL Clock Input Signals."Revised "Missing Input Clock or Feedback Clock" section. Added waveforms to Figure 3-13. Corrected the Virtex-4 port mapping in Figure 3-17 and Table 3-8, page 108. Chapter 4: Revised and clarified "Built-in Error Correction." Edited WE signal throughout. Clarified Readback limitation in "Simple Dual-Port Block RAM" on page 119. Edited "Set/Reset - SSR[A | B]," page 123. Added "Block RAM Retargeting," page 138. Revised latency values and added Note 1 to Table 4-16, page 144. Updated "Cascading FIFOs to Increase Depth," page 156. Chapter 5: Clarified information about common control signals in a slice in "Storage Elements" on page 176. Chapter 6: Updated the DCI cascading guidelines on page 221. Removed references to "HSLVDCI Controlled Impedance Driver with Unidirectional Termination" since it is not supported in software. Added note 3 to Table 6-17, page 254. Clarified the introduction to "SSTL (Stub-Series Terminated Logic)," page 272. Revised "DIFF_SSTL2_II_DCI, DIFF_SSTL18_II_DCI" on page 273. Fixed DIFF_SSTL2_II references in Figure 6-74, page 280. Revised rules 2 and 3 in "Rules for Combining I/O Standards in the Same Bank," page 296. Deleted of absolute maximum table from "Overshoot/Undershoot," page 300. Chapter 7: Removed DDLY port from IDDR primitive page 319. Added the SIGNAL _PATTERN, DELAY_SRC, and REFCLK_FREQUENCY attributes to Table 7-10, page 327. Revised Figure 7-9, page 328. Removed Table 7-12: "Generating Reference Clock From DCM" and updated REFCLK section in "IDELAYCTRL Ports" on page 336. Clarified introduction in "IDELAYCTRL Locations," page 337. Changed ODDR "Clock Forwarding," page 345. Chapter 8: Updated SR and O in Figure 8-2 and Table 8-1, page 353. Updated the entire section for "BITSLIP Submodule," page 364. Fixed typographical errors in Figure 8-14, page 368.
12/11/07	3.2	Chapter 1: Revised description in "Clock Gating for Power Savings," page 22. Added the XC5VLX20T, XC5VLX155, and XC5VLX155T devices to Table 1-5. Chapter 2: Added the XC5VLX20T, XC5VLX155, and XC5VLX155T devices to Table 2-1. Chapter 3: Revised "Clock Network Deskew," page 90. Removed note 2 and revised descriptions of CLKFBOUT and DEN in Table 3-3, page 93. Revised allowed value of CLKOUT[0:5]_PHASE and CLKFBOUT_MULT description in Table 3-4, page 95. Revised Figure 3-13 and Figure 3-14 including waveforms. Chapter 5: Added the XC5VLX20T, XC5VLX155, and XC5VLX155T devices to Table 5-2. Chapter 6: Clarified discussion of cascading across CMT tiles in "DCI Cascading." Changed the split termination to V_{TT} = 0.9V in Figure 6-84, page 290. Chapter 7: Added to the descriptions of the "HIGH_PERFORMANCE_MODE Attribute," and the "SIGNAL_PATTERN Attribute," page 328 including Table 7-10. Revised description in "Instantiating IDELAYCTRL Without LOC Constraints," page 338. Chapter 8: Complete rewrite of the chapter. Many changes to descriptions, tables, and figures.

Date	Version	Revision
02/05/08	3.3	Chapter 1: Updated discussion under "I/O Clock Buffer - BUFIO" on page 37. Chapter 3: Revised LOCKED description in Table 3-3, page 93. Revised discussion under "Detailed VCO and Output Counter Waveforms," page 100. Chapter 5: Updated description of Figure 5-17. Chapter 7: Updated description under "Clock Input - C" on page 325. Updated default value to TRUE for HIGH_PERFORMANCE_MODE in Table 7-10, page 327. Chapter 8: Revised TRISTATE_WIDTH in Table 8-7, page 372. Updated discussion under "TRISTATE_WIDTH Attribute" and added section on "OSERDES Clocking Methods," page 373.
03/31/08	4.0	Added the FXT platform to Table 1-5, Table 2-1, and Table 5-2. Revised timing event description under Figure 1-21, page 40. Revised "Dynamic Reconfiguration," page 69 to remove adjustment of PHASE_SHIFT. Added CLKOUT[0:5]_DESKEW_ADJUST to Table 3-4, page 95. Corrected READ_WIDTH_B = 9 to WRITE_WIDTH_B = 9 in the block RAM usage rules on page 112. Revised "High-Speed Clock for Strobe-Based Memory Interfaces - OCLK," page 355. Corrected BITSLIP_ENABLE value from string to boolean in "ISERDES_NODELAY Attributes," page 356.
04/25/08	4.1	Added the XC5VSX240T to Table 1-5, Table 2-1, and Table 5-2. Revised Figure 1-21, page 40. Removed a pad notation from the ODDR output of Figure 2-9. Removed the BUFG on the output of Figure 2-10. Updated CLKOUT[0:5]_DESKEW_ADJUST description in Table 3-4, page 95. Revised equations Equation 3-5 and Equation 3-6. Updated the notes in Table 4-16, page 144. Revised description of "Instantiating IDELAYCTRL with Location (LOC) Constraints," page 340.
05/09/08	4.2	Revised clock routing resources in "BUFGCTRL to DCM," page 69. Removed example Figure 2-10 on page 72. Corrected note 1 in Table 4-5, page 122. Added "Legal Block RAM and FIFO Combinations," page 170. Clarified Note 7 in DCI in Virtex-5 Device I/O Standards. Master DCI is not supported in Banks 1 and 2.
09/23/08	4.3	Added the TXT platform to Table 1-5, Table 2-1, and Table 5-2. Chapter 2: Revised "Reset Input - RST" on page 49 and "System-Synchronous Setting (Default)," page 62. Chapter 3: Updated "Jitter Filter," page 91. Chapter 4: Updated "Write Modes" on page 115 and "Asynchronous Clocking" on page 117. Chapter 6: Labeled all the DCI_18 standards consistently in Table 6-39 and Table 6-40. Replaced the link to the "Full Device SSO Calculator." Chapter 8: Updated CLKB in Table 8-1, page 353 and "High-Speed Clock Input - CLKB," page 355.
12/02/08	4.4	Chapter 2: Changed "edge" to "half" in IBUFG – Global Clock Input Buffer description on page 47, page 48, and page 49. Chapter 4: Added new text and equation to "Almost Empty Flag," page 145. Added note 1 to Table 4-19, page 147. Chapter 5: Changed RAM#XM to RAM#M in Figure 5-32, page 210. Chapter 6: Corrected PCI acronym definition in "PCI-X, PCI-33, PCI-66 (Peripheral Component Interconnect)," page 245. Added to the description of the SSTL18_II_T_DCI standard in "SSTL18_II_T_DCI (1.8V) Split-Thevenin Termination," page 291. Chapter 7: Added mode to caption of Figure 7-7, page 321 for clarification. Chapter 8: Added statement about shared resources between OCLK and CLK in "High-Speed Clock for Strobe-Based Memory Interfaces - OCLK," page 355.

Date	Version	Revision
01/09/09	4.5	Chapter 4: Revised the paragraph below Equation 4-1 on page 145. Chapter 6: Added IBUFDS_DIFF_OUT to the list of primitive names for differential I/O standards in "Virtex-5 FPGA SelectIO Primitives," page 231. Added new section "IBUFDS_DIFF_OUT," page 233. Chapter 7: In the Verilog code segment for bidirectional IODELAY on page 331, corrected the setting of RST.

Table of Contents

Chapter 3: Phase-Locked Loops (PLLs)

Chapter 4: Block RAM

Chapter 5: Configurable Logic Blocks (CLBs)

Chapter 6: SelectIO Resources

Chapter 7: SelectIO Logic Resources

Chapter 8: Advanced SelectIO Logic Resources

About This Guide

This document describes the Virtex®-5 architecture. Complete and up-to-date documentation of the Virtex-5 family of FPGAs is available on the Xilinx website at http://www.xilinx.com/virtex5.

Additional Documentation

The following documents are also available for download at http://www.xilinx.com/virtex5.

- Virtex-5 Family Overview

  The features and product selection of the Virtex-5 family are outlined in this overview.

- Virtex-5 FPGA Data Sheet: DC and Switching Characteristics

  This data sheet contains the DC and Switching Characteristic specifications for the Virtex-5 family.

- Virtex-5 FPGA RocketIO GTP Transceiver User Guide

  This guide describes the RocketIO™ GTP transceivers available in the Virtex-5 LXT and SXT platforms.

- Virtex-5 FPGA RocketIO GTX Transceiver User Guide

  This guide describes the RocketIO GTX transceivers available in the Virtex-5 TXT and FXT platforms.

- Virtex-5 FPGA Embedded Processor Block for PowerPC® 440 Designs

  This reference guide is a description of the embedded processor block available in the Virtex-5 FXT platform.

- Virtex-5 FPGA Tri-Mode Ethernet Media Access Controller

  This guide describes the dedicated Tri-Mode Ethernet Media Access Controller available in the Virtex-5 LXT, SXT, TXT and FXT platforms.

- Virtex-5 FPGA Integrated Endpoint Block User Guide for PCI Express Designs

  This guide describes the integrated Endpoint blocks in the Virtex-5 LXT, SXT, TXT and FXT platforms used for PCI Express® designs.

- XtremeDSP Design Considerations

  This guide describes the XtremeDSP™ slice and includes reference designs for using the DSP48E slice.

- Virtex-5 FPGA Configuration Guide

  This all-encompassing configuration guide includes chapters on configuration interfaces (serial and SelectMAP), bitstream encryption, Boundary-Scan and JTAG configuration, reconfiguration techniques, and readback through the SelectMAP and JTAG interfaces.

- Virtex-5 FPGA System Monitor User Guide

  The System Monitor functionality available in all the Virtex-5 devices is outlined in this guide.

- Virtex-5 FPGA Packaging and Pinout Specifications

  This specification includes the tables for device/package combinations and maximum I/Os, pin definitions, pinout tables, pinout diagrams, mechanical drawings, and thermal specifications.

- Virtex-5 FPGA PCB Designer's Guide

  This guide provides information on PCB design for Virtex-5 devices, with a focus on strategies for making design decisions at the PCB and interface level.

Additional Support Resources

To search the database of silicon and software questions and answers, or to create a technical support case in WebCase, see the Xilinx website at: http://www.xilinx.com/support.

Typographical Conventions

This document uses the following typographical conventions. An example illustrates each convention.

Convention	Meaning or Use	Example
Italic font	References to other documents	See the *Virtex-5 Configuration Guide* for more information.
	Emphasis in text	The address (F) is asserted *after* clock event 2.
Underlined Text	Indicates a link to a web page.	http://www.xilinx.com/virtex5

Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section "Additional Documentation" for details. Refer to "Clock Management Technology" in Chapter 2 for details.
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest documentation.

XILINX®

Chapter 1

Clock Resources

Global and Regional Clocks

For clocking purposes, each Virtex®-5 device is divided into regions. The number of regions varies with device size, eight regions in the smallest device to 24 regions in the largest one.

Global Clocks

Each Virtex-5 device has 32 global clock lines that can clock all sequential resources on the whole device (CLB, block RAM, CMTs, and I/O), and also drive logic signals. Any ten of these 32 global clock lines can be used in any region. Global clock lines are only driven by a global clock buffer, which can also be used as a clock enable circuit, or a glitch-free multiplexer. It can select between two clock sources, and can also switch away from a failed clock source.

A global clock buffer is often driven by a Clock Management Tile (CMT) to eliminate the clock distribution delay, or to adjust its delay relative to another clock. There are more global clocks than CMTs, but a CMT often drives more than one global clock.

Regional Clocks and I/O Clocks

Each region has two regional clock buffers and four regional clock trees. A Virtex-5 I/O bank spans exactly one region with the exception of some banks in the center column. Each bank with the size identical to a region contains four clock-capable clock inputs. Each of these inputs can differentially or single-endedly drive four I/O clocks and two regional clocks in the same bank or region. In addition, regional clocks can drive regional clock trees in the adjacent regions. When the clock-capable I/Os are driven by single-ended clocks, then the clock must be connected to the positive (P) side of the differential "clock capable" pin pair. The negative (N) side can be used as a general purpose I/O or left unconnected.

The regional clock buffer can be programmed to divide the incoming clock rate by any integer number from 1 to 8. This feature, in conjunction with the programmable serializer/deserializer in the IOB, (see Chapter 8, "Advanced SelectIO Logic Resources"), allows source-synchronous systems to cross clock domains without using additional logic resources.

A third type of clocking resource, I/O clocks, are very fast and serve localized I/O serializer/deserializer circuits. See Chapter 8, "Advanced SelectIO Logic Resources."

Global Clocking Resources

Global clocks are a dedicated network of interconnect specifically designed to reach all clock inputs to the various resources in an FPGA. These networks are designed to have low skew and low duty cycle distortion, low power, and improved jitter tolerance. They are also designed to support very high frequency signals.

Understanding the signal path for a global clock expands the understanding of the various global clock resources. The global clocking resources and network consist of the following paths and components:

- Global Clock Inputs
- Global Clock Buffers
- Clock Tree and Nets - GCLK
- Clock Regions

Global Clock Inputs

Virtex-5 FPGAs contain specialized global clock input locations for use as regular user I/Os if not used as clock inputs. There are 20 global clock inputs per device. Clock inputs can be configured for any I/O standard, including differential I/O standards. Each clock input can be either single-ended or differential. All 20 clock inputs can be differential if desired. When used as outputs, global clock input pins can be configured for any output standard. Each global clock input pin supports any single-ended output standard or any output differential standard.

Global Clock Input Buffer Primitives

The primitives in Table 1-1 are different configurations of the input clock I/O input buffer.

Table 1-1: **Clock Buffer Primitives**

Primitive	Input	Output	Description
IBUFG	I	O	Input clock buffer for single-ended I/O
IBUFGDS	I, IB	O	Input clock buffer for differential I/O

These two primitives work in conjunction with the Virtex-5 I/O resource by setting the IOSTANDARD attribute to the desired standard. Refer to Chapter 6, "I/O Compatibility" Table 6-39 for a complete list of possible I/O standards.

Clock Gating for Power Savings

The Virtex-5 clock architecture provides a straightforward means of implementing clock gating for the purposes of powering down portions of a design. Most designs contain several unused BUFGCE resources. A clock can drive a BUFGCE input, and a BUFGCE output can drive distinct regions of logic. For example, if all the logic that is required to always be operating is constrained to a few clocking regions, then the BUFGCE output can drive those regions. Toggling the enable of the BUFGCE provides a simple means of stopping all dynamic power consumption in the logic regions available for power savings.

The Xilinx Power Estimator (XPE) or the Xilinx Power Analyzer (XPower) tools are used to estimate power savings. The difference is calculated by setting the frequency on the corresponding clock net to 0 MHz or providing the appropriate stimulus data to the tool.

Global Clock Buffers

There are 32 global clock buffers in every Virtex-5 device. Each half of the die (top/bottom) contains 16 global clock buffers. A global clock input can directly connect from the P-side of the differential input pin pair to any global clock buffer input in the same half, either top or bottom, of the device. Each differential global clock pin pair can connect to either a differential or single-ended clock on the PCB. If using a single-ended clock, then the P-side of the pin pair must be used because a direct connection only exists on this pin. For pin naming conventions please refer to the *Virtex-5 Family Packaging Specifications*. A single-ended clock must be connected to the positive (P) side of the differential global clock pins. If a single-ended clock is connected to the P-side of a differential pin pair, then the N-side can not be used as another single-ended clock pin. However, it can be used as a user I/O. The 20 global clock pins on Virtex-5 devices can be connected to 20 differential or 20 single-ended board clocks.

Global clock buffers allow various clock/signal sources to access the global clock trees and nets. The possible sources for input to the global clock buffers include:

- Global clock inputs
- Clock Management Tile (CMT) outputs including:
  - Digital Clock Managers (DCMs)
  - Phase-Locked Loops (PLLs)
- Other global clock buffer outputs
- General interconnect

The global clock buffers can only be driven by sources in the same half of the die (top/bottom).

All global clock buffers can drive all clock regions in Virtex-5 devices. The primary/secondary rules from Virtex-II and Virtex-II Pro FPGAs do not apply. However, only ten different clocks can be driven in a single clock region. A clock region (20 CLBs) is a branch of the clock tree consisting of ten CLB rows up and ten CLB rows down. A clock region only spans halfway across the device.

The clock buffers are designed to be configured as a synchronous or asynchronous glitch-free 2:1 multiplexer with two clock inputs. Virtex-5 control pins provide a wide range of functionality and robust input switching. The following subsections detail the various configurations, primitives, and use models of the Virtex-5 clock buffers.

Global Clock Buffer Primitives

The primitives in Table 1-2 are different configurations of the global clock buffers.

Table 1-2: **Global Clock Buffer Primitives**

Primitive	Input	Output	Control
BUFGCTRL	I0, I1	O	CE0, CE1, IGNORE0, IGNORE1, S0, S1
BUFG	I	O	–
BUFGCE	I	O	CE
BUFGCE_1	I	O	CE
BUFGMUX	I0, I1	O	S
BUFGMUX_1	I0, I1	O	S
BUFGMUX_VIRTEX4 (2)	I0, I1	O	S

Notes:

1. All primitives are derived from a software preset of BUFGCTRL.
2. BUFGMUX_VIRTEX4 is a legacy primitive name left over from the Virtex-4 family.

BUFGCTRL

The BUFGCTRL primitive shown in Figure 1-1, can switch between two asynchronous clocks. All other global clock buffer primitives are derived from certain configurations of BUFGCTRL. The ISE® software tools manage the configuration of all these primitives.

BUFGCTRL has four select lines, S0, S1, CE0, and CE1. It also has two additional control lines, IGNORE0 and IGNORE1. These six control lines are used to control the input I0 and I1.

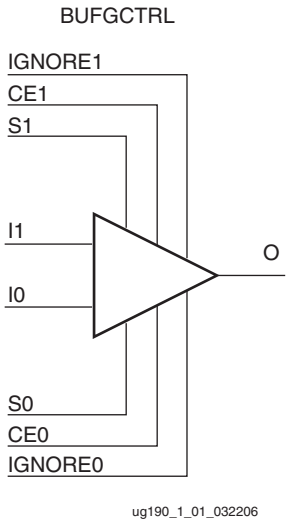


Figure 1-1: **BUFGCTRL Primitive**

BUFGCTRL is designed to switch between two clock inputs without the possibility of a glitch. When the presently selected clock transitions from High to Low after S0 and S1 change, the output is kept Low until the other (to-be-selected) clock has transitioned from High to Low. Then the new clock starts driving the output.The default configuration for BUFGCTRL is falling edge sensitive and held at Low prior to the input switching. BUFGCTRL can also be rising edge sensitive and held at High prior to the input switching.

In some applications the conditions previously described are not desirable. Asserting the IGNORE pins will bypass the BUFGCTRL from detecting the conditions for switching between two clock inputs. In other words, asserting IGNORE causes the mux to switch the inputs at the instant the select pin changes. IGNORE0 causes the output to switch away from the I0 input immediately when the select pin changes, while IGNORE1 causes the output to switch away from the I1 input immediately when the select pin changes.

Selection of an input clock requires a “select” pair (S0 and CE0, or S1 and CE1) to be asserted High. If either S or CE is not asserted High, the desired input will not be selected. In normal operation, both S and CE pairs (all four select lines) are not expected to be asserted High simultaneously. Typically only one pin of a "select" pair is used as a select line, while the other pin is tied High. The truth table is shown in Table 1-3.

Table 1-3: **Truth Table for Clock Resources**

CE0	S0	CE1	S1	O
1	1	0	X	I0
1	1	X	0	I0
0	X	1	1	I1
X	0	1	1	I1
1	1	1	1	Old Input (1)

Notes:

1. Old input refers to the valid input clock before this state is achieved.
2. For all other states, the output becomes the value of INIT_OUT and does not toggle.

Although both S and CE are used to select a desired output, each one of these pins behaves slightly different. When using CE to switch clocks, the change in clock selection can be faster than when using S. Violation in Setup/Hold time of the CE pins causes a glitch at the clock output. On the other hand, using the S pins allows the user to switch between the two clock inputs without regard to Setup/Hold times. It will not result in a glitch. See “BUFGMUX_VIRTEX4.” The CE pin is designed to allow backward compatibility from Virtex-II and Virtex-II Pro FPGAs.

The timing diagram in Figure 1-2 illustrates various clock switching conditions using the BUFGCTRL primitives. Exact timing numbers are best found using the speed specification.

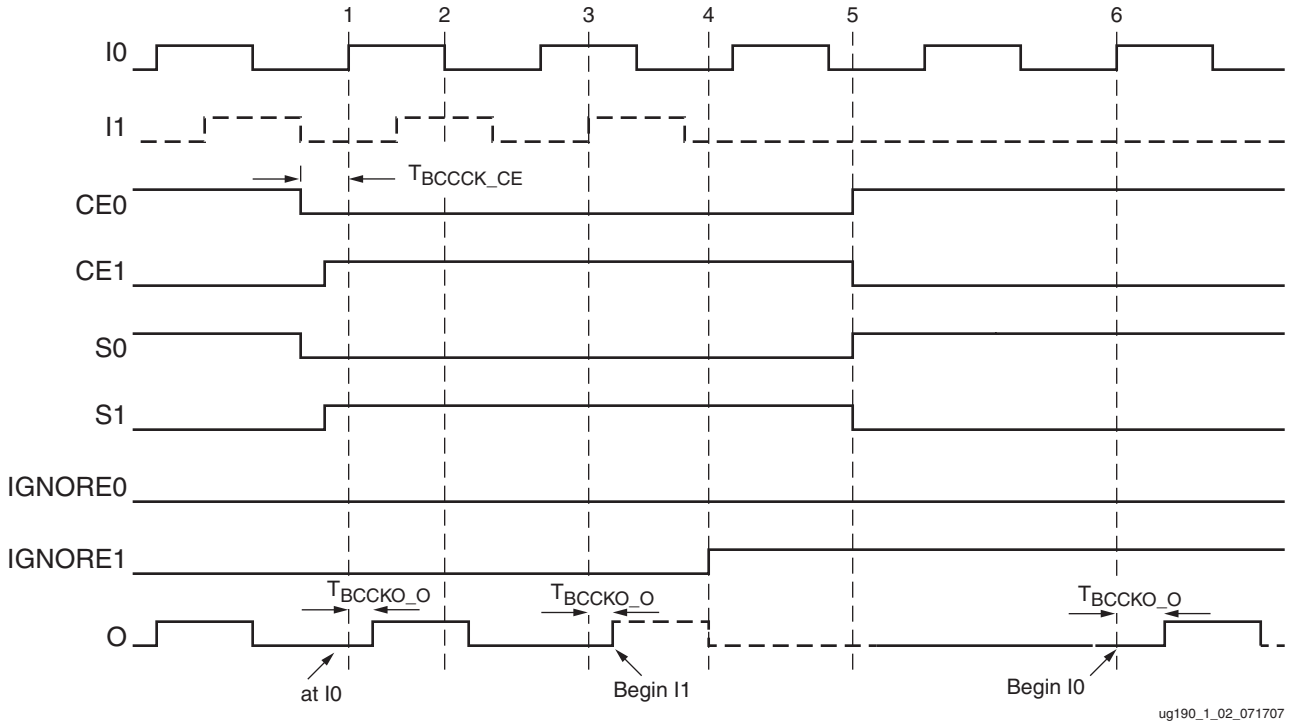


Figure 1-2: **BUFGCTRL Timing Diagram**

- Before time event 1, output O uses input I0.
- At time T_{BCCCK_CE}, before the rising edge at time event 1, both CE0 and S0 are deasserted Low. At about the same time, both CE1 and S1 are asserted High.
- At time T_{BCCKO_O}, after time event 3, output O uses input I1. This occurs after a High to Low transition of I0 (event 2) followed by a High to Low transition of I1.
- At time event 4, IGNORE1 is asserted.
- At time event 5, CE0 and S0 are asserted High while CE1 and S1 are deasserted Low. At T_{BCCKO_O}, after time event 6, output O has switched from I1 to I0 without requiring a High to Low transition of I1.

Other capabilities of BUFGCTRL are:

- Pre-selection of the I0 and I1 inputs are made after configuration but before device operation.
- The initial output after configuration can be selected as either High or Low.
- Clock selection using CE0 and CE1 only (S0 and S1 tied High) can change the clock selection without waiting for a High to Low transition on the previously selected clock.

Table 1-4 summarizes the attributes for the BUFGCTRL primitive.

Table 1-4: **BUFGCTRL Attributes**

Attribute Name	Description	Possible Values
INIT_OUT	Initializes the BUFGCTRL output to the specified value after configuration. Sets the positive or negative edge behavior. Sets the output level when changing clock selection.	0 (default), 1
PRESELECT_I0	If TRUE, BUFGCTRL output will use the I0 input after configuration(1)	FALSE (default), TRUE
PRESELECT_I1	If TRUE, BUFGCTRL output will use the I1 input after configuration(1)	FALSE (default), TRUE

Notes:

1. Both PRESELECT attributes cannot be TRUE at the same time.
2. The LOC constraint is available.

BUFG

BUFG is simply a clock buffer with one clock input and one clock output. This primitive is based on BUFGCTRL with some pins connected to logic High or Low. Figure 1-3 illustrates the relationship of BUFG and BUFGCTRL. A LOC constraint is available for BUFG.

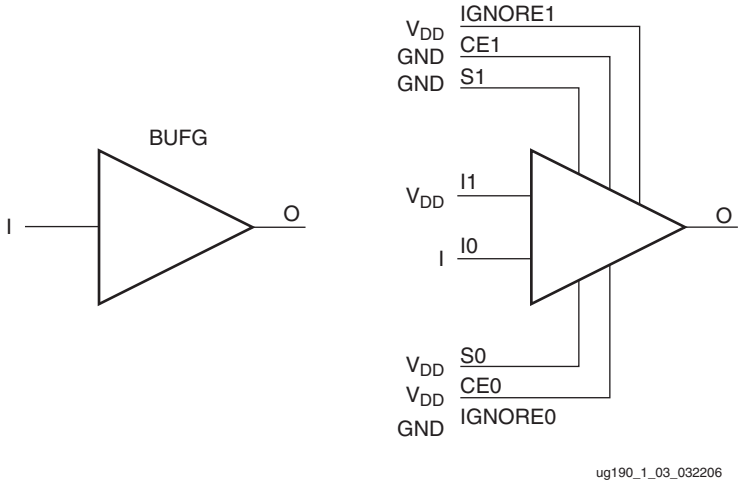


Figure 1-3: **BUFG as BUFGCTRL**

The output follows the input as shown in the timing diagram in Figure 1-4.

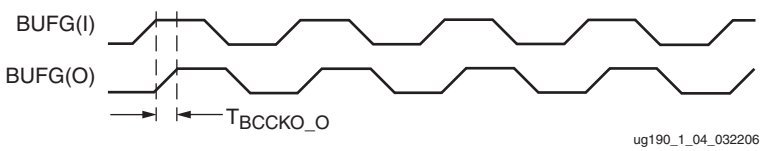


Figure 1-4: **BUFG Timing Diagram**

BUFGCE and BUFGCE_1

Unlike BUFG, BUFGCE is a clock buffer with one clock input, one clock output and a clock enable line. This primitive is based on BUFGCTRL with some pins connected to logic High or Low. Figure 1-5 illustrates the relationship of BUFGCE and BUFGCTRL. A LOC constraint is available for BUFGCE and BUFGCE_1.

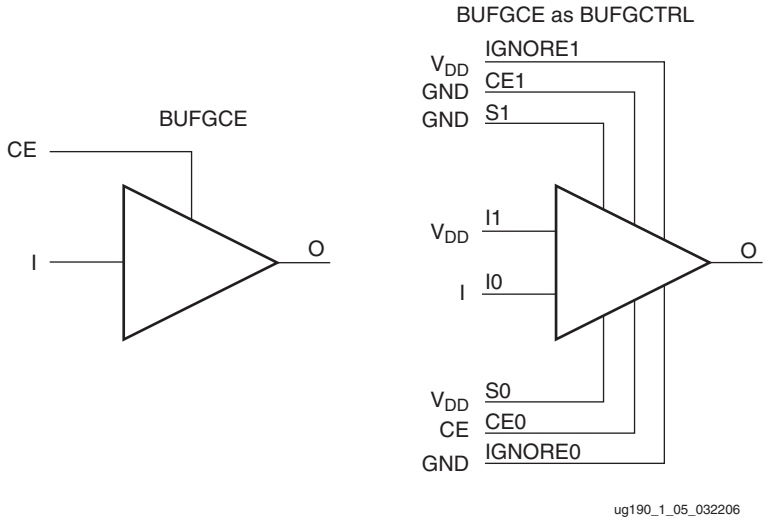


Figure 1-5: **BUFGCE as BUFGCTRL**

The switching condition for BUFGCE is similar to BUFGCTRL. If the CE input is Low prior to the incoming rising clock edge, the following clock pulse does not pass through the clock buffer, and the output stays Low. Any level change of CE during the incoming clock High pulse has no effect until the clock transitions Low. The output stays Low when the clock is disabled. However, when the clock is being disabled it completes the clock High pulse.

Since the clock enable line uses the CE pin of the BUFGCTRL, the select signal must meet the setup time requirement. Violating this setup time may result in a glitch. Figure 1-6 illustrates the timing diagram for BUFGCE.

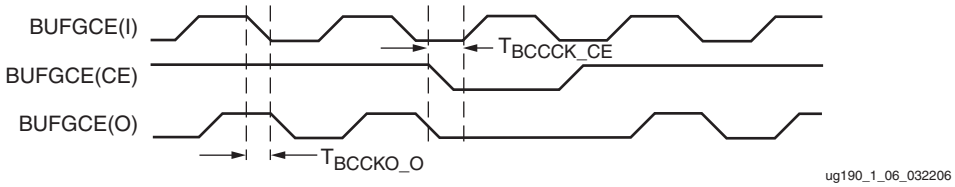


Figure 1-6: **BUFGCE Timing Diagram**

BUFGCE_1 is similar to BUFGCE, with the exception of its switching condition. If the CE input is Low prior to the incoming falling clock edge, the following clock pulse does not pass through the clock buffer, and the output stays High. Any level change of CE during the incoming clock Low pulse has no effect until the clock transitions High. The output stays High when the clock is disabled. However, when the clock is being disabled it completes the clock Low pulse.

Figure 1-7 illustrates the timing diagram for BUFGCE_1.

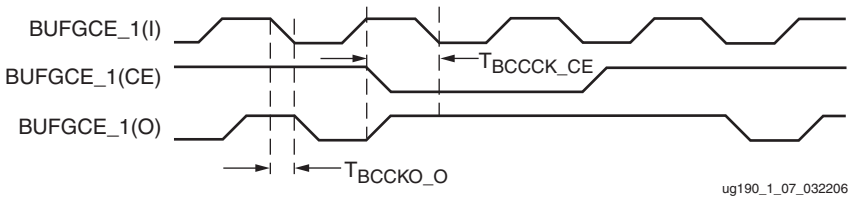


Figure 1-7: **BUFGCE_1 Timing Diagram**

BUFGMUX and BUFGMUX_1

BUFGMUX is a clock buffer with two clock inputs, one clock output, and a select line. This primitive is based on BUFGCTRL with some pins connected to logic High or Low. Figure 1-8 illustrates the relationship of BUFGMUX and BUFGCTRL. A LOC constraint is available for BUFGMUX and BUFGCTRL.

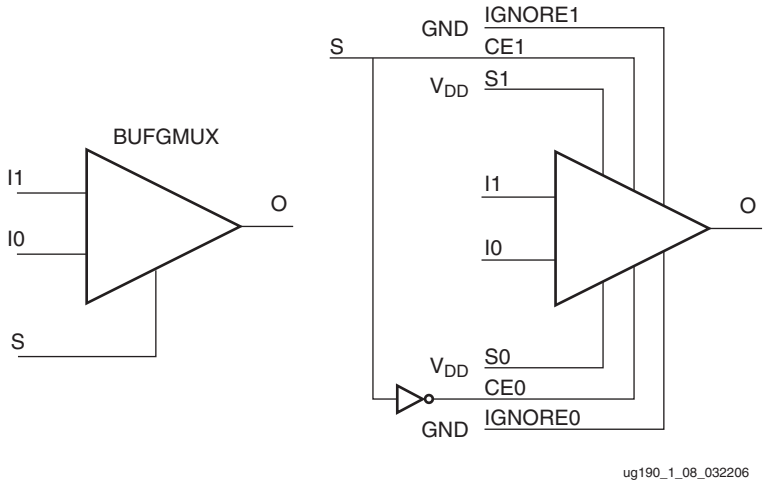


Figure 1-8: **BUFGMUX as BUFGCTRL**

Since the BUFGMUX uses the CE pins as select pins, when using the select, the setup time requirement must be met. Violating this setup time may result in a glitch.

Switching conditions for BUFGMUX are the same as the CE pins on BUFGCTRL. Figure 1-9 illustrates the timing diagram for BUFGMUX.

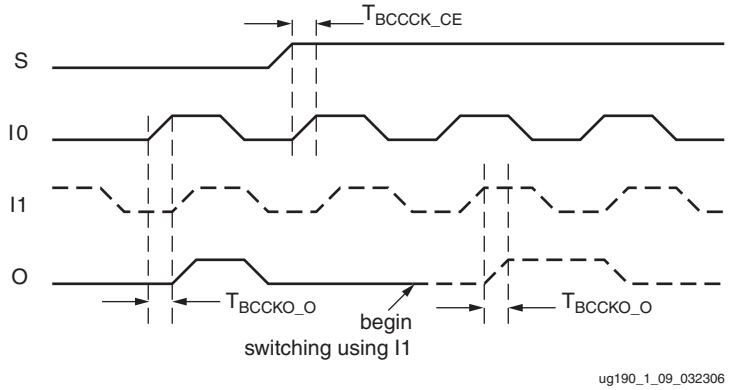


Figure 1-9: **BUFGMUX Timing Diagram**

In Figure 1-9:

- The current clock is I0.
- S is activated High.
- If I0 is currently High, the multiplexer waits for I0 to deassert Low.
- Once I0 is Low, the multiplexer output stays Low until I1 transitions High to Low.
- When I1 transitions from High to Low, the output switches to I1.
- If Setup/Hold are met, no glitches or short pulses can appear on the output.

BUFGMUX_1 is rising edge sensitive and held at High prior to input switch. Figure 1-10 illustrates the timing diagram for BUFGMUX_1. A LOC constraint is available for BUFGMUX and BUFGMUX_1.

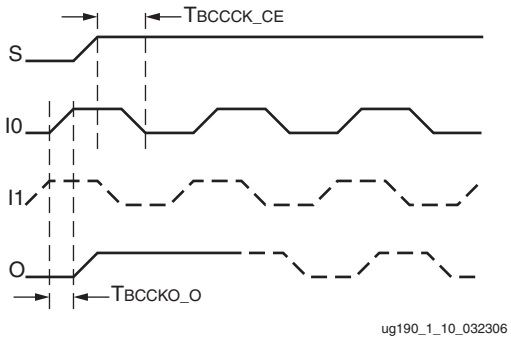


Figure 1-10: **BUFGMUX_1 Timing Diagram**

In Figure 1-10:

- The current clock is I0.
- S is activated High.
- If I0 is currently Low, the multiplexer waits for I0 to be asserted High.
- Once I0 is High, the multiplexer output stays High until I1 transitions Low to High.
- When I1 transitions from Low to High, the output switches to I1.
- If Setup/Hold are met, no glitches or short pulses can appear on the output.

BUFGMUX_VIRTEX4

BUFGMUX_VIRTEX4 is a clock buffer with two clock inputs, one clock output, and a select line. This primitive is based on BUFGCTRL with some pins connected to logic High or Low. Figure 1-11 illustrates the relationship of BUFGMUX_VIRTEX4 and BUFGCTRL.

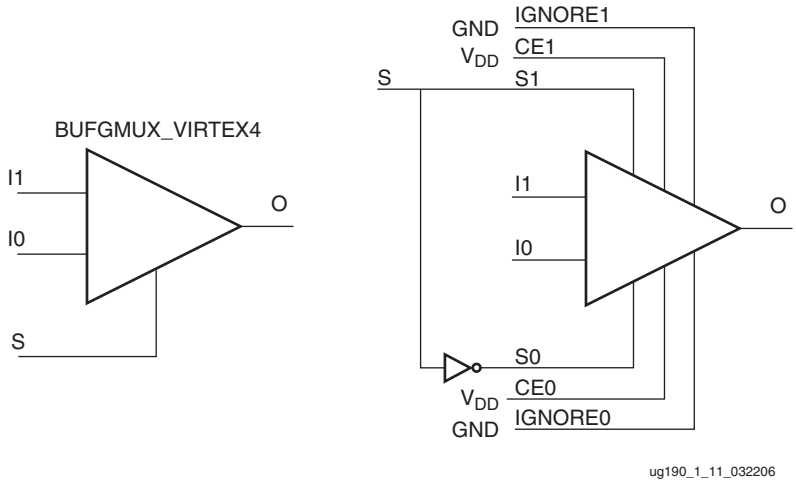


Figure 1-11: **BUFGMUX_VIRTEX4 as BUFGCTRL**

BUFGMUX_VIRTEX4 uses the S pins as select pins. S can switch anytime without causing a glitch. The Setup/Hold time on S is for determining whether the output will pass an extra pulse of the previously selected clock before switching to the new clock. If S changes as shown in Figure 1-12, prior to the setup time T_{BCCCK_S} and before I0 transitions from High to Low, then the output will not pass an extra pulse of I0. If S changes following the hold time for S, then the output will pass an extra pulse. If S violates the Setup/Hold requirements, the output might pass the extra pulse, but it will not glitch. In any case, the output will change to the new clock within three clock cycles of the slower clock.

The Setup/Hold requirements for S0 and S1 are with respect to the falling clock edge (assuming INIT_OUT = 0), not the rising edge as for CE0 and CE1.

Switching conditions for BUFGMUX_VIRTEX4 are the same as the S pin of BUFGCTRL. Figure 1-12 illustrates the timing diagram for BUFGMUX_VIRTEX4.

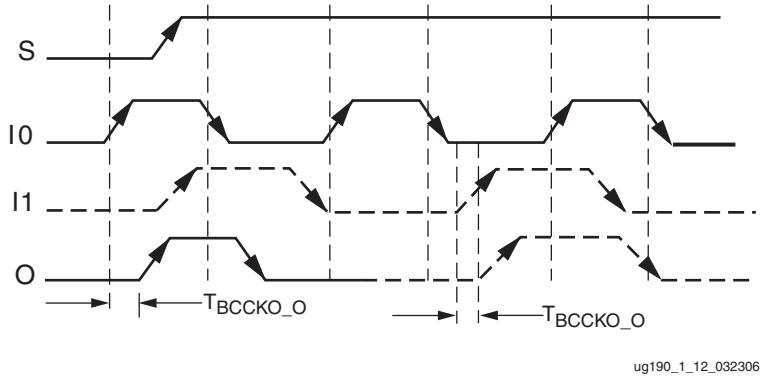


Figure 1-12: **BUFGMUX_VIRTEX4 Timing Diagram**

Other capabilities of the BUFGMUX_VIRTEX4 primitive are:

- Pre-selection of I0 and I1 input after configuration.
- Initial output can be selected as High or Low after configuration.

Additional Use Models

Asynchronous Mux Using BUFGCTRL

In some cases an application requires immediate switching between clock inputs or bypassing the edge sensitivity of BUFGCTRL. An example is when one of the clock inputs is no longer switching. If this happens, the clock output would not have the proper switching conditions because the BUFGCTRL never detected a clock edge. This case uses the asynchronous mux. Figure 1-13 illustrates an asynchronous mux with BUFGCTRL design example. Figure 1-14 shows the asynchronous mux timing diagram.

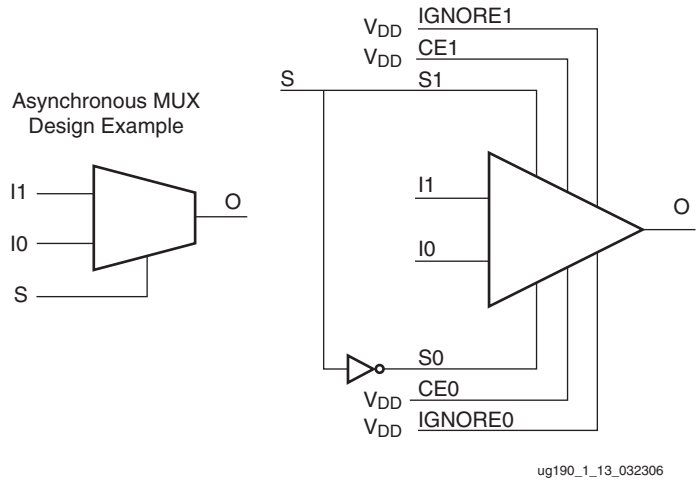


Figure 1-13: **Asynchronous Mux with BUFGCTRL Design Example**

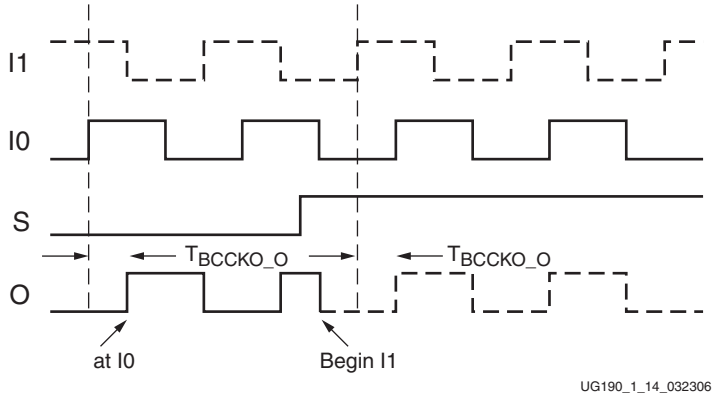


Figure 1-14: **Asynchronous Mux Timing Diagram**

In Figure 1-14:

- The current clock is from I0.
- S is activated High.
- The Clock output immediately switches to I1.
- When Ignore signals are asserted High, glitch protection is disabled.

BUFGMUX_VIRTEX4 with a Clock Enable

A BUFGMUX_VIRTEX4 with a clock enable BUFGCTRL configuration allows the user to choose between the incoming clock inputs. If needed, the clock enable is used to disable the output. Figure 1-15 illustrates the BUFGCTRL usage design example and Figure 1-16 shows the timing diagram.

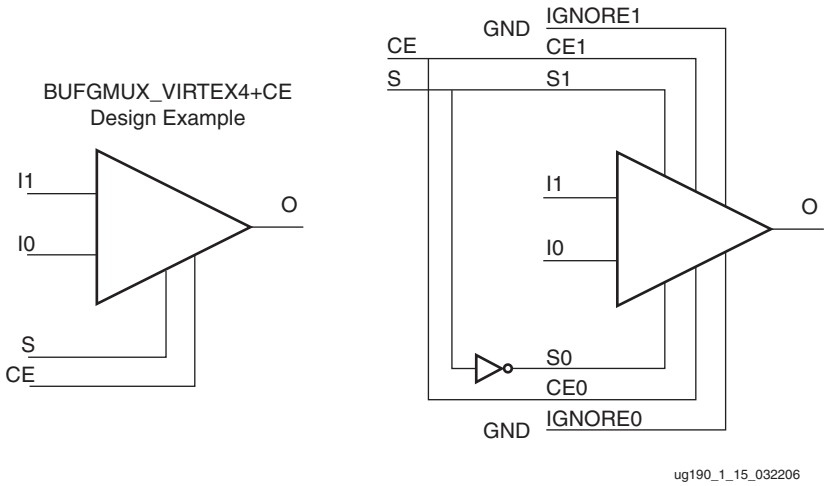


Figure 1-15: **BUFGMUX_VIRTEX4 with a CE and BUFGCTRL**

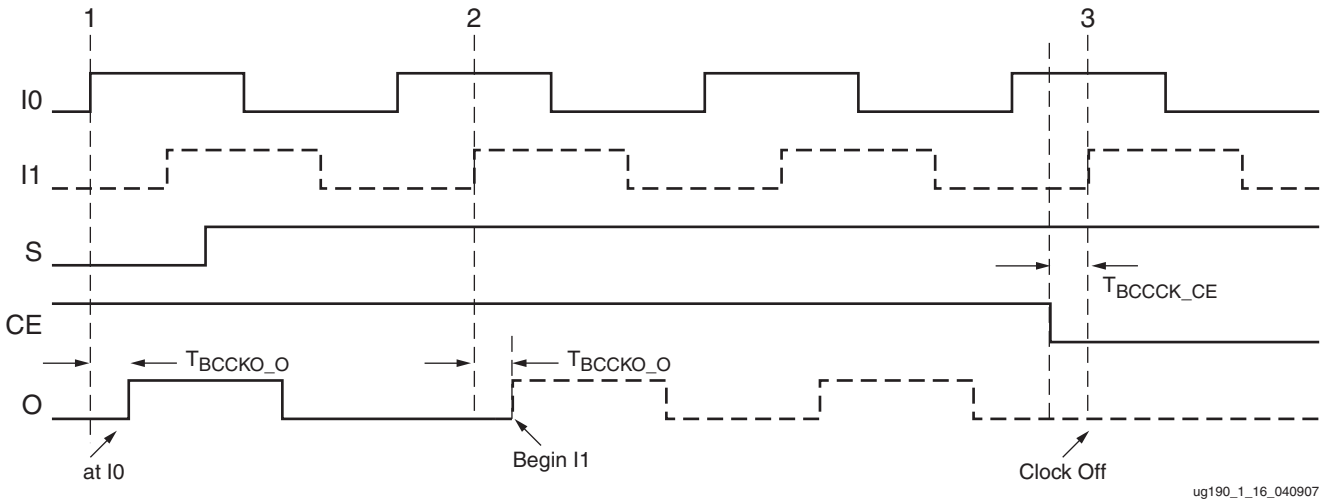


Figure 1-16: **BUFGMUX_VIRTEX4 with a CE Timing Diagram**

In Figure 1-16:

- At time event 1, output O uses input I0.
- Before time event 2, S is asserted High.
- At time T_{BCCKO_O}, after time event 2, output O uses input I1. This occurs after a High to Low transition of I0 followed by a High to Low transition of I1 is completed.
- At time T_{BCCCK_CE}, before time event 3, CE is asserted Low. The clock output is switched Low and kept at Low after a High to Low transition of I1 is completed.

Clock Tree and Nets - GCLK

Virtex-5 clock trees are designed for low-skew and low-power operation. Any unused branch is disconnected. The clock trees also manage the load/fanout when all the logic resources are used.

All global clock lines and buffers are implemented differentially. This facilitates much better duty cycles and common-mode noise rejection.

In the Virtex-5 architecture, the pin access of the global clock lines are not limited to the logic resources clock pins. The global clock lines can access other pins in the CLBs without using local interconnects. Applications requiring a very fast signal connection and large load/fanout benefit from this architecture.

Clock Regions

Virtex-5 devices improve the clocking distribution by the use of clock regions. Each clock region can have up to ten global clock domains. These ten global clocks can be driven by any combination of the 32 global clock buffers. The dimensions of a clock region are fixed to 20 CLBs tall (40 IOBs) and spanning half of the die (Figure 1-17). By fixing the dimensions of the clock region, larger Virtex-5 devices can have more clock regions. As a result, Virtex-5 devices can support many more multiple clock domains than previous FPGA architectures. Table 1-5 shows the number of clock regions in each Virtex-5 device. The logic resources in the center column (CMTs, IOBs, etc.) are located in the left clock regions.

The CMTs, if used, utilize the global clocks in the left regions as feedback lines. Up to four CMTs can be in a specific region. If used in the same region, IDELAYCTRL uses another global clock in that region. See Chapter 2, "Clock Management Technology."

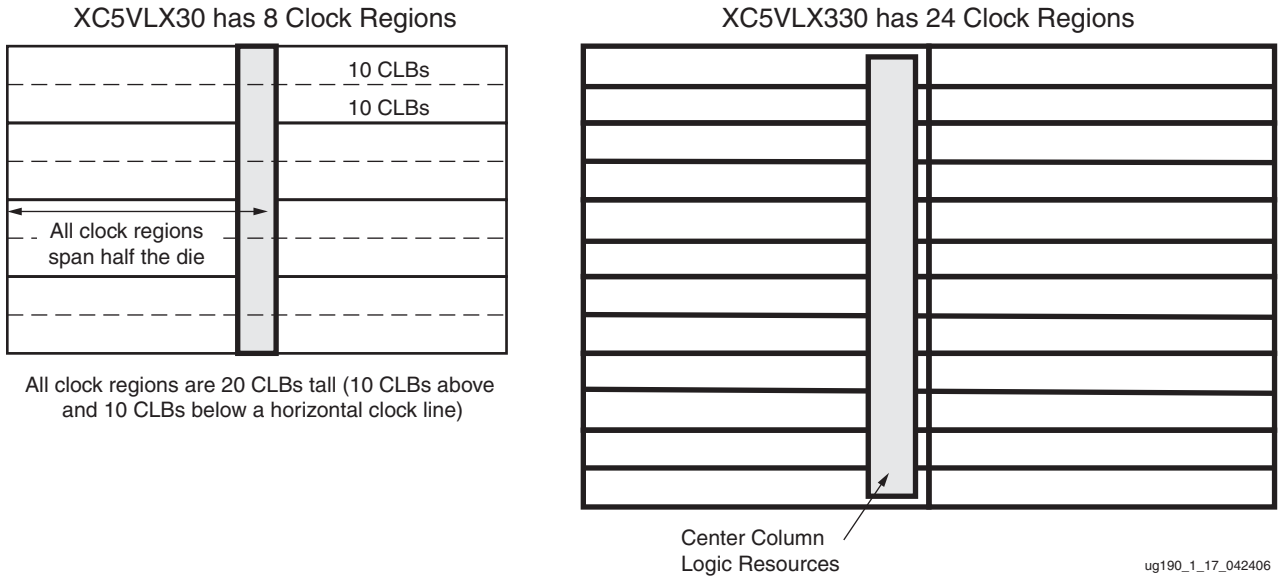


Figure 1-17: **Clock Regions**

Table 1-5: **Virtex-5 FPGA Clock Regions**

Device	Number of Clock Regions	Notes
XC5VLX30	8	
XC5VLX50	12	
XC5VLX85	12	
XC5VLX110	16	
XC5VLX155	16	
XC5VLX220	16	
XC5VLX330	24	
XC5VLX20T	6	There are 3 regions on each side of the device. There are no BUFRs on the right side of this device.
XC5VLX30T	8	
XC5VLX50T	12	
XC5VLX85T	12	
XC5VLX110T	16	
XC5VLX155T	16	
XC5VLX220T	16	
XC5VLX330T	24	
XC5VTX150T	20	
XC5VTX240T	24	
XC5VSX35T	8	
XC5VSX50T	12	
XC5VSX95T	16	
XC5VSX240T	24	
XC5VFX30T	8	
XC5VFX70T	16	
XC5VFX100T	16	
XC5VFX130T	20	
XC5VFX200T	24	

Regional Clocking Resources

Regional clock networks are a set of clock networks independent of the global clock network. Unlike global clocks, the span of a regional clock signal (BUFR) is limited to three clock regions, while the I/O clock signal drives a single region only. These networks are especially useful for source-synchronous interface designs. The I/O banks in Virtex-5 devices are the same size as a clock region.

To understand how regional clocking works, it is important to understand the signal path of a regional clock signal. The regional clocking resources and network in Virtex-5 devices consist of the following paths and components:

- Clock Capable I/O
- I/O Clock Buffer - BUFIO
- Regional Clock Buffer - BUFR
- Regional Clock Nets

Clock Capable I/O

In a typical clock region there are four clock-capable I/O pin pairs (there are exceptions in the center column). Clock-capable I/O pairs are regular I/O pairs in select locations with special hardware connections to nearby regional clock resources. Some global clock inputs are also clock-capable I/Os. There are four dedicated clock-capable I/O sites in every bank. When used as clock inputs, clock-capable pins can drive BUFIO and BUFR. Clock-capable I/Os in the center column can not drive BUFRs. Clock-capable I/Os can not directly connect to the global clock buffers. When used as single-ended clock pins, then as described in "Global Clock Buffers," the P-side of the pin pair must be used because a direct connection only exists on this pin.

I/O Clock Buffer - BUFIO

The I/O clock buffer (BUFIO) is a clock buffer available in Virtex-5 devices. The BUFIO drives a dedicated clock net within the I/O column, independent of the global clock resources. Thus, BUFIOs are ideally suited for source-synchronous data capture (forwarded/receiver clock distribution). BUFIOs can only be driven by clock capable I/Os located in the same clock region. In a typical clock region, there are four BUFIOs. Each BUFIO can drive a single I/O clock network in the same region/bank, as well as the regional clock buffers (BUFR). BUFIOs cannot drive logic resources (CLB, block RAM, IODELAY, etc.) because the I/O clock network only reaches the I/O column in the same bank or clock region.

BUFIO Primitive

BUFIO is simply a clock in, clock out buffer. There is a phase delay between input and output. Figure 1-18 shows the BUFIO. Table 1-6 lists the BUFIO ports. A location constraint is available for BUFIO.

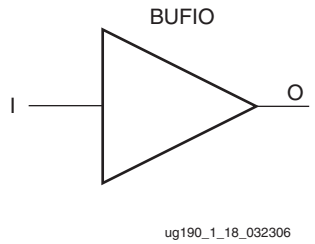


Figure 1-18: **BUFIO Primitive**

Table 1-6: **BUFIO Port List and Definitions**

Port Name	Type	Width	Definition
O	Output	1	Clock output port
I	Input	1	Clock input port

BUFIO Use Models

In Figure 1-19, a BUFIO is used to drive the I/O logic using the clock capable I/O. This implementation is ideal in source-synchronous applications where a forwarded clock is used to capture incoming data.

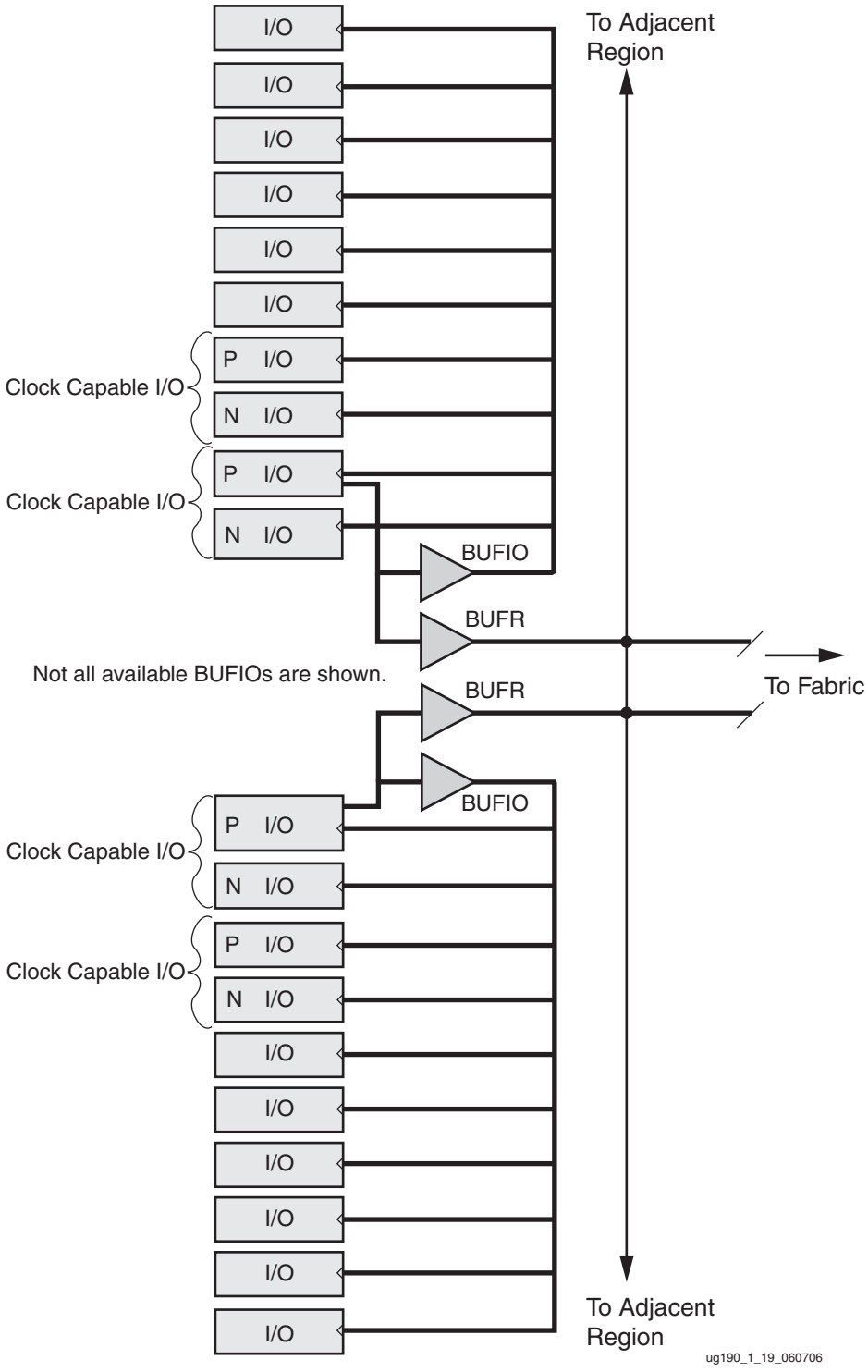


Figure 1-19: **BUFIO Driving I/O Logic In a Single Clock Region**

Regional Clock Buffer - BUFR

The regional clock buffer (BUFR) is another clock buffer available in Virtex-5 devices. BUFRs drive clock signals to a dedicated clock net within a clock region, independent from the global clock tree. Each BUFR can drive the four regional clock nets in the region it is located, and the four clock nets in the adjacent clock regions (up to three clock regions). Unlike BUFIOs, BUFRs can drive the I/O logic *and* logic resources (CLB, block RAM, etc.) in the existing and adjacent clock regions. BUFRs can be driven by clock capable pins or local interconnect. In addition, BUFR is capable of generating divided clock outputs with respect to the clock input. The divide values are an integer between one and eight. BUFRs are ideal for source-synchronous applications requiring clock domain crossing or serial-to-parallel conversion. There are two BUFRs in a typical clock region (four regional clock networks). The center column does not have BUFRs.

BUFR Primitive

BUFR is a clock-in/clock-out buffer with the capability to divide the input clock frequency.

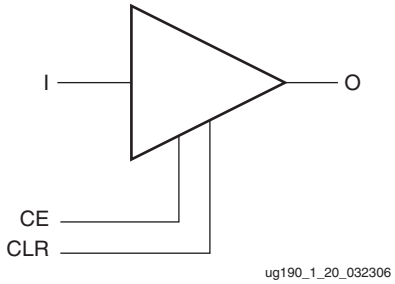


Figure 1-20: **BUFR Primitive**

Table 1-7: **BUFR Port List and Definitions**

Port Name	Type	Width	Definition
O	Output	1	Clock output port
CE	Input	1	Clock enable port. Cannot be used in BYPASS mode.
CLR	Input	1	Asynchronous clear for the divide logic, and sets the output Low. Cannot be used in BYPASS mode.
I	Input	1	Clock input port

Additional Notes on the CE Pin

When CE is asserted/deasserted, the output clock signal turns on/off. When global set/reset (GSR) signal is High, BUFR does not toggle, even if CE is held High. The BUFR output toggles after the GSR signal is deasserted when a clock is on the BUFR input port.

BUFR Attributes and Modes

Clock division in the BUFR is controlled in software through the BUFR_DIVIDE attribute. Table 1-8 lists the possible values when using the BUFR_DIVIDE attribute.

Table 1-8: **BUFR_DIVIDE Attribute**

Attribute Name	Description	Possible Values
BUFR_DIVIDE	Defines whether the output clock is a divided version of the input clock.	1, 2, 3, 4, 5, 6, 7, 8 BYPASS (default)

Notes:

1. Location constraint is available for BUFR.

The propagation delay through BUFR is different for BUFR_DIVIDE = 1 and BUFR_DIVIDE = BYPASS. When set to 1, the delay is slightly more than BYPASS. All other divisors have the same delay BUFR_DIVIDE = 1. The phase relationship between the input clock and the output clock is the same for all possible divisions except BYPASS.

The timing relationship between the inputs and output of BUFR when using the BUFR_DIVIDE attribute is illustrated in Figure 1-21. In this example, the BUFR_DIVIDE attribute is set to three. Sometime before this diagram CLR was asserted.

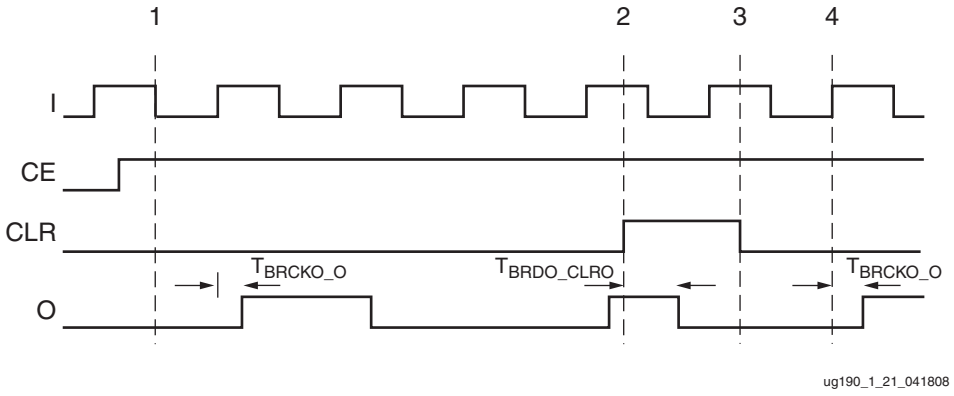


Figure 1-21: **BUFR Timing Diagrams with BUFR_DIVIDE Values**

In Figure 1-21:

- Before clock event 1, CE is asserted High.
- After CE is asserted and time T_{BRCKO_O}, the output O begins toggling at the divide by three rate of the input I. T_{BRCKO_O} and other timing numbers are best found in the speed specification.

  Note: The duty cycle is not 50/50 for odd division. The Low pulse is one cycle of I longer.
- At time event 2, CLR is asserted. After T_{BRDO_CLRO} from time event 2, O stops toggling.
- At time event 3, CLR is deasserted.
- At time T_{BRCKO_O} after clock event 4, O begins toggling again at the divided by three rate of I.

BUFR Use Models

BUFRs are ideal for source-synchronous applications requiring clock domain crossing or serial-to-parallel conversion. Unlike BUFIOs, BUFRs are capable of clocking logic resources in the FPGAs other than the IOBs. Figure 1-22 is a BUFR design example.

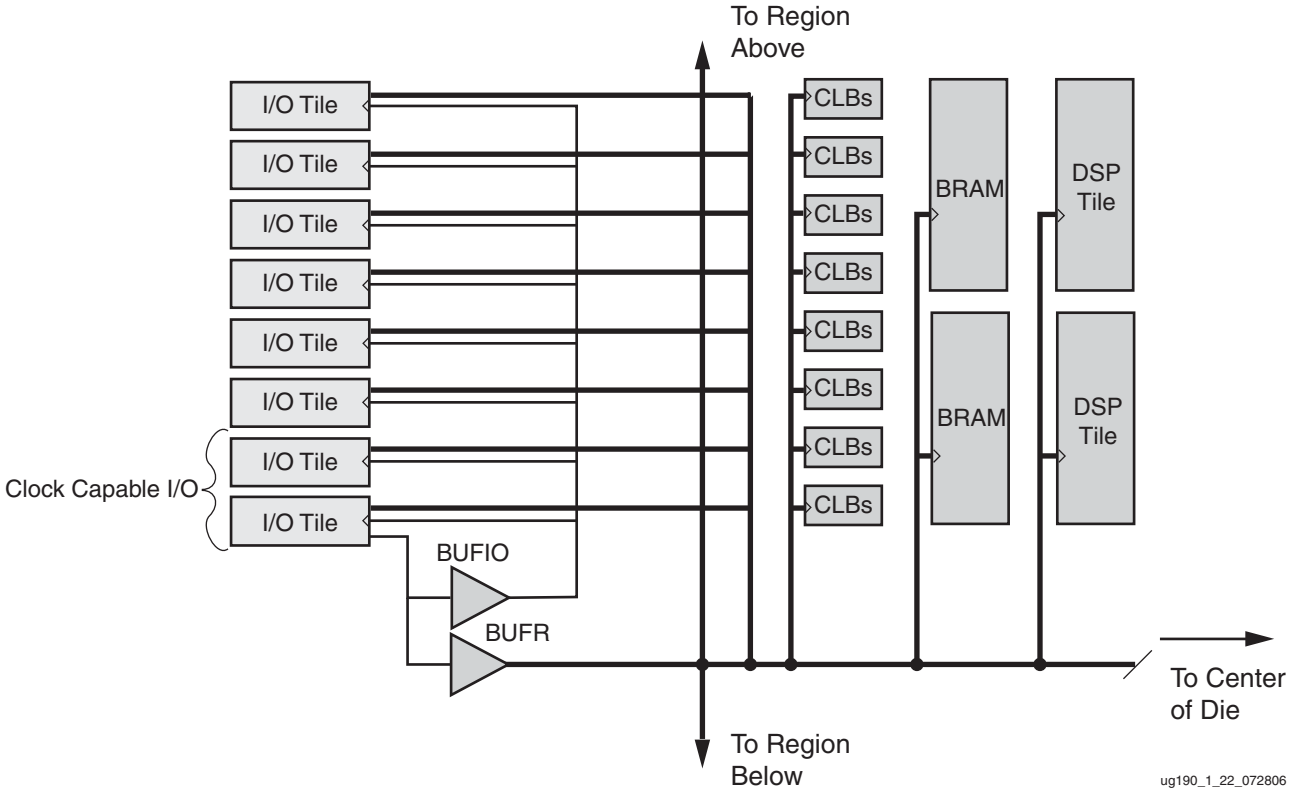


Figure 1-22: **BUFR Driving Various Logic Resources**

Regional Clock Nets

In addition to global clock trees and nets, Virtex-5 devices contain regional clock nets. These clock trees are also designed for low-skew and low-power operation. Unused branches are disconnected. The clock trees also manage the load/fanout when all the logic resources are used.

Regional clock nets do not propagate throughout the whole Virtex-5 device. Instead, they are limited to only one clock region. One clock region contains four independent regional clock nets.

To access regional clock nets, BUFRs must be instantiated. A BUFR can drive regional clocks in up to two adjacent clock regions (Figure 1-23). BUFRs in the top or bottom region can only access one adjacent region; below or above respectively. The left side BUFRs can feed the center column I/Os.

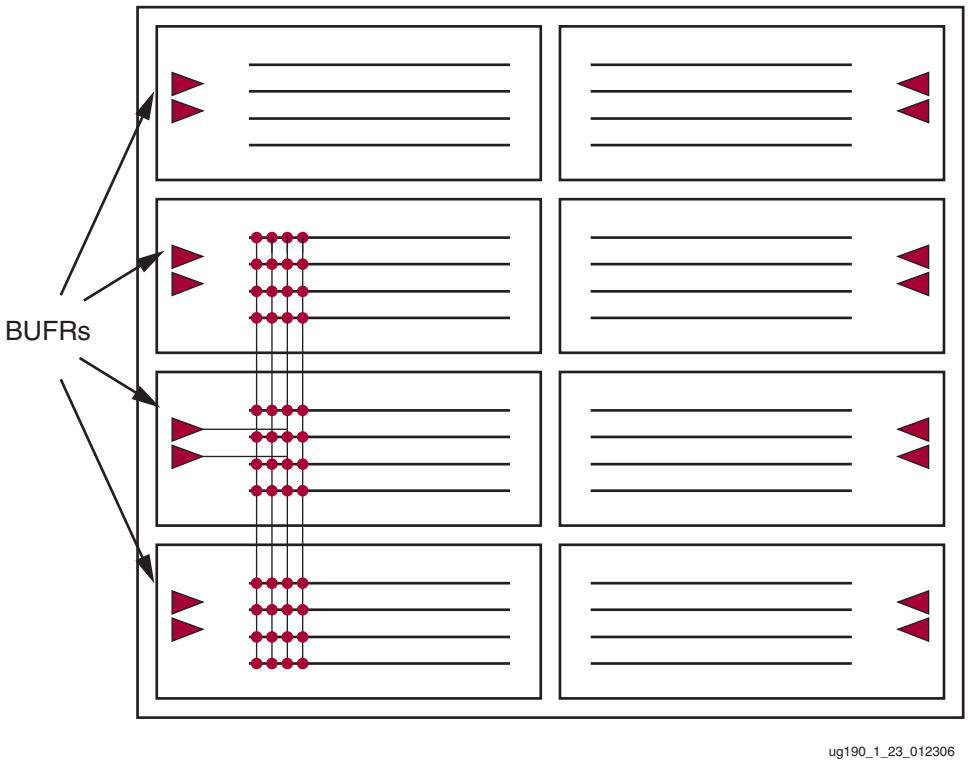


Figure 1-23: **BUFR Driving Multiple Regions**

VHDL and Verilog Templates

The VHDL and Verilog code for all clocking resource primitives and ISE language templates are available in the Libraries Guide.

Chapter 2

Clock Management Technology

Clock Management Summary

The Clock Management Tiles (CMTs) in the Virtex-5 family provide very flexible, high-performance clocking. Each CMT contains two DCMs and one PLL. Figure 2-1 shows a simplified view of the center column resources including the CMT block, where the DCM is located. Each CMT block contains two DCMs and one PLL.

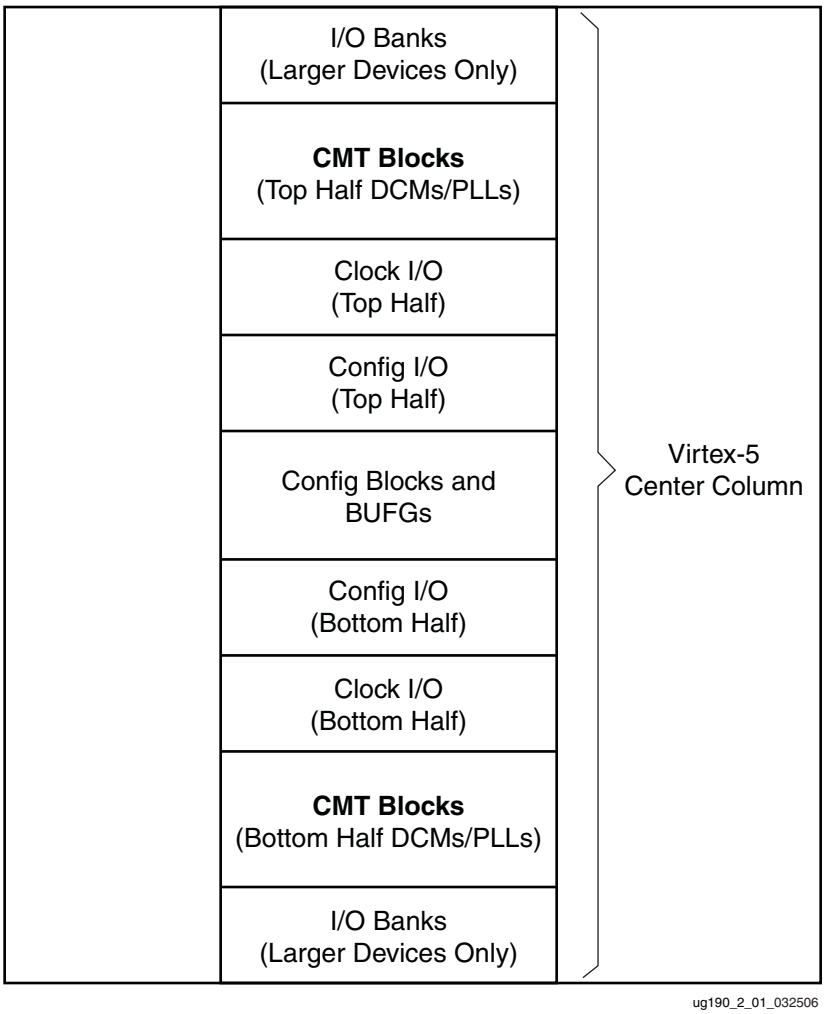


Figure 2-1: **CMT Location**

Table 2-1 summarizes the availability of CMTs, DCMs, and PLLs in each Virtex-5 device.

Table 2-1: **Available CMT, DCM, and PLL Resources**

Device	Number of CMTs	Available DCMs	Site Names
XC5VLX20T	1	2	Bottom half: DCM_ADV_X0Y0, DCM_ADV_X0Y1, PLL_ADV_X0Y0
XC5VLX30 XC5VFX30T XC5VLX30T XC5VSX35T	2	4	Bottom half: DCM_ADV_X0Y0, DCM_ADV_X0Y1, PLL_ADV_X0Y0 Top half: DCM_ADV_X0Y2, DCM_ADV_X0Y3, PLL_ADV_X0Y1
XC5VLX50 XC5VLX50T XC5VSX50T XC5VFX70T XC5VLX85 XC5VLX85T XC5VSX95T XC5VFX100T XC5VLX110 XC5VLX110T XC5VFX130T XC5VTX150T XC5VLX155 XC5VLX155T XC5VFX200T XC5VLX220 XC5VLX220T XC5VSX240T XC5VTX240T XC5VLX330 XC5VLX330T	6	12	Bottom half: DCM_ADV_X0Y0, DCM_ADV_X0Y1, PLL_ADV_X0Y0 DCM_ADV_X0Y2, DCM_ADV_X0Y3, PLL_ADV_X0Y1 DCM_ADV_X0Y4, DCM_ADV_X0Y5, PLL_ADV_X0Y2 Top half: DCM_ADV_X0Y6, DCM_ADV_X0Y7, PLL_ADV_X0Y3 DCM_ADV_X0Y8, DCM_ADV_X0Y9, PLL_ADV_X0Y4 DCM_ADV_X0Y10, DCM_ADV_X0Y11, PLL_ADV_X0Y5

DCM Summary

The Digital Clock Managers (DCMs) in Virtex-5 FPGAs provide a wide range of powerful clock management features:

- **Clock Deskew**

  The DCM contains a delay-locked loop (DLL) to completely eliminate clock distribution delays, by deskewing the DCM's output clocks with respect to the input clock. The DLL contains delay elements (individual small buffers) and control logic. The incoming clock drives a chain of delay elements, thus the output of every delay element represents a version of the incoming clock delayed at a different point.

  The control logic contains a phase detector and a delay-line selector. The phase detector compares the incoming clock signal (CLKIN) against a feedback input (CLKFB) and steers the delay line selector, essentially adding delay to the output of DCM until the CLKIN and CLKFB coincide.

- **Frequency Synthesis**

  Separate outputs provide a doubled frequency (CLK2X and CLK2X180). Another output, CLKDV, provides a frequency that is a specified fraction of the input frequency.

  Two other outputs, CLKFX and CLKFX180, provide an output frequency derived from the input clock by simultaneous frequency division and multiplication. The user can specify any integer multiplier (M) and divisor (D) within the range specified in the DCM Timing Parameters section of the *Virtex-5 FPGA Data Sheet*. An internal calculator determines the appropriate tap selection, to make the output edge coincide with the input clock whenever mathematically possible. For example, M = 9 and D = 5, multiply the frequency by 1.8, and the output rising edge is coincident with the input rising edge after every fifth input period, or after every ninth output period.

- **Phase Shifting**

  The DCM allows coarse and fine-grained phase shifting. The coarse phase shifting uses the 90°, 180°, and 270° phases of CLK0 to make CLK90, CLK180, and CLK270 clock outputs. The 180° phase of CLK2X and CLKFX provide the respective CLK2X180 and CLKFX180 clock outputs.

  There are also four modes of fine-grained phase-shifting; fixed, variable-positive, variable-center, and direct modes. Fine-grained phase shifting allows all DCM output clocks to be phase-shifted with respect to CLKIN while maintaining the relationship between the coarse phase outputs. With fixed mode, a fixed fraction of phase shift can be defined during configuration and in multiples of the clock period divided by 256. Using the variable-positive and variable-center modes the phase can be dynamically and repetitively moved forward and backwards by 1/256 of the clock period. With the direct mode the phase can be dynamically and repetitively moved forward and backwards by the value of one DCM_TAP. See the DCM Timing Parameters section in the *Virtex-5 FPGA Data Sheet*.

- **Dynamic Reconfiguration**

  There is a bus connection to the DCM to change DCM attributes without reconfiguring the rest of the device. For more information, see the Dynamic Reconfiguration chapter of the *Virtex-5 FPGA Configuration Guide*.

  The DADDR[6:0], DI[15:0], DWE, DEN, DCLK inputs and DO[15:0], and DRDY outputs are available to dynamically reconfigure select DCM functions. With dynamic reconfiguration, DCM attributes can be changed to select a different phase shift, multiply (M) or divide (D) from the currently configured settings.

DCM Primitives

The DCM primitives DCM_BASE and DCM_ADV are shown in Figure 2-2.

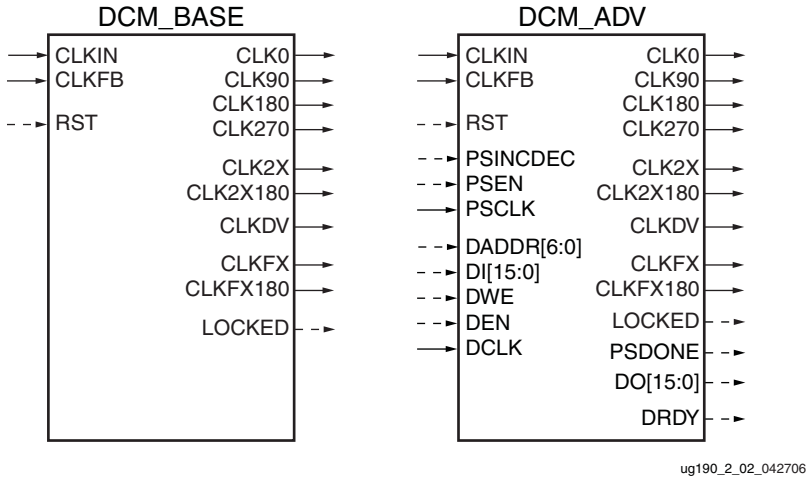


Figure 2-2: **DCM Primitives**

DCM_BASE Primitive

The DCM_BASE primitive accesses the basic frequently used DCM features and simplifies the user-interface ports. The clock deskew, frequency synthesis, and fixed-phase shifting features are available to use with DCM_BASE. Table 2-2 lists the available ports in the DCM_BASE primitive.

Table 2-2: **DCM_BASE Primitive**

Available Ports	Port Names
Clock Input	CLKIN, CLKFB
Control and Data Input	RST
Clock Output	CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, CLKDV, CLKFX, CLKFX180
Status and Data Output	LOCKED

DCM_ADV Primitive

The DCM_ADV primitive has access to all DCM features and ports available in DCM_BASE plus additional ports for the dynamic reconfiguration feature. It is a superset of the DCM_BASE primitive. DCM_ADV uses all the DCM features including clock deskew, frequency synthesis, fixed or variable phase shifting, and dynamic reconfiguration. Table 2-3 lists the available ports in the DCM_ADV primitive.

Table 2-3: **DCM_ADV Primitive**

Available Ports	Port Names
Clock Input	CLKIN, CLKFB, PSCLK, DCLK
Control and Data Input	RST, PSINCDEC, PSEN, DADDR[6:0], DI[15:0], DWE, DEN
Clock Output	CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, CLKDV, CLKFX, CLKFX180
Status and Data Output	LOCKED, PSDONE, DO[15:0], DRDY

DCM Ports

There are four types of DCM ports available in the Virtex-5 architecture:

- DCM Clock Input Ports
- DCM Control and Data Input Ports
- DCM Clock Output Ports
- DCM Status and Data Output Ports

DCM Clock Input Ports

Source Clock Input - CLKIN

The source clock (CLKIN) input pin provides the source clock to the DCM. The CLKIN frequency must fall in the ranges specified in the *Virtex-5 FPGA Data Sheet*. The clock input signal comes from one of the following buffers:

1. IBUFG – Global Clock Input Buffer

   The DCM compensates for the clock input path when CLKFB is connected and an IBUFG on the same half (top or bottom) of the device as the DCM is used.

2. BUFGCTRL – Internal Global Clock Buffer

   Any BUFGCTRL can drive any DCM in the Virtex-5 device using dedicated global routing. A BUFGCTRL can drive the DCM CLKIN pin when used to connect two DCMs in series.

3. PLL – Phase-Locked Loop

   A PLL block within the same CMT can drive the CLKIN input of either DCM in the CMT block. No global buffer is required in between. See "Application Examples," page 71 for more information.

4. IBUF – Input Buffer

   When an IBUF drives the CLKIN input, the PAD to DCM input skew is not compensated.

Feedback Clock Input - CLKFB

The feedback clock (CLKFB) input pin provides a reference or feedback signal to the DCM to delay-compensate the clock outputs, and align them with the clock input. To provide the necessary feedback to the DCM, connect only the CLK0 DCM output to the CLKFB pin. When the CLKFB pin is connected, all clock outputs are deskewed to CLKIN. When the CLKFB pin is not connected, DCM clock outputs are not deskewed to CLKIN. However, the relative phase relationship between all output clocks is preserved.

During internal feedback configuration, the CLK0 output of a DCM connects to a global buffer on the same top or bottom half of the device. The output of the global buffer connects to the CLKFB input of the same DCM.

During the external feedback configuration, the following rules apply:

1. To forward the clock, the CLK0 of the DCM must directly drive an OBUF or a BUFG-to-DDR configuration.
2. External to the FPGA, the forwarded clock signal must be connected to the IBUFG (GCLK pin) or the IBUF driving the CLKFB of the DCM. Both CLK and CLKFB should have identical I/O buffers.

Figure 2-9 illustrates clock forwarding with external feedback configuration.

The feedback clock input signal can be driven by one of the following buffers:

1. IBUFG – Global Clock Input Buffer

   This is the preferred source for an external feedback configuration. When an IBUFG drives a CLKFB pin of a DCM in the same top or bottom half of the device, the pad to DCM skew is compensated for deskew.

2. BUFGCTRL – Internal Global Clock Buffer

   This is an internal feedback configuration driven by CLK0.

3. IBUF – Input Buffer

   This is an external feedback configuration. When IBUF is used, the PAD to DCM input skew is not compensated and performance can not be guaranteed.

Phase-Shift Clock Input - PSCLK

The phase-shift clock (PSCLK) input pin provides the source clock for the DCM phase shift. The PSCLK can be asynchronous (in phase and frequency) to CLKIN. The phase-shift clock signal can be driven by any clock source (external or internal), including:

1. IBUF – Input Buffer
2. IBUFG – Global Clock Input Buffer

   To access the dedicated routing, only the IBUFGs on the same half of the device (top or bottom) as the DCM can be used to drive a PSCLK input of the DCM.

3. BUFGCTRL – An Internal Global Buffer
4. Internal Clock – Any internal clock using general purpose routing.

The frequency range of PSCLK is defined by PSCLK_FREQ_LF/HF. See the *Virtex-5 FPGA Data Sheet*. This input must be tied to ground when the CLKOUT_PHASE_SHIFT attribute is set to NONE or FIXED.

Dynamic Reconfiguration Clock Input - DCLK

The dynamic reconfiguration clock (DCLK) input pin provides the source clock for the DCM's dynamic reconfiguration circuit. The frequency of DCLK can be asynchronous (in phase and frequency) to CLKIN. The dynamic reconfiguration clock signal is driven by any clock source (external or internal), including:

1. IBUF – Input Buffer
2. IBUFG – Global Clock Input Buffer

   Only the IBUFGs on the same half of the device (top or bottom) as the DCM can be used to drive a CLKIN input of the DCM.
3. BUFGCTRL – An Internal Global Buffer
4. Internal Clock – Any internal clock using general purpose routing.

The frequency range of DCLK is described in the *Virtex-5 FPGA Data Sheet*. When dynamic reconfiguration is not used, this input must be tied to ground. See the dynamic reconfiguration chapter in the *Virtex-5 FPGA Configuration Guide* for more information.

DCM Control and Data Input Ports

Reset Input - RST

The reset (RST) input pin resets the DCM circuitry. The RST signal is an active High asynchronous reset. Asserting the RST signal asynchronously forces all DCM outputs Low (the LOCKED signal, all status signals, and all output clocks) after some propagation delay. When the reset is asserted, the last cycle of the clocks can exhibit a short pulse and a severely distorted duty cycle, or no longer be deskewed with respect to one another while asserting High. Deasserting the RST signal starts the locking process at the next CLKIN cycle.

To ensure a proper DCM reset and locking process, the RST signal must be held until the CLKIN signal is present and stable for at least three CLKIN cycles.

The time it takes for the DCM to lock after a reset is specified in the *Virtex-5 FPGA Data Sheet* as LOCK_DLL (for a DLL output) and LOCK_FX (for a DFS output). These are the CLK and CLKFX outputs described in "DCM Clock Output Ports." The DCM locks faster at higher frequencies. The worse-case numbers are specified in the *Virtex-5 FPGA Data Sheet*. In all designs, the DCM must be held in reset until CLKIN is stable.

Phase-Shift Increment/Decrement Input - PSINCDEC

The phase-shift increment/decrement (PSINCDEC) input signal must be synchronous with PSCLK. The PSINCDEC input signal is used to increment or decrement the phase-shift factor when PSEN is activated. As a result, the output clocks are shifted. The PSINCDEC signal is asserted High for increment or deasserted Low for decrement. This input must be tied to ground when the CLKOUT_PHASE_SHIFT attribute is set to NONE or FIXED.

Phase-Shift Enable Input - PSEN

The phase-shift enable (PSEN) input signal must be synchronous with PSCLK. A variable phase-shift operation is initiated by the PSEN input signal. It must be activated for one period of PSCLK. After PSEN is initiated, the phase change is gradual with completion indicated by a High pulse on PSDONE. There are no sporadic changes or glitches on any output during the phase transition. From the time PSEN is enabled until PSDONE is flagged, the DCM output clock moves bit-by-bit from its original phase shift to the target phase shift. The phase shift is complete when PSDONE is flagged. PSEN must be tied to ground when the CLKOUT_PHASE_SHIFT attribute is set to NONE or FIXED. Figure 2-6 shows the timing for this input.

Dynamic Reconfiguration Data Input - DI[15:0]

The dynamic reconfiguration data (DI) input bus provides reconfiguration data for dynamic reconfiguration. When not used, all bits must be assigned zeros. See the Dynamic Reconfiguration chapter of the *Virtex-5 FPGA Configuration Guide* for more information.

Dynamic Reconfiguration Address Input - DADDR[6:0]

The dynamic reconfiguration address (DADDR) input bus provides a reconfiguration address for the dynamic reconfiguration. When not used, all bits must be assigned zeros and the DO output bus reflects the DCM's status. See the Dynamic Reconfiguration chapter of the *Virtex-5 FPGA Configuration Guide* for more information.

Dynamic Reconfiguration Write Enable Input - DWE

The dynamic reconfiguration write enable (DWE) input pin provides the write enable control signal to write the DI data into the DADDR address. When not used, it must be tied Low. See the Dynamic Reconfiguration chapter of the *Virtex-5 FPGA Configuration Guide* for more information.

Dynamic Reconfiguration Enable Input - DEN

The dynamic reconfiguration enable (DEN) input pin provides the enable control signal to access the dynamic reconfiguration feature. When the dynamic reconfiguration feature is not used, DEN must be tied Low. When DEN is tied Low, DO reflects the DCM status signals. See the Dynamic Reconfiguration chapter of the *Virtex-5 FPGA Configuration Guide* for more information.

DCM Clock Output Ports

A DCM provides nine clock outputs with specific frequency and phase relationships. When CLKFB is connected, all DCM clock outputs have a fixed phase relationship to CLKIN. When CLKFB is not connected, the DCM outputs are not phase aligned. However, the phase relationship between all output clocks is preserved.

1x Output Clock - CLK0

The CLK0 output clock provides a clock with the same frequency as the DCM's effective CLKIN frequency. By default, the effective input clock frequency is equal to the CLKIN frequency. Set the CLKIN_DIVIDE_BY_2 attribute to TRUE to make the effective CLKIN frequency ½ the actual CLKIN frequency. The CLKIN_DIVIDE_BY_2 Attribute description provides further information. When CLKFB is connected, CLK0 is phase aligned to CLKIN.

1x Output Clock, 90° Phase Shift - CLK90

The CLK90 output clock provides a clock with the same frequency as the DCM's CLK0 phase-shifted by 90°.

1x Output Clock, 180° Phase Shift - CLK180

The CLK180 output clock provides a clock with the same frequency as the DCM's CLK0 phase-shifted by 180°.

1x Output Clock, 270° Phase Shift - CLK270

The CLK270 output clock provides a clock with the same frequency as the DCM's CLK0 phase-shifted by 270°.

2x Output Clock - CLK2X

The CLK2X output clock provides a clock that is phase aligned to CLK0, with twice the CLK0 frequency, and with an automatic 50/50 duty-cycle correction. Until the DCM is locked, the CLK2X output appears as a 1x version of the input clock with a 25/75 duty cycle. This behavior allows the DCM to lock on the correct edge with respect to the source clock.

2x Output Clock, 180° Phase Shift - CLK2X180

The CLK2X180 output clock provides a clock with the same frequency as the DCM's CLK2X phase-shifted by 180°.

Frequency Divide Output Clock - CLKDV

The CLKDV output clock provides a clock that is phase aligned to CLK0 with a frequency that is a fraction of the effective CLKIN frequency. The fraction is determined by the CLKDV_DIVIDE attribute. Refer to the CLKDV_DIVIDE Attribute for more information.

Frequency-Synthesis Output Clock - CLKFX

The CLKFX output clock provides a clock with the following frequency definition:

$$\text{CLKFX frequency} = (M/D) \times \text{effective CLKIN frequency}$$

In this equation, M is the multiplier (numerator) with a value defined by the CLKFX_MULTIPLY attribute. D is the divisor (denominator) with a value defined by the CLKFX_DIVIDE attribute. Specifications for M and D, as well as input and output frequency ranges for the frequency synthesizer, are provided in the *Virtex-5 FPGA Data Sheet*.

The rising edge of CLKFX output is phase aligned to the rising edges of CLK0, CLK2X, and CLKDV. When M and D to have no common factor, the alignment occurs only once every D cycles of CLK0.

Frequency-Synthesis Output Clock, 180° - CLKFX180

The CLKFX180 output clock provides a clock with the same frequency as the DCM's CLKFX phase-shifted by 180°.

DCM Status and Data Output Ports

Locked Output - LOCKED

The LOCKED output indicates whether the DCM clock outputs are valid, i.e., the outputs exhibit the proper frequency and phase. After a reset, the DCM samples several thousand clock cycles to achieve lock. After the DCM achieves lock, the LOCKED signal is asserted High. The DCM timing parameters section of the *Virtex-5 FPGA Data Sheet* provides estimates for locking times.

To guarantee an established system clock at the end of the start-up cycle, the DCM can delay the completion of the device configuration process until after the DCM is locked. The STARTUP_WAIT attribute activates this feature. The STARTUP_WAIT Attribute description provides further information.

Until the LOCKED signal is asserted High, the DCM output clocks are not valid and can exhibit glitches, spikes, or other spurious movement. In particular, the CLK2X output appears as a 1x clock with a 25/75 duty cycle.

Phase-Shift Done Output - PSDONE

The phase-shift done (PSDONE) output signal is synchronous to PSCLK. At the completion of the requested phase shift, PSDONE pulses High for one period of PSCLK. This signal also indicates a new change to the phase shift can be initiated. The PSDONE output signal is not valid if the phase-shift feature is not being used or is in fixed mode.

Status or Dynamic Reconfiguration Data Output - DO[15:0]

The DO output bus provides DCM status or data output when using dynamic reconfiguration (Table 2-4). Further information on using DO as the data output is available in the Dynamic Reconfiguration chapter of the *Virtex-5 FPGA Configuration Guide* for more information.

If the dynamic reconfiguration port is not used, using DCM_BASE instead of DCM_ADV is strongly recommended.

Table 2-4: **DCM Status Mapping to DO Bus**

DO Bit	Status	Description
DO[0]	Phase-shift overflow	Asserted when the DCM is phase-shifted beyond the allowed phase-shift value or when the absolute delay range of the phase-shift delay line is exceeded. DO[0] is deasserted if the phase shift feature is not used (CLKOUT_PHASE_SHIFT=NONE).
DO[1]	CLKIN stopped	Asserted when the input clock is stopped (CLKIN remains High or Low for one or more clock cycles). When CLKIN is stopped, the DO[1] CLKIN stopped status is asserted within nine CLKIN cycles. When CLKIN is restarted, CLK0 starts toggling and DO[1] is deasserted within nine clock cycles.
DO[2]	CLKFX stopped	Asserted when CLKFX stops. The DO[2] CLKFX stopped status is asserted within 260 cycles after CLKFX stopped. CLKFX does not resume, and DO[2] is not deasserted until the DCM is reset. DO[2] is deasserted if the CLKFX/CLKFX180 output is not used.

Table 2-4: **DCM Status Mapping to DO Bus *(Continued)***

DO Bit	Status	Description
DO[3]	CLKFB stopped	Asserted when the feedback clock is stopped (CLKFB remains High or Low for one or more clock cycles). The DO[3] CLKFB stopped status is asserted within six CLKIN cycles after CLKFB is stopped. CLKFB stopped will deassert within six CLKIN cycles when CLKFB resumes after being stopped momentarily. An occasionally skipped CLKFB does not affect the DCM operation. However, stopping CLKFB for a long time can result in the DCM losing LOCKED. When LOCKED is lost, the DCM needs to be reset to resume operation. When the DLL portion of the DCM is not used (for example, when using CLKFX output only), the CLKFB can be left unconnected. In this case, DO[3] is deasserted.
DO[15:4]	Not assigned	

When LOCKED is Low (during reset or the locking process), all the status signals are deasserted Low.

Dynamic Reconfiguration Ready Output - DRDY

The dynamic reconfiguration ready (DRDY) output pin provides the response to the DEN signal for the DCM's dynamic reconfiguration feature. Further information on the DRDY pin is available in the dynamic reconfiguration section in the *Virtex-5 FPGA Configuration Guide*.

DCM Attributes

A handful of DCM attributes govern the DCM functionality. Table 2-6 summarizes all the applicable DCM attributes. This section provides a detailed description of each attribute. For more information on applying these attributes in UCF, VHDL, or Verilog code, refer to the Constraints Guide at:
http://www.support.xilinx.com/support/software_manuals.htm.

CLKDV_DIVIDE Attribute

The CLKDV_DIVIDE attribute controls the CLKDV frequency. The source clock frequency is divided by the value of this attribute. The possible values for CLKDV_DIVIDE are: 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, or 16. The default value is 2. In the low frequency mode, any CLKDV_DIVIDE value produces a CLKDV output with a 50/50 duty-cycle. In the high frequency mode, the CLKDV_DIVIDE value must be set to an integer value to produce a CLKDV output with a 50/50 duty-cycle. For non-integer CLKDV_DIVIDE values, the CLKDV output duty cycle is shown in Table 2-5.

Table 2-5: **Non-Integer CLKDV_DIVIDE**

CLKDV_DIVIDE Value	CLKDV Duty Cycle in High Frequency Mode (High Pulse/Low Pulse Value)
1.5	1/3
2.5	2/5
3.5	3/7
4.5	4/9
5.5	5/11
6.5	6/13
7.5	7/15

CLKFX_MULTIPLY and CLKFX_DIVIDE Attribute

The CLKFX_MULTIPLY attribute sets the multiply value (M) of the CLKFX output. The CLKFX_DIVIDE attribute sets the divisor (D) value of the CLKFX output. Both control the CLKFX output making the CLKFX frequency equal the effective CLKIN (source clock) frequency multiplied by M/D. The possible values for M are any integer from two to 33. The possible values for D are any integer from 1 to 32. The default settings are M = 4 and D = 1.

CLKIN_PERIOD Attribute

The CLKIN_PERIOD attribute specifies the source clock period (in nanoseconds). The default value is 0.0 ns. Setting this attribute to the input period values produces the best results.

CLKIN_DIVIDE_BY_2 Attribute

The CLKIN_DIVIDE_BY_2 attribute is used to enable a toggle flip-flop in the input clock path to the DCM. When set to FALSE, the effective CLKIN frequency of the DCM equals the source clock frequency driving the CLKIN input. When set to TRUE, the CLKIN frequency is divided by two before it reaches the rest of the DCM. Thus, the DCM sees half the frequency applied to the CLKIN input and operates based on this frequency. For example, if a 100 MHz clock drives CLKIN, and CLKIN_DIVIDE_BY_2 is set to TRUE; then the effective CLKIN frequency is 50 MHz. Thus, CLK0 output is 50 MHz and CLK2X output is 100 MHz. The effective CLKIN frequency must be used to evaluate any operation or specification derived from CLKIN frequency. The possible values for CLKIN_DIVIDE_BY_2 are TRUE and FALSE. The default value is FALSE.

CLKOUT_PHASE_SHIFT Attribute

The CLKOUT_PHASE_SHIFT attribute indicates the mode of the phase shift applied to the DCM outputs. The possible values are NONE, FIXED, VARIABLE_POSITIVE, VARIABLE_CENTER, or DIRECT. The default value is NONE.

When set to NONE, a phase shift cannot be performed and a phase-shift value has no effect on the DCM outputs. When set to FIXED, the DCM outputs are phase-shifted by a fixed phase from the CLKIN. The phase-shift value is determined by the PHASE_SHIFT attribute. If the CLKOUT_PHASE_SHIFT attribute is set to FIXED or NONE, then the PSEN, PSINCDEC, and the PSCLK inputs must be tied to ground.

When set to VARIABLE_POSITIVE, the DCM outputs can be phase-shifted in variable mode in the positive range with respect to CLKIN. When set to VARIABLE_CENTER, the DCM outputs can be phase-shifted in variable mode, in the positive and negative range with respect to CLKIN. If set to VARIABLE_POSITIVE or VARIABLE_CENTER, each phase-shift increment (or decrement) increases (or decreases) the phase shift by a period of 1/256 x CLKIN period.

When set to DIRECT, the DCM output can be phase-shifted in variable mode in the positive range with respect to CLKIN. Each phase-shift increment/decrement will increase/decrease the phase shift by one DCM_TAP. See the *Virtex-5 FPGA Data Sheet*.

The starting phase in the VARIABLE_POSITIVE and VARIABLE_CENTER modes is determined by the phase-shift value. The starting phase in the DIRECT mode is always zero, regardless of the value specified by the PHASE_SHIFT attribute. Thus, the PHASE_SHIFT attribute should be set to zero when DIRECT mode is used. A non-zero phase-shift value for DIRECT mode can be loaded to the DCM using Dynamic Reconfiguration Ports in the *Virtex-5 FPGA Configuration Guide*.

CLK_FEEDBACK Attribute

The CLK_FEEDBACK attribute determines the type of feedback applied to the CLKFB. The possible values are 1X or NONE. The default value is 1X. When set to 1X, CLKFB pin must be driven by CLK0. When set to NONE leave the CLKFB pin unconnected.

DESKEW_ADJUST Attribute

The DESKEW_ADJUST attribute affects the amount of delay in the feedback path. The possible values are SYSTEM_SYNCHRONOUS, SOURCE_SYNCHRONOUS, 0, 1, 2, 3, ..., or 31. The default value is SYSTEM_SYNCHRONOUS.

For most designs, the default value is appropriate. In a source-synchronous design, set this attribute to SOURCE_SYNCHRONOUS. The remaining values should only be used after consulting with Xilinx. For more information, consult the "Source-Synchronous Setting"section.

DFS_FREQUENCY_MODE Attribute

The DFS_FREQUENCY_MODE attribute specifies the frequency mode of the digital frequency synthesizer (DFS). The possible values are Low and High. The default value is Low. The frequency ranges for both frequency modes are specified in the *Virtex-5 FPGA Data Sheet*. DFS_FREQUENCY_MODE determines the frequency range of CLKIN, CLKFX, and CLKFX180.

DLL_FREQUENCY_MODE Attribute

The DLL_FREQUENCY_MODE attribute specifies either the High or Low frequency mode of the delay-locked loop (DLL). The default value is Low. The frequency ranges for both frequency modes are specified in the *Virtex-5 FPGA Data Sheet*.

DUTY_CYCLE_CORRECTION Attribute

The DUTY_CYCLE_CORRECTION attribute controls the duty cycle correction of the 1x clock outputs: CLK0, CLK90, CLK180, and CLK270. The possible values are TRUE and FALSE. The default value is TRUE. When set to TRUE, the 1x clock outputs are duty cycle corrected to be within specified limits. See the *Virtex-5 FPGA Data Sheet* for details. It is strongly recommended to always set the DUTY_CYCLE_CORRECTION attribute to TRUE. Setting this attribute to FALSE does not necessarily produce output clocks with the same duty cycle as the source clock.

DCM_PERFORMANCE_MODE Attribute

The DCM_PERFORMANCE_MODE attribute allows the choice of optimizing the DCM either for high frequency and low jitter or for low frequency and a wide phase-shift range. The attribute values are MAX_SPEED and MAX_RANGE. The default value is MAX_SPEED. When set to MAX_SPEED, the DCM is optimized to produce high frequency clocks with low jitter. However, the phase-shift range is smaller than when MAX_RANGE is selected. When set to MAX_RANGE, the DCM is optimized to produce low-frequency clocks with a wider phase-shift range. The DCM_PERFORMANCE_MODE affects the following specifications: DCM input and output frequency range, phase-shift range, output jitter, DCM_TAP, CLKIN_CLKFB_PHASE, CLKOUT_PHASE, and duty-cycle precision. The *Virtex-5 FPGA Data Sheet* specifies these values.

For most cases, the DCM_PERFORMANCE_MODE attribute should be set to MAX_SPEED (default). Consider changing to MAX_RANGE only in the following situations:

- The frequency needs to be below the low-frequency limit of the MAX_SPEED setting.
- A greater absolute phase-shift range is required.

FACTORY_JF Attribute

The Factory_JF attribute affects the DCMs jitter filter characteristics. This attribute controls the DCM tap update rate. The default value is `0xF0F0` corresponding to DLL_FREQUENCY_MODE = LOW and DLL_FREQUENCY_MODE = HIGH.

PHASE_SHIFT Attribute

The PHASE_SHIFT attribute determines the amount of phase shift applied to the DCM outputs. This attribute can be used in both fixed or variable phase-shift mode. If used with variable mode, the attribute sets the starting phase shift. When CLKOUT_PHASE_SHIFT = VARIABLE_POSITIVE, the PHASE_SHIFT value range is 0 to 255. When CLKOUT_PHASE_SHIFT = VARIABLE_CENTER or FIXED, the PHASE_SHIFT value range is –255 to 255. When CLKOUT_PHASE_SHIFT = DIRECT, the PHASE_SHIFT value range is 0 to 1023. The default value is 0.

Refer to the Phase Shifting section for information on the phase-shifting operation and its relationship with the CLKOUT_PHASE_SHIFT and PHASE_SHIFT attributes.

STARTUP_WAIT Attribute

The STARTUP_WAIT attribute determines whether the DCM waits in one of the startup cycles for the DCM to lock. The possible values for this attribute are TRUE and FALSE. The default value is FALSE. When STARTUP_WAIT is set to TRUE, and the LCK_cycle BitGen option is used, then the configuration startup sequence waits in the startup cycle specified by LCK_cycle until the DCM is locked.

Table 2-6: **DCM Attributes**

DCM Attribute Name	Description	Values	Default Value
CLKDV_DIVIDE	This attribute controls CLKDV such that the source clock is divided by N. This feature provides automatic duty cycle correction such that the CLKDV output pin has a 50/50 duty cycle always in low-frequency mode, as well as for all integer values of the division factor N in high-frequency mode.	Real: 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, 16	2.0
CLKFX_DIVIDE		Integer: 1 to 32	1
CLKFX_MULTIPLY		Integer: 2 to 32	4
CLKIN_PERIOD	This specifies the source clock period to help DCM adjust for optimum CLKFX/CLKFX180 outputs.	Real in ns	0.0
CLKIN_DIVIDE_BY_2	This attribute allows for the input clock frequency to be divided in half when such a reduction is necessary to meet the DCM input clock frequency requirements.	Boolean: FALSE or TRUE	FALSE

Table 2-6: **DCM Attributes *(Continued)***

DCM Attribute Name	Description	Values	Default Value
CLKOUT_PHASE_SHIFT	This attribute specifies the phase-shift mode.	String: NONE or FIXED or VARIABLE_POSITIVE or VARIABLE_CENTER or DIRECT	NONE
DESKEW_ADJUST	This affects the amount of delay in the feedback path, and should be used for source-synchronous interfaces.	String: SYSTEM_SYNCHRONOUS or SOURCE_SYNCHRONOUS	SYSTEM_ SYNCHRONOUS
DFS_FREQUENCY_MODE	This specifies the frequency mode of the frequency synthesizer.	String: LOW or HIGH	LOW
DLL_FREQUENCY_MODE	This specifies the frequency mode of the DLL.	String: LOW or HIGH	LOW
DUTY_CYCLE_CORRECTION	This controls the DCM 1X outputs (CLK0, CLK90, CLK180, and CLK270), to exhibit a 50/50 duty cycle. Leave this attribute set at the default value.	Boolean: TRUE or FALSE	TRUE
DCM_PERFORMANCE_MODE	Allows selection between maximum frequency/ minimum jitter, and low frequency/maximum phase-shift range	String: MAX_SPEED or MAX_RANGE	MAX_SPEED
FACTORY_JF	DLL_FREQUENCY_MODE=LOW default (`0xF0F0`). DLL_FREQUENCY_MODE=HIGH default (`0xF0F0`).	BIT_VECTOR	`0xF0F0`
PHASE_SHIFT	This specifies the phase-shift numerator. The value range depends on CLKOUT_PHASE_SHIFT and clock frequency.	Integer: –255 to 1023	0
STARTUP_WAIT	When this attribute is set to TRUE, the configuration startup sequence waits in the specified cycle until the DCM locks.	Boolean: FALSE or TRUE	FALSE

DCM Design Guidelines

This section provides a detailed description on using the Virtex-5 FPGA DCM and design guidelines.

Clock Deskew

The Virtex-5 FPGA DCM offers a fully digital, dedicated, on-chip clock deskew. The deskew feature provides zero propagation delay between the source clock and output clock, low clock skew among output clock signals distributed throughout the device, and advanced clock domain control.

The deskew feature also functions as a clock mirror of a board-level clock serving multiple devices. This is achieved by driving the CLK0 output off-chip to the board (and to other devices on the board) and then bringing the clock back in as a feedback clock. See the "Application Examples" section. Taking advantage of the deskew feature greatly simplifies and improves system-level design involving high-fanout, high-performance clocks.

Clock Deskew Operation

The deskew feature utilizes the DLL circuit in the DCM. In its simplest form, the DLL consists of a single variable delay line (containing individual small delay elements or buffers) and control logic. The incoming clock drives the delay line. The output of every delay element represents a version of the incoming clock (CLKIN) delayed at a different point. The clock distribution network routes the clock to all internal registers and to the clock feedback CLKFB pin. The control logic contains a phase detector and a delay-line selector. The phase detector compares the incoming clock signal (CLKIN) against a feedback input (CLKFB) and steers the delay-line selector, essentially adding delay to the DCM output until the CLKIN and CLKFB coincide, putting the two clocks 360° out-of-phase, (thus, in phase). When the edges from the input clock line up with the edges from the feedback clock, the DCM achieves a lock. The two clocks have no discernible difference. Thus, the DCM output clock compensates for the delay in the clock distribution network, effectively removing the delay between the source clock and its loads. The size of each intrinsic delay element is a DCM_TAP (see the AC Characteristics table in the *Virtex-5 FPGA Data Sheet*). Figure 2-3 illustrates a simplified DLL circuit.

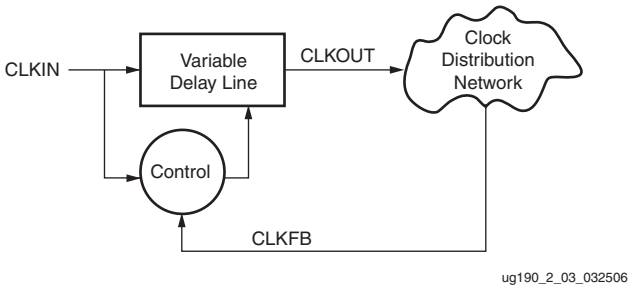


Figure 2-3: **Simplified DLL Circuit**

To provide the correct clock deskew, the DCM depends on the dedicated routing and resources used at the clock source and feedback input. An additional delay element (see "Deskew Adjust") is available to compensate for the clock source or feedback path. The Xilinx ISE tools analyze the routing around the DCM to determine if a delay must be

inserted to compensate for the clock source or feedback path. Thus, using dedicated routing is required to achieve predictable deskew.

Input Clock Requirements

The clock input of the DCM can be driven either by an IBUFG/IBUFGDS, IBUF, BUFGMUX, or a BUFGCTRL. Since there is no dedicated routing between an IBUF and a DCM clock input, using an IBUF causes additional input delay that is not compensated by the DCM and performance can not be guaranteed.

The DCM output clock signal is essentially a delayed version of the input clock signal. It reflects any instability on the input clock in the output waveform. The DCM input clock requirements are specified in the *Virtex-5 FPGA Data Sheet*.

Once locked, the DCM can tolerate input clock period variations of up to the value specified by CLKIN_PER_JITT_DLL_HF (at high frequencies) or CLKIN_PER_JITT_DLL_LF (at low frequencies). Larger jitter (period changes) can cause the DCM to lose lock, indicated by the LOCKED output deasserting. The user must then reset the DCM. The cycle-to-cycle input jitter must be kept to less than CLKIN_CYC_JITT_DLL_LF in the low frequencies and CLKIN_CYC_JITT_DLL_HF for the high frequencies.

Input Clock Changes

Changing the period of the input clock beyond the maximum input period jitter specification requires a manual reset of the DCM. Failure to reset the DCM produces an unreliable LOCKED signal and output clock. It is possible to temporarily stop the input clock and feedback clock with little impact to the deskew circuit, as long as CLKFX or CLKFX180 is not used.

If the input clock is stopped and CLKFX or CLKFX180 is used, the CLKFX or CLKFX180 outputs might stop toggling, and DO[2] (CLKFX stopped) is asserted. The DCM must be reset to recover from this event.

The DO[2] CLKFX stopped status is asserted 100 µs after CLKFX is stopped. CLKFX does not resume and DO[2] does not deassert until the DCM is reset.

In any other case, the clock should not be stopped for more than 100 ms to minimize the effect of device cooling; otherwise, the tap delays might change. The clock should be stopped during a Low or a High phase, and must be restored with the same input clock period/frequency. During this time, LOCKED stays High and remains High when the clock is restored. Thus, a High on LOCKED does not necessarily mean that a valid clock is available.

When stopping the input clock (CLKIN remains High or Low for one or more clock cycles), one to nine more output clock cycles are still generated as the delay line is flushed. When the output clock stops, the CLKIN stopped (DO[1]) signal is asserted. When the clock is restarted, the output clock cycles are not generated for one to eight clocks while the delay line is filled. The most common case is two or three clocks. The DO[1] signal is deasserted once the output clock is generated. CLKIN can be restarted with any phase relationship to the previous clock. If the frequency has changed, the DCM requires a reset. The DO[1] is forced Low whenever LOCKED is Low. When the DCM is in the locking process, DO[1] status is held Low until LOCKED is achieved.

Output Clocks

Any or all of the DCM's nine clock outputs can be used to drive a global clock network. The fully-buffered global clock distribution network minimizes clock skew caused by loading differences. By monitoring a sample of the output clock (CLK0), the deskew circuit compensates for the delay on the routing network, effectively eliminating the delay from the external input port to the individual clock loads within the device.

Output pin connectivity carries some restrictions. The DCM clock outputs must drive a global clock buffer BUFGCTRL. The DCM clock outputs can not drive general routing. To use dedicated routing, the DCM clock outputs must drive BUFGCTRLs on the same top or bottom half of the device. If the DCM and BUFGCTRL are not on the same top or bottom half, local routing is used and the DCM might not deskew properly.

Do not use the DCM output clock signals until after activation of the LOCKED signal. Prior to the activation of the LOCKED signal, the DCM output clocks are not valid.

DCM During Configuration and Startup

During the FPGA configuration, the DCM is in reset and starts to lock at the beginning of the startup sequence. A DCM requires both CLKIN and CLKFB input clocks to be present and stable when the DCM begins to lock. If the device enters the configuration startup sequence without an input clock, or with an unstable input clock, then the DCM must be reset after configuration with a stable clock.

The following startup cycle dependencies are of note:

1. The default value is **-g LCK_cycle:NoWait**. When this setting is used, the startup sequence does not wait for the DCM to lock. When the LCK_cycle is set to other values, the configuration startup remains in the specified startup cycle until the DCM is locked.
2. Before setting the **LCK_cycle** option to a startup cycle in BitGen, the DCM's STARTUP_WAIT attribute must be set to TRUE.
3. If the startup sequence is altered (by using the BitGen option), do not place the **LCK_cycle** (wait for the DCM to lock) before the **GTS_cycle** (deassert GTS). Incorrect implementation results in the DCM not locking and an incomplete configuration.

Deskew Adjust

The DESKEW_ADJUST attribute sets the value for a configurable, variable-tap delay element to control the amount of delay added to the DCM feedback path (see Figure 2-4).

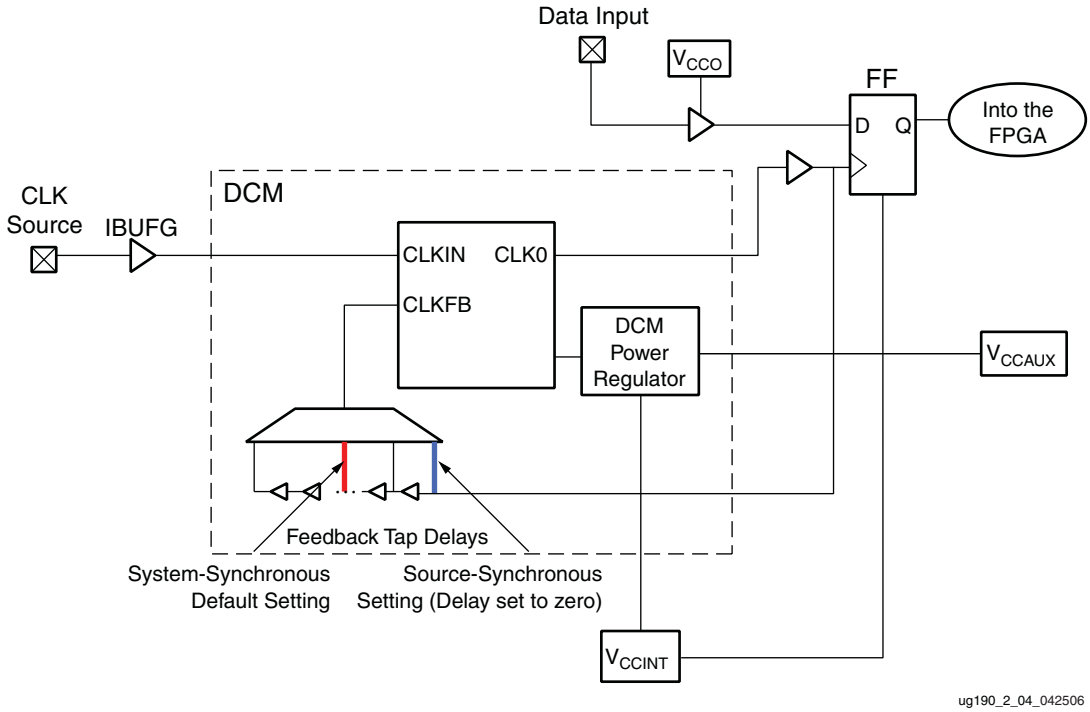


Figure 2-4: **DCM and Feedback Tap-Delay Elements**

This delay element allows adjustment of the effective clock delay between the clock source and CLK0 to guarantee non-positive hold times of IOB input flip-flop in the device. Adding more delay to the DCM feedback path decreases the effective delay of the actual clock path from the FPGA clock input pin to the clock input of any flip-flop. Decreasing the clock delay increases the setup time represented in the input flip-flop, and reduces any positive hold times required. The clock path delay includes the delay through the IBUFG, route, DCM, BUFG, and clock-tree to the destination flip-flop. If the feedback delay equals the clock-path delay, the effective clock-path delay is zero.

System-Synchronous Setting (Default)

By default, the feedback delay is set to system-synchronous mode. The primary timing requirements for a system-synchronous system are non-positive hold times (or minimally positive hold times) and minimal clock-to-out and setup times. Faster clock-to-out and setup times allow shorter system clock periods. Ideally, the purpose of a DLL is to zero-out the clock delay to produce faster clock-to-out and non-positive hold times. The system-synchronous setting (default) for DESKEW_ADJUST configures the feedback delay element to guarantee non-positive hold times for all input IOB registers. The exact delay number added to the feedback path is device size dependent. This is determined by characterization. In the timing report, this is included as timing reduction to input clock path represented by the T_{DCMINO} parameter. As shown in Figure 2-4, the feedback path includes tap delays in the default setting (red line). The pin-to-pin timing parameters (with DCM) on the *Virtex-5 FPGA Data Sheet* reflects the setup/hold and clock-to-out times when the DCM is in system-synchronous mode.

In some situations, the DCM does not add this extra feedback delay, and the DESKEW_ADJUST parameter has no effect. BitGen selects the appropriate DCM Tap settings. These situations include:

- Downstream DCMs when two or more DCMs are cascaded
- DCMs with external feedback
- DCMs with an external CLKIN that does not come from a dedicated clock input pin.

Source-Synchronous Setting

When DESKEW_ADJUST is set to source-synchronous mode, the DCM feedback delay element is set to zero. As shown in Figure 2-4, in source-synchronous mode, the DCM clock feedback delay element is set to minimize the sampling window. This results in a more positive hold time and a longer clock-to-out compared to system-synchronous mode. The source-synchronous switching characteristics section in the *Virtex-5 FPGA Data Sheet* reflects the various timing parameters for the source-synchronous design when the DCM is in source-synchronous mode.

Characteristics of the Deskew Circuit

- Eliminate clock distribution delay by effectively adding one clock period delay.
- Clocks are deskewed to within CLKOUT_PHASE, specified in the *Virtex-5 FPGA Data Sheet*.
- Eliminate on-chip as well as off-chip clock delay.
- No restrictions on the delay in the feedback clock path.
- Requires a continuously running input clock.
- Adapts to a wide range of frequencies. However, once locked to a frequency, large input frequency variations are not tolerated.
- Does not eliminate jitter. The deskew circuit output jitter is the accumulation of input jitter and any added jitter value due to the deskew circuit.
- The completion of configuration can be delayed until after DCM locks to guarantee the system clock is established prior to initiating the device.

Frequency Synthesis

The DCM provides several flexible methods for generating new clock frequencies. Each method has a different operating frequency range and different AC characteristics. The CLK2X and CLK2X180 outputs double the clock frequency. The CLKDV output provides a divided output clock (lower frequency) with division options of 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, and 16.

The DCM also offers fully digital, dedicated frequency-synthesizer outputs CLKFX and its opposite phase CLKFX180. The output frequency can be any function of the input clock frequency described by M ÷ D, where M is the multiplier (numerator) and D is the divisor (denominator).

The frequency synthesized outputs can drive the global-clock routing networks within the device. The well-buffered global-clock distribution network minimizes clock skew due to differences in distance or loading.

Frequency Synthesis Operation

The DCM clock output CLKFX is any M ÷ D factor of the clock input to the DCM. Specifications for M and D, as well as input and output frequency ranges for the frequency synthesizer, are provided in the *Virtex-5 FPGA Data Sheet*.

Only when feedback is provided to the CLKFB input of the DCM is the frequency synthesizer output phase aligned to the clock output, CLK0.

The internal operation of the frequency synthesizer is complex and beyond the scope of this document. As long as the frequency synthesizer is within the range specified in the *Virtex-5 FPGA Data Sheet*, it multiplies the incoming frequencies by the pre-calculated quotient M ÷ D and generates the correct output frequencies.

For example, assume an input frequency of 50 MHz, M = 25, and D = 8 (M and D values do not have common factors and cannot be reduced). The output frequency is 156.25 MHz although separate calculations, 25 x 50 MHz = 1.25 GHz and 50 MHz ÷ 8 = 6.25 MHz, seem to produce separate values outside the range of the input frequency.

Frequency Synthesizer Characteristics

- The frequency synthesizer provides an output frequency equal to the input frequency multiplied by M and divided by D.
- The outputs CLKFX and CLKFX180 always have a 50/50 duty-cycle.
- Smaller M and D values achieve faster lock times. Whenever possible, divide M and D by the largest common factor to get the smallest values. (e.g., if the required CLKFX = 9/6 x CLKIN, instead of using M = 9 and D = 6, use M = 3 and D = 2.)
- When CLKFB is connected, CLKFX is phase aligned with CLK0 every D cycles of CLK0 and every M cycles of CLKFX if M/D is a reduced fraction.

Phase Shifting

The DCM provides coarse and fine-grained phase shifting. For coarse-phase control, the CLK0, CLK90, CLK180, and CLK270 outputs are each phase-shifted by ¼ of the input clock period relative to each other. Similarly, CLK2X180 and CLKFX180 provide a 180° coarse phase shift of CLK2X and CLKFX, respectively. The coarse phase-shifted clocks are produced from the delay lines of the DLL circuit. The phase relationship of these clocks is retained when CLKFB is not connected.

Fine-grained phase shifting uses the CLKOUT_PHASE_SHIFT and PHASE_SHIFT attributes to phase-shift DCM output clocks relative to CLKIN. Since the CLKIN is used as the reference clock, the feedback (CLKFB) connection is required for the phase-shifting circuit to compare the incoming clock with the phase-shifted clock. The rest of this section describes fine-grained phase shifting in the Virtex-5 FPGA DCM.

Phase-Shifting Operation

All nine DCM output clocks are adjusted when fine-grained phase shifting is activated. The phase shift between the rising edges of CLKIN and CLKFB is a specified fraction of the input clock period or a specific amount of DCM_TAP. All other DCM output clocks retain their phase relation to CLK0.

Phase-Shift Range

The allowed phase shift between CLKIN and CLKFB is limited by the phase-shift range. There are two separate phase-shift ranges:

- PHASE_SHIFT attribute range
- FINE_SHIFT_RANGE DCM timing parameter range

In the FIXED, VARIABLE_POSITIVE, and VARIABLE_CENTER phase-shift mode, the PHASE_SHIFT attribute is in the numerator of the following equation.

$$\text{Phase Shift (ns)} = (\text{PHASE\_SHIFT}/256) \times \text{PERIOD}_{\text{CLKIN}}$$

Where $\text{PERIOD}_{\text{CLKIN}}$ denotes the effective CLKIN frequency.

In VARIABLE_CENTER and FIXED modes, the full range of the PHASE_SHIFT attribute is always –255 to +255. In the VARIABLE_POSITIVE mode, the range of the PHASE_SHIFT attribute is 0 to +255.

In the DIRECT phase-shift mode, the PHASE_SHIFT attribute is the multiplication factor in the following equation:

$$\text{Phase Shift (ns)} = \text{PHASE\_SHIFT} \times \text{DCM\_TAP}$$

In DIRECT modes, the full range of the PHASE_SHIFT attribute is 0 to 1023.

FINE_SHIFT_RANGE represents the total delay achievable by the phase-shift delay line. Total delay is a function of the number of delay taps used in the circuit. The absolute range is specified in the DCM Timing Parameters section of the *Virtex-5 FPGA Data Sheet* across process, voltage, and temperature. The different absolute ranges are outlined in this section.

The fixed mode allows the DCM to insert a delay line in the CLKFB or the CLKIN path. This gives access to the +FINE_SHIFT_RANGE when the PHASE_SHIFT attribute is set to a positive value, and –FINE_SHIFT_RANGE when the PHASE_SHIFT attribute is set to a negative value.

Absolute Range (Variable-Center Mode) = ± FINE_SHIFT_RANGE ÷ 2

The variable-center mode allows symmetric, dynamic sweeps from –255/256 to +255/256, by having the DCM set the zero-phase-skew point in the middle of the delay line. This divides the total delay-line range in half.

Absolute Range (Fixed) = ± FINE_SHIFT_RANGE

In the fixed mode, a phase shift is set during configuration in the range of –255/256 to +255/256.

Absolute Range (Variable-Positive and Direct Modes) = + FINE_SHIFT_RANGE

In the variable-positive and direct modes, the phase-shift only operates in the positive range. The DCM sets the zero-phase-skew point at the beginning of the delay line. This produces a full delay line in one direction.

Both the PHASE_SHIFT attribute and the FINE_SHIFT_RANGE parameter need to be considered to determine the limiting range of each application. The "Phase-Shift Examples" section illustrates possible scenarios.

In variable and direct mode, the PHASE_SHIFT value can dynamically increment or decrement as determined by PSINCDEC synchronously to PSCLK, when the PSEN input is active.

Phase-Shift Examples

The following usage examples take both the PHASE_SHIFT attribute and FINE_SHIFT_RANGE into consideration:

- If PERIODCLKIN = 2 × FINE_SHIFT_RANGE, then the PHASE_SHIFT in fixed mode is limited to ±128. In variable-positive mode, PHASE_SHIFT is limited to +128. In variable-center mode the PHASE_SHIFT is limited to ±64.

- If PERIODCLKIN = FINE_SHIFT_RANGE, then the PHASE_SHIFT in variable-positive mode is limited to +255. In fixed and variable-center mode the PHASE_SHIFT is limited to ±255.
- If PERIODCLKIN ≤ FINE_SHIFT_RANGE, then the PHASE_SHIFT in variable-positive mode is limited to +255. In fixed and variable-center mode the PHASE_SHIFT is limited to ±255.
- For all previously described cases, the direct mode is always limited to +1023.

If the phase shift is limited by the FINE_SHIFT_RANGE, use the coarse-grained phase shift to extend the phase-shift range or set DCM_PERFORMANCE_MODE attribute to MAX_RANGE to increase the FINE_SHIFT_RANGE. Figure 2-5 illustrates using CLK90, CLK180, and CLK270 outputs assuming FINE_SHIFT_RANGE = 10 ns.

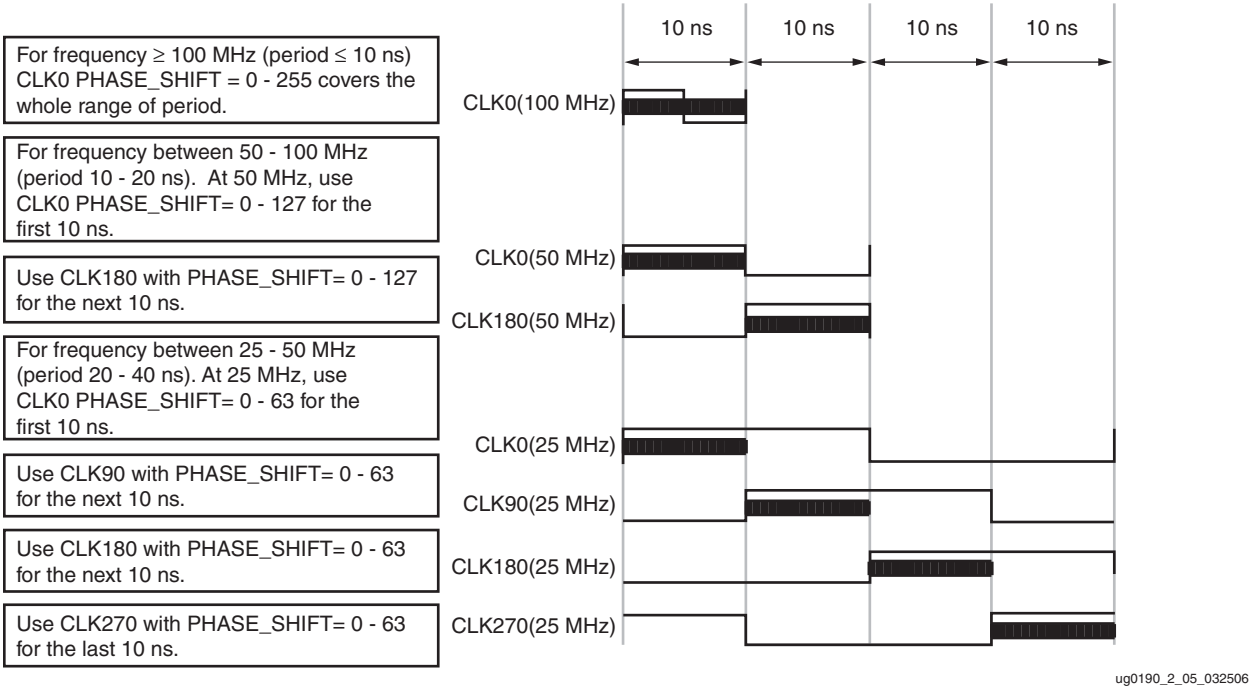


Figure 2-5: **Fixed Phase-Shift Examples**

In variable mode, the phase-shift factor is changed by activating PSEN for one period of PSCLK. At the PSCLK clock cycle where PSEN is activated, the level of PSINCDEC input determines whether the phase-shift increases or decreases. A High on PSINCDEC increases the phase shift, and a Low decreases the phase shift.

After the deskew circuit increments or decrements, the signal PSDONE is asserted High for a single PSCLK cycle. This allows the next change to be performed.

The user interface and the physical implementation are different. The user interface describes the phase shift as a fraction of the clock period (N/256). The physical implementation adds the appropriate number of buffer stages (each DCM_TAP) to the clock delay. The DCM_TAP granularity limits the phase resolution at higher clock frequencies.

All phase-shift modes, with the exception of DIRECT mode, are temperature and voltage adjusted. Hence, a V_{CC} or temperature adjustment does not change the phase shift. The DIRECT phase shift is not temperature or voltage adjusted since it directly controls

DCM_TAP. Changing the ratio of V_{CC}/temperature results in a phase-shift change proportional to the size of the DCM_TAP at the specific voltage and temperature.

Interaction of PSEN, PSINCDEC, PSCLK, and PSDONE

The variable and direct phase-shift modes are controlled by the PSEN, PSINCDEC, PSCLK, and PSDONE ports. In addition, a phase-shift overflow (DO[0]) status indicates when the phase-shift counter has reached the end of the phase-shift delay line or the maximum value (±255 for variable mode, +1023 for direct mode).

After the DCM locks, the initial phase in the VARIABLE_POSITIVE and VARIABLE_CENTER modes is determined by the PHASE_SHIFT value. The initial phase in the DIRECT mode is always 0, regardless of the value specified by the PHASE_SHIFT attribute. The phase of the DCM output clock is incremented/decremented according to the interaction of PSEN, PSINCDEC, PSCLK, and PSDONE from the initial or dynamically reconfigured phase.

PSEN, PSINCDEC, and PSDONE are synchronous to PSCLK. When PSEN is asserted for one PSCLK clock period, a phase-shift increment/decrement is initiated. When PSINCDEC is High, an increment is initiated and when PSINCDEC is Low, a decrement is initiated. Each increment adds to the phase shift of DCM clock outputs by 1/256 of the CLKIN period. Similarly, each decrement decreases the phase shift by 1/256 of the CLKIN period. PSEN must be active for exactly one PSCLK period; otherwise, a single phase-shift increment/decrement is not guaranteed. PSDONE is High for exactly one clock period when the phase shift is complete. The time required to complete a phase-shift operation varies. As a result, PSDONE must be monitored for phase-shift status. Between enabling PSEN and PSDONE is flagged, the DCM output clocks gradually change from their original phase shift to the incremented/decremented phase shift. The completion of the increment or decrement is signaled when PSDONE asserts High. After PSDONE has pulsed High, another increment/decrement can be initiated.

Figure 2-6 illustrates the interaction of phase-shift ports.

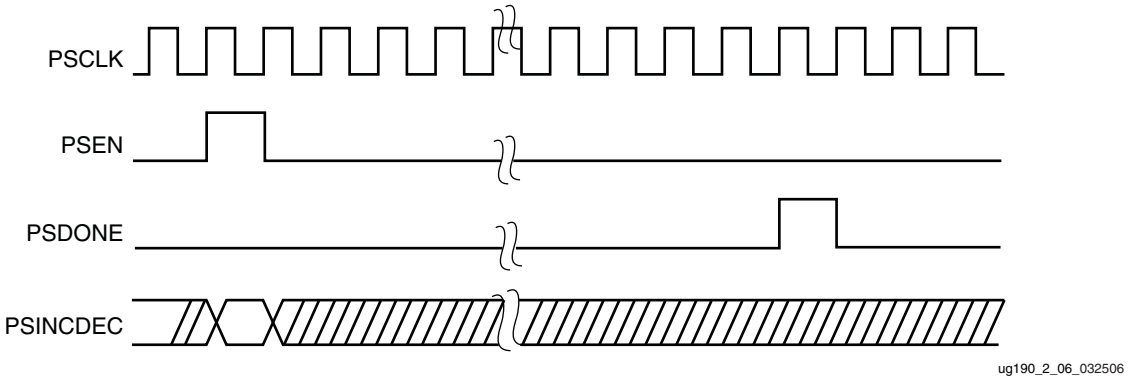


Figure 2-6: **Phase-Shift Timing Diagram**

When PSEN is activated after the phase-shift counter has reached the maximum value of PHASE_SHIFT, the PSDONE is still pulsed High for one PSCLK period some time after the PSEN is activated (as illustrated in Figure 2-6). However, the phase-shift overflow pin, STATUS(0), or DO(0) is High to flag this condition, and no phase adjustment is performed.

Phase-Shift Overflow

The phase-shift overflow (DO[0]) status signal is asserted when either of the following conditions is true:

- The DCM is phase-shifted beyond the allowed phase-shift value. In this case, the phase-shift overflow signal is asserted High when the phase shift is decremented beyond –255 and incremented beyond +255 for VARIABLE_CENTER mode, incremented beyond +255 for VARIABLE_POSITIVE mode, or decremented beyond 0 and incremented beyond 1023 for DIRECT mode.
- The DCM is phase-shifted beyond the absolute range of the phase-shift delay line. In this case, the phase-shift overflow signal is asserted High when the phase-shift in time (ns) exceeds the ±FINE_SHIFT_RANGE/2 in the VARIABLE_CENTER mode, the +FINE_SHIFT_RANGE in the VARIABLE_POSITIVE mode, or exceeds 0 to +FINE_SHIFT_RANGE in the DIRECT mode. The phase-shift overflow signal can toggle once it is asserted. The condition determining if the delay line is exceeded is calibrated dynamically. Therefore, at the boundary of exceeding the delay line, it is possible for the phase-shift overflow signal to assert and deassert without a change in phase shift. Once asserted, it remains asserted for at least 40 CLKIN cycles. If the DCM is operating near the FINE_SHIFT_RANGE limit, do not use the phase-shift overflow signal as a flag to reverse the phase shift direction. When the phase-shift overflow is asserted, deasserted, then asserted again in a short phase shift range, it can falsely reverse the phase shift direction. Instead, use a simple counter to track the phase shift value and reverse the phase shift direction (PSINCDEC) only when the counter reaches a previously determined maximum/minimum phase shift value. For example, if the phase shift must be within 0 to 128, set the counter to toggle PSINCDEC when it reaches 0 or 128.

Phase-Shift Characteristics

- Offers fine-phase adjustment with a resolution of ±1/256 of the clock period (or ± one DCM_TAP, whichever is greater). It can be dynamically changed under user control.
- The phase-shift settings affect all nine DCM outputs.
- V_{CC} and temperature do not affect the phase shift except in direct phase-shift mode.
- In either fixed or variable mode, the phase-shift range can be extended by choosing CLK90, CLK180, or CLK270, rather than CLK0, choosing CLK2X180 rather than CLK2X, or choosing CLKFX180 rather than CLKFX. Even at 25 MHz (40 ns period), the fixed mode coupled with the various CLK phases allows shifting throughout the entire input clock period range.
- MAX_RANGE mode extends the phase-shift range.
- The phase-shifting (DPS) function in the DCM requires the CLKFB for delay adjustment.

  Because CLKFB must be from CLK0, the DLL output is used. The minimum CLKIN frequency for the DPS function is determined by DLL frequency mode.

Dynamic Reconfiguration

The Dynamic Reconfiguration Ports (DRPs) can update the initial DCM settings without reloading a new bit stream to the FPGA. The DRP address mapping changed in Virtex-5 FPGAs. The *Virtex-5 FPGA Configuration Guide* provides more information on using DRPs. Specific to the DCM, DRPs allow dynamic adjustment of the CLKFX_MULTIPLY(M) and CLKFX_DIVIDE(D) values to produce a new CLKFX frequency.

The following steps are required when using DRPs to load new M and D values:

- Subtract the desired M and D values by one. For example, if the desired M/D = 9/4, then load M/D = 8/3.
- Hold DCM in reset (assert RST signal) and release it after the new M and D values are written. The CLKFX outputs can be used after LOCKED is asserted High again.
- Read DADDR0 to restore the default status on D0.
- Release RST.

Connecting DCMs to Other Clock Resources in Virtex-5 Devices

Most DCM functions require connection to dedicated clock resources, including dedicated clock I/O (IBUFG), clock buffers (BUFGCTRLs), and PLLs. These clock resources are located in the center column of the Virtex-5 devices. This section provides guidelines on connecting the DCM to dedicated clock resources.

IBUFG to DCM

Virtex-5 devices contain 20 clock inputs. These clock inputs are accessible by instantiating the IBUFG. Each top and bottom half of a Virtex-5 device contains 10 IBUFGs. Any of the IBUFG in top or bottom half of the Virtex-5 device can drive the clock input pins (CLKIN, CLKFB, PSCLK, or DCLK) of a DCM located in the same top/bottom half of the device.

DCM to BUFGCTRL

Any DCM clock output can drive any BUFGCTRL input in the same top/bottom half of the device. There are no restrictions on how many DCM outputs can be used simultaneously.

BUFGCTRL to DCM

Any BUFGCTRL can drive any DCM in the Virtex-5 devices. However, only up to ten dedicated clock routing resources exist in a particular clock region. Since the clock routing is accessed via the BUFGCTRL outputs, this indirectly limits the BUFGCTRL to DCM connection. If ten BUFGCTRL outputs are already accessing a clock region, and a DCM is in that region, then no additional BUFGCTRL can be used in that region, including a connection to the CLKFB pin of the DCM.

PLL To and From DCM

Figure 2-7 summarizes the dedicated connection between the DCM and the PLL in the same CMT block. The PLL can drive either DCM in the same CMT block using a dedicated connection. Similarly, the DCM can drive the PLL within the same CMT block with a dedicated connection. There is no BUFGCTRL required between the PLL and the DCM.

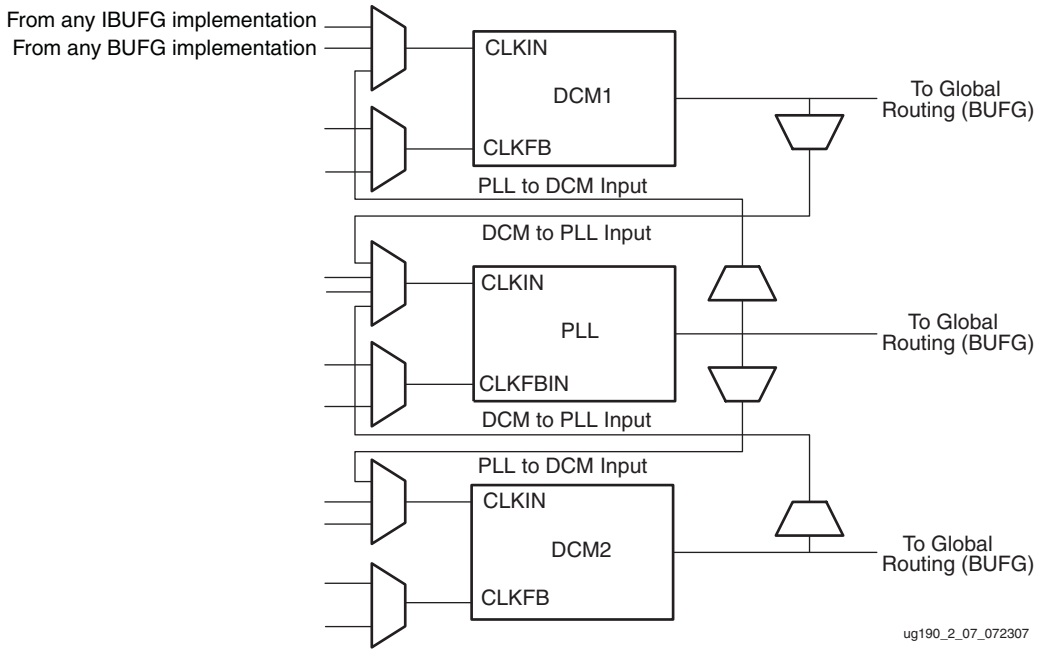


Figure 2-7: **DCM and PLL Connection in Same CMT Block**

DCM To and From PMCD

The PMCD block is not available in the Virtex-5 devices. However, a limited retargeting using the PLL is possible. Refer to "PLL in Virtex-4 FPGA PMCD Legacy Mode" in Chapter 3 for more information.

Application Examples

The Virtex-5 FPGA DCM can be used in a variety of creative and useful applications. The following examples show some of the more common applications.

Standard Usage

The circuit in Figure 2-8 shows DCM_BASE implemented with internal feedback and access to RST and LOCKED pins. This example shows the simplest use case for a DCM.

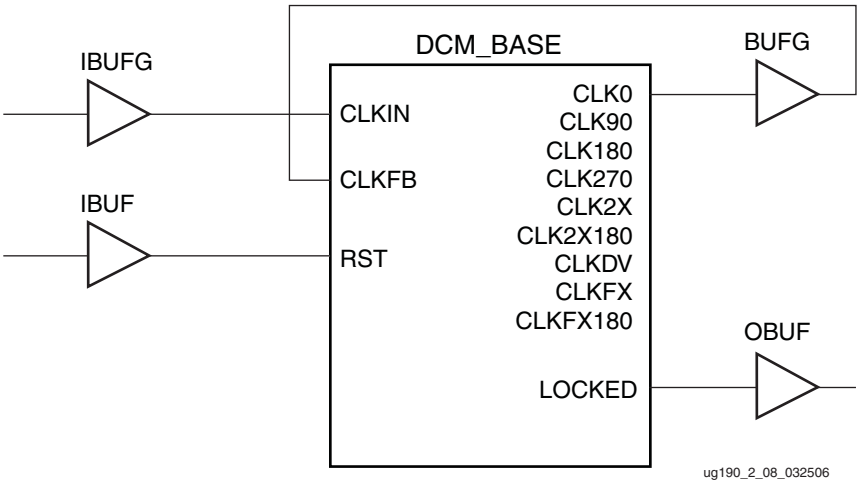


Figure 2-8: **Standard Usage**

Board-Level Clock Generation

The board-level clock generation example in Figure 2-9 illustrates how to use a DCM to generate output clocks for other components on the board. This clock can then be used to interface with other devices. In this example, a DDR register is used with its inputs connected to GND and V_{CC}. Because the output of the DCM is routed to BUFG, the clock stays within global routing until it reaches the output register. The quality of the clock is maintained.

The board-level clock generation example in Figure 2-10, with internal feedback, illustrates the clock generation for a forwarded clock on the board.

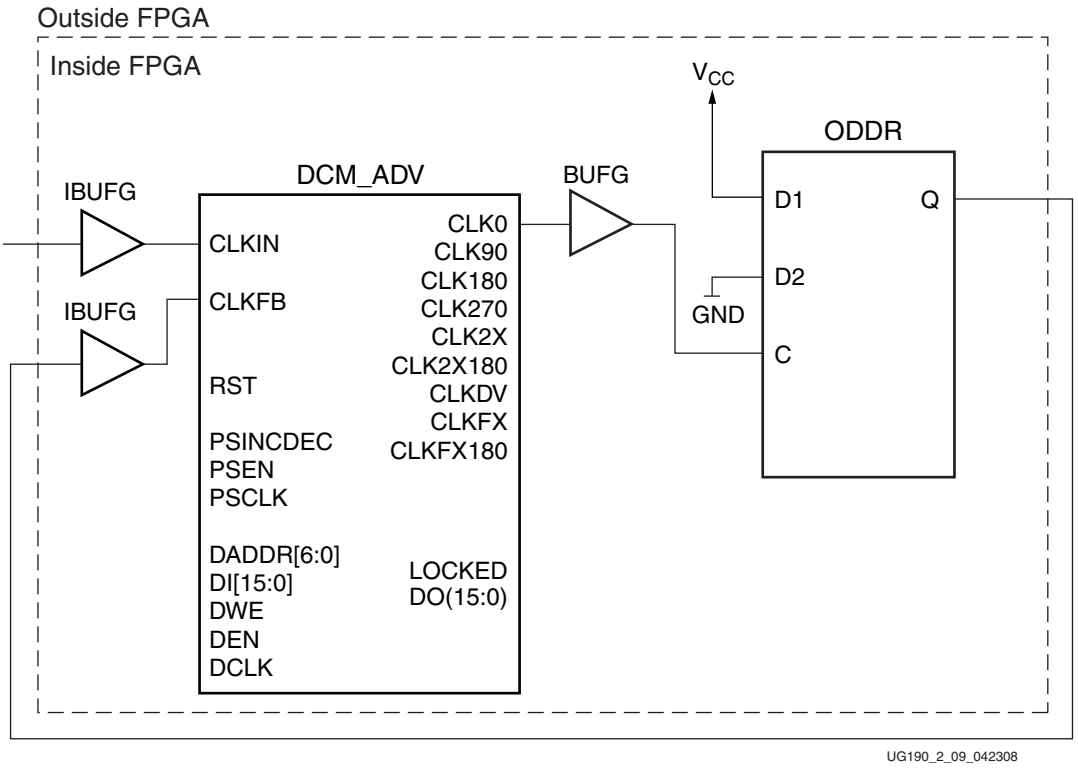


Figure 2-9: **Board-Level Clock Using DDR Register with External Feedback**

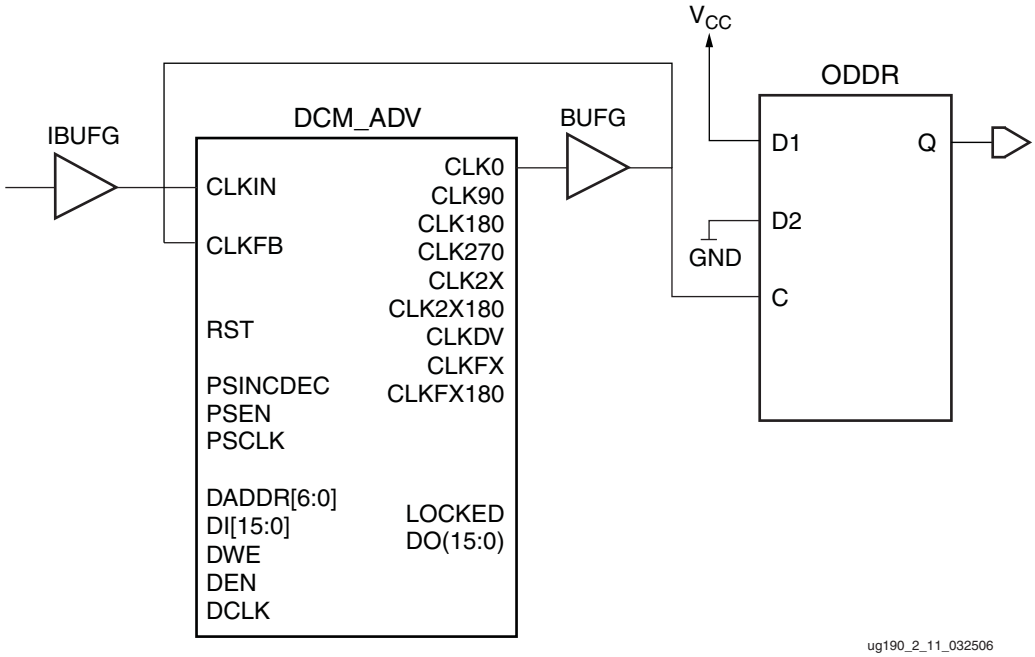


Figure 2-10: **Board-Level Clock with Internal Feedback**

Board Deskew with Internal Deskew

Some applications require board deskew with internal deskew to interface with other devices. These applications can be implemented using two or more DCM. The circuit shown in Figure 2-11 can be used to deskew a system clock between multiple Virtex devices in the same system.

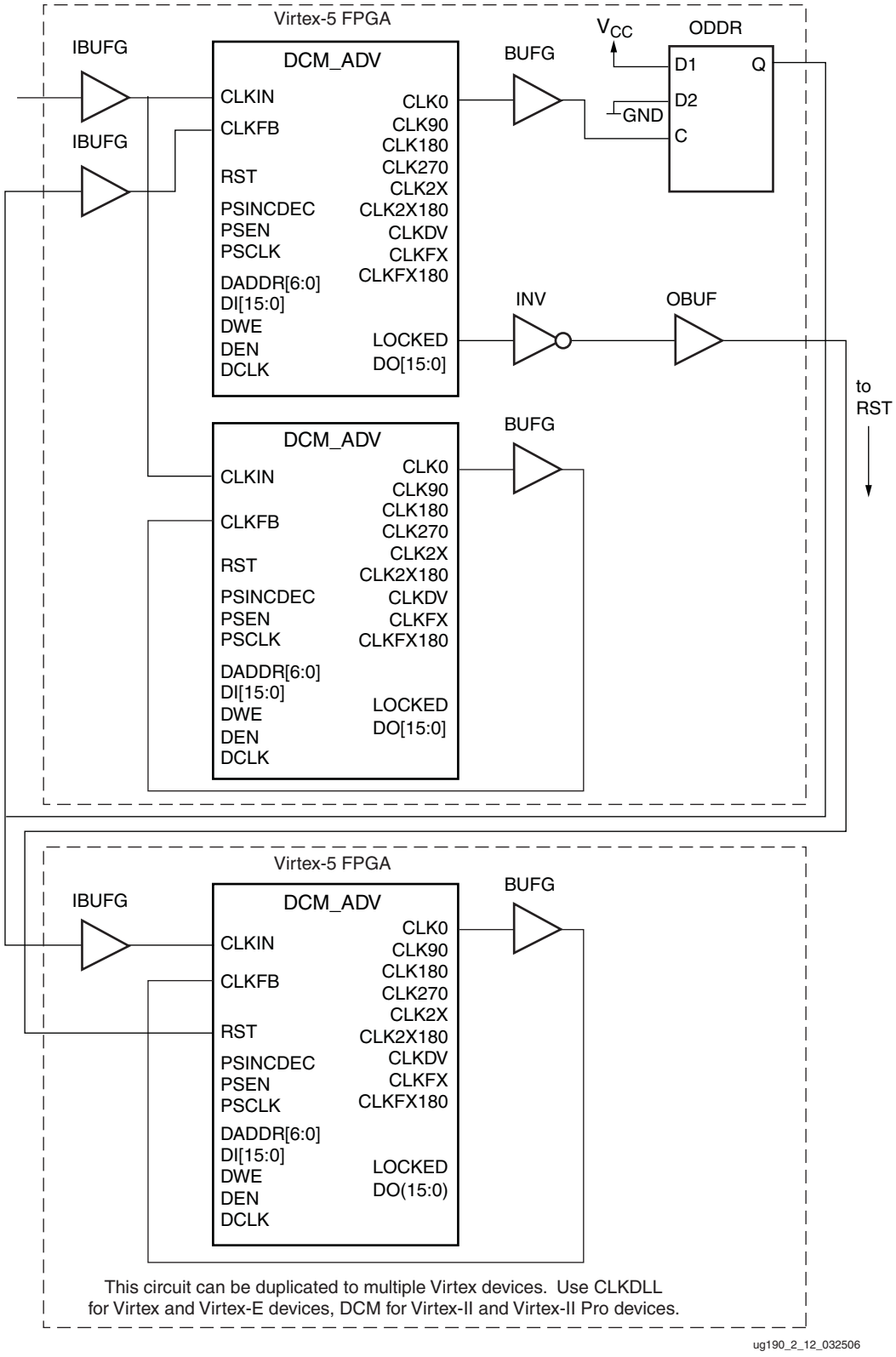


Figure 2-11: **Board Deskew with Internal Deskew Interfacing to Other Virtex Devices**

The example in Figure 2-12 shows an interface from Virtex-5 FPGAs to components other than Virtex FPGAs.

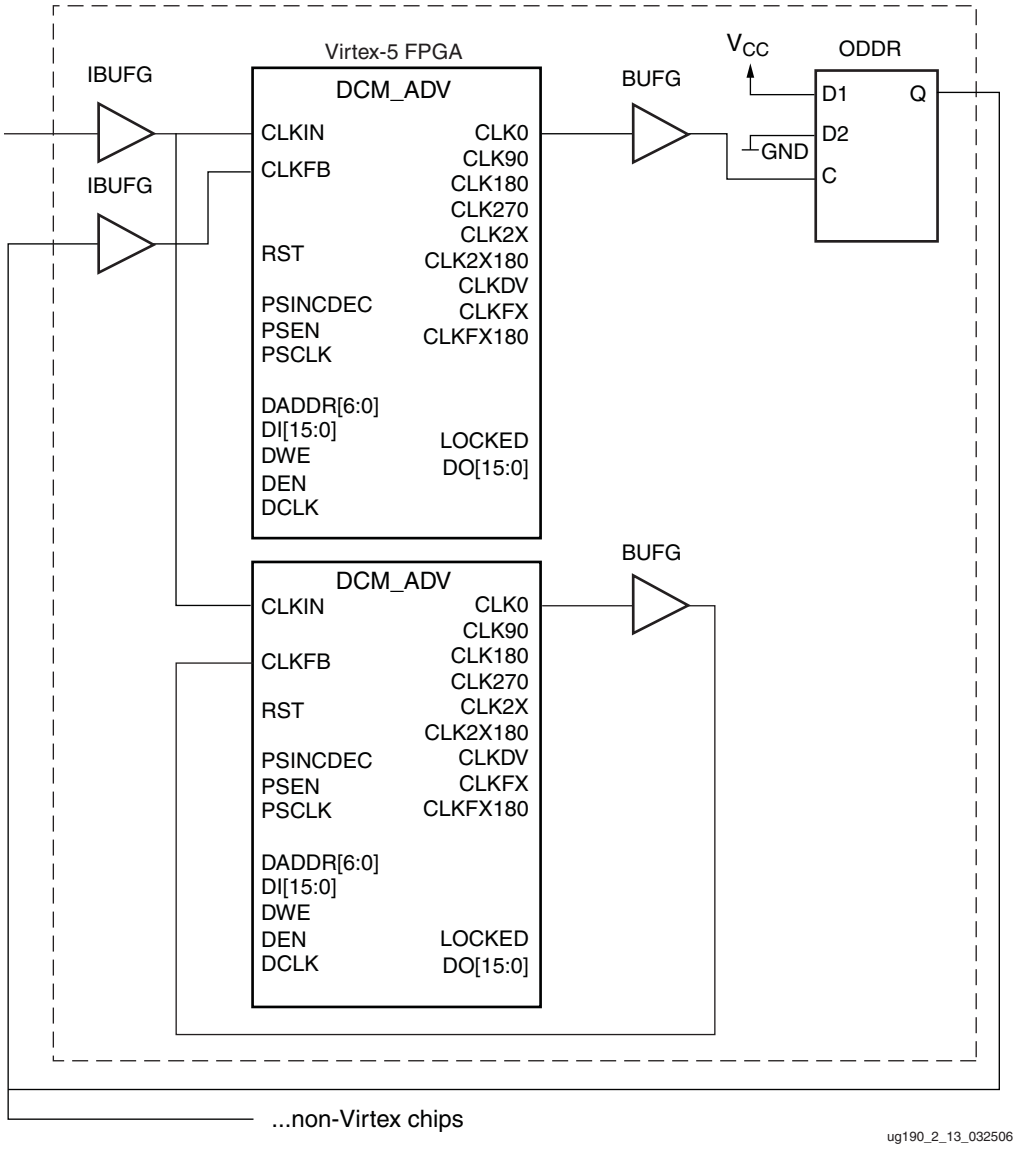


Figure 2-12: **Board Deskew with Internal Deskew Interfacing to Other Components**

Clock Switching Between Two DCMs

Figure 2-13 illustrates switching between two clocks from two DCMs while keeping both DCMs locked.

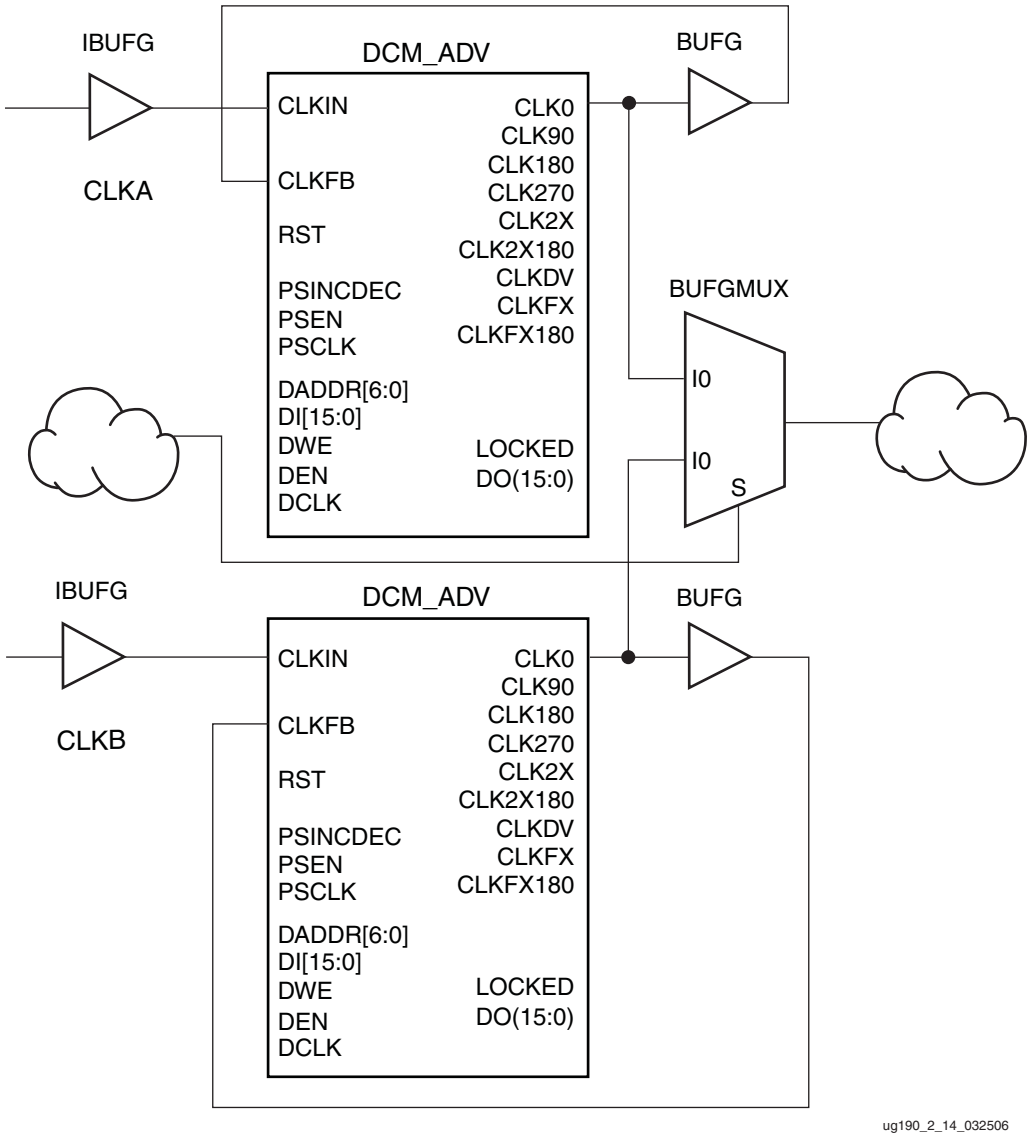


Figure 2-13: **Clock Switching Between Two DCMs**

DCM with PLL

The PLL can be used to drive the DCM to reduce the source clock's incoming jitter before inputting DCM. This setup reduces the source clock jitter while enabling user access to all available DCM clock outputs. Figure 2-14 illustrates the PLL driving a DCM within the same CMT block using the dedicated routing resource (without BUFG).

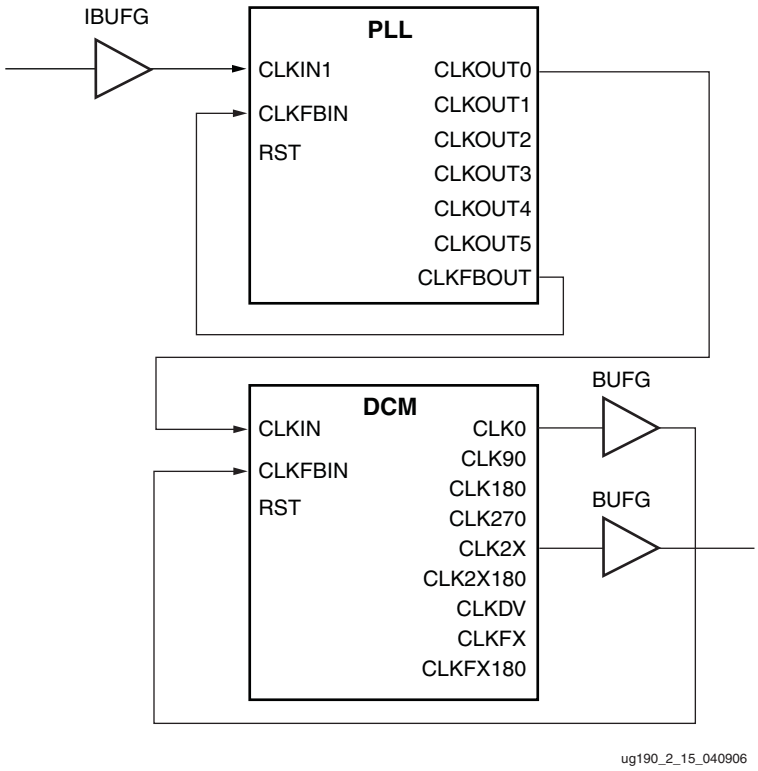


Figure 2-14: **PLL Driving DCM**

It is also possible to use the DCM to drive a PLL. This setup reduces the overall jitter of both the source clock and the DCM clock output. In this case, only up to two of the DCM output clocks can drive the PLL. Therefore, only up to two DCM clocks can access the PLL and benefit from the reduced jitter.

Figure 2-15 and Figure 2-16 illustrate two scenarios of the DCM driving a PLL. Figure 2-15 illustrates the direct connection between DCM and PLL within a CMT. Only one DCM output can drive PLL using the direct connection within a CMT without routing through a global buffer (BUFG). The DCM and PLL can be within the same or different CMTs. Figure 2-16 illustrates two DCMs driving a PLL. In this case, BUFG must also be inserted between the DCM clocks driving the PLL input clocks. The DCM and PLL can be within the same or different CMTs. Refer to Chapter 3, "Phase-Locked Loops (PLLs)," for more information on PLLs.

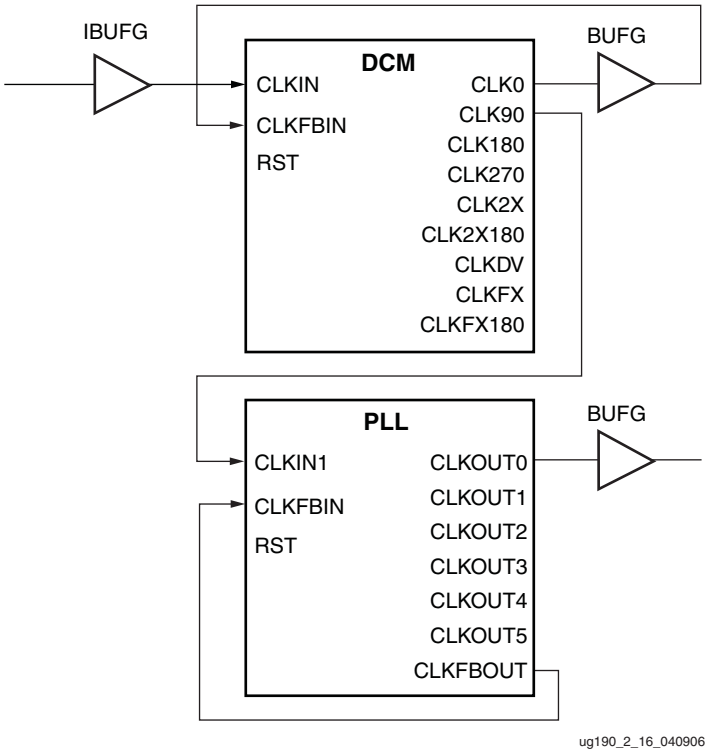


Figure 2-15: **Direct Connection between DCM and PLL**

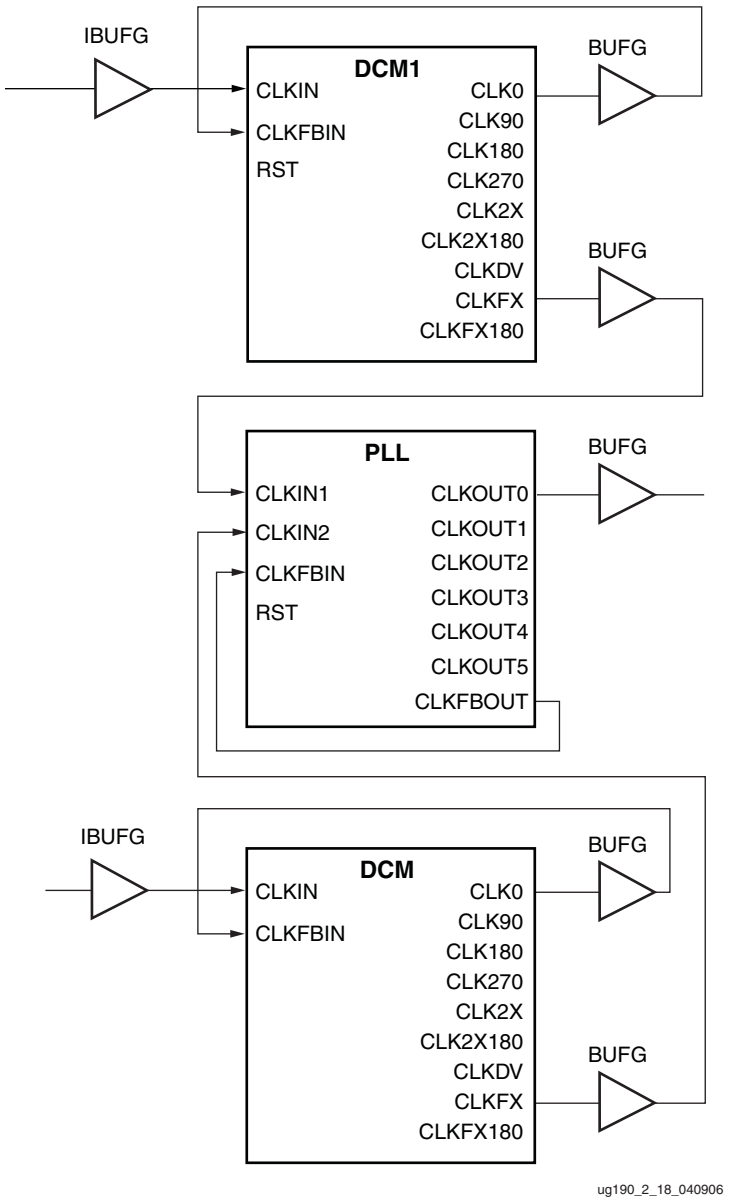


Figure 2-16: **Two DCMs Driving a PLL**

VHDL and Verilog Templates, and the Clocking Wizard

VHDL and Verilog instantiation templates are available in the Libraries Guide for all primitives. In addition, VHDL and Verilog files are generated by the Clocking Wizard in the ISE software. The Clocking Wizard sets appropriate DCM attributes, input/output clocks, and buffers for general use cases.

DCM Timing Models

The following timing diagrams describe the behavior of the DCM clock outputs under four different conditions:

1. Reset/Lock
2. Fixed-Phase Shifting
3. Variable-Phase Shifting
4. Status Flags

Reset/Lock

In Figure 2-17, the DCM is already locked. After the reset signal is applied, all output clocks are stabilized to the desired values, and the LOCKED signal is asserted.

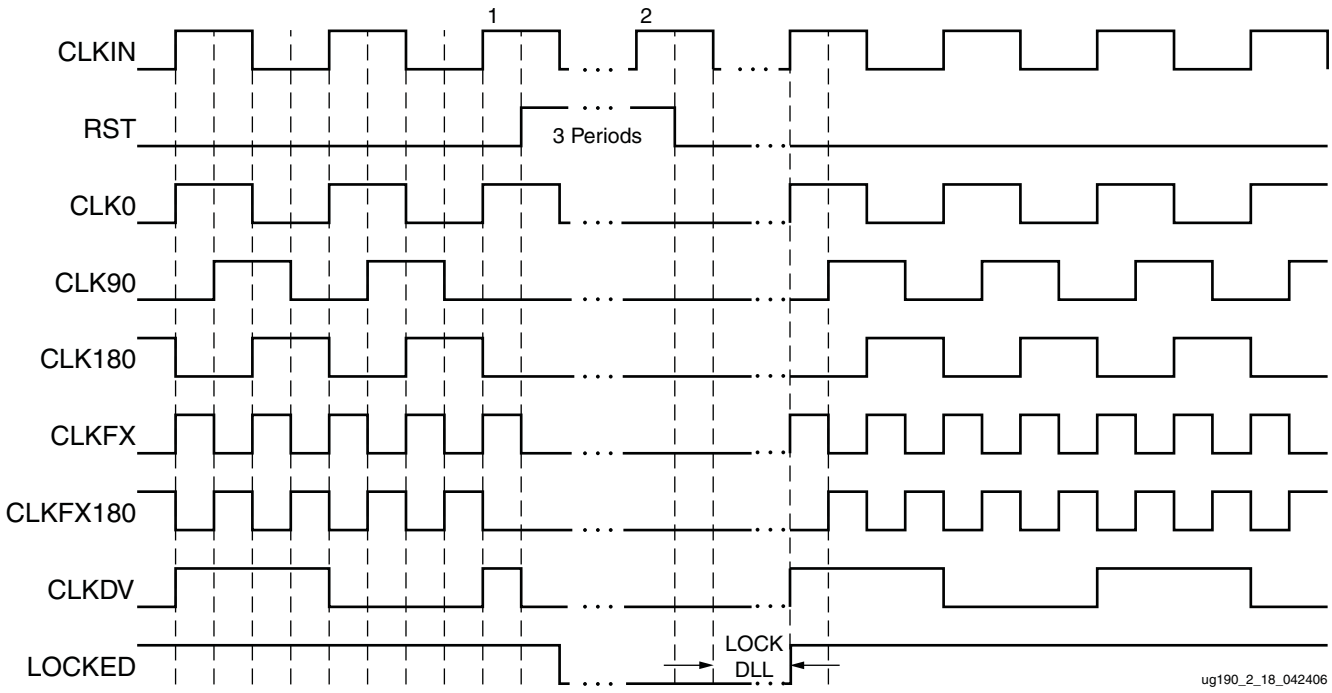


Figure 2-17: **RESET/LOCK Example**

- Prior to Clock Event 1

  Prior to clock event 1, the DCM is locked. All clock outputs are in phase with the correct frequency and behavior.

- Clock Event 1

  Some time after clock event 1 the reset signal is asserted at the (RST) pin. While reset is asserted, all clock outputs become a logic zero. The reset signal is an asynchronous reset. Note: the diagram is not shown to scale. For the DCM to operate properly, the reset signal must be asserted for at least three CLKIN periods.

- Clock Event 2

  Clock event 2 occurs a few cycles after reset is asserted and deasserted. At clock event 2, the lock process begins. At time LOCK_DLL, after clock event 2, if no fixed phase

shift was selected then all clock outputs are stable and in phase. LOCKED is also asserted to signal completion.

Fixed-Phase Shifting

In Figure 2-18, the DCM outputs the correct frequency. However, the clock outputs are not in phase with the desired clock phase. The clock outputs are phase-shifted to appear sometime later than the input clock, and the LOCKED signal is asserted.

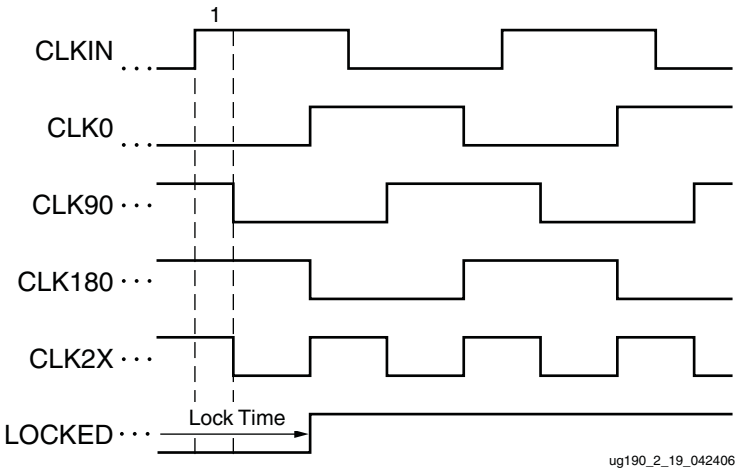


Figure 2-18: **Phase Shift Example: Fixed**

- Clock Event 1

  Clock event 1 appears after the desired phase shifts are applied to the DCM. In this example, the shifts are positive shifts. CLK0 and CLK2X are no longer aligned to CLKIN. However, CLK0, and CLK2X are aligned to each other, while CLK90 and CLK180 remain as 90° and 180° versions of CLK0. The LOCK signal is also asserted once the clock outputs are ready.

Variable-Phase Shifting

In Figure 2-19, the CLK0 output is phase-shifted using the dynamic phase-shift adjustments in the synchronous user interface. The PSDONE signal is asserted for one cycle when the DCM completes one phase adjustment. After PSDONE is deasserted, PSEN can be asserted again, allowing an additional phase shift to occur.

As shown in Figure 2-19, all the variable-phase shift control and status signals are synchronous to the rising edge of PSCLK.

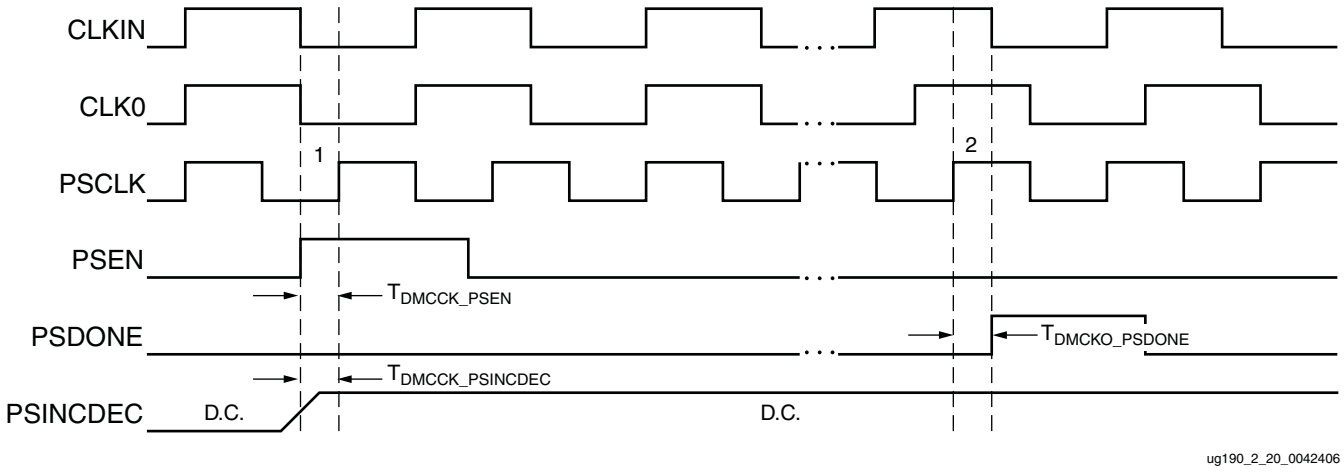


Figure 2-19: **Phase Shift Example: Variable**

- Clock Event 1

  At T_{DMCCK_PSEN}, before clock event 1, PSEN is asserted. PSEN must be active for exactly one clock period; otherwise, a single increment/decrement of phase shift is not guaranteed. Also, the PSINCDEC value at $T_{DMCCK\_PSINCDEC}$, before clock event 1, determines whether it is an increment (logic High) or a decrement (logic Low).

- Clock Event 2

  At T_{DMCKO_PSDONE}, after clock event 2, PSDONE is asserted to indicate one increment or decrement of the DCM outputs. PSDONE is High for exactly one clock period when the phase shift is complete. The time required for a complete phase shift varies. As a result, PSDONE must be monitored for phase-shift status.

Status Flags

The example in Figure 2-20 shows the behavior of the status flags in the event of a phase-shift overflow and CLKIN/CLKFB/CLKFX failure.

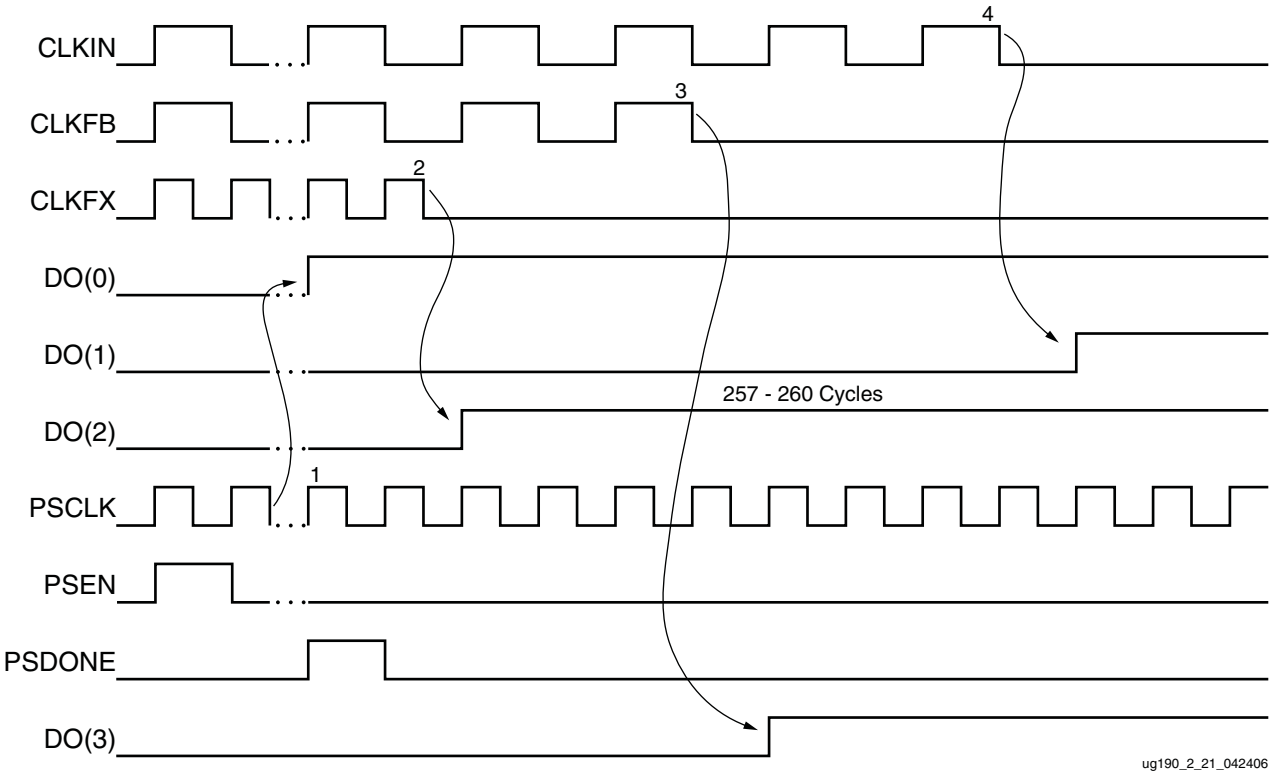


Figure 2-20: **Status Flags Example**

- Clock Event 1

  Prior to the beginning of this timing diagram, CLK0 (not shown) is already phase-shifted at its maximum value. At clock event 1, PSDONE is asserted. However, since the DCM has reached its maximum phase-shift capability no phase adjustment is performed. Instead, the phase-shift overflow status pin DO(0) is asserted to indicate this condition.

- Clock Event 2

  The CLKFX output stops toggling. Within 257 to 260 clock cycles after this event, the CLKFX stopped status DO(2) is asserted to indicate that the CLKFX output stops toggling.

- Clock Event 3

  The CLKFB input stops toggling. Within 257 to 260 clock cycles after this event, the CLKFB stopped status DO(3) is asserted to indicate that the CLKFB output stops toggling.

- Clock Event 4

  The CLKIN input stops toggling. Within 9 clock cycles after this event, DO(1) is asserted to indicate that the CLKIN output stops toggling.

Legacy Support

The Virtex-5 FPGA DCMs (DCM_BASE and DCM_ADV) have exactly the same port names as the Virtex-4 FPGA DCMs. However, the DRP address mapping has changed. Refer to the *Virtex-5 FPGA Configuration Guide* for more information.

The Virtex-5 device supports the Virtex-II family and Virtex-II Pro FPGA DCM primitives. The mapping of Virtex-II or Virtex-II Pro FPGA DCMs to Virtex-5 FPGA DCM_ADVs are as follows:

- CLKIN, CLKFB, PSCLK, PSINDEC, PSEN, RST, CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, CLKFX, CLKFX180, CLKDV, PSDONE, LOCKED of Virtex-5 FPGA primitives (DCM_BASE/DCM_ADV) map to the same corresponding pins of a Virtex-II or Virtex-II Pro FPGA DCM.
- Dynamic reconfiguration pins of Virtex-5 FPGA DCM_ADV are not accessible when a Virtex-II or Virtex-II Pro FPGA DCM is used, except for DO[15:0].
- DO[7:0] pins of Virtex-5 FPGA DCM_ADV map to Status[7:0] of the Virtex-II or Virtex-II Pro FPGA DCMs. DO[15:8] of DCM_ADV are not available when using Virtex-II or Virtex-II Pro FPGA DCMs.

XILINX®

Chapter 3

Phase-Locked Loops (PLLs)

Introduction

The clock management tile (CMT) in Virtex-5 FPGAs includes two DCMs and one PLL. There are dedicated routes within a CMT to couple together various components. Each block within the tile can be treated separately, however, there exists a dedicated routing between blocks creating restrictions on certain connections. Using these dedicated routes frees up global resources for other design elements. Additionally, the use of local routes within the CMT provides an improved clock path because the route is handled locally, reducing chances for noise coupling.

The CMT diagram (Figure 3-1) shows a high-level view of the connection between the various clock input sources and the DCM-to-PLL and PLL-to-DCM dedicated routing. The six (total) PLL output clocks are muxed into a single clock signal for use as a reference clock to the DCMs. Two output clocks from the PLL can drive the DCMs. These two clocks are 100% independent. PLL output clock 0 could drive DCM1 while PLL output clock 1 could drive DCM2. Each DCM output can be muxed into a single clock signal for use as a reference clock to the PLL. Only one DCM can be used as the reference clock to the PLL at any given time. A DCM can not be inserted in the feedback path of the PLL. Both the PLLs or DCMs of a CMT can be used separately as stand-alone functions. The outputs from the PLL are not spread spectrum.

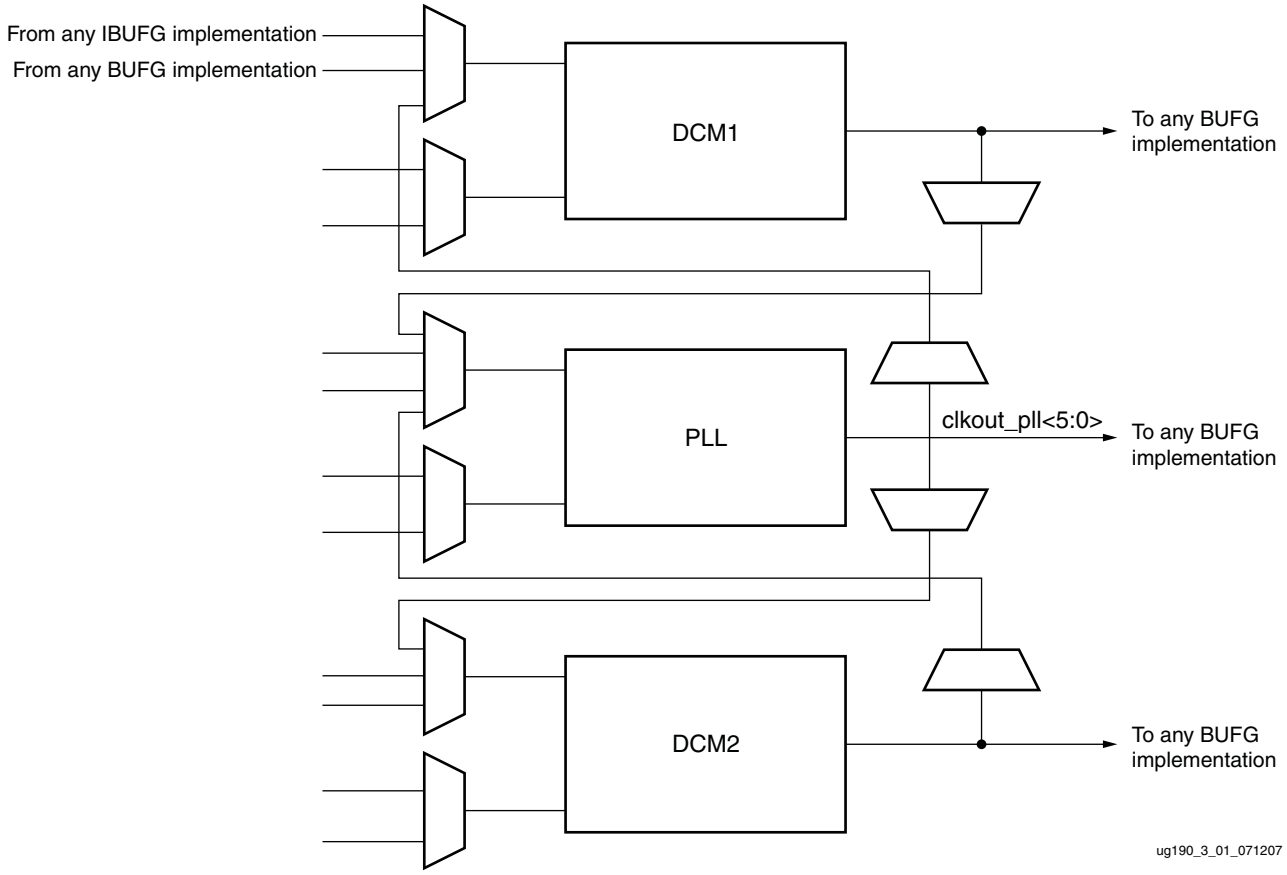


Figure 3-1: **Block Diagram of the Virtex-5 FPGA CMT**

Phase Lock Loop (PLL)

Virtex-5 devices contain up to six CMT tiles. The PLLs main purpose is to serve as a frequency synthesizer for a wide range of frequencies, and to serve as a jitter filter for either external or internal clocks in conjunction with the DCMs of the CMT.

The PLL block diagram shown in Figure 3-2 provides a general overview of the PLL components.

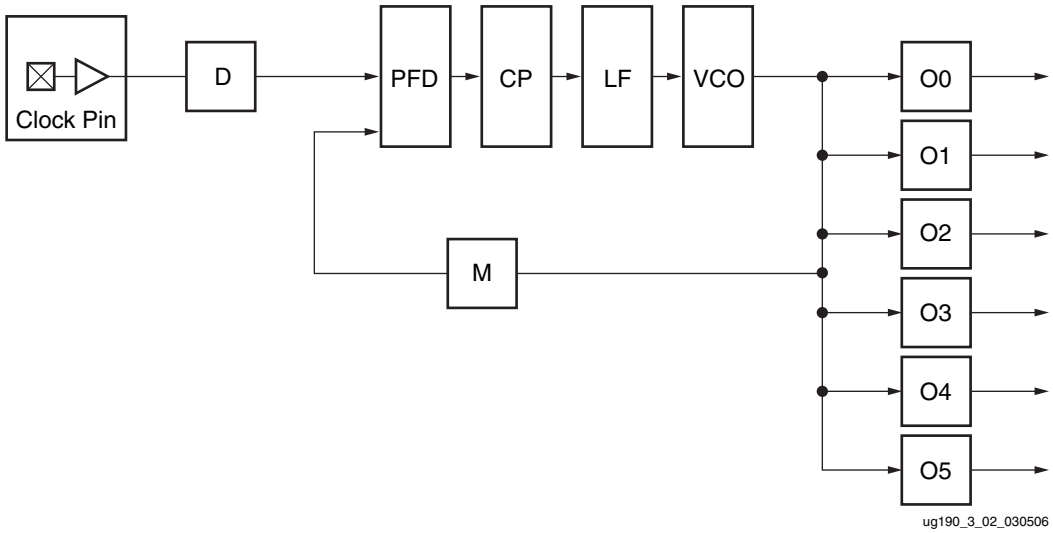


Figure 3-2: **Block Diagram of the Virtex-5 FPGA PLL**

Input muxes select the reference and feedback clocks from either the IBUFG, BUFG, IBUF, PLL outputs, or one of the DCMs. Each clock input has a programmable counter D. The Phase-Frequency Detector (PFD) compares both phase and frequency of the input (reference) clock and the feedback clock. Only the rising edges are considered because as long as a minimum High/Low pulse is maintained, the duty cycle is not important. The PFD is used to generate a signal proportional to the phase and frequency between the two clocks. This signal drives the Charge Pump (CP) and Loop Filter (LF) to generate a reference voltage to the VCO. The PFD produces an up or down signal to the charge pump and loop filter to determine whether the VCO should operate at a higher or lower frequency. When VCO operates at too high of a frequency, the PFD activates a down signal, causing the control voltage to be reduced decreasing the VCO operating frequency. When the VCO operates at too low of a frequency, an up signal will increase voltage. The VCO produces eight output phases. Each output phase can be selected as the reference clock to the output counters (Figure 3-3.) Each counter can be independently programmed for a given customer design. A special counter, M, is also provided. This counter controls the feedback clock of the PLL allowing a wide range of frequency synthesis.

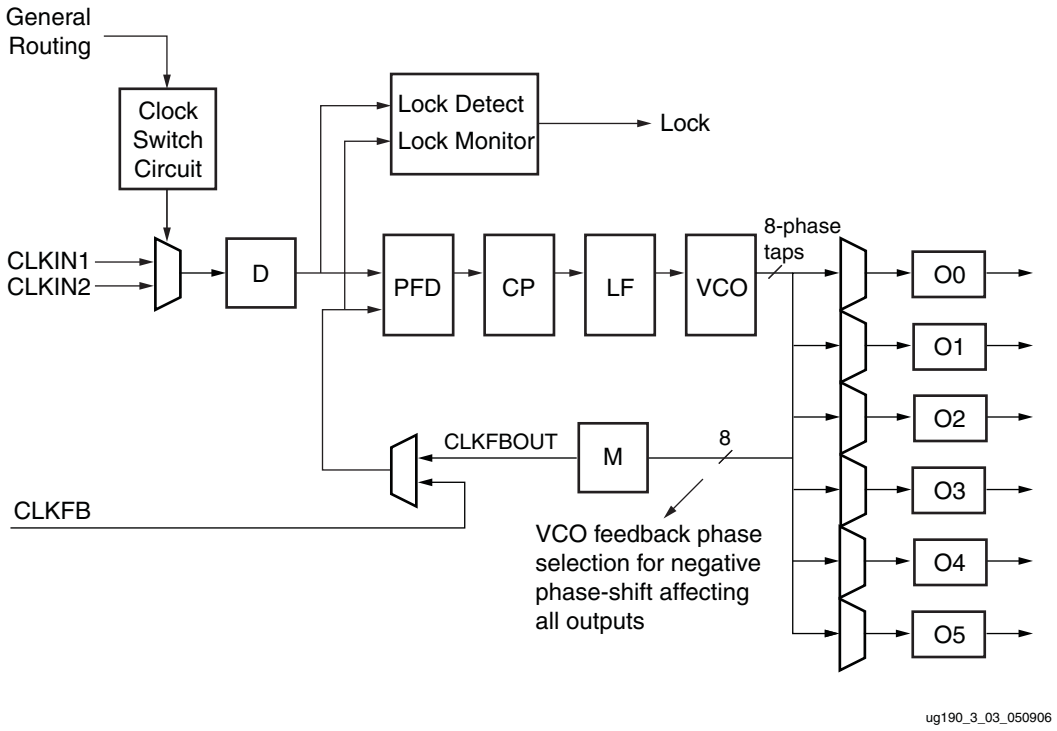


Figure 3-3: **Detailed PLL Block Diagram**

General Usage Description

PLL Primitives

The two Virtex-5 FPGA PLL primitives, PLL_BASE and PLL_ADV, are shown in Figure 3-4.

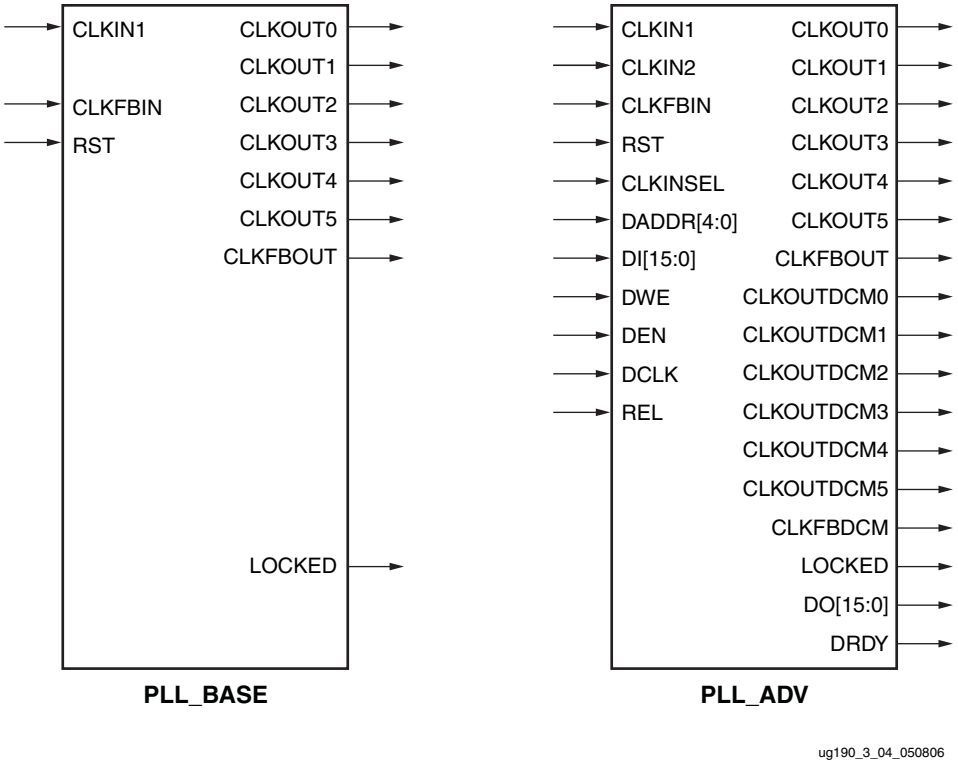


Figure 3-4: **PLL Primitives**

PLL_BASE Primitive

The PLL_BASE primitive provides access to the most frequently used features of a stand alone PLL. Clock deskew, frequency synthesis, coarse phase shifting, and duty cycle programming are available to use with the PLL_BASE. The ports are listed in Table 3-1.

Table 3-1: **PLL_BASE Ports**

Description	Port
Clock Input	CLKIN, CLKFBIN
Control Inputs	RST
Clock Output	CLKOUT0 to CLKOUT5, CLKFBOUT
Status and Data Outputs	LOCKED

PLL_ADV Primitive

The PLL_ADV primitive provides access to all PLL_BASE features plus additional ports for clock switching, connectivity to DCMs in the same CMT, and access to the Dynamic Reconfiguration Port (DRP). The ports are listed in Table 3-2.

Table 3-2: **PLL_ADV Ports**

Description	Port
Clock Input	CLKIN1, CLKIN2, CLKFBIN, DCLK
Control and Data Input	RST, CLKINSEL, DWE, DEN, DADDR, DI, REL(1)
Clock Output	CLKOUT0 to CLKOUT5, CLKFBOUT, CLKOUTDCM0 to CLKOUTDCM5, CLKFBDCM
Status and Data Output	LOCKED, DO, DRDY

Notes:

1. REL is used in PMCD mode only. In PLL mode, leave REL unconnected or tied Low.

The Virtex-5 FPGA PLL is a mixed signal block designed to support clock network deskew, frequency synthesis, and jitter reduction. These three modes of operation are discussed in more detail within this section. The Voltage Controlled Oscillator (VCO) operating frequency can be determined by using the following relationship:

$$F_{VCO} = F_{CLKIN} \times \frac{M}{D} \quad \text{Equation 3-1}$$

$$F_{OUT} = F_{CLKIN} \times \frac{M}{DO} \quad \text{Equation 3-2}$$

where the M, D, and O counters are shown in Figure 3-3.

The six "O" counters can be independently programmed. For example, O0 can be programmed to do a divide-by-two while O1 is programmed for a divide by three. The only constraint is that the VCO operating frequency must be the same for all the output counters since a single VCO drives all the counters.

Clock Network Deskew

In many cases, designers do not want to incur the delay on a clock network in their I/O timing budget therefore they use a PLL or DLL to compensate for the clock network delay. Virtex-5 FPGA PLLs support this feature. A clock output matching the reference clock CLKIN frequency (usually CLKFBOUT) is connected to a BUFG and fed back to the CLKFBIN feedback pin of the PLL. The remaining outputs can still be used to divide the clock down for additionally synthesized frequencies. In this case, all output clocks have a defined phase relationship to the input reference clock.

Frequency Synthesis Only

The PLLs can also be used for stand alone frequency synthesis. In this application, the PLL can not be used to deskew a clock network, but rather generate an output clock frequency for other blocks. In this mode, the PLL feedback path should be set to INTERNAL since it keeps all the routing local and should minimize the jitter. Figure 3-5 shows the PLL configured as a frequency synthesizer. In this example, an external 33 MHz reference clock is available. The reference clock can be a crystal oscillator or the output of another PLL. Setting the M counter to 16 makes the VCO oscillate at 533 MHz (33.333 MHz x 16). The six

PLL outputs are programmed to provide a 533 MHz PowerPC® processor clock, a 266 MHz PowerPC processor gasket clock, a 178 MHz clock, a 133 MHz memory interface clock, a 66 MHz PCI™ clock, and a 33 MHz PCI clock. In this example, there are no required phase relationships between the reference clock and the output clocks, but there are required relationships between the output clocks.

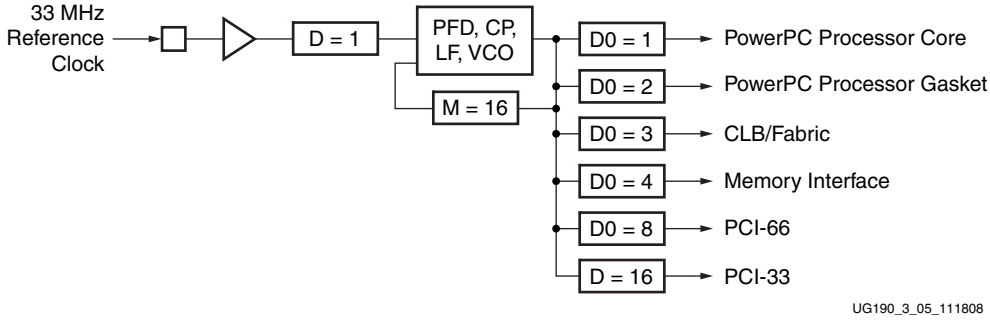


Figure 3-5: **PLL as a Frequency Synthesizer**

Jitter Filter

PLLs always reduce the jitter inherent on a reference clock. The PLL can be instantiated as a standalone function to simply support filtering jitter from an external clock before it is driven into the another block (including the DCM). As a jitter filter, it is usually assumed that the PLL acts as a buffer and regenerates the input frequency on the output (e.g., F_{IN} = 100 MHz, F_{OUT} = 100 MHz). In general, greater jitter filtering is possible by using the PLL attribute BANDWIDTH set to Low. Setting the BANDWIDTH to Low can incur an increase in the static offset of the PLL.

Limitations

The PLL has some restrictions that must be adhered to. These are summarized in the PLL electrical specification in the *Virtex-5 FPGA Data Sheet*. In general, the major limitations are VCO operation range, input frequency, duty cycle programmability, and phase shift.

VCO Operating Range

The minimum and maximum VCO operating frequencies are defined in the electrical specification of the *Virtex-5 FPGA Data Sheet*. These values can also be extracted from the speed specification.

Minimum and Maximum Input Frequency

The minimum and maximum CLKIN input frequency are defined in the electrical specification of the *Virtex-5 FPGA Data Sheet*.

Duty Cycle Programmability

Only discrete duty cycles are possible given a VCO operating frequency. The counter settings to determine the output duty cycle is further discussed under "Counter Control."

Phase Shift

In many cases, there needs to be a phase shift between clocks. The phase shift resolution in time units is defined as: PS = 1/8 F_{VCO} or D/8MF_{IN} since the VCO can provide eight phase shifted clocks at 45° each.

The higher the VCO frequency, the smaller the phase shift resolution. Since the VCO has a distinct operating range, it is possible to bound the phase shift resolution using from 1/8 F_{VCO_MIN} to 1/8 F_{VCO_MAX}.

Each output counter is individually programmable allowing each counter to have a different phase shift based on the output frequency of the VCO.

Note: Phase shifts other than 45° are possible. A finer phase shift resolution depends on the output duty cycle and 0 value. Consult the architecture wizard for other phase-shift settings.

PLL Programming

Programming of the PLL must follow a set flow to ensure configuration that guarantees stability and performance. This section describes how to program the PLL based on certain design requirements. A design can be implement in two ways, directly through the GUI interface (the PLL Wizard) or directly implementing the PLL through instantiation. Regardless of the method selected, the following information is necessary to program the PLL:

- Reference clock period
- Output clock frequencies (up to six maximum)
- Output clock duty cycle (default is 50%)
- Output clock phase shift relative in number of clock cycles relative to the fastest output clock.
- Desired bandwidth of the PLL (default is OPTIMIZED and the bandwidth is chosen in software)
- Compensation mode (automatically determined by the software)
- Reference clock jitter in UI (i.e., a percentage of the reference clock period)

Determine the Input Frequency

The first step is to determine the input frequency. This allows all possible output frequencies to be determined by using the minimum and maximum input frequencies to define the D counter range, the VCO operating range to determine the M counter range, and the output counter range since it has no restrictions. There can be a very large number of frequencies. In the worst case, there will be 52 x 64 x 128 = 425,984 possible combinations. In reality, the total number of different frequencies is less since the entire range of the M and D counters cannot be realized and there is overlap between the various settings. As an example, consider F_{IN} = 100 MHz. If the minimum PFD frequency is 20 MHz, then D can only go from 1 to 5. For D = 1, M can only have values from four to 11. If D = 2, M can have values from 8 to 22. In addition, D = 1 M = 4 is a subset of D = 2 M = 8 allowing the D = 1 M = 4 case to be dropped. For this case, only D = 3, 4, and 5 are considered since all other D values are subsets of these cases. This drastically reduces the number of possible output frequencies. The output frequencies are sequentially selected. The desired output frequency should be checked against the possible output frequencies generated. Once the first output frequency is determined, an additional constraint can be imposed on the values of M and D. This can further limit the possible output frequencies

for the second output frequency. Continue this process until all the output frequencies are selected.

The constraints used to determine the allowed M and D values are shown in the following equations:

$$D_{MIN} = roundup\frac{f_{IN}}{f_{PFD\ MAX}} \quad \text{Equation 3-3}$$

$$D_{MAX} = rounddown\frac{f_{IN}}{f_{PFD\ MIN}} \quad \text{Equation 3-4}$$

$$M_{MIN} = \left(roundup\frac{f_{VCOMIN}}{f_{IN}}\right) \times D_{MIN} \quad \text{Equation 3-5}$$

$$M_{MAX} = rounddown\frac{D_{MAX} \times f_{VCOMAX}}{f_{IN}} \quad \text{Equation 3-6}$$

Determine the M and D Values

Determining the input frequency can result in several possible M and D values. The next step is to determine the optimum M and D values. The starting M value is first determined. This is based off the VCO target frequency, the ideal operating frequency of the VCO.

$$M_{IDEAL} = \frac{D_{MIN} \times f_{VCOMAX}}{f_{IN}} \quad \text{Equation 3-7}$$

The goal is to find the M value closest to the ideal operating point of the VCO. The minimum D value is used to start the process. The goal is to make D and M values as small as possible while keeping f_{VCO} as high as possible.

PLL Ports

Table 3-3 summarizes the PLL ports. Table 3-4 lists the PLL attributes.

Table 3-3: **PLL Ports**

Pin Name	I/O	Pin Description
CLKIN1	Input	General clock input.
CLKIN2	Input	Secondary clock input to dynamically switch the PLL reference clock.
CLKFBIN	Input	Feedback clock input.
CLKINSEL	Input	Signal controls the state of the input mux, High = CLKIN1, Low = CLKIN2
RST	Input	Asynchronous reset signal. The RST signal is an asynchronous reset for the PLL. The PLL will synchronously re-enable itself when this signal is released (i.e., PLL re-enabled). A reset is required when the input clock conditions change (e.g., frequency).
DADDR[4:0]	Input	The dynamic reconfiguration address (DADDR) input bus provides a reconfiguration address for the dynamic reconfiguration. When not used, all bits must be assigned zeros.

Table 3-3: **PLL Ports *(Continued)***

Pin Name	I/O	Pin Description
DI[15:0]	Input	The dynamic reconfiguration data input (DI) bus provides reconfiguration data. When not used, all bits must be set to zero.
DWE	Input	The dynamic reconfiguration write enable (DWE) input pin provides the write enable control signal to write the DI data into the DADDR address. When not used, it must be tied Low.
DEN	Input	The dynamic reconfiguration enable (DEN) provides the enable control signal to access the dynamic reconfiguration feature. When the dynamic reconfiguration feature is not used, DEN must be tied Low.
DCLK	Input	The DCLK signal is the reference clock for the dynamic reconfiguration port.
REL	Input	The release pin is used when the PLL is in PMCD mode. When in PLL mode, leave unconnected or tied Low. Only use this pin when porting existing Virtex-4 designs containing the legacy PMCD mode.
CLKOUT[0:5](1)	Output	User configurable clock outputs (0 through 5) that can be divided versions of the VCO phase outputs (user controllable) from 1 (bypassed) to 128. The input clock and output clocks are phase aligned.
CLKFBOUT	Output	Dedicated PLL feedback output.
CLKOUTDCM[0:5](1)	Output	User configurable clocks (0 through 5) that can only connect to the DCM within the same CMT as the PLL.
CLKFBDCM	Output	PLL feedback used to compensate if the PLL is driving the DCM. If the CLKFBOUT pin is used for this purpose, the software will automatically map to the correct port.
LOCKED	Output	Synchronous output from the PLL that indicates when the PLL has achieved phase alignment within a predefined window and frequency matching within a predefined PPM range. The PLL automatically locks after power on, no extra reset is required. LOCKED will be deasserted if the input clock stops or the phase alignment is violated (e.g., input clock phase shift). The PLL must be reset after LOCKED is deasserted.
DO[15:0]	Output	The dynamic reconfiguration output bus provides PLL data output when using dynamic reconfiguration.
DRDY	Output	The dynamic reconfiguration ready output (DRDY) provides the response to the DEN signal for the PLLs dynamic reconfiguration feature.

Notes:

1. $CLKOUT_N$ and $CLKOUTDCM_N$ are utilizing the same output counters and can not be operated independently.

PLL Attributes

Table 3-4: **PLL Attributes**

Attribute	Type	Allowed Values	Default	Description
COMPENSATION	String	SYSTEM_SYNCHRONOUS SOURCE_SYNCHRONOUS	SYSTEM_ SYNCHRONOUS	Specifies the PLL phase compensation for the incoming clock. SYSTEM_SYNCHRONOUS attempts to compensate all clock delay for 0 hold time. SOURCE_SYNCHRONOUS is used when a clock is provided with data and thus phased with the clock. Additional attributes automatically selected by the ISE software: INTERNAL EXTERNAL DCM2PLL PLL2DCM
BANDWIDTH	String	HIGH LOW OPTIMIZED	OPTIMIZED	Specifies the PLL programming algorithm affecting the jitter, phase margin and other characteristics of the PLL.
CLKOUT[0:5]_DIVIDE	Integer	1 to 128	1	Specifies the amount to divide the associated CLKOUT clock output if a different frequency is desired. This number in combination with the CLKFBOUT_MULT and DIVCLK_DIVIDE values will determine the output frequency.
CLKOUT[0:5]_PHASE	Real	–360.0 to 360.0	0.0	Allows specification of the output phase relationship of the associated CLKOUT clock output in number of degrees offset (i.e., 90 indicates a 90° or ¼ cycle offset phase offset while 180 indicates a 180° offset or ½ cycle phase offset).
CLKOUT[0:5]_ DUTY_CYCLE	Real	0.01 to 0.99	0.50	Specifies the Duty Cycle of the associated CLKOUT clock output in percentage (i.e., 0.50 will generate a 50% duty cycle).
CLKFBOUT_MULT	Integer	1 to 64	1	Specifies the amount to multiply all CLKOUT clock outputs if a different frequency is desired. This number, in combination with the associated CLKOUT#_DIVIDE value and DIVCLK_DIVIDE value, will determine the output frequency.
DIVCLK_DIVIDE	Integer	1 to 52	1	Specifies the division ratio for all output clocks with respect to the input clock.

Table 3-4: **PLL Attributes *(Continued)***

Attribute	Type	Allowed Values	Default	Description
CLKFBOUT_PHASE	Real	0.0 to 360.0	0.0	Specifies the phase offset in degrees of the clock feedback output. Shifting the feedback clock results in a negative phase shift of all output clocks to the PLL.
REF_JITTER	Real	0.000 to 0.999	0.100	Allows specification of the expected jitter on the reference clock in order to better optimize PLL performance. A bandwidth setting of OPTIMIZED will attempt to choose the best parameter for input clocking when unknown. If known, then the value provided should be specified in terms of the UI percentage (the maximum peak to peak value) of the expected jitter on the input clock.
CLKIN1_PERIOD	Real	1.408 to 52.630	0.000	Specifies the input period in ns to the PLL CLKIN1 input. Resolution is down to the ps. This information is mandatory and must be supplied.
CLKIN2_PERIOD	Real	1.408 to 52.630	0.000	Specifies the input period in ns to the PLL CLKIN2 input. Resolution is down to the ps. This information is mandatory and must be supplied.
CLKOUT[0:5]_ DESKEW_ADJUST	String	PPC or None	None	Fixed delay used when the PLL is used in a PPC440 system. See UG200: *Embedded Processor Block in Virtex-5 FPGAs Reference Guide* for details.
RESET_ON_LOSS _OF_LOCK	String	FALSE	FALSE	Must be set to FALSE, not supported in silicon.

PLL CLKIN1 and CLKIN2 Usage

CLKIN1 is the general purpose input to the PLL. The CLKIN2 pin is used to dynamically switch between CLKIN1 and CLKIN2 during operation, as selected by the CLKINSEL pin. If both CLKIN1 and CLKIN2 are used, and the PLL input clocks are driven by global clock pins, there are several restrictions on the placement of both clock signal pins. CLKIN1 can only come from IBUFG[4-0]. CLKIN2 can only come from IBUFG[9-5]. Further, CLKIN2 has to be mapped to a specific location depending on the value of CLKIN1. These rules are as follows:

If CLKIN1 is connected to IBUFG [x], CLKIN2 needs to be IBUFG [y] of the same type. Table 3-5 shows the general clock pin pairing.

Table 3-5: **Mapping Locations**

CLKIN1	CLKIN2
[0]	[5]
[1]	[6]
[2]	[7]
[3]	[8]
[4]	[9]

When the PLL input clocks are driven by the global clock trees (BUFGs), both clock inputs must be connected to the same clock input type. Driving one PLL clock input with a IBUFG and the other with a BUFG is not possible.

The following tables map the Virtex-5 FPGA global clock IBUFG pins with respect to CLKIN1 and CLKIN2. PLLs in the top half of the Virtex-5 device are driven by the global clock pins in bank3 and can be paired as listed in Table 3-6.

Table 3-6: **PLLs in the Top Half Pairing**

CLKIN1	CLKIN2
IO_L9P_GC_3	IO_L4P_GC_3
IO_L8P_GC_3	IO_L3P_GC_3
IO_L7P_GC_3	IO_L2P_GC_3
IO_L6P_GC_3	IO_L1P_GC_3
IO_L5P_GC_3	IO_L0P_GC_3

PLLs in the bottom half of the Virtex-5 device are driven by the global clock pins in bank4 and can be paired as listed in Table 3-6.

Table 3-7: **PLLs in the Bottom Half Pairing**

CLKIN1	CLKIN2
IO_L9P_GC_4	IO_L4P_GC_4
IO_L8P_GC_4	IO_L3P_GC_4
IO_L7P_GC_4	IO_L2P_GC_4
IO_L6P_GC_4	IO_L1P_GC_4
IO_L5P_GC_4	IO_L0P_GC_4

Other important notes on these pairings:

- The pin description names do not contain other possible multipurpose functions such as _CC, _VRN, _VRP or _VREF.
- Only the P-side pins are shown. For differential clock connections use the equivalent N-side pin. Inside the FPGA, only the P-side of the differential pin pair can connect to the CMT.
- For a mapping to the actual pin numbers consult the *Virtex-5 Family Packaging Specifications*.

PLL Clock Input Signals

The PLL clock source can come from several sources including:

- IBUFG - Global clock input buffer, the PLL will compensate the delay of this path.
- BUFGCTRL - Internal global clock buffer, the PLL will not compensate the delay of this path.
- IBUF - Not recommended since the PLL can not compensate for the delay of the general route. An IBUF clock input must route to a BUFG before routing to a PLL.
- DCMOUT - Any DCM output to PLL will compensate the delay of this path.

Counter Control

The PLL output counters provide a wide variety of synthesized clock using a combination of DIVIDE, DUTY_CYCLE, and PHASE. Figure 3-6 illustrates how the counter settings impact the counter output.

The top waveform represents either the output from the VCO in PLL mode.

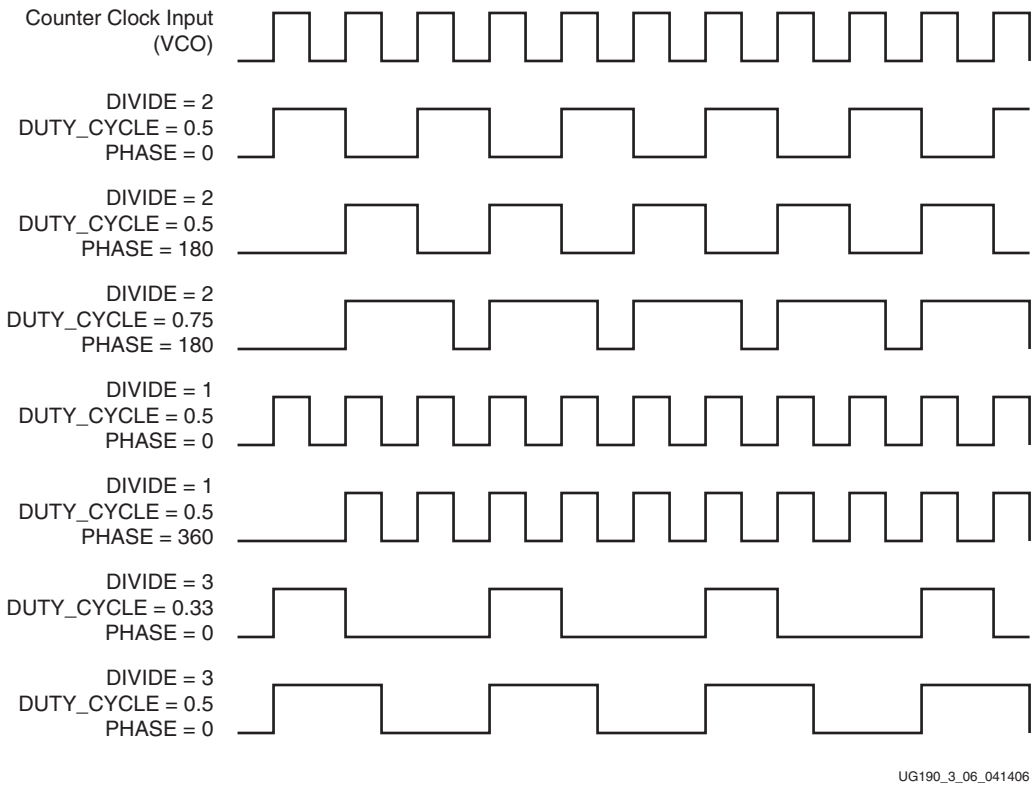


Figure 3-6: **Output Counter Clock Synthesis Examples**

Clock Shifting

The PLL output clocks can be shifted by inserting delay by selecting one of the eight phases in either the reference or the feedback path. The following figure shows the effect on a clock signal edge at the output of the PLL without any shifting versus the two cases (delay inserted in the feedback path and delay inserted in the reference path).

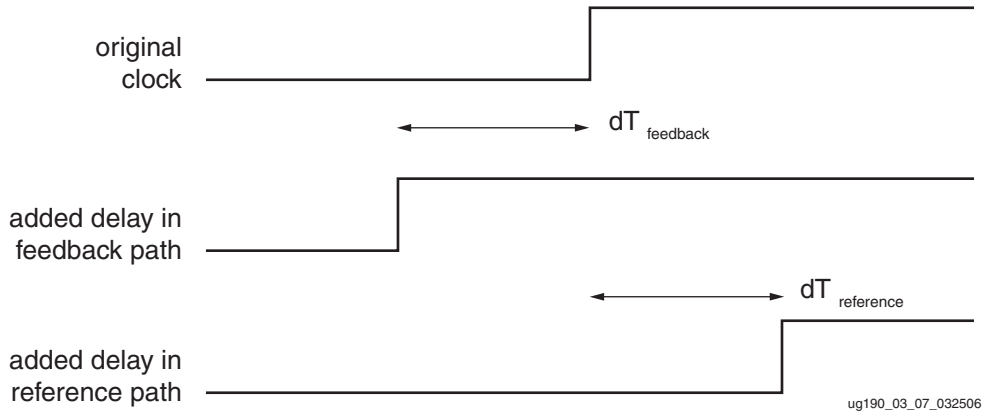


Figure 3-7: **Basic Output Clock Shifting**

Detailed VCO and Output Counter Waveforms

Figure 3-8 shows the eight VCO phase outputs and four different counter outputs. Each VCO phase is shown with the appropriate start-up sequence. The phase relationship and start-up sequence are guaranteed to insure the correct phase is maintained. This means the rising edge of the 0° phase will happen before the rising edge of the 45° phase. The O0 counter is programmed to do a simple divide by two with the 0° phase tap as the reference clock. The O1 counter is programmed to do a simple divide by two but uses the 180° phase tap from the VCO. Phase shifts greater than one VCO period are possible. This counter setting could be used to generate a clock for a DDR interface where the reference clock is edge aligned to the data transition. The O2 counter is programmed to do a divide by three. The O3 output has the same programming as the O2 output except the phase is set for a one cycle delay.

If the PLL is configured to provide a certain phase relationship and the input frequency is changed, then this phase relationship is also changed since the VCO frequency changes and therefore the absolute shift in picoseconds will change. This aspect must be considered when designing with the PLL. When an important aspect of the design is to maintain a certain phase relationship amongst various clock outputs, (e.g., CLK and CLK90) then this relationship will be maintained regardless of the input frequency.

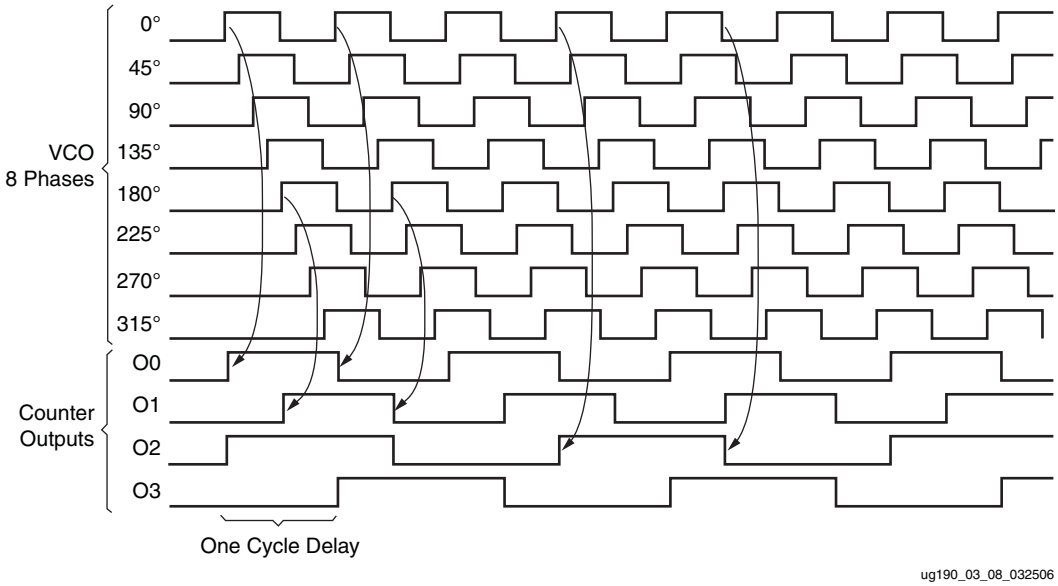


Figure 3-8: **Selecting VCO Phases**

All "O" counters are equivalent, anything O0 can do, O1 can do. The PLL outputs are flexible when connecting to the global clock network since they are identical. In most cases, this level of detail is imperceptible to the designer as the software and PLL Wizard determines the proper settings through the PLL attributes and Wizard inputs.

Reference Clock Switching

The PLL reference clock can be dynamically switched by using the CLKINSEL pin. The switching is done asynchronously. Since the clock signal can generate a narrow pulse resulting in erroneous behavior of the PLL, the PLL should be held in RESET while selecting the alternate clock with the CLKINSEL (CLKSRC) signal. The PLL clock mux switching is shown in Figure 3-9. The CLKINSEL (CLKSRC) signal directly controls the mux. No synchronization logic is present.

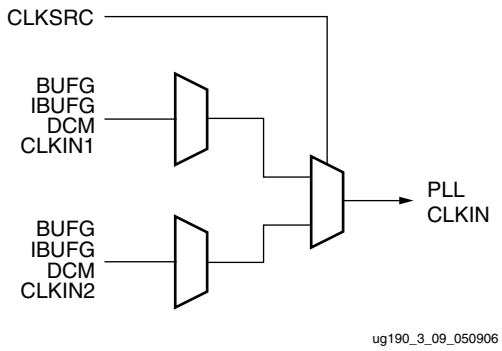


Figure 3-9: **Input Clock Switching**

Missing Input Clock or Feedback Clock

When the input clock or feedback clock is lost, the PLL will drive the output clocks to a lower or higher frequency, causing all of the output clocks to increase/decrease in frequency. The frequency increase/decrease can cause the clock output frequencies to change to as much as six times the original configuration.

PLL Use Models

There are several methods to design with the PLL. The PLL wizard in ISE software can assist with generating the various PLL parameters. Additionally, the PLL can be manually instantiated as a component. It is also possible for the PLL to be merge with an IP core. The IP core would contain and manage the PLL.

Clock Network Deskew

One of the predominant uses of the PLL is for clock network deskew. Figure 3-10 shows the PLL in this mode. The clock output from one of the O counters is used to drive logic within the fabric and/or the I/Os. The feedback counter is used to control the exact phase relationship between the input clock and the output clock (if, for example a 90° phase shift is required). The associated clock waveforms are shown to the right for the case where the input clock and output clock need to be phase aligned. This configuration is the most flexible, but it does require two global clock networks (Figure 3-10).

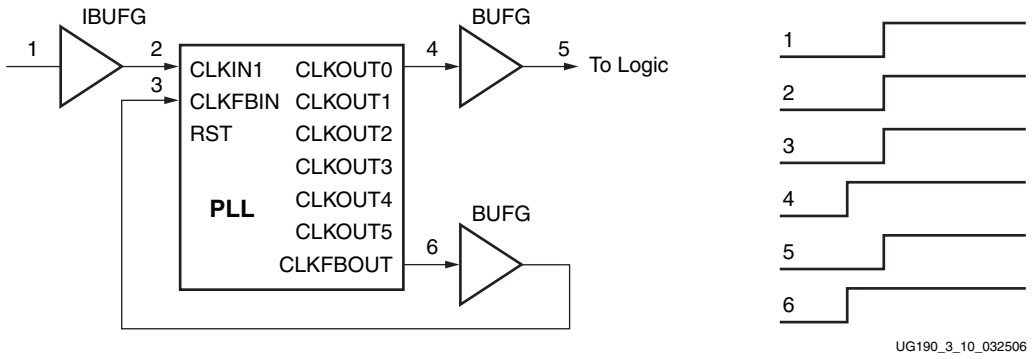


Figure 3-10: **Clock Deskew Using Two BUFGs**

There are certain restrictions on implementing the feedback. The CLKFBOUT output can be used to provide the feedback clock signal. The fundamental restriction is that both input frequencies to the PFD must be identical. Therefore, the following relationship must be met:

$$\frac{f_{IN}}{D} = f_{FB} = \frac{f_{VCO}}{M} \qquad \textit{Equation 3-8}$$

As an example, if f_{IN} is 166 MHz, D = 1, M = 3, and O = 1, then VCO and the clock output frequency are both 498 MHz. Since the M value in the feedback path is 3, both input frequencies at the PFD are 166 MHz.

In another more complex scenario has an input frequency of 66.66 MHz and D = 2, M = 15, and O = 2. The VCO frequency in this case is 500 MHz and the O output frequency is 250 MHz. Therefore, the feedback frequency at the PFD is 500/15 or 33.33 MHz, matching the 66.66MHz/2 input clock frequency at the PFD.

PLL with Internal Feedback

The PLL feedback can be internal to the PLL when the PLL is used as a synthesizer or jitter filter and there is no required phase relationship between the PLL input clock and the PLL output clock. The PLL performance should increase since the feedback clock is not subjected to noise on the core supply since it never passes through a block powered by this supply. Of course, noise introduced on the CLKIN signal and the BUFG will still be present (Figure 3-11).

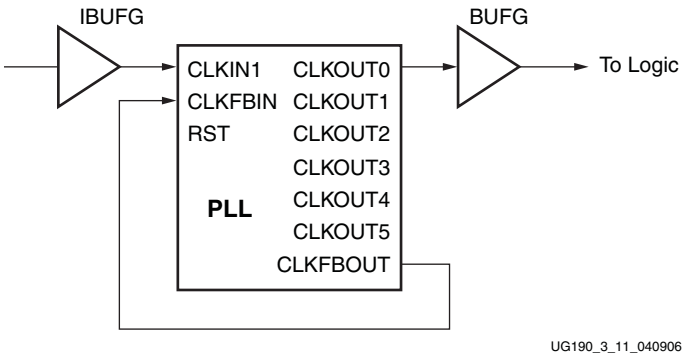


Figure 3-11: **PLL with Internal Feedback**

Zero Delay Buffer

The PLL can also be used to generate a zero delay buffer clock. A zero delay buffer can be useful for applications where there is a single clock signal fan out to multiple destinations with a low skew between them. This configuration is shown in the Figure 3-12. Here, the feedback signal drives off chip and the board trace feedback is designed to match the trace to the external components. In this configuration, it is assumed that the clock edges are aligned at the input of the FPGA and the input of the external component. There will be a limitation on the maximum delay allowed in the feedback path.

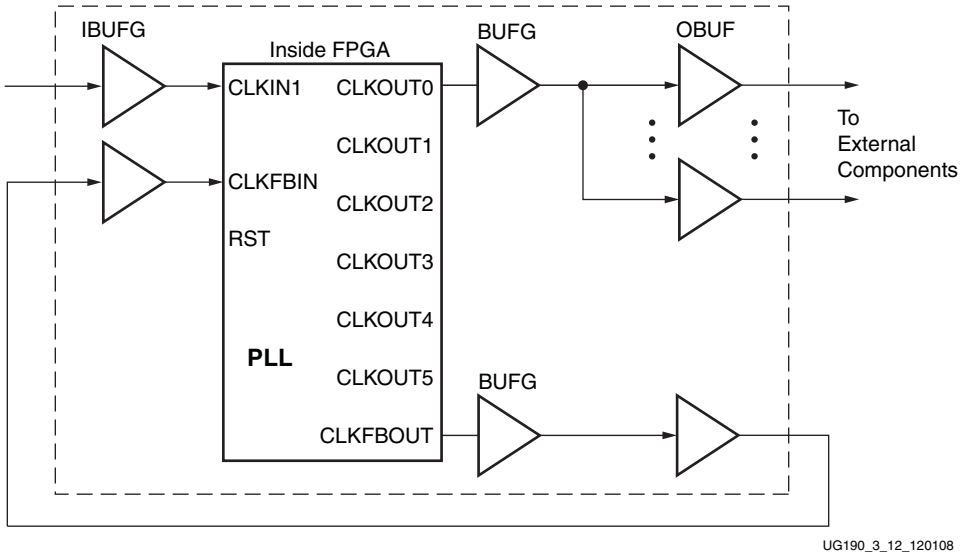


Figure 3-12: **Zero Delay Buffer**

In some cases precise alignment will not occur because of the difference in loading between the input capacitance of the external component and the feedback path capacitance of the FPGA. For example, the external components can have an input capacitance on 1 pF to 4 pF while the FPGA has an input capacitance of around 8 pF. There is a difference in the signal slope, which is basically skew. Designers need to be aware of this effect to ensure timing.

DCM Driving PLL

The DCM provides an excellent method for generating precision phase-shifted clocks. However, the DCM cannot reduce the jitter on the reference clock. The PLL can be used to reduce the output jitter of one DCM clock output. This configuration is shown in Figure 3-13. The PLL is configured to not introduce any phase shift (zero delay through the PLL). The associated waveforms are shown to the right of the block diagram. When the output of the DCM is used to drive the PLL directly, both DCM and PLL *must* reside within the same CMT block. This is the preferred implementation since it produces a minimal amount of noise on the local, dedicated route. However, a connection can also be made by connecting the DCM to a BUFG and then to the CLKIN input of a PLL.

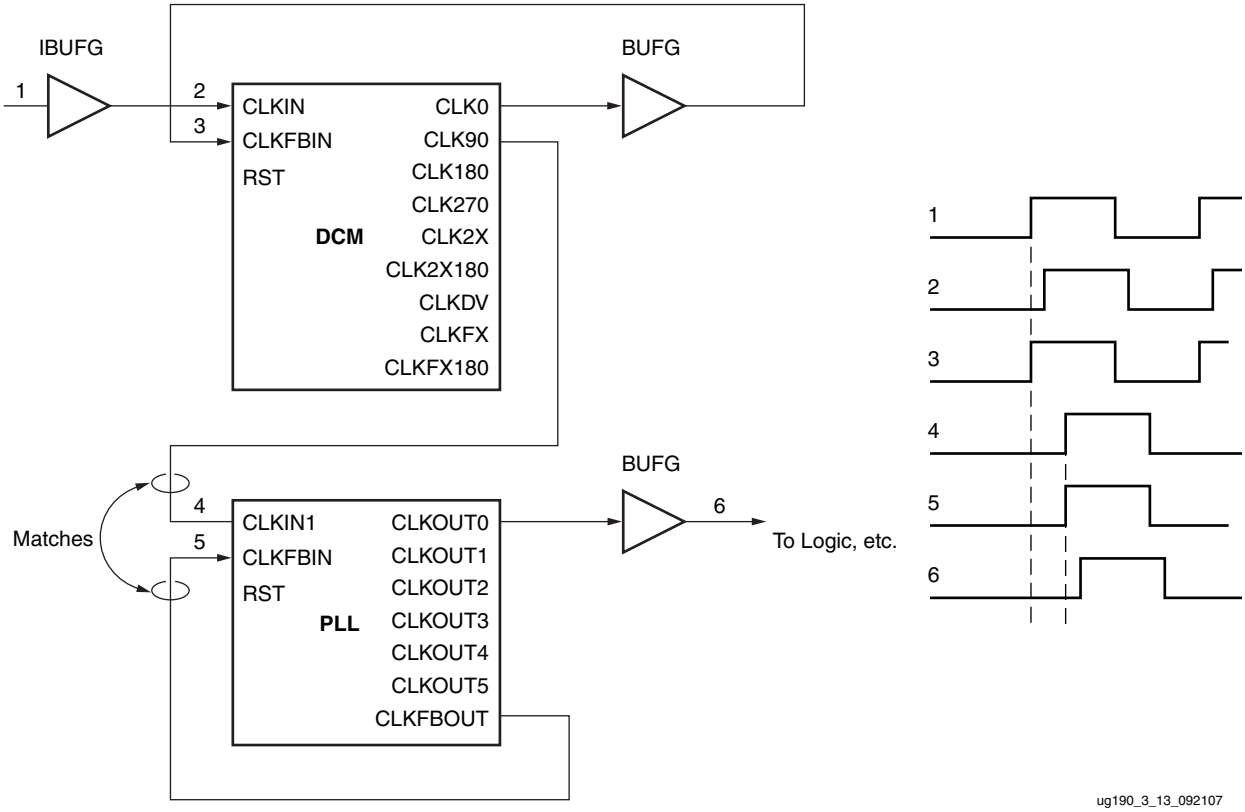


Figure 3-13: **DCM Driving a PLL**

PLL Driving DCM

A second option for reduce clock jitter is to use the PLL to clean-up the input clock jitter before driving into the DCM. This will improve the output jitter of all DCM outputs, but any added jitter by the DCM will still be passed to the clock outputs. Both PLL and DCM should reside in the same CMT block because dedicated resources exist between the PLL and DCM to support the zero delay mode. When the PLL and DCM do not reside in the same CMT, then the only connection is through a BUFG hindering the possibility of deskew.

One PLL can drive multiple DCMs as long as the reference frequency can be generated by a single PLL. For example, if a 33 MHz reference clock is driven into the PLL, and the design uses one DCM to operate at 200 MHz and the other to run at 100 MHz, then the VCO can be operated at 600 MHz (M1 = 18). The VCO frequency can be divided by three to generate a 200 MHz clock and another counter can be divided by six to generate the 100 MHz clock. For the example in Figure 3-14, one PLL can drive both DCMs.

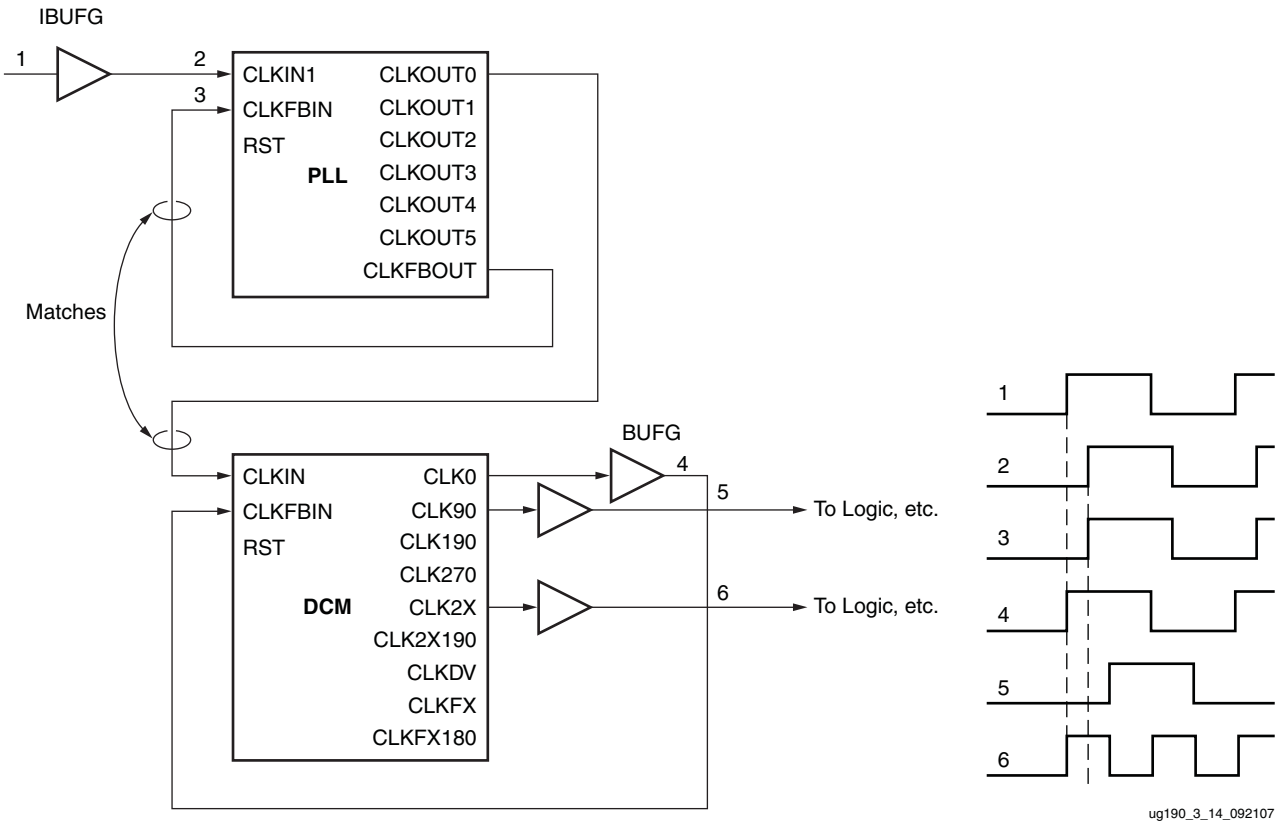


Figure 3-14: **PLL Driving a DCM**

PLL to PLL Connection

The PLL can be cascaded to allow generation of a greater range of clock frequencies. The frequency range restrictions still apply. Equation 3-9 shows the relationship between the final output frequency and the input frequency and counter settings of the two PLLs (Figure 3-15.) The phase relationship between the output clock of the second PLL and the input clock is undefined.To cascade PLLs, route the output of the first PLL to a BUFG and then to the CLKIN pin of the second PLL. This path provides the lowest device jitter.

$$f_{OUTPLL2} = f_{OUTPLL1}\frac{M_{PLL2}}{D_{PLL2} \times O_{PLL2}} = f_{IN}\frac{M_{PLL1}}{D_{PLL1} \times O_{PLL1}} \times \frac{M_{PLL2}}{D_{PLL2} \times O_{PLL2}} \quad \text{Equation 3-9}$$

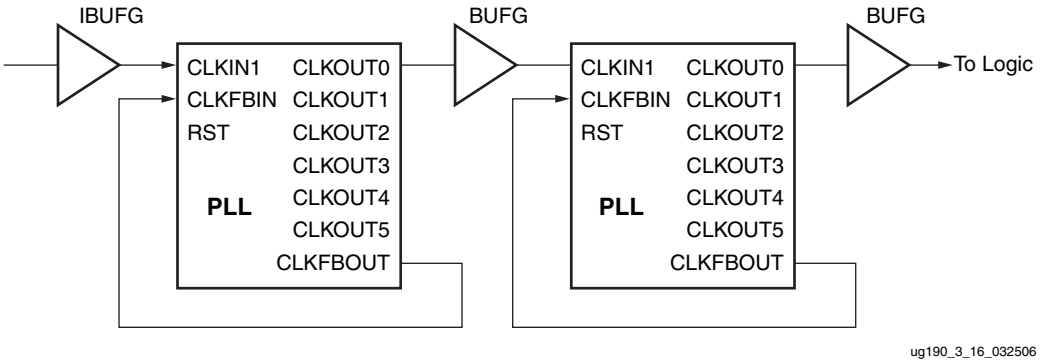


Figure 3-15: **Cascading Two PLLs**

Application Guidelines

This section summarizes when to select a DCM over a PLL, or a PLL over a DCM.

Virtex-5 FPGA PLLs support up to six independent outputs. Designs using several different outputs should use PLLs. An example of designs using several different outputs follows. The PLL is an ideal solution for this type of application because it can generate a configurable set of outputs over a wide range while the DCM has a fixed number of predetermined outputs based off the reference clock. When the application requires a fine phase shift or a dynamic variable phase shift, a DCM could be a better solution.

PLL Application Example

The following PLL attribute settings result in a wide variety of synthesized clocks:

```
CLKOUT0_PHASE = 0;
CLKOUT0_DUTY_CYCLE = 0.5;
CLKOUT0_DIVIDE = 2;
CLKOUT1_PHASE = 90;
CLKOUT1_DUTY_CYCLE = 0.5;
CLKOUT1_DIVIDE = 2;
CLKOUT2_PHASE = 0;
CLKOUT2_DUTY_CYCLE = 0.25;
CLKOUT2_DIVIDE = 4;
CLKOUT3_PHASE = 90;
CLKOUT3_DUTY_CYCLE = 0.5;
CLKOUT3_DIVIDE = 8;
CLKOUT4_PHASE = 0;
CLKOUT4_DUTY_CYCLE = 0.5;
CLKOUT4_DIVIDE = 8;
CLKOUT5_PHASE = 135;
CLKOUT5_DUTY_CYCLE = 0.5;
CLKOUT5_DIVIDE = 8;
CLKFBOUT_PHASE = 0;
CLKFBOUT_MULT = 8;
DIVCLK_DIVIDE = 1;
CLKIN1_PERIOD = 10.0;
```

Figure 3-16 displays the resulting waveforms.

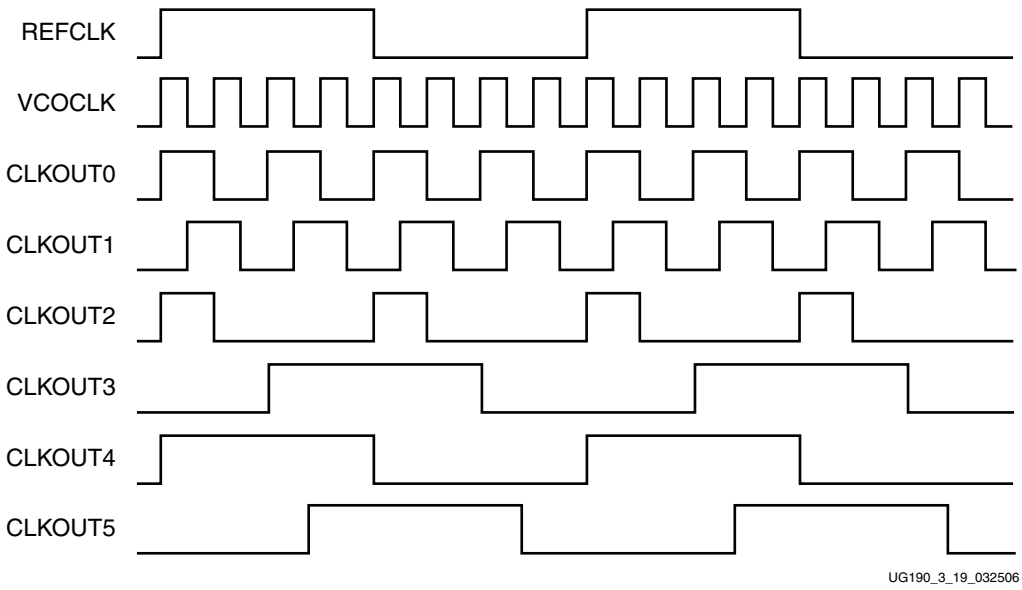


Figure 3-16: **Example Waveform**

PLL in Virtex-4 FPGA PMCD Legacy Mode

Virtex-5 devices do not have Phase-Matched Clock Dividers (PMCDs). The Virtex-5 FPGA PLL supports the Virtex-4 FPGA PMCD mode of operation. To take advantage of the inherently more powerful features of the Virtex-5 FPGA PLL, Xilinx recommends redesigning Virtex-4 FPGA PMCDs by implementing PLLs directly. The difference between the Virtex-5 FPGA PLL and the Virtex-4 FPGA PMCD block in Virtex-4 FPGA PMCD legacy mode is that only two clock inputs are supported in the Virtex-5 device implementation. The Virtex-4 device implementation supported up to four clock inputs. If four clock inputs must be used, then two PLLs can be put into PMCD mode. In this case, delay matching is not optimal.

Figure 3-17 shows the Virtex-4 FPGA PMCD primitive implemented using a PLL. A PLL can not be used as a PLL if it is already being used as a PMCD. To design-in the Virtex-5 FPGA PMCD functionality, instantiate a Virtex-4 FPGA PMCD primitive. ISE software maps the Virtex-4 FPGA PMCD primitive into a Virtex-5 FPGA PLL.

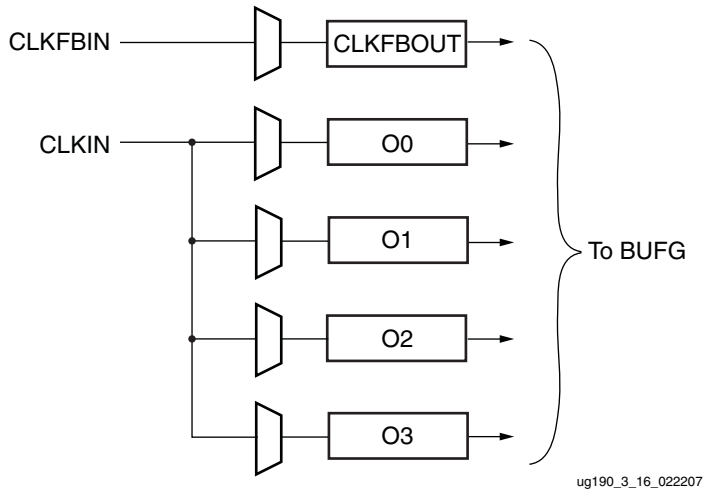


Figure 3-17: **PMCD Primitive Implemented Using the PLL in PMCD Legacy Mode**

Table 3-8 shows the port mapping between Virtex-5 FPGA PLL in PMCD legacy mode and the Virtex-4 FPGA PMCD port names.

Table 3-8: **Mapping of Port Names**

Virtex-4 FPGA Port Name	Virtex-5 FPGA Port Name
CLKA	CLKIN
CLKB	CLKFBIN
CLKC	n/a
CLKD	n/a
CLKA1	CLKOUT3
CLKA1D2	CLKOUT2
CLKA1D4	CLKOUT1
CLKA1D8	CLKOUT0

Table 3-8: **Mapping of Port Names *(Continued)***

Virtex-4 FPGA Port Name	Virtex-5 FPGA Port Name
CLKB1	CLKFBOUT
CLKC1	n/a
CLKD1	n/a
RST	RST
REL	REL

Table 3-9 shows the PLL attributes in Virtex-4 FPGA PMCD legacy mode.

Table 3-9: **PLL Attributes When in Virtex-4 FPGA PMCD Legacy Mode**

Attribute	Type	Allowed Values	Default	Description
PLL_PMCD_MODE	Boolean	TRUE or FALSE	FALSE	Enables PLL to act as PMCDs
EN_REL	Boolean	TRUE or FALSE	FALSE	When in PMCD mode (PLL_PMCD_MODE = TRUE), specifies release of divided clock CLKA outputs when the REL input pin is asserted.
RST_DEASSERT_CLK	String	CLKA CLKB	CLKA	When in PMCD mode (PLL_PMCD_MODE = TRUE), specifies a clock to synchronize with the release of RST.

Table 3-10 shows the PLL ports in Virtex-4 FPGA PMCD legacy mode.

Table 3-10: **PLL Ports in Virtex-4 FPGA PMCD Legacy Mode**

Port Name	I/O	Pin Description
CLKFB	Input	Virtex-4 FPGA PMCD legacy mode CLKB input clock to the PMCD.
CLKIN	Input	Virtex-4 FPGA PMCD legacy mode CLKA input clock to the PMCD.
RST	Input	RST is the reset input to the Virtex-4 FPGA PMCD legacy mode. Asserting RST signal asynchronously forces all outputs Low. Deasserting RST synchronously allows all outputs to toggle.
REL	Input	REL is the release input to the Virtex-4 FPGA PMCD legacy mode. Asserting the REL signal releases the divided outputs synchronous to CLKA.
CLKOUT0	Output	Virtex-4 FPGA PMCD legacy mode CLKB1.
CLKOUT1	Output	Virtex-4 FPGA PMCD legacy mode CLKA1.
CLKOUT2	Output	Virtex-4 FPGA PMCD legacy mode CLKA1D2.
CLKOUT3	Output	Virtex-4 FPGA PMCD legacy mode CLKA1D4.
CLKOUT4	Output	Virtex-4 FPGA PMCD legacy mode CLKA1D8.

XILINX®

Chapter 4

Block RAM

Block RAM Summary

The block RAM in Virtex-5 FPGAs stores up to 36K bits of data and can be configured as either two independent 18 Kb RAMs, or one 36 Kb RAM. Each 36 Kb block RAM can be configured as a 64K x 1 (when cascaded with an adjacent 36 Kb block RAM), 32K x 1, 16K x 2, 8K x 4, 4K x 9, 2K x 18, or 1K x 36 memory. Each 18 Kb block RAM can be configured as a 16K x 1, 8K x2 , 4K x 4, 2K x 9, or 1K x 18 memory.

Similar to the Virtex-4 FPGA block RAMs, Write and Read are synchronous operations; the two ports are symmetrical and totally independent, sharing only the stored data. Each port can be configured in one of the available widths, independent of the other port. In addition, the read port width can be different from the write port width for each port. The memory content can be initialized or cleared by the configuration bitstream. During a write operation the memory can be set to have the data output either remain unchanged, reflect the new data being written or the previous data now being overwritten.

Virtex-5 FPGA block RAM enhancements include:

- Increased memory storage capability per block. Each block RAM can store up to 36K bits of data.
- Support of two independent 18K blocks, or a single 36K block RAM.
- Each 36K block RAM can be set to simple dual-port mode, doubling data width of the block RAM to 72 bits. The 18K block RAM can also be set to simple dual-port mode, doubling data width to 36 bits. Simple dual-port mode is defined as having one read-only port and one write-only port with independent clocks.
- Two adjacent block RAMs can be combined to one deeper 64K x 1 memory without any external logic.
- One 64-bit Error Correction Coding block is provided per 36 Kb block RAM or 36 Kb FIFO. Separate encode/decode functionality is available.
- Synchronous Set/Reset of the outputs to an initial value is available for both the latch and register modes of the block RAM output.
- An attribute to configure the block RAM as a synchronous FIFO to eliminate flag latency uncertainty.
- The Virtex-5 FIFO does not have FULL flag assertion latency.

Virtex-5 FPGA block RAM features:

- 18, 36, or 72-bit wide ports can have an individual write enable per byte. This feature is popular for interfacing to an on-chip microprocessor.
- Each block RAM contains optional address sequencing and control circuitry to operate as a built-in multirate FIFO memory. In Virtex-5 architecture, the block RAM can be configured as an 18Kb or 36Kb FIFO.

- All inputs are registered with the port clock and have a setup-to-clock timing specification.
- All outputs have a read function or a read-during-write function, depending on the state of the write enable (WE) pin. The outputs are available after the clock-to-out timing interval. The read-during-write outputs have one of three operating modes: WRITE_FIRST, READ_FIRST, and NO_CHANGE.
- A write operation requires one clock edge.
- A read operation requires one clock edge.
- All output ports are latched. The state of the output port does not change until the port executes another read or write operation. The default block RAM output is latch mode.
- The output data path has an optional internal pipeline register. Using the register mode is strongly recommended. This allows a higher clock rate, however, it adds a clock cycle latency of one.

Virtex-5 FPGA block RAM usage rules:

- The Synchronous Set/Reset (SSR) port cannot be used when the ECC decoder is enabled (EN_ECC_READ = TRUE).
- The setup time of the block RAM address and write enable pins must not be violated. Violating the address setup time (even if write enable is Low) will corrupt the data contents of the block RAM.
- The block RAM register mode SSR requires REGCE = 1 to reset the output DO register value. The block RAM array data output latch does not get reset in this mode. The block RAM latch mode SSR requires the block RAM enable, EN = 1, to reset the output DO latch value.
- Although RAMB18SDP (x36 18k block RAM) and RAMB36SDP (x72 36k block RAM) are simple dual-port primitives, the true dual-port primitives (RAMB18 and RAMB36) can be used with one read-only port and one write-only port. For example: a RAMB18s READ_WIDTH_A = 18, WRITE_WIDTH_B = 9, with WEA = 0 and WEB = 1 is effectively a simple dual-port block RAM with a smaller port width having been derived from the true dual-port primitive. Similarly, a ROM function can be built out of either the true dual-port (RAMB18 or RAMB36) or the simple dual-port block RAM primitives (RAMB18SDP or RAMB36SDP).
- Different read and write port width choices are available when using specific block RAM primitives. The parity bits are only available for the x9, x18, and x36 port widths. The parity bits should not be used when the read width is x1, x2, or x4. If the read width is x1, x2 or x4, the effective write width is x1, x2, x4, x8, x16, or x32. Similarly, when a write width is x1, x2, or x4, the actual available read width is x1, x2, x4, x8, x16, or x32 even though the primitive attribute is set to 1, 2, 4, 9, 18, or 36 respectively. Table 4-1 shows some possible scenarios.

Table 4-1: **Parity Use Sceneries**

Primitive	Settings		Effective Read Width	Effective Write Width
	Read Width	Write Width		
RAMB18	1, 2, or 4	9 or 18	Same as setting	8 or 16
RAMB18	9 or 18	1, 2, or 4	8 or 16	Same as setting
RAMB18	1, 2, or 4	1, 2, or 4	Same as setting	Same as setting
RAMB18	9 or 18	9 or 18	Same as setting	Same as setting

Table 4-1: **Parity Use Sceneries *(Continued)***

Primitive	Settings		Effective Read Width	Effective Write Width
	Read Width	Write Width		
RAMB36	1, 2, or 4	9, 18, or 36	Same as setting	8, 16, or 32
RAMB36	9, 18, or 36	1, 2, or 4	8, 16, or 32	Same as setting
RAMB36	1, 2, or 4	1, 2, or 4	Same as setting	Same as setting
RAMB36	9, 18, or 36	9, 18, or 36	Same as setting	Same as setting

Notes:

1. Do not use parity bits DIP/DOP when one port widths is less than nine and another port width is nine or greater.

Block RAM Introduction

In addition to distributed RAM memory and high-speed SelectIO™ memory interfaces, Virtex-5 devices feature a large number of 36 Kb block RAMs. Each 36 Kb block RAM contains two independently controlled 18 Kb RAMs. Block RAMs are placed in columns, and the total number of block RAM memory depends on the size of the Virtex-5 device. The 36 Kb blocks are cascadable to enable a deeper and wider memory implementation, with a minimal timing penalty.

Embedded dual- or single-port RAM modules, ROM modules, synchronous FIFOs, and data width converters are easily implemented using the Xilinx CORE Generator™ block memory modules. Multirate FIFOs can be generated using the CORE Generator FIFO Generator module. The synchronous or asynchronous (multirate) FIFO implementation does not require additional CLB resources for the FIFO control logic since it uses dedicated hardware resources.

Synchronous Dual-Port and Single-Port RAMs

Data Flow

The true dual-port 36 Kb block RAM dual-port memories consist of a 36 Kb storage area and two completely independent access ports, A and B. Similarly, each 18 Kb block RAM dual-port memory consists of an 18 Kb storage area and two completely independent access ports, A and B. The structure is fully symmetrical, and both ports are interchangeable. Figure 4-1 illustrates the true dual-port data flow. Table 4-2 lists the port names and descriptions.

Data can be written to either or both ports and can be read from either or both ports. Each write operation is synchronous, each port has its own address, data in, data out, clock, clock enable, and write enable. The read and write operations are synchronous and require a clock edge.

There is no dedicated monitor to arbitrate the effect of identical addresses on both ports. It is up to the user to time the two clocks appropriately. Conflicting simultaneous writes to the same location never cause any physical damage but can result in data uncertainty.

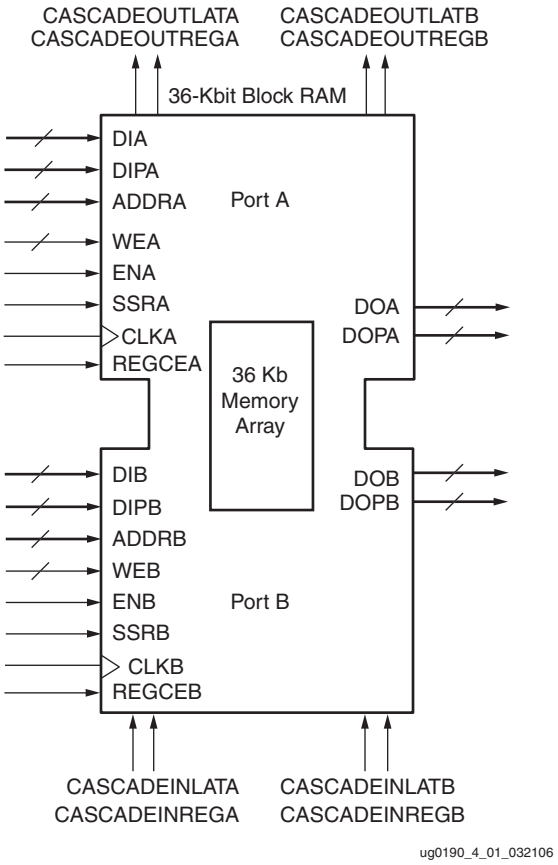


Figure 4-1: **True Dual-Port Data Flows**

Table 4-2: **True Dual-Port Names and Descriptions**

Port Name	Description
DI[A | B]	Data Input Bus
DIP[A | B](1)	Data Input Parity Bus, can be used for additional data inputs.
ADDR[A | B]	Address Bus
WE[A | B]	Byte-wide Write Enable
EN[A | B]	When inactive no data is written to the block RAM and the output bus remains in its previous state.
SSR[A | B]	Synchronous Set/Reset for either latch or register modes.
CLK[A | B]	Clock Input
DO[A | B]	Data Output Bus.
DOP[A | B](1)	Data Output Parity Bus, can be used for additional data outputs.
REGCE[A | B]	Output Register Enable

Table 4-2: **True Dual-Port Names and Descriptions *(Continued)***

Port Name	Description
CASCADEINLAT[A | B]	Cascade input pin for 64K x 1 mode when optional output registers are not enabled
CASCADEOUTLAT[A | B]	Cascade output pin for 64K x 1 mode when optional output registers are not enabled
CASCADEINREG[A | B]	Cascade input for 64K x 1 mode when optional input register is enabled
CASCADEOUTREG[A | B]	Cascade output for 64K x 1 mode when optional output register is enabled

Notes:

1. The "Data-In Buses - DI[A | B]<#:0> & DIP[A | B]<#:0>" section has more information on data parity pins.

Read Operation

In latch mode, the read operation uses one clock edge. The read address is registered on the read port, and the stored data is loaded into the output latches after the RAM access time. When using the output register, the read operation will take one extra latency cycle.

Write Operation

A write operation is a single clock-edge operation. The write address is registered on the write port, and the data input is stored in memory.

Write Modes

Three settings of the write mode determines the behavior of the data available on the output latches after a write clock edge: WRITE_FIRST, READ_FIRST, and NO_CHANGE. Write mode selection is set by configuration. The Write mode attribute can be individually selected for each port. The default mode is WRITE_FIRST. WRITE_FIRST outputs the newly written data onto the output bus. READ_FIRST outputs the previously stored data while new data is being written. NO_CHANGE maintains the output previously generated by a read operation.

For the simple dual port block RAM, the Write mode is always READ_FIRST and therefore no collision can occur when used in synchronous mode.

WRITE_FIRST or Transparent Mode (Default)

In WRITE_FIRST mode, the input data is simultaneously written into memory and stored in the data output (transparent write), as shown in Figure 4-2. These waveforms correspond to latch mode when the optional output pipeline register is not used.

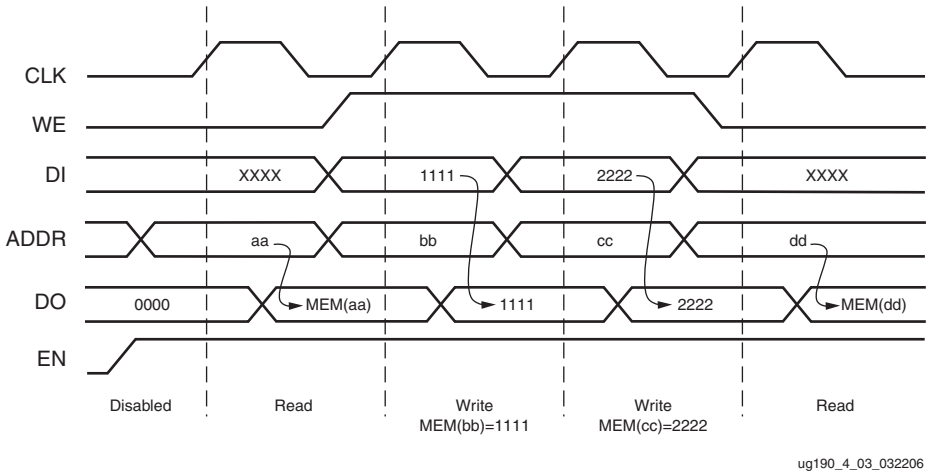


Figure 4-2: **WRITE_FIRST Mode Waveforms**

READ_FIRST or Read-Before-Write Mode

In READ_FIRST mode, data previously stored at the write address appears on the output latches, while the input data is being stored in memory (read before write). The waveforms in Figure 4-3 correspond to latch mode when the optional output pipeline register is not used.

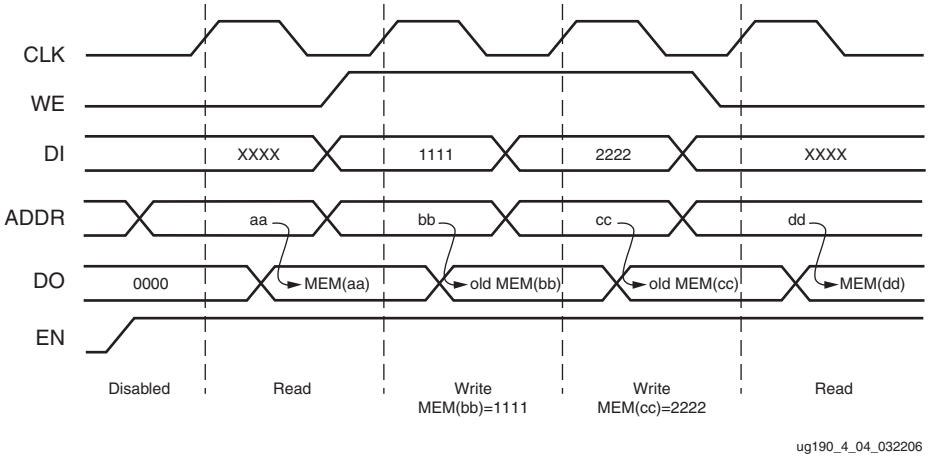


Figure 4-3: **READ_FIRST Mode Waveforms**

NO_CHANGE Mode

In NO_CHANGE mode, the output latches remain unchanged during a write operation. As shown in Figure 4-4, data output remains the last read data and is unaffected by a write operation on the same port. These waveforms correspond to latch mode when the optional output pipeline register is not used.

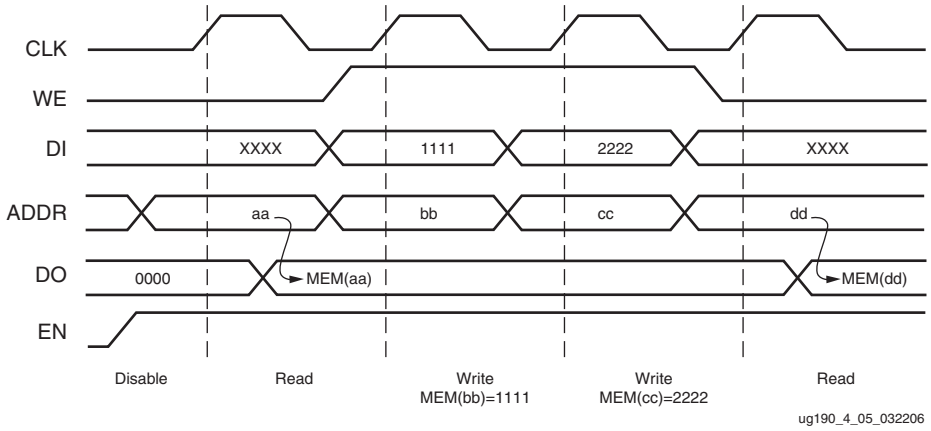


Figure 4-4: **NO_CHANGE Mode Waveforms**

Conflict Avoidance

Virtex-5 FPGA block RAM memory is a true dual-port RAM where both ports can access any memory location at any time. When accessing the same memory location from both ports, the user must, however, observe certain restrictions. There are two fundamentally different situations: The two ports either have a common clock (synchronous clocking), or the clock frequency and phase is different for the two ports (asynchronous clocking).

Asynchronous Clocking

Asynchronous clocking is the more general case, where the active edges of both clocks do not occur simultaneously:

- There are no timing constraints when both ports perform a read operation.
- When one port performs a write operation, the other port must not read- or write-access the same memory location. The simulation model will produce an error if this condition is violated. If this restriction is ignored, a read or write operation will produce unpredictable results. There is, however, no risk of physical damage to the device. If a read and write operation is performed, then the write will store valid data at the write location.

Synchronous Clocking

Synchronous clocking is the special case, where the active edges of both port clocks occur simultaneously:

- There are no timing constraints when both ports perform a read operation.
- When one port performs a write operation, the other port must not write into the same location, unless both ports write identical data.
- When one port performs a write operation, the write operation succeeds; the other port can reliably read data from the same location if the write port is in READ_FIRST mode. DATA_OUT on both ports will then reflect the previously stored data.

  If the write port is in either WRITE_FIRST or in NO_CHANGE mode, then the DATA-OUT on the read port would become invalid (unreliable). The mode setting of the read-port does not affect this operation.

Additional Block RAM Features in Virtex-5 Devices

Optional Output Registers

The optional output registers improve design performance by eliminating routing delay to the CLB flip-flops for pipelined operation. An independent clock and clock enable input is provided for these output registers. As a result the output data registers hold the value independent of the input register operation. Figure 4-5 shows the optional output register.

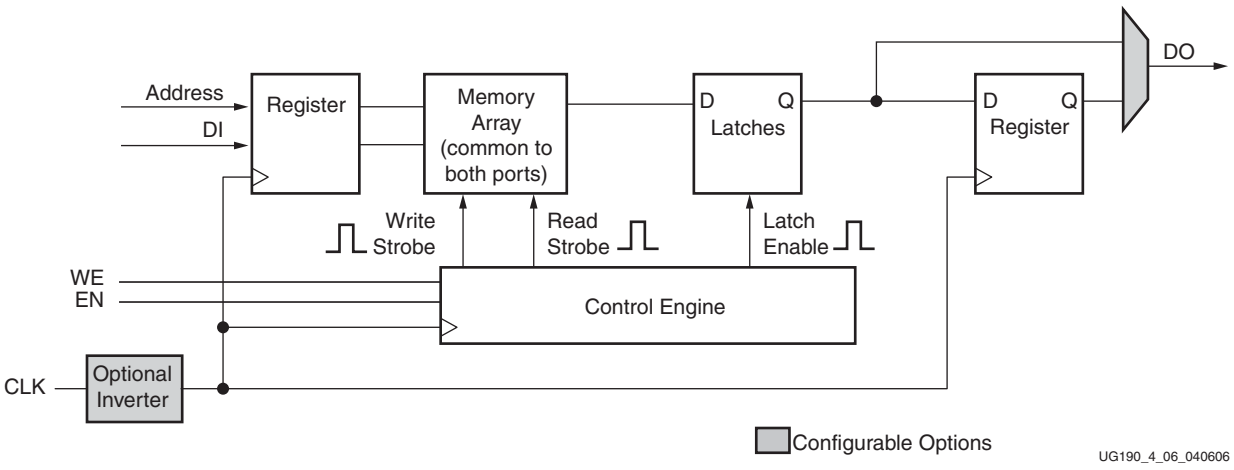


Figure 4-5: **Block RAM Logic Diagram (One Port Shown)**

Independent Read and Write Port Width Selection

Each block RAM port has control over data width and address depth (aspect ratio). The true dual-port block RAM in Virtex-5 FPGAs extends this flexibility to Read and Write where each individual port can be configured with different data bit widths. For example, port A can have a 36-bit Read width and a 9-bit Write width, and port B can have a 18-bit Read width and a 36-bit Write width. See "Block RAM Attributes," page 126.

If the Read port width differs from the Write port width, and is configured in WRITE_FIRST mode, then DO shows valid new data for all the enabled write bytes. The DO port outputs the original data stored in memory for all not enabled bytes.

Independent Read and Write port width selection increases the efficiency of implementing a content addressable memory (CAM) in block RAM. This option is available for all Virtex-5 FPGA true dual-port RAM port sizes and modes.

Simple Dual-Port Block RAM

Each 18 Kb block and 36 Kb block can also be configured in a simple dual-port RAM mode. In this mode, the block RAM port width doubles to 36 bits for the 18 Kb block RAM and 72 bits for the 36 Kb block RAM. In simple dual-port mode, independent Read and Write operations can occur simultaneously, where port A is designated as the Read port and port B as the Write port. When the Read and Write port access the same data location at the same time, it is treated as a collision, similar to the port collision in true dual-port mode. Readback through the configuration port is not supported in simple dual-port block RAM mode. Figure 4-6 shows the simple dual-port data flow.

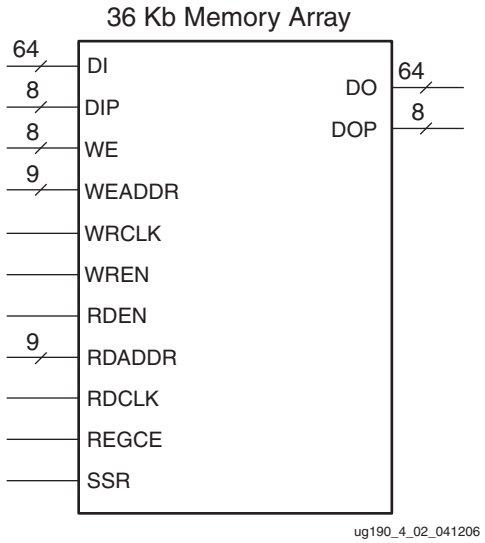


Figure 4-6: **Simple Dual-Port Data Flow**

Table 4-3: **Simple Dual-Port Names and Descriptions**

Port Names	Descriptions
DO	Data Output Bus
DOP	Data Output Parity Bus
DI	Data Input Bus
DIP	Data Input Parity Bus
RDADDR	Read Data Address Bus
RDCLK	Read Data Clock
RDEN	Read Port Enable
REGCE	Output Register Clock Enable
SSR	Synchronous Set/Reset
WE	Byte-wide Write Enable
WRADDR	Write Data Address Bus
WRCLK	Write Data Clock
WREN	Write Port Enable

Cascadable Block RAM

In the Virtex-5 block RAM architecture, two 32K x 1 RAMs can be combined to form one 64K x 1 RAM without using local interconnect or additional CLB logic resources. Any two adjacent block RAMs can be cascaded to generate a 64K x 1 block RAM. Increasing the depth of the block RAM by cascading two block RAMs is available only in the 64K x 1 mode. Further information on cascadeable block RAM is described in the "Additional RAMB18 and RAMB36 Primitive Design Considerations" section. For other wider and/or deeper sizes, consult the Creating Larger RAM Structures section. Figure 4-7 shows the block RAM with the appropriate ports connected in the Cascadable mode.

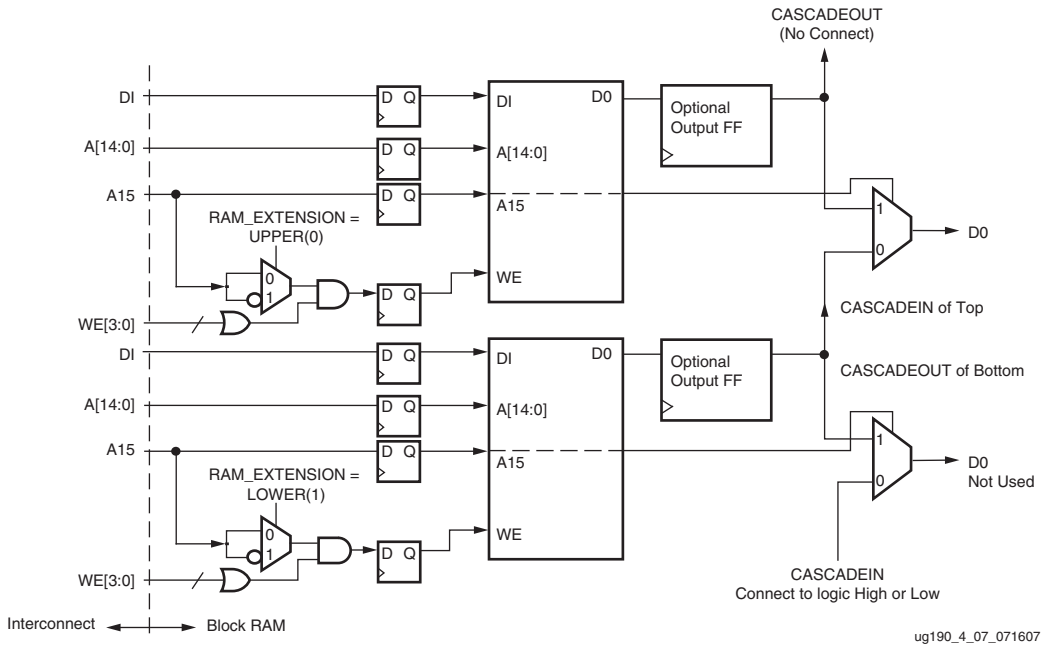


Figure 4-7: **Cascadable Block RAM**

Byte-wide Write Enable

The byte-wide write enable feature of the block RAM gives the capability to write eight bit (one byte) portions of incoming data. There are four independent byte-wide write enable inputs to the RAMB36 true dual-port RAM. There are eight independent byte-wide write enable inputs to block RAM in simple dual-port mode (RAMB36SDP). Table 4-4 summarizes the byte-wide write enables for the 36K and 18K block RAM. Each byte-wide write enable is associated with one byte of input data and one parity bit. All byte-wide write enable inputs must be driven in all data width configurations. This feature is useful when using block RAM to interface with a microprocessor. Byte-wide write enable is not available in the multirate FIFO or ECC mode. Byte-wide write enable is further described in the "Additional RAMB18 and RAMB36 Primitive Design Considerations" section. Figure 4-8 shows the byte-wide write-enable timing diagram for the RAMB36.

Table 4-4: **Available Byte-wide Write Enables**

Primitive	Maximum Bit Width	Number of Byte-wide Write Enables
RAMB36	36	4
RAMB36SDP	72	8
RAMB18	18	2
RAMB18SDP	36	4

When the RAMB36 is configured for a 36-bit or 18-bit wide data path, any port can restrict writing to specified byte locations within the data word. If configured in READ_FIRST mode, the DO bus shows the previous content of the whole addressed word. In WRITE_FIRST mode, DO shows a combination of the newly written enabled byte(s), and the initial memory contents of the unwritten bytes.

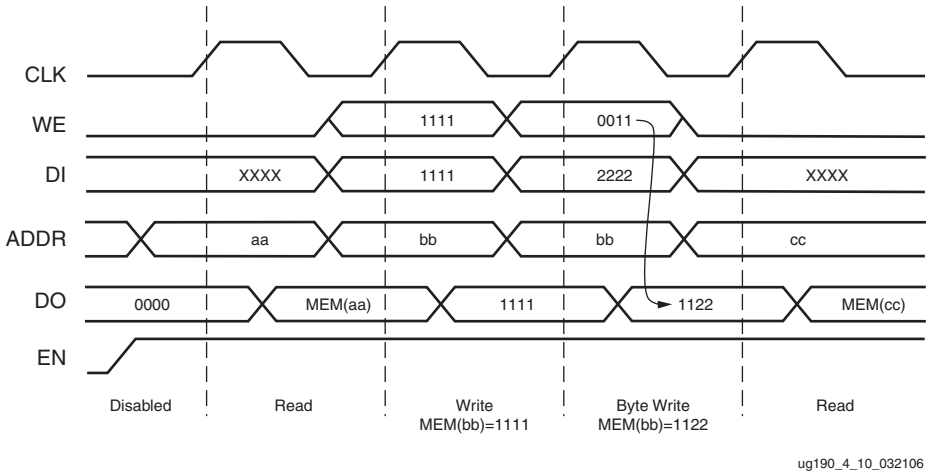


Figure 4-8: **Byte-wide Write Operation Waveforms (x36 WRITE_FIRST)**

Block RAM Error Correction Code

Both block RAM and FIFO implementations of the 36 Kb block RAM support a 64-bit Error Correction Code (ECC) implementation. The code is used to detect single and double-bit errors in block RAM data read out. Single-bit errors are then corrected in the output data.

Block RAM Library Primitives

The Virtex-5 FPGA block RAM library primitives, RAMB18 and RAMB36, are the basic building blocks for all block RAM configurations. Other block RAM primitives and macros are based on these primitives. Some block RAM attributes can only be configured using one of these primitives (e.g., pipeline register, cascade, etc.). See the "Block RAM Attributes" section.

The input and output data buses are represented by two buses for 9-bit width (8 + 1), 18-bit width (16 + 2), and 36-bit width (32 + 4) configurations. The ninth bit associated with each byte can store parity/error correction bits or serve as additional data bits. No specific function is performed on the ninth bit. The separate bus for parity bits facilitates some designs. However, other designs safely use a 9-bit, 18-bit, or 36-bit bus by merging the

regular data bus with the parity bus. Read/write and storage operations are identical for all bits, including the parity bits.

Figure 4-9 illustrates all the I/O ports of the 36 Kb true dual-port block RAM primitive (RAMB36). Table 4-5 lists these primitives.

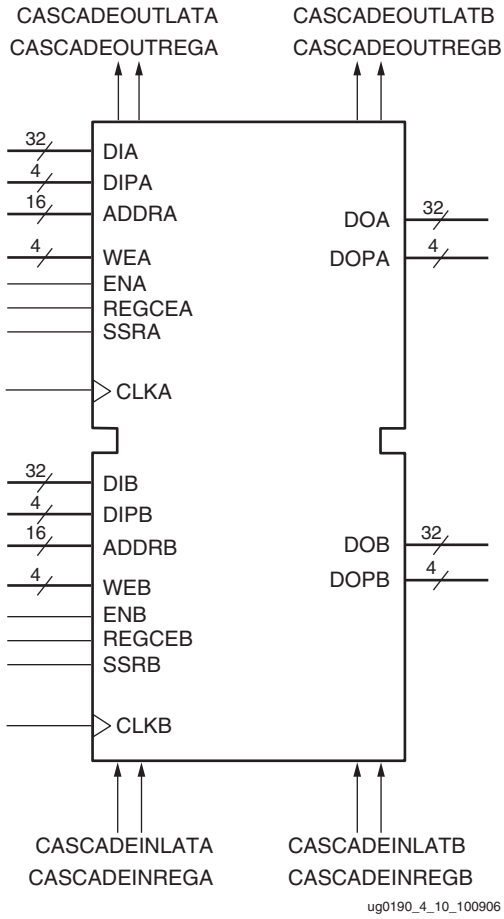


Figure 4-9: **Block RAM Port Signals (RAMB36)**

Table 4-5: **Virtex-5 FPGA Block RAM, FIFO, Simple Dual Port, and ECC Primitives**

Primitive	Description
RAMB36	Supports port widths of x1, x2, x4, x9, x18, x36
RAMB36SDP	Simple dual port (port width x72) and 64-bit ECC primitive (see Figure 4-29)
FIFO36	Supports port widths of x4, x9, x18, x36
FIFO36_72	FIFO (port width x72), optional ECC support
RAMB18	Supports port widths of x1, x2, x4, x9, x18
RAMB18SDP	Simple dual port (port width x36)
FIFO18	Supports port widths of x4, x9, x18

Table 4-5: **Virtex-5 FPGA Block RAM, FIFO, Simple Dual Port, and ECC Primitives**

Primitive	Description
FIFO18_36	FIFO (port width x36)

Notes:

1. All eight primitives are described in the software Libraries guide as well as the language templates.

Block RAM Port Signals

Each block RAM port operates independently of the other while accessing the same set of 36K-bit memory cells.

Clock - CLK[A|B]

Each port is fully synchronous with independent clock pins. All port input pins have setup time referenced to the port CLK pin. The output data bus has a clock-to-out time referenced to the CLK pin. Clock polarity is configurable (rising edge by default).

Enable - EN[A|B]

The enable pin affects the read, write, and set/reset functionality of the port. Ports with an inactive enable pin keep the output pins in the previous state and do not write data to the memory cells. Enable polarity is configurable (active High by default).

Byte-wide Write Enable - WE[A|B]

To write the content of the data input bus into the addressed memory location, both EN and WE must be active within a set-up time before the active clock edge. The output latches are loaded or not loaded according to the write configuration (WRITE_FIRST, READ_FIRST, NO_CHANGE). When inactive, a read operation occurs, and the contents of the memory cells referenced by the address bus appear on the data-out bus, regardless of the write mode attribute. Write enable polarity is not configurable (active High).

Register Enable - REGCE[A|B]

The register enable pin (REGCE) controls the optional output register. When the RAM is in register mode, REGCE = 1 registers the output into a register at a clock edge. The polarity of REGCE is not configurable (active High).

Set/Reset - SSR[A|B]

In latch mode, the SSR pin forces the data output latches, to contain the value SRVAL. See "Block RAM Attributes," page 126. When the optional output registers are enabled, the data output registers can also be forced by the SSR pin to contain the value SRVAL. SSR does not affect the latched value. The data output latches or output registers are synchronously asserted to 0 or 1, including the parity bit. Each port has an independent SRVAL[A | B] attribute of 36 bits. This operation does not affect RAM memory cells and does not disturb write operations on the other port. Similar to the read and write operation, the set/reset function is active only when the enable pin of the port is active. Set/reset polarity is configurable (active High by default).

Address Bus - ADDR[A|B]<13:#><14:#><15:#>

The address bus selects the memory cells for read or write. The data bit width of the port determines the required address bus width for a single RAMB18 or RAMB36, as shown in Table 4-6 and Table 4-7.

Table 4-6: **Port Aspect Ratio for RAMB18 and RAMB18SDP**

Port Data Width	Port Address Width	Depth	ADDR Bus	DI Bus / DO Bus	DIP Bus / DOP Bus
1	14	16,384	<13:0>	<0>	NA
2	13	8,192	<13:1>	<1:0>	NA
4	12	4,096	<13:2>	<3:0>	NA
9	11	2,048	<13:3>	<7:0>	<0>
18	10	1,024	<13:4>	<15:0>	<1:0>
36 (RAMB18SDP)	9	512	<13:5>	<31:0>	<3:0>

Table 4-7: **Port Aspect Ratio for RAMB36**

Port Data Width	Port Address Width	Depth	ADDR Bus	DI Bus / DO Bus	DIP Bus / DOP Bus
1	15	32,768	<14:0>	<0>	NA
2	14	16,384	<14:1>	<1:0>	NA
4	13	8,192	<14:2>	<3:0>	NA
9	12	4,096	<14:3>	<7:0>	<0>
18	11	2,048	<14:4>	<15:0>	<1:0>
36	10	1,024	<14:5>	<31:0>	<3:0>
72 (RAMB36SDP)	9	512	<14:6>	<63:0>	<7:0>
1 (Cascade)	16	65536	<15:0>	<0>	NA

For cascadable block RAM using the RAMB36, the data width is one bit, and the address bus is 16 bits <15:0>. The address bit 15 is only used in cascadable block RAM. For non-cascading block RAM, connect High.

Data and address pin mapping is further described in the "Additional RAMB18 and RAMB36 Primitive Design Considerations"section.

Data-In Buses - DI[A|B]<#:0> & DIP[A|B]<#:0>

Data-in buses provide the new data value to be written into RAM. The regular data-in bus (DI), plus the parity data-in bus (DIP) when available, have a total width equal to the port width. For example the 36-bit port data width is represented by DI<31:0> and DIP<3:0>, as shown in Table 4-6 and Table 4-7.

Data-Out Buses - DO[A|B]<#:0> and DOP[A|B]<#:0>

Data-out buses reflect the contents of memory cells referenced by the address bus at the last active clock edge during a read operation. During a write operation (WRITE_FIRST or READ_FIRST configuration), the data-out buses reflect either the data being written or the stored value before write. During a write operation in NO_CHANGE mode, data-out buses are not changed. The regular data-out bus (DO) plus the parity data-out bus (DOP) (when available) have a total width equal to the port width, as shown in Table 4-6 and Table 4-7.

Cascade In - CASCADEINLAT[A|B] and CASCADEINREG[A|B]

The CASCADEIN pins are used to connect two block RAMs to form the 64K x 1 mode (Figure 4-10.) This pin is used when the block RAM is the UPPER block RAM, and is connected to the CASCADEOUT pins of the LOWER block RAM of the same port. When cascade mode is not used, this pin does not need to be connected. Refer to the "Cascadable Block RAM" for further information.

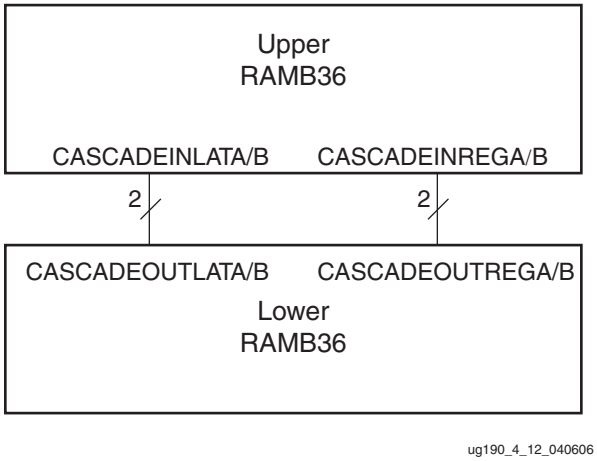


Figure 4-10: **Two RAMB36s Cascaded**

Cascade Out - CASCADEOUTLAT[A|B] and CASCADEOUTREG[A|B]

The CASCADEOUT pins are used to connect two block RAMs to form the 64K x 1 mode. This pin is used when the block RAM is the LOWER block RAM, and is connected to the CASCADEIN pins of the UPPER block RAM of the same port. When cascade mode is not used, this pin does not need to be connected. Refer to the "Cascadable Block RAM" for further information.

Inverting Control Pins

For each port, the six control pins (CLK, EN, and SSR) each have an individual inversion option. EN and SSR control signals can be configured as active High or Low, and the clock can be active on a rising or falling edge (active High on rising edge by default), without requiring other logic resources.

GSR

The global set/reset (GSR) signal of a Virtex-5 device is an asynchronous global signal that is active at the end of device configuration. The GSR can also restore the initial Virtex-5 device state at any time. The GSR signal initializes the output latches to the INIT (simple dual port), or to the INIT_A and INIT_B value (true dual port.) See "Block RAM Attributes." A GSR signal has no impact on internal memory contents. Because it is a global signal, the GSR has no input pin at the functional level (block RAM primitive).

Unused Inputs

Unused data and/or address inputs should be connected High.

Block RAM Address Mapping

Each port accesses the same set of 18,432 or 36,864 memory cells using an addressing scheme dependent on whether it is a RAMB18 or RAMB36. The physical RAM locations addressed for a particular width are determined using the following formula (of interest only when the two ports use different aspect ratios):

```
END = ((ADDR + 1) × Width) -1
START = ADDR × Width
```

Table 4-8 shows low-order address mapping for each port width.

Table 4-8: **Port Address Mapping**

Port Width	Parity Locations	Data Locations
1	N.A.	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
2		15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
4		7 6 5 4 3 2 1 0
8 + 1	3 2 1 0	3 2 1 0
16 + 2	1 0	1 0
32 + 4	0	0

Block RAM Attributes

All attribute code examples are discussed in the "Block RAM Initialization in VHDL or Verilog Code" section. Further information on using these attributes is available in the "Additional RAMB18 and RAMB36 Primitive Design Considerations" section.

Content Initialization - INIT_xx

INIT_xx attributes define the initial memory contents. By default, block RAM memory is initialized with all zeros during the device configuration sequence. The 64 initialization attributes from INIT_00 through INIT_3F for the RAMB18, and the 128 initialization attributes from INIT_00 through INIT_7F for the RAMB36 represent the regular memory contents. Each INIT_xx is a 64-digit hex-encoded bit vector. The memory contents can be partially initialized and are automatically completed with zeros.

The following formula is used for determining the bit positions for each INIT_xx attribute.

Given yy = conversion hex-encoded to decimal (xx), INIT_xx corresponds to the memory cells as follows:

- from [(yy + 1) × 256] – 1
- to (yy) × 256

For example, for the attribute INIT_1F, the conversion is as follows:

- yy = conversion hex-encoded to decimal (xx) "1F" = 31
- from [(31+1) × 256] – 1 = 8191
- to 31 × 256 = 7936

More examples are given in Table 4-9.

Table 4-9: **Block RAM Initialization Attributes**

Attribute	Memory Location	
	From	To
INIT_00	255	0
INIT_01	511	256
INIT_02	767	512
…	…	…
INIT_0E	3839	3584
INIT_0F	4095	3840
INIT_10	4351	4096
…	…	…
INIT_1F	8191	7936
INIT_20	8447	8192
…	…	…
INIT_2F	12287	12032
INIT_30	12543	12288
…	…	…
INIT_3F	16383	16128
…	…	…
INIT_7F	32767	32512

Content Initialization - INITP_xx

INITP_xx attributes define the initial contents of the memory cells corresponding to DIP/DOP buses (parity bits). By default these memory cells are also initialized to all zeros. The initialization attributes represent the memory contents of the parity bits. The eight initialization attributes are INITP_00 through INITP_07 for the RAMB18. The 16 initialization attributes are INITP_00 through INITP_0F for the RAMB36. Each INITP_xx is a 64-digit hex-encoded bit vector with a regular INIT_xx attribute behavior. The same formula can be used to calculate the bit positions initialized by a particular INITP_xx attribute.

Output Latches Initialization - INIT (INIT_A or INIT_B)

The INIT (single-port) or INIT_A and INIT_B (dual-port) attributes define the output latches or output register values after configuration. The width of the INIT (INIT_A and INIT_B) attribute is the port width, as shown in Table 4-10. These attributes are hex-encoded bit vectors, and the default value is 0. In cascade mode, both the upper and lower block RAM should be initialized to the same value.

Output Latches/Registers Synchronous Set/Reset (SRVAL_[A|B])

The SRVAL (single-port) or SRVAL_A and SRVAL_B (dual-port) attributes define output latch values when the SSR input is asserted. The width of the SRVAL (SRVAL_A and SRVAL_B) attribute is the port width, as shown in Table 4-10. These attributes are hex-encoded bit vectors and the default value is 0. This attribute sets the value of the output register when the optional output register attribute is set. When the register is not used, the latch gets set to the SRVAL instead. In the 36-bit mode, SRVAL[35:32] corresponds to DP[3:0].

Table 4-10: **Port Width Values**

Port Data Width	DOP Bus	DO Bus	INIT / SRVAL
1	NA	<0>	1
2	NA	<1:0>	2
4	NA	<3:0>	4
9	<0>	<7:0>	(1 + 8) = 9
18	<1:0>	<15:0>	(2 + 16) = 18
36	<3:0>	<31:0>	(4 + 32) = 36

Optional Output Register On/Off Switch - DO[A|B]_REG

This attribute sets the number of pipeline register at A/B output of the block RAM. The valid values are 0 (default) or 1.

Extended Mode Address Determinant - RAM_EXTENSION_[A|B]

This attribute determines whether the block RAM of interest has its A/B port as UPPER/LOWER address when using the cascade mode. Refer to the "Cascadable Block RAM" section. When the block RAM is not used in cascade mode, the default value is NONE.

Read Width - READ_WIDTH_[A|B]

This attribute determines the A/B read port width of the block RAM. The valid values are: 0 (default), 1, 2, 4, 9, 18, and 36.

Write Width - WRITE_WIDTH_[A|B]

This attribute determines the A/B write port width of the block RAM. The valid values are: 0 (default), 1, 2, 4, 9, 18, and 36.

Write Mode - WRITE_MODE_[A|B]

This attribute determines the write mode of the A/B input ports. The possible values are WRITE_FIRST (default), READ_FIRST, and NO_CHANGE. Additional information on the write modes is in the "Write Modes" section.

Block RAM Location Constraints

Block RAM instances can have LOC properties attached to them to constrain placement. Block RAM placement locations differ from the convention used for naming CLB locations, allowing LOC properties to transfer easily from array to array.

The LOC properties use the following form:

```
LOC = RAMB36_X#Y#
```

The RAMB36_X0Y0 is the bottom-left block RAM location on the device. If RAMB36 is constrained to RAMB36_X#Y#, the FIFO cannot be constrained to FIFO36_X#Y# since they share a location.

Two RAMB18s can be placed in the same RAMB36 location by using the BEL UPPER/LOWER constraint:

```
inst "my_ramb18" LOC = RAMB36_X0Y0 | BEL = UPPER
inst "my_ramb18" LOC = RAMB36_X0Y0 | BEL = LOWER
```

In addition, one FIFO18 and one RAMB16 can be placed in the same RAMB36 location, no BEL constraint is required:

```
inst "my_fifo18" LOC = RAMB36_X0Y0
inst "my_ramb18" LOC = RAMB36_X0Y0
```

Block RAM Initialization in VHDL or Verilog Code

Block RAM memory attributes and content can be initialized in VHDL or Verilog code for both synthesis and simulation by using generic maps (VHDL) or defparams (Verilog) within the instantiated component. Modifying the values of the generic map or defparam will effect both the simulation behavior and the implemented synthesis results. The Virtex-5 FPGA Libraries Guide includes the code to instantiate the RAMB36 primitive.

Additional RAMB18 and RAMB36 Primitive Design Considerations

The RAMB18 and RAMB36 primitives are integral in the Virtex-5 FPGA block RAM solution.

Optional Output Registers

Optional output registers can be used at either or both A | B output ports of RAMB18 and RAMB36. The choice is made using the DO[A | B]_REG attribute. The two independent clock enable pins are REGCE[A | B]. When using the optional output registers at port [A | B], assertion of the synchronous set/reset (SSR) pin of ports [A | B] causes the value specified by the attribute SRVAL to be registered at the output. Figure 4-5 shows an optional output register.

Independent Read and Write Port Width

To specify the port widths using the dual-port mode of the block RAM, designers must use the READ_WIDTH_[A | B] and WRITE_WIDTH_[A | B] attributes. The following rules should be considered:

- Designing a single port block RAM requires the port pair widths of one write and one read to be set (e.g., READ_WIDTH_A and WRITE_WIDTH_A).
- Designing a dual-port block RAM requires all port widths to be set.
- When using these attributes, if both write ports or both read ports are set to 0, the Xilinx ISE tools will not implement the design. In simple dual-port mode, the port width is fixed and the read port width is equal to the write port width. The RAMB18 has a data port width of 36, while the RAMB36 has a data port width of 72.

RAMB18 and RAMB36 Port Mapping Design Rules

The Virtex-5 FPGA block RAM are configurable to various port widths and sizes. Depending on the configuration, some data pins and address pins are not used. Table 4-6, page 124 shows the pins used in various configurations. In addition to the information in Table 4-6, the following rules are useful to determine port connections for the RAMB36:

1. When using RAMB36, if the DI[A | B] pins are less than 32-bits wide, concatenate (32 – DI_BIT_WIDTH) logic zeros to the front of DI[A | B].
2. If the DIP[A | B] pins are less than 4-bits wide, concatenate (4 – DIP_BIT_WIDTH) logic zeros to the front of DIP[A | B]. DIP[A | B] can be left unconnected when not in use.
3. DO[A | B] pins must be 32-bits wide. However, valid data are only found on pins DO_BIT_WIDTH – 1 down to 0.
4. DOP[A | B] pins must be 4-bits wide. However, valid data are only found on pins DOP_BIT_WIDTH – 1 down to 0. DOP[A | B] can be left unconnected when not in use.
5. ADDR[A | B] pins must be 16-bits wide. However, valid addresses for non-cascadable block RAM are only found on pin 14 to (15 – address width). The remaining pins, including pin 15, should be tied High. Address width is defined in Table 4-6, page 124.

Cascadeable Block RAM

To use the cascadeable block RAM feature:

1. Two RAMB36 primitives must be instantiated.
2. Set the RAM_EXTENSION_A and RAM_EXTENSION_B attribute for one RAMB36 to UPPER, and another to LOWER.
3. Connect the upper RAMB36's CASCADEINA and CASCADEINB ports to the CASCADEOUTA and CASCADEOUTB ports of the lower RAMB36. The CASCADEOUT ports for the upper RAMB36 do not require a connection. Connect the CASCADEIN ports for the lower RAMB36 to either logic High or Low.
4. The data output ports of the lower RAMB36 are not used. These pins are unconnected.
5. If placing location constraints on the two RAMB36s, they must be adjacent. If no location constraint is specified, the Xilinx ISE software will automatically manage the RAMB36 locations.
6. The address pins ADDR[A | B] must be 16 bits wide. Both read and write ports must be one bit wide.

Figure 4-7 shows the cascadeable block RAM.

Byte-wide Write Enable

The following rules should be considered when using the byte-wide write enable feature:

- In x36 mode, WE[3:0] is connected to the four user WE inputs.
- In x18 mode, WE[0] and WE[2] are connected and driven by the user WE[0], while WE[1], and WE[3] are driven by the user WE[1].
- In x9, x4, x2, x1, WE[3:0] are all connected to a single user WE.
- In x72 simple dual-port mode, WE[7:0] is connected to the eight user WE inputs.

Additional Block RAM Primitives

In addition to RAMB18 and RAMB36, there are other block RAM primitives available for specific implementations. RAMB18SDP and RAMB36SDP implement the simple dual-port mode configurations of the block RAM. Figure 4-3, page 119 shows the ports available for the 18 Kb block RAM configured in simple dual-port mode.

The RAMB36SDP can also be configured for the built-in block RAM ECC. For more information on RAMB36SDP with the ECC feature, see "Built-in Error Correction," page 157.

Block RAM Applications

Creating Larger RAM Structures

Block RAM columns have special routing to create wider/deeper blocks using 36 Kb block RAMs with minimal routing delays. Wider or deeper RAM structures are achieved with a smaller timing penalty than is encountered when using normal routing resources.

The Xilinx CORE Generator program offers the designer an easy way to generate wider and deeper memory structures using multiple block RAM instances. This program outputs VHDL or Verilog instantiation templates and simulation models, along with an EDIF file for inclusion in a design.

Block RAM SSR in Register Mode

A block RAM SSR in register mode can be used to control the output register as a true pipeline register independent of the block RAM. As shown in Figure 4-11, block RAM can be read and written independent of register enable or set/reset. In register mode SSR sets DO to the SRVAL and data can be read from the block RAM to DBRAM. Data at DBRAM can be clocked out (DO) on the next cycle. The timing diagrams in Figure 4-12 and Figure 4-13 show different cases of the SSR operation.

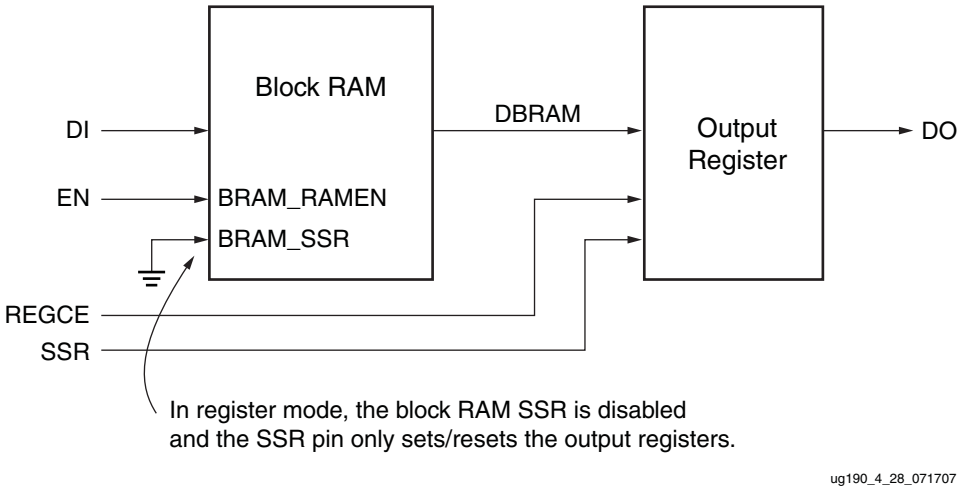


Figure 4-11: **Block RAM SSR in Register Mode**

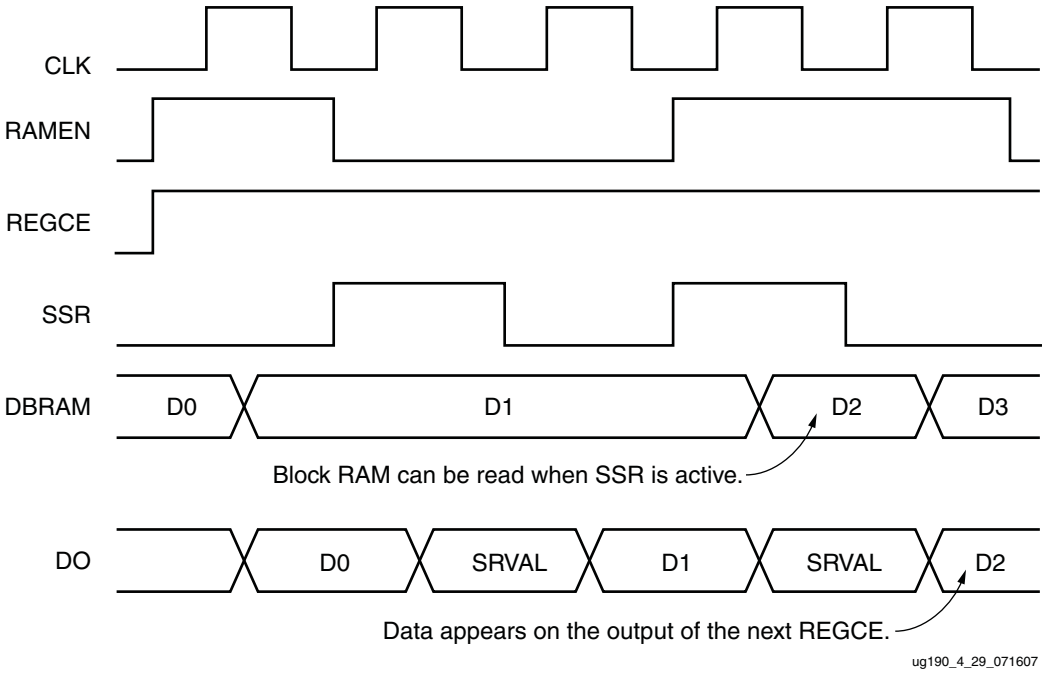


Figure 4-12: **SSR Operation in Register Mode with REGCE High**

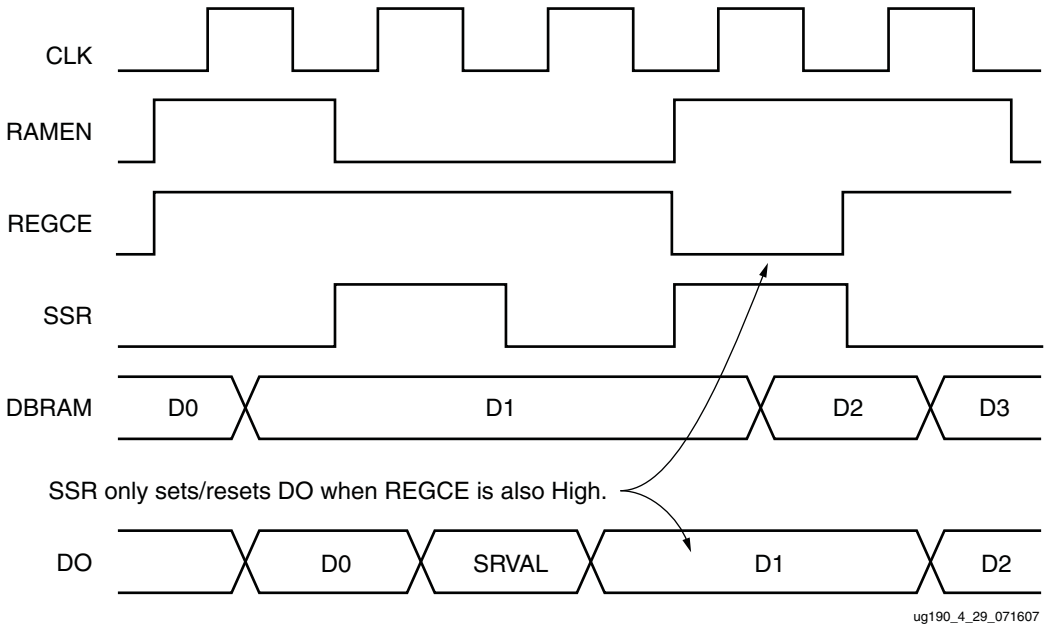


Figure 4-13: **SSR Operation in Register Mode with Variable REGCE**

Block RAM Timing Model

This section describes the timing parameters associated with the block RAM in Virtex-5 devices (illustrated in Figure 4-14). The switching characteristics section in the *Virtex-5 FPGA Data Sheet* and the Timing Analyzer (TRCE) report from Xilinx software are also available for reference.

Block RAM Timing Parameters

Table 4-11 shows the Virtex-5 FPGA block RAM timing parameters.

Table 4-11: **Block RAM Timing Parameters**

Parameter	Function	Control Signal	Description
Setup and Hold Relative to Clock (CLK)			
T_{RxCK_x} = Setup time (before clock edge) and T_{RCKx_x} = Hold time (after clock edge)			
T_{RCCK_ADDR}	Address inputs	ADDR	Time before the clock that address signals must be stable at the ADDR inputs of the block RAM.(1)
T_{RCKC_ADDR}			Time after the clock that address signals must be stable at the ADDR inputs of the block RAM.(1)
T_{RDCK_DI}	Data inputs	DI	Time before the clock that data must be stable at the DI inputs of the block RAM.
T_{RCKD_DI}			Time after the clock that data must be stable at the DI inputs of the block RAM.
T_{RCCK_EN}	Enable	EN	Time before the clock that the enable signal must be stable at the EN input of the block RAM.
T_{RCKC_EN}			Time after the clock that the enable signal must be stable at the EN input of the block RAM.
T_{RCCK_SSR}	Synchronous Set/Reset	SSR	Time before the clock that the synchronous set/reset signal must be stable at the SSR input of the block RAM.
T_{RCKC_SSR}			Time after the clock that the synchronous set/reset signal must be stable at the SSR input of the block RAM.
T_{RCCK_WE}	Write Enable	WE	Time before the clock that the write enable signal must be stable at the WE input of the block RAM.
T_{RCKC_WE}			Time after the clock that the write enable signal must be stable at the WE input of the block RAM.
T_{RCCK_REGCE}	Optional Output Register Enable	REGCE	Time before the CLK that the register enable signal must be stable at the REGCE input of the block RAM.
T_{RCKC_REGCE}			Time after the clock that the register enable signal must be stable at the REGCE input of the block RAM.
Clock to Out Delays			
T_{RCKO_DO} (latch mode)	Clock to Output	CLK to DO	Time after the clock that the output data is stable at the DO outputs of the block RAM (without output register).
T_{RCKO_DO} (register mode)	Clock to Output	CLK to DO	Time after the clock that the output data is stable at the DO outputs of the block RAM (with output register).

Notes:

1. While EN is active, ADDR inputs must be stable during the entire setup/hold time window, even if WE is inactive. Violating this requirement can result in block RAM data corruption. If ADDR timing could violate the specified requirements, EN must be inactive (disabled).

Block RAM Timing Characteristics

The timing diagram in Figure 4-14 describes a single-port block RAM in write-first mode without the optional output register. The timing for read-first and no-change modes are similar. For timing using the optional output register, an additional clock latency appears at the DO pin. These waveforms correspond to latch mode when the optional output pipeline register is not used.

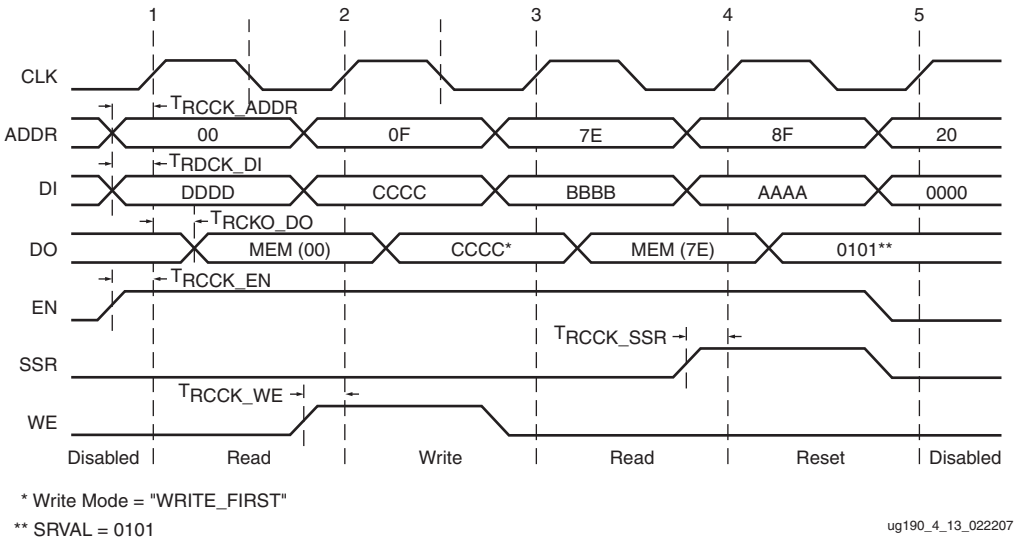


Figure 4-14: **Block RAM Timing Diagram**

At time 0, the block RAM is disabled; EN (enable) is Low.

Clock Event 1

Read Operation

During a read operation, the contents of the memory at the address on the ADDR inputs remain unchanged.

- T_{RCCK_ADDR} before clock event 1, address 00 becomes valid at the ADDR inputs of the block RAM.
- At time T_{RCCK_EN} before clock event 1, enable is asserted High at the EN input of the block RAM, enabling the memory for the READ operation that follows.
- At time T_{RCKO_DO} after clock event 1, the contents of the memory at address 00 become stable at the DO pins of the block RAM.
- Whenever EN is asserted, all address changes must meet the specified setup and hold window. Asynchronous address changes can affect the memory content and block RAM functionality in an unpredictable way.

Clock Event 2

Write Operation

During a write operation, the content of the memory at the location specified by the address on the ADDR inputs is replaced by the value on the DI pins and is immediately reflected on the output latches (in WRITE_FIRST mode); when Write Enable (WE) is High.

- At time T_{RCCK_ADDR} before clock event 2, address 0F becomes valid at the ADDR inputs of the block RAM.
- At time T_{RDCK_DI} before clock event 2, data CCCC becomes valid at the DI inputs of the block RAM.
- At time T_{RCCK_WE} before clock event 2, write enable becomes valid at the WE following the block RAM.
- At time T_{RCKO_DO} after clock event 2, data CCCC becomes valid at the DO outputs of the block RAM.

Clock Event 4

SSR (Synchronous Set/Reset) Operation

During an SSR operation, initialization parameter value SRVAL is loaded into the output latches of the block RAM. The SSR operation does NOT change the contents of the memory and is independent of the ADDR and DI inputs.

- At time T_{RCCK_SSR} before clock event 4, the synchronous set/reset signal becomes valid (High) at the SSR input of the block RAM.
- At time T_{RCKO_DO} after clock event 4, the SRVAL 0101 becomes valid at the DO outputs of the block RAM.

Clock Event 5

Disable Operation

Deasserting the enable signal EN disables any write, read, or SSR operation. The disable operation does NOT change the contents of the memory or the values of the output latches.

- At time T_{RCCK_EN} before clock event 5, the enable signal becomes invalid (Low) at the EN input of the block RAM.
- After clock event 5, the data on the DO outputs of the block RAM is unchanged.

Block RAM Timing Model

Figure 4-15 illustrates the delay paths associated with the implementation of block RAM. This example takes the simplest paths on and off chip (these paths can vary greatly depending on the design). This timing model demonstrates how and where the block RAM timing parameters are used.

- **NET** = Varying interconnect delays
- $\mathbf{T_{IOPI}}$ = Pad to I-output of IOB delay
- $\mathbf{T_{IOOP}}$ = O-input of IOB to pad delay
- $\mathbf{T_{BCCKO\_O}}$ = BUFGCTRL delay

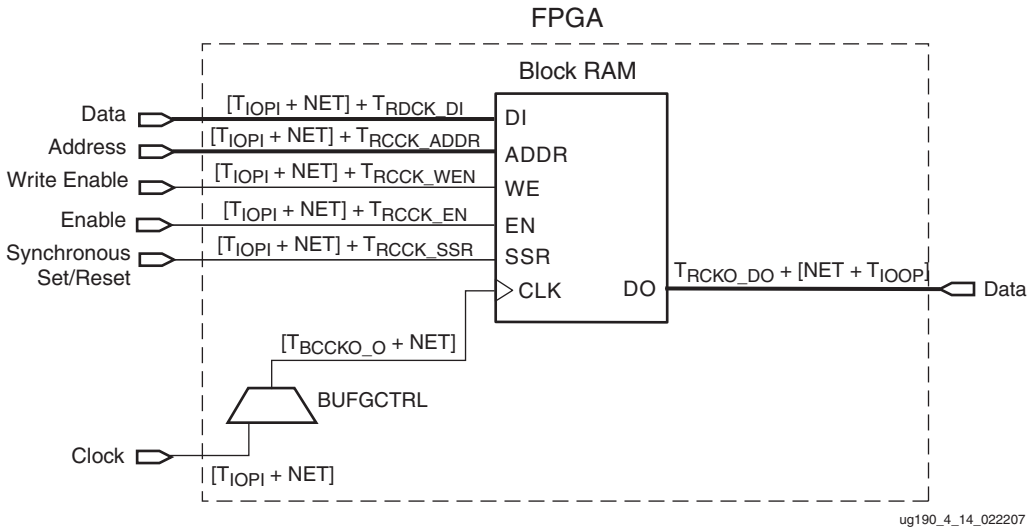


Figure 4-15: **Block RAM Timing Model**

Block RAM Retargeting

Table 4-12 suggests the most appropriate primitives to choose when mapping a Virtex-4 FPGA block RAM design in a new Virtex-5 FPGA design.

Table 4-12: **Block RAM Retargeting**

Virtex-4 Block RAM			18k Virtex-5 Block RAM			36k Virtex-5 Block RAM		
Primitive	**Depth**	**Port Width R/W**	**Primitive**	**Depth**	**Port Width R/W**	**Primitive**	**Depth**	**Port Width R/W**
RAMB16 True dual port	1k to 16k	1, 2, 4, 9, 18	RAMB18	1k to 16k	1, 2, 4, 9, 18	RAMB36	2k to 32k	1, 2, 4, 9, 18
RAMB16 True dual port	512	36/36	N/A	N/A	N/A	RAMB36	1k	36/36
RAMB16 Simple dual port	512	36/36	RAMB18 Simple dual port	512	36/36	RAMB36 Simple dual port	1k	36/36
RAMB16 Simple dual port		Variable	Use closest RAMB18 True dual-port	N/A	N/A	Use closest RAMB36 True dual port	N/A	N/A
CASC of two RAMB16s	32k	1	N/A	N/A	N/A	RAMB36	32k	1

Built-in FIFO Support

Many FPGA designs use block RAMs to implement FIFOs. In the Virtex-5 architecture, dedicated logic in the block RAM enables users to easily implement synchronous or multirate (asynchronous) FIFOs. This eliminates the need for additional CLB logic for counter, comparator, or status flag generation, and uses just one block RAM resource per FIFO. Both standard and first-word fall-through (FWFT) modes are supported.

In the Virtex-5 architecture, the FIFO can be configured as a 18 Kb or 36 Kb memory. For the 18 Kb mode, the supported configurations are 4K x 4, 2K x 9, 1K x 18, and 512 x 36. The supported configurations for the 36 Kb FIFO are 8K x 4, 4K x 9, 2K x 18, 1K x 36, and 512 x 72.

The block RAM can be configured as first-in/first-out (FIFO) memory with common or independent read and write clocks. Port A of the block RAM is used as a FIFO read port, and Port B is a FIFO write port. Data is read from the FIFO on the rising edge of read clock and written to the FIFO on the rising edge of write clock. Independent read and write port width selection is not supported in FIFO mode without the aid of external CLB logic.

Multirate FIFO

The multirate FIFO offers a very simple user interface. The design relies on free-running write and read clocks, of identical or different frequencies up to the specified maximum frequency limit. The design avoids any ambiguity, glitch, or metastable problems, even when the two frequencies are completely unrelated.

The write operation is synchronous, writing the data word available at DI into the FIFO whenever WREN is active a set-up time before the rising WRCLK edge.

The read operation is also synchronous, presenting the next data word at DO whenever the RDEN is active one set-up time before the rising RDCLK edge.

Data flow control is automatic; the user need not be concerned about the block RAM addressing sequence, although WRCOUNT and RDCOUNT are also brought out, if needed for special applications.

The user must, however, observe the FULL and EMPTY flags, and stop writing when FULL is High, and stop reading when EMPTY is High. If these rules are violated, an active WREN while FULL is High will activate the WRERR flag, and an active RDEN while EMPTY is High will activate the RDERR flag. In either violation, the FIFO content will, however, be preserved, and the address counters will stay valid.

Programmable ALMOSTFULL and ALMOSTEMPTY flags are brought out to give the user an early warning when the FIFO is approaching its limits. Both these flag values can be set by configuration to (almost) anywhere in the FIFO address range.

Two operating modes affect the reading of the first word after the FIFO is emptied:

- In standard mode, the first word written into an empty FIFO will appear at DO after the user has activated RDEN. The user must pull the data out of the FIFO.
- In FWFT mode, the first word written into an empty FIFO will automatically appear at DO without the user activating RDEN. The next RDEN will then pull the subsequent data word onto DO.
- Standard and FWFT mode differ only in the reading of the first word entry after the FIFO is empty.

Use the EN_SYN = FALSE setting in the following cases:

- when the clocks are asynchronous
- when the frequencies of the two clocks are the same but the phase is different
- when one frequency is a multiple of the other.

Synchronous FIFO

Virtex-4 FPGA designs used the same FIFO logic for multirate and synchronous FIFOs, thus flag latency in synchronous FIFOs can vary. By setting the EN_SYN attribute to TRUE when using Virtex-5 FPGA synchronous FIFOs, any clock cycle latency when asserting or deasserting flags is eliminated.

First-word fall-through (FWFT) mode is only supported in the multirate FIFO (EN_SYN = FALSE). Table 4-13 shows the FIFO capacity in the two modes.

Table 4-13: **FIFO Capacity**

Standard Mode		FWFT Mode	
18 Kb FIFO	**36 Kb FIFO**	**18 Kb FIFO**	**36 Kb FIFO**
4k + 1 entries by 4 bits	8k + 1 entries by 4 bits	4k + 2 entries by 4 bits	8k + 2 entries by 4 bits
2k + 1 entries by 9 bits	4k + 1 entries by 9 bits	2k + 2 entries by 9 bits	4k + 2 entries by 9 bits
1k + 1 entries by 18 bits	2k + 1 entries by 18 bits	1k + 2 entries by 18 bits	2k + 2 entries by 18 bits
512 + 1 entries by 36 bits	1k + 1 entries by 36 bits	512 + 2 entries by 36 bits	1k + 2 entries by 36 bits
	512 + 1 entries by 72 bits		512 + 2 entries by 72 bits

Synchronous FIFO Implementations

Table 4-14 outlines varied implementations of synchronous FIFOs. Figure 4-16 shows the timing differences.

Table 4-14: **Comparison of Synchronous FIFO Implementations**

Synchronous FIFO Implementations	Advantages	Disadvantages
EN_SYN = TRUE, DO_REG = 0	No flag uncertainty	Longer clock-to-out signals
EN_SYN = TRUE, DO_REG = 1	Faster clock-to-out signals, no flag uncertainty	Data Latency increased by one. Behaves like a synchronous FIFO with an extra data output pipeline register
EN_SYN = FALSE, DO_REG = 1 RDCLK = WRCLK	Faster clock-to-out signals. Similar to a Virtex-4 FIFO.	Falling-edge flag uncertainty. Rising-edge guaranteed on FULL and EMPTY

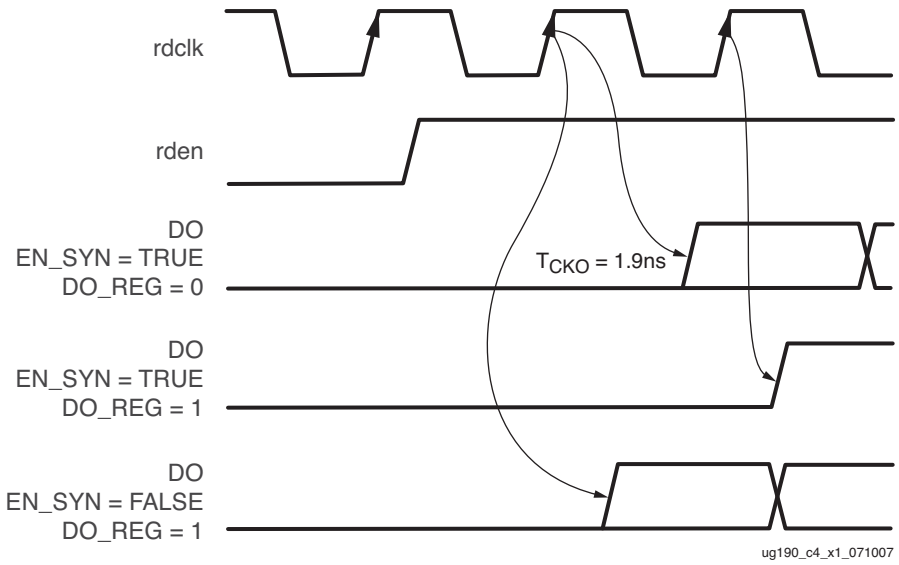


Figure 4-16: **Synchronous FIFO Data Timing Diagram**

FIFO Architecture: a Top-Level View

Figure 4-17 shows a top-level view of the Virtex-5 FIFO architecture. The read pointer, write pointer, and status flag logic are dedicated for FIFO use only.

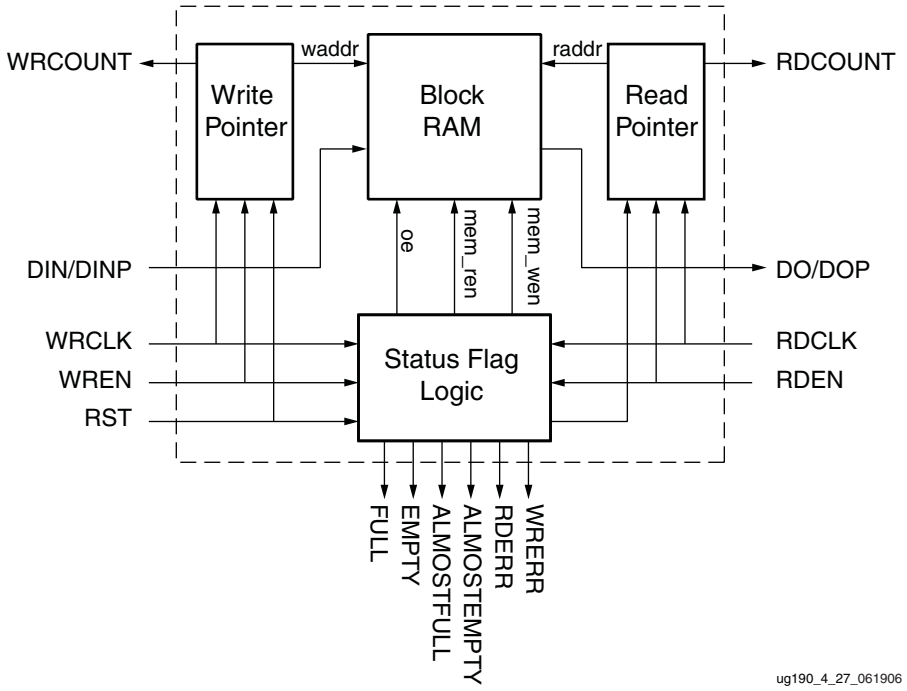


Figure 4-17: **Top-Level View of FIFO in Block RAM**

FIFO Primitives

Figure 4-18 shows the FIFO36 primitive.

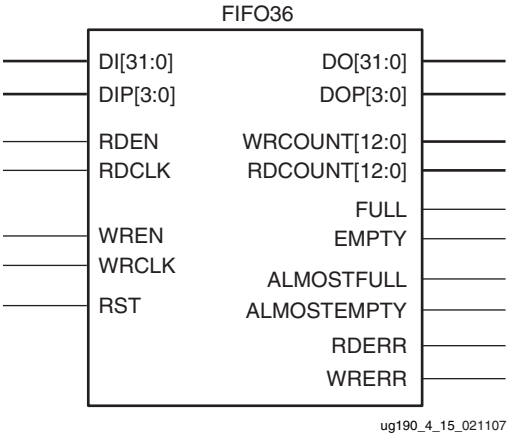


Figure 4-18: **FIFO36 Primitive**

Figure 4-19 shows the FIFO18 primitive.

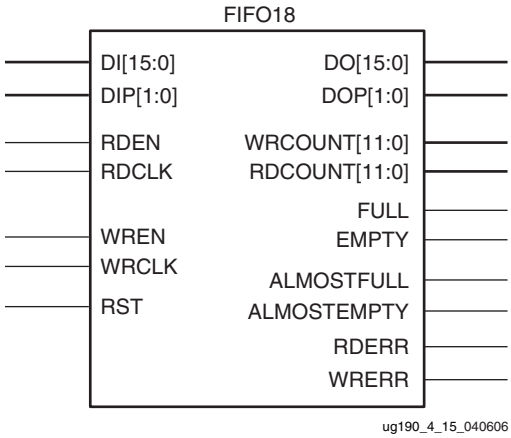


Figure 4-19: **FIFO18 Primitive**

FIFO Port Descriptions

Table 4-15 lists the FIFO I/O port names and descriptions.

Table 4-15: **FIFO I/O Port Names and Descriptions**

Port Name	Direction	Description
DI	Input	Data input.
DIP	Input	Parity-bit input.
WREN	Input	Write enable. When WREN = 1, data will be written to memory. When WREN = 0, write is disabled.
WRCLK	Input	Clock for write domain operation.
RDEN	Input	Read enable. When RDEN = 1, data will be read to output register. When RDEN = 0, read is disabled.
RDCLK	Input	Clock for read domain operation.
RESET	Input	Asynchronous reset of all FIFO functions, flags, and pointers. RESET must be asserted for three clock cycles.
DO	Output	Data output, synchronous to RDCLK.
DOP	Output	Parity-bit output, synchronous to RDCLK.
FULL	Output	All entries in FIFO memory are filled. No additional writes are accepted. Synchronous to WRCLK.
ALMOSTFULL	Output	Almost all entries in FIFO memory have been filled. Synchronous to WRCLK. The offset for this flag is user configurable. See Table 4-16 for the clock latency for flag deassertion.
EMPTY	Output	FIFO is empty. No additional reads are accepted. Synchronous to RDCLK.

Table 4-15: **FIFO I/O Port Names and Descriptions *(Continued)***

Port Name	Direction	Description
ALMOSTEMPTY	Output	Almost all valid entries in FIFO have been read. Synchronous with RDCLK. The offset for this flag is user configurable. See Table 4-16 for the clock latency for flag deassertion.
RDCOUNT	Output	The FIFO data read pointer. It is synchronous with RDCLK. The value will wrap around if the maximum read pointer value has been reached.
WRCOUNT	Output	The FIFO data write pointer. It is synchronous with WRCLK. The value will wrap around if the maximum write pointer value has been reached.
WRERR	Output	When the FIFO is full, any additional write operation generates an error flag. Synchronous with WRCLK.
RDERR	Output	When the FIFO is empty, any additional read operation generates an error flag. Synchronous with RDCLK.

FIFO Operations

Reset

Reset is an asynchronous signal for both multirate and synchronous FIFO. Reset must be asserted for three cycles to reset all read and write address counters and initialize flags after power up. Reset does not clear the memory, nor does it clear the output register. When reset is asserted High, EMPTY and ALMOST_EMPTY will be set to 1, FULL and ALMOST_FULL will be reset to 0. The reset signal must be High for at least three read clock and write clock cycles to ensure all internal states are reset to the correct values. During RESET, RDEN and WREN must be held Low.

Operating Mode

There are two operating modes in FIFO functions. They differ only in output behavior immediately after the first word is written to a previously empty FIFO.

Standard Mode

After the first word is written into an empty FIFO, the Empty flag deasserts synchronously with RDCLK. After Empty is deasserted Low and RDEN is asserted, the first word will appear at DO on the rising edge of RDCLK.

First Word Fall Through (FWFT) Mode

After the first word is written into an empty FIFO, this word automatically appears at DO before RDEN is asserted. Subsequent Read operations require Empty to be Low and RDEN to be High. Figure 4-20 illustrates the difference between standard mode and FWFT mode.

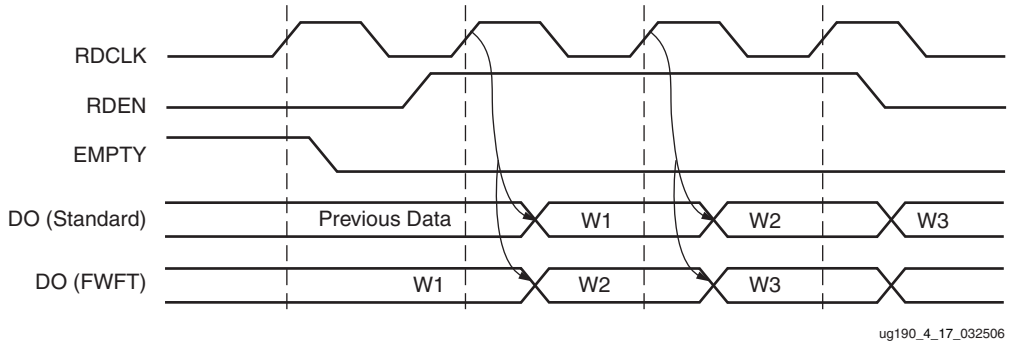


Figure 4-20: **Read Cycle Timing (Standard and FWFT Modes)**

Status Flags

Table 4-16 shows the number of clock cycles to assert or deassert each flag of a multirate FIFO. Synchronous FIFOs do not have a clock cycle latency when asserting or deasserting flags. Due to the asynchronous nature of the clocks, the simulation model only reflects the deassertion latency cycles listed.

Table 4-16: **Multirate FIFO Flag Assertion and Deassertion Latency**

Status Flag	Clock Cycle Latency(1)			
	Assertion		Deassertion	
	Standard	FWFT	Standard	FWFT
EMPTY(2)	0	0	3	4
FULL(2)	0	0	3	3
ALMOST EMPTY(3)	1	1	3	3
ALMOST FULL(3)	1	1	3	3
READ ERROR	0	0	0	0
WRITE ERROR	0	0	0	0

Notes:

1. Latency is with respect to RDCLK and WRCLK.
2. Depending on the offset between read and write clock edges, the Empty and Full flags can deassert one cycle later.
3. Depending on the offset between read and write clock edges, the Almost Empty and Almost Full flags can deassert one cycle later.

Empty Flag

The Empty flag is synchronous with RDCLK, and is asserted when the last entry in the FIFO is read. When there are no more valid entries in the FIFO queue, the read pointer will be frozen. The Empty flag is deasserted after three (in standard mode) or four (in FWFT mode) read clocks after new data is written into the FIFO.

The empty flag is used in the read clock domain. The rising edge of EMPTY is inherently synchronous with RDCLK. The empty condition can only be terminated by WRCLK, usually asynchronous to RDCLK. The falling edge of EMPTY must, therefore, artificially

be moved onto the RDCLK time domain. Since the two clocks have an unknown phase relationship, it takes several cascaded flip-flops to guarantee that such a move does not cause glitches or metastable problems. The falling edge of EMPTY is thus delayed by several RDCLK periods after the first write into the previously empty FIFO. This delay guarantees proper operation under all circumstances, and causes an insignificant loss of performance after the FIFO had gone empty.

Almost Empty Flag

The Almost Empty flag is set when the FIFO contains the number of entries specified by the ALMOST_EMPTY_OFFSET value or fewer entries. The Almost Empty flag warns the user to stop reading. It deasserts when the number of entries in the FIFO is greater than the ALMOST_EMPTY_OFFSET value plus one. Assertion and deassertion is synchronous to RDCLK. Flag latency is described in Table 4-16.

When a Virtex-5 FPGA FIFO is instantiated in FWFT mode, ALMOST_EMPTY_OFFSET must be set to a value that satisfies Equation 4-1.

$$\text{ALMOST\_EMPTY\_OFFSET} \geq 4 \times \text{Roundup}\left(\frac{\text{WRCLK frequency}}{\text{RDCLK frequency}}\right) \qquad \textit{Equation 4-1}$$

For example, if the read frequency is 1/2 the write frequency, ALMOST_EMPTY_OFFSET needs to be greater than or equal to 8. This equation also means that any time the read frequency is greater than or equal to the write frequency, any legal value of ALMOST_EMPTY_OFFSET works.

Read Error Flag

Once the Empty flag has been asserted, any further read attempts will not increment the read address pointer but will trigger the Read Error flag. The Read Error flag is deasserted when Read Enable or Empty is deasserted Low. The Read Error flag is synchronous to RDCLK.

Full Flag

The Full flag is synchronous with WRCLK, and is asserted when there are no more available entries in the FIFO queue. When the FIFO is full, the write pointer will be frozen. The Virtex-5 FPGA Full flag is deasserted three write clock cycles after two subsequent read operations. In Virtex-4 FPGA designs a Full flag is asserted one write clock cycle after the last write, and is deasserted three write clock cycle after the first read.

Write Error Flag

Once the Full flag has been asserted, any further write attempts will not increment the write address pointer but will trigger the Write Error flag. The Write Error flag is deasserted when Write Enable or Full is deasserted Low. This signal is synchronous to WRCLK.

Almost Full Flag

The Almost Full flag is set when the FIFO has the number of available empty spaces specified by the ALMOST_FULL_OFFSET value or fewer spaces. The Almost Full flag warns the user to stop writing. It deasserts when the number of empty spaces in the FIFO is greater than the ALMOST_FULL_OFFSET value plus one. Assertion and deassertion is synchronous to WRCLK. Flag latency is described in Table 4-16.

FIFO Attributes

Table 4-17 lists the FIFO18 and FIFO36 attributes. The size of the multirate FIFO can be configured by setting the DATA_WIDTH attribute. The "FIFO VHDL and Verilog Templates" section has examples for setting the attributes.

Table 4-17: **FIFO18 and FIFO36 Attributes**

Attribute Name	Type	Values	Default	Notes
ALMOST_FULL_OFFSET	13-bit HEX	See Table 4-19		Setting determines the difference between FULL and ALMOSTFULL conditions. Must be set using hexadecimal notation.
ALMOST_EMPTY_OFFSET	13-bit HEX	See Table 4-19		Setting determines the difference between EMPTY and ALMOSTEMPTY conditions. Must be set using hexadecimal notation.
FIRST_WORD_FALL_THROUGH	Boolean	FALSE, TRUE	FALSE	If TRUE, the first word written into the empty FIFO appears at the FIFO output without RDEN asserted.
DO_REG	1-bit Binary	0, 1	1	For multirate (asynchronous) FIFO, must be set to 1. For synchronous FIFO, DO_REG must be set to 0 for flags and data to follow a standard synchronous FIFO operation. When DO_REG is set to 1, effectively a pipeline register is added to the output of the synchronous FIFO. Data then has a one clock cycle latency. However, the clock-to-out timing is improved.
DATA_WIDTH	Integer	4, 9, 18, 36, 72	4	
LOC(1, 2)	String	Valid FIFO18 or FIFO36 location		Sets the location of the FIFO18 or FIFO36.
EN_SYN	Boolean	FALSE, TRUE	FALSE	When set to TRUE, ties WRCLK and RDCLK together. When set to TRUE, FWFT must be FALSE. When set to FALSE, DO_REG must be 1.

Notes:

1. If FIFO18 is constrained to FIFO18_X#Y#, then RAMB18 can not be constrained to RAMB18_X#Y# since the same location would be used.
2. If a FIFO18 is constrained to FIFO18_X#Y#, corresponding to the lower RAMB18_X#Y# of the RAMB18 pair, a RAMB18 can be constrained to the upper RAMB18_X#Y# of the pair.

FIFO Almost Full/Empty Flag Offset Range

The offset ranges for Almost Empty and Almost Full are listed in Table 4-19.

Table 4-18: **FIFO Data Depth**

Data Width		Block RAM Memory	FIFO Capacity	
FIFO18	FIFO36		Standard	FWFT
	x4	8192	8193	8194
x4	x9	4096	4097	4098
x9	x18	2048	2049	2050
x18	x36	1024	1025	1026
x36	x72	512	513	514

Notes:

1. ALMOST_EMPTY_OFFSET and ALMOST_FULL_OFFSET for any design must be less than the total FIFO depth.

Table 4-19: **FIFO Almost Full/Empty Flag Offset Range**

Data Width		ALMOST_EMPTY_OFFSET				ALMOST_FULL_OFFSET	
		Standard		FWFT(1)			
FIFO18	FIFO36	Min	Max	Min	Max	Min	Max
Multirate (Asynchronous) – EN_SYN=FALSE							
	x4	5	8187	6	8188	4	8187
x4	x9	5	4091	6	4092	4	4091
x9	x18	5	2043	6	2044	4	2043
x18	x36	5	1019	6	1020	4	1019
x36	x72	5	507	6	508	4	507
Synchronous mode – EN_SYN=TRUE							
	x4	1	8190			1	8190
x4	x9	1	4094			1	4094
x9	x18	1	2046			1	2046
x18	x36	1	1022			1	1022
x36	x72	1	510			1	510

Notes:

1. For limitations under certain conditions, refer to Equation 4-1 on page 145.

The Almost Full and Almost Empty offsets are usually set to a small value of less than 10 to provide a warning that the FIFO is about to reach its limits. Since the full capacity of any FIFO is normally not critical, most applications use the ALMOST_FULL flag not only as a warning but also as a signal to stop writing.

Similarly, the ALMOST_EMPTY flag can be used to stop reading. However, this would make it impossible to read the very last entries remaining in the FIFO. The user can ignore the Almost Empty signal and continue to read until EMPTY is asserted.

The Almost Full and Almost Empty offsets can also be used in unstoppable block transfer applications to signal that a complete block of data can be written or read.

When setting the offset ranges in the design tools, use hexadecimal notation.

FIFO VHDL and Verilog Templates

VHDL and Verilog templates are available in the Libraries Guide.

FIFO Timing Models and Parameters

Table 4-20 shows the FIFO parameters.

Table 4-20: **FIFO Timing Parameters**

Parameter	Function	Control Signal	Description
Setup and Hold Relative to Clock (CLK)			
T_{RXCK} = Setup time (before clock edge) T_{RCKX} = Hold time (after clock edge)			
T_{RDCK_DI}/ T_{RCKD_DI}(4)	Data inputs	DI	Time before/after WRCLK that D1 must be stable.
T_{RCCK_RDEN}/ T_{RCKC_RDEN}(5)	Read enable	RDEN	Time before/after RDCLK that RDEN must be stable.
T_{RCCK_WREN}/ T_{RCKC_WREN}(5)	Write enable	WREN	Time before/after WRCLK that WREN must be stable.
Clock to Out Delays			
T_{RCKO_DO}(1)	Clock to data output	DO	Time after RDCLK that the output data is stable at the DO outputs of the FIFO. The synchronous FIFO with DO_REG = 0 is different than in multirate mode.
T_{RCKO_AEMPTY}(2)	Clock to almost empty output	AEMPTY	Time after RDCLK that the Almost Empty signal is stable at the ALMOSTEMPTY outputs of the FIFO.
T_{RCKO_AFULL}(2)	Clock to almost full output	AFULL	Time after WRCLK that the Almost Full signal is stable at the ALMOSTFULL outputs of the FIFO.
T_{RCKO_EMPTY}(2)	Clock to empty output	EMPTY	Time after RDCLK that the Empty signal is stable at the EMPTY outputs of the FIFO.
T_{RCKO_FULL}(2)	Clock to full output	FULL	Time after WRCLK that the Full signal is stable at the FULL outputs of the FIFO.
T_{RCKO_RDERR}(2)	Clock to read error output	RDERR	Time after RDCLK that the Read Error signal is stable at the RDERR outputs of the FIFO.
T_{RCKO_WRERR}(2)	Clock to write error output	WRERR	Time after WRCLK that the Write Error signal is stable at the WRERR outputs of the FIFO.
$T_{RCKO\_RDCOUNT}$(3)	Clock to read pointer output	RDCOUNT	Time after RDCLK that the Read pointer signal is stable at the RDCOUNT outputs of the FIFO.
$T_{RCKO\_WRCOUNT}$(3)	Clock to write pointer output	WRCOUNT	Time after WRCLK that the Write pointer signal is stable at the WRCOUNT outputs of the FIFO.

Table 4-20: **FIFO Timing Parameters *(Continued)***

Parameter	Function	Control Signal	Description
Reset to Out			
T_{RCO_AEMPTY}	Reset to almost empty output	AEMPTY	Time after reset that the Almost Empty signal is stable at the ALMOSTEMPTY outputs of the FIFO.
T_{RCO_AFULL}	Reset to almost full output	AFULL	Time after reset that the Almost Full signal is stable at the ALMOSTFULL outputs of the FIFO.
T_{RCO_EMPTY}	Reset to empty output	EMPTY	Time after reset that the Empty signal is stable at the EMPTY outputs of the FIFO.
T_{RCO_FULL}	Reset to full output	FULL	Time after reset that the Full signal is stable at the FULL outputs of the FIFO.
T_{RCO_RDERR}	Reset to read error output	RDERR	Time after reset that the Read error signal is stable at the RDERR outputs of the FIFO.
T_{RCO_WRERR}	Reset to write error output	WRERR	Time after reset that the Write error signal is stable at the WRERR outputs of the FIFO.
$T_{RCO\_RDCOUNT}$	Reset to read pointer output	RDCOUNT	Time after reset that the Read pointer signal is stable at the RDCOUNT outputs of the FIFO.
$T_{RCO\_WRCOUNT}$	Reset to write pointer output	WRCOUNT	Time after reset that the Write pointer signal is stable at the WRCOUNT outputs of the FIFO.

Notes:

1. T_{RCKO_DO} includes parity output (T_{RCKO_DOP}).
2. In the *Virtex-5 FPGA Data Sheet*, T_{RCKO_AEMPTY}, T_{RCKO_AFULL}, T_{RCKO_EMPTY}, T_{RCKO_FULL}, T_{RCKO_RDERR}, T_{RCKO_WRERR} are combined into T_{RCKO_FLAGS}.
3. In the *Virtex-5 FPGA Data Sheet*, $T_{RCKO\_RDCOUNT}$ and $T_{RCKO\_WRCOUNT}$ are combined into $T_{RCKO\_POINTERS}$.
4. T_{RCDCK_DI} includes parity inputs (T_{RCDCK_DIP}).
5. In the *Virtex-5 FPGA Data Sheet*, WRITE and READ enables are combined into T_{RCCK_EN}.

FIFO Timing Characteristics

The various timing parameters in the FIFO are described in this section. There is also additional data on FIFO functionality. The timing diagrams describe the behavior in these six cases.

- "Case 1: Writing to an Empty FIFO"
- "Case 2: Writing to a Full or Almost Full FIFO"
- "Case 3: Reading From a Full FIFO"
- "Case 4: Reading From An Empty or Almost Empty FIFO"
- "Case 5: Resetting All Flags"
- "Case 6: Simultaneous Read and Write for Multirate FIFO"

Case 1: Writing to an Empty FIFO

Prior to the operations performed in Figure 4-21, the FIFO is completely empty.

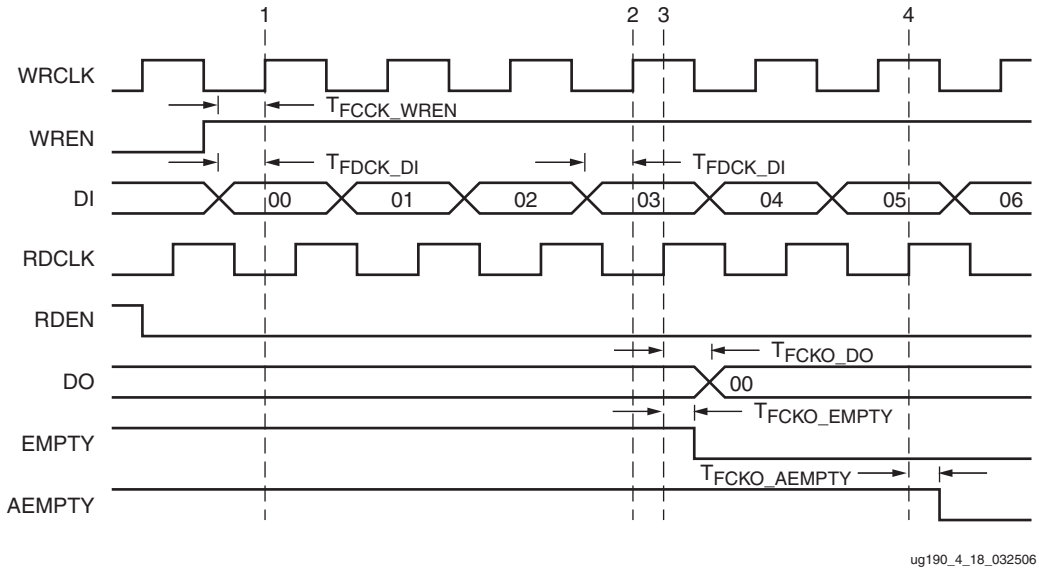


Figure 4-21: **Writing to an Empty FIFO in FWFT Mode**

Clock Event 1 and Clock Event 3: Write Operation and Deassertion of EMPTY Signal

During a write operation to an empty FIFO, the content of the FIFO at the first address is replaced by the data value on the DI pins. Three read-clock cycles later (four read-clock cycles for FWFT mode), the EMPTY pin is deasserted when the FIFO is no longer empty. The RDCOUNT also increments by one due to an internal read preloading the data to the output registers.

For the example in Figure 4-21, the timing diagram is drawn to reflect FWFT mode. Clock event 1 is with respect to the write-clock, while clock event 3 is with respect to the read-clock. Clock event 3 appears four read-clock cycles after clock event 1.

- At time T_{FDCK_DI}, before clock event 1 (WRCLK), data 00 becomes valid at the DI inputs of the FIFO.
- At time T_{FCCK_WREN}, before clock event 1 (WRCLK), write enable becomes valid at the WREN input of the FIFO.
- At time T_{FCKO_DO}, after clock event 3 (RDCLK), data 00 becomes valid at the DO output pins of the FIFO. In standard mode, data 00 does not appear at the DO output pins of the FIFO.
- At time T_{FCKO_EMPTY}, after clock event 3 (RDCLK), EMPTY is deasserted. In standard mode, EMPTY is deasserted one read-clock earlier than clock event 3.

If the rising WRCLK edge is close to the rising RDCLK edge, EMPTY could be deasserted one RDCLK period later.

Clock Event 2 and Clock Event 4: Write Operation and Deassertion of Almost EMPTY Signal

Three read-clock cycles after the fourth data is written into the FIFO, the Almost EMPTY pin is deasserted to signify that the FIFO is not in the almost EMPTY state.

For the example in Figure 4-21, the timing diagram is drawn to reflect FWFT mode. Clock event 2 is with respect to write-clock, while clock event 4 is with respect to read-clock. Clock event 4 appears three read-clock cycles after clock event 2.

- At time T_{FDCK_DI}, before clock event 2 (WRCLK), data 03 becomes valid at the DI inputs of the FIFO.
- Write enable remains asserted at the WREN input of the FIFO.
- At clock event 4, DO output pins of the FIFO remains at 00 since no read has been performed. In the case of standard mode, data 00 will never appear at the DO output pins of the FIFO.
- At time T_{FCKO_AEMPTY}, after clock event 4 (RDCLK), almost empty is deasserted at the AEMPTY pin. In the case of standard mode, AEMPTY deasserts in the same way as in FWFT mode.

If the rising WRCLK edge is close to the rising RDCLK edge, AEMPTY could be deasserted one RDCLK period later.

Case 2: Writing to a Full or Almost Full FIFO

Prior to the operations performed in Figure 4-22, the FIFO is almost completely full. In this example, the timing diagram reflects of both standard and FWFT modes.

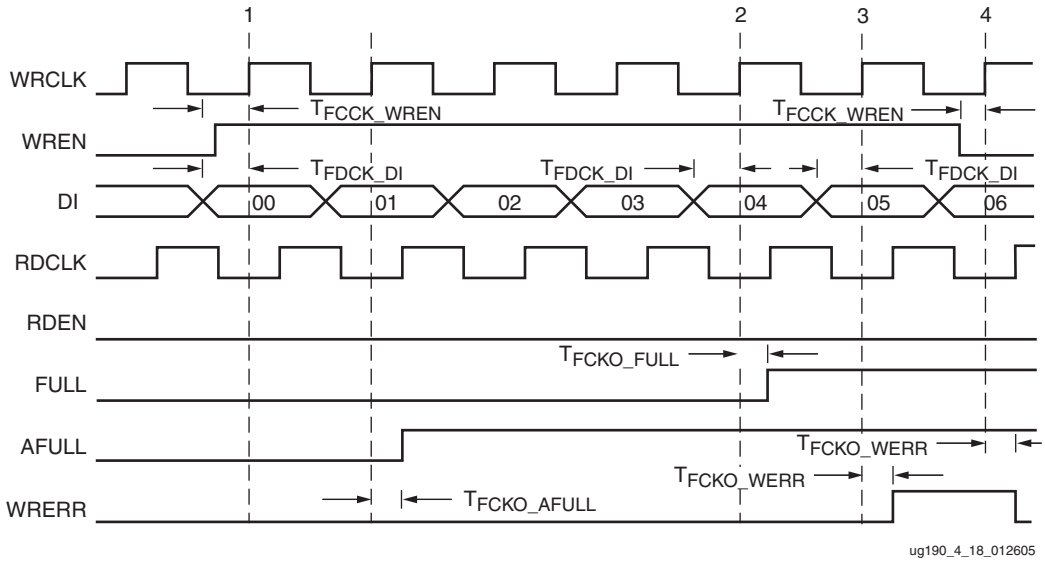


Figure 4-22: **Writing to a Full / Almost Full FIFO**

Clock Event 1: Write Operation and Assertion of Almost FULL Signal

During a write operation to an almost full FIFO, the Almost FULL signal is asserted.

- At time T_{FDCK_DI}, before clock event 1 (WRCLK), data 00 becomes valid at the DI inputs of the FIFO.
- At time T_{FCCK_WREN}, before clock event 1 (WRCLK), write enable becomes valid at the WREN input of the FIFO.
- At time T_{FCKO_AFULL}, one clock cycle after clock event 1 (WRCLK), Almost Full is asserted at the AFULL output pin of the FIFO.

Clock Event 2: Write Operation, and Assertion of FULL Signal

The FULL signal pin is asserted when the FIFO is full.

- At time T_{FDCK_DI}, before clock event 2 (WRCLK), data 04 becomes valid at the DI inputs of the FIFO.
- Write enable remains asserted at the WREN input of the FIFO.
- At time T_{FCKO_FULL}, after clock event 2 (WRCLK), Full is asserted at the FULL output pin of the FIFO.

If the FIFO is full, and a read followed by a write is performed, the FULL signal remains asserted.

Clock Event 3: Write Operation and Assertion of Write Error Signal

The write error signal pin is asserted when data going into the FIFO is not written because the FIFO is in a Full state.

- At time T_{FDCK_DI}, before clock event 3 (WRCLK), data 05 becomes valid at the DI inputs of the FIFO.
- Write enable remains asserted at the WREN input of the FIFO.
- At time T_{FCKO_WRERR}, after clock event 3 (WRCLK), a write error is asserted at the WRERR output pin of the FIFO. Data 05 is not written into the FIFO.

Clock Event 4: Write Operation and Deassertion of Write Error Signal

The write error signal pin is deasserted when a user stops trying to write into a full FIFO.

- At time T_{FCCK_WREN}, before clock event 4 (WRCLK), write enable is deasserted at the WREN input of the FIFO.
- At time T_{FCKO_WRERR}, after clock event 4 (WRCLK), write error is deasserted at the WRERR output pin of the FIFO.

The write error signal is asserted/deasserted at every write-clock positive edge. As long as both the write enable and Full signals are true, write error will remain asserted.

Case 3: Reading From a Full FIFO

Prior to the operations performed in Figure 4-23, the FIFO is completely full.

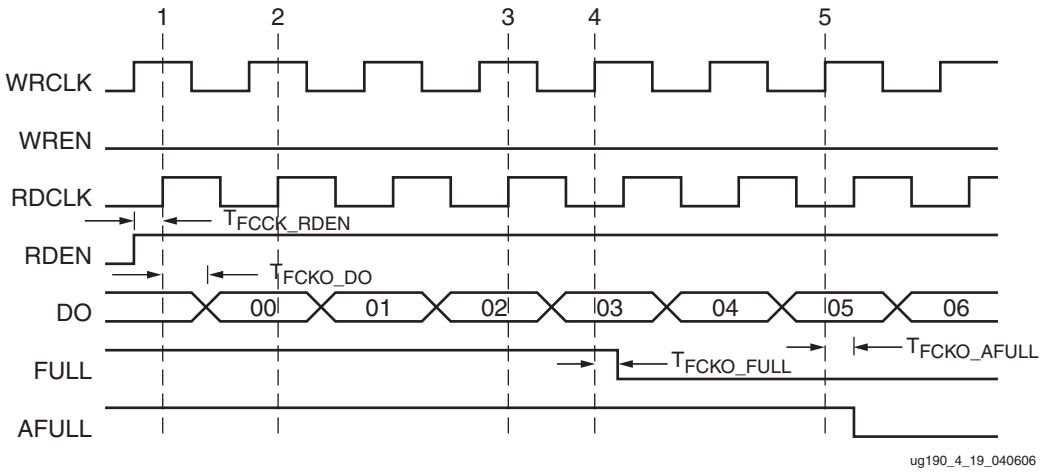


Figure 4-23: **Reading From a Full FIFO**

Clock Event 1 and Clock Event 2: Read Operation and Deassertion of Full Signal

During a read operation on a full FIFO, the content of the FIFO at the first address is asserted at the DO output pins of the FIFO. Two RDEN operations ensure that the FIFO is no longer full, and after three WRCLK cycles the FULL pin is deasserted.

The example in Figure 4-23 reflects both standard and FWFT modes. Clock event 1 and 2 are with respect to read-clock. Clock event 4 appears three write-clock cycles after clock event 2.

- At time T_{FCCK_RDEN}, before clock event 1 (RDCLK), read enable becomes valid at the RDEN input of the FIFO.
- At time T_{FCKO_DO}, after clock event 1 (RDCLK), data 00 becomes valid at the DO outputs of the FIFO.
- At time T_{FCKO_FULL}, after clock event 4 (WRCLK), FULL is deasserted.

If the rising RDCLK edge is close to the rising WRCLK edge, FULL could be deasserted one WRCLK period later.

Clock Event 3 and Clock Event 5: Read Operation and Deassertion of Almost FULL Signal

Three write-clock cycles after the fourth data is read from the FIFO, the Almost FULL pin is deasserted to signify that the FIFO is not in the almost FULL state.

The example in Figure 4-23 reflects both standard and FWFT modes. Clock event 3 is with respect to read-clock, while clock event 5 is with respect to write-clock. Clock event 5 appears three write-clock cycles after clock event 3.

- Read enable remains asserted at the RDEN input of the FIFO.
- At time T_{FCKO_AFULL}, after clock event 5 (RDCLK), Almost FULL is deasserted at the AFULL pin.

There is minimum time between a rising read-clock and write-clock edge to guarantee that AFULL will be deasserted. If this minimum is not met, the deassertion of AFULL can take an additional write clock cycle.

Case 4: Reading From An Empty or Almost Empty FIFO

Prior to the operations performed in Figure 4-24, the FIFO is almost completely empty. In this example, the timing diagram reflects standard mode. For FWFT mode, data at DO appears one read-clock cycle earlier.

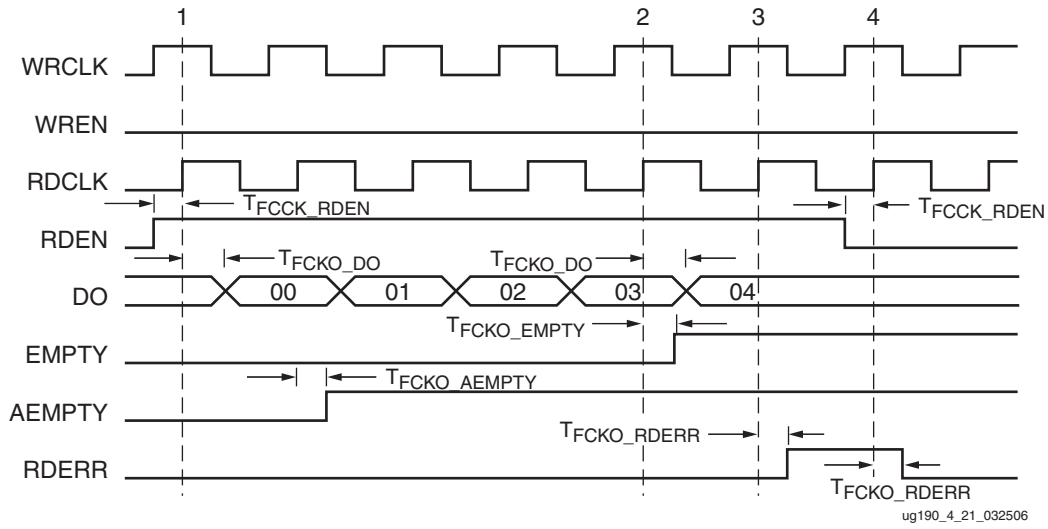


Figure 4-24: **Reading From an Empty / Almost Empty FIFO (Standard Mode)**

Clock Event 1: Read Operation and Assertion of Almost EMPTY Signal

During a read operation to an almost empty FIFO, the Almost EMPTY signal is asserted.

- At time T_{FCCK_RDEN}, before clock event 1 (RDCLK), read enable becomes valid at the RDEN input of the FIFO.
- At time T_{FCKO_DO}, after clock event 1 (RDCLK), data `00` becomes valid at the DO outputs of the FIFO.
- At time T_{FCKO_AEMPTY}, one clock cycle after clock event 1 (RDCLK), Almost Empty is asserted at the AEMPTY output pin of the FIFO.

Clock Event 2: Read Operation and Assertion of EMPTY Signal

The EMPTY signal pin is asserted when the FIFO is empty.

- Read enable remains asserted at the RDEN input of the FIFO.
- At time T_{FCKO_DO}, after clock event 2 (RDCLK), data `04` (last data) becomes valid at the DO outputs of the FIFO.
- At time T_{FCKO_EMPTY}, after clock event 2 (RDCLK), Empty is asserted at the EMPTY output pin of the FIFO.

In the event that the FIFO is empty and a write followed by a read is performed, the EMPTY signal remains asserted.

Clock Event 3: Read Operation and Assertion of Read Error Signal

The read error signal pin is asserted when there is no data to be read because the FIFO is in an empty state.

- Read enable remains asserted at the RDEN input of the FIFO.
- At time T_{FCKO_RDERR}, after clock event 3 (RDCLK), read error is asserted at the RDERR output pin of the FIFO.
- Data 04 remains unchanged at the DO outputs of the FIFO.

Clock Event 4: Read Operation and Deassertion of Read Error Signal

The read error signal pin is deasserted when a user stops trying to read from an empty FIFO.

- At time T_{FCCK_RDEN}, before clock event 4 (RDCLK), read enable is deasserted at the RDEN input of the FIFO.
- At time T_{FCKO_RDERR}, after clock event 4 (RDCLK), read error is deasserted at the RDERR output pin of the FIFO.

The read error signal is asserted/deasserted at every read-clock positive edge. As long as both the read enable and empty signals are true, read error will remain asserted.

Case 5: Resetting All Flags

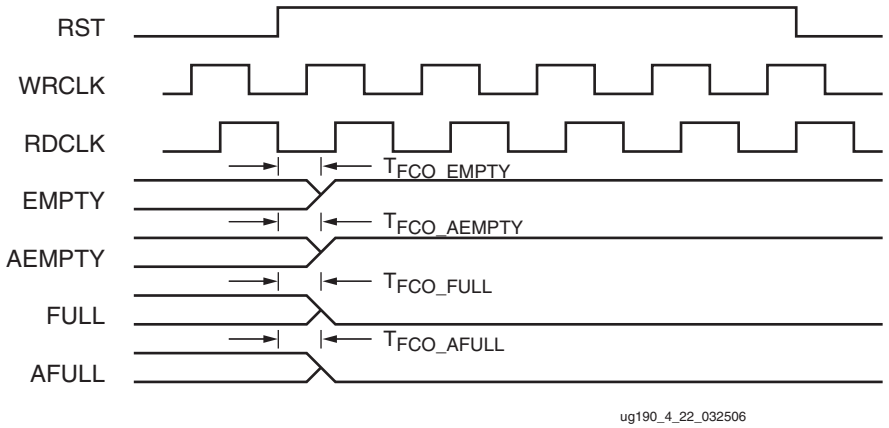


Figure 4-25: **Resetting All Flags**

When the reset signal is asserted, all flags are reset.

- At time T_{FCO_EMPTY}, after reset (RST), empty is asserted at the EMPTY output pin of the FIFO.
- At time T_{FCO_AEMPTY}, after reset (RST), almost empty is asserted at the AEMPTY output pin of the FIFO.
- At time T_{FCO_FULL}, after reset (RST), full is deasserted at the FULL output pin of the FIFO.
- At time T_{FCO_AFULL}, after reset (RST), almost full is deasserted at the AFULL output pin of the FIFO.

Reset is an asynchronous signal used to reset all flags. Hold the reset signal High for three read and write clock cycles to ensure that all internal states and flags are reset to the correct value.

Case 6: Simultaneous Read and Write for Multirate FIFO

Simultaneous read and write operations for an asynchronous FIFO is not deterministic when the FIFO is at the condition to assert a status flag. The FIFO logic resolves the situation (either assert or not assert the flag), the software simulation model can not reflect this behavior and mismatch can occur. When using a single clock for RDCLK and WRCLK, use the FIFO in synchronous mode (EN_SYN=TRUE).

FIFO Applications

A FIFO larger than a single Virtex-5 FPGA FIFO block can be created by:

- Cascading two or more FIFOs to form a deeper FIFO.
- Building a wider FIFO by connecting two or more FIFOs in parallel.

Cascading FIFOs to Increase Depth

Figure 4-26 shows a way of cascading N FIFO36s to increase depth. The application sets the first N–1 FIFOs in FWFT mode and uses external resources to connect them together. The data latency of this application is the sum of the individual FIFO latencies. The maximum frequency is limited by the feedback path. The NOR gate is implemented using CLB logic.

- N can be 2 or more; if N is 2, the middle FIFOs are not needed.
- If WRCLK is faster than RDCLK, then INTCLK = WRCLK
- If WRCLK is equal to or slower than RDCLK, then INTCLK = RDCLK
- ALMOST_EMPTY threshold is set in the Nth FIFO; ALMOST_FULL threshold is set in 1st FIFO.

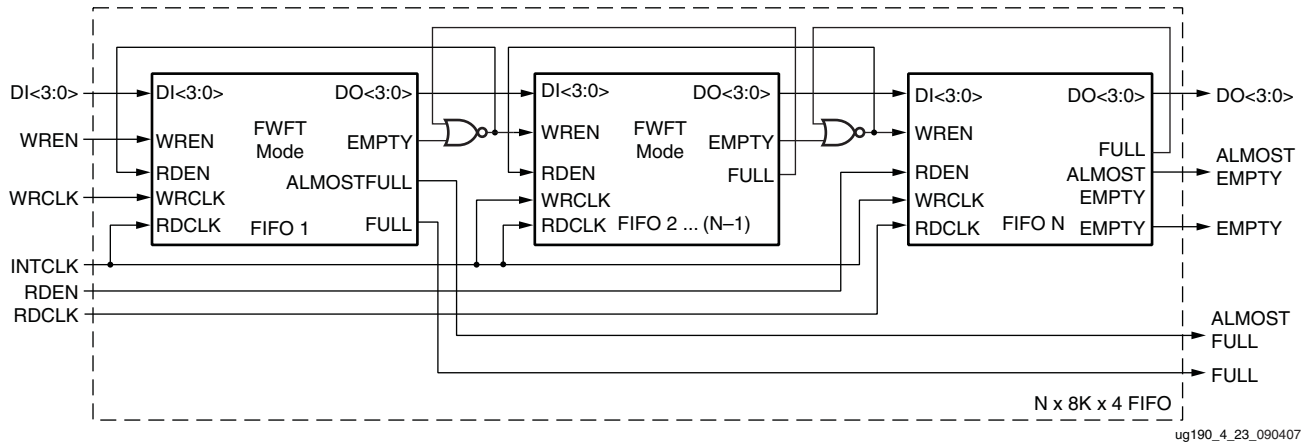


Figure 4-26: **Example: Cascading Multiple FIFOs by Depth**

Connecting FIFOs in Parallel to Increase Width

As shown in Figure 4-27, the Virtex-5 FPGA FIFO36 can be connected to add width to the design. CLB logic is used to implement the AND/OR gates. All the FIFO AFULL signals must be ORed together to created the output AFULL signal and all the FIFO EMPTY signals must be ORed together to created the output EMPTY signal. The maximum frequency is limited by the logic gate feedback path.

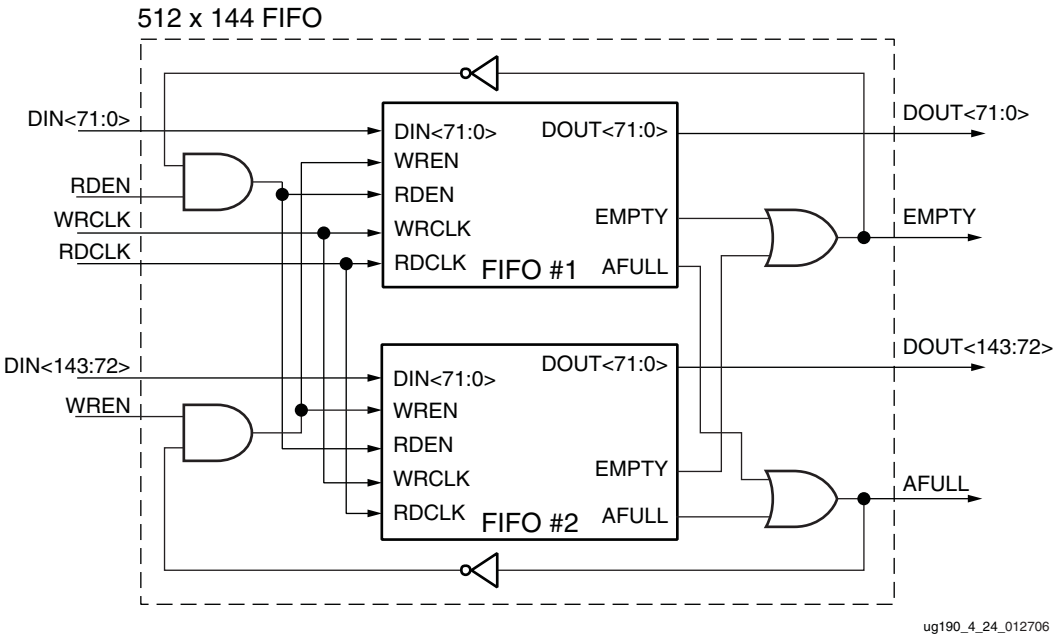


Figure 4-27: **Example: Connecting FIFOs in Parallel to Increase Width**

Built-in Error Correction

Each simple dual-port block RAM can be configured as a single 512 x 64 RAM with built in Hamming code error correction, using the extra eight bits in the 72-bit wide RAM. The operation is transparent to the user.

Eight protection bits (ECCPARITY) are generated during each write operation and stored with the 64-bit data into the memory. These ECCPARITY bits are used during each read operation to correct any single-bit error, or to detect (but not correct) any double-bit error. The ECCPARITY bits are written into the memory and output to the FPGA fabric at each rising edge of the WRCLK. There are no optional output registers available on the ECCPARITY output bits.

During each read operation, 72 bits of data (64 bits of data and an 8-bit parity) are read from the memory and fed into the ECC decoder. The ECC decoder generates two status outputs (SBITERR and DBITERR) that are used to indicate the three possible read results: No error, single-bit error corrected, double-bit error detected. In the standard ECC mode, the read operation does not correct the error in the memory array, it only presents corrected data on DO. To improve F_{MAX}, optional registers controlled by the DO_REG attribute are available for data output (DO), SBITERR, and DBITERR.

This ECC configuration option is available with a 36K block RAM simple dual-port primitive (RAMB36SDP) or a 36K FIFO primitive (FIFO36_72). A Virtex-4 FPGA ECC 18K

block RAM mapped for a Virtex-5 FPGA design will occupy the entire RAMB36 site. FIFO36_72 supports standard ECC mode only.

ECC Modes Overview

In the standard ECC mode (EN_ECC_READ = TRUE and EN_ECC_WRITE = TRUE), both encoder and decoder are enabled. During write, 64-bit data and 8-bit ECC generated parity are stored in the array. The external parity bits are ignored. During read, the 72-bit decoded data and parity are read out.

The encoder and decoder can be accessed separately for external use in RAMB36SDP. To use the encoder by itself, send the data in through the DI port and sample the ECCPARITY output port. To use the decoder by itself, disable the encoder, write the data into the block RAM and read the corrected data and status bits out of the block RAM. See "Block RAM (RAMB36SDP) Attributes."

To use the decoder in ECC decode-only mode, set EN_ECC_WRITE = FALSE and EN_ECC_READ = TRUE.

The encoder can be used in two ways:

- To use the encoder in standard ECC mode, set (EN_ECC_WRITE = TRUE and EN_ECC_READ = TRUE). In this mode, the DI setup time is smaller but the clock-to-out for ECCPARITY is larger.
- To use the encoder-only mode, set (EN_ECC_WRITE = TRUE and EN_ECC_READ = FALSE). In this mode, the DI setup time is larger but the clock-to-out for ECCPARITY is smaller.

The functionality of the block RAM when using the ECC mode is described as follows:

- The block RAM ports still have independent address, clocks, and enable inputs, but one port is a dedicated write port, and the other is a dedicated read port (simple dual-port).
- DO represents the read data after correction.
- DO stays valid until the next active read operation.
- Simultaneous decoding and encoding, even with asynchronous clocks, is allowed, but requires careful clock timing if read and write addresses are identical.
- The READ_FIRST or WRITE_FIRST modes of the normal block RAM operation are not applicable to the ECC configuration.

Top-Level View of the Block RAM ECC Architecture

Figure 4-28 shows the top-level view of a Virtex-5 FPGA block RAM in ECC mode.

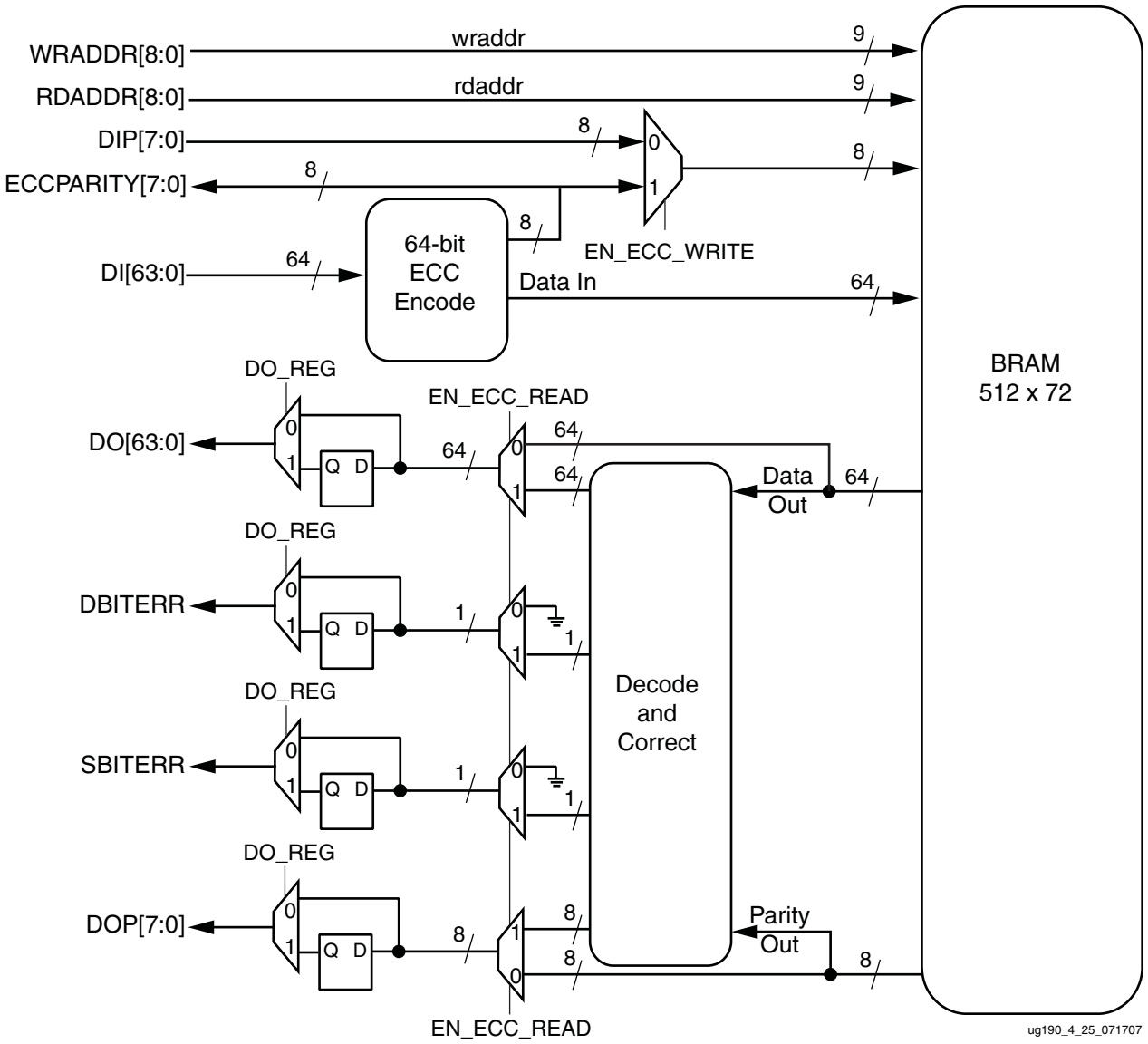


Figure 4-28: **Top-Level View of Block RAM ECC**

Block RAM and FIFO ECC Primitive

Figure 4-29 shows the block RAM (RAMB36SDP) ECC primitive. Figure 4-30 shows the FIFO36_72 ECC primitive. The FIFO36_72 only supports standard mode.

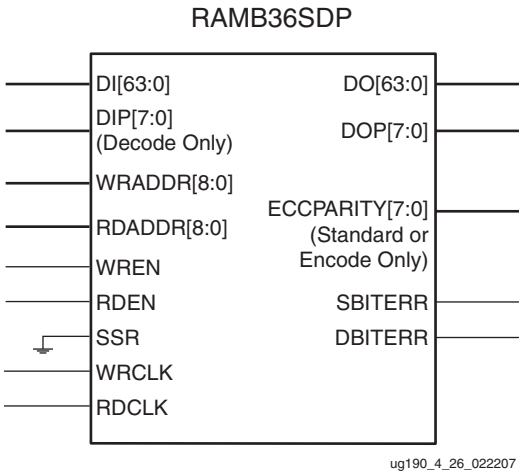


Figure 4-29: **RAMB36SDP: Block RAM ECC Primitive**

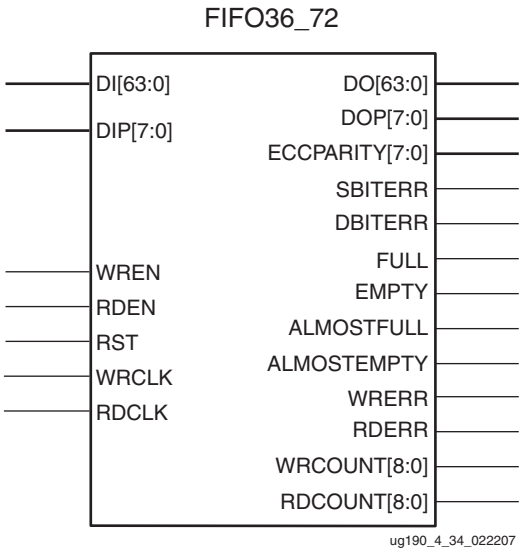


Figure 4-30: **FIFO36_72: FIFO ECC Primitive**

Block RAM and FIFO ECC Port Descriptions

Table 4-21 lists and describes the block RAM ECC I/O port names.

Table 4-21: **Block RAM ECC Port Names and Descriptions**

Port Name	Direction	Signal Description
DI[63:0]	Input	Data input bus.
DIP[7:0]	Input	Data input parity bus. Used in decode-only mode to input the precalculated ECC parity bits.
WRADDR[8:0]	Input	Write address bus.
RDADDR[8:0]	Input	Read address bus.
WREN	Input	Write enable. When WREN = 1, data will be written into memory. When WREN = 0, write is disabled
RDEN	Input	Read enable. When RDEN = 1, data will be read from memory. When RDEN = 0, read is disabled.
SSR	Input	Not supported when using the block RAM ECC primitive. Always connect to GND.
WRCLK	Input	Clock for write operations.
RDCLK	Input	Clock for read operations.
DO[63:0]	Output	Data output bus.
DOP[7:0]	Output	Data output parity bus. Used in encode-only mode to output the stored ECC parity bits.
SBITERR(1)	Output	Single-bit error status.
DBITERR(1)	Output	Double-bit error status.
ECCPARITY[7:0]	Output	ECC encoder output bus.

Notes:

1. Hamming code implemented in the block RAM ECC logic detects one of three conditions: no detectable error, single-bit error detected and corrected on DO (but not corrected in the memory), and double-bit error detected without correction. SBITERR and DBITERR indicate these three conditions.

Table 4-22 lists and describes the FIFO ECC I/O port names.

Table 4-22: **FIFO ECC Port Names and Descriptions**

Port Name	Direction	Signal Description
DI[63:0]	Input	Data input bus.
DIP[7:0]	Input	Data input parity bus. Not used when standard mode is used.
WREN	Input	Write enable. When WREN = 1, data will be written into memory. When WREN = 0, write is disabled
RDEN	Input	Read enable. When RDEN = 1, data will be read from memory. When RDEN = 0, read is disabled.
RST	Input	Asynchronous reset of FIFO counter and flags. Reset must be asserted for three clock cycles. Reset does not affect DO or ECC signals.
WRCLK	Input	Clock for write operations.
RDCLK	Input	Clock for read operations.
DO[63:0]	Output	Data output bus.
DOP[7:0]	Output	Data output parity bus.
SBITERR(1)	Output	Single-bit error status.
DBITERR(1)	Output	Double-bit error status.
ECCPARITY[7:0]	Output	ECC encoder output bus.
FULL	Output	FIFO FULL flag.
ALMOSTFULL	Output	FIFO ALMOSTFULL flag.
EMPTY	Output	FIFO EMPTY flag.
ALMOSTEMPTY	Output	FIFO ALMOSTEMPTY flag.
RDCOUNT	Output	The FIFO data read pointer.
WRCOUNT	Output	The FIFO data write pointer.
WRERR	Output	When the FIFO is full, any additional write operation generates an error flag.
RDERR	Output	When the FIFO is empty, any additional read operation generates an error flag.

Notes:

1. Hamming code implemented in the FIFO ECC logic detects one of three conditions: no detectable error, single-bit error detected and corrected on DO (but not corrected in the memory), and double-bit error detected without correction. SBITERR and DBITERR indicate these three conditions.

Block RAM and FIFO ECC Attributes

In addition to the built-in registers in the decode and correct logic, the RAMB36SDP primitive allows the use of optional pipeline registers controlled by the DO_REG attribute to produce higher performance with one additional latency. Table 4-23 and Table 4-24 list the block RAM and FIFO ECC attributes.

Table 4-23: **Block RAM (RAMB36SDP) Attributes**

Attribute Name	Type	Values	Default	Notes
EN_ECC_WRITE	Boolean	TRUE, FALSE	FALSE	Set to TRUE to enable ECC encoder.
EN_ECC_READ	Boolean	TRUE, FALSE	FALSE	Set to TRUE to enable ECC decoder.
DO_REG	1-bit Binary	0, 1	0	Enables register mode or latch mode.

Table 4-24: **FIFO (FIFO36_72) Attributes**

Attribute Name	Type	Values	Default	Notes
EN_ECC_WRITE	Boolean	TRUE, FALSE	FALSE	Both attributes must be set to TRUE to enable ECC functionality in a FIFO36_72.
EN_ECC_READ	Boolean	TRUE, FALSE	FALSE	
DO_REG	1-bit Binary	0, 1	1	Enables register mode or latch mode. See Table 4-17 for details on multirate and synchronous FIFOs.
EN_SYN	Boolean	TRUE, FALSE	FALSE	When set to TRUE, ties WRCLK and RDCLK together. When set to TRUE, FWFT must be FALSE. When set to FALSE, DO_REG must be 1.
ALMOST_EMPTY_OFFSET	9-bit Hex	See Table 4-19	See Table 4-19	Setting determines the difference between EMPTY and ALMOST_EMPTY conditions. Must be set using hexadecimal notation.
ALMOST_FULL_OFFSET	9-bit Hex	See Table 4-19	See Table 4-19	Setting determines the difference between FULL and ALMOST_FULL conditions. Must be set using hexadecimal notation.
FIRST_WORD_FALL_THROUGH	Boolean	TRUE, FALSE	FALSE	When set to TRUE, the first word written into the empty FIFO36_72 appears at the FIFO36_72 output without RDEN asserted.

ECC Modes of Operation

There are three types of ECC operation: standard, encode only, and decode only. The standard ECC mode uses both the encoder and decoder.

The various modes of ECC operation in both block RAM and FIFO are shown in Figure 4-31 and Figure 4-32. The block RAM WRADDR and RDADDR address inputs are supplied by the user. The FIFO WRADDR and RDADDR addresses are generated internally from the write counter and read counter.

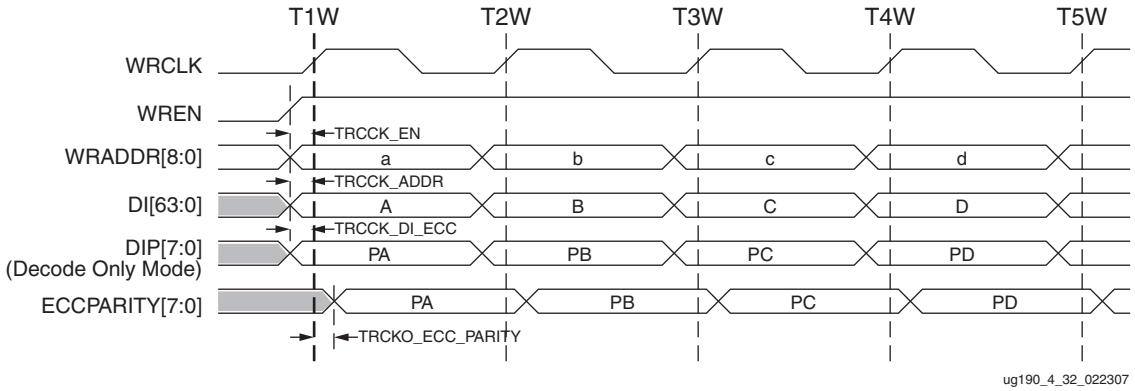


Figure 4-31: **ECC Write Operation**

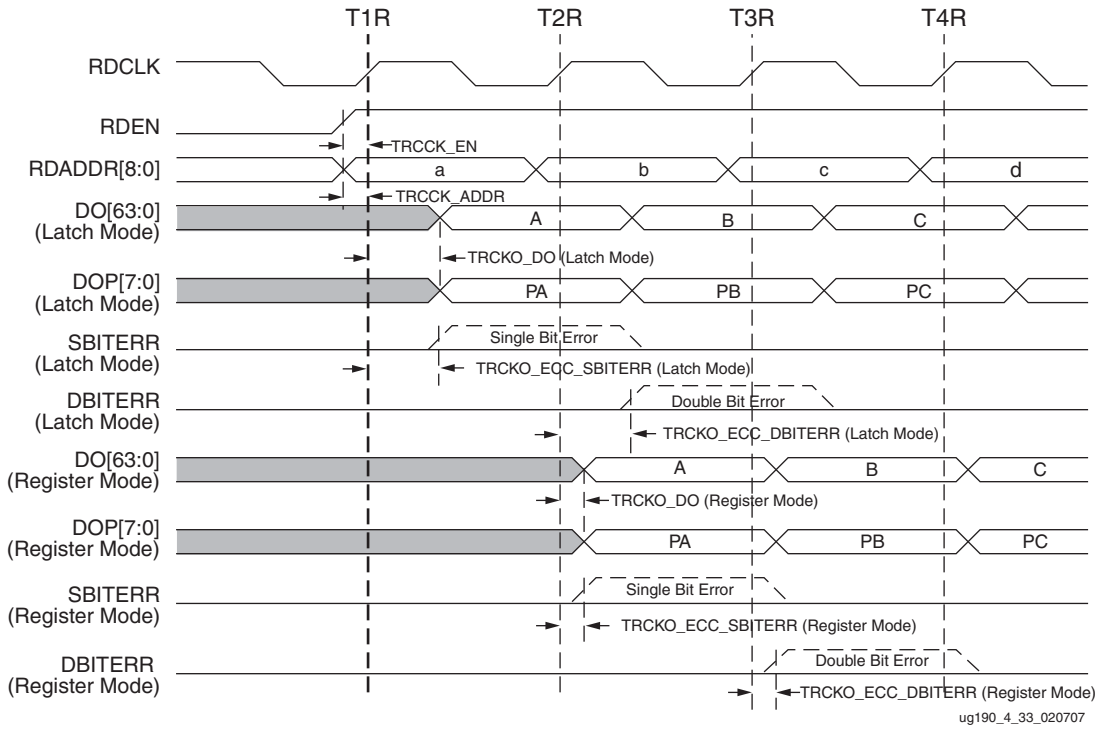


Figure 4-32: **ECC Read Operation**

Standard ECC

Set by Attributes

EN_ECC_READ = TRUE
EN_ECC_WRITE = TRUE

Standard ECC Write

At time T1W, DI[63:0] = A is written into memory location *a*. The corresponding 8 bits of ECC parity PA (hex) are generated internally, appended to the 64 data bits, and written into the memory. Immediately after the write, the parity value PA appears at output ECCPARITY[7:0]. Since ECC parity is generated internally, DIP[7:0] pins are not used.

Similarly, at time T2W and T3W, DI[63:0] = B and C, together with their corresponding parity bits PB (hex) and PC (hex) are written into memory locations *b* and *c*. PB and PC appear at output ECCPARITY[7:0] shortly after T2W and T3W.

Standard ECC Read

At time T1R, the 72-bit memory content, consisting 64 bits of data A and 8 bits of parity PA (hex), of address location *a* is read and decoded internally. If there is no error, the original data and parity are output at DO[63:0] and DOP[7:0]. If there is a single-bit error in either the data or the parity, the error is corrected, and SBITERR is High. If there is a double-bit error in the data and parity, the error is not corrected. The original data and parity is output and DBITERR is High.

If attribute DO_REG is set to 0, DO[63:0] = A and DOP[7:0] = PA shortly after T1R. Similarly, at time T2R and T3R, the memory content at address locations *b* and *c* are read and decoded at DO[63:0] and DOP[7:0]. SBITERR/DBITERR outputs can also switch after T1R if a single or double-bit error is detected on dataset A. Figure 4-32 shows a single-bit error (SBITERR) being detected on data A in latch mode after clock edge T1R and a double-bit error (DBITERR) being detected on data B in latch mode after clock edge T2R.

If attribute DO_REG is set to 1, DO[63:0] = A and DOP[7:0] = PA shortly after T2R. Similarly, at time T3R and T4R, the memory content at address locations *b* and *c* are read and decoded at DO[63:0] and DOP[7:0]. SBITERR/DBITERR outputs may also switch after T2R if a single or double-bit error is detected on dataset A. Figure 4-32 shows a single-bit error (SBITERR) being detected on data A in register mode after clock edge T2R and a double-bit error (DBITERR) being detected on data B in register mode after clock edge T3R.

In ECC mode, the encode-only port and the decode-only port operate independently of each other.

ECC Encode-Only

Set by Attributes

EN_ECC_READ = FALSE
EN_ECC_WRITE = TRUE

ECC Encode-Only Write

At time T1W, DI[63:0] = A is written into memory location a. The corresponding 8 bits of ECC parity PA (hex) are generated internally, appended to the 64 data bits, and written into the memory. Immediately after the write, the parity value PA appears at output ECCPARITY[7:0]. Since ECC parity is generated internally, DIP[7:0] pins are not used.

Similarly, at time T2W and T3W, DI[63:0] = B and C, together with their corresponding parity bits PB (hex) and PC (hex) are written into memory locations *b* and *c*. PB and PC appear at output ECCPARITY[7:0] shortly after T2W and T3W.

ECC Encode-Only Read

ECC encode-only read is identical to normal block RAM read. 64-bit data appears at DO[63:0] and 8-bit parity appears at DOP[7:0]. Single-bit error correction does not happen, and the error flags SBITERR and DBITERR is never asserted.

ECC Decode-Only

Set by Attributes

EN_ECC_READ = TRUE
EN_ECC_WRITE = FALSE

In ECC decode-only, only the ECC decoder is enabled. The ECC encoder is disabled. Decode-only mode is used to inject single-bit or double-bit errors to test the functionality of the ECC decoder. The ECC parity bits must be externally supplied using the DIP[7:0] pins.

Using ECC Decode-Only to Inject Single-Bit Error

- At time T1W, T2W, T3W, DI[63:0] = A, B, C with single-bit error and DIP[7:0] = PA (hex), PB (hex), PC (hex), the corresponding ECC parity bits for A, B, and C are written into memory locations *a*, *b*, and *c*.
- At time T1R, T2R, T3R, the contents of address *a*, *b*, and *c* are read out and corrected as needed.
- Latch mode: DO[63:0] = A, B, C, DOP[7:0] = PA, PB, PC shortly after T1R, T2R, T3R.
- Register mode: DO[63:0] = A, B, C, DOP[7:0] = PA, PB, PC shortly after T2R, T3R, T4R.
- SBITERR lines up with the corresponding DO/DOP data.

The ECC decoder also corrects single-bit error in parity bits.

Using the ECC Decode-Only to Inject Double-Bit Error

- At time T1W, T2W, T3W, DI[63:0] = A, B, C with double-bit error and DIP[7:0] = PA (hex), PB (hex), PB (hex), the corresponding ECC parity bits for A, B, and C are written into memory location *a*, *b*, and *c*.
- At time T1R, T2R, T3R, the original contents of address *a*, *b*, and *c* are read out and a double-bit error is detected.
- Latch mode: DO[63:0] = A, B, C with double-bit error, DOP[7:0] = PA, PB, PC shortly after T1R, T2R, T3R.
- Register mode: DO[63:0] = A, B, C with double-bit error, DOP[7:0] = PA, PB, PC shortly after T2R, T3R, T4R.
- DBITERR lines up with the corresponding DO/DOP data.

The ECC decoder also detects when double-bit error in parity bits occurs, and when a single-bit error in the data bits and a single-bit error in the corresponding parity bits occurs.

ECC Timing Characteristics

The various ECC timing parameters are also shown in Figure 4-31 and Figure 4-32.

Since write clock and read clock are independent of each other, all write timing in Figure 4-31 is referenced to WRCLK. All read timing in Figure 4-32 is referenced to RDCLK.

Standard ECC Write Timing (Figure 4-31)

- At time TRCCK_EN, before time T1W, write enable becomes valid at the WREN input of the block RAM.
- At time TRCCK_ADDR, before time T1W, write address *a* becomes valid at the WRADDR[8:0] inputs of the block RAM. WRADDR input is not needed for FIFO.
- At time TRDCK_DI_ECC (standard ECC), before time T1W, write data A (hex) becomes valid at the DI[63:0] inputs of the block RAM.
- At time TRCKO_ECC_PARITY (standard ECC), after time T1W, ECC parity data PA (hex) becomes valid at the ECCPARITY[7:0] output pins of the block RAM.

Standard ECC Read Timing (Figure 4-32)

- At time TRCCK_EN, before time T1R, read enable becomes valid at the RDEN input of the block RAM.
- At time TRCCK_ADDR, before time T1R, write address *a* becomes valid at the RDADDR[8:0] inputs of the block RAM. RDADDR input is not needed for FIFO.

DO_REG = 0

- At time TRCKO_DO (latch mode), after time T1R, data A (hex) becomes valid at the DO[63:0] output pins of the block RAM.
- At time TRCKO_DOP (latch mode), after time T1R, data PA (hex) becomes valid at the DOP[7:0] output pins of the block RAM.
- At time TRCKO_ECC_SBITERR (latch mode), after time T1R, SBITERR is asserted if single-bit error is detected and corrected on data set A.
- At time TRCKO_ECC_DBITERR (latch mode), after time T2R, DBITERR is asserted if double-bit error is detected on data set B.

DO_REG = 1

- At time TRCKO_DO (register mode), after time T2R, data A (hex) becomes valid at the DO[63:0] output pins of the block RAM.
- At time TRCKO_DOP (register mode), after time T2R, data PA (hex) becomes valid at the DOP[7:0] output pins of the block RAM.
- At time TRCKO_ECCR_SBITERR (register mode), after time T2R, SBITERR is asserted if single-bit error is detected and corrected on data set A.
- At time TRCKO_ECCR_DBITERR (register mode), after time T3R, DBITERR is asserted if double-bit error is detected on data set B.

Encode-Only ECC Write Timing (Figure 4-31)

- Setup/hold time for WREN and WRADDR are the same as standard ECC.
- At time TRDCK_DI_ECC (encode-only ECC), before time T1W, write data A (hex) becomes valid at the DI[63:0] inputs of the block RAM.
- At time TRCKO_ECC_PARITY (encode-only ECC), after time T1W, ECC parity data PA (hex) becomes valid at the ECCPARITY[7:0] output pins of the block RAM.

Encode-Only ECC Read Timing

- Encode-only ECC read timing are the same as normal block RAM read timing.

Decode-Only ECC Write Timing

- Decode-only ECC write timing is the same as normal block RAM write timing.

Decode-Only ECC Read Timing

- Decode-only ECC read timing is the same as standard ECC read timing.

Block RAM ECC Mode Timing Parameters

Table 4-25 shows the Virtex-5 FPGA block RAM ECC mode timing parameters.

Table 4-25: **Block RAM ECC Mode Timing Parameters**

Parameter	Function	Control Signal	Description
Setup and Hold Relative to Clock (CLK)			
T_{RxCK_x} = Setup time (before clock edge) and T_{RCKx_x} = Hold time (after clock edge)			
$T_{RDCK\_DI\_ECC}$ (Standard ECC Mode)	Data inputs(1)	DI	Time before the clock that data must be stable at the DI inputs of the block RAM. Standard ECC mode.
$T_{RCKD\_DI\_ECC}$ (Standard ECC Mode)			Time after the clock that data must be stable at the DI inputs of the block RAM. Standard ECC mode.
$T_{RDCK\_DI\_ECC}$ (Encode-only Mode)	Data inputs(1)	DI	Time before the clock that data must be stable at the DI inputs of the block RAM. Encode-only mode.
$T_{RCKD\_DI\_ECC}$ (Encode-only Mode)			Time after the clock that data must be stable at the DI inputs of the block RAM. Encode-only mode.
Clock to Out Delays			
T_{RCKO_DO} (latch mode)	Clock to Output(2)	CLK to DO	Time after the clock that the output data is stable at the DO outputs of the block RAM (without output register).
T_{RCKO_DO} (register mode)	Clock to Output(2)	CLK to DO	Time after the clock that the output data is stable at the DO outputs of the block RAM (with output register).

Table 4-25: **Block RAM ECC Mode Timing Parameters *(Continued)***

Parameter	Function	Control Signal	Description
Clock to ECC Delays			
$T_{RCKO\_ECC\_PARITY}$ (encode-only mode)(3)	Clock to ECC Parity Output	ECCPARITY	Time after WRCLK that the ECC parity signals are stable at the ECCPARITY outputs of the block RAM (in encode-only mode).
$T_{RCKO\_ECC\_SBITERR}$(3)	Clock to ECC Single-Bit-Error Output	SBITERR	Time after RDCLK that the single-bit-error signal is stable at the SBITERR output of the block RAM (without output register).
$T_{RCKO\_ECCR\_SBITERR}$(4)	Clock to ECC Single-Bit-Error Output	SBITERR	Time after RDCLK that the single-bit-error signal is stable at the SBITERR output of the block RAM (with output register).
$T_{RCKO\_ECC\_DBITERR}$(3)	Clock to ECC Double-Bit-Error Output	DBITERR	Time after RDCLK that the double-bit-error signal is stable at the DBITERR output of the block RAM (without output register).
$T_{RCKO\_ECCR\_DBITERR}$(4)	Clock to ECC Double-Bit-Error Output	DBITERR	Time after RDCLK that the double-bit-error signal is stable at the DBITERR output of the block RAM (with output register).

Notes:

1. $T_{RDCK\_DI\_ECC}$/$T_{RCKD\_DI\_ECC}$ include the parity input $T_{RDCK\_DIP\_ECC}$/$T_{RCKD\_DIP\_ECC}$.
2. T_{RCKO_DO} includes parity output (T_{RCKO_DOP}).
3. $T_{RCKO\_ECC\_PARITY}$, $T_{RCKO\_ECC\_SBITERR}$, and $T_{RCKO\_ECC\_DBITERR}$ are combined into the T_{RCKO_ECC} parameter in the *Virtex-5 FPGA Data Sheet*.
4. $T_{RCKO\_ECC\_SBITERR}$ and $T_{RCKO\_ECC\_DBITERR}$ are combined into the T_{RCKO_ECCR} parameter in the *Virtex-5 FPGA Data Sheet*.

Creating a Deliberate Error in a 72-bit Word

To deliberately create an error in a 72-bit word, configure the ECC decode-only mode and create a 72-bit word with one or two bit errors. Write the word into the block RAM. Reading the 72-bit word automatically corrects the single-bit error and asserts the SBITERR error flag or it detects the double-bit error and asserts the DBITERR error flag.

Creating Eight Parity Bits for a 64-bit Word

Using logic external to the block RAM (a large number of XOR circuits), eight parity bits can be created for a 64-bit word. However, using ECC encoder-only mode, the eight parity bits can be automatically created without additional logic by writing any 64-bit word into a separate block RAM. The encoded 8-bit ECC parity data is immediately available, or the complete 72-bit word can be read out.

Inserting a Single or Double Bit Error into a 72-bit Word

By reading a 72-bit word and selectively modifying one or two bits, then writing all 72-bits into the block RAM under test in ECC decode-only mode, a single or double bit error can be inserted.

Block RAM ECC VHDL and Verilog Templates

VHDL and Verilog templates are available in the Libraries Guide.

Legal Block RAM and FIFO Combinations

The block RAM–FIFO combinations shown in Figure 4-33 are supported in a single RAMB36 primitive. When placing block RAM and FIFO primitives in the same location, the FIFO must occupy the lower port.

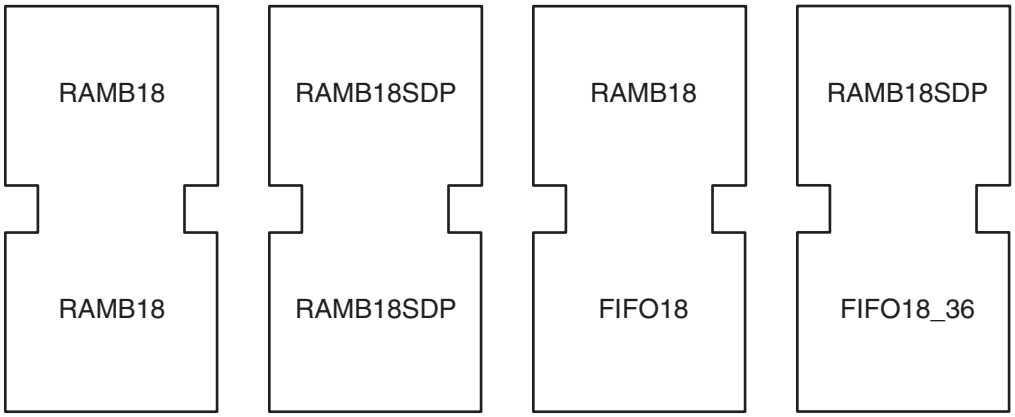


Figure 4-33: **Legal Block RAM and FIFO Combinations**

Chapter 5

Configurable Logic Blocks (CLBs)

CLB Overview

The Configurable Logic Blocks (CLBs) are the main logic resources for implementing sequential as well as combinatorial circuits. Each CLB element is connected to a switch matrix for access to the general routing matrix (shown in Figure 5-1). A CLB element contains a pair of slices. These two slices do not have direct connections to each other, and each slice is organized as a column. Each slice in a column has an independent carry chain. For each CLB, slices in the bottom of the CLB are labeled as SLICE(0), and slices in the top of the CLB are labeled as SLICE(1).

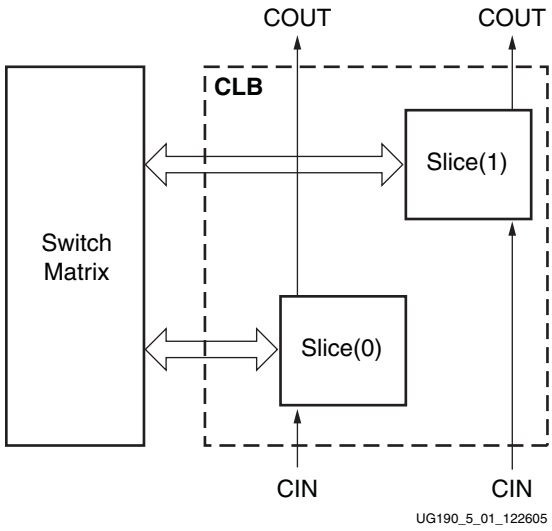


Figure 5-1: **Arrangement of Slices within the CLB**

The Xilinx tools designate slices with the following definitions. An "X" followed by a number identifies the position of each slice in a pair as well as the column position of the slice. The "X" number counts slices starting from the bottom in sequence 0, 1 (the first CLB column); 2, 3 (the second CLB column); etc. A "Y" followed by a number identifies a row of slices. The number remains the same within a CLB, but counts up in sequence from one CLB row to the next CLB row, starting from the bottom. Figure 5-2 shows four CLBs located in the bottom-left corner of the die.

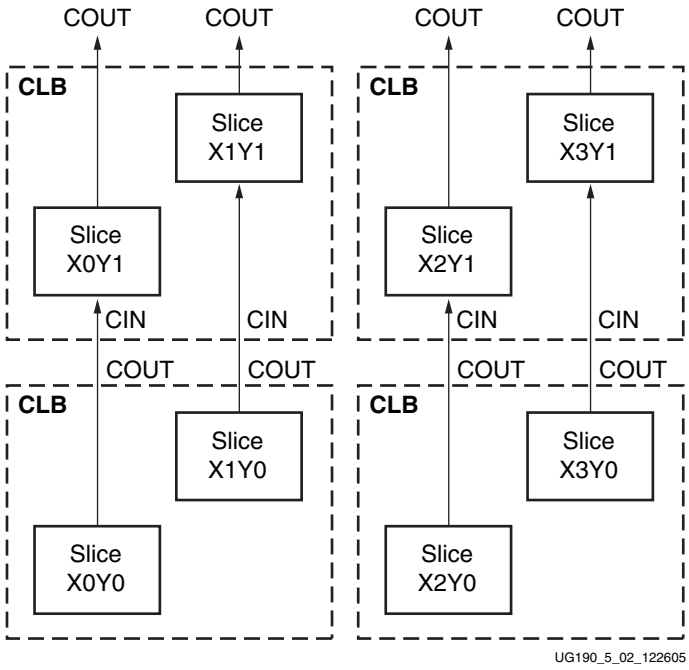


Figure 5-2: **Row and Column Relationship between CLBs and Slices**

Slice Description

Every slice contains four logic-function generators (or look-up tables), four storage elements, wide-function multiplexers, and carry logic. These elements are used by all slices to provide logic, arithmetic, and ROM functions. In addition to this, some slices support two additional functions: storing data using distributed RAM and shifting data with 32-bit registers. Slices that support these additional functions are called SLICEM; others are called SLICEL. SLICEM (shown in Figure 5-3) represents a superset of elements and connections found in all slices. SLICEL is shown in Figure 5-4.

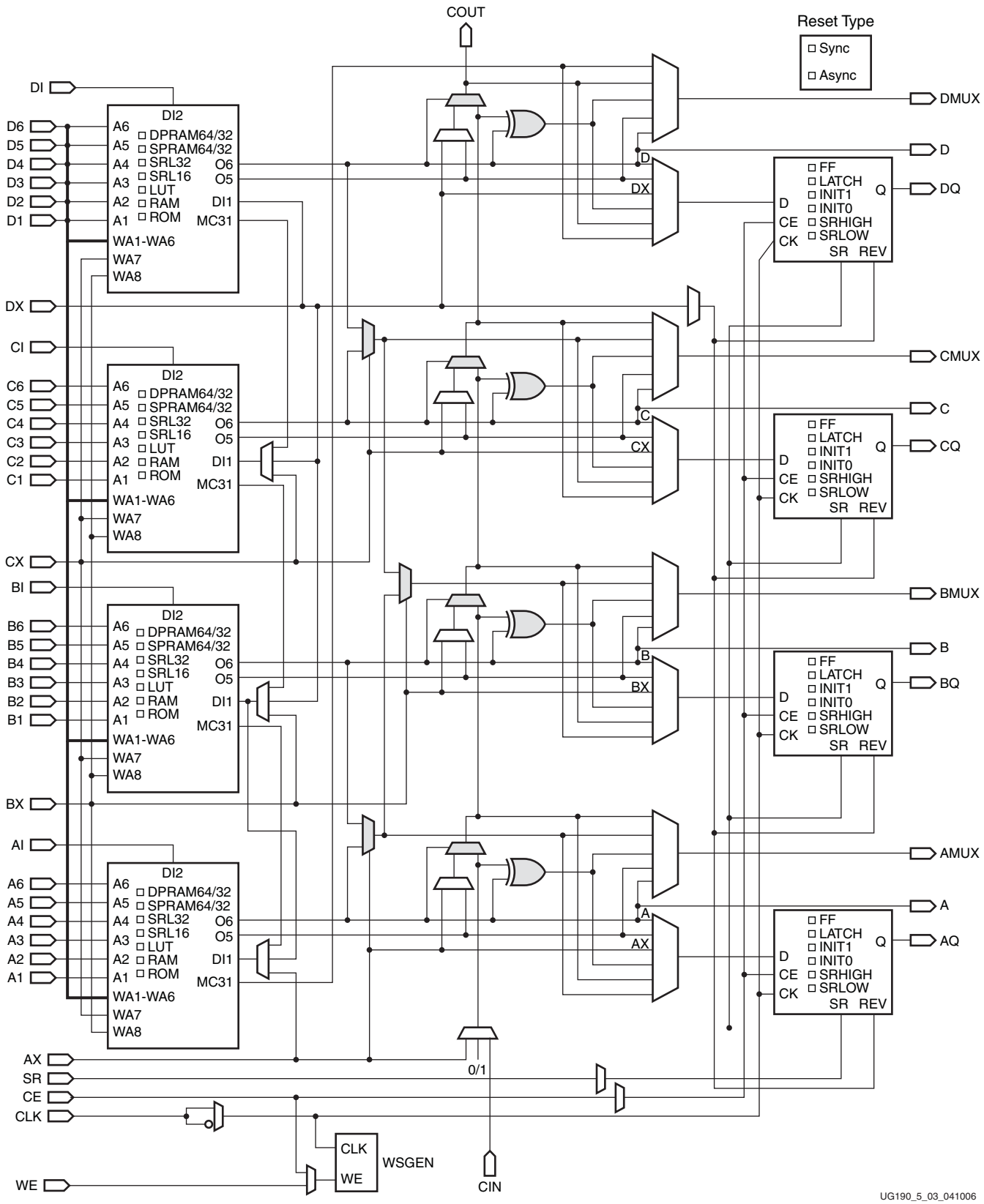


Figure 5-3: **Diagram of SLICEM**

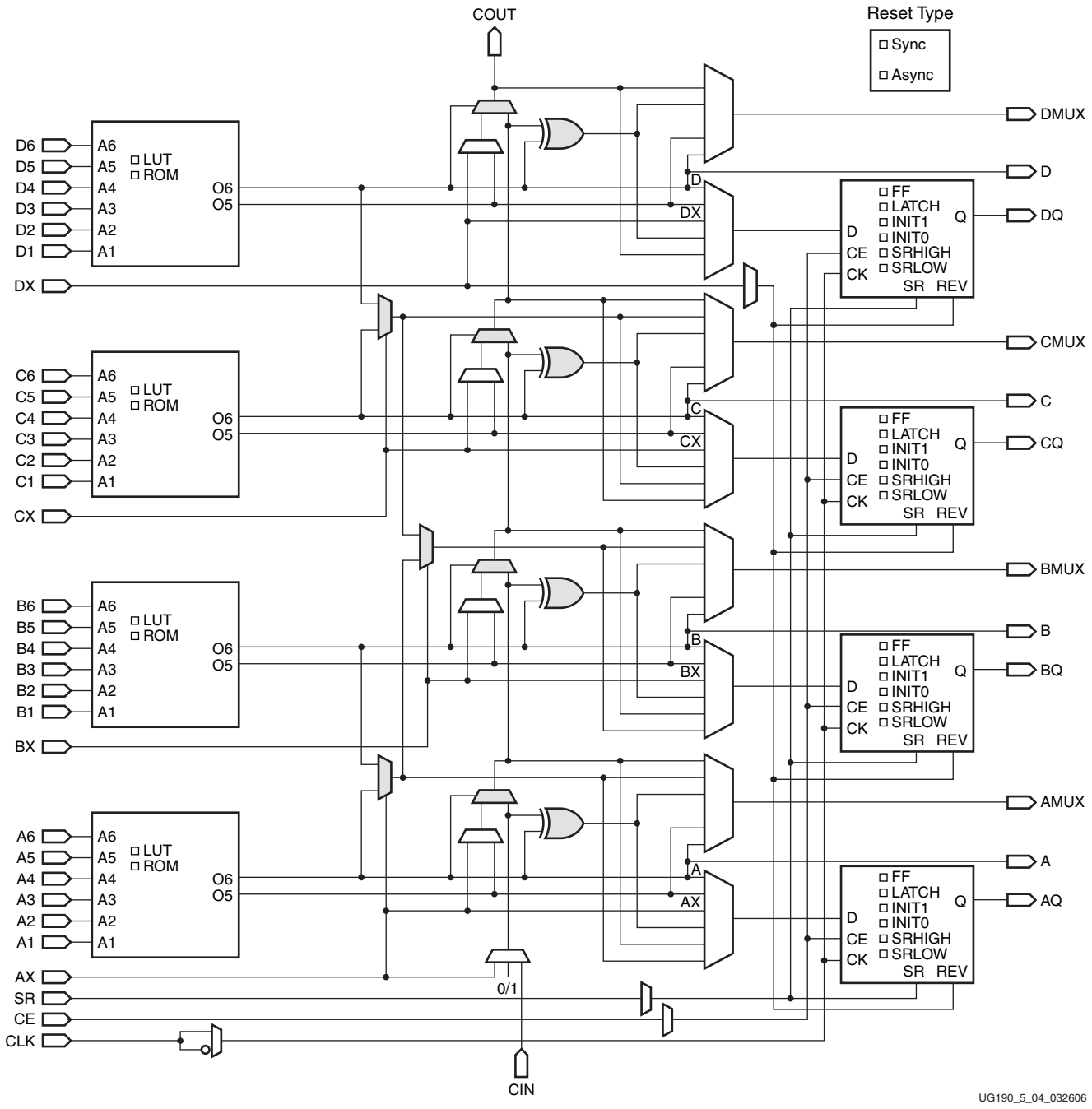


Figure 5-4: **Diagram of SLICEL**

Each CLB can contain zero or one SLICEM. Every other CLB column contains a SLICEMs. In addition, the two CLB columns to the left of the DSP48E columns both contain a SLICEL and a SLICEM.

CLB/Slice Configurations

Table 5-1 summarizes the logic resources in one CLB. Each CLB or slice can be implemented in one of the configurations listed. Table 5-2 shows the available resources in all CLBs.

Table 5-1: **Logic Resources in One CLB**

Slices	LUTs	Flip-Flops	Arithmetic and Carry Chains	Distributed RAM(1)	Shift Registers(1)
2	8	8	2	256 bits	128 bits

Notes:

1. SLICEM only, SLICEL does not have distributed RAM or shift registers.

Table 5-2: **Virtex-5 FPGA Logic Resources Available in All CLBs**

Device	CLB Array Row x Column	Number of 6-Input LUTs	Maximum Distributed RAM (Kb)	Shift Register (Kb)	Number of Flip-Flops
XC5VLX20T	60 x 26	12,480	210	105	12,480
XC5VLX30	80 x 30	19,200	320	160	19,200
XC5VFX30T	80 x 38	20,480	380	190	20,480
XC5VLX30T	80 x 30	19,200	320	160	19,200
XC5VSX35T	80 x 34	21,760	520	260	21,760
XC5VLX50	120 x 30	28,800	480	240	28,800
XC5VLX50T	120 x 30	28,800	480	240	28,800
XC5VSX50T	120 x 34	32,640	780	390	32,640
XC5VFX70T	160 x 38	44,800	820	410	44,800
XC5VLX85	120 x 54	51,840	840	420	51,840
XC5VLX85T	120 x 54	51,840	840	420	51,840
XC5VSX95T	160 x 46	58,880	1,520	760	58,880
XC5VFX100T	160 x 56	64,000	1,240	620	64,000
XC5VLX110	160 x 54	69,120	1,120	560	69,120
XC5VLX110T	160 x 54	69,120	1,120	560	69,120
XC5VFX130T	200 x 56	81,920	1,580	790	81,920
XC5VTX150T	200 x 58	92,800	1,500	750	92,800
XC5VLX155	160 x 76	97,280	1,640	820	97,280
XC5VLX155T	160 x 76	97,280	1,640	820	97,280
XC5VFX200T	240 x 68	122,880	2,280	1140	122,880
XC5VLX220	160 x 108	138,240	2,280	1140	138,240
XC5VLX220T	160 x 108	138,240	2,280	1140	138,240
XC5VSX240T	240 x 78	149,760	4,200	2100	149,760
XC5VTX240T	240 x 78	149,760	2,400	1200	149,760
XC5VLX330	240 x 108	207,360	3,420	1710	207,360
XC5VLX330T	240 x 108	207,360	3,420	1710	207,360

Look-Up Table (LUT)

The function generators in Virtex-5 FPGAs are implemented as six-input look-up tables (LUTs). There are six independent inputs (A inputs - A1 to A6) and two independent outputs (O5 and O6) for each of the four function generators in a slice (A, B, C, and D). The function generators can implement any arbitrarily defined six-input Boolean function. Each function generator can also implement two arbitrarily defined five-input Boolean functions, as long as these two functions share common inputs. Only the O6 output of the function generator is used when a six-input function is implemented. Both O5 and O6 are used for each of the five-input function generators implemented. In this case, A6 is driven High by the software. The propagation delay through a LUT is independent of the function implemented, or whether one six-input or two five-input generators are implemented. Signals from the function generators can exit the slice (through A, B, C, D output for O6 or AMUX, BMUX, CMUX, DMUX output for O5), enter the XOR dedicated gate from an O6 output (see "Fast Lookahead Carry Logic"), enter the carry-logic chain from an O5 output (see "Fast Lookahead Carry Logic"), enter the select line of the carry-logic multiplexer from O6 output (see "Fast Lookahead Carry Logic"), feed the D input of the storage element, or go to F7AMUX/F7BMUX from O6 output.

In addition to the basic LUTs, slices contain three multiplexers (F7AMUX, F7BMUX, and F8MUX). These multiplexers are used to combine up to four function generators to provide any function of seven or eight inputs in a slice. F7AMUX and F7BMUX are used to generate seven input functions from slice A and B, or C and D, while F8MUX is used to combine all slices to generate eight input functions. Functions with more than eight inputs can be implemented using multiple slices. There are no direct connections between slices to form function generators greater than eight inputs within a CLB or between slices.

Storage Elements

The storage elements in a slice can be configured as either edge-triggered D-type flip-flops or level-sensitive latches. The D input can be driven directly by a LUT output via AFFMUX, BFFMUX, CFFMUX or DFFMUX, or by the BYPASS slice inputs bypassing the function generators via AX, BX, CX, or DX input. When configured as a latch, the latch is transparent when the CLK is Low.

The control signals clock (CK), clock enable (CE), set/reset (SR), and reverse (REV) are common to all storage elements in one slice. When one flip-flop in a slice has SR or CE enabled, the other flip-flops used in the slice will also have SR or CE enabled by the common signal. Only the CLK signal has independent polarity. Any inverter placed on the clock signal is automatically absorbed. The CE, SR, and REV signals are active High. All flip-flop and latch primitives have CE and non-CE versions.

The SR signal forces the storage element into the state specified by the attribute SRHIGH or SRLOW. SRHIGH forces a logic High at the storage element output when SR is asserted, while SRLOW forces a logic Low at the storage element output. When SR is used, an optional second input (DX) forces the storage element output into the opposite state via the REV pin. The reset condition is predominant over the set condition (see Figure 5-5). Table 5-3 and Table 5-4 provide truth tables for SR and REV depending on whether SRLOW or SRHIGH is used.

Table 5-3: **Truth Table when SRLOW is Used (Default Condition)**

SR	REV	Function
0	0	No Logic Change
0	1	1

Table 5-3: **Truth Table when SRLOW is Used (Default Condition) *(Continued)***

SR	REV	Function
1	0	0
1	1	0

Table 5-4: **Truth Table when SRHIGH is Used**

SR	REV	Function
0	0	No Logic Change
0	1	0
1	0	1
1	1	0

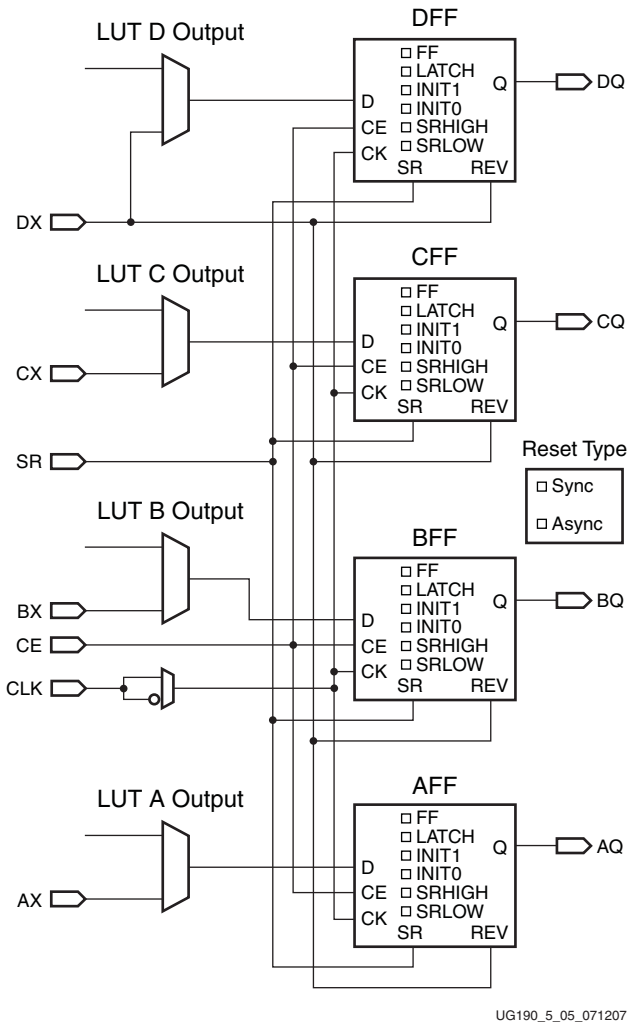


Figure 5-5: **Register/Latch Configuration in a Slice**

SRHIGH and SRLOW can be set individually for each storage element in a slice. The choice of synchronous (SYNC) or asynchronous (ASYNC) set/reset (SRTYPE) cannot be set individually for each storage element in a slice.

The initial state after configuration or global initial state is defined by separate INIT0 and INIT1 attributes. By default, setting the SRLOW attribute sets INIT0, and setting the SRHIGH attribute sets INIT1. Virtex-5 devices can set INIT0 and INIT1 independent of SRHIGH and SRLOW.

The configuration options for the set and reset functionality of a register or a latch are as follows:

- No set or reset
- Synchronous set
- Synchronous reset
- Synchronous set and reset
- Asynchronous set (preset)
- Asynchronous reset (clear)
- Asynchronous set and reset (preset and clear)

Distributed RAM and Memory (Available in SLICEM only)

Multiple LUTs in a SLICEM can be combined in various ways to store larger amount of data.

The function generators (LUTs) in SLICEMs can be implemented as a synchronous RAM resource called a distributed RAM element. RAM elements are configurable within a SLICEM to implement the following:

- Single-Port 32 x 1-bit RAM
- Dual-Port 32 x 1-bit RAM
- Quad-Port 32 x 2-bit RAM
- Simple Dual-Port 32 x 6-bit RAM
- Single-Port 64 x 1-bit RAM
- Dual-Port 64 x 1-bit RAM
- Quad-Port 64 x 1-bit RAM
- Simple Dual-Port 64 x 3-bit RAM
- Single-Port 128 x 1-bit RAM
- Dual-Port 128 x 1-bit RAM
- Single-Port 256 x 1-bit RAM

Distributed RAM modules are synchronous (write) resources. A synchronous read can be implemented with a storage element or a flip-flop in the same slice. By placing this flip-flop, the distributed RAM performance is improved by decreasing the delay into the clock-to-out value of the flip-flop. However, an additional clock latency is added. The distributed elements share the same clock input. For a write operation, the Write Enable (WE) input, driven by either the CE or WE pin of a SLICEM, must be set High.

Table 5-5 shows the number of LUTs (four per slice) occupied by each distributed RAM configuration.

Table 5-5: **Distributed RAM Configuration**

RAM	Number of LUTs
32 x 1S	1
32 x 1D	2
32 x 2Q(2)	4
32 x 6SDP(2)	4
64 x 1S	1
64 x 1D	2
64 x 1Q(3)	4
64 x 3SDP(3)	4
128 x 1S	2
128 x 1D	4
256 x 1S	4

Notes:

1. S = single-port configuration; D = dual-port configuration; Q = quad-port configuration; SDP = simple dual-port configuration.
2. RAM32M is the associated primitive for this configuration.
3. RAM64M is the associated primitive for this configuration.

For single-port configurations, distributed RAM has a common address port for synchronous writes and asynchronous reads. For dual-port configurations, distributed RAM has one port for synchronous writes and asynchronous reads, and another port for asynchronous reads. In simple dual-port configuration, there is no data out (read port) from the write port. For quad-port configurations, distributed RAM has one port for synchronous writes and asynchronous reads, and three additional ports for asynchronous reads.

In single-port mode, read and write addresses share the same address bus. In dual-port mode, one function generator is connected with the shared read and write port address. The second function generator has the A inputs connected to a second read-only port address and the WA inputs shared with the first read/write port address.

Figure 5-6 through Figure 5-14 illustrate various example distributed RAM configurations occupying one SLICEM. When using x2 configuration (RAM32X2Q), A6 and WA6 are driven High by the software to keep O5 and O6 independent.

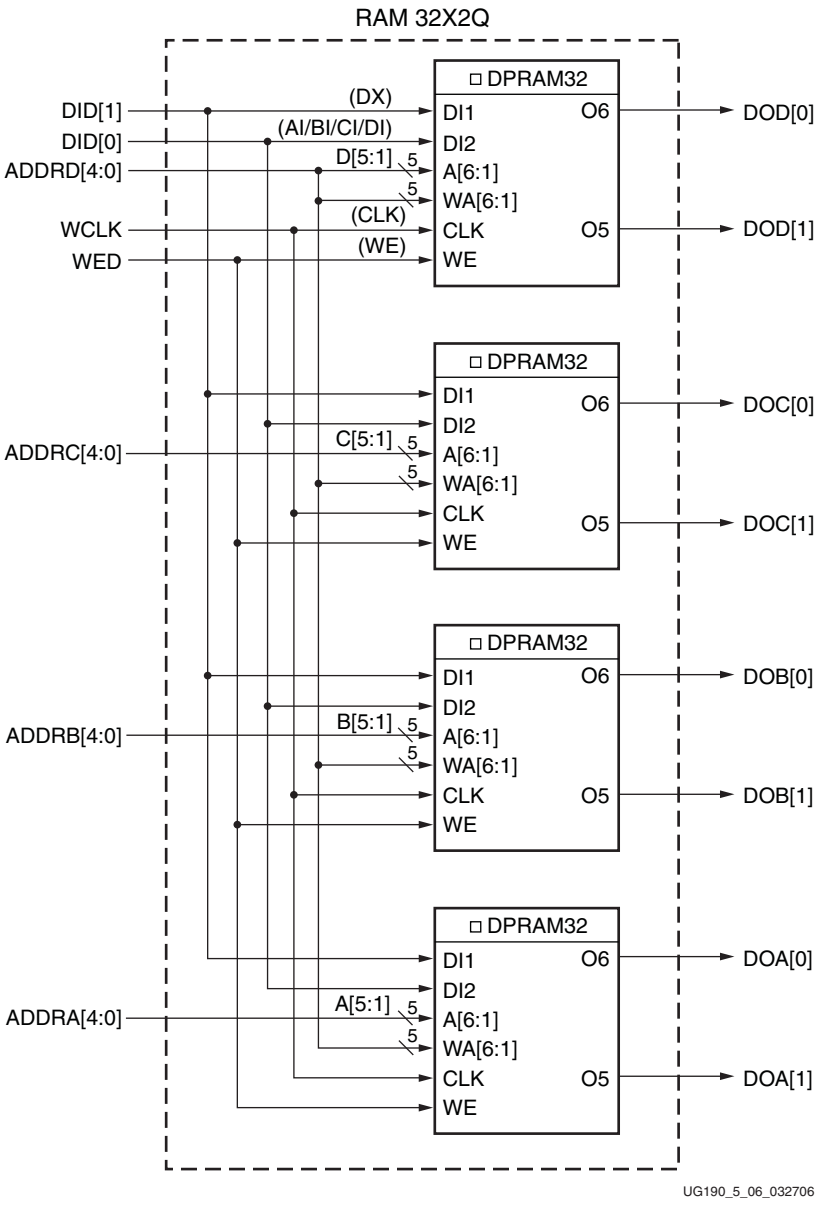


Figure 5-6: **Distributed RAM (RAM32X2Q)**

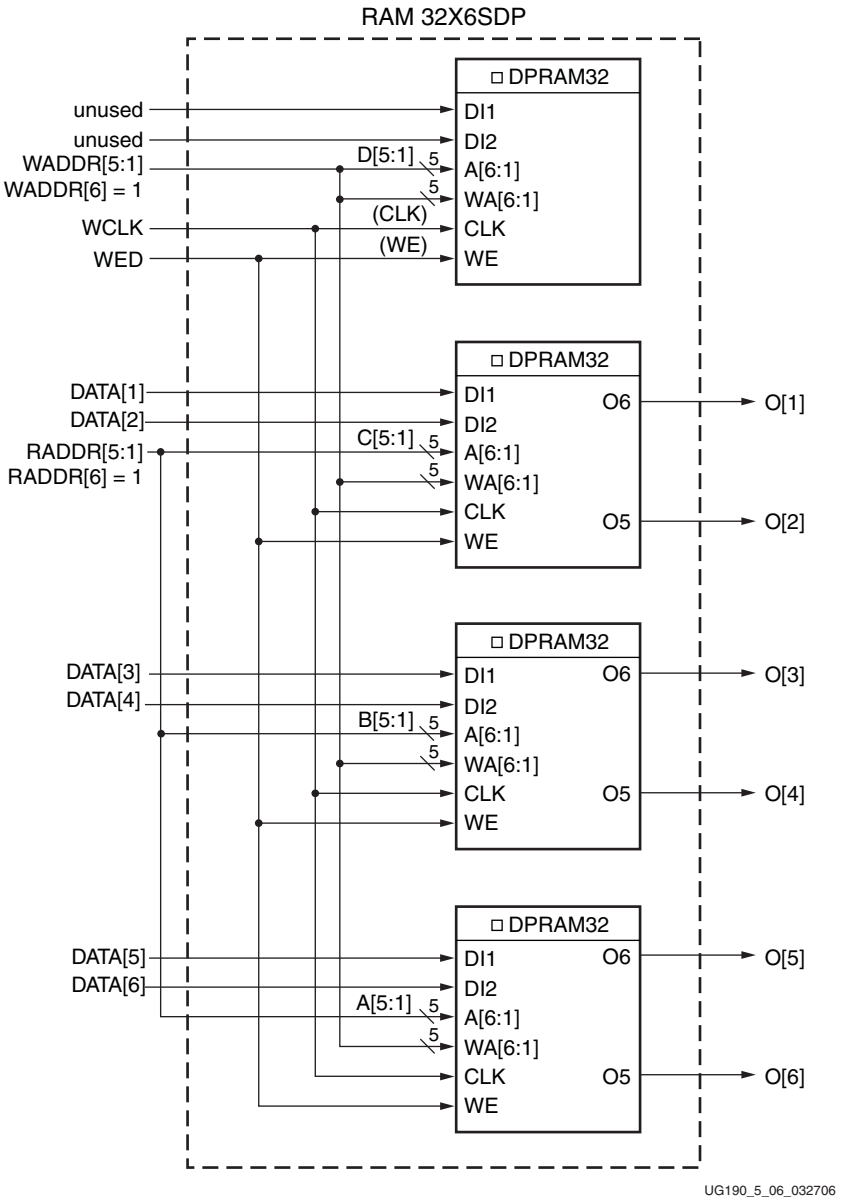


Figure 5-7: **Distributed RAM (RAM32X6SDP)**

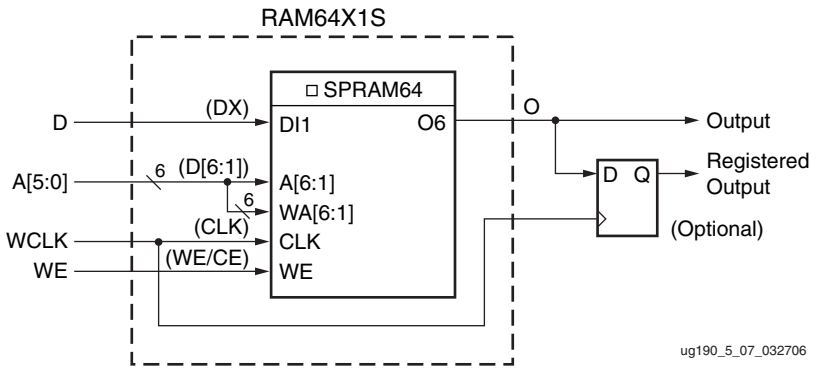


Figure 5-8: **Distributed RAM (RAM64X1S)**

If four single-port 64 x 1-bit modules are built, the four RAM64X1S primitives can occupy a SLICEM, as long as they share the same clock, write enable, and shared read and write port address inputs. This configuration equates to 64 x 4-bit single-port distributed RAM.

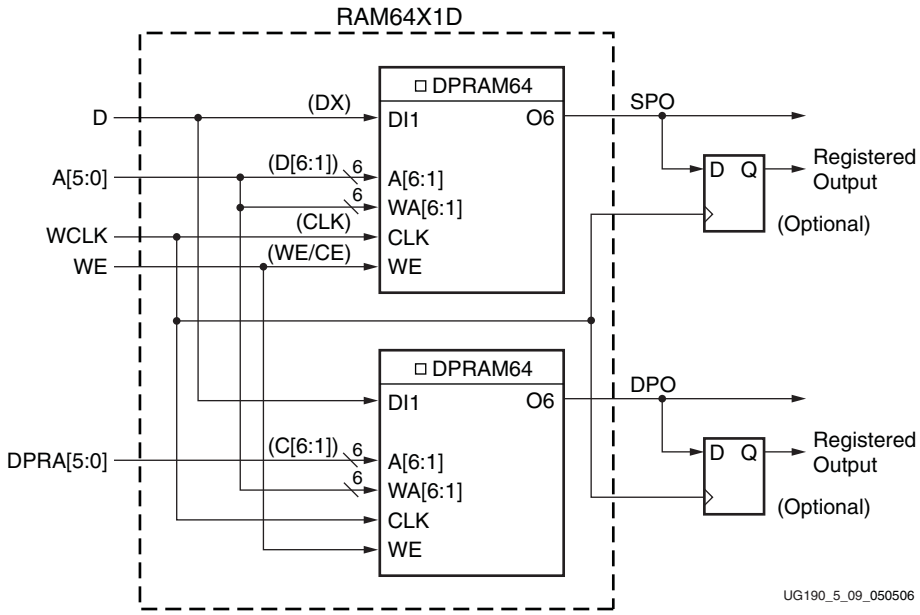


Figure 5-9: **Distributed RAM (RAM64X1D)**

If two dual-port 64 x 1-bit modules are built, the two RAM64X1D primitives can occupy a SLICEM, as long as they share the same clock, write enable, and shared read and write port address inputs. This configuration equates to 64 x 2-bit dual-port distributed RAM.

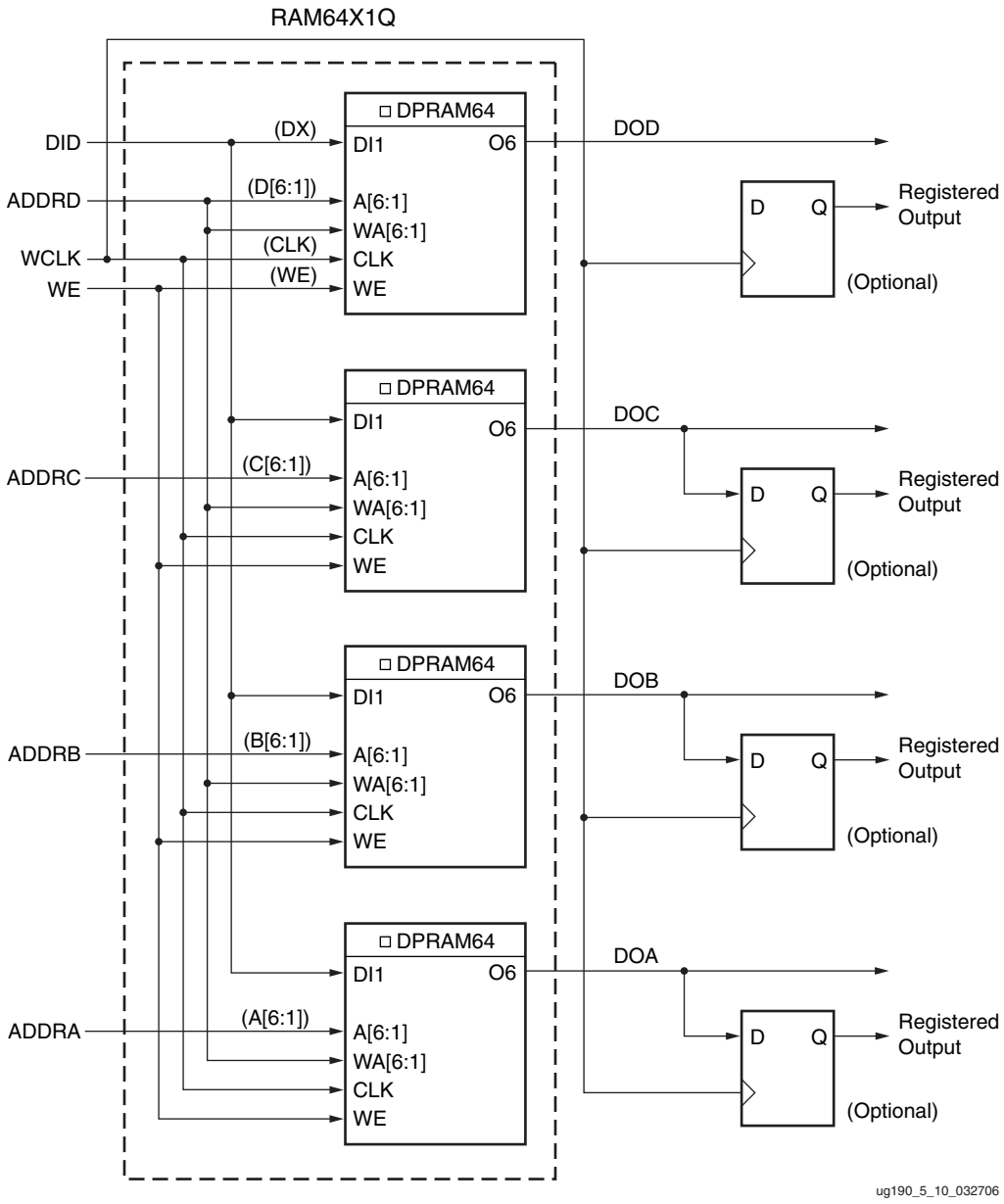


Figure 5-10: **Distributed RAM (RAM64X1Q)**

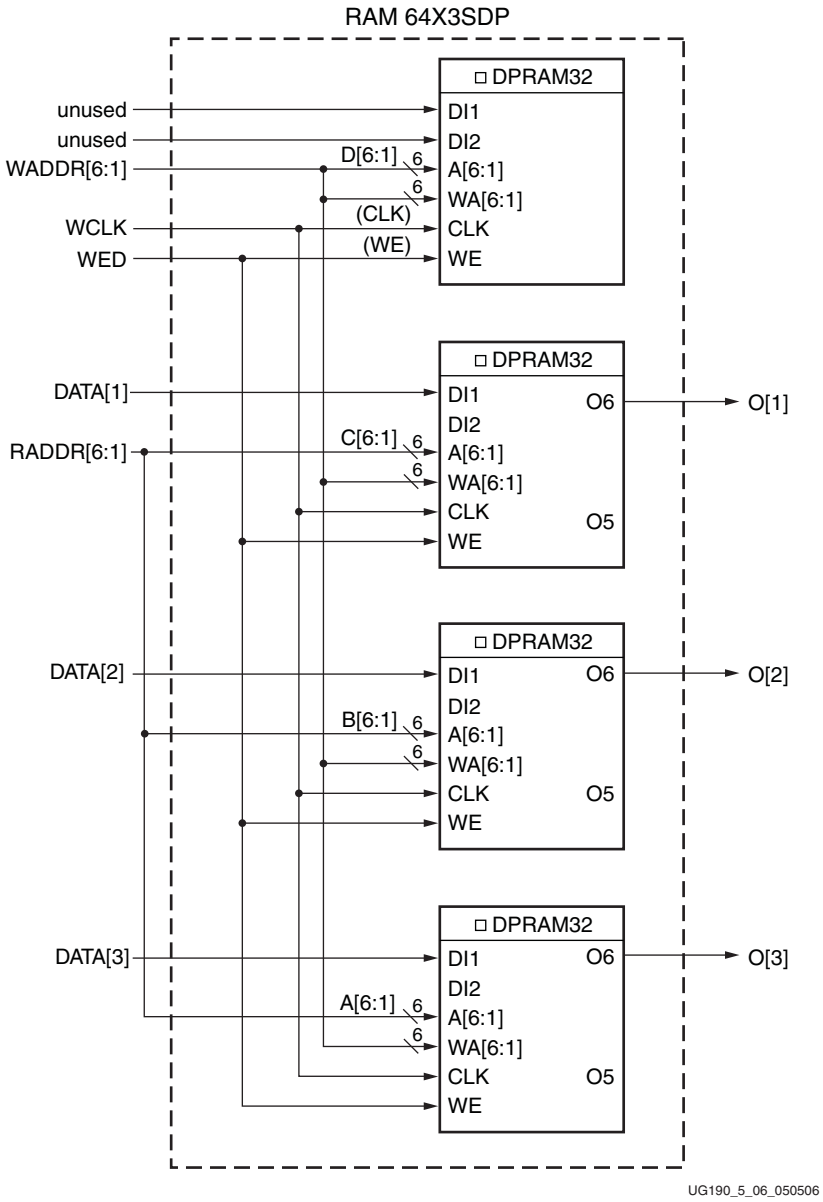


Figure 5-11: **Distributed RAM (RAM64X3SDP)**

Implementation of distributed RAM configurations with depth greater than 64 requires the usage of wide-function multiplexers (F7AMUX, F7BMUX, and F8MUX).

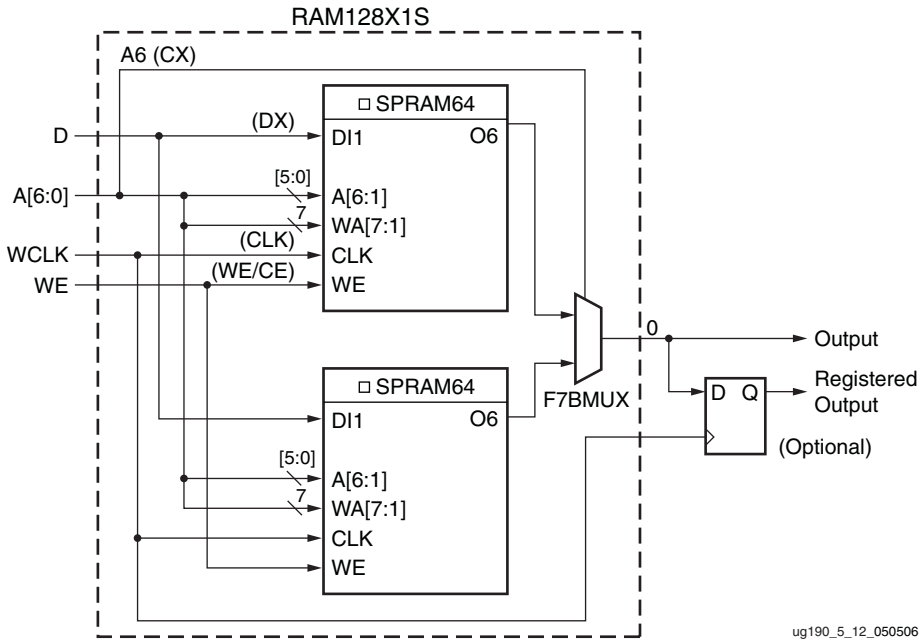


Figure 5-12: **Distributed RAM (RAM128X1S)**

If two single-port 128 x 1-bit modules are built, the two RAM128X1S primitives can occupy a SLICEM, as long as they share the same clock, write enable, and shared read and write port address inputs. This configuration equates to 128 x 2-bit single-port distributed RAM.

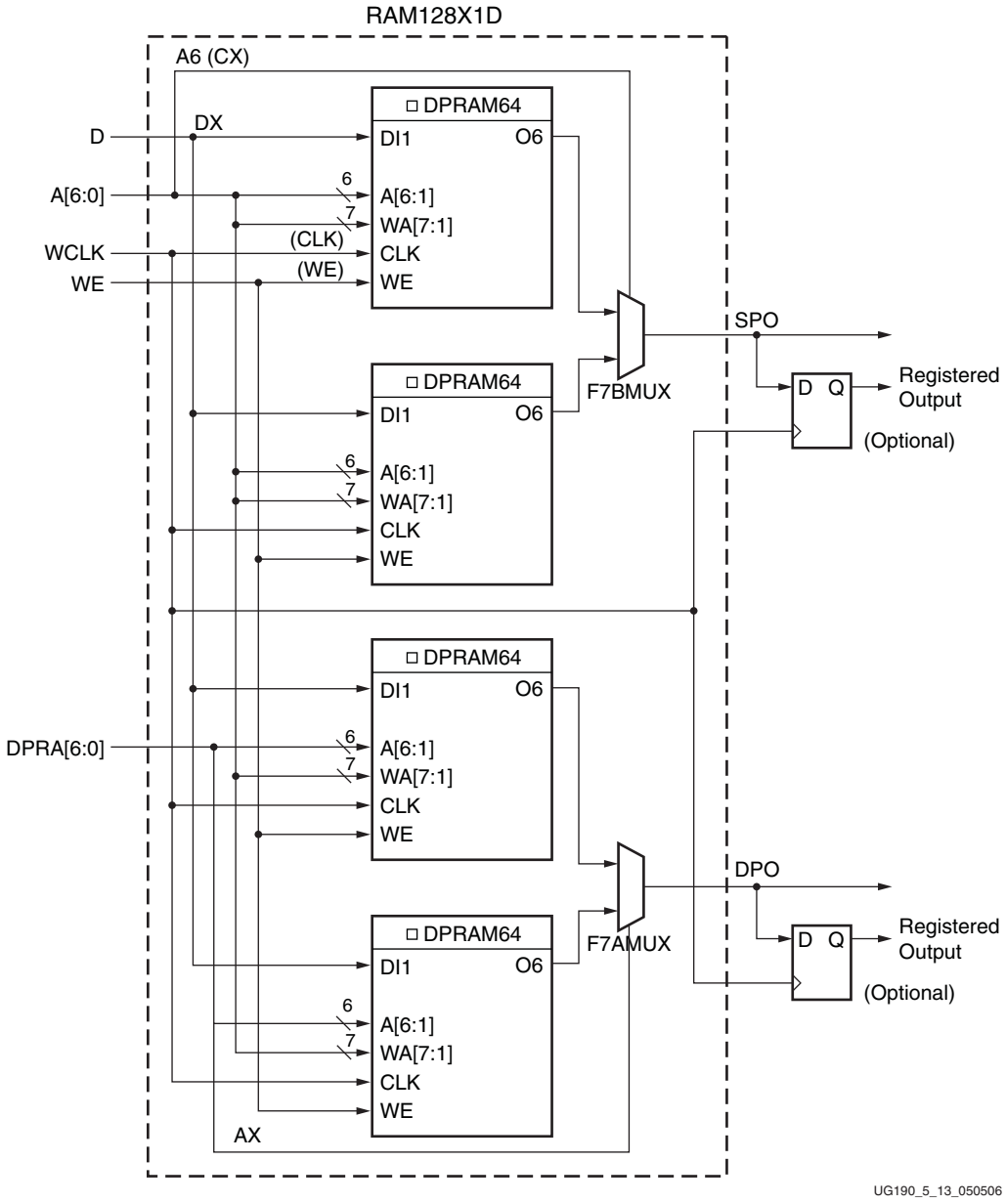


Figure 5-13: **Distributed RAM (RAM128X1D)**

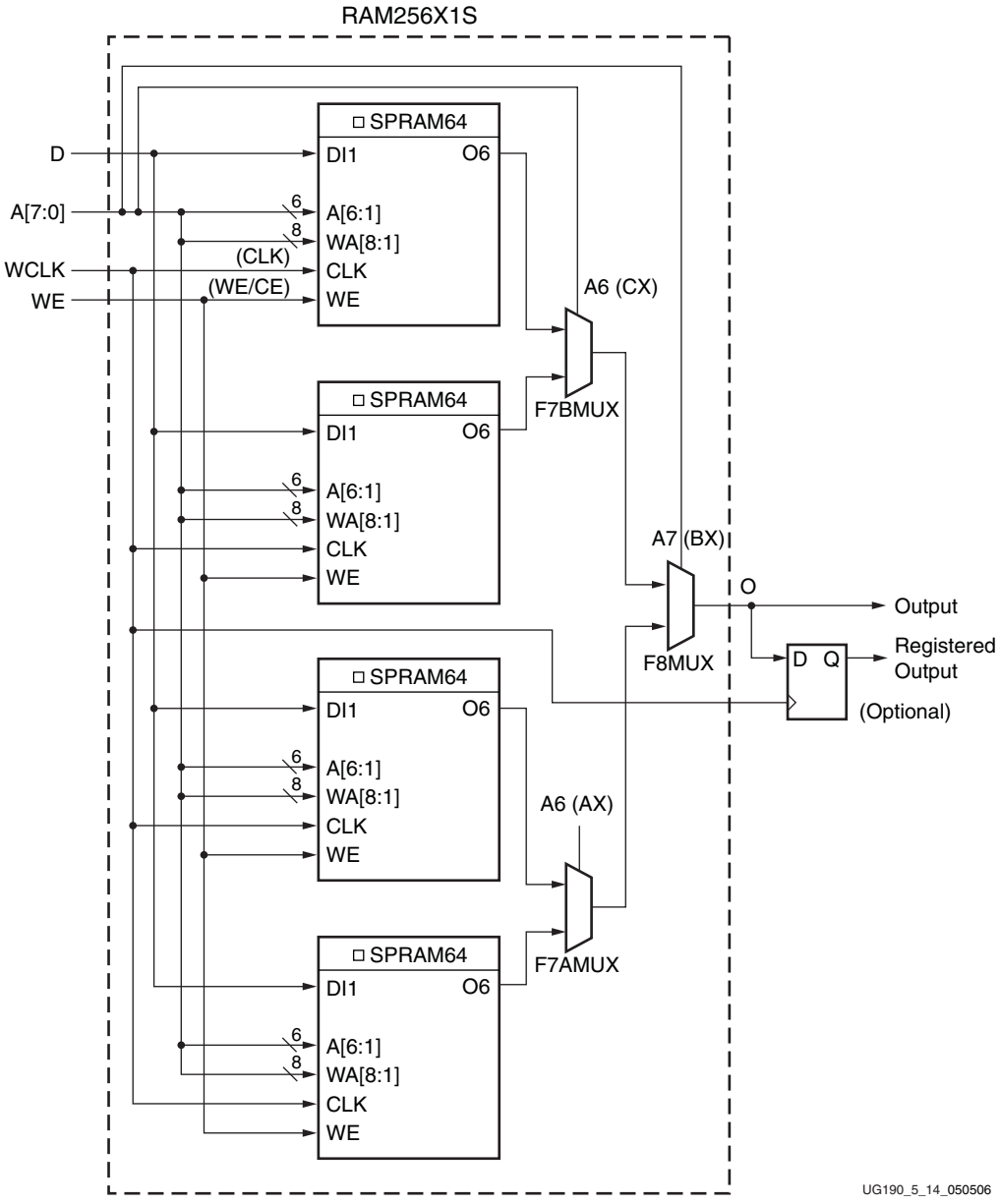


Figure 5-14: **Distributed RAM (RAM256X1S)**

Distributed RAM configurations greater than the provided examples require more than one SLICEM. There are no direct connections between slices to form larger distributed RAM configurations within a CLB or between slices.

Distributed RAM Data Flow

Synchronous Write Operation

The synchronous write operation is a single clock-edge operation with an active-High write-enable (WE) feature. When WE is High, the input (D) is loaded into the memory location at address A.

Asynchronous Read Operation

The output is determined by the address A (for single-port mode output/SPO output of dual-port mode), or address DPRA (DPO output of dual-port mode). Each time a new address is applied to the address pins, the data value in the memory location of that address is available on the output after the time delay to access the LUT. This operation is asynchronous and independent of the clock signal.

Distributed RAM Summary

- Single-port and dual-port modes are available in SLICEMs.
- A write operation requires one clock edge.
- Read operations are asynchronous (Q output).
- The data input has a setup-to-clock timing specification.

Read Only Memory (ROM)

Each function generator in SLICEMs and SLICELs can implement a 64 x 1-bit ROM. Three configurations are available: ROM64x1, ROM128x1, and ROM256x1. ROM contents are loaded at each device configuration. Table 5-6 shows the number of LUTs occupied by each ROM configuration.

Table 5-6: **ROM Configuration**

ROM	Number of LUTs
64 x 1	1
128 x 1	2
256 x 1	4

Shift Registers (Available in SLICEM only)

A SLICEM function generator can also be configured as a 32-bit shift register without using the flip-flops available in a slice. Used in this way, each LUT can delay serial data anywhere from one to 32 clock cycles. The shiftin D (DI1 LUT pin) and shiftout Q31 (MC31 LUT pin) lines cascade LUTs to form larger shift registers. The four LUTs in a SLICEM are thus cascaded to produce delays up to 128 clock cycles. It is also possible to combine shift registers across more than one SLICEM. Note that there are no direct connections between slices to form longer shift registers, nor is the MC31 output at LUT B/C/D available. The resulting programmable delays can be used to balance the timing of data pipelines.

Applications requiring delay or latency compensation use these shift registers to develop efficient designs. Shift registers are also useful in synchronous FIFO and content addressable memory (CAM) designs.

The write operation is synchronous with a clock input (CLK) and an optional clock enable (CE). A dynamic read access is performed through the 5-bit address bus, A[4:0]. The LSB of the LUT is unused and the software automatically ties it to a logic High. The configurable shift registers cannot be set or reset. The read is asynchronous; however, a storage element

or flip-flop is available to implement a synchronous read. In this case, the clock-to-out of the flip-flop determines the overall delay and improves performance. However, one additional cycle of clock latency is added. Any of the 32 bits can be read out asynchronously (at the O6 LUT outputs) by varying the 5-bit address. This capability is useful in creating smaller shift registers (less than 32 bits). For example, when building a 13-bit shift register, simply set the address to the 13th bit. Figure 5-15 is a logic block diagram of a 32-bit shift register.

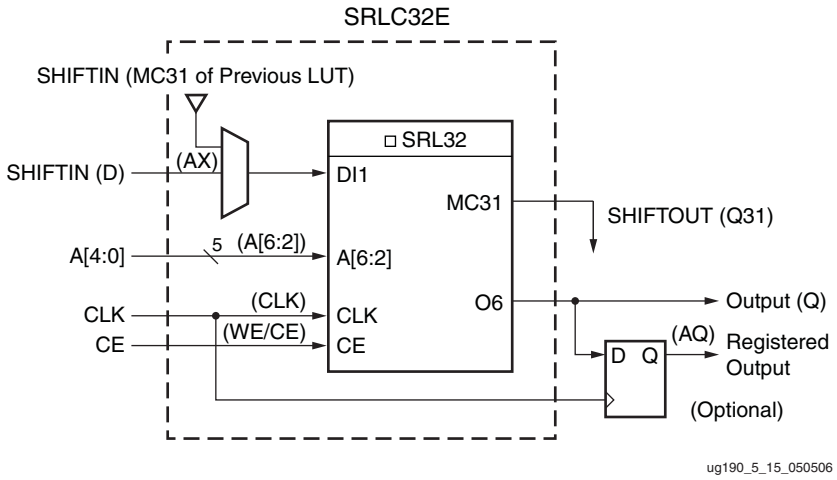


Figure 5-15: **32-bit Shift Register Configuration**

Figure 5-16 illustrates an example shift register configuration occupying one function generator.

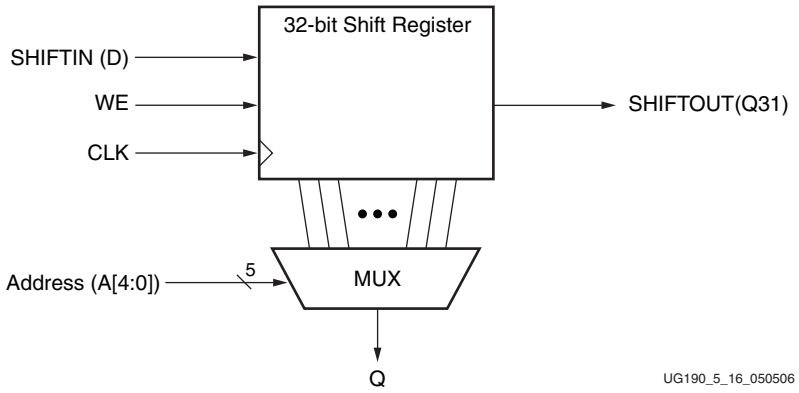


Figure 5-16: **Representation of a Shift Register**

Figure 5-17 shows two 16-bit shift registers. The example shown can be implemented in a single LUT.

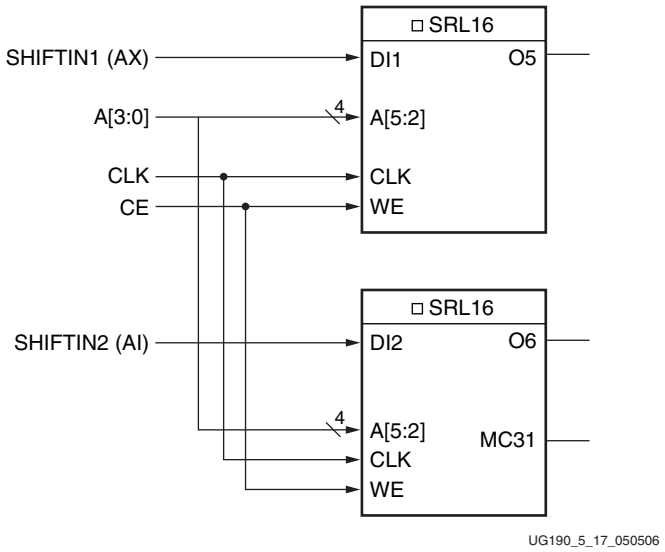


Figure 5-17: **Dual 16-bit Shift Register Configuration**

As mentioned earlier, an additional output (MC31) and a dedicated connection between shift registers allows connecting the last bit of one shift register to the first bit of the next, without using the LUT O6 output. Longer shift registers can be built with dynamic access to any bit in the chain. The shift register chaining and the F7AMUX, F7BMUX, and F8MUX multiplexers allow up to a 128-bit shift register with addressable access to be implemented in one SLICEM. Figure 5-18 through Figure 5-20 illustrate various example shift register configurations that can occupy one SLICEM.

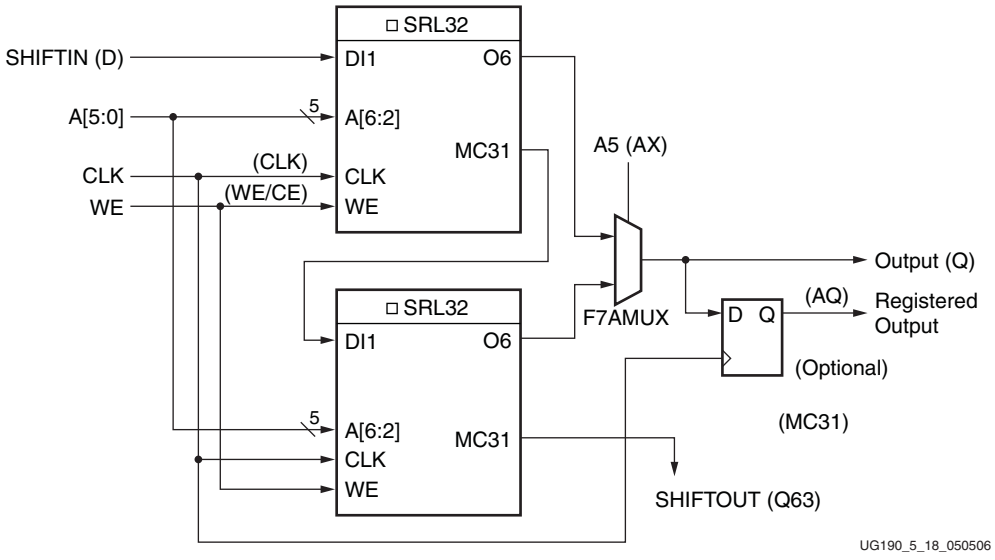


Figure 5-18: **64-bit Shift Register Configuration**

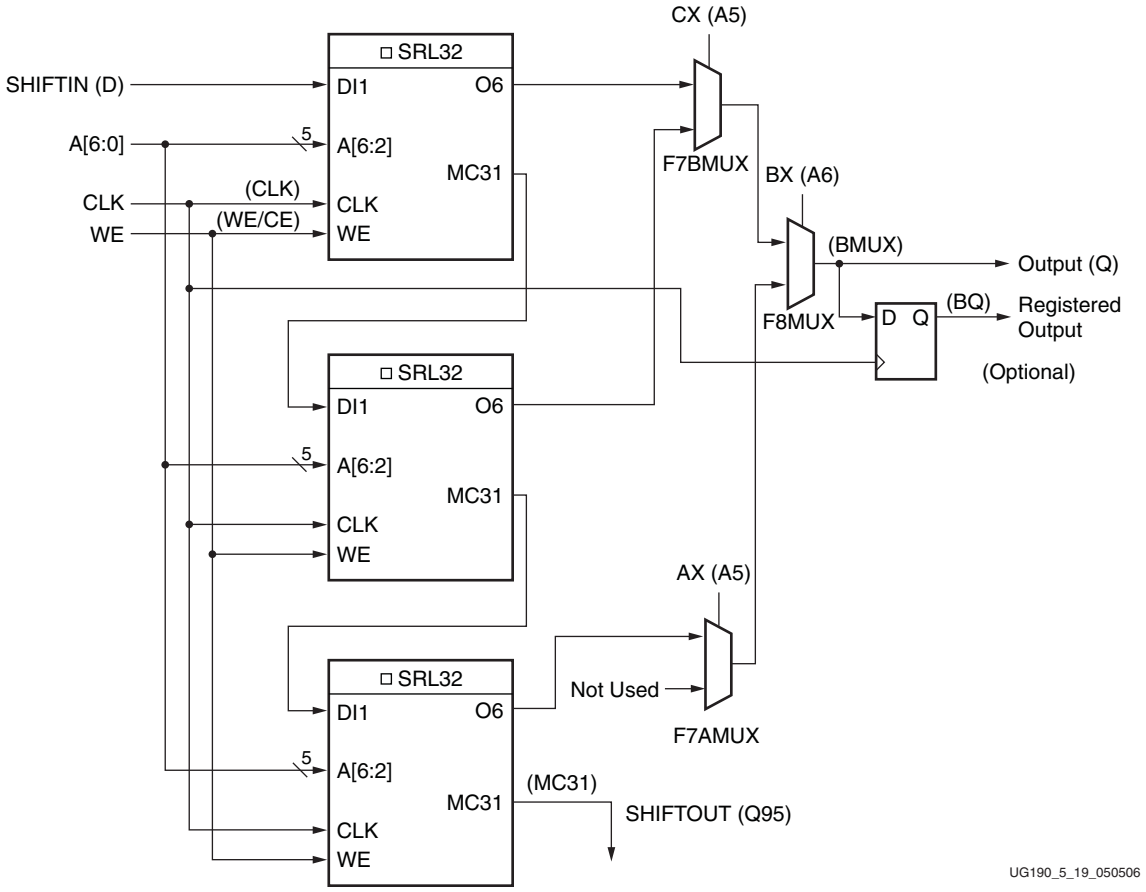


Figure 5-19: **96-bit Shift Register Configuration**

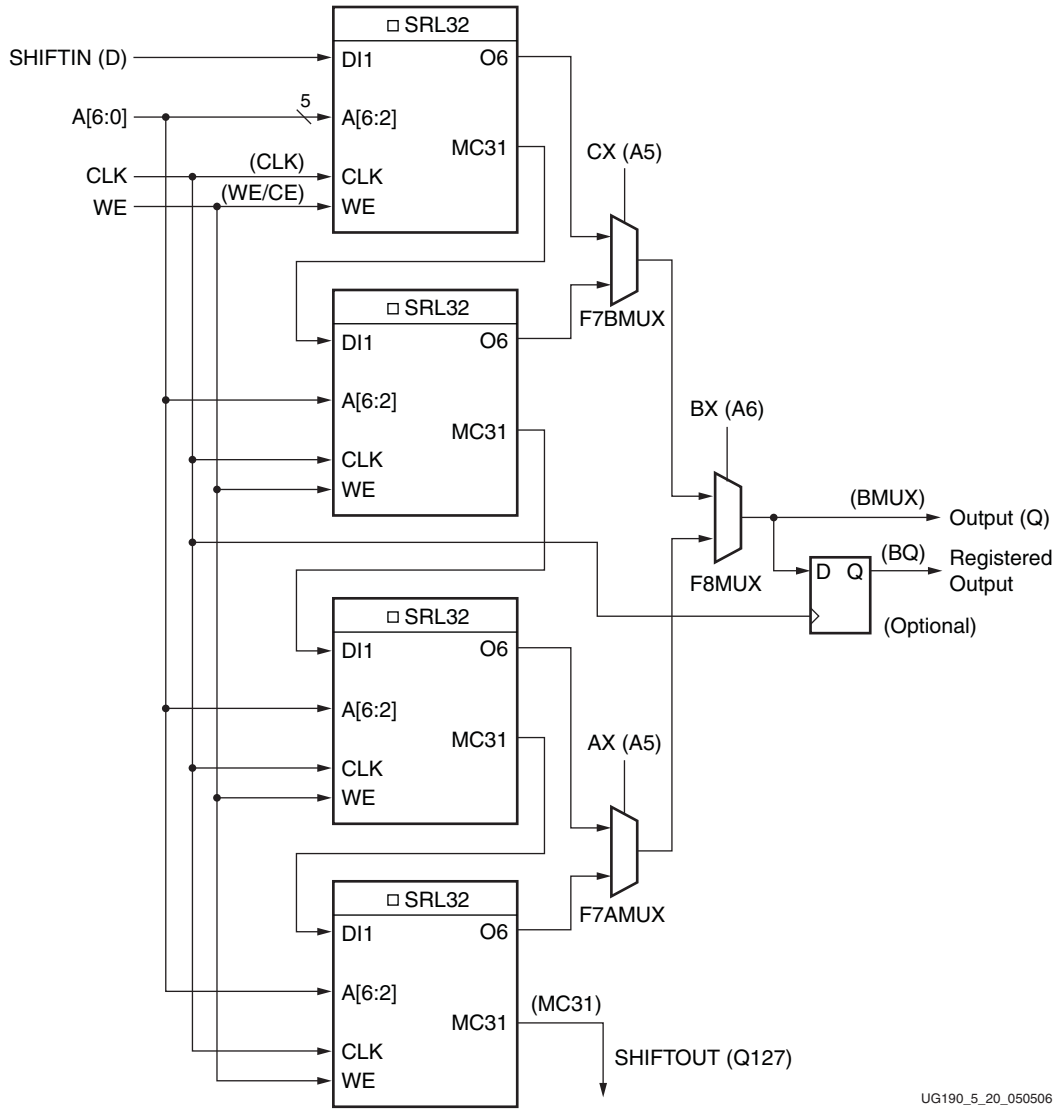


Figure 5-20: **128-bit Shift Register Configuration**

It is possible to create shift registers longer than 128 bits across more than one SLICEM. However, there are no direct connections between slices to form these shift registers.

Shift Register Data Flow

Shift Operation

The shift operation is a single clock-edge operation, with an active-High clock enable feature. When enable is High, the input (D) is loaded into the first bit of the shift register. Each bit is also shifted to the next highest bit position. In a cascadable shift register configuration, the last bit is shifted out on the M31 output.

The bit selected by the 5-bit address port (A[4:0]) appears on the Q output.

Dynamic Read Operation

The Q output is determined by the 5-bit address. Each time a new address is applied to the 5-input address pins, the new bit position value is available on the Q output after the time

delay to access the LUT. This operation is asynchronous and independent of the clock and clock-enable signals.

Static Read Operation

If the 5-bit address is fixed, the Q output always uses the same bit position. This mode implements any shift-register length from 1 to 16 bits in one LUT. The shift register length is (N+1), where N is the input address (0 – 31).

The Q output changes synchronously with each shift operation. The previous bit is shifted to the next position and appears on the Q output.

Shift Register Summary

- A shift operation requires one clock edge.
- Dynamic-length read operations are asynchronous (Q output).
- Static-length read operations are synchronous (Q output).
- The data input has a setup-to-clock timing specification.
- In a cascadable configuration, the Q31 output always contains the last bit value.
- The Q31 output changes synchronously after each shift operation.

Multiplexers

Function generators and associated multiplexers in Virtex-5 FPGAs can implement the following:

- 4:1 multiplexers using one LUT
- 8:1 multiplexers using two LUTs
- 16:1 multiplexers using four LUTs

These wide input multiplexers are implemented in one level or logic (or LUT) using the dedicated F7AMUX, F7BMUX, and F8MUX multiplexers. These multiplexers allow LUT combinations of up to four LUTs in a slice.

Designing Large Multiplexers

4:1 Multiplexer

Each LUT can be configured into a 4:1 MUX. The 4:1 MUX can be implemented with a flip-flop in the same slice. Up to four 4:1 MUXes can be implemented in a slice, as shown in Figure 5-21.

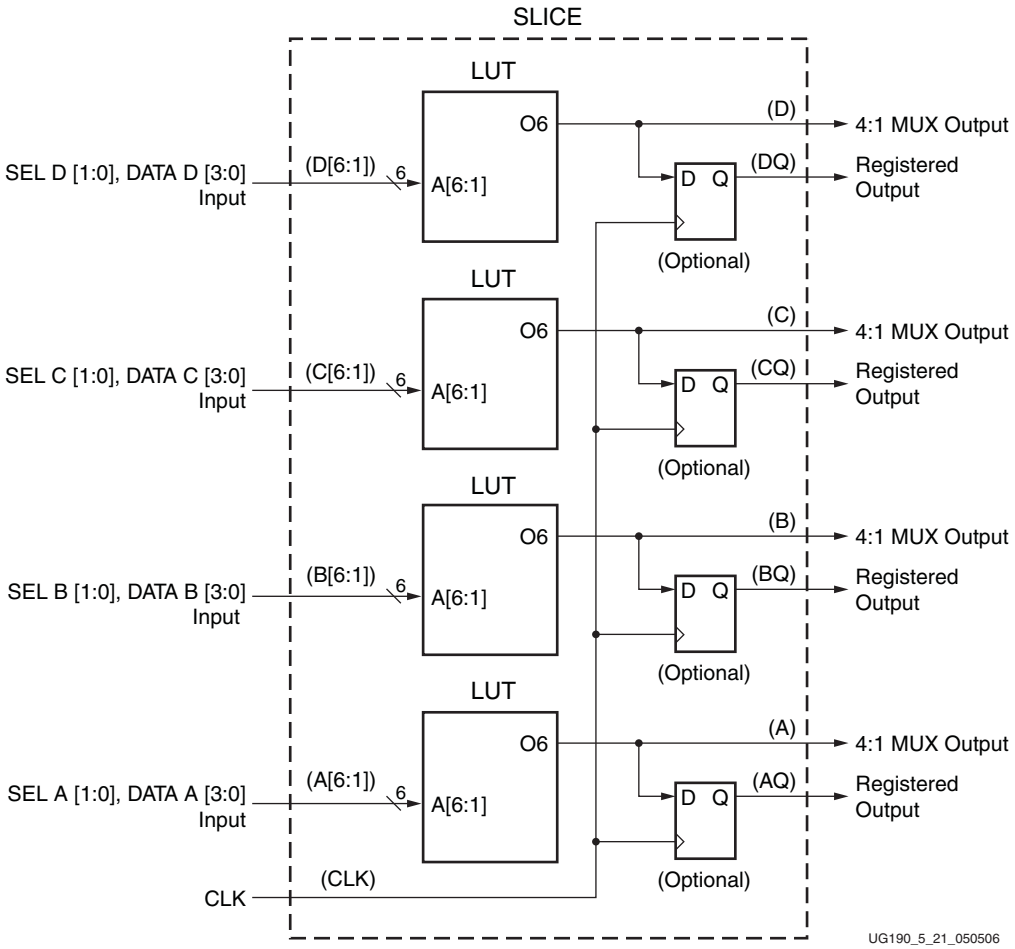


Figure 5-21: **Four 4:1 Multiplexers in a Slice**

8:1 Multiplexer

Each slice has an F7AMUX and an F7BMUX. These two muxes combine the output of two LUTs to form a combinatorial function up to 13 inputs (or an 8:1 MUX). Up to two 8:1 MUXes can be implemented in a slice, as shown in Figure 5-22.

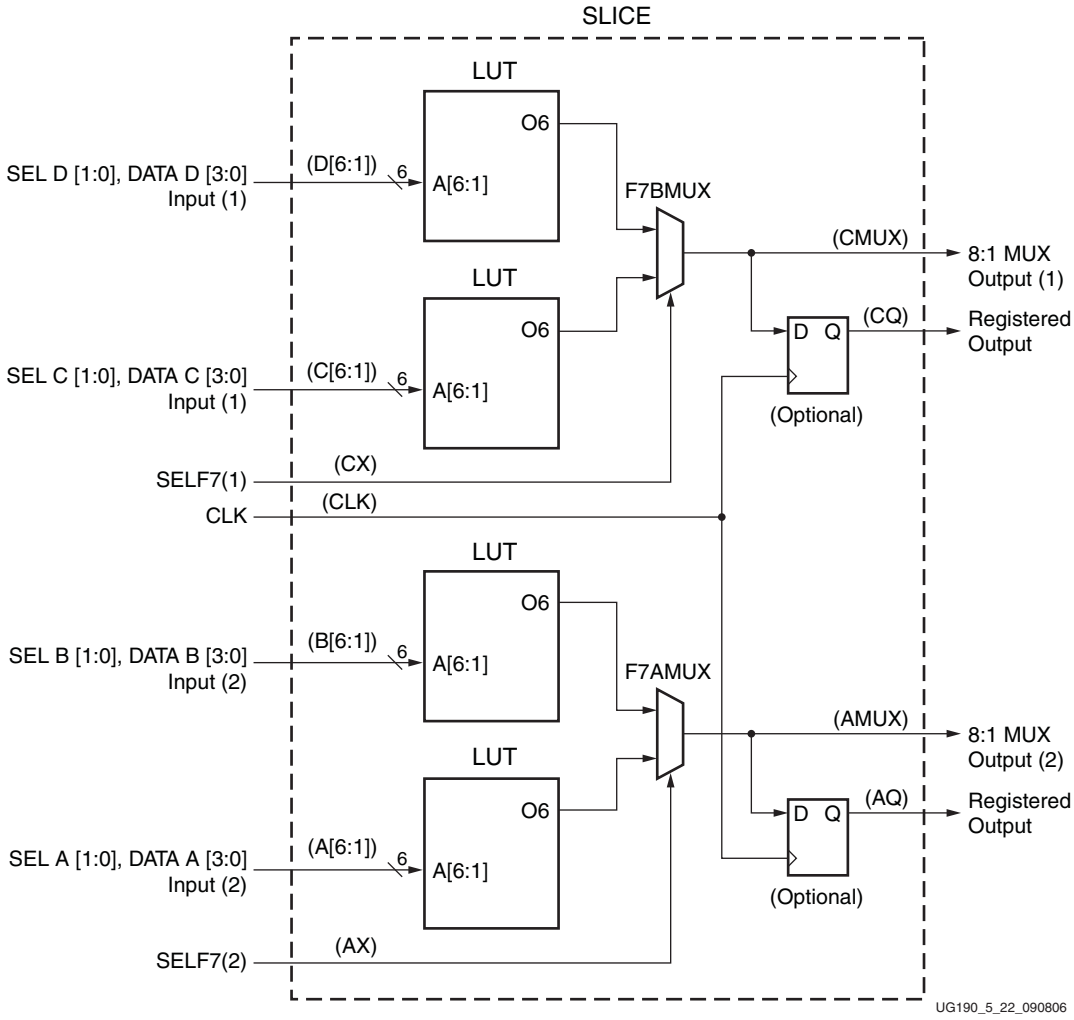


Figure 5-22: **Two 8:1 Multiplexers in a Slice**

16:1 Multiplexer

Each slice has an F8MUX. F8MUX combines the outputs of F7AMUX and F7BMUX to form a combinatorial function up to 27 inputs (or a 16:1 MUX). Only one 16:1 MUX can be implemented in a slice, as shown in Figure 5-23.

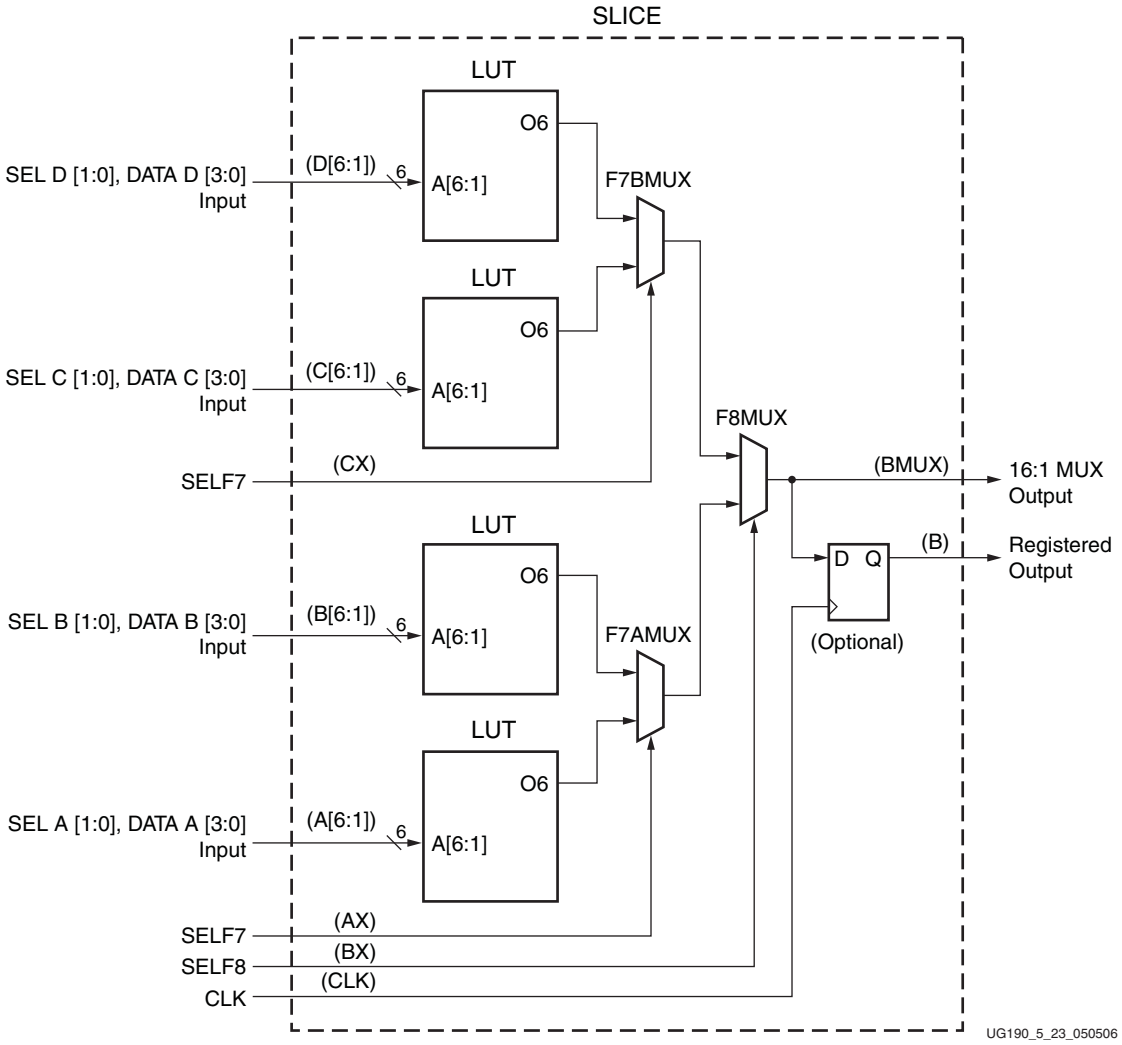


Figure 5-23: **16:1 Multiplexer in a Slice**

It is possible to create multiplexers wider than 16:1 across more than one SLICEM. However, there are no direct connections between slices to form these wide multiplexers.

Fast Lookahead Carry Logic

In addition to function generators, dedicated carry logic is provided to perform fast arithmetic addition and subtraction in a slice. A Virtex-5 FPGA CLB has two separate carry chains, as shown in Figure 5-1. The carry chains are cascadable to form wider add/subtract logic, as shown in Figure 5-2.

The carry chain in the Virtex-5 device is running upward and has a height of four bits per slice. For each bit, there is a carry multiplexer (MUXCY) and a dedicated XOR gate for adding/subtracting the operands with a selected carry bits. The dedicated carry path and

carry multiplexer (MUXCY) can also be used to cascade function generators for implementing wide logic functions.

Figure 5-24 illustrates the carry chain with associated logic elements in a slice.

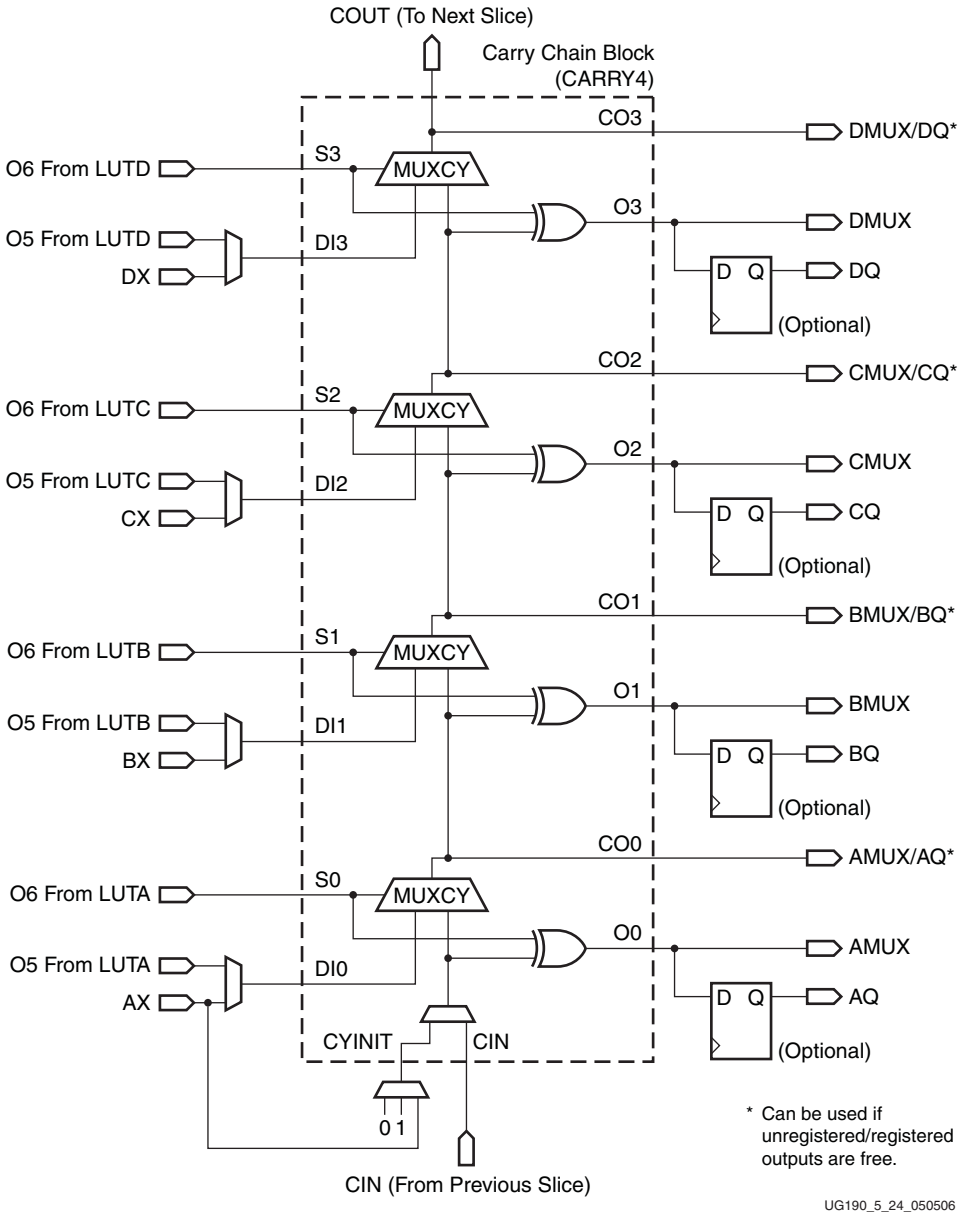


Figure 5-24: **Fast Carry Logic Path and Associated Elements**

The carry chains carry lookahead logic along with the function generators. There are ten independent inputs (S inputs – S0 to S3, DI inputs – DI1 to DI4, CYINIT and CIN) and eight independent outputs (O outputs – O0 to O3, and CO outputs – CO0 to CO3).

The S inputs are used for the "propagate" signals of the carry lookahead logic. The "propagate" signals are sourced from the O6 output of a function generator. The DI inputs are used for the "generate" signals of the carry lookahead logic. The "generate" signals are sourced from either the O5 output of a function generator or the BYPASS input (AX, BX, CX, or DX) of a slice. The former input is used to create a multiplier, while the latter is used

to create an adder/accumulator. CYINIT is the CIN of the first bit in a carry chain. The CYINIT value can be 0 (for add), 1 (for subtract), or AX input (for the dynamic first carry bit). The CIN input is used to cascade slices to form a longer carry chain. The O outputs contain the sum of the addition/subtraction. The CO outputs compute the carry out for each bit. CO3 is connected to COUT output of a slice to form a longer carry chain by cascading multiple slices. The propagation delay for an adder increases linearly with the number of bits in the operand, as more carry chains are cascaded. The carry chain can be implemented with a storage element or a flip-flop in the same slice.

CLB / Slice Timing Models

Due to the large size and complexity of Virtex-5 FPGAs, understanding the timing associated with the various paths and functional elements is a difficult and important task. Although it is not necessary to understand the various timing parameters to implement most designs using Xilinx software, a thorough timing model can assist advanced users in analyzing critical paths or planning speed-sensitive designs.

Three timing model sections are described:

- Functional element diagram – basic architectural schematic illustrating pins and connections
- Timing parameters – definitions of *Virtex-5 FPGA Data Sheet* timing parameters
- Timing Diagram - illustrates functional element timing parameters relative to each other

Use the models in this chapter in conjunction with both the Xilinx Timing Analyzer software (TRCE) and the section on switching characteristics in the *Virtex-5 FPGA Data Sheet*. All pin names, parameter names, and paths are consistent with the post-route timing and pre-route static timing reports. Most of the timing parameters found in the section on switching characteristics are described in this chapter.

All timing parameters reported in the *Virtex-5 FPGA Data Sheet* are associated with slices and CLBs. The following sections correspond to specific switching characteristics sections in the *Virtex-5 FPGA Data Sheet*:

- "General Slice Timing Model and Parameters" (CLB Switching Characteristics)
- "Slice Distributed RAM Timing Model and Parameters (Available in SLICEM only)" (CLB Distributed RAM Switching Characteristics)
- "Slice SRL Timing Model and Parameters (Available in SLICEM only)" (CLB SRL Switching Characteristics)
- "Slice Carry-Chain Timing Model and Parameters" (CLB Application Switching Characteristics)

General Slice Timing Model and Parameters

A simplified Virtex-5 FPGA slice is shown in Figure 5-25. Some elements of the slice are omitted for clarity. Only the elements relevant to the timing paths described in this section are shown.

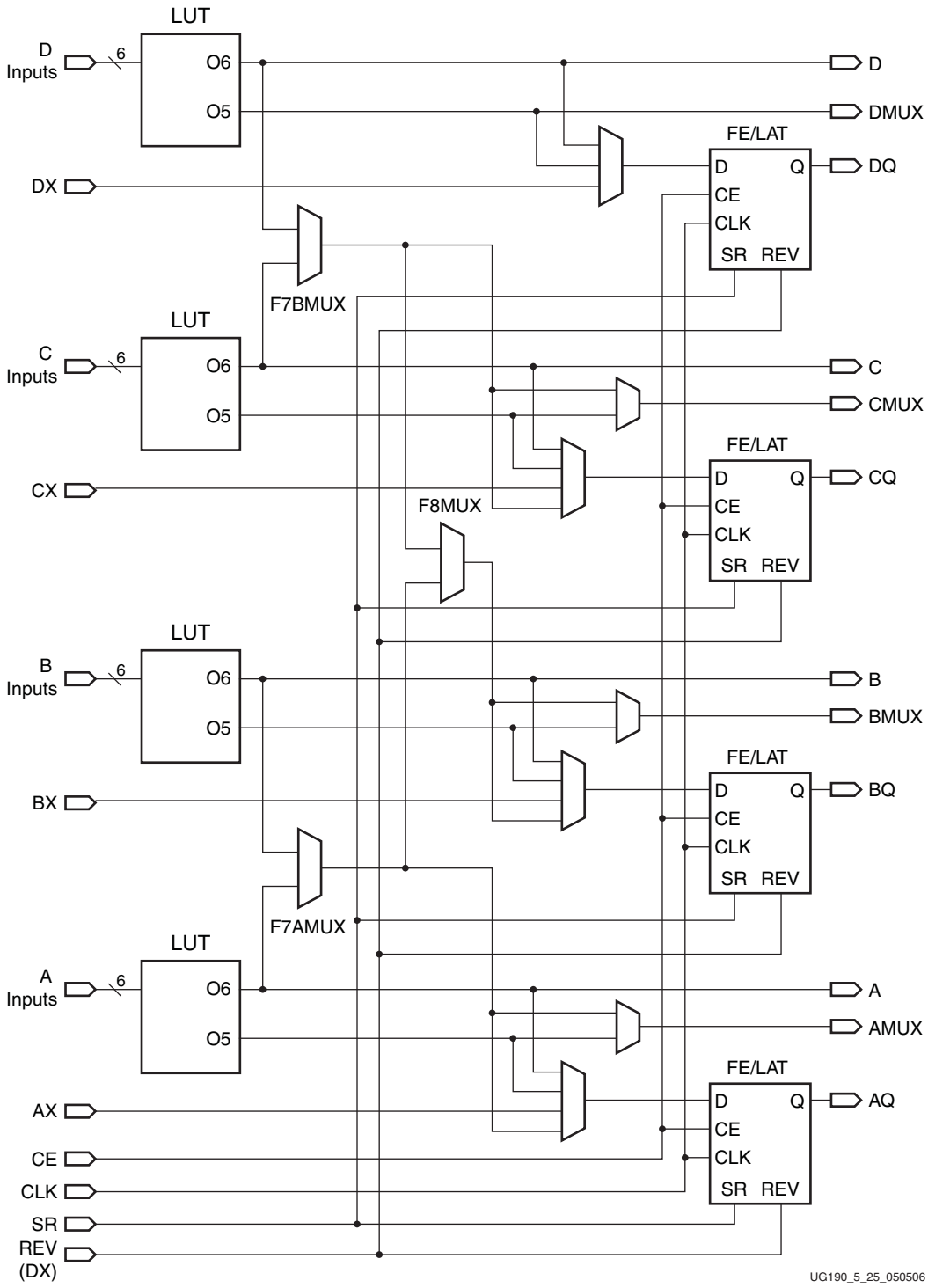


Figure 5-25: **Simplified Virtex-5 FPGA Slice**

Timing Parameters

Table 5-7 shows the general slice timing parameters for a majority of the paths in Figure 5-25.

Table 5-7: **General Slice Timing Parameters**

Parameter	Function	Description
Combinatorial Delays		
T_{ILO}[1]	A/B/C/D inputs to A/B/C/D outputs	Propagation delay from the A/B/C/D inputs of the slice, through the look-up tables (LUTs), to the A/B/C/D outputs of the slice (six-input function).
T_{ILO_2}	A/B/C/D inputs to AMUX/CMUX outputs	Propagation delay from the A/B/C/D inputs of the slice, through the LUTs and F7AMUX/F7BMUX to the AMUX/CMUX outputs (seven-input function).
T_{ILO_3}	A/B/C/D inputs to BMUX output	Propagation delay from the A/B/C/D inputs of the slice, through the LUTs, F7AMUX/F7BMUX, and F8MUX to the BMUX output (eight-input function).
Sequential Delays		
T_{CKO}	FF Clock (CLK) to AQ/BQ/CQ/DQ outputs	Time after the clock that data is stable at the AQ/BQ/CQ/DQ outputs of the slice sequential elements (configured as a flip-flop).
T_{CKLO}	Latch Clock (CLK) to AQ/BQ/CQ/DQ outputs	Time after the clock that data is stable at the XQ/YQ outputs of the slice sequential elements (configured as a latch).
Setup and Hold Times for Slice Sequential Elements[2]		
T_{DICK}/T_{CKDI}	AX/BX/CX/DX inputs	Time before/after the CLK that data from the AX/BX/CX/DX inputs of the slice must be stable at the D input of the slice sequential elements (configured as a flip-flop).
T_{CECK}/T_{CKCE}	CE input	Time before/after the CLK that the CE input of the slice must be stable at the CE input of the slice sequential elements (configured as a flip-flop).
T_{SRCK}/T_{CKSR}	SR/BY input	Time before/after the CLK that the SR (Set/Reset) and the BY (Rev) inputs of the slice must be stable at the SR/Rev inputs of the slice sequential elements (configured as a flip-flop).
Set/Reset		
T_{RPW}		Minimum Pulse Width for the SR (Set/Reset) and BY (Rev) pins.
T_{RQ}		Propagation delay for an asynchronous Set/Reset of the slice sequential elements. From the SR/BY inputs to the AQ/BQ/CQ/DQ outputs.

Table 5-7: **General Slice Timing Parameters *(Continued)***

Parameter	Function	Description
F_{TOG}		Toggle Frequency – Maximum frequency that a CLB flip-flop can be clocked: 1 / (T_{CH} + T_{CL}).

Notes:

1. This parameter includes a LUT configured as two five-input functions.
2. T_{XXCK} = Setup Time (before clock edge), and T_{CKXX} = Hold Time (after clock edge).

Timing Characteristics

Figure 5-26 illustrates the general timing characteristics of a Virtex-5 FPGA slice.

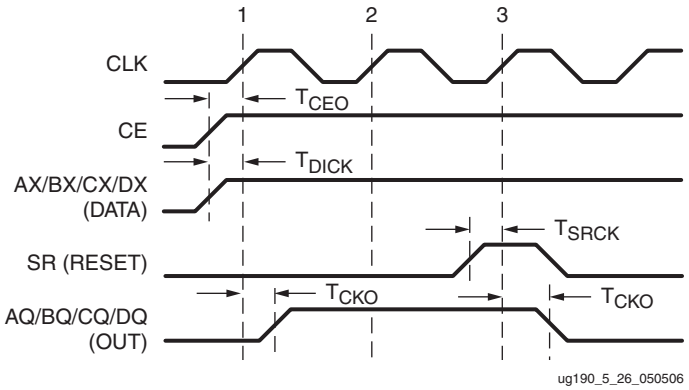


Figure 5-26: **General Slice Timing Characteristics**

- At time T_{CEO} before clock event (1), the clock-enable signal becomes valid-high at the CE input of the slice register.
- At time T_{DICK} before clock event (1), data from either AX, BX, CX, or DX inputs become valid-high at the D input of the slice register and is reflected on either the AQ, BQ, CQ, or DQ pin at time T_{CKO} after clock event (1).
- At time T_{SRCK} before clock event (3), the SR signal (configured as synchronous reset) becomes valid-high, resetting the slice register. This is reflected on the AQ, BQ, CQ, or DQ pin at time T_{CKO} after clock event (3).

Slice Distributed RAM Timing Model and Parameters (Available in SLICEM only)

Figure 5-27 illustrates the details of distributed RAM implemented in a Virtex-5 FPGA slice. Some elements of the slice are omitted for clarity. Only the elements relevant to the timing paths described in this section are shown.

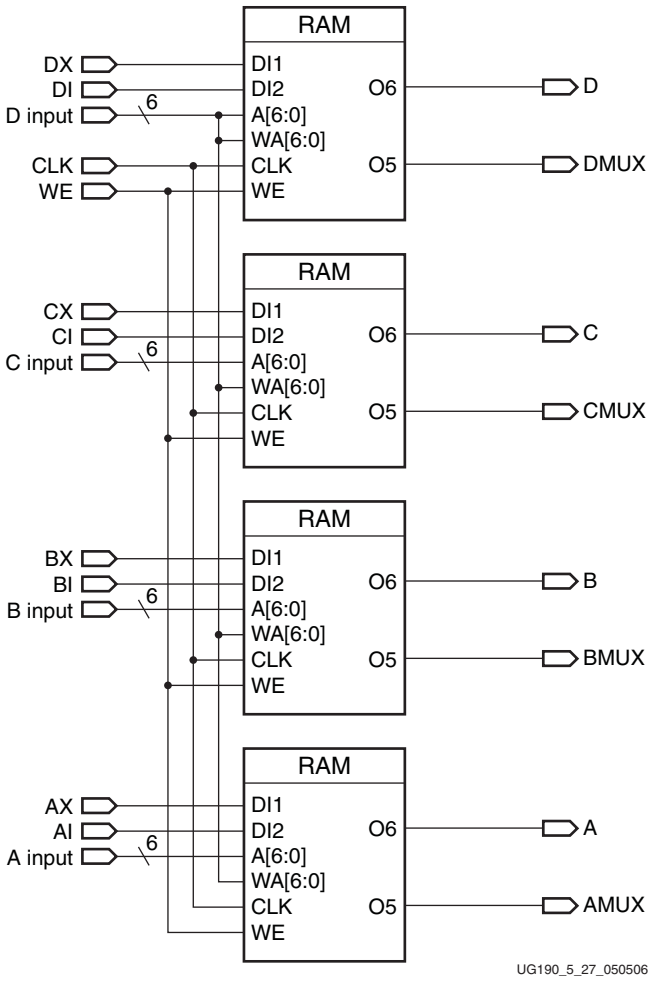


Figure 5-27: **Simplified Virtex-5 FPGA SLICEM Distributed RAM**

Distributed RAM Timing Parameters

Table 5-8 shows the timing parameters for the distributed RAM in SLICEM for a majority of the paths in Figure 5-27.

Table 5-8: **Distributed RAM Timing Parameters**

Parameter	Function	Description
Sequential Delays for a Slice LUT Configured as RAM (Distributed RAM)		
T_{SHCKO}(1)	CLK to A/B/C/D outputs	Time after the CLK of a write operation that the data written to the distributed RAM is stable on the A/B/C/D output of the slice.
Setup and Hold Times for a Slice LUT Configured as RAM (Distributed RAM)(2)		
T_{DS}/T_{DH}(3)	AX/BX/CX/DX configured as data input (DI1)	Time before/after the clock that data must be stable at the AX/BX/CX/DX input of the slice.
T_{ACK}/T_{CKA}	A/B/C/D address inputs	Time before/after the clock that address signals must be stable at the A/B/C/D inputs of the slice LUT (configured as RAM).
T_{WS}/T_{WH}	WE input	Time before/after the clock that the write enable signal must be stable at the WE input of the slice LUT (configured as RAM).
Clock CLK		
T_{WPH}		Minimum Pulse Width, High
T_{WPL}		Minimum Pulse Width, Low
T_{WC}		Minimum clock period to meet address write cycle time.

Notes:

1. This parameters includes a LUT configured as a two-bit distributed RAM.
2. T_{XXCK} = Setup Time (before clock edge), and T_{CKXX} = Hold Time (after clock edge).
3. Parameter includes AI/BI/CI/DI configured as a data input (DI2).

Distributed RAM Timing Characteristics

The timing characteristics of a 16-bit distributed RAM implemented in a Virtex-5 FPGA slice (LUT configured as RAM) are shown in Figure 5-28.

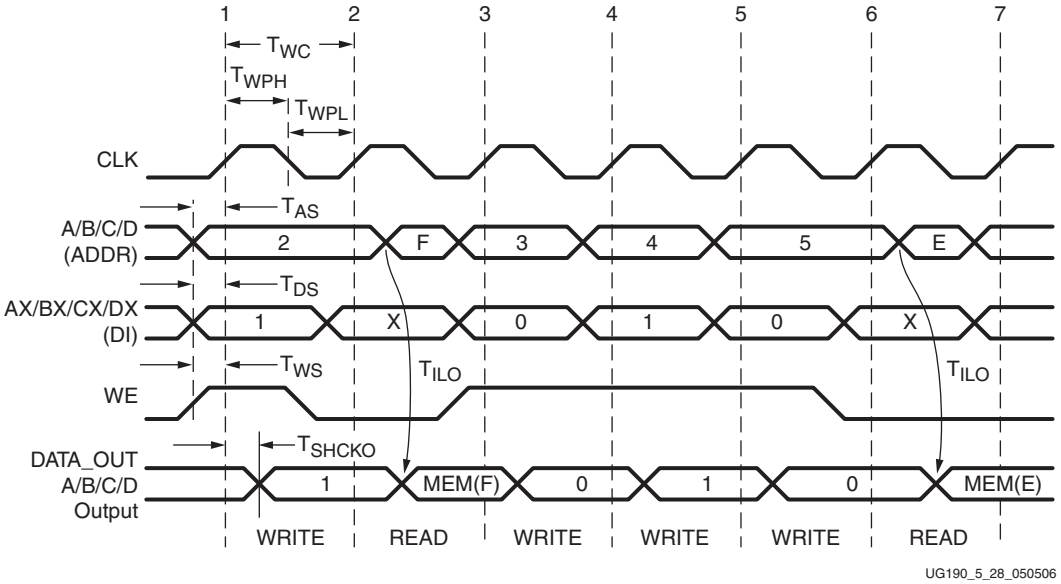


Figure 5-28: **Slice Distributed RAM Timing Characteristics**

Clock Event 1: Write Operation

During a Write operation, the contents of the memory at the address on the ADDR inputs are changed. The data written to this memory location is reflected on the A/B/C/D outputs synchronously.

- At time T_{WS} before clock event 1, the write-enable signal (WE) becomes valid-high, enabling the RAM for a Write operation.
- At time T_{AS} before clock event 1, the address (2) becomes valid at the A/B/C/D inputs of the RAM.
- At time T_{DS} before clock event 1, the DATA becomes valid (1) at the DI input of the RAM and is reflected on the A/B/C/D output at time T_{SHCKO} after clock event 1.

This is also applicable to the AMUX, BMUX, CMUX, DMUX, and COUT outputs at time T_{SHCKO} and T_{WOSCO} after clock event 1.

Clock Event 2: Read Operation

All Read operations are asynchronous in distributed RAM. As long as WE is Low, the address bus can be asserted at any time. The contents of the RAM on the address bus are reflected on the A/B/C/D outputs after a delay of length T_{ILO} (propagation delay through a LUT). The address (F) is asserted *after* clock event 2, and the contents of the RAM at address (F) are reflected at the output after a delay of length T_{ILO}.

Slice SRL Timing Model and Parameters (Available in SLICEM only)

Figure 5-29 illustrates shift register implementation in a Virtex-5 FPGA slice. Some elements of the slice have been omitted for clarity. Only the elements relevant to the timing paths described in this section are shown.

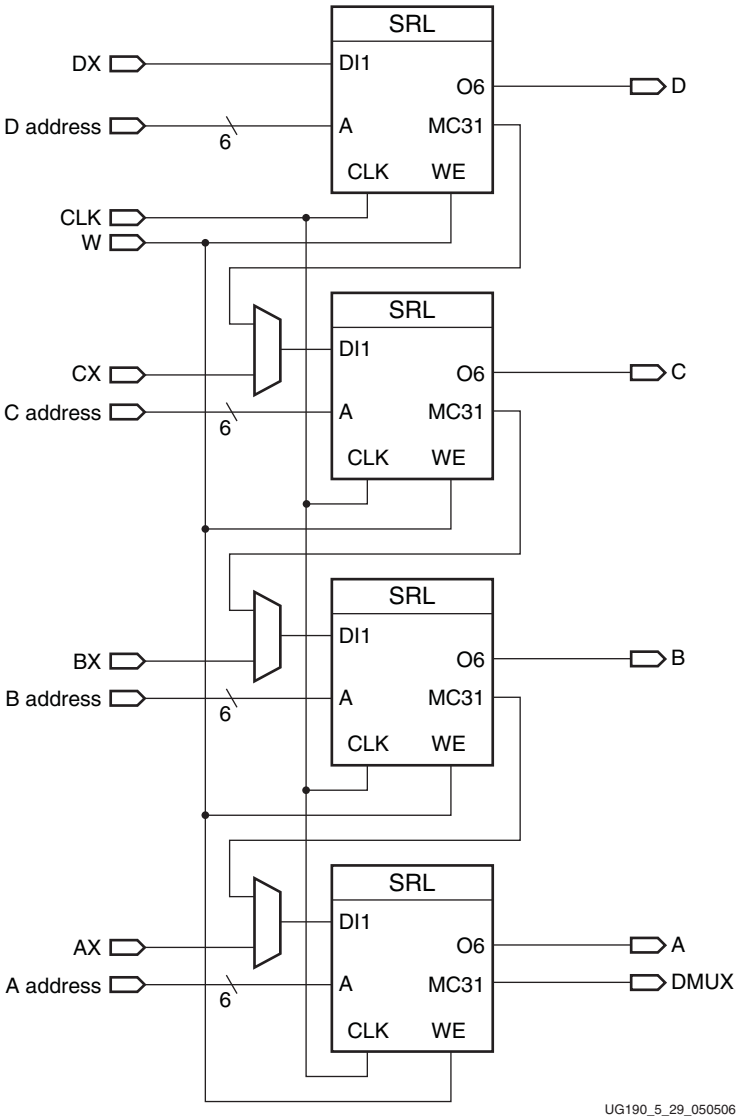


Figure 5-29: **Simplified Virtex-5 FPGA Slice SRL**

Slice SRL Timing Parameters

Table 5-9 shows the SLICEM SRL timing parameters for a majority of the paths in Figure 5-29.

Table 5-9: **Slice SRL Timing Parameters**

Parameter	Function	Description
Sequential Delays for a Slice LUT Configured as an SRL		
T_{REG}(1)	CLK to A/B/C/D outputs	Time after the CLK of a write operation that the data written to the SRL is stable on the A/B/C/D outputs of the slice.
T_{REG_MUX}(1)	CLK to AMUX - DMUX output	Time after the CLK of a write operation that the data written to the SRL is stable on the DMUX output of the slice.
T_{REG_M31}	CLK to DMUX output via MC31 output	Time after the CLK of a write operation that the data written to the SRL is stable on the DMUX output via MC31 output.
Setup and Hold Times for a Slice LUT Configured SRL(2)		
T_{WS}/T_{WH}	CE input (WE)	Time before/after the clock that the write enable signal must be stable at the WE input of the slice LUT (configured as an SRL).
T_{DS}/T_{DH}(3)	AX/BX/CX/DX configured as data input (DI)	Time before the clock that the data must be stable at the AX/BX/CX/DX input of the slice (configured as an SRL).

Notes:

1. This parameter includes a LUT configured as a two-bit shift register.
2. T_{XXCK} = Setup Time (before clock edge), and T_{CKXX} = Hold Time (after clock edge).
3. Parameter includes AI/BI/CI/DI configured as a data input (DI2) or two bits with a common shift.

Slice SRL Timing Characteristics

Figure 5-30 illustrates the timing characteristics of a 16-bit shift register implemented in a Virtex-5 FPGA slice (a LUT configured as an SRL).

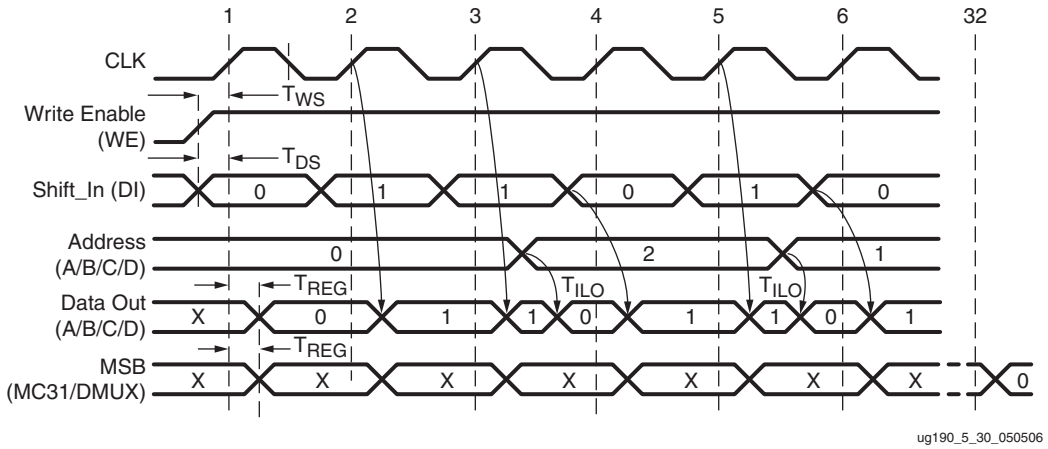


Figure 5-30: **Slice SRL Timing Characteristics**

Clock Event 1: Shift In

During a write (Shift In) operation, the single-bit content of the register at the address on the A/B/C/D inputs is changed, as data is shifted through the SRL. The data written to this register is reflected on the A/B/C/D outputs synchronously, if the address is unchanged during the clock event. If the A/B/C/D inputs are changed during a clock event, the value of the data at the addressable output (A/B/C/D outputs) is invalid.

- At time T_{WS} before clock event 1, the write-enable signal (WE) becomes valid-High, enabling the SRL for the Write operation that follows.
- At time T_{DS} before clock event 1 the data becomes valid (0) at the DI input of the SRL and is reflected on the A/B/C/D output after a delay of length T_{REG} after clock event 1. Since the address 0 is specified at clock event 1, the data on the DI input is reflected at A/B/C/D output, because it is written to register 0.

Clock Event 2: Shift In

- At time T_{DS} before clock event 2, the data becomes valid (1) at the DI input of the SRL and is reflected on the A/B/C/D output after a delay of length T_{REG} after clock event 2. Since the address 0 is still specified at clock event 2, the data on the DI input is reflected at the D output, because it is written to register 0.

Clock Event 3: Shift In/Addressable (Asynchronous) READ

All Read operations are asynchronous to the CLK signal. If the address is changed (between clock events), the contents of the register at that address are reflected at the addressable output (A/B/C/D outputs) after a delay of length T_{ILO} (propagation delay through a LUT).

- At time T_{DS} before clock event 3, the data becomes valid (1) at the DI input of the SRL and is reflected on the A/B/C/D output T_{REG} time after clock event 3.
- The address is changed (from 0 to 2). The value stored in register 2 at this time is a 0 (in this example, this was the first data shifted in), and it is reflected on the A/B/C/D output after a delay of length T_{ILO}.

Clock Event 32: MSB (Most Significant Bit) Changes

At time T_{REG} after clock event 32, the first bit shifted into the SRL becomes valid (logical 0 in this case) on the DMUX output of the slice via the MC31 output of LUT A (SRL). This is also applicable to the AMUX, BMUX, CMUX, DMUX, and COUT outputs at time T_{REG} and T_{WOSCO} after clock event 1.

Slice Carry-Chain Timing Model and Parameters

Figure 5-24, page 197 illustrates a carry chain in a Virtex-5 FPGA slice. Some elements of the slice have been omitted for clarity. Only the elements relevant to the timing paths described in this section are shown.

Slice Carry-Chain Timing Parameters

Table 5-10 shows the slice carry-chain timing parameters for a majority of the paths in Figure 5-24, page 197.

Table 5-10: **Slice Carry-Chain Timing Parameters**

Parameter	Function	Description
Sequential Delays for Slice LUT Configured as Carry Chain		
$T_{AXCY}/T_{BXCY}/T_{CXCY}/T_{DXCY}$	AX/BX/CX/DX input to COUT output	Propagation delay from the AX/BX/CX/DX inputs of the slice to the COUT output of the slice.
T_{BYP}	CIN input to COUT output	Propagation delay from the CIN input of the slice to the COUT output of the slice.
$T_{OPCYA}/T_{OPCYB}/T_{OPCYC}/T_{OPCYD}$	A/B/C/D input to COUT output	Propagation delay from the A/B/C/D inputs of the slice to the COUT output of the slice.
$T_{CINA}/T_{CINB}/T_{CINC}/T_{CIND}$	A/B/C/D input to AMUX/BMUX/CMUX/DMUX output	Propagation delay from the A/B/C/D inputs of the slice to AMUX/BMUX/CMUX/DMUX output of the slice using XOR (sum).
Setup and Hold Times for a Slice LUT Configured as a Carry Chain(1)		
T_{CINCK}/T_{CKCIN}	CIN Data inputs	Time before the CLK that data from the CIN input of the slice must be stable at the D input of the slice sequential elements (configured as a flip-flop).

Notes:

1. T_{XXCK} = Setup Time (before clock edge), and T_{CKXX} = Hold Time (after clock edge).

Slice Carry-Chain Timing Characteristics

Figure 5-31 illustrates the timing characteristics of a slice carry chain implemented in a Virtex-5 FPGA slice.

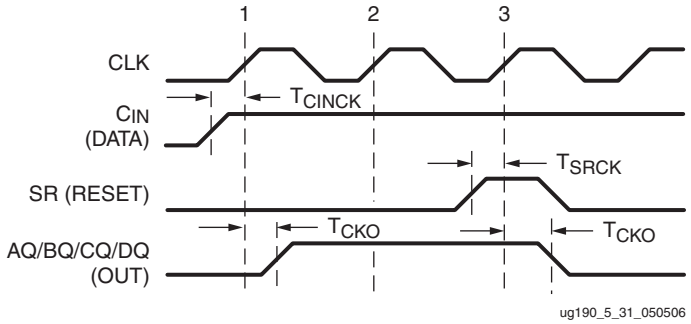


Figure 5-31: **Slice Carry-Chain Timing Characteristics**

- At time T_{CINCK} before clock event 1, data from CIN input becomes valid-high at the D input of the slice register. This is reflected on any of the AQ/BQ/CQ/DQ pins at time T_{CKO} after clock event 1.
- At time T_{SRCK} before clock event 3, the SR signal (configured as synchronous reset) becomes valid-high, resetting the slice register. This is reflected on any of the AQ/BQ/CQ/DQ pins at time T_{CKO} after clock event 3.

CLB Primitives

More information on the CLB primitives are available in the software libraries guide.

Distributed RAM Primitives

Seven primitives are available; from 32 x 2 bits to 256 x 1 bit. Three primitives are single-port RAM, two primitives are dual-port RAM, and two primitives are quad-port RAM, as shown in Table 5-11.

Table 5-11: **Single-Port, Dual-Port, and Quad-Port Distributed RAM**

Primitive	RAM Size	Type	Address Inputs
RAM32X1S	32-bit	Single-port	A[4:0] (read/write)
RAM32X1D	32-bit	Dual-port	A[4:0] (read/write) DPRA[4:0] (read)
RAM32M	32-bit	Quad-port	ADDRA[4:0] (read) ADDRB[4:0] (read) ADDRC[4:0] (read) ADDRD[4:0] (read/write)
RAM64X1S	64-bit	Single-port	A[5:0] (read/write)
RAM64X1D	64-bit	Dual-port	A[5:0] (read/write) DPRA[5:0] (read)
RAM64M	64-bit	Quad-port	ADDRA[5:0] (read) ADDRB[5:0] (read) ADDRC[5:0] (read) ADDRD[5:0] (read/write)
RAM128X1S	128-bit	Single-port	A[6:0] (read/write)
RAM128X1D	128-bit	Dual-port	A[6:0], (read/write) DPRA[6:0] (read)
RAM256X1S	256-bit	Single-port	A[7:0] (read/write)

The input and output data are 1-bit wide (with the exception of the 32-bit RAM).

Figure 5-32 shows generic single-port, dual-port, and quad-port distributed RAM primitives. The A, ADDR, and DPRA signals are address buses.

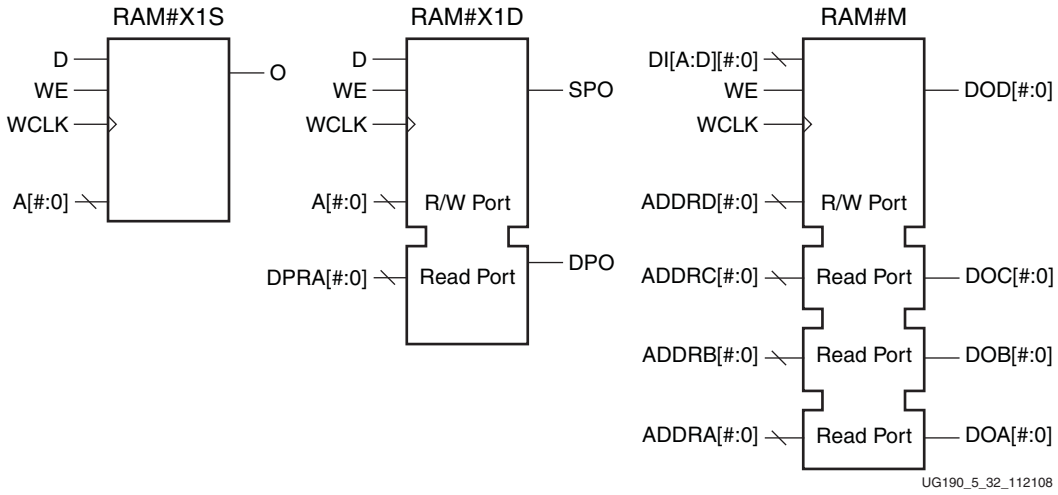


Figure 5-32: **Single-Port, Dual-Port, and Quad-Port Distributed RAM Primitives**

Instantiating several distributed RAM primitives can be used to implement wide memory blocks.

Port Signals

Each distributed RAM port operates independently of the other while reading the same set of memory cells.

Clock – WCLK

The clock is used for the synchronous write. The data and the address input pins have setup times referenced to the WCLK pin.

Enable – WE/WED

The enable pin affects the write functionality of the port. An active write enable prevents any writing to memory cells. An active write enable causes the clock edge to write the data input signal to the memory location pointed to by the address inputs.

Address – A[#:0], DPRA[#:0], and ADDRA[#:0] – ADDRD[#:0]

The address inputs A[#:0] (for single-port and dual-port), DPRA[#:0] (for dual-port), and ADDRA[#:0] – ADDRD[#:0] (for quad-port) select the memory cells for read or write. The width of the port determines the required address inputs. Some of the address inputs are not buses in VHDL or Verilog instantiations. Table 5-11 summarizes the function of each address pins.

Data In – D, DID[#:0]

The data input D (for single-port and dual-port) and DID[#:0] (for quad-port) provide the new data value to be written into the RAM.

Data Out – O, SPO, DPO and DOA[#:0] – DOD[#:0]

The data out O (single-port or SPO), DPO (dual-port), and DOA[#:0] – DOD[#:0] (quad-port) reflects the contents of the memory cells referenced by the address inputs. Following an active write clock edge, the data out (O, SPO, or DOD[#:0]) reflects the newly written data.

Inverting Clock Pins

The clock pin (CLK) has an individual inversion option. The clock signal can be active at the negative edge of the clock or the positive edge for the clock without requiring other logic resources. The default is at the positive clock edge

Global Set/Reset – GSR

The global set/reset (GSR) signal does not affect distributed RAM modules.

Shift Registers (SRLs) Primitive

One primitive is available for the 32-bit shift register (SRLC32E). Figure 5-33 shows the 32-bit shift register primitive.

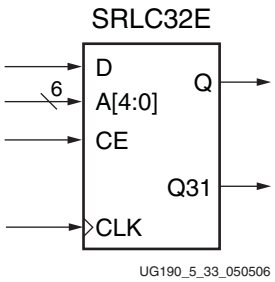


Figure 5-33: **32-bit Shift Register**

Instantiating several 32-bit shift register with dedicated multiplexers (F7AMUX, F7BMUX, and F8MUX) allows a cascadable shift register chain of up to 128-bit in a slice. Figure 5-18 through Figure 5-20 in the *"Shift Registers (Available in SLICEM only)"* section of this document illustrate the various implementation of cascadable shift registers greater than 32 bits.

Port Signals

Clock – CLK

Either the rising edge or the falling edge of the clock is used for the synchronous shift operation. The data and clock enable input pins have setup times referenced to the chosen edge of CLK.

Data In – D

The data input provides new data (one bit) to be shifted into the shift register.

Clock Enable - CE

The clock enable pin affects shift functionality. An inactive clock enable pin does not shift data into the shift register and does not write new data. Activating the clock enable allows the data in (D) to be written to the first location and all data to be shifted by one location. When available, new data appears on output pins (Q) and the cascadable output pin (Q31).

Address – A[4:0]

The address input selects the bit (range 0 to 31) to be read. The nth bit is available on the output pin (Q). Address inputs have no effect on the cascadable output pin (Q31). It is always the last bit of the shift register (bit 31).

Data Out – Q

The data output Q provides the data value (1 bit) selected by the address inputs.

Data Out – Q31 (optional)

The data output Q31 provides the last bit value of the 32-bit shift register. New data becomes available after each shift-in operation.

Inverting Clock Pins

The clock pin (CLK) has an individual inversion option. The clock signal can be active at the negative or positive edge of the clock without requiring other logic resources. The default is positive clock edge.

Global Set/Reset – GSR

The global set/reset (GSR) signal does not affect the shift registers.

Other Shift Register Applications

Synchronous Shift Registers

The shift-register primitive does not use the register available in the same slice. To implement a fully synchronous read and write shift register, output pin Q must be connected to a flip-flop. Both the shift register and the flip-flop share the same clock, as shown in Figure 5-34.

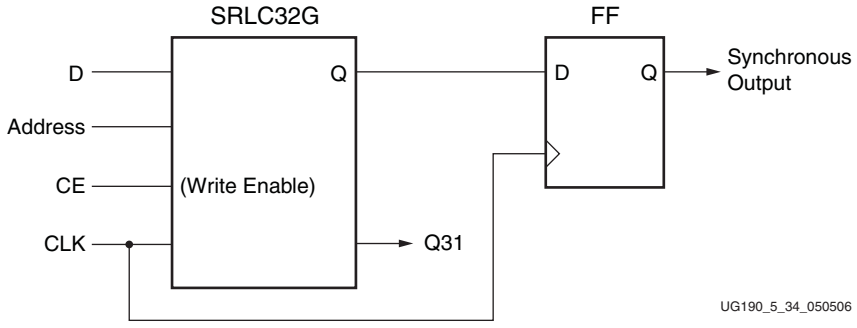


Figure 5-34: **Synchronous Shift Register**

This configuration provides a better timing solution and simplifies the design. Because the flip-flop must be considered to be the last register in the shift-register chain, the static or dynamic address should point to the desired length minus one. If needed, the cascadable output can also be registered in a flip-flop.

Static-Length Shift Registers

The cascadable 32-bit shift register implements any static length mode shift register without the dedicated multiplexers (F7AMUX, F7BMUX, and F8MUX). Figure 5-35 illustrates a 72-bit shift register. Only the last SRLC32E primitive needs to have its address inputs tied to `0b00111`. Alternatively, shift register length can be limited to 71 bits (address tied to `0b00110`) and a flip-flop can be used as the last register. (In an SRLC32E primitive, the shift register length is the address input + 1).

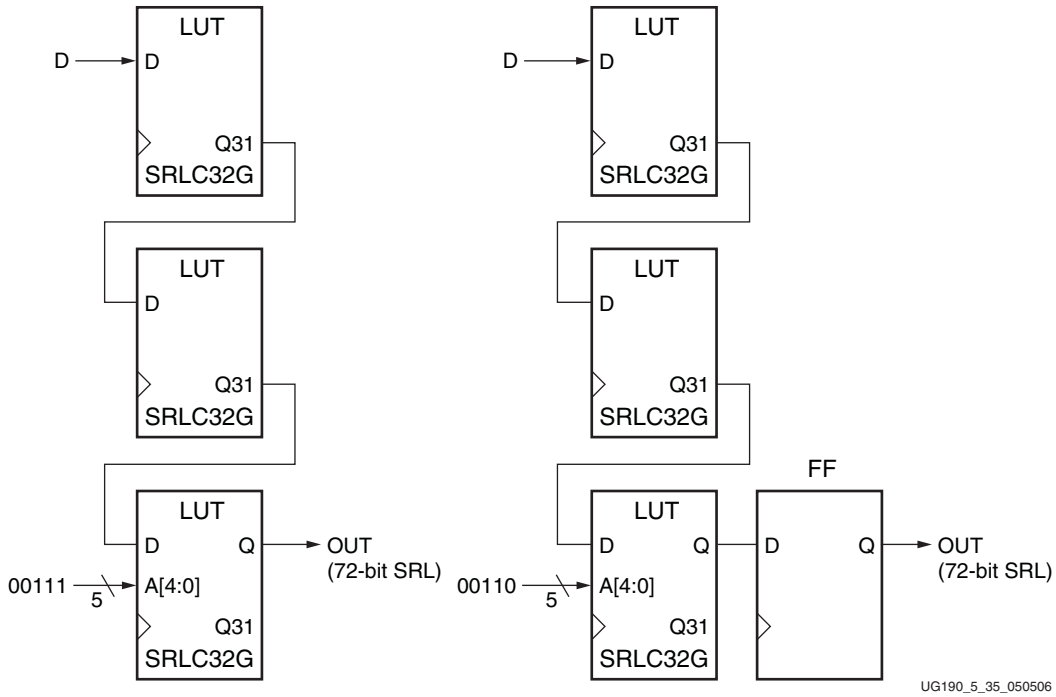


Figure 5-35: **Example Static-Length Shift Register**

Multiplexer Primitives

Two primitives (MUXF7 and MUXF8) are available for access to the dedicated F7AMUX, F7BMUX and F8MUX in each slice. Combined with LUTs, these multiplexer primitives are also used to build larger width multiplexers (from 8:1 to 16:1). The "Designing Large Multiplexers" section provides more information on building larger multiplexers.

Port Signals

Data In – I0, I1

The data input provides the data to be selected by the select signal (S).

Control In – S

The select input signal determines the data input signal to be connected to the output O. Logic 0 selects the I0 input, while logic 1 selects the I1 input.

Data Out – O

The data output O provides the data value (one bit) selected by the control inputs.

Carry Chain Primitive

The CARRY4 primitive represents the fast carry logic for a slice in the Virtex-5 architecture. This primitive works in conjunction with LUTs in order to build adders and multipliers. This primitive is generally inferred by synthesis tools from standard RTL code. The synthesis tool can identify the arithmetic and/or logic functionality that best maps to this

logic in terms of performance and area. It also automatically uses and connects this function properly. Figure 5-24, page 197 illustrates the CARRY4 block diagram.

Port Signals

Sum Outputs – O[3:0]

The sum outputs provide the final result of the addition/subtraction.

Carry Outputs – CO[3:0]

The carry outputs provide the carry out for each bit. A longer carry chain can be created if CO[3] is connected to CI input of another CARRY4 primitive.

Data Inputs – DI[3:0]

The data inputs are used as "generate" signals to the carry lookahead logic. The "generate" signals are sourced from LUT outputs.

Select Inputs – S[3:0]

The select inputs are used as "propagate" signals to the carry lookahead logic. The "propagate" signals are sourced from LUT outputs.

Carry Initialize – CYINIT

The carry initialize input is used to select the first bit in a carry chain. The value for this pin is either 0 (for add), 1 (for subtract), or AX input (for the dynamic first carry bit).

Carry In – CI

The carry in input is used to cascade slices to form longer carry chain. To create a longer carry chain, the CO[3] output of another CARRY4 is simply connected to this pin.

Chapter 6

SelectIO Resources

I/O Tile Overview

Input/output characteristics and logic resources are covered in three consecutive chapters.

Chapter 6, "SelectIO Resources" describes the electrical behavior of the output drivers and input receivers, and gives detailed examples of many standard interfaces. Chapter 7, "SelectIO Logic Resources," describes the input and output data registers and their Double-Data-Rate (DDR) operation, and the programmable input delay (IDELAY). Chapter 8, "Advanced SelectIO Logic Resources," describes the data serializer/deserializer (SERDES).

An I/O tile contains two IOBs, two ILOGICs, two OLOGICs, and two IODELAYs. Figure 6-1 shows a Virtex-5 FPGA I/O tile.

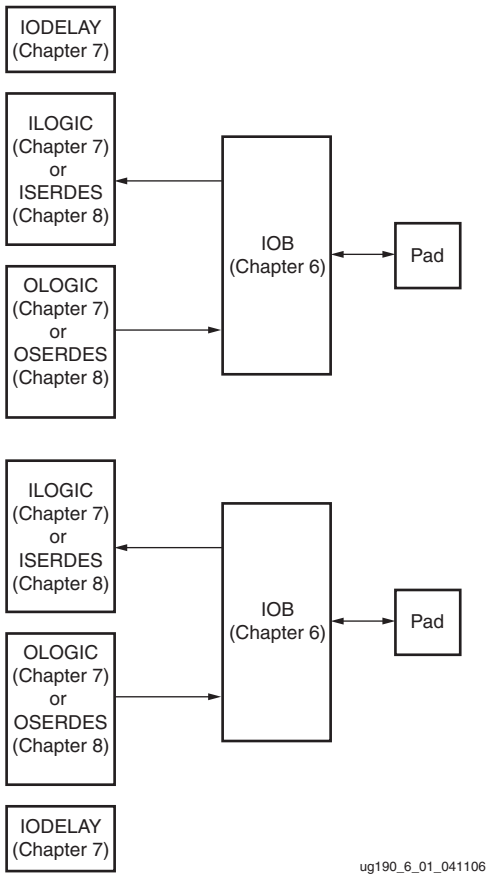


Figure 6-1: **Virtex-5 FPGA I/O Tile**

SelectIO Resources Introduction

All Virtex-5 FPGAs have configurable high-performance SelectIO™ drivers and receivers, supporting a wide variety of standard interfaces. The robust feature set includes programmable control of output strength and slew rate, and on-chip termination using Digitally Controlled Impedance (DCI).

Each IOB contains both input, output, and 3-state SelectIO drivers. These drivers can be configured to various I/O standards. Differential I/O uses the two IOBs grouped together in one tile.

- Single-ended I/O standards (LVCMOS, LVTTL, HSTL, SSTL, GTL, PCI)
- Differential I/O standards (LVDS, HT, LVPECL, BLVDS, Differential HSTL and SSTL)
- Differential and V_{REF} dependent inputs are powered by V_{CCAUX}

Each Virtex-5 FPGA I/O tile contains two IOBs, and also two ILOGIC blocks and two OLOGIC blocks, as described in Chapter 7, "SelectIO Logic Resources."

Figure 6-2 shows the basic IOB and its connections to the internal logic and the device Pad.

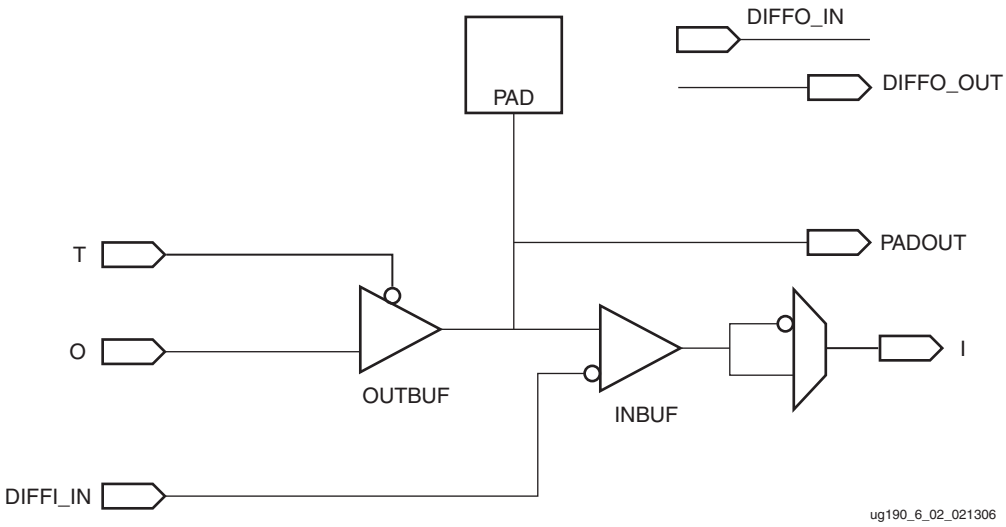


Figure 6-2: **Basic IOB Diagram**

Each IOB has a direct connection to an ILOGIC/OLOGIC pair containing the input and output logic resources for data and 3-state control for the IOB. Both ILOGIC and OLOGIC can be configured as ISERDES and OSERDES, respectively, as described in Chapter 8, "Advanced SelectIO Logic Resources."

SelectIO Resources General Guidelines

This section summarizes the general guidelines to be considered when designing with the SelectIO resources in Virtex-5 FPGAs.

Virtex-5 FPGA I/O Bank Rules

In Virtex-5 devices, with some exceptions in the center column, an I/O bank consists of 40 IOBs (20 CLBs high and a single clock region). There are always four half-sized banks (20 IOBs) and a single configuration bank in the center column. The number of banks depends upon the device size, and in larger devices, there are additional full-sized banks in the center column. In the *Virtex-5 Family Overview* the total number of I/O banks is listed by device type. The XC5VLX30 has 12 usable I/O banks and one configuration bank. Figure 6-3 is an example of a columnar floorplan showing the XC5VLX30 I/O banks.

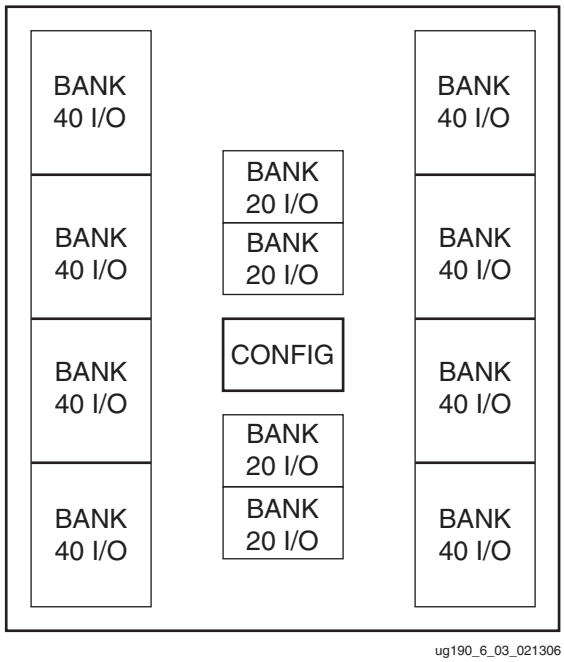


Figure 6-3: **Virtex-5 FPGA XC5VLX30 I/O Banks**

Reference Voltage (V_{REF}) Pins

Low-voltage, single-ended I/O standards with a differential amplifier input buffer require an input reference voltage (V_{REF}). V_{REF} is an external input into Virtex-5 devices. Within each I/O bank, one of every 20 I/O pins is automatically configured as a V_{REF} input, if using a single-ended I/O standard that requires a differential amplifier input buffer.

Output Drive Source Voltage (V_{CCO}) Pins

Many of the low-voltage I/O standards supported by Virtex-5 devices require a different output drive voltage (V_{CCO}). As a result, each device often supports multiple output drive source voltages.

Output buffers within a given V_{CCO} bank must share the same output drive source voltage. The following input buffers use the V_{CCO} voltage: LVTTL, LVCMOS, PCI, LVDCI and other DCI standards.

Virtex-5 FPGA Digitally Controlled Impedance (DCI)

Introduction

As FPGAs get bigger and system clock speeds get faster, PC board design and manufacturing becomes more difficult. With ever faster edge rates, maintaining signal integrity becomes a critical issue. PC board traces must be properly terminated to avoid reflections or ringing.

To terminate a trace, resistors are traditionally added to make the output and/or input match the impedance of the receiver or driver to the impedance of the trace. However, due to increased device I/Os, adding resistors close to the device pins increases the board area and component count, and can in some cases be physically impossible. To address these issues and to achieve better signal integrity, Xilinx developed the Digitally Controlled Impedance (DCI) technology.

DCI adjusts the output impedance or input termination to accurately match the characteristic impedance of the transmission line. DCI actively adjusts the impedance of the I/O to equal an external reference resistance. This compensates for changes in I/O impedance due to process variation. It also continuously adjusts the impedance of the I/O to compensate for variations of temperature and supply voltage fluctuations.

In the case of controlled impedance drivers, DCI controls the driver impedance to match two reference resistors, or optionally, to match half the value of these reference resistors. DCI eliminates the need for external series termination resistors.

DCI provides the parallel or series termination for transmitters or receivers. This eliminates the need for termination resistors on the board, reduces board routing difficulties and component count, and improves signal integrity by eliminating stub reflection. Stub reflection occurs when termination resistors are located too far from the end of the transmission line. With DCI, the termination resistors are as close as possible to the output driver or the input buffer, thus, eliminating stub reflections.

DCI Cascading

Previously, using DCI I/O standards in a bank required connecting external reference resistors to the VRN and VRP pins in that same bank. The VRN/VRP pins provide a reference voltage used by internal DCI circuitry to adjust the I/O output impedance to match the external reference resistors. As shown in Figure 6-4, a digital control bus is internally distributed throughout the bank to control the impedance of each I/O.

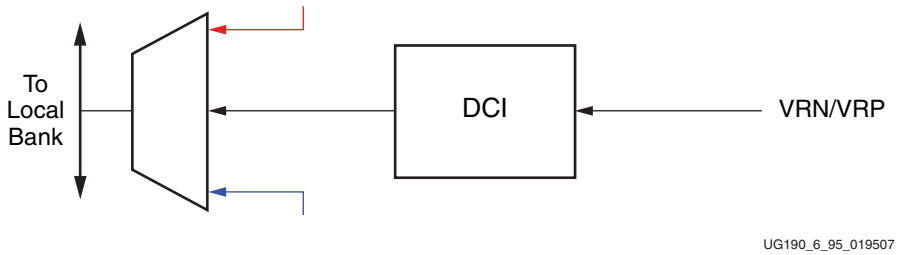


Figure 6-4: **DCI Use within a Bank**

The Virtex-5 FPGA banks using DCI I/O standards now have the option of deriving the DCI impedance values from another DCI bank. With DCI cascading, one bank (the master bank) must have its VRN/VRP pins connected to external reference resistors. Other banks in the same column (slave banks) can use DCI standards with the same impedance as the master bank, without connecting the VRN/VRP pins on these banks to external resistors. DCI impedance control in cascaded banks is received from the master bank.

When using DCI cascading, the DCI control circuitry in the master bank creates and routes DCI control to the cascaded banks in daisy-chain style. DCI control for a particular bank can come from the bank immediately above or below. Only the master bank's VRN/VRP pins are required when using DCI cascading.

Also, when using DCI cascading, only one set of VRN/VRP pins provides the DCI reference voltage for multiple banks. DCI cascading:

- Reduces overall power, since fewer voltage references are required
- Frees up VRN/VRP pins on slave banks for general customer use
- DCI in banks 1 and 2 is supported only through cascading. These two banks do not have VRN/VRP pins and therefore cannot be used as master or stand-alone DCI banks. Cascading is not possible through bank 0.

Similarly, due to the center column architecture, the half-size banks 1, 2, 3, and 4 are separated from all the other banks in the center column by the CMT tiles. It is not possible to cascade across the CMT tiles. This affects the larger devices that have more than four user I/O center column banks (plus bank 0). For instance, bank 4 cannot be cascaded with bank 6, and bank 3 cannot be cascaded with bank 5. Bank 3 can only be cascaded with bank 1, and bank 4 can only be cascaded with bank 2.

Figure 6-5 shows DCI cascading support over multiple banks. Bank B is the master bank.

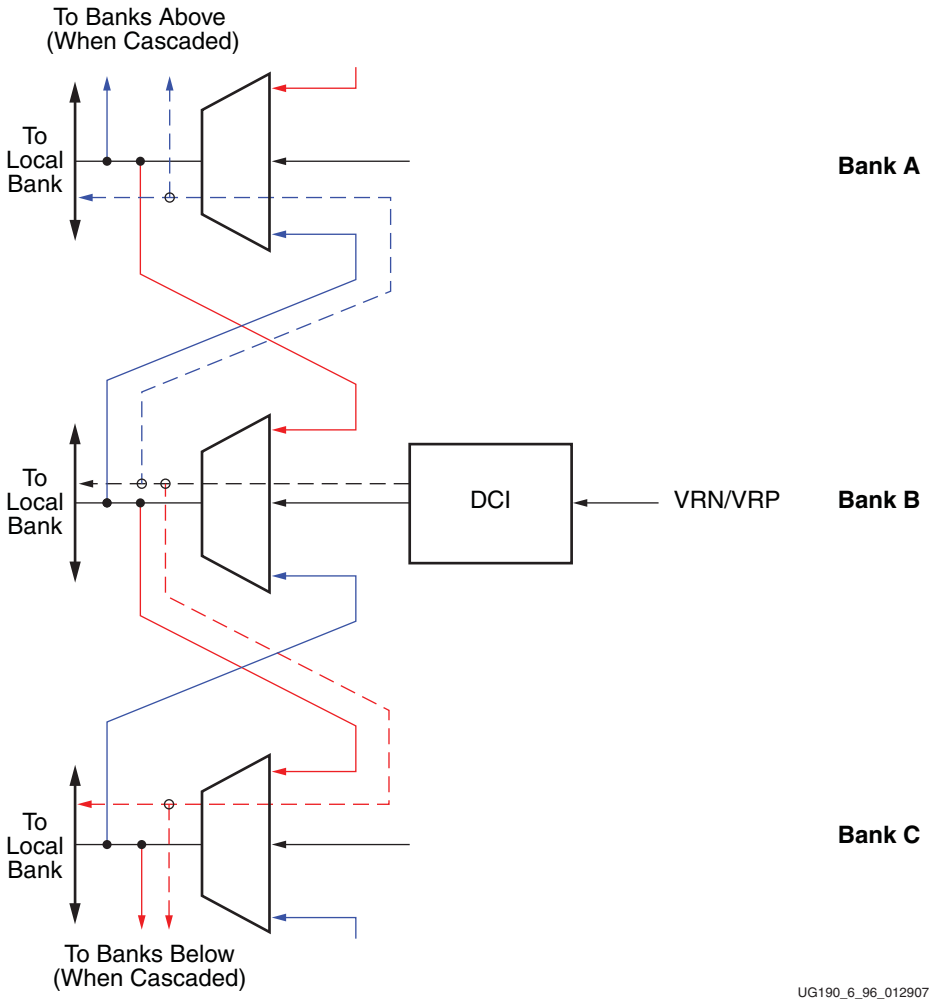


Figure 6-5: **DCI Cascading Supported Over Multiple Banks**

The guidelines when using DCI cascading are as follows:

- The master and slave banks must all reside on the same column (left, center, or right) on the device.
- Master and slave banks must have the same V_{CCO} and V_{REF} (if applicable) voltage.
- DCI I/O banking compatibility rules must be satisfied across all master and slave banks (for example, only one DCI I/O standard using single termination type is allowed across all master and slave banks). DCI I/O standard compatibility is not constrained to one bank when DCI cascading is implemented; it extends across all master and slave banks.

- DCI cascading must extend across consecutive banks in the same column. It is not possible to skip banks when using DCI cascading. For example, consider four banks in a column A, B, C, and D, from top to bottom. In this case, the following are valid possibilities for DCI cascading, assuming all other guidelines are met:
  - DCI cascading can extend to Bank A, Bank C, or both banks.
  - DCI cascading can also extend to Bank D, since Bank D is in the same column. However, DCI cascading must also extend to the intervening Bank C. If DCI I/O standards are implemented in Bank C, DCI I/O banking compatibility must be observed across all three banks (B, C, and D).
- DCI cascading can span the entire column as long as the above guidelines are met.
- Locate adjacent banks. Bank location information is best determined from partgen generated package files (`partgen -v XC5VLX50TFF1136`). The resulting package file with a .pkg extension contains XY I/O location information. The X designator indicates I/Os in the same column. The Y designator indicates the position of an I/O within a specific bank. The bank number is also shown. Consecutive Y locations across bank boundaries show adjacent banks. For example, the XC5VLXT in an FF1136 package shows bank 11 starting with I/O X0Y159 end ending with I/O location X0Y120. Bank 13 starts with I/O X0Y119 and ends with X0Y80. Bank 15 starts with X0Y199 and ends with X0Y160. This indicates that bank 13 is to the south of bank 11, and bank 15 is to the north. As the Y coordinates of these two banks are consecutive, these two banks are considered consecutive banks and can be DCI cascaded. It is possible to cascade through an unbonded bank.
- DCI cascade is enabled by using the DCI_CASCADE constraint described in the constraints guide.

Xilinx DCI

DCI uses two multi-purpose reference pins in each bank to control the impedance of the driver or the parallel termination value for all of the I/Os of that bank. The N reference pin (VRN) must be pulled up to V_{CCO} by a reference resistor, and the P reference pin (VRP) must be pulled down to ground by another reference resistor. The value of each reference resistor should be equal to the characteristic impedance of the PC board traces, or should be twice that value. See "Driver with Termination to $V_{CCO}/2$ (Split Termination)," page 226.

When a DCI I/O standard is used on a particular bank, the two multi-purpose reference pins cannot be used as regular I/Os. However, if DCI I/O standards are not used in the bank, these pins are available as regular I/O pins. The *Virtex-5 Family Packaging Specifications* gives detailed pin descriptions.

DCI adjusts the impedance of the I/O by selectively turning transistors in the I/Os on or off. The impedance is adjusted to match the external reference resistors. The impedance adjustment process has two phases. The first phase compensates for process variations by controlling the larger transistors in the I/Os. It occurs during the device startup sequence. The second phase maintains the impedance in response to temperature and supply voltage changes by controlling the smaller transistors in the I/Os. It begins immediately after the first phase and continues indefinitely, even while the device is operating. By default, the DONE pin does not go High until the first phase of the impedance adjustment process is complete.

The coarse impedance calibration during the first phase of impedance adjustment can be invoked after configuration by instantiating the DCIRESET primitive. By toggling the RST input to the DCIRESET primitive while the device is operating, the DCI state machine is

reset and both phases of impedance adjustment proceed in succession. All I/Os using DCI will be unavailable until the LOCKED output from the DCIRESET block is asserted.

This functionality is useful in applications where the temperature and/or supply voltage changes significantly from device power-up to the nominal operating condition. Once at the nominal operating temperature and voltage, performing the first phase of impedance adjustment allows optimal headroom for the second phase of impedance adjustment.

For controlled impedance output drivers, the impedance can be adjusted either to match the reference resistors or half the resistance of the reference resistors. For on-chip termination, the termination is always adjusted to match the reference resistors.

DCI can configure output drivers to be the following types:

1. Controlled Impedance Driver (Source Termination)
2. Controlled Impedance Driver with Half Impedance (Source Termination)

It can also configure inputs to have the following types of on-chip terminations:

1. Input termination to V_{CCO} (Single Termination)
2. Input termination to $V_{CCO}/2$ (Split Termination, Thevenin equivalent)

For bidirectional operation, both ends of the line can be DCI-terminated regardless of direction:

1. Driver with termination to V_{CCO} (Single Termination)
2. Driver with termination to $V_{CCO}/2$ (Split Termination, Thevenin equivalent)

Alternatively, bidirectional point-to-point lines can use controlled-impedance drivers (with 3-state buffers) on both ends.

Controlled Impedance Driver (Source Termination)

Some I/O standards, such as LVCMOS, must have a drive impedance matching the characteristic impedance of the driven line. DCI can provide controlled impedance output drivers to eliminate reflections without an external source termination. The impedance is set by the external reference resistors with resistance equal to the trace impedance.

The DCI I/O standards supporting the controlled impedance driver are: LVDCI_15, LVDCI_18, LVDCI_25, LVDCI_33, HSLVDCI_15, HSLVDCI_18, HSLVDCI_25, and HSLVDCI_33. Figure 6-6 illustrates a controlled impedance driver in a Virtex-5 device.

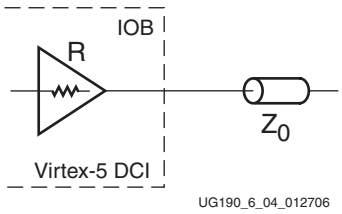


Figure 6-6: **Controlled Impedance Driver**

Controlled Impedance Driver with Half Impedance (Source Termination)

DCI also provides drivers with one half of the impedance of the reference resistors. This doubling of the reference resistor value reduces the static power consumption through these resistors by a factor of half. The DCI I/O standards supporting controlled impedance drivers with half-impedance are LVDCI_DV2_15, LVDCI_DV2_18, and LVDCI_DV2_25.

Figure 6-7 illustrates a controlled driver with half impedance inside a Virtex-5 device. The reference resistors R must be $2 \times Z_0$ in order to match the impedance of Z_0.

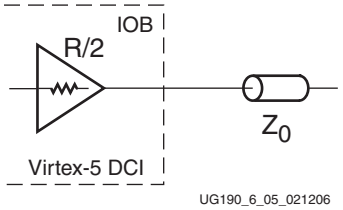


Figure 6-7: **Controlled Impedance Driver with Half Impedance**

Input Termination to V_{CCO} (Single Termination)

Some I/O standards require an input termination to V_{CCO} (see Figure 6-8).

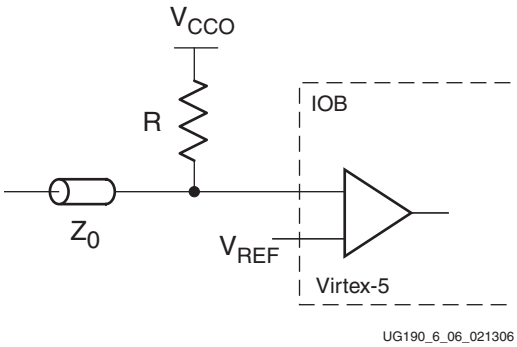


Figure 6-8: **Input Termination to V_{CCO} without DCI**

DCI can also provide input termination to V_{CCO} using single termination. The termination resistance is set by the reference resistors. Both GTL and HSTL standards are controlled by 50 Ω reference resistors. The DCI I/O standards supporting single termination are: GTL_DCI, GTLP_DCI, HSTL_III_DCI, HSTL_III_DCI_18, HSTL_IV_DCI, and HSTL_IV_DCI_18.

Figure 6-9 illustrates DCI single termination inside a Virtex-5 device.

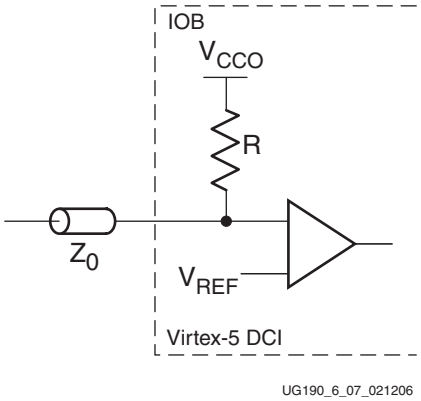


Figure 6-9: **Input Termination Using DCI Single Termination**

Input Termination to $V_{CCO}/2$ (Split Termination)

Some I/O standards (e.g., HSTL Class I and II) require an input termination voltage of $V_{CCO}/2$ (see Figure 6-10).

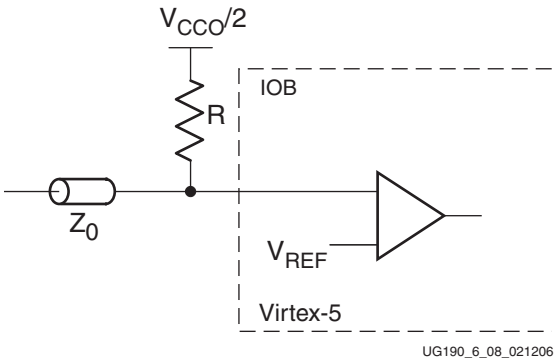


Figure 6-10: **Input Termination to $V_{CCO}/2$ without DCI**

This is equivalent to having a split termination composed of two resistors. One terminates to V_{CCO}, the other to ground. The resistor values are 2R. DCI provides termination to $V_{CCO}/2$ using split termination. The termination resistance is set by the external reference resistors, i.e., the resistors to V_{CCO} and ground are each twice the reference resistor value. Both HSTL and SSTL standards need 50 Ω external reference resistors. The DCI input standards supporting split termination are shown in Table 6-1.

Table 6-1: **DCI Input Standards Supporting Split Termination**

HSTL_I_DCI	DIFF_HSTL_I_DCI	SSTL2_I_DCI	DIFF_SSTL2_I_DCI
HSTL_I_DCI_18	DIFF_HSTL_I_DCI_18	SSTL2_II_DCI	DIFF_SSTL2_II_DCI
HSTL_II_DCI	DIFF_HSTL_II_DCI	SSTL18_I_DCI	DIFF_SSTL18_I_DCI
HSTL_II_DCI_18	DIFF_HSTL_II_DCI_18	SSTL18_II_DCI	DIFF_SSTL18_II_DCI
HSTL_II_T_DCI		SSTL2_II_T_DCI	
HSTL_II_T_DCI_18		SSTL18_II_T_DCI	

Figure 6-11 illustrates split termination inside a Virtex-5 device.

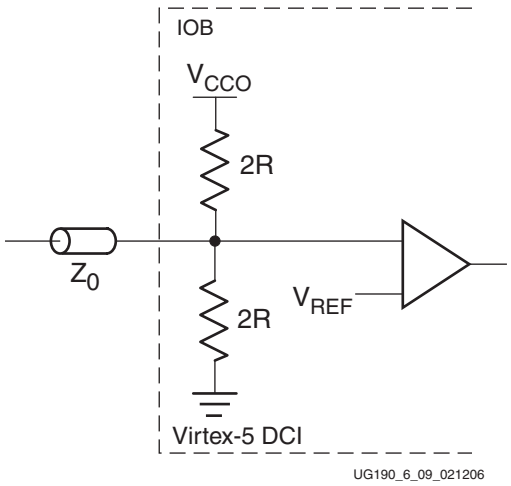


Figure 6-11: **Input Termination to V_{CCO}/2 Using DCI Split Termination**

Driver with Termination to V_{CCO} (Single Termination)

Some I/O standards (e.g., HSTL Class IV) require an output termination to V_{CCO}. Figure 6-12 illustrates an output termination to V_{CCO}.

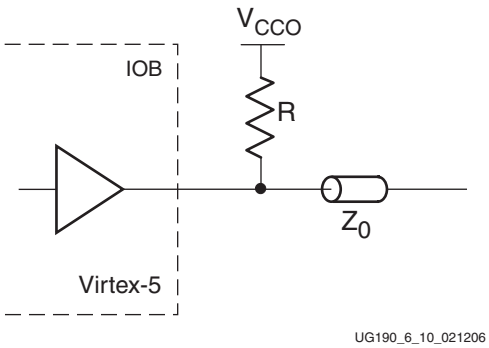


Figure 6-12: **Driver with Termination to V_{CCO} without DCI**

DCI can provide an output termination to V_{CCO} using single termination. In this case, DCI only controls the impedance of the termination, but not the driver. Both GTL and HSTL standards need 50 Ωexternal reference resistors. The DCI I/O standards supporting drivers with single termination are: GTL_DCI, GTLP_DCI, HSTL_IV_DCI, and HSTL_IV_DCI_18.

Figure 6-13 illustrates a driver with single termination inside a Virtex-5 device.

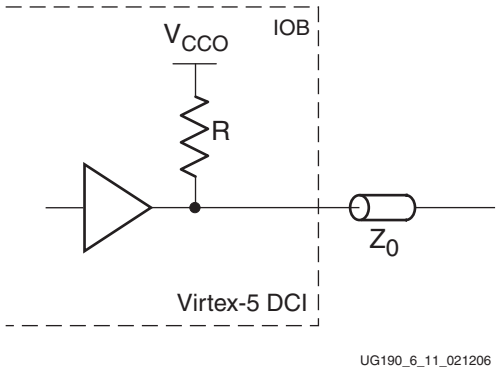


Figure 6-13: **Driver with Termination to V_{CCO} Using DCI Single Termination**

Driver with Termination to $V_{CCO}/2$ (Split Termination)

Some I/O standards, such as HSTL Class II, require an output termination to $V_{CCO}/2$ (see Figure 6-14).

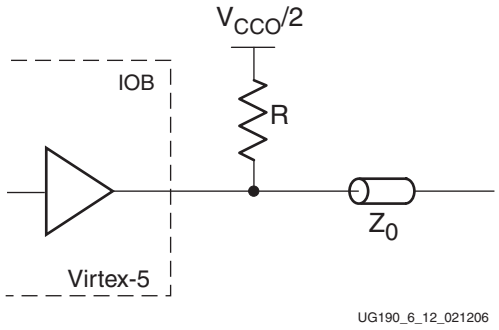


Figure 6-14: **Driver with Termination to $V_{CCO}/2$ without DCI**

DCI can provide output termination to $V_{CCO}/2$ using split termination. DCI only controls the impedance of the termination, but not the driver. Both HSTL and SSTL standards need 50 Ω external reference resistors. The DCI output standards supporting drivers with split termination are shown in Table 6-2.

Table 6-2: **DCI Output Standards Supporting Split Termination**

HSTL_II_DCI	DIFF_HSTL_II_DCI	SSTL2_II_DCI	DIFF_SSTL2_II_DCI
HSTL_II_DCI_18	DIFF_HSTL_II_DCI_18	SSTL18_II_DCI	DIFF_SSTL18_II_DCI

Figure 6-15 illustrates a driver with split termination inside a Virtex-5 device.

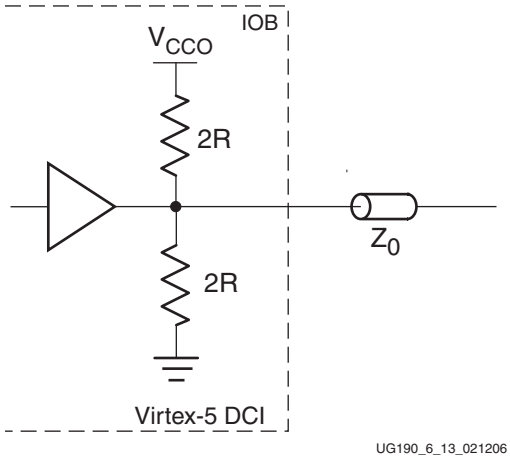


Figure 6-15: **Driver with Termination to V_{CCO}/2 Using DCI Split Termination**

DCI in Virtex-5 Device I/O Standards

DCI works with single-ended I/O standards. DCI supports the standards shown in Table 6-3.

Table 6-3: **Virtex-5 Device DCI I/O Standards**

LVDCI	HSTL_I_DCI	DIFF_HSTL_I_DCI	HSTL_III_DCI	SSTL2_I_DCI	DIFF_SSTL2_I_DCI
HSLVDCI	HSTL_I_DCI_18	DIFF_HSTL_I_DCI_18	HSTL_III_DCI_18	SSTL2_II_DCI	DIFF_SSTL2_II_DCI
LVDCI_DV2	HSTL_II_DCI	DIFF_HSTL_II_DCI	HSTL_IV_DCI	SSTL18_I_DCI	DIFF_SSTL18_I_DCI
GTL_DCI	HSTL_II_DCI_18	DIFF_HSTL_II_DCI_18	HSTL_IV_DCI_18	SSTL18_II_DCI	DIFF_SSTL18_II_DCI
GTLP_DCI	HSTL_II_T_DCI			SSTL2_II_T_DCI	
	HSTL_II_T_DCI_18			SSTL18_II_T_DCI	

To correctly use DCI in a Virtex-5 device, users must follow the following rules:

1. V_{CCO} pins must be connected to the appropriate V_{CCO} voltage based on the IOSTANDARDs in that bank.
2. Correct DCI I/O buffers must be used in the software either by using IOSTANDARD attributes or instantiations in the HDL code.

3. Some DCI standards require connecting the external reference resistors to the multipurpose pins (VRN and VRP) in the bank. Where this is required, these two multipurpose pins cannot be used as general-purpose I/O. Refer to the Virtex-5 FPGA pinout tables for the specific pin locations. Pin VRN must be pulled up to V_{CCO} by its reference resistor. Pin VRP must be pulled down to ground by its reference resistor.

   Some DCI standards do not require connecting the external reference resistors to the VRP/VRN pins. When these DCI-based I/O standards are the only ones in a bank, the the VRP and VRN pins in that bank *can be used* as general-purpose I/O.

   - DCI outputs that do not require reference resistors on VRP/VRN:

     HSTL_I_DCI
     HSTL_III_DCI
     HSTL_I_DCI_18
     HSTL_III_DCI_18
     SSTL2_I_DCI
     SSTL18_I_DCI

   - DCI inputs that do not require reference resistors on VRP/VRN:

     LVDCI_15
     LVDCI_18
     LVDCI_25
     LVDCI_33
     LVDCI_DV2_15
     LVDCI_DV2_18
     LVDCI_DV2_25

4. The value of the external reference resistors should be selected to give the desired output impedance. If using GTL_DCI, HSTL_DCI, or SSTL_DCI I/O standards, then the external reference resistors should be 50 Ω
5. The values of the reference resistors must be within the supported range (20 Ω – 100 Ω).
6. Follow the DCI I/O banking rules:
   a. V_{REF} must be compatible for all of the inputs in the same bank.
   b. V_{CCO} must be compatible for all of the inputs and outputs in the same bank.
   c. No more than one DCI I/O standard using single termination type is allowed per bank.
   d. No more than one DCI I/O standard using split termination type is allowed per bank.
   e. Single termination and split termination, controlled impedance driver, and controlled impedance driver with half impedance can co-exist in the same bank.
7. Master DCI is not supported in Banks 1 and 2.

The behavior of a DCI 3-state outputs is as follows:

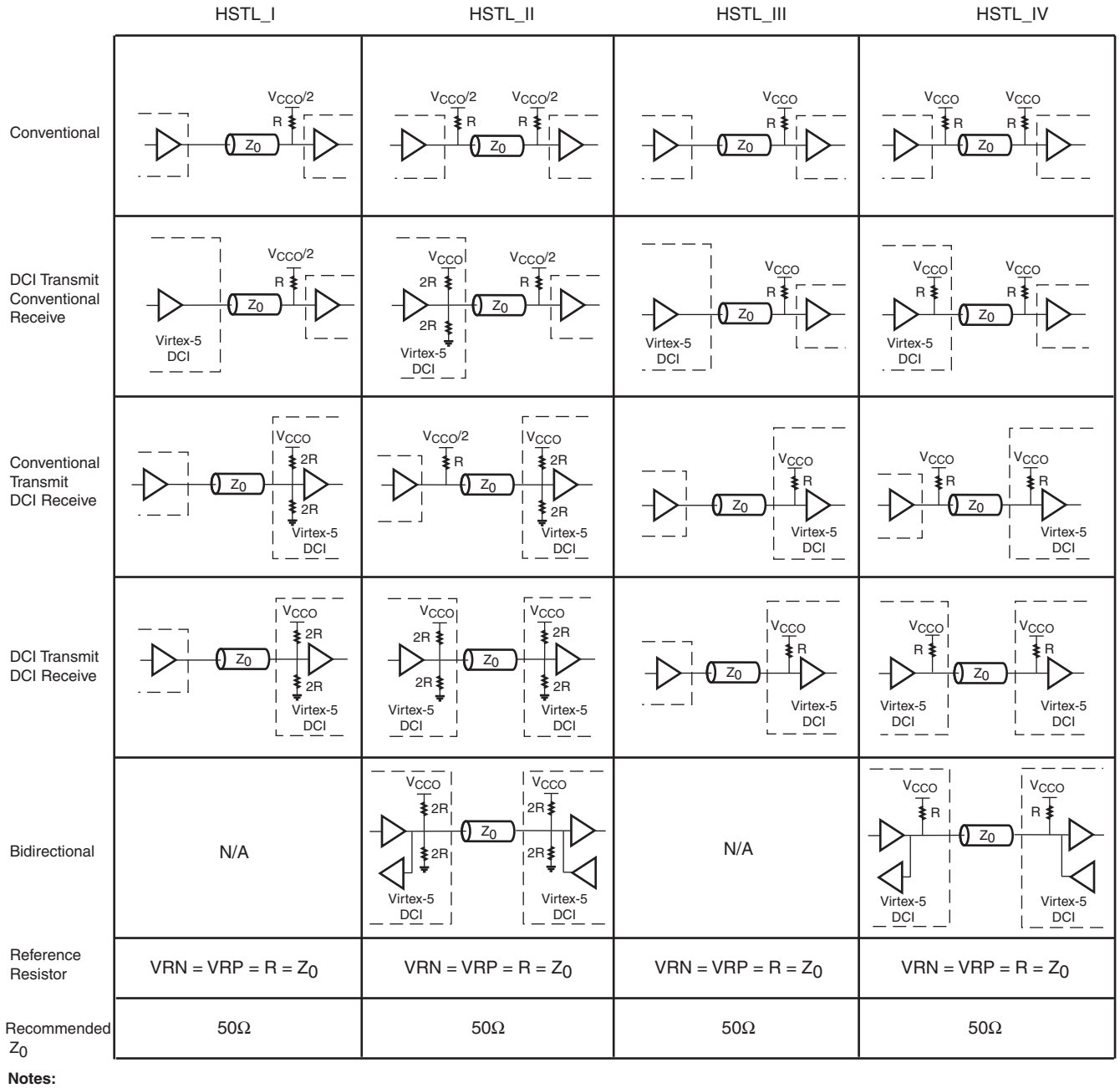
If a LVDCI or LVDCI_DV2 driver is in 3-state, the driver is 3-stated. If a driver with single or split termination is in 3-state, the driver is 3-stated but the termination resistor remains.

The following section lists actions that must be taken for each DCI I/O standard.

DCI Usage Examples

- Figure 6-16 provides examples illustrating the use of the HSTL_I_DCI, HSTL_II_DCI, HSTL_III_DCI, and HSTL_IV_DCI I/O standards.

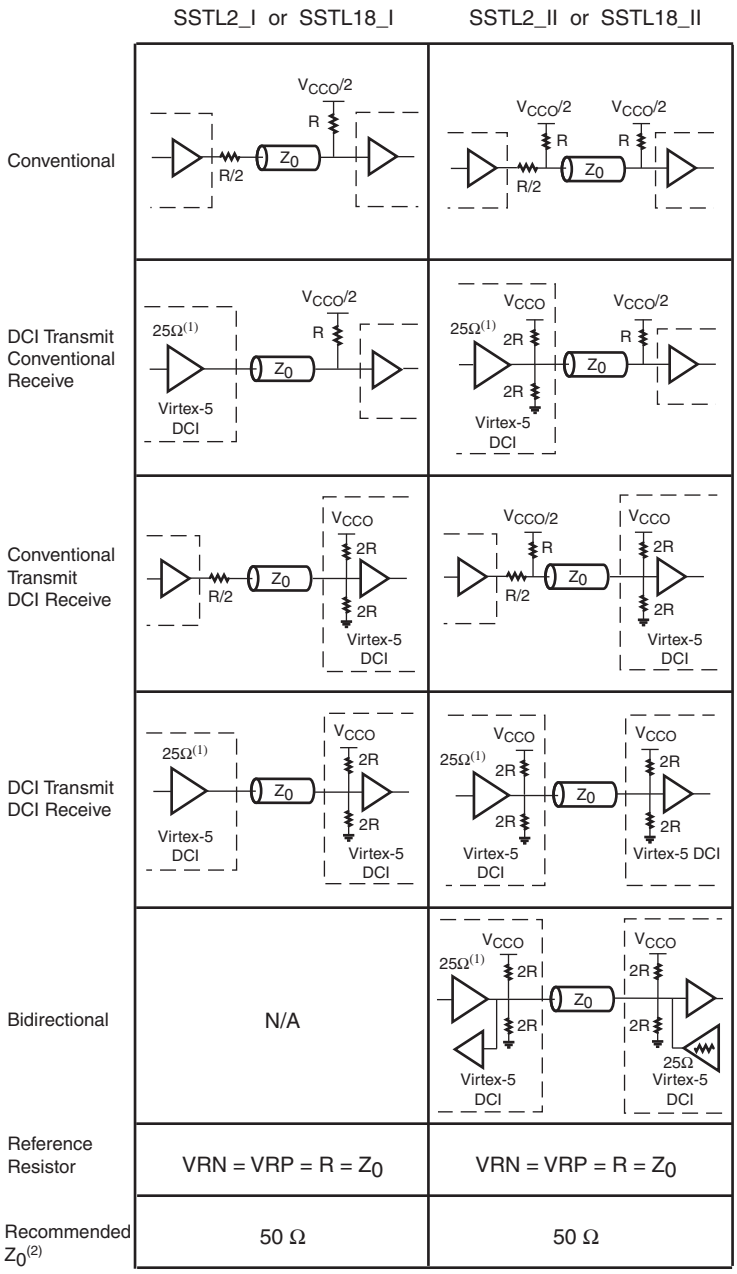
- Figure 6-17 provides examples illustrating the use of the SSTL2_I_DCI and SSTL2_II_DCI I/O standards.



Notes:
1. Z_0 is the recommended PCB trace impedance.

ug190_6_14_021206

Figure 6-16: **HSTL DCI Usage Examples**



Notes:
1. The SSTL-compatible 25 Ω or 20 Ω series resistor is accounted for in the DCI buffer, and it is not DCI controlled.
2. Z_0 is the recommended PCB trace impedance.

ug190_6_15_041106

Figure 6-17: **SSTL DCI Usage Examples**

Virtex-5 FPGA SelectIO Primitives

The Xilinx software library includes an extensive list of primitives to support a variety of I/O standards available in the Virtex-5 FPGA I/O primitives. The following are five generic primitive names representing most of the available single-ended I/O standards.

- IBUF (input buffer)
- IBUFG (clock input buffer)
- OBUF (output buffer)
- OBUFT (3-state output buffer)
- IOBUF (input/output buffer)

These six generic primitive names represent most of the available differential I/O standards:

- IBUFDS (input buffer)
- IBUFGDS (clock input buffer)
- OBUFDS (output buffer)
- OBUFTDS (3-state output buffer)
- IOBUFDS (input/output buffer)
- IBUFDS_DIFF_OUT (input buffer)

IBUF and IBUFG

Signals used as inputs to Virtex-5 devices must use an input buffer (IBUF). The generic Virtex-5 FPGA IBUF primitive is shown in Figure 6-18.

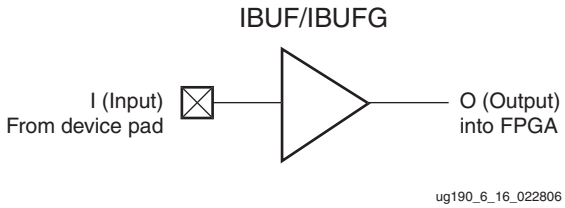


Figure 6-18: **Input Buffer (IBUF/IBUFG) Primitives**

The IBUF and IBUFG primitives are the same. IBUFGs are used when an input buffer is used as a clock input. In the Xilinx software tools, an IBUFG is automatically placed at clock input sites.

OBUF

An output buffer (OBUF) must be used to drive signals from Virtex-5 devices to external output pads. A generic Virtex-5 FPGA OBUF primitive is shown in Figure 6-19.

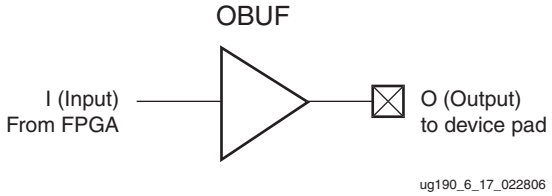


Figure 6-19: **Output Buffer (OBUF) Primitive**

OBUFT

The generic 3-state output buffer OBUFT, shown in Figure 6-20, typically implements 3-state outputs or bidirectional I/O.

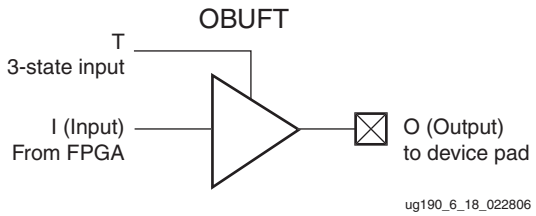


Figure 6-20: **3-State Output Buffer (OBUFT) Primitive**

IOBUF

The IOBUF primitive is needed when bidirectional signals require both an input buffer and a 3-state output buffer with an active High 3-state pin. Figure 6-21 shows a generic Virtex-5 FPGA IOBUF.

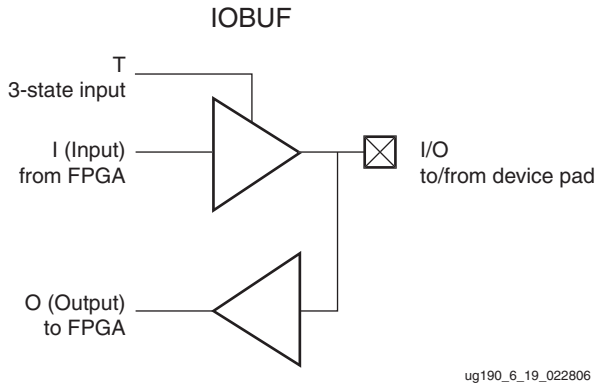


Figure 6-21: **Input/Output Buffer (IOBUF) Primitive**

IBUFDS and IBUFGDS

The usage and rules corresponding to the differential primitives are similar to the single-ended SelectIO primitives. Differential SelectIO primitives have two pins to and from the device pads to show the P and N channel pins in a differential pair. N channel pins have a "B" suffix.

Figure 6-22 shows the differential input buffer primitive.

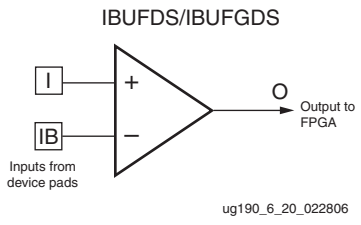


Figure 6-22: **Differential Input Buffer Primitive (IBUFDS/IBUFGDS)**

IBUFDS_DIFF_OUT

Figure 6-24 shows the differential input buffer primitive with a complementary output (OB). This primitive is for expert users only.

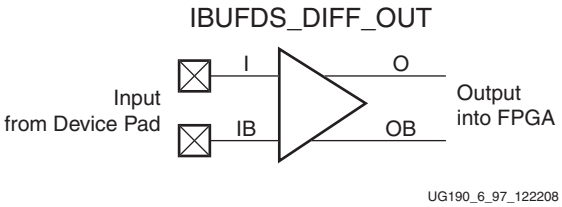


Figure 6-23: **Differential Input Buffer Primitive (IBUFDS_DIFF_OUT)**

OBUFDS

Figure 6-24 shows the differential output buffer primitive.

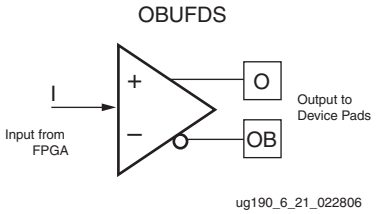


Figure 6-24: **Differential Output Buffer Primitive (OBUFDS)**

OBUFTDS

Figure 6-25 shows the differential 3-state output buffer primitive.

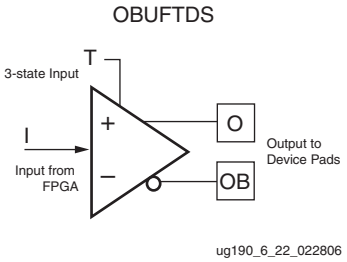


Figure 6-25: **Differential 3-state Output Buffer Primitive (OBUFTDS)**

IOBUFDS

Figure 6-26 shows the differential input/output buffer primitive.

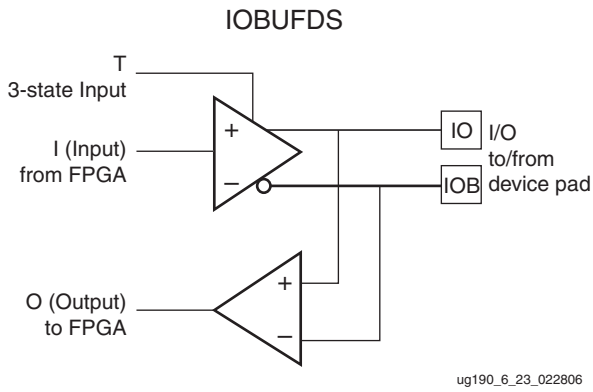


Figure 6-26: **Differential Input/Output Buffer Primitive (IOBUFDS)**

Virtex-5 FPGA SelectIO Attributes/Constraints

Access to some Virtex-5 FPGA I/O resource features (e.g., location constraints, input delay, output drive strength, and slew rate) is available through the attributes/constraints associated with these features. For more information a Constraints Guide is available on the Xilinx web site with syntax examples and VHDL/Verilog reference code. This guide is available inside the Software Manuals at:
http://www.support.xilinx.com/support/software_manuals.htm

Location Constraints

The location constraint (LOC) must be used to specify the I/O location of an instantiated I/O primitive. The possible values for the location constraint are all the external port identifiers (e.g., A8, M5, AM6, etc.). These values are device and package size dependent.

The LOC attribute uses the following syntax in the UCF file:

```
INST <I/O_BUFFER_INSTANTIATION_NAME> LOC = "<EXTERNAL_PORT_IDENTIFIER>";
```

Example:

```
INST MY_IO LOC=R7;
```

IOSTANDARD Attribute

The IOSTANDARD attribute is available to choose the values for an I/O standard for all I/O buffers. The supported I/O standards are listed in Table 6-39. The IOSTANDARD attribute uses the following syntax in the UCF file:

```
INST <I/O_BUFFER_INSTANTIATION_NAME> IOSTANDARD="<IOSTANDARD VALUE>";
```

The IOSTANDARD default for single-ended I/O is LVCMOS25, for differential I/Os the default is LVDS_25.

Output Slew Rate Attributes

A variety of attribute values provide the option of choosing the desired slew rate for single-ended I/O output buffers. For LVTTL and LVCMOS output buffers (OBUF, OBUFT, and IOBUF), the desired slew rate can be specified with the SLEW attribute.

The allowed values for the SLEW attribute are:

- SLEW = SLOW (Default)
- SLEW = FAST

The SLEW attribute uses the following syntax in the UCF file:

```
INST <I/O_BUFFER_INSTANTIATION_NAME> SLEW = "<SLEW_VALUE>";
```

By the default, the slew rate for each output buffer is set to SLOW. This is the default used to minimize the power bus transients when switching non-critical signals.

Output Drive Strength Attributes

For LVTTL and LVCMOS output buffers (OBUF, OBUFT, and IOBUF), the desired drive strength (in mA) can be specified with the DRIVE attribute.

The allowed values for the DRIVE attribute are:

- DRIVE = 2
- DRIVE = 4
- DRIVE = 6
- DRIVE = 8
- DRIVE = 12 (Default)
- DRIVE = 16
- DRIVE = 24

LVCMOS12 only supports the 2, 4, 6, 8 mA DRIVE settings. LVCMOS15 and LVCMOS18 only support the 2, 4, 6, 8, 12, and 16 mA DRIVE settings.

The DRIVE attribute uses the following syntax in the UCF file:

```
INST <I/O_BUFFER_INSTANTIATION_NAME> DRIVE = "<DRIVE_VALUE>";
```

PULLUP/PULLDOWN/KEEPER for IBUF, OBUFT, and IOBUF

When using 3-state output (OBUFT) or bidirectional (IOBUF) buffers, the output can have a weak pull-up resistor, a weak pull-down resistor, or a weak “keeper” circuit. For input (IBUF) buffers, the input can have either a weak pull-up resistor or a weak pull-down resistor. This feature can be invoked by adding the following possible constraint values to the relevant net of the buffers:

- PULLUP
- PULLDOWN
- KEEPER

Differential Termination Attribute

The differential termination (DIFF_TERM) attribute is designed for the Virtex-5 FPGA supported differential input I/O standards. It is used to turn the built-in, 100Ω, differential termination on or off.

The allowed values for the DIFF_TERM attribute are:

- TRUE
- FALSE (Default)

To specify the DIFF_TERM attribute, set the appropriate value in the generic map (VHDL) or inline parameter (Verilog) of the instantiated IBUFDS or IBUGDS component. Please refer to the ISE Language Templates or the Virtex-5 FPGA HDL Libraries Guide for the proper syntax for instantiating this component and setting the DIFF_TERM attribute.

Virtex-5 FPGA I/O Resource VHDL/Verilog Examples

The VHDL and Verilog example syntaxes to declare a standard for Virtex-5 FPGA I/O resources are found in the Virtex-5 FPGA Libraries Guide.

Specific Guidelines for I/O Supported Standards

The following subsections provide an overview of the I/O standards supported by all Virtex-5 devices.

While most Virtex-5 FPGA I/O supported standards specify a range of allowed voltages, this chapter records typical voltage values only. Detailed information on each specification can be found on the Electronic Industry Alliance JEDEC web site at http://www.jedec.org.

LVTTL (Low Voltage Transistor-Transistor Logic)

The low-voltage TTL (LVTTL) standard is a general purpose EIA/JESDSA standard for 3.3V applications using an LVTTL input buffer and a push-pull output buffer. This standard requires a 3.3V input and output supply voltage (V_{CCO}), but does not require the use of a reference voltage (V_{REF}) or a termination voltage (V_{TT}).

Sample circuits illustrating both unidirectional and bidirectional LVTTL termination techniques are shown in Figure 6-27 and Figure 6-28.

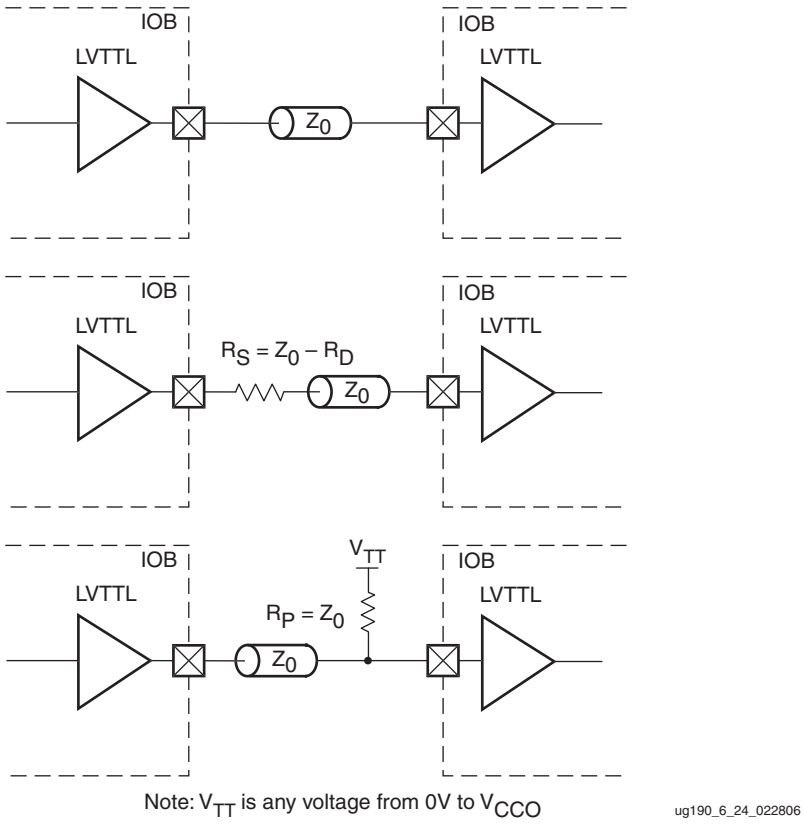


Figure 6-27: **LVTTL Unidirectional Termination**

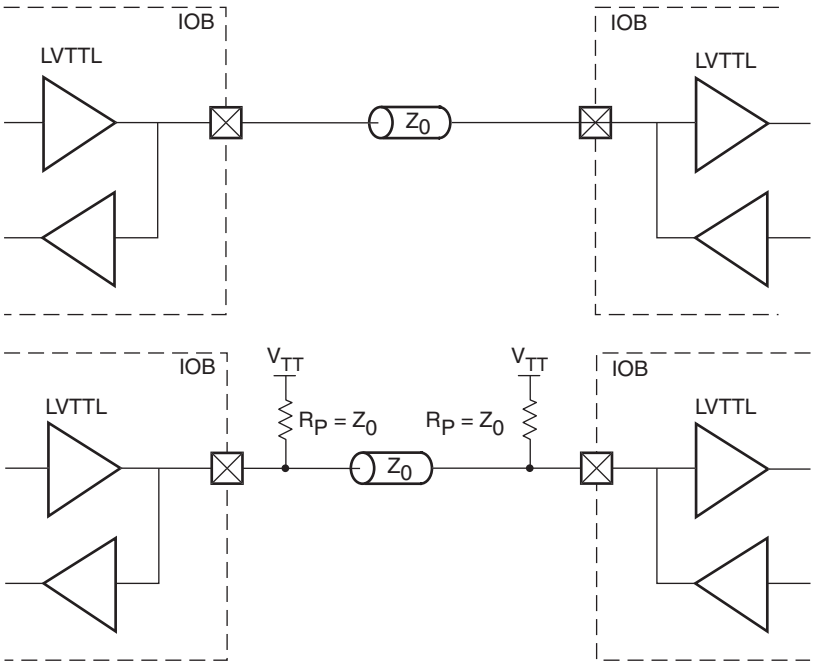


Figure 6-28: **LVTTL Bidirectional Termination**

Table 6-4 lists the LVTTL DC voltage specifications.

Table 6-4: **LVTTL DC Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	3.0	3.3	3.45
V_{REF}	–	–	–
V_{TT}	–	–	–
V_{IH}	2.0	–	3.45
V_{IL}	–0.2	–	0.8
V_{OH}	2.4	–	–
V_{OL}	–	–	0.4
I_{OH} at V_{OH} (mA)	Note 2	–	–
I_{OL} at V_{OL} (mA)	Note 2	-	–

Notes:

1. V_{OL} and V_{OH} for lower drive currents are sample tested.
2. Supported DRIVE strengths are 2, 4, 6, 8, 12, 16, and 24 mA

Table 6-5 details the allowed attributes that can be applied to the LVTTL I/O standard.

Table 6-5: **Allowed Attributes for the LVTTL I/O Standard**

Attributes	Primitives		
	IBUF/IBUFG	OBUF/OBUFT	IOBUF
IOSTANDARD	LVTTL	LVTTL	LVTTL
DRIVE	UNUSED	2, 4, 6, 8, 12, 16, 24	2, 4, 6, 8, 12, 16, 24
SLEW	UNUSED	{FAST, SLOW}	{FAST, SLOW}

LVCMOS (Low Voltage Complementary Metal Oxide Semiconductor)

LVCMOS is a widely used switching standard implemented in CMOS transistors. This standard is defined by JEDEC (JESD 8-5). The LVCMOS standards supported in Virtex-5 FPGAs are: LVCMOS12, LVCMOS15, LVCMOS18, LVCMOS25, and LVCMOS33.

Sample circuits illustrating both unidirectional and bidirectional LVCMOS termination techniques are shown in Figure 6-29 and Figure 6-30.

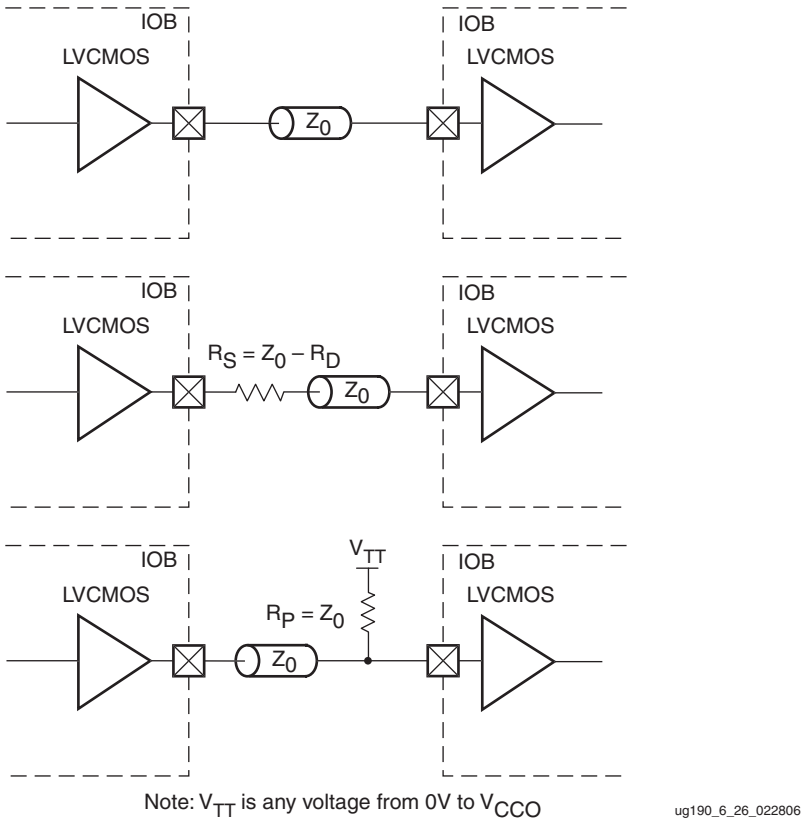


Figure 6-29: **LVCMOS Unidirectional Termination**

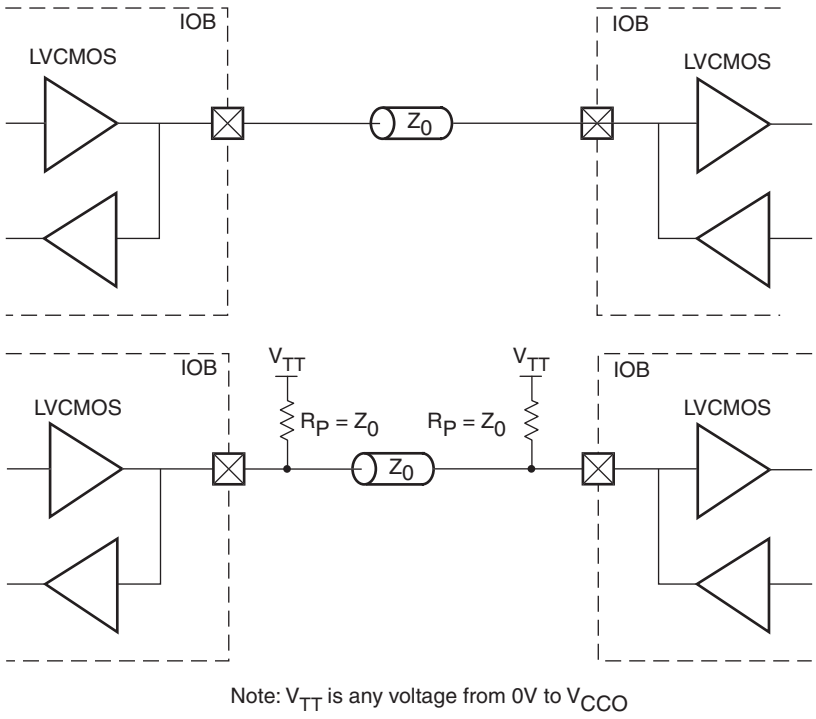


Figure 6-30: **LVCMOS Bidirectional Termination**

Table 6-6 details the allowed attributes that can be applied to the LVCMOS33 and LVCMOS25 I/O standards.

Table 6-6: **Allowed Attributes for the LVCMOS33 and LVCMOS25 I/O Standards**

Attributes	Primitives		
	IBUF/IBUFG	OBUF/OBUFT	IOBUF
IOSTANDARD	LVCMOS33 LVCMOS25	LVCMOS33 LVCMOS25	LVCMOS33 LVCMOS25
DRIVE	UNUSED	2, 4, 6, 8, 12, 16, 24	2, 4, 6, 8, 12, 16, 24
SLEW	UNUSED	{FAST, SLOW}	{FAST, SLOW}

Table 6-7 details the allowed attributes that can be applied to the LVCMOS18 and LVCMOS15 I/O standards.

Table 6-7: **Allowed Attributes for the LVCMOS18 and LVCMOS15 I/O Standard**

Attributes	Primitives		
	IBUF/IBUFG	OBUF/OBUFT	IOBUF
IOSTANDARD	LVCMOS18 LVCMOS15	LVCMOS18 LVCMOS15	LVCMOS18 LVCMOS15
DRIVE	UNUSED	2, 4, 6, 8, 12, 16	2, 4, 6, 8, 12, 16
SLEW	UNUSED	{FAST, SLOW}	{FAST, SLOW}

Table 6-8 details the allowed attributes that can be applied to the LVCMOS12 I/O standard.

Table 6-8: **Allowed Attributes for the LVCMOS12 I/O Standard**

Attributes	Primitives		
	IBUF/IBUFG	OBUF/OBUFT	IOBUF
IOSTANDARD	LVCMOS12	LVCMOS12	LVCMOS12
DRIVE	UNUSED	2, 4, 6, 8	2, 4, 6, 8
SLEW	UNUSED	{FAST, SLOW}	{FAST, SLOW}

LVDCI (Low Voltage Digitally Controlled Impedance)

Using these I/O buffers configures the outputs as controlled impedance drivers. The receiver of LVDCI is identical to a LVCMOS receiver. Some I/O standards, such as LVTTL, LVCMOS, etc., must have a drive impedance that matches the characteristic impedance of the driven line. Virtex-5 devices provide a controlled impedance output driver to provide series termination without external source termination resistors. The impedance is set by the common external reference resistors, with resistance equal to the trace characteristic impedance, Z_0.

Sample circuits illustrating both unidirectional and bidirectional termination techniques for a controlled impedance driver are shown in Figure 6-31 and Figure 6-32. The DCI I/O standards supporting a controlled impedance driver are: LVDCI_15, LVDCI_18, LVDCI_25, and LVDCI_33.

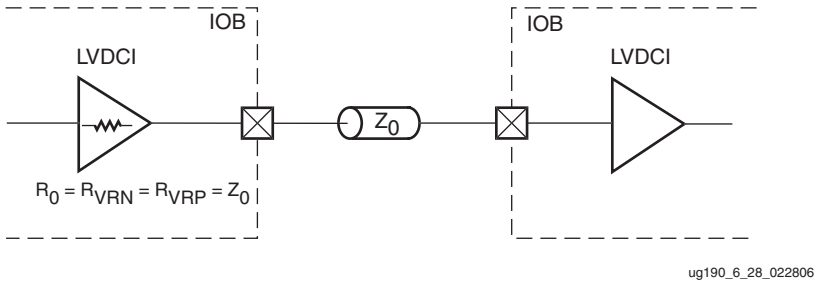


Figure 6-31: **Controlled Impedance Driver with Unidirectional Termination**

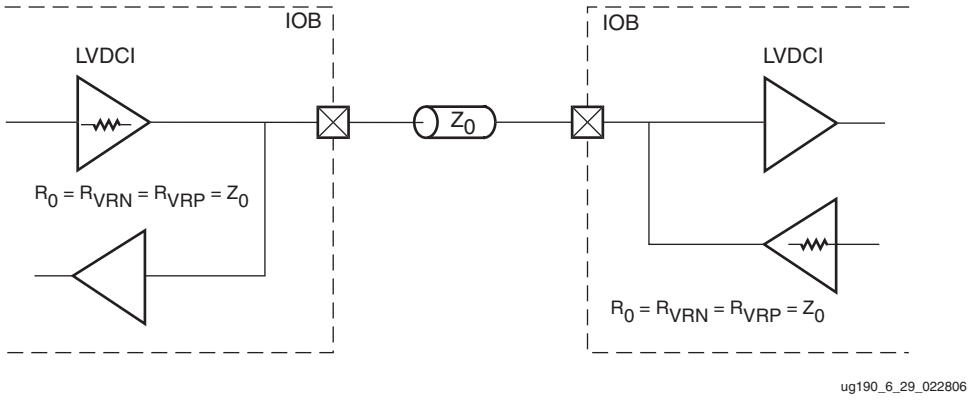


Figure 6-32: **Controlled Impedance Driver with Bidirectional Termination**

LVDCI_DV2

A controlled impedance driver with half impedance (source termination) can also provide drivers with one half of the impedance of the reference resistors. This allows reference resistors to be twice as large, thus reducing static power consumption through VRN/VRP. The I/O standards supporting a controlled impedance driver with half impedance are: LVDCI_DV2_15, LVDCI_DV2_18, and LVDCI_DV2_25. Figure 6-33 and Figure 6-34 illustrate a controlled driver with half impedance unidirectional and bidirectional termination.

To match the drive impedance to Z_0 when using a driver with half impedance, the reference resistor R must be twice Z_0.

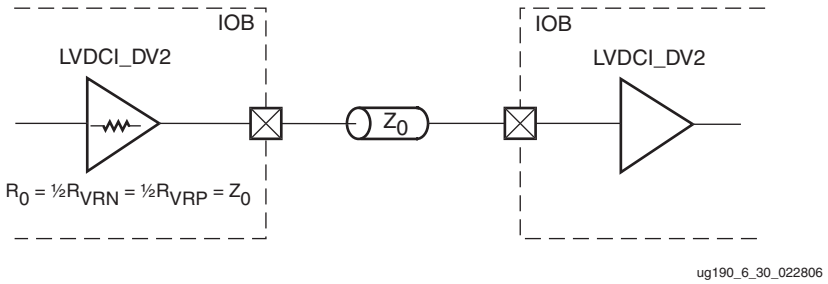


Figure 6-33: **Controlled Impedance Driver with Half Impedance Unidirectional Termination**

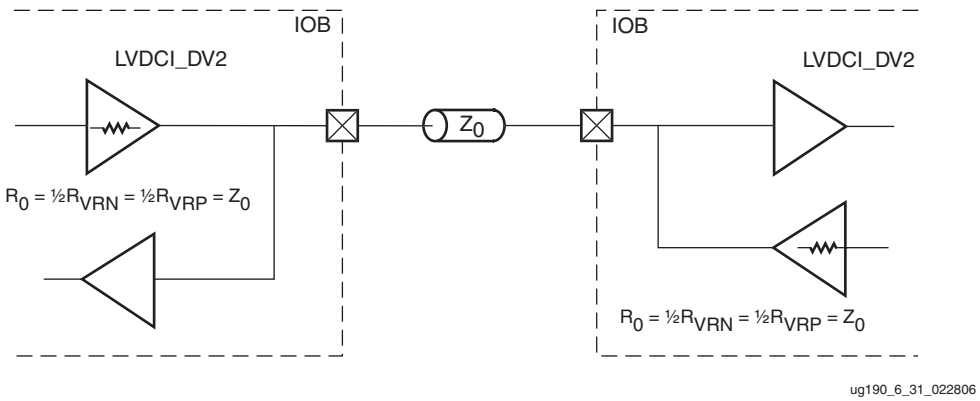


Figure 6-34: **Controlled Impedance Driver with Half Impedance Bidirectional Termination**

There are no drive strength settings for LVDCI drivers. When the driver impedance is one-half of the VRN/VRP reference resistors, it is indicated by the addition of DV2 to the attribute name.

Table 6-9 lists the LVCMOS, LVDCI, and LVDCI_DV2 voltage specifications.

Table 6-9: **LVCMOS, LVDCI, and LVDCI_DV2 DC Voltage Specifications at Various Voltage References**

Standard	+3.3V			+2.5V			+1.8V			+1.5V			+1.2V(2)		
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
V_{CCO} [V]	3.0	3.3	3.45	2.3	2.5	2.7	1.7	1.8	1.9	1.4	1.5	1.6	1.1	1.2	1.3
V_{IH} [V]	2.0	–	3.45	1.7	–	V_{CCO}+0.3	1.105	–	V_{CCO}+0.3	0.91	–	V_{CCO}+0.3	0.715	–	V_{CCO}+0.3
V_{IL} [V]	–0.2	–	0.8	–0.3	–	0.7	–0.3	–	0.665	–0.3	–	0.56	0.3	–	0.455
V_{OH} [V]	2.6	–	–	1.9	–	–	1.25	–	–	1.05	–	0.825	–	–	–
V_{OL} [V]	–	–	0.4	–	–	0.4	–	–	0.45	–	–	0.4	–	–	0.325
I_{IN} [µA]	–	–	± 5	–	–	± 5	–	–	± 5	–	–	± 10	–	–	± 10

Notes:

1. V_{OL} and V_{OH} for lower drive currents are sample tested.
2. Only LVCMOS is supported at + 1.2V with valid DRIVE attributes of 2, 4, 6, 8.

HSLVDCI (High-Speed Low Voltage Digitally Controlled Impedance)

The HSLVDCI standard is intended for bidirectional use. The driver is identical to LVDCI, while the input is identical to HSTL and SSTL. By using a V_{REF}-referenced input, HSLVDCI allows greater input sensitivity at the receiver than when using a single-ended LVCMOS-type receiver.

A sample circuit illustrating bidirectional termination techniques for an HSLVDCI controlled impedance driver is shown in Figure 6-35. The DCI I/O standards supporting a controlled impedance driver with a V_{REF} referenced input are: HSLVDCI_15, HSLVDCI_18, HSLVDCI_25, and HSLVDCI_33.

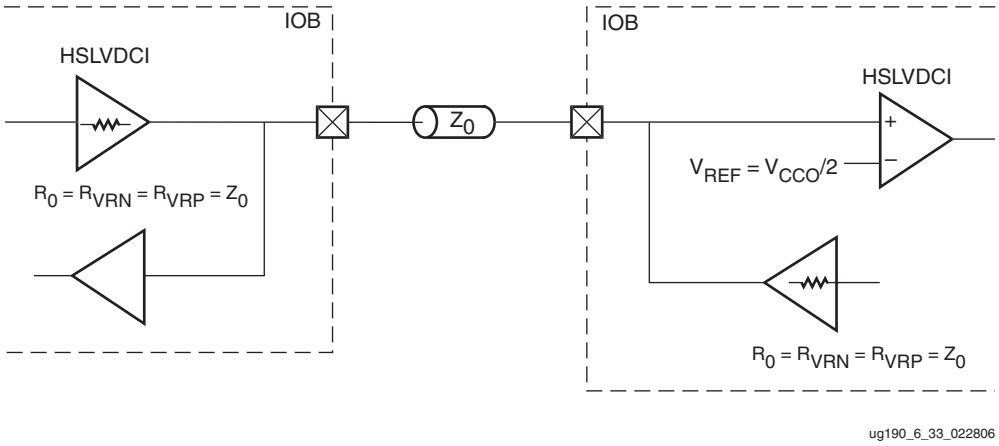


Figure 6-35: **HSLVDCI Controlled Impedance Driver with Bidirectional Termination**

For output DC voltage specifications, refer to the LVDCI V_{OH} and V_{OL} entries in Table 6-9 "LVCMOS, LVDCI, and LVDCI_DV2 DC Voltage Specifications at Various Voltage References." Table 6-10 lists the input DC voltage specifications when using HSLVDCI. Valid values of V_{CCO} are 1.5V, 1.8V, 2.5V, and 3.3V. Select V_{REF} to provide the optimum noise margin in specific use conditions.

Table 6-10: **HSLVDCI Input DC Voltage Specifications**

Standard	Min	Typ	Max
V_{REF}	–	$V_{CCO}/2$	–
V_{IH}	V_{REF} + 0.1	–	–
V_{IL}	–	–	V_{REF} – 0.1

PCI-X, PCI-33, PCI-66 (Peripheral Component Interconnect)

The PCI™ standard specifies support for 33 MHz and 66 MHz bus applications. The PCI-X™ standard specifies support for 66 MHz and 133 MHz bus applications. These standards use an LVTTL input buffer and a push-pull output buffer. These standards do not require the use of a reference voltage (V_{REF}) or a board termination voltage (V_{TT}). However, they do require 3.3V input/output source voltage (V_{CCO}).

A PCI undershoot/overshoot specification could require V_{CCO} to be regulated at 3.0V as discussed in "Regulating V_{CCO} at 3.0V," page 302. This is not necessary if overshoot and undershoot are controlled by careful design.

Table 6-11 and Table 6-12 lists the DC voltage specifications.

Table 6-11: **PCI33_3, PCI66_3 Voltage Specifications(2)**

Parameter	Min	Typ	Max
V_{CCO}	3.0	3.3	3.5
V_{REF}	–	–	–
V_{TT}	–	–	–
$V_{IH} = 0.5 \times V_{CCO}$	1.5	1.65	V_{CCO}
$V_{IL} = 0.3 \times V_{CCO}$	–0.2	0.99	1.05
$V_{OH} = 0.9 \times V_{CCO}$	2.7	–	–
$V_{OL} = 0.1 \times V_{CCO}$	–	–	0.35
I_{OH} at V_{OH} (mA)	(Note 1)	–	–
I_{OL} at V_{OL} (mA)	(Note 1)	–	–

Notes:

1. Tested according to the relevant specification.
2. For complete specifications, refer to the PCI specification.

Table 6-12: **PCI-X DC Voltage Specifications(2)**

Parameter	Min	Typ	Max
V_{CCO}	3.0	3.3	3.5
V_{REF}	–	–	–
V_{TT}	–	–	–
$V_{IH} = 0.5 \times V_{CCO}$	1.5	1.65	V_{CCO}
$V_{IL} = 0.35 \times V_{CCO}$	–0.2	1.155	1.225
$V_{OH} = 0.9 \times V_{CCO}$	2.7	–	–
$V_{OL} = 0.1 \times V_{CCO}$	–	–	0.35
I_{OH} at V_{OH} (mA)	(Note 1)	–	–
I_{OL} at V_{OL} (mA)	(Note 1)	–	–

Notes:

1. Tested according to the relevant specification.
2. For complete specifications, refer to the PCI-X specification.

GTL (Gunning Transceiver Logic)

The Gunning Transceiver Logic (GTL) standard is a high-speed bus standard (JESD8.3) invented by Xerox. Xilinx has implemented the terminated variation for this standard. This standard requires a differential amplifier input buffer and an open-drain output buffer. The negative terminal of the differential input buffer is referenced to the V_{REF} pin.

A sample circuit illustrating a valid termination technique for GTL with external parallel termination and unconnected V_{CCO} is shown in Figure 6-36.

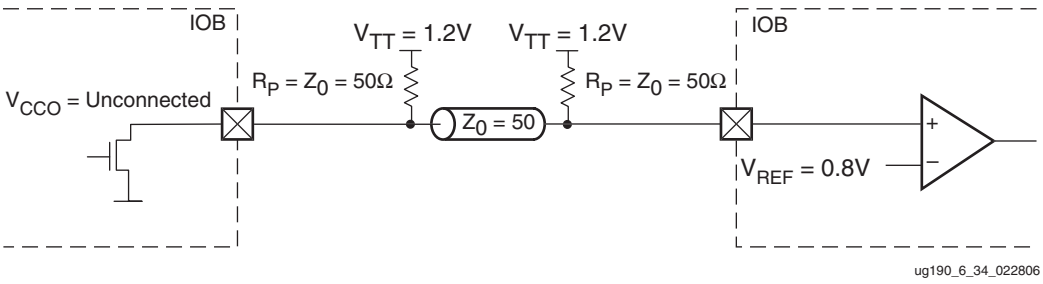


Figure 6-36: **GTL with External Parallel Termination and Unconnected V_{CCO}**

GTL_DCI Usage

GTL does not require a V_{CCO} voltage. However, for GTL_DCI, V_{CCO} must be connected to 1.2V. GTL_DCI provides single termination to V_{CCO} for inputs or outputs.

A sample circuit illustrating a valid termination technique for GTL_DCI with internal parallel driver and receiver termination is shown in Figure 6-37.

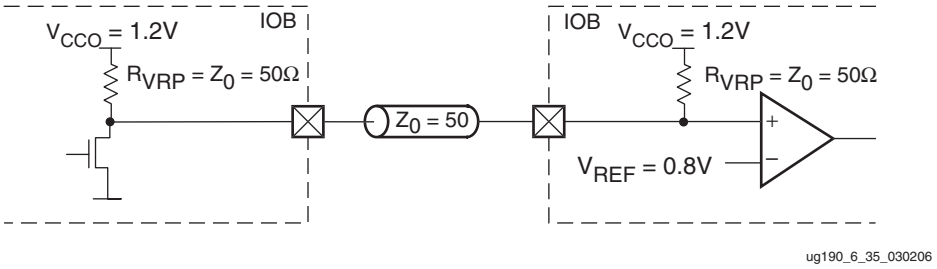


Figure 6-37: **GTL_DCI with Internal Parallel Driver and Receiver Termination**

Table 6-13 lists the GTL DC voltage specifications.

Table 6-13: **GTL DC Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	–	N/A	-
$V_{REF} = N \times V_{TT}$ (1)	0.74	0.8	0.86
V_{TT}	1.14	1.2	1.26
$V_{IH} = V_{REF} + 0.05$	0.79	0.83	–
$V_{IL} = V_{REF} - 0.05$	–	0.77	0.81
V_{OH}	–	–	–

Table 6-13: **GTL DC Voltage Specifications *(Continued)***

Parameter	Min	Typ	Max
V_{OL}	–	0.2	0.4
I_{OH} at V_{OH} (mA)	–	–	–
I_{OL} at V_{OL} (mA) at 0.4V	32	–	–
I_{OL} at V_{OL} (mA) at 0.2V	-	–	40

Notes:

1. N must be greater than or equal to 0.653 and less than or equal to 0.68.

GTLP (Gunning Transceiver Logic Plus)

The Gunning Transceiver Logic Plus, or GTL+ standard is a high-speed bus standard (JESD8.3) first used by the Pentium Pro Processor. This standard requires a differential amplifier input buffer and a open-drain output buffer. The negative terminal of the differential input buffer is referenced to the V_{REF} pin.

A sample circuit illustrating a valid termination technique for GTL+ with external parallel termination and unconnected V_{CCO} is shown in Figure 6-38.

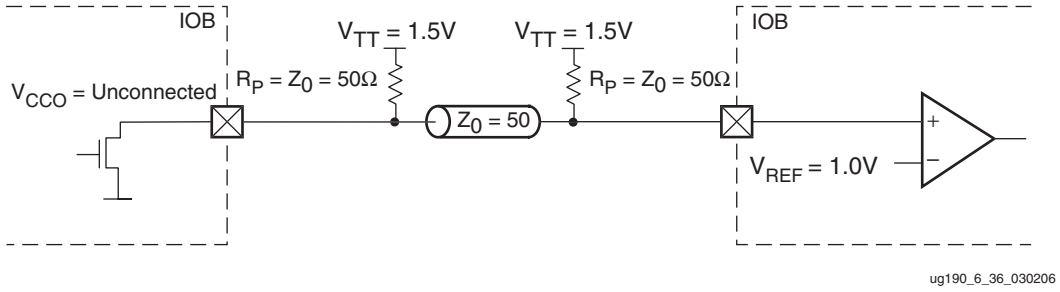


Figure 6-38: **GTL+ with External Parallel Termination and Unconnected V_{CCO}**

GTLP_DCI Usage

GTL+ does not require a V_{CCO} voltage. However, for GTLP_DCI, V_{CCO} must be connected to 1.5V. GTLP_DCI provides single termination to V_{CCO} for inputs or outputs.

A sample circuit illustrating a valid termination technique for GTLP_DCI with internal parallel driver and receiver termination is shown in Figure 6-39.

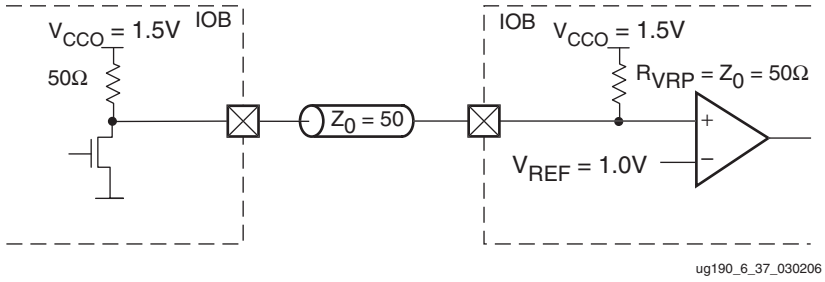


Figure 6-39: **GTLP_DCI Internal Parallel Driver and Receiver Termination**

Table 6-14 lists the GTLP DC voltage specifications.

Table 6-14: **GTLP DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	–	–	–
$V_{REF} = N \times V_{TT}$ (1)	0.88	1.0	1.12
V_{TT}	1.35	1.5	1.65
$V_{IH} = V_{REF} + 0.1$	0.98	1.1	–
$V_{IL} = V_{REF} - 0.1$	–	0.9	1.02
V_{OH}	–	–	–
V_{OL}	0.3	0.45	0.6
I_{OH} at V_{OH} (mA)	–	–	–
I_{OL} at V_{OL} (mA) at 0.6V	36	–	–
I_{OL} at V_{OL} (mA) at 0.3V	–	–	48

Notes:

1. N must be greater than or equal to 0.653 and less than or equal to 0.68.

HSTL (High-Speed Transceiver Logic)

The High-Speed Transceiver Logic (HSTL) standard is a general purpose high-speed bus standard sponsored by IBM (EIA/JESD8-6). The 1.5V and 1.8V have four variations or classes. To support clocking high speed memory interfaces, a differential version of this standard was added. Virtex-5 FPGA I/O supports all four classes for 1.5V and 1.8V and the differential versions of classes I and II. These differential versions of the standard require a differential amplifier input buffer and a push-pull output buffer.

HSTL_ I, HSTL_ III, HSTL_ I_18, HSTL_ III_18, HSTL_I_12

HSTL_I uses $V_{CCO}/2$ as a parallel termination voltage (V_{TT}). HSTL_III uses V_{CCO} as a parallel termination voltage (V_{TT}). HSTL_I and HSTL_III are intended to be used in unidirectional links.

HSTL_ I_DCI, HSTL_ III_DCI, HSTL_ I_DCI_18, HSTL_ III_DCI_18

HSTL_I_DCI provides on-chip split thevenin termination powered from V_{CCO}, creating an equivalent parallel termination voltage (V_{TT}) of $V_{CCO}/2$. HSTL_III_DCI provides on-chip single termination powered from V_{CCO}. HSTL_I_DCI and HSTL_III_DCI are intended to be used in unidirectional links.

HSTL_ II, HSTL_ IV, HSTL_ II_18, HSTL_ IV_18

HSTL_II uses $V_{CCO}/2$ as a parallel termination voltage (V_{TT}). HSTL_IV uses V_{CCO} as a parallel termination voltage (V_{TT}). HSTL_II and HSTL_IV are intended to be used in bidirectional links.

HSTL_ II_DCI, HSTL_ IV_DCI, HSTL_ II_DCI_18, HSTL_ IV_DCI_18

HSTL_II_DCI provides on-chip split thevenin termination powered from V_{CCO}, creating an equivalent termination voltage of $V_{CCO}/2$. HSTL_IV_ DCI provides single termination to V_{CCO} (V_{TT}). HSTL_II_DCI and HSTL_IV_ DCI are intended to be used in bidirectional links.

HSTL_ II_T_DCI, HSTL_ II_T_DCI_18

HSTL_ II_T_DCI and HSTL_ II_T_DCI_18 provide on-chip split-thevenin termination powered from V_{CCO} that creates an equivalent termination voltage of $V_{CCO}/2$ when these standards are 3-stated. When not 3-stated, these two standards do not have termination.

DIFF_HSTL_ II, DIFF_HSTL_II_18

Differential HSTL class II pairs complimentary single-ended HSTL_II type drivers with a differential receiver. Differential HSTL class II is intended to be used in bidirectional links. Differential HSTL can also be used for differential clock and DQS signals in memory interface designs.

DIFF_HSTL_II_DCI, DIFF_HSTL_II_DCI_18

Differential HSTL class II pairs complimentary single-ended HSTL_II type drivers with a differential receiver, including on-chip differential split-thevenin termination. Differential HSTL class II is intended to be used in bidirectional links. Differential HSTL can also be used for differential clock and DQS signals in memory interface designs.

DIFF_HSTL_I, DIFF_HSTL_I_18

Differential HSTL class I pairs complimentary single-ended HSTL_I type drivers with a differential receiver. Differential HSTL class I is intended to be used in unidirectional links.

DIFF_HSTL_I_DCI, DIFF_HSTL_I_DCI_18

Differential HSTL class I pairs complimentary single-ended HSTL_I type drivers with a differential receiver, including on-chip differential split-thevenin termination. Differential HSTL class I is intended to be used in unidirectional links.

HSTL Class I

Figure 6-40 shows a sample circuit illustrating a valid termination technique for HSTL Class I.

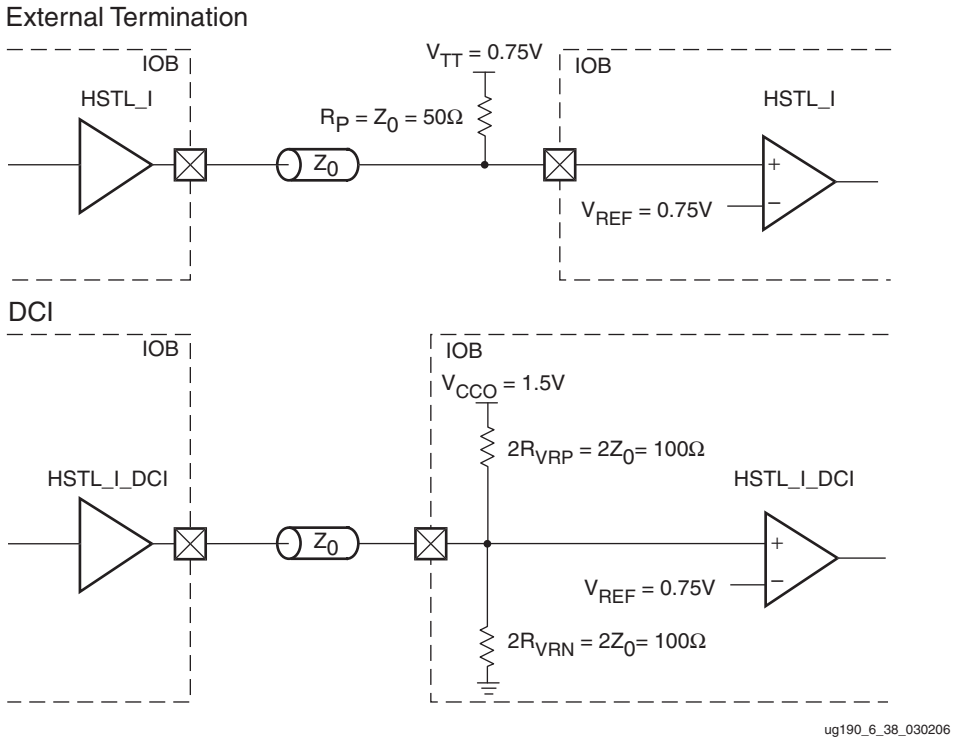


Figure 6-40: **HSTL Class I Termination**

Table 6-15 lists the HSTL Class I DC voltage specifications.

Table 6-15: **HSTL Class I DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	1.40	1.50	1.60
V_{REF} (2)	0.68	0.75	0.90
V_{TT}	–	$V_{CCO} \times 0.5$	–
V_{IH}	V_{REF} + 0.1	–	–
V_{IL}	–	–	V_{REF} – 0.1
V_{OH}	V_{CCO} – 0.4	–	–
V_{OL}	–	–	0.4
I_{OH} at V_{OH} (mA)(1)	–8	–	–
I_{OL} at V_{OL} (mA)(1)	8	–	–

Notes:

1. V_{OL} and V_{OH} for lower drive currents are sample tested.
2. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."

Differential HSTL Class I

Figure 6-41 shows a sample circuit illustrating a valid termination technique for differential HSTL Class I (1.5V) with unidirectional termination.

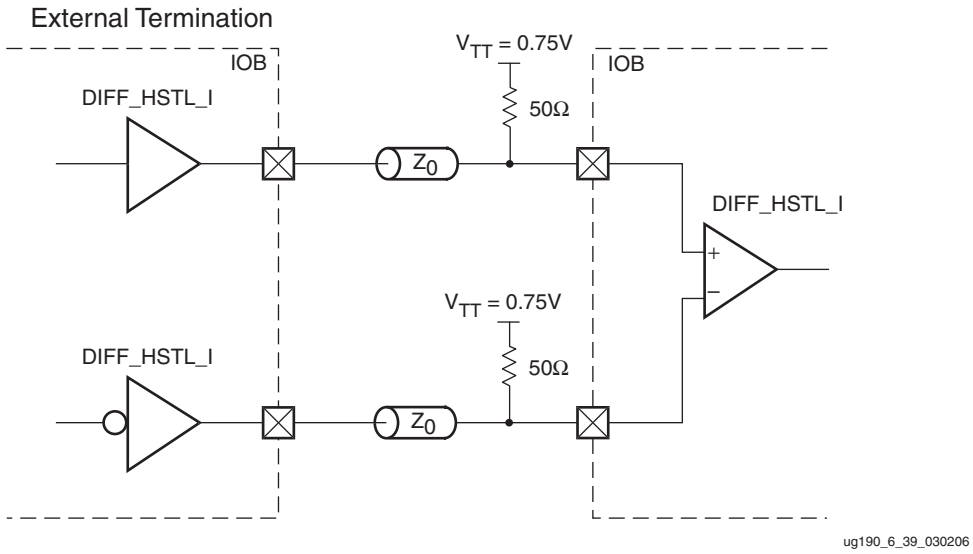


Figure 6-41: **Differential HSTL (1.5V) Class I Unidirectional Termination**

Figure 6-42 shows a sample circuit illustrating a valid termination technique for differential HSTL Class I (1.5V) with unidirectional DCI termination.

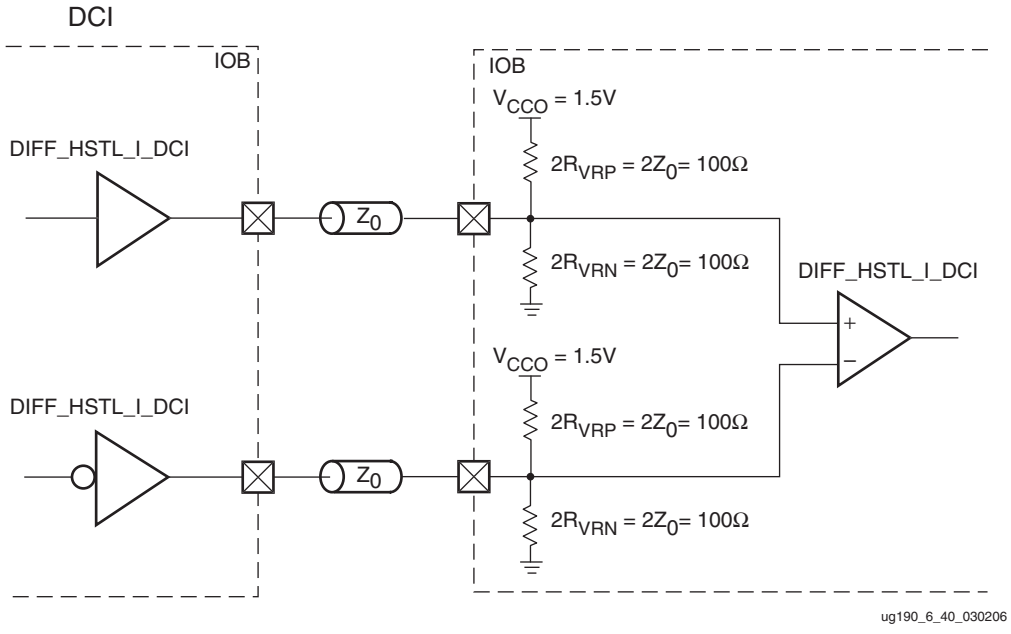


Figure 6-42: **Differential HSTL (1.5V) Class I DCI Unidirectional Termination**

Table 6-16 lists the differential HSTL Class I DC voltage specifications.

Table 6-16: **Differential HSTL Class I DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	1.40	1.50	1.60
V_{TT}	–	$V_{CCO} \times 0.5$	–
V_{IN} (DC)	–0.30	–	V_{CCO} + 0.30
V_{DIFF} (DC)	0.20	–	V_{CCO} + 0.60
V_{CM} (DC)(1)	0.68	–	0.90
V_{DIFF} (AC)	0.40	–	V_{CCO} + 0.60
V_X (Crossover)(2)	0.68	–	0.90

Notes:

1. Common mode voltage: $V_{CM} = V_P - ((V_P - V_N)/2)$
2. Crossover point: V_X where $V_P - V_N = 0$ (AC coupled)

HSTL Class II

Figure 6-43 shows a sample circuit illustrating a valid termination technique for HSTL Class II (1.5V) with unidirectional termination.

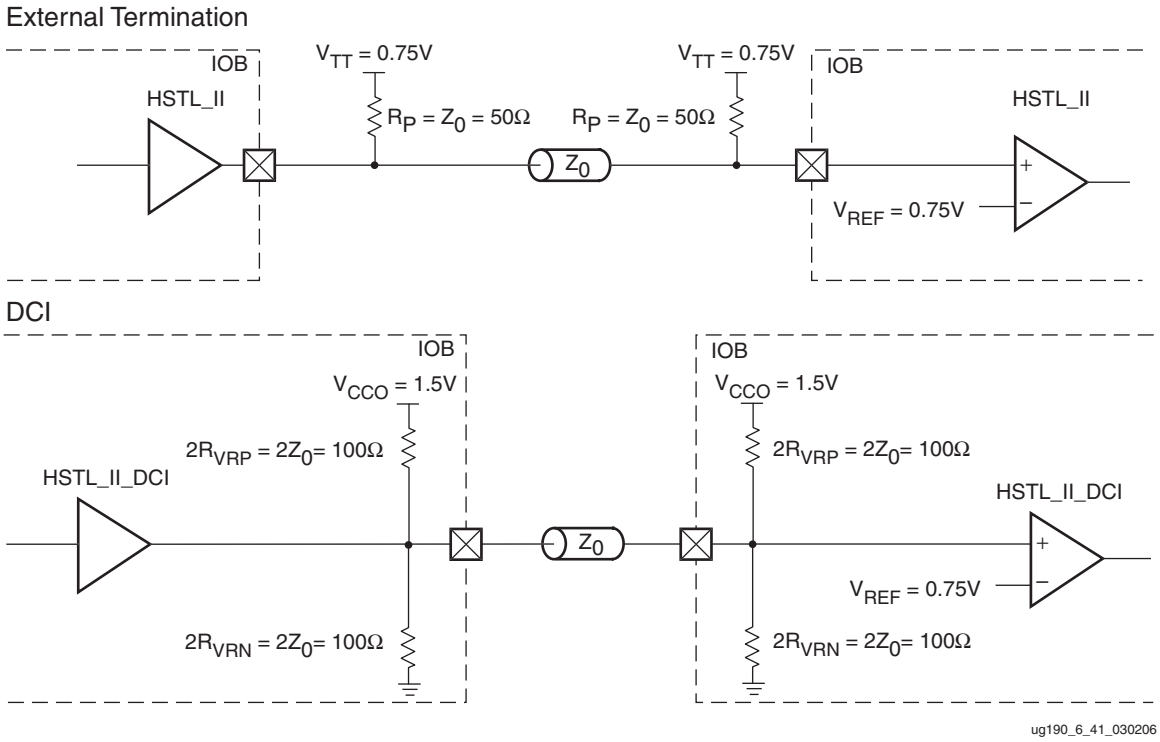


Figure 6-43: **HSTL (1.5V) Class II Unidirectional Termination**

Figure 6-44 shows a sample circuit illustrating a valid termination technique for HSTL Class II (1.5V) with bidirectional termination.

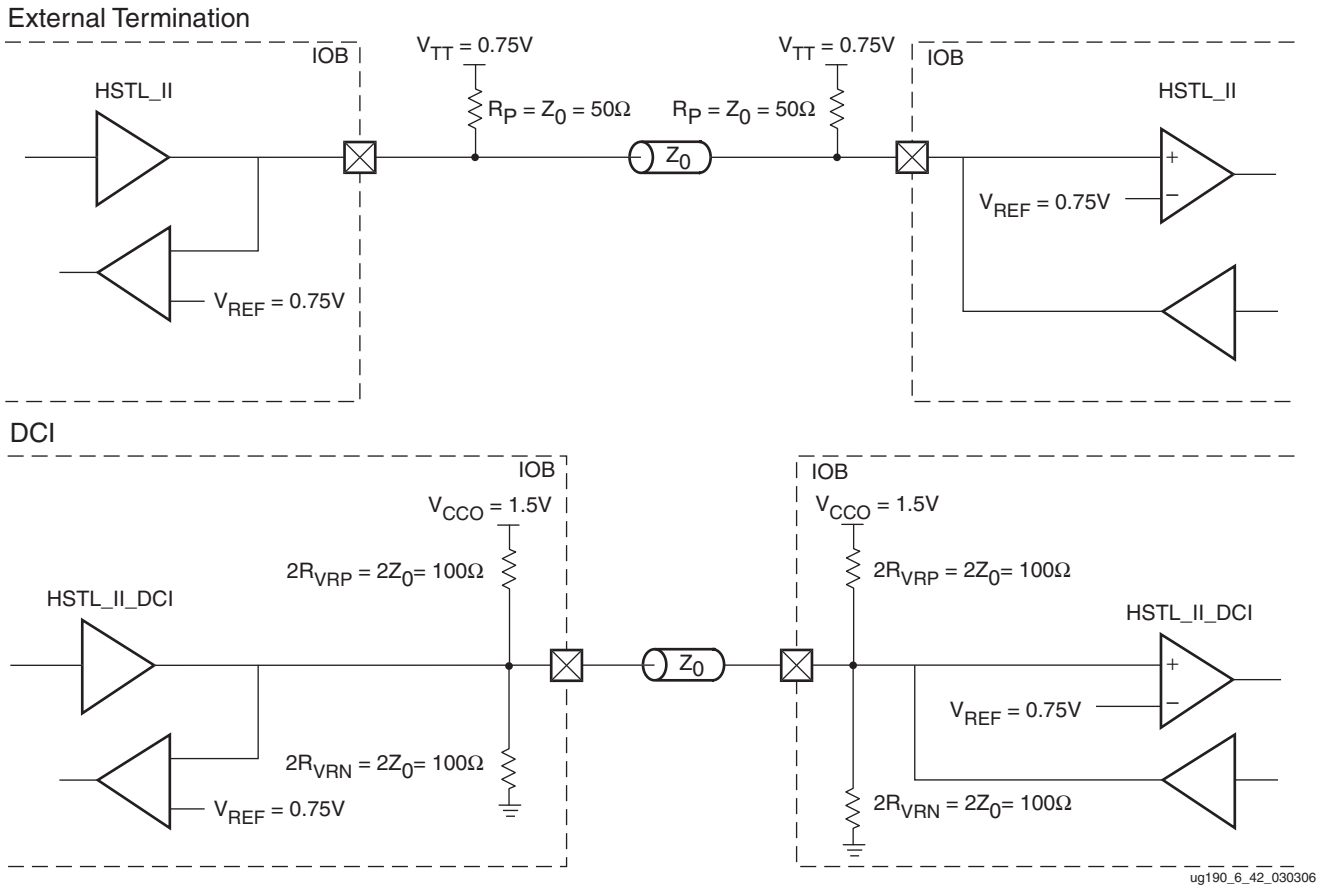


Figure 6-44: **HSTL (1.5V) Class II Bidirectional Termination**

Table 6-17 lists the HSTL (1.5V) Class II DC voltage specifications.

Table 6-17: **HSTL (1.5V) Class II DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	1.40	1.50	1.60
V_{REF} (2)	0.68	0.75	0.90
V_{TT}	–	$V_{CCO} \times 0.5$	–
V_{IH}	$V_{REF} + 0.1$	–	–
V_{IL}	–	–	$V_{REF} - 0.1$
V_{OH}	$V_{CCO} - 0.4$	–	–
V_{OL}	–	–	0.4
I_{OH} at V_{OH} (mA)(1)	–16	–	–
I_{OL} at V_{OL} (mA)(1) (3)	16	–	–

Notes:

1. V_{OL} and V_{OH} for lower drive currents are sample tested.
2. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."
3. HSTL_II_T_DCI has a weaker driver than HSTL_II_DCI.

Differential HSTL Class II

Figure 6-45 shows a sample circuit illustrating a valid termination technique for differential HSTL Class II (1.5V) with unidirectional termination.

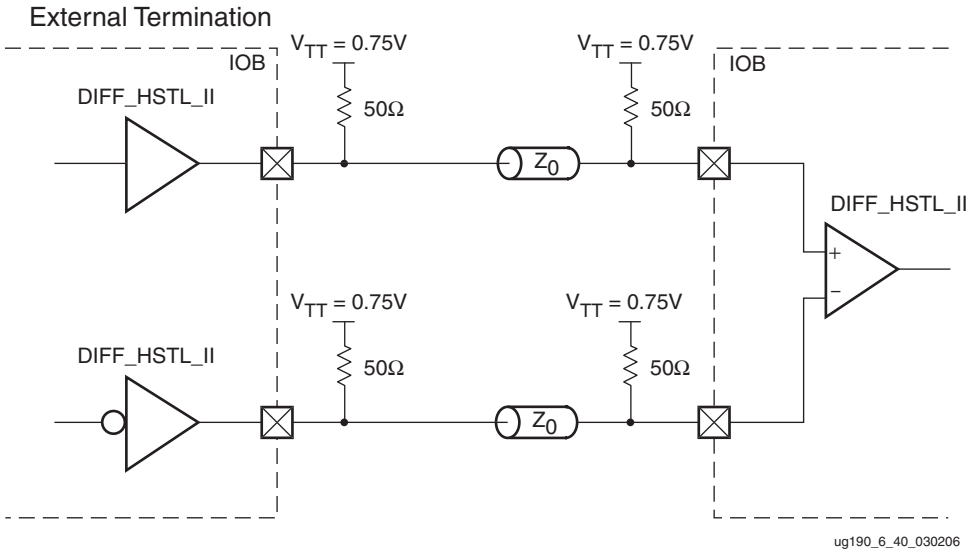


Figure 6-45: **Differential HSTL (1.5V) Class II Unidirectional Termination**

Figure 6-46 shows a sample circuit illustrating a valid termination technique for differential HSTL Class II (1.5V) with unidirectional DCI termination.

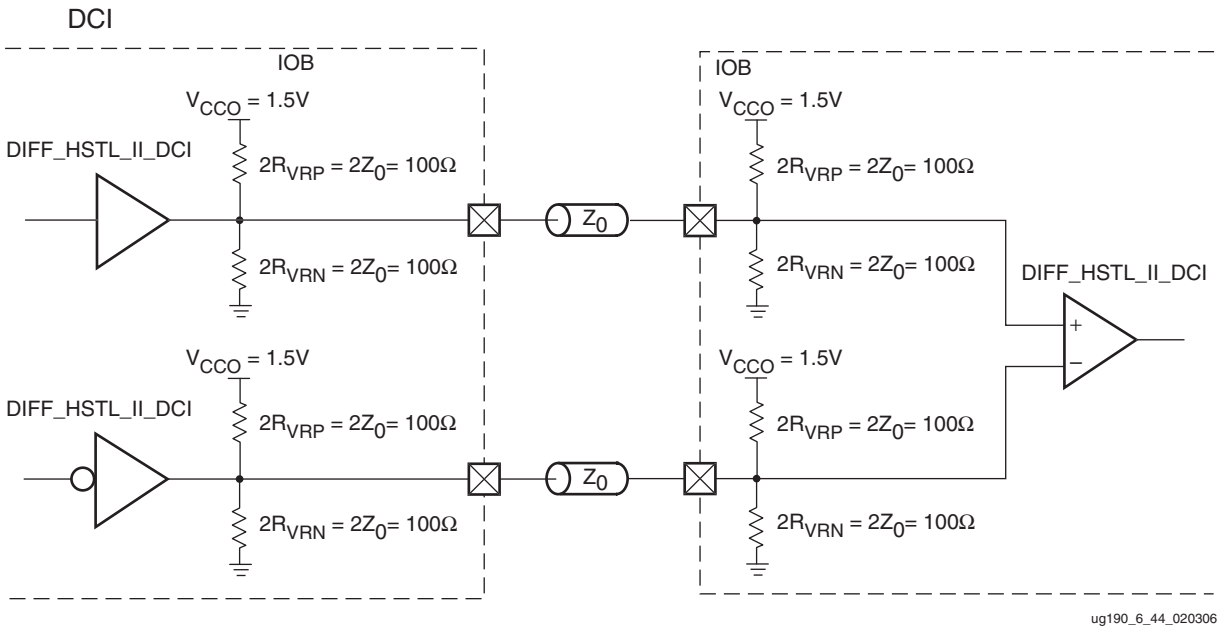


Figure 6-46: **Differential HSTL (1.5V) Class II DCI Unidirectional Termination**

Figure 6-47 shows a sample circuit illustrating a valid termination technique for differential HSTL Class II (1.5V) with bidirectional termination.

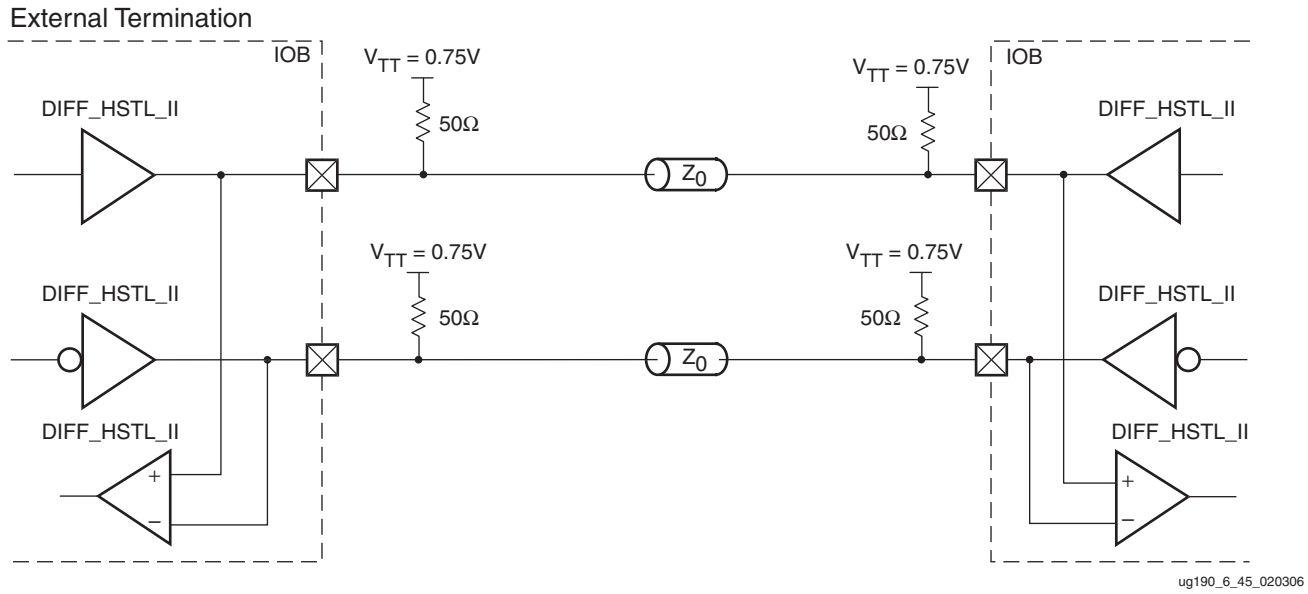


Figure 6-47: **Differential HSTL (1.5V) Class II Bidirectional Termination**

Figure 6-48 shows a sample circuit illustrating a valid termination technique for differential HSTL Class II (1.5V) with bidirectional DCI termination.

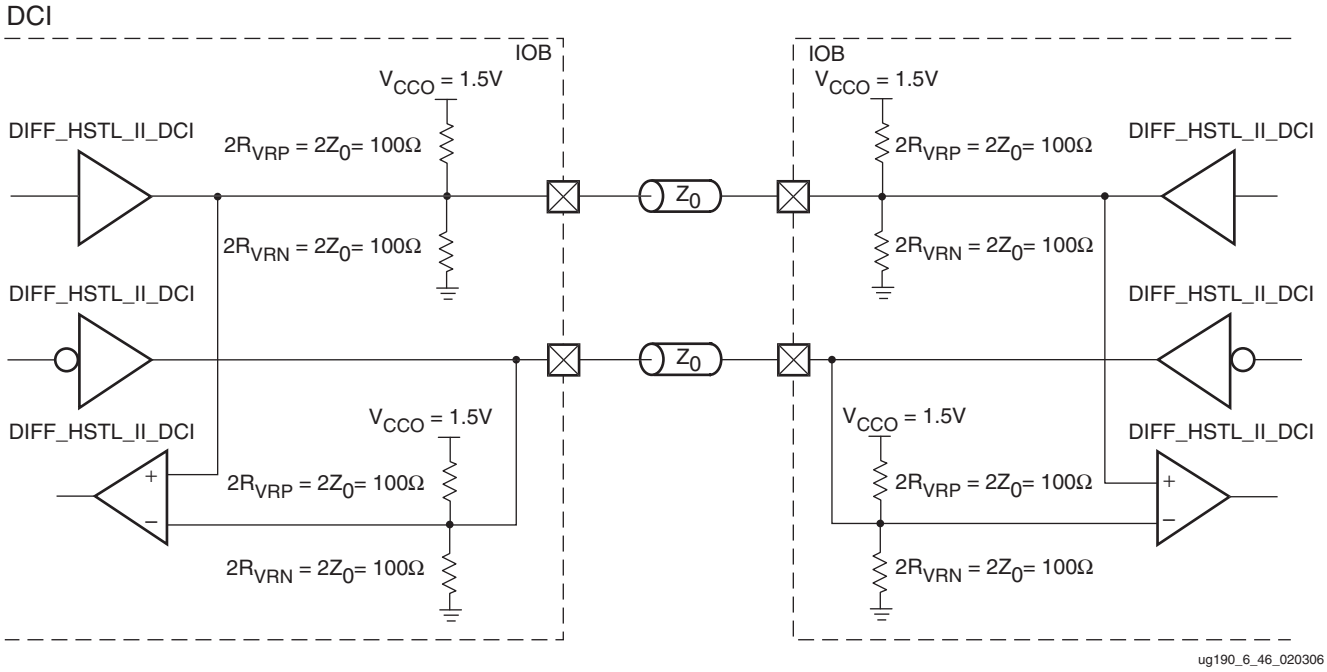


Figure 6-48: **Differential HSTL (1.5V) Class II DCI Bidirectional Termination**

Table 6-18 lists the differential HSTL Class II DC voltage specifications.

Table 6-18: **Differential HSTL Class II DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	1.40	1.50	1.60
V_{TT}	–	$V_{CCO} \times 0.5$	–
V_{IN} (DC)	–0.30	–	$V_{CCO} + 0.30$
V_{DIFF} (DC)	0.20	–	$V_{CCO} + 0.60$
V_{CM} (DC)(1)	0.68	–	0.90
V_{DIFF} (AC)	0.40	–	$V_{CCO} + 0.60$
V_X (Crossover)(2)	0.68	–	0.90

Notes:

1. Common mode voltage: $V_{CM} = V_P - ((V_P - V_N)/2)$
2. Crossover point: V_X where $V_P - V_N = 0$ (AC coupled)

HSTL Class III

Figure 6-49 shows a sample circuit illustrating a valid termination technique for HSTL Class III.

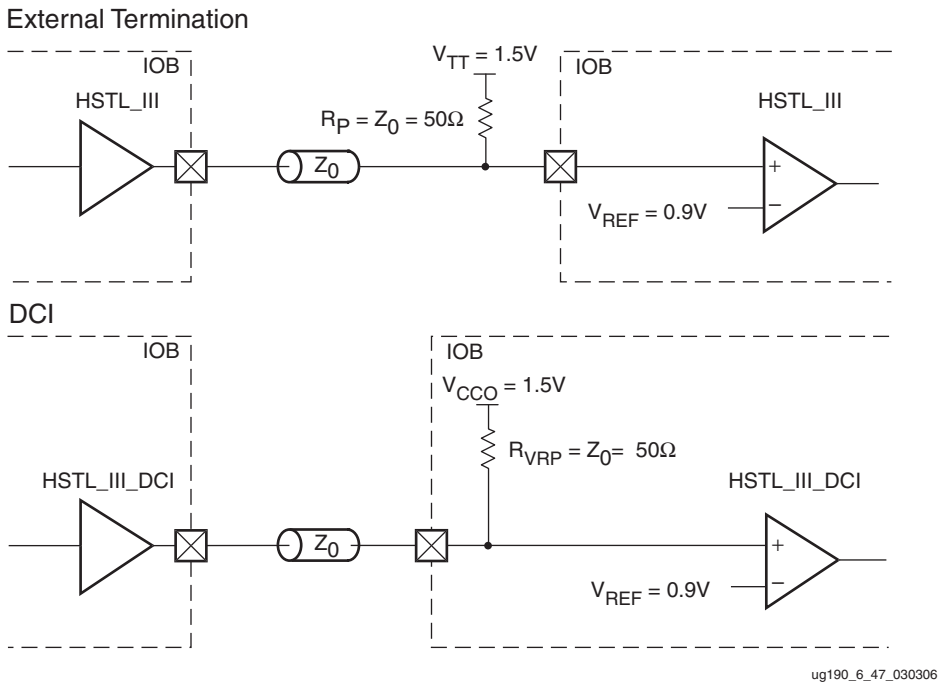


Figure 6-49: **HSTL Class III Termination**

Table 6-19 lists the HSTL Class III DC voltage specifications.

Table 6-19: **HSTL Class III DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	1.40	1.50	1.60
V_{REF} (2)	–	0.90	–
V_{TT}	–	V_{CCO}	–
V_{IH}	V_{REF} + 0.1	–	–
V_{IL}	–	–	V_{REF} – 0.1
V_{OH}	V_{CCO} – 0.4	–	–
V_{OL}	–	–	0.4
I_{OH} at V_{OH} (mA)(1)	–8	–	–
I_{OL} at V_{OL} (mA)(1)	24	–	–

Notes:

1. V_{OL} and V_{OH} for lower drive currents are sample tested.
2. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."

HSTL Class IV

Figure 6-50 shows a sample circuit illustrating a valid unidirectional termination technique for HSTL Class IV.

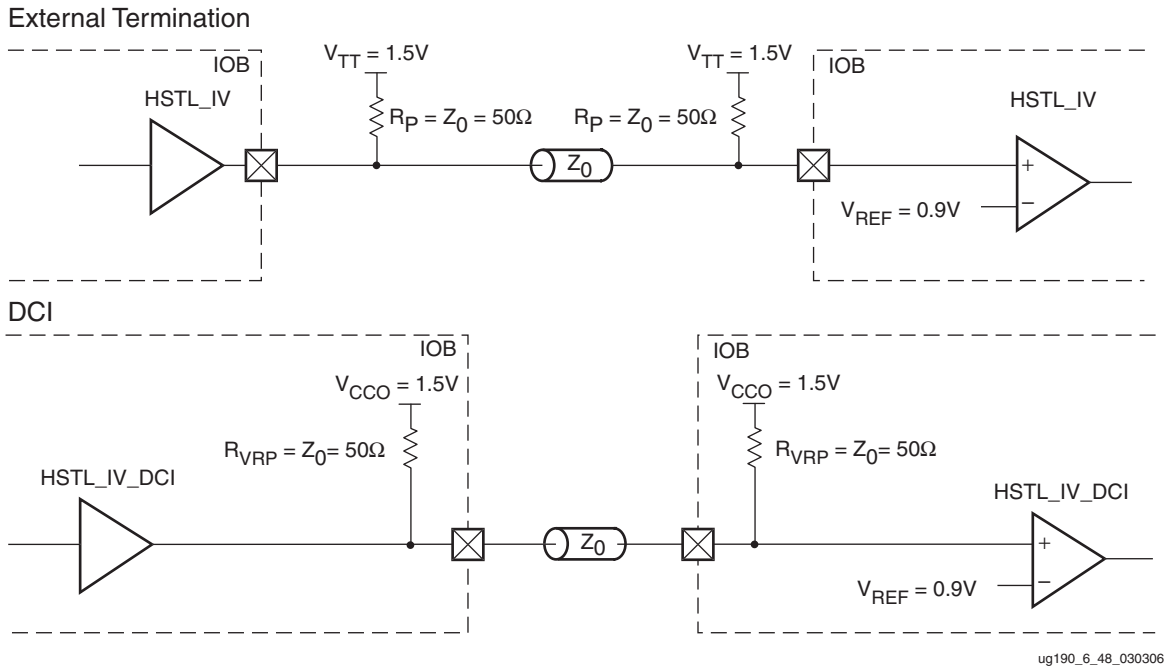


Figure 6-50: **HSTL Class IV Unidirectional Termination**

Figure 6-51 shows a sample circuit illustrating a valid bidirectional termination technique for HSTL Class IV.

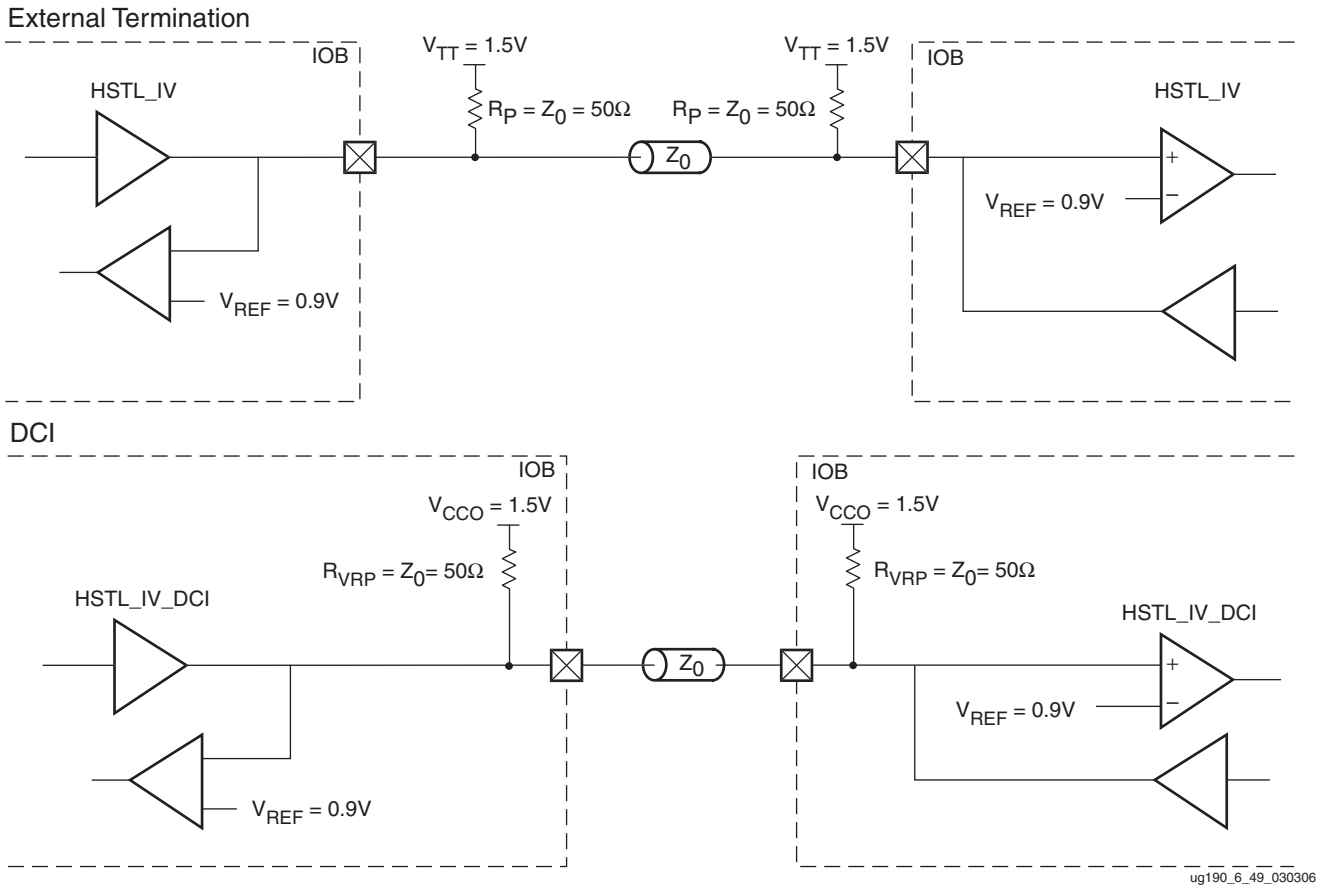


Figure 6-51: **HSTL Class IV Bidirectional Termination**

Table 6-20 lists the HSTL Class IV DC voltage specifications.

Table 6-20: **HSTL Class IV DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	1.40	1.50	1.60
V_{REF} (2)	–	0.90	–
V_{TT}	–	V_{CCO}	–
V_{IH}	V_{REF} + 0.1	–	–
V_{IL}	–	–	V_{REF} – 0.1
V_{OH}	V_{CCO} – 0.4	–	–
V_{OL}	–	–	0.4
I_{OH} at V_{OH} (mA)(1)	–8	–	–
I_{OL} at V_{OL} (mA)(1)	48	–	–

Notes:

1. V_{OL} and V_{OH} for lower drive currents are sample tested.
2. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."

HSTL_II_T_DCI (1.5V) Split-Thevenin Termination

Figure 6-52 shows a sample circuit illustrating a valid termination technique for HSTL_II_T_DCI (1.5V) with on-chip split-thevenin termination. In this bidirectional case, when 3-stated, the termination is invoked on the receiver and not on the driver.

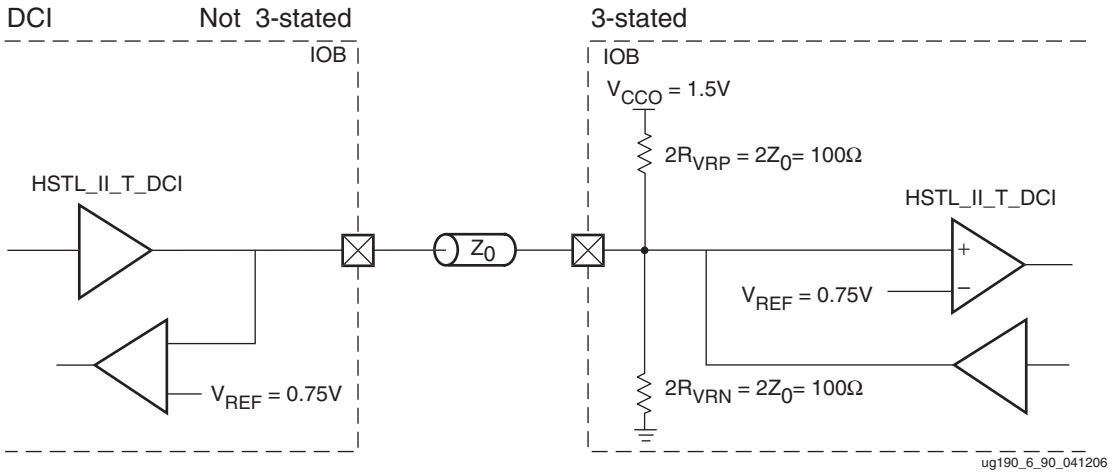


Figure 6-52: **HSTL_II_T_DCI (1.5V) Split-Thevenin Termination**

HSTL Class I (1.8V)

Figure 6-53 shows a sample circuit illustrating a valid termination technique for HSTL Class I (1.8V).

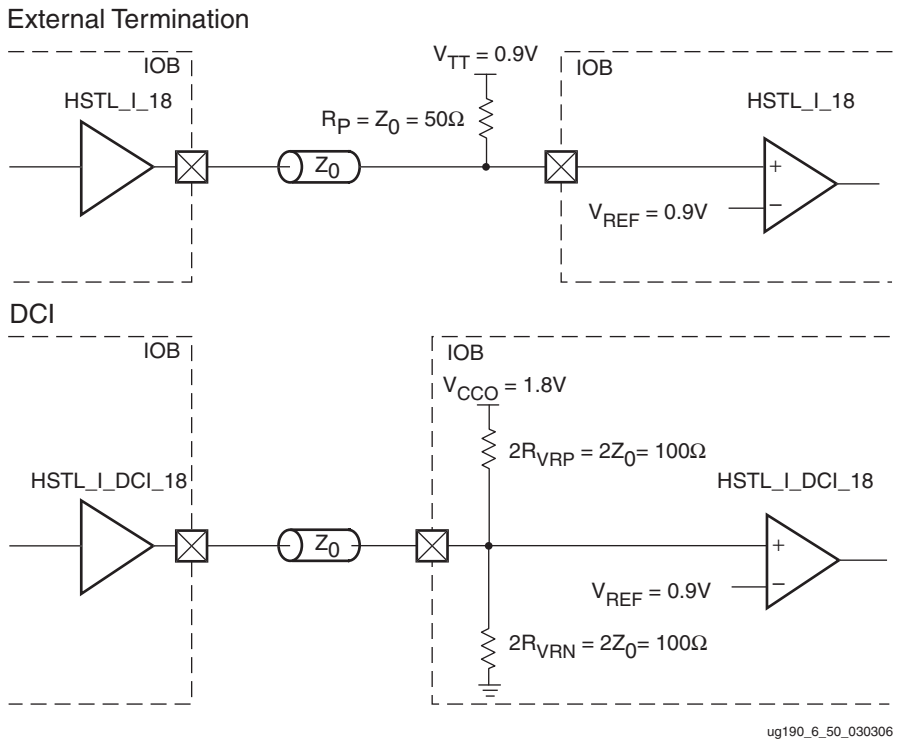


Figure 6-53: **HSTL Class I (1.8V) Termination**

Table 6-21 lists the HSTL Class I (1.8V) DC voltage specifications.

Table 6-21: **HSTL Class I (1.8V) DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	1.7	1.8	1.9
V_{REF} (2)	0.83	0.9	1.08
V_{TT}	–	$V_{CCO} \times 0.5$	–
V_{IH}	V_{REF} + 0.1	–	–
V_{IL}	–	–	V_{REF} – 0.1
V_{OH}	V_{CCO} – 0.4	–	–
V_{OL}	–	–	0.4
I_{OH} at V_{OH} (mA)(1)	–8	–	–
I_{OL} at V_{OL} (mA)(1)	8	–	–

Notes:

1. V_{OL} and V_{OH} for lower drive currents are sample tested.
2. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."

Differential HSTL Class I (1.8V)

Figure 6-54 shows a sample circuit illustrating a valid termination technique for differential HSTL Class I (1.8V) with unidirectional termination.

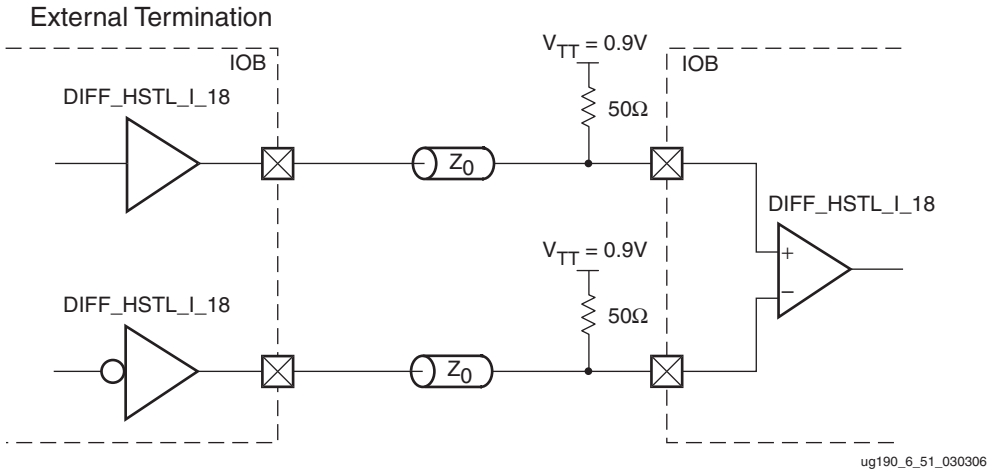


Figure 6-54: **Differential HSTL (1.8V) Class I Unidirectional Termination**

Figure 6-55 shows a sample circuit illustrating a valid termination technique for differential HSTL Class I (1.8V) with unidirectional DCI termination.

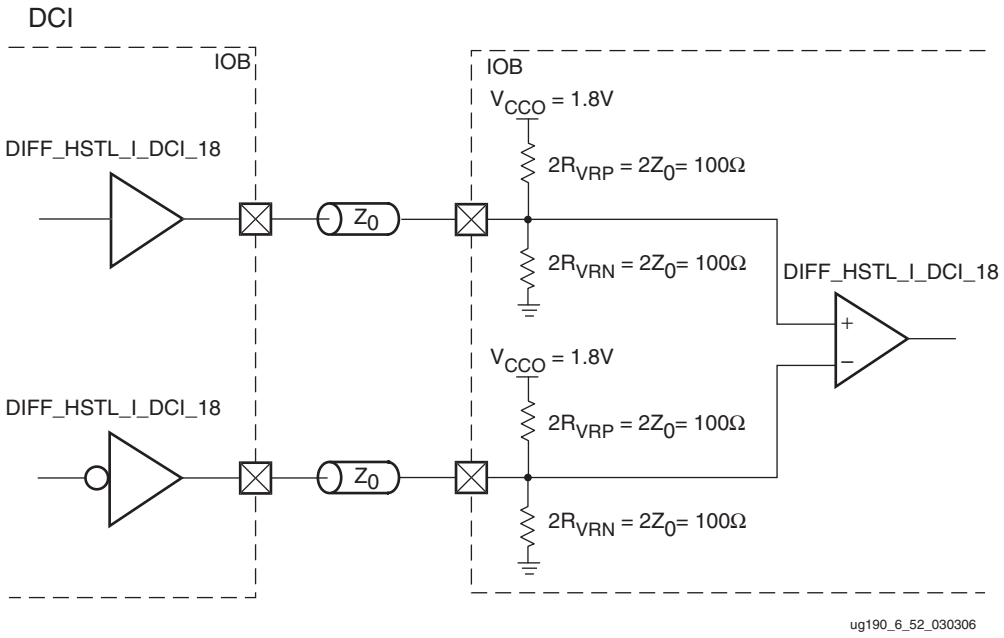


Figure 6-55: **Differential HSTL (1.8V) Class I DCI Unidirectional Termination**

Table 6-22 lists the differential HSTL Class I (1.8V) DC voltage specifications.

Table 6-22: **Differential HSTL Class I (1.8V) DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	1.7	1.8	1.9
V_{TT}	–	$V_{CCO} \times 0.5$	–
V_{IN} (DC)	–0.30	–	V_{CCO} + 0.30
V_{DIFF} (DC)	0.20	–	V_{CCO} + 0.60
V_{CM} (DC)(1)	0.83	–	1.08
V_{DIFF} (AC)	0.40	–	V_{CCO} + 0.60
V_X (Crossover)(2)	0.83	–	1.08

Notes:

1. Common mode voltage: $V_{CM} = V_P - ((V_P - V_N)/2)$
2. Crossover point: V_X where $V_P - V_N = 0$ (AC coupled)

HSTL Class II (1.8V)

Figure 6-56 shows a sample circuit illustrating a valid termination technique for HSTL Class II (1.8V) with unidirectional termination.

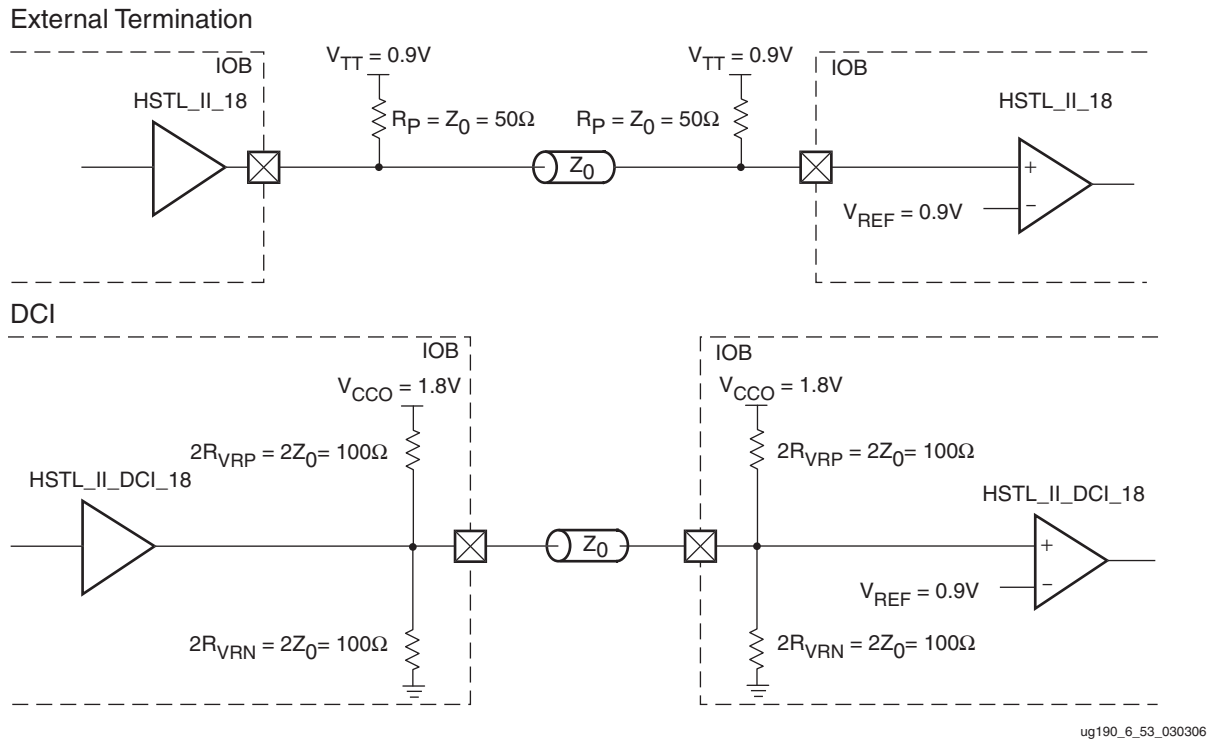


Figure 6-56: **HSTL Class II (1.8V) with Unidirectional Termination**

Figure 6-57 shows a sample circuit illustrating a valid termination technique for HSTL Class II (1.8V) with bidirectional termination.

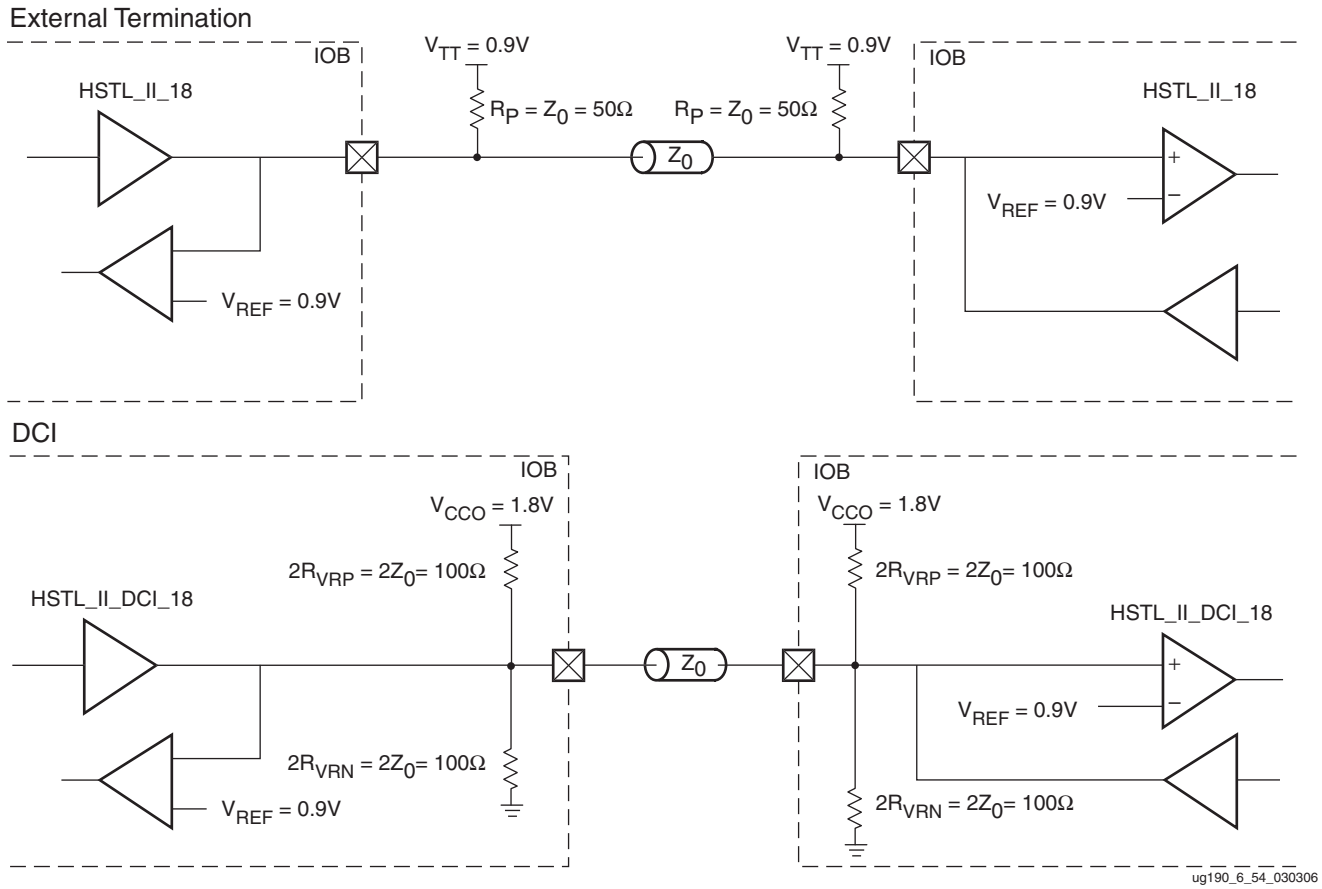


Figure 6-57: **HSTL Class II (1.8V) with Bidirectional Termination**

Table 6-23 lists the HSTL Class II (1.8V) DC voltage specifications.

Table 6-23: **HSTL Class II (1.8V) DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	1.7	1.8	1.9
V_{REF} (2)	–	0.9	–
V_{TT}	–	$V_{CCO} \times 0.5$	–
V_{IH}	$V_{REF} + 0.1$	–	–
V_{IL}	–	–	$V_{REF} - 0.1$
V_{OH}	$V_{CCO} - 0.4$	–	–
V_{OL}	–	–	0.4
I_{OH} at V_{OH} (mA)(1)	–16	–	–
I_{OL} at V_{OL} (mA)(1)	16	–	–

Notes:

1. V_{OL} and V_{OH} for lower drive currents are sample tested.
2. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."

Differential HSTL Class II (1.8V)

Figure 6-58 shows a sample circuit illustrating a valid termination technique for differential HSTL Class II (1.8V) with unidirectional termination.

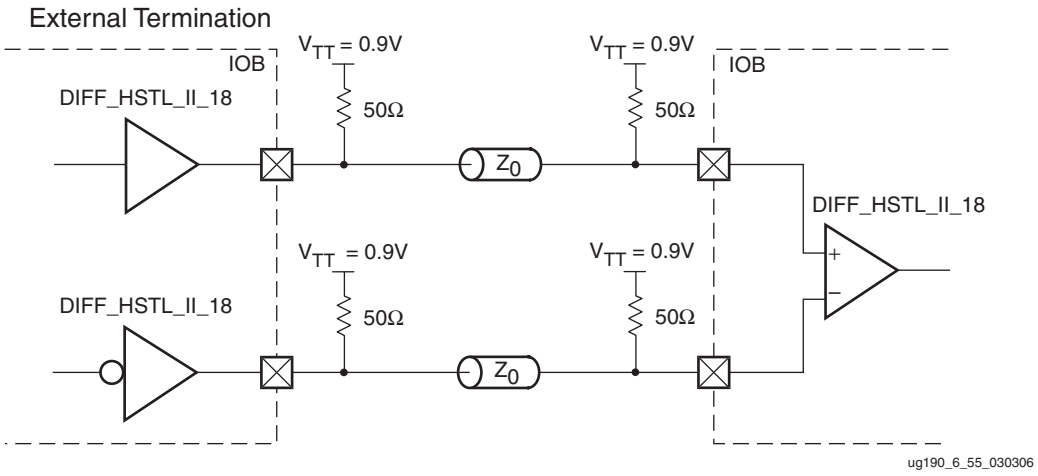


Figure 6-58: **Differential HSTL (1.8V) Class II Unidirectional Termination**

Figure 6-59 shows a sample circuit illustrating a valid termination technique for differential HSTL Class II (1.8V) with unidirectional DCI termination.

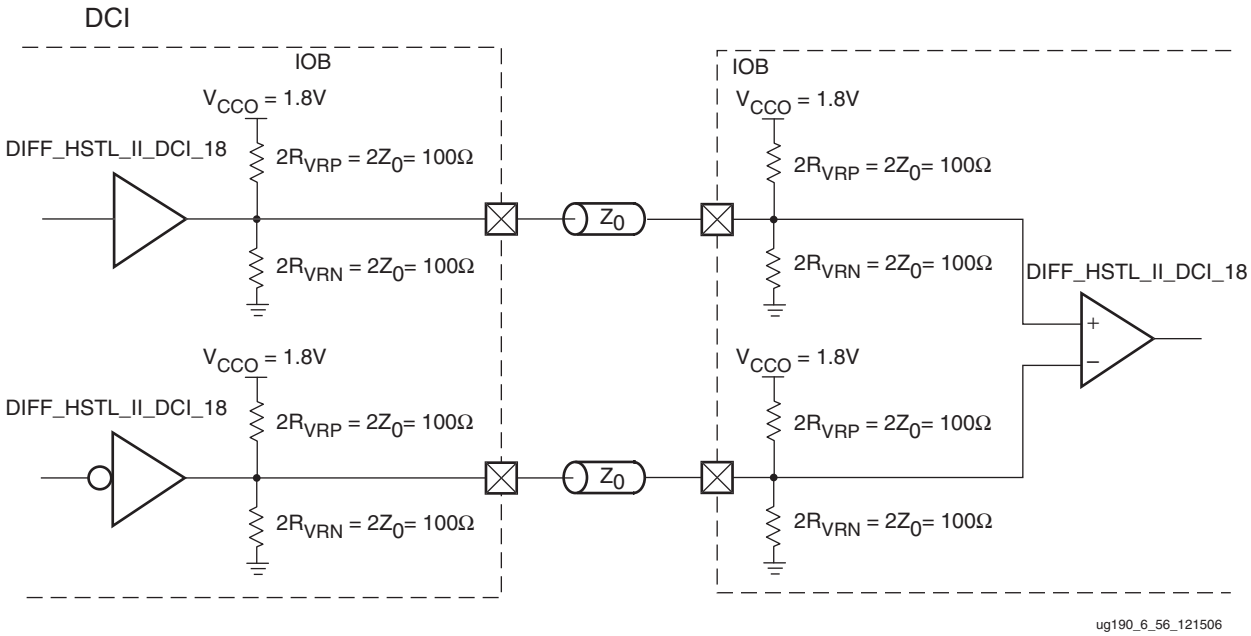


Figure 6-59: **Differential HSTL (1.8V) Class II DCI Unidirectional Termination**

Figure 6-60 shows a sample circuit illustrating a valid termination technique for differential HSTL Class II (1.8V) with bidirectional termination.

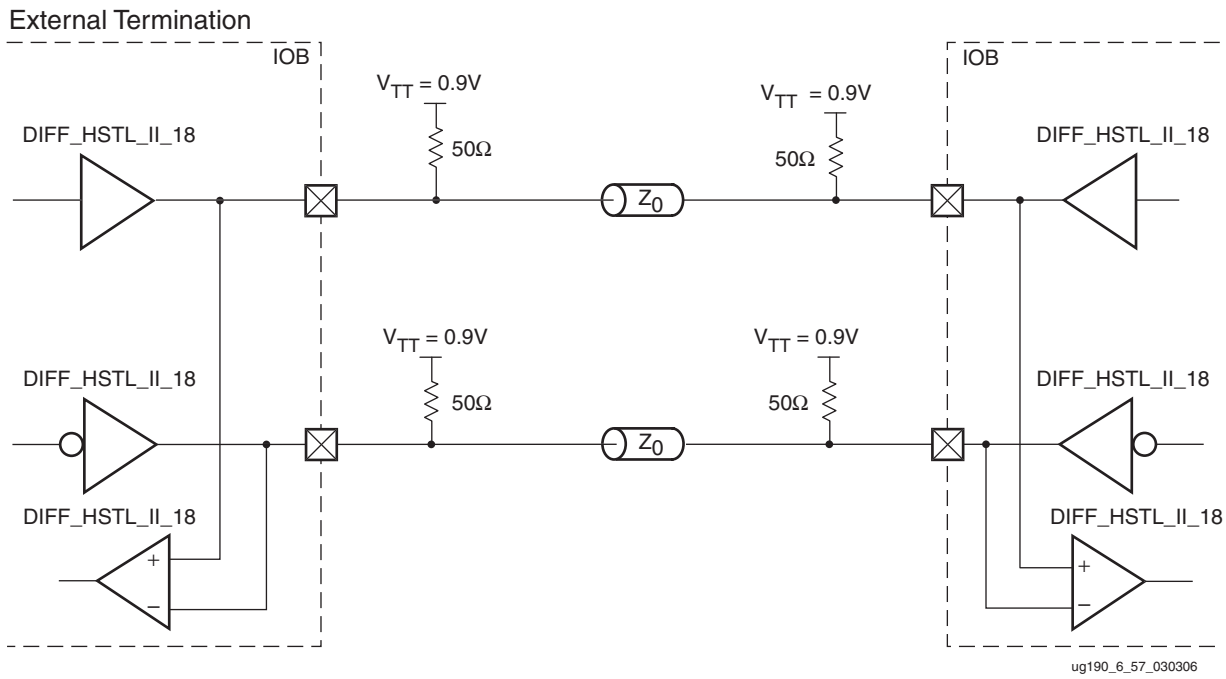


Figure 6-60: **Differential HSTL (1.8V) Class II Bidirectional Termination**

Figure 6-61 shows a sample circuit illustrating a valid termination technique for differential HSTL Class II (1.8V) with bidirectional DCI termination.

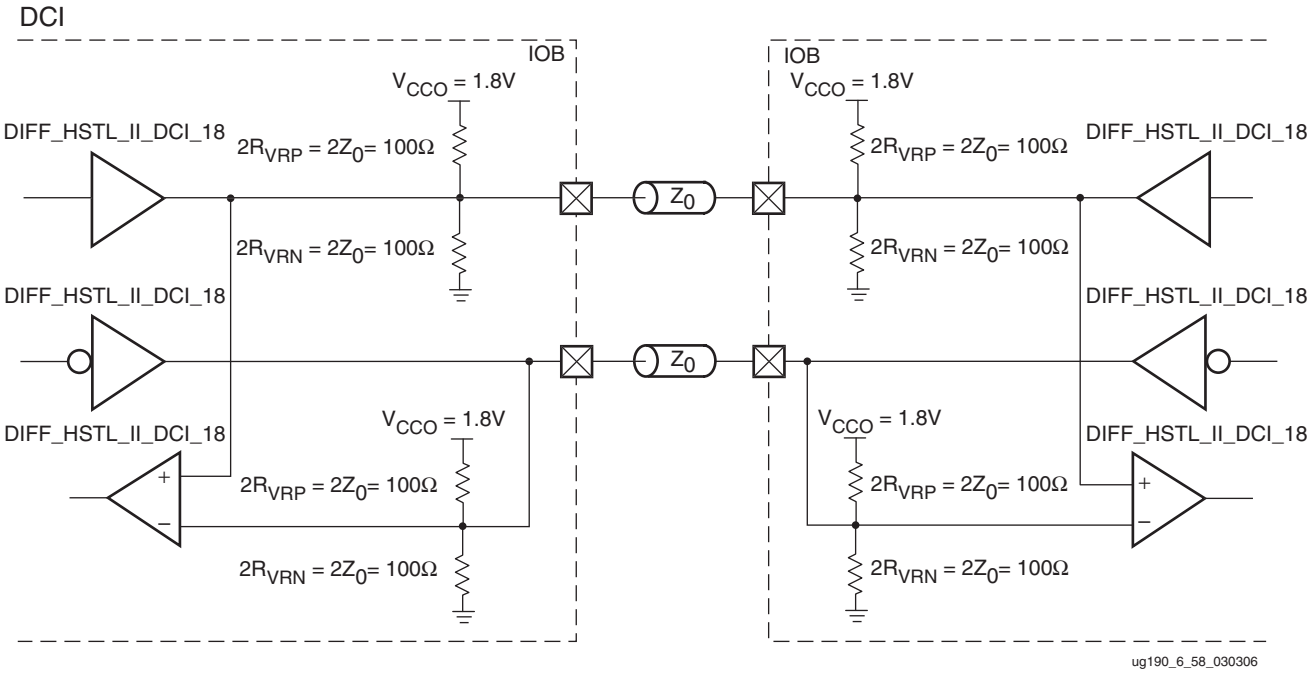


Figure 6-61: **Differential HSTL (1.8V) Class II DCI Bidirectional Termination**

Table 6-24 lists the differential HSTL Class II (1.8V) DC voltage specifications.

Table 6-24: **Differential HSTL Class II (1.8V) DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	1.7	1.8	1.9
V_{TT}	–	$V_{CCO} \times 0.5$	–
V_{IN} (DC)	–0.30	–	$V_{CCO} + 0.30$
V_{DIFF} (DC)	0.20	–	$V_{CCO} + 0.60$
V_{CM} (DC)(1)	0.83	–	1.08
V_{DIFF} (AC)	0.40	–	$V_{CCO} + 0.60$
V_X (Crossover)(2)	0.83	–	1.08

Notes:

1. Common mode voltage: $V_{CM} = V_P - ((V_P - V_N)/2)$
2. Crossover point: V_X where $V_P - V_N = 0$ (AC coupled)

HSTL Class III (1.8V)

Figure 6-62 shows a sample circuit illustrating a valid termination technique for HSTL Class III (1.8V).

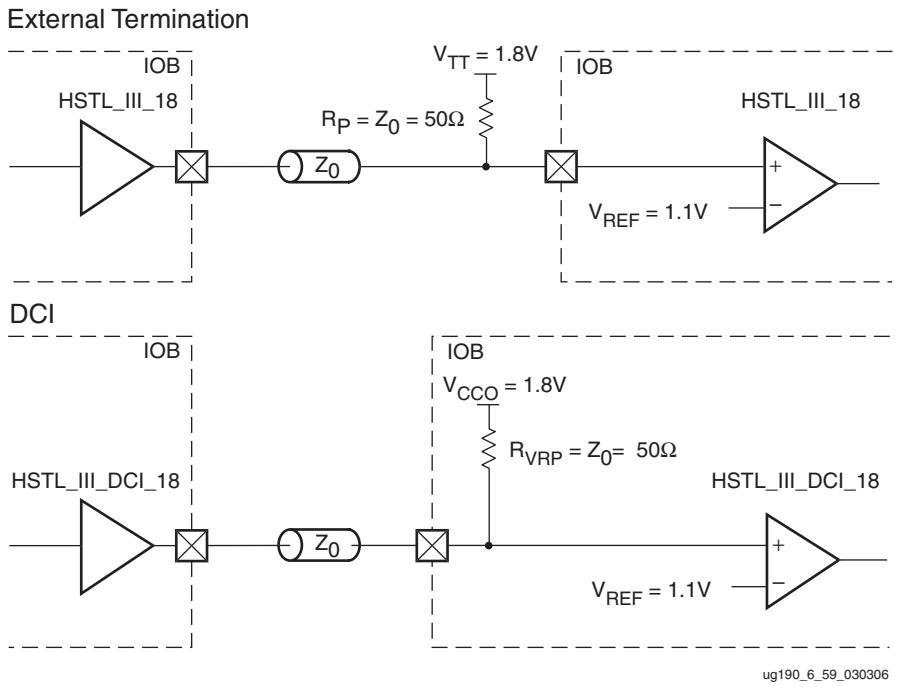


Figure 6-62: **HSTL Class III (1.8V) Termination**

Table 6-25 lists the HSTL Class III (1.8V) DC voltage specifications.

Table 6-25: **HSTL Class III (1.8V) DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	1.7	1.8	1.9
V_{REF} (2)	–	1.1	–
V_{TT}	–	V_{CCO}	–
V_{IH}	V_{REF} + 0.1	–	–
V_{IL}	–	–	V_{REF} – 0.1
V_{OH}	V_{CCO} – 0.4	–	–
V_{OL}	–	–	0.4
I_{OH} at V_{OH} (mA)(1)	–8	–	–
I_{OL} at V_{OL} (mA)(1)	24	–	–

Notes:

1. V_{OL} and V_{OH} for lower drive currents are sample tested.
2. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."

HSTL Class IV (1.8V)

Figure 6-63 shows a sample circuit illustrating a valid unidirectional termination technique for HSTL Class IV (1.8V).

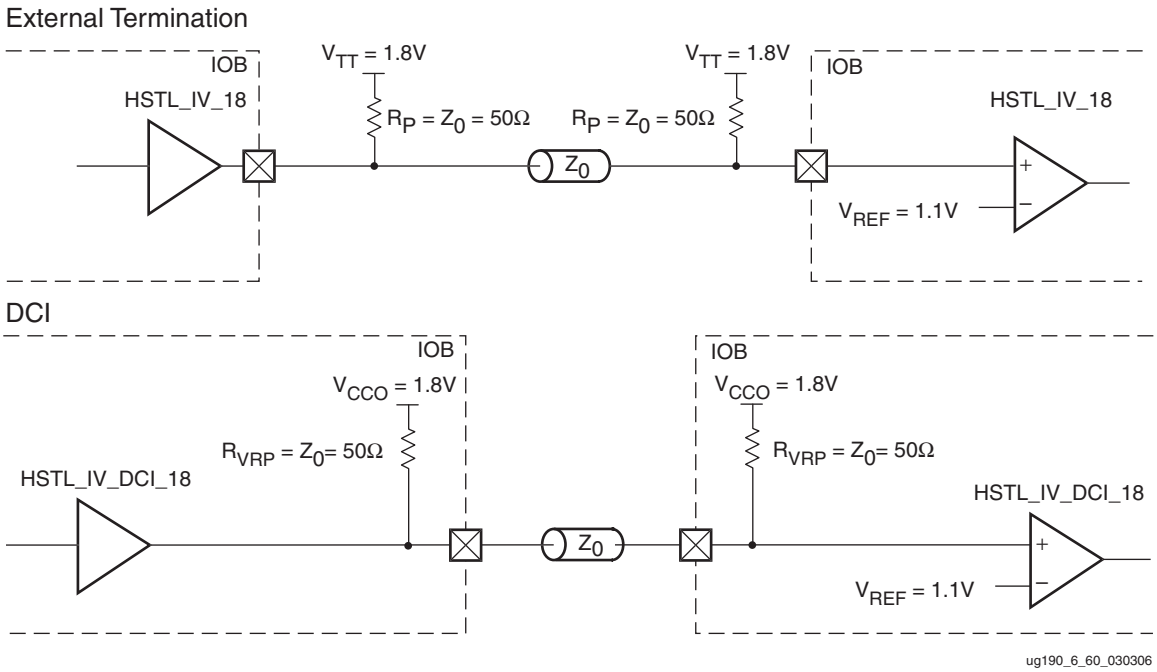


Figure 6-63: **HSTL Class IV (1.8V) with Unidirectional Termination**

Figure 6-64 shows a sample circuit illustrating a valid bidirectional termination technique for HSTL Class IV (1.8V).

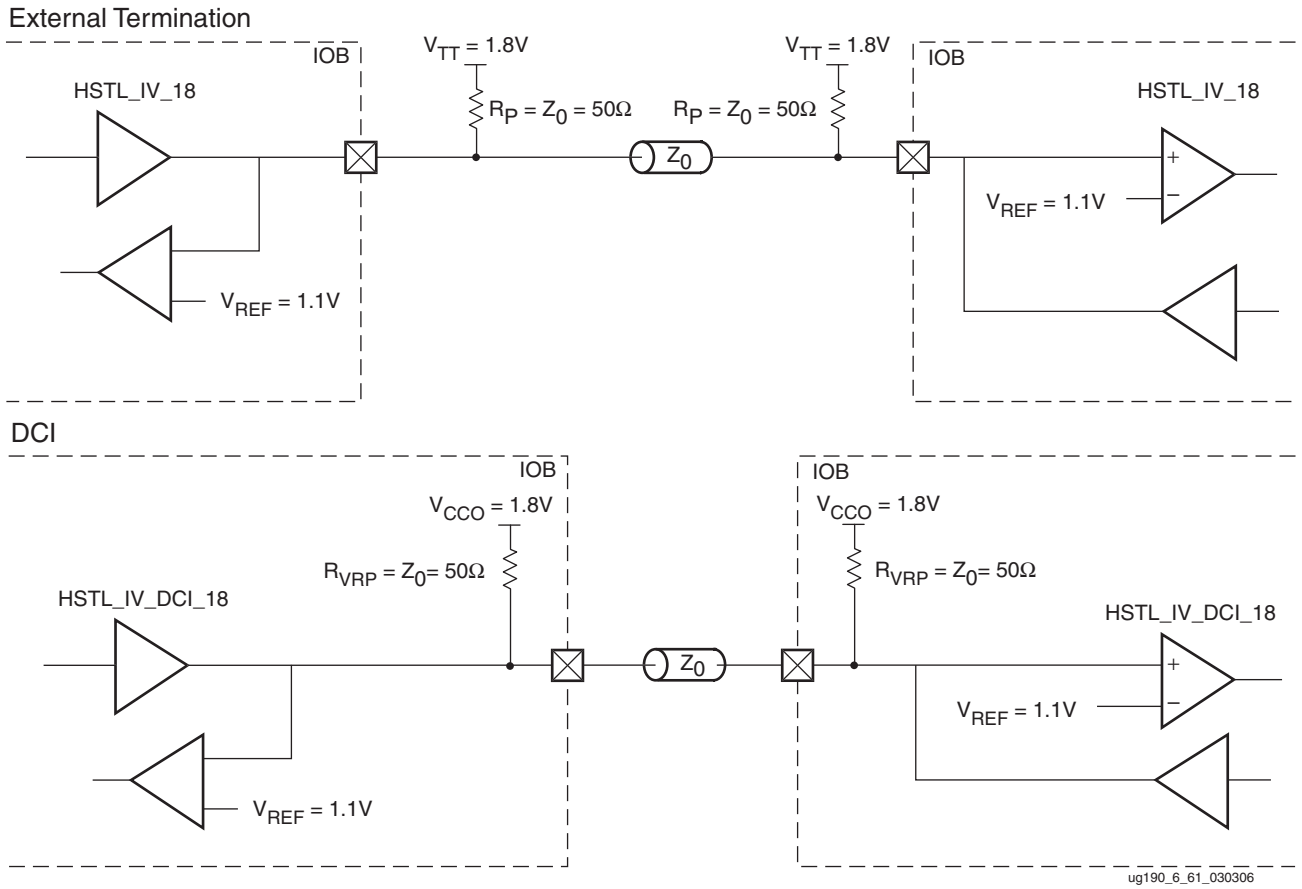


Figure 6-64: **HSTL Class IV (1.8V) with Bidirectional Termination**

Table 6-26 lists the HSTL Class IV (1.8V) DC voltage specifications.

Table 6-26: **HSTL Class IV (1.8V) DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	1.7	1.8	1.9
V_{REF} (2)	–	1.1	–
V_{TT}	–	V_{CCO}	–
V_{IH}	V_{REF} + 0.1	–	–
V_{IL}	–	–	V_{REF} – 0.1
V_{OH}	V_{CCO} – 0.4	–	–
V_{OL}	–	–	0.4
I_{OH} at V_{OH} (mA)(1)	–8	–	–
I_{OL} at V_{OL} (mA)(1)	48	–	–

Notes:

1. V_{OL} and V_{OH} for lower drive currents are sample tested.
2. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."

HSTL_II_T_DCI_18 (1.8V) Split-Thevenin Termination

Figure 6-65 shows a sample circuit illustrating a valid termination technique for HSTL_II_T_DCI_18 (1.8V) with on-chip split-thevenin termination. In this bidirectional case, when 3-stated, the termination is invoked on the receiver and not on the driver.

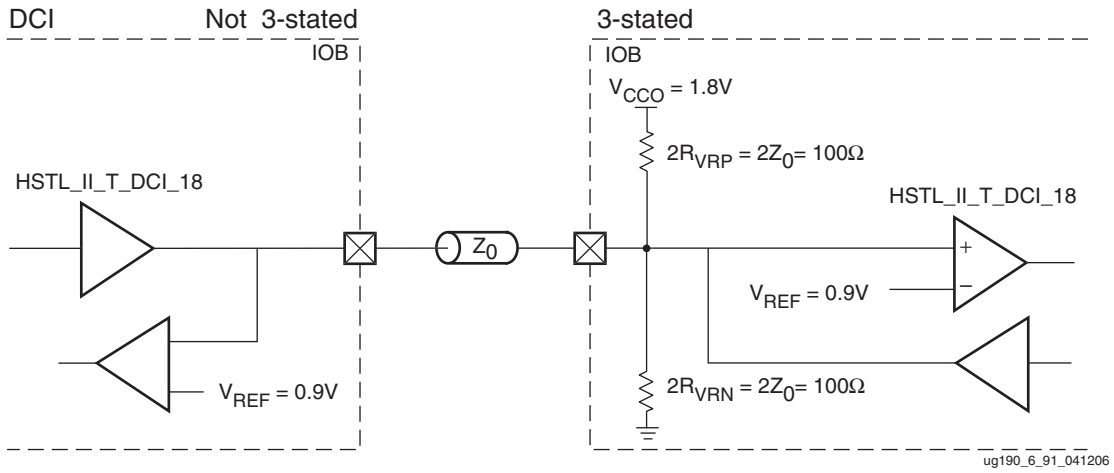


Figure 6-65: **HSTL_II_T_DCI_18 Split-Thevenin Termination**

HSTL Class I (1.2V)

Figure 6-66 shows a sample circuit illustrating a valid termination technique for HSTL Class I (1.2V). It is used for unidirectional links.

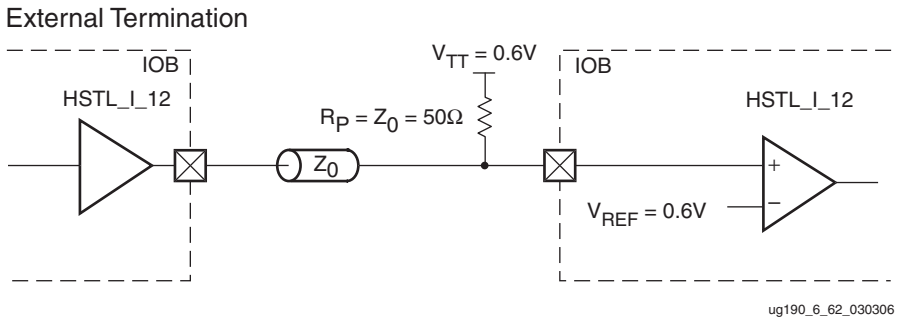


Figure 6-66: **HSTL Class I (1.2V) Termination**

Table 6-21 lists the HSTL Class I (1.2V) DC voltage specifications.

Table 6-27: **HSTL Class I (1.2V) DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	1.14	1.2	1.26
V_{REF} (2)	V_{CCO} × 0.48	0.6	V_{CCO} × 0.52
V_{TT}	–	V_{CCO} × 0.5	–
V_{IH}	V_{REF} + 0.08	–	–
V_{IL}	–	–	V_{REF} – 0.08
V_{OH}	V_{CCO} – .0.315	–	–
V_{OL}	–	–	0.315
I_{OH} at V_{OH} (mA)(1)	–6.3	–	–
I_{OL} at V_{OL} (mA)(1)	6.3	–	–

Notes:

1. V_{OL} and V_{OH} for lower drive currents are sample tested.
2. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."

SSTL (Stub-Series Terminated Logic)

The Stub-Series Terminated Logic (SSTL) for 2.5V (SSTL2) and 1.8V (SSTL18) standards are for general purpose memory buses. SSTL2 is defined by the JEDEC standard JESD8-9B and SSTL18 is defined by the JEDEC standard JESD8-15. The SSTL2 standard has two classes; Class I is for unidirectional and class II is for bidirectional signaling. Virtex-5 FPGA I/O supports both standards for single-ended signaling and differential signaling. This standard requires a differential amplifier input buffer and a push-pull output buffer.

SSTL2_I, SSTL18_I

Class I signaling uses V_{TT} ($V_{CCO}/2$) as a parallel termination voltage to a 50 Ω resistor at the receiver. A series resistor (25 Ω at 2.5V, 20 Ω at 1.8V) must be connected to the transmitter output.

SSTL2_I_DCI, SSTL18_I_DCI

The DCI transmitter provides the internal series resistance (25 Ω at 2.5V, 20 Ω at 1.8V). The DCI receiver has an internal split thevenin termination powered from V_{CCO} creating an equivalent V_{TT} voltage and termination impedance.

SSTL2_II, SSTL18_II

Class II signaling uses V_{TT} ($V_{CCO}/2$) as a parallel termination voltage to a 50 Ω resistor at the receiver and transmitter respectively. A series resistor (25 Ω at 2.5V, 20 Ω at 1.8V) must be connected to the transmitter output for a unidirectional link. For a bidirectional link, 25 Ω series resistors must connected the transmitters of the transceivers.

SSTL2_II_DCI, SSTL18_II_DCI

The DCI circuits have a split thevenin termination powered from V_{CCO} and an internal series resistor (25 Ω at 2.5V, 20 Ω at 1.8V). For a unidirectional link the internal series resistance is supplied only for the transmitter. A bidirectional link has the internal series resistor for both transmitters.

DIFF_SSTL2_I, DIFF_SSTL18_I

Differential SSTL 2.5V and 1.8V Class I pairs complementary single-ended SSTL_I type drivers with a differential receiver.

DIFF_SSTL2_I_DCI, DIFF_SSTL18_I_DCI

Differential SSTL 2.5V and 1.8V Class I pairs complementary single-ended SSTL_II type drivers with a differential receiver, including on-chip differential split thevenin termination.

DIFF_SSTL2_II, DIFF_SSTL18_II

Differential SSTL 2.5V and 1.8V Class II pairs complementary single-ended SSTL_II type drivers with a differential receiver. For a bidirectional link, a series resistor must be connected to both transmitters.

DIFF_SSTL2_II_DCI, DIFF_SSTL18_II_DCI

Differential SSTL 2.5V and 1.8V Class II pairs complementary single-ended SSTL_II type drivers with a differential receiver, including on-chip differential termination. DCI can be used for unidirectional and bidirectional links.

SSTL2_II_T_DCI, SSTL18_II_T_DCI

SSTL2_II_T_DCI and SSTL18_II_T_DCI provide on-chip split thevenin termination powered from V_{CCO} that creates an equivalent termination voltage of $V_{CCO}/2$ when these standards are 3-stated. When not 3-stated, these two standards do not have parallel termination but when invoked they have an internal series resistor (25 Ω at 2.5V and 20 Ω at 1.8V.)

SSTL2 Class I (2.5V)

Figure 6-67 shows a sample circuit illustrating a valid termination technique for SSTL2 Class I.

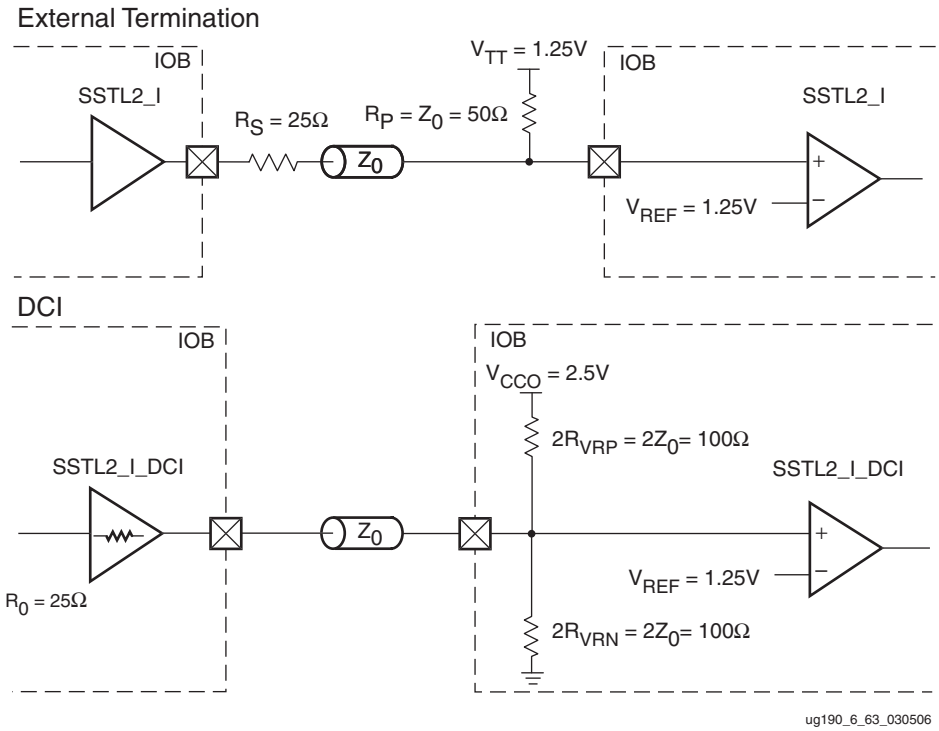


Figure 6-67: **SSTL2 Class I Termination**

Table 6-28 lists the SSTL2 DC voltage specifications for Class I.

Table 6-28: **SSTL2 DC Voltage Specifications Class I**

	Min	Typ	Max
V_{CCO}	2.3	2.5	2.7
$V_{REF} = 0.5 \times V_{CCO}$	1.13	1.25	1.38
$V_{TT} = V_{REF} + N^{(1)}$	1.09	1.25	1.42
$V_{IH} \geq V_{REF} + 0.15$	1.28	1.4	$V_{CCO} + 0.3^{(2)}$
$V_{IL} \leq V_{REF} - 0.15$	$-0.3^{(3)}$	1.1	1.23
$V_{OH} \geq V_{REF} + 0.61$	1.74	1.84	1.94
$V_{OL} \leq V_{REF} - 0.61^{(4)}$	0.56	0.66	0.76
I_{OH} at V_{OH} (mA)	–8.1	–	–
I_{OL} at V_{OL} (mA)	8.1	–	–

Notes:

1. N must be greater than or equal to –0.04 and less than or equal to 0.04.
2. V_{IH} maximum is V_{CCO} +0.3.
3. V_{IL} minimum does not conform to the formula.
4. Because SSTL2_I_DCI uses a controlled-impedance driver, V_{OH} and V_{OL} are different.

Differential SSTL2 Class I (2.5V)

Figure 6-68 shows a sample circuit illustrating a valid termination technique for differential SSTL2 Class I (2.5V) with unidirectional termination.

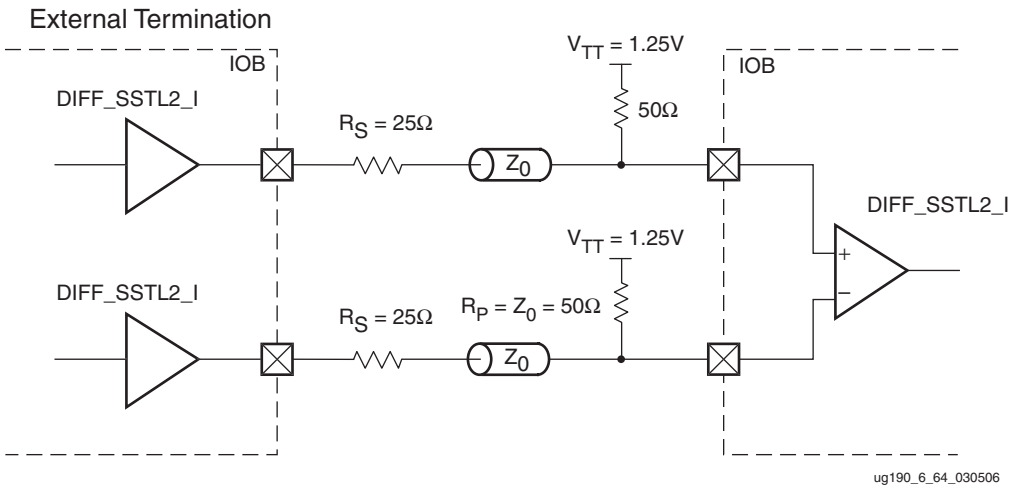


Figure 6-68: **Differential SSTL2 Class I Unidirectional Termination**

Figure 6-69 shows a sample circuit illustrating a valid termination technique for differential SSTL2 Class I (2.5V) with unidirectional DCI termination.

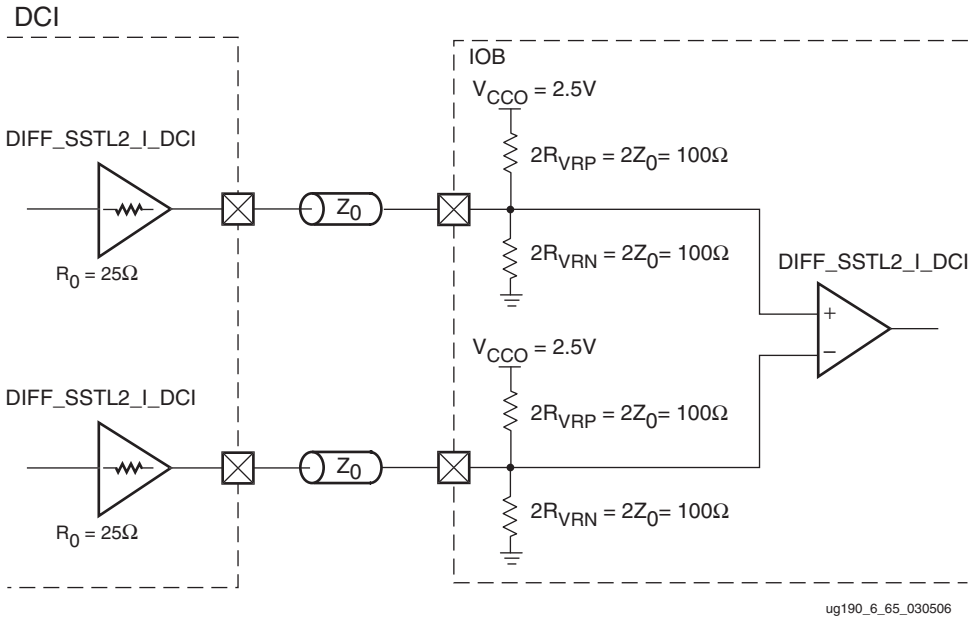


Figure 6-69: **Differential SSTL2 (2.5V) Class I Unidirectional DCI Termination**

Table 6-29 lists the differential SSTL2 Class I DC voltage specifications.

Table 6-29: **Differential SSTL2 Class I DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	2.3	2.5	2.7
Input Parameters			
V_{TT}	–	$V_{CCO} \times 0.5$	–
V_{IN} (DC)(1)	–0.30	–	V_{CCO} + 0.30
V_{ID} (DC)(2)	0.3	–	V_{CCO} + 0.60
V_{ID} (AC)	0.62	–	V_{CCO} + 0.60
V_{IX} (AC)(3)	0.95	–	1.55
Output Parameters			
V_{OX} (AC)(4)	1.0	–	1.5

Notes:

1. V_{IN} (DC) specifies the allowable DC excursion of each differential input.
2. V_{ID} (DC) specifies the input differential voltage required for switching.
3. V_{IX} (AC) indicates the voltage where the differential input signals must cross.
4. V_{OX} (AC) indicates the voltage where the differential output signals must cross.

SSTL2 Class II (2.5V)

Figure 6-70 shows a sample circuit illustrating a valid unidirectional termination technique for SSTL2 Class II.

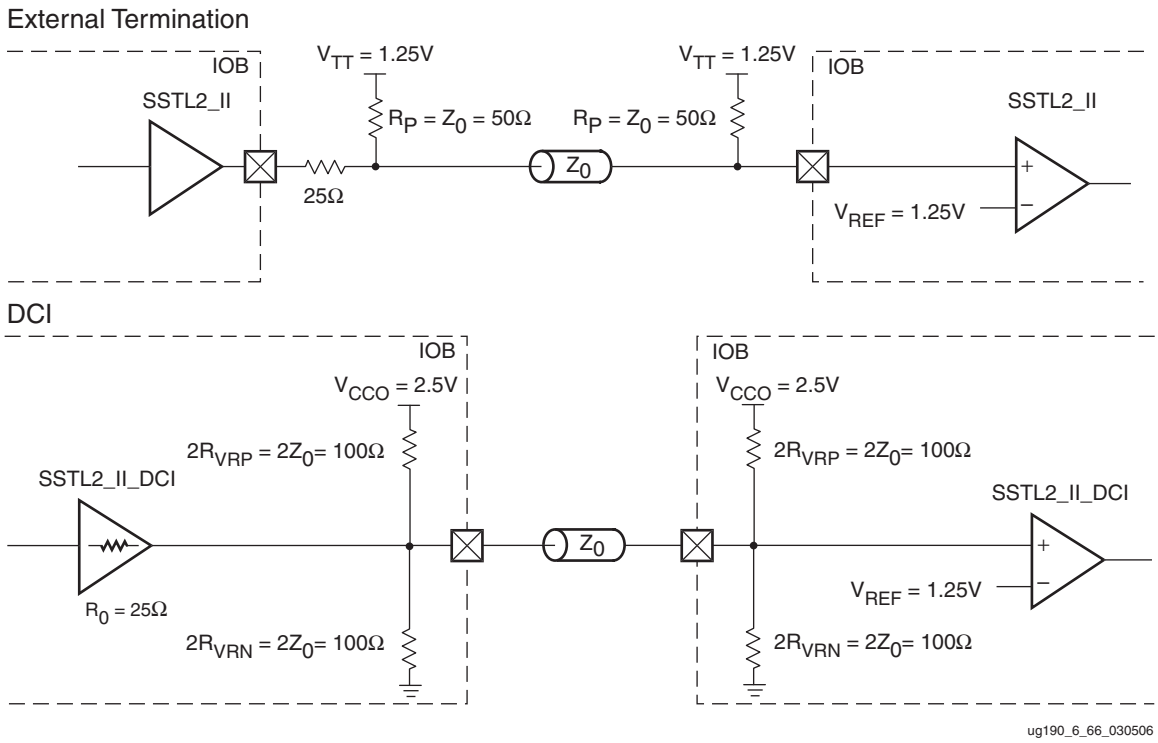


Figure 6-70: **SSTL2 Class II with Unidirectional Termination**

Figure 6-71 shows a sample circuit illustrating a valid bidirectional termination technique for SSTL2 Class II.

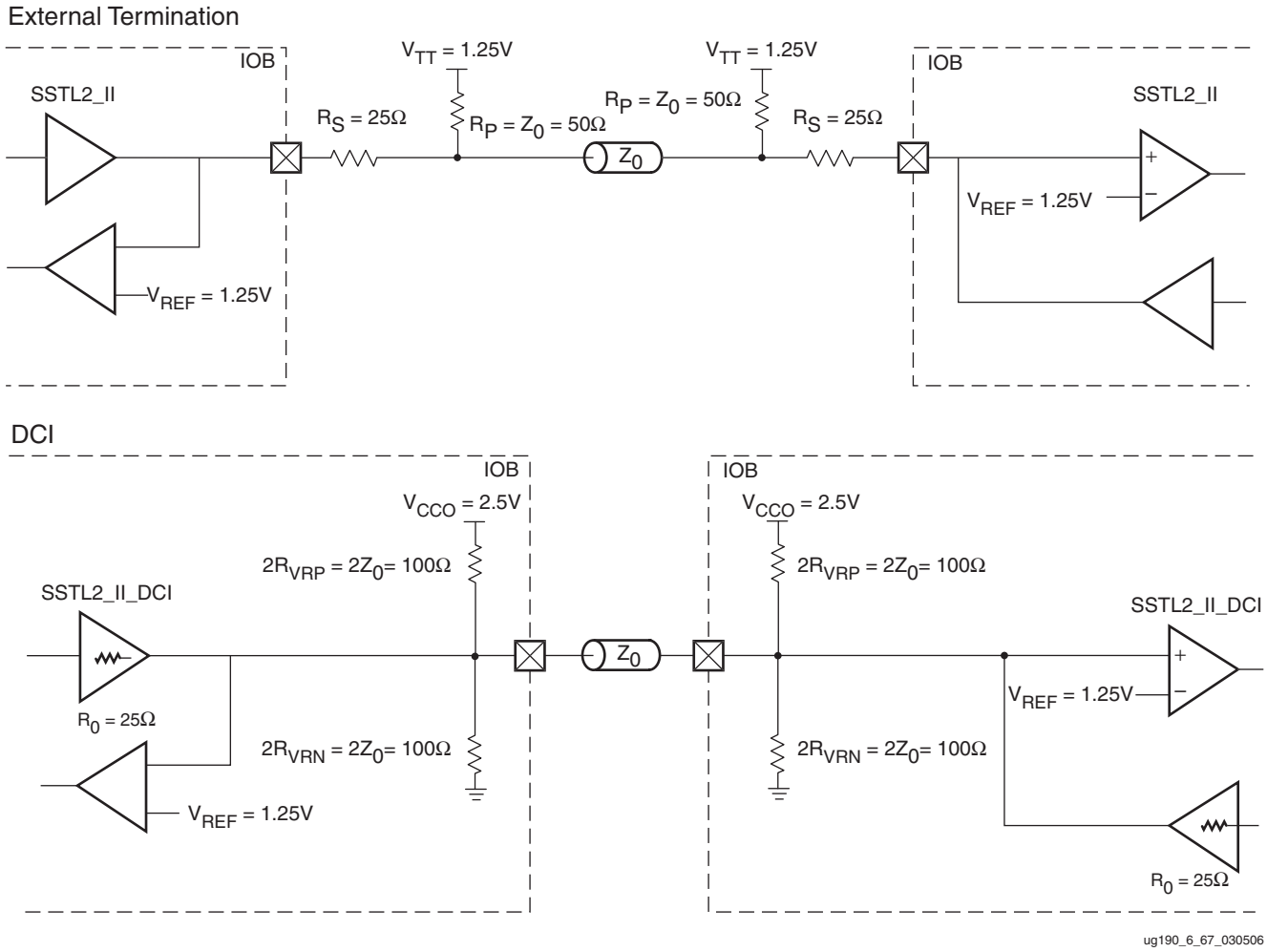


Figure 6-71: **SSTL2 Class II with Bidirectional Termination**

Table 6-30 lists the SSTL2 DC voltage specifications for Class II.

Table 6-30: **SSTL2 DC Voltage Specifications Class II**

	Min	Typ	Max
V_{CCO}	2.3	2.5	2.7
$V_{REF} = 0.5 \times V_{CCO}$	1.13	1.25	1.38
$V_{TT} = V_{REF} + N^{(1)}$	1.09	1.25	1.42
$V_{IH} \geq V_{REF} + 0.15$	1.28	1.40	$V_{CCO} + 0.3^{(2)}$
$V_{IL} \leq V_{REF} - 0.15$	$-0.3^{(3)}$	1.1	1.27
$V_{OH} \geq V_{REF} + 0.81$	1.93	2.03	2.13
$V_{OL} \leq V_{REF} - 0.81^{(4)}$	0.36	0.46	0.55
I_{OH} at V_{OH} (mA)	–16.2	–	–
I_{OL} at V_{OL} (mA)	16.2	–	–

Notes:

1. N must be greater than or equal to –0.04 and less than or equal to 0.04.
2. V_{IH} maximum is V_{CCO} +0.3.
3. V_{IL} minimum does not conform to the formula.
4. Because SSTL2_I_DCI uses a controlled-impedance driver, V_{OH} and V_{OL} are different.

Differential SSTL2 Class II (2.5V)

Figure 6-72 shows a sample circuit illustrating a valid termination technique for differential SSTL2 Class II (2.5V) with unidirectional termination.

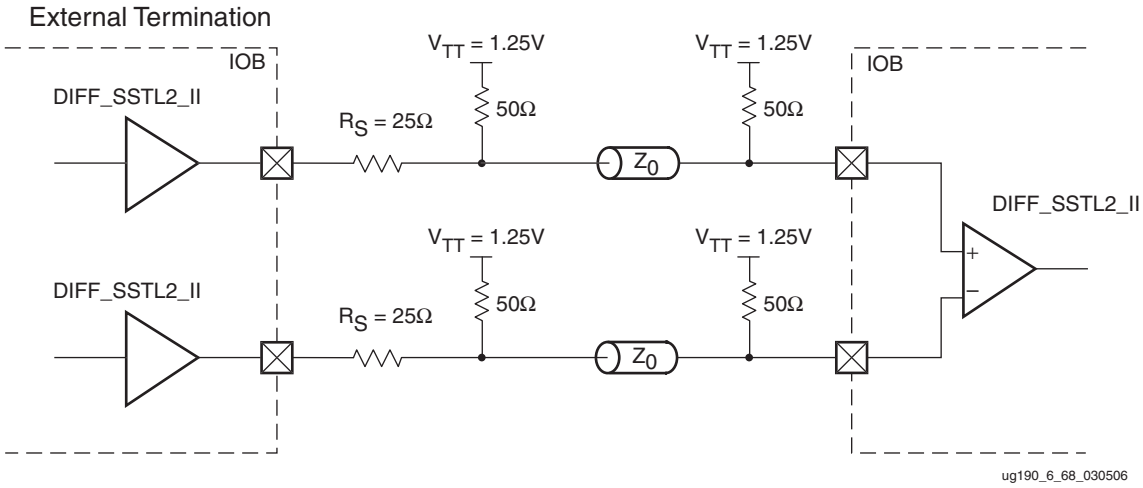


Figure 6-72: **Differential SSTL2 Class II Unidirectional Termination**

Figure 6-73 shows a sample circuit illustrating a valid termination technique for differential SSTL2 Class II (2.5V) with unidirectional DCI termination.

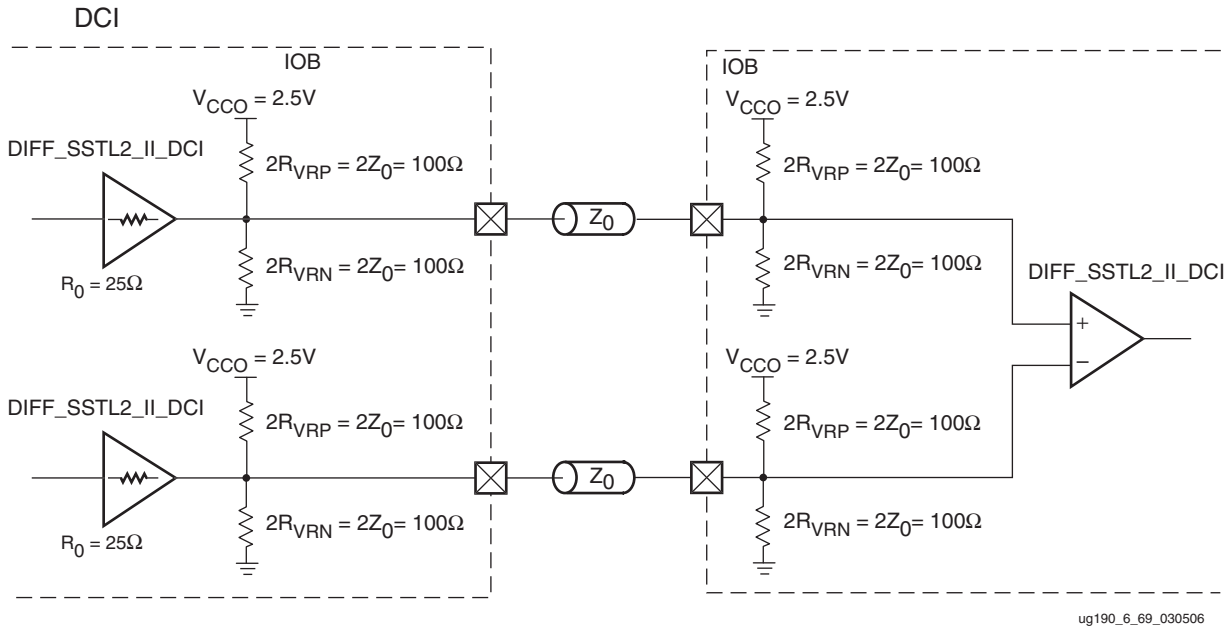


Figure 6-73: **Differential SSTL2 (2.5V) Class II Unidirectional DCI Termination**

Figure 6-74 shows a sample circuit illustrating a valid termination technique for differential SSTL2 Class II (2.5V) with bidirectional termination.

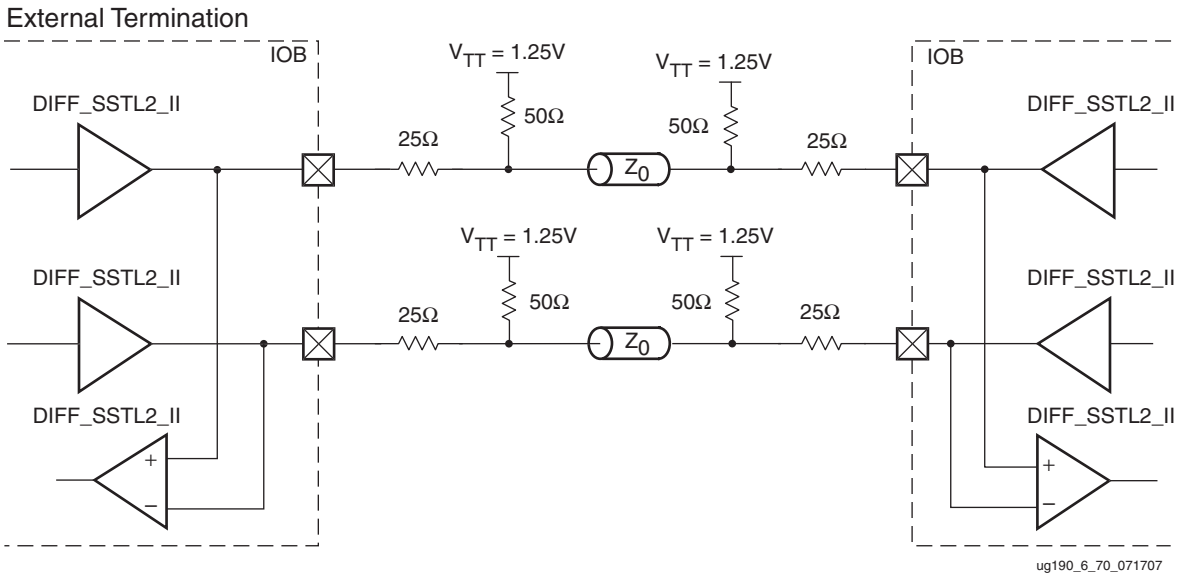


Figure 6-74: **Differential SSTL2 (2.5V) Class II with Bidirectional Termination**

Figure 6-75 shows a sample circuit illustrating a valid termination technique for differential SSTL2 Class II (2.5V) with bidirectional DCI termination.

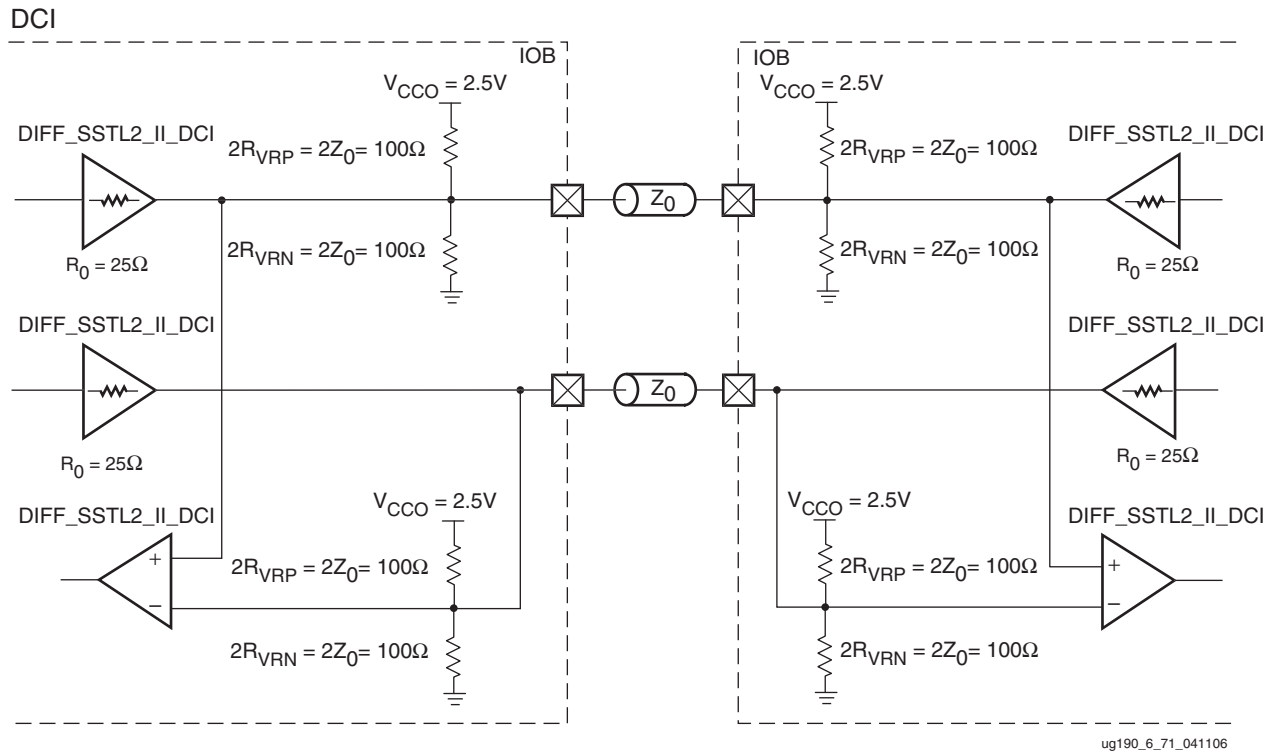


Figure 6-75: **Differential SSTL2 (2.5V) Class II with DCI Bidirectional Termination**

Table 6-31 lists the differential SSTL2 Class II DC voltage specifications.

Table 6-31: **Differential SSTL2 Class II DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	2.3	2.5	2.7
Input Parameters			
V_{TT}	–	$V_{CCO} \times 0.5$	–
V_{IN} (DC)(1)	–0.30	–	V_{CCO} + 0.30
V_{ID} (DC)(2)	0.3	–	V_{CCO} + 0.60
V_{ID} (AC)	0.62	–	V_{CCO} + 0.60
V_{IX} (AC)(3)	0.95	–	1.55
Output Parameters			
V_{OX} (AC)(4)	1.0	–	1.5

Notes:

1. V_{IN} (DC) specifies the allowable DC excursion of each differential input.
2. V_{ID} (DC) specifies the input differential voltage required for switching.
3. V_{IX} (AC) indicates the voltage where the differential input signals must cross.
4. V_{OX} (AC) indicates the voltage where the differential output signals must cross.

SSTL2_II_T_DCI (2.5V) Split-Thevenin Termination

Figure 6-76 shows a sample circuit illustrating a valid termination technique for SSTL2_II_T_DCI (2.5V) with on-chip split-thevenin termination. In this bidirectional I/O standard, when 3-stated, the termination is invoked on the receiver and not on the driver.

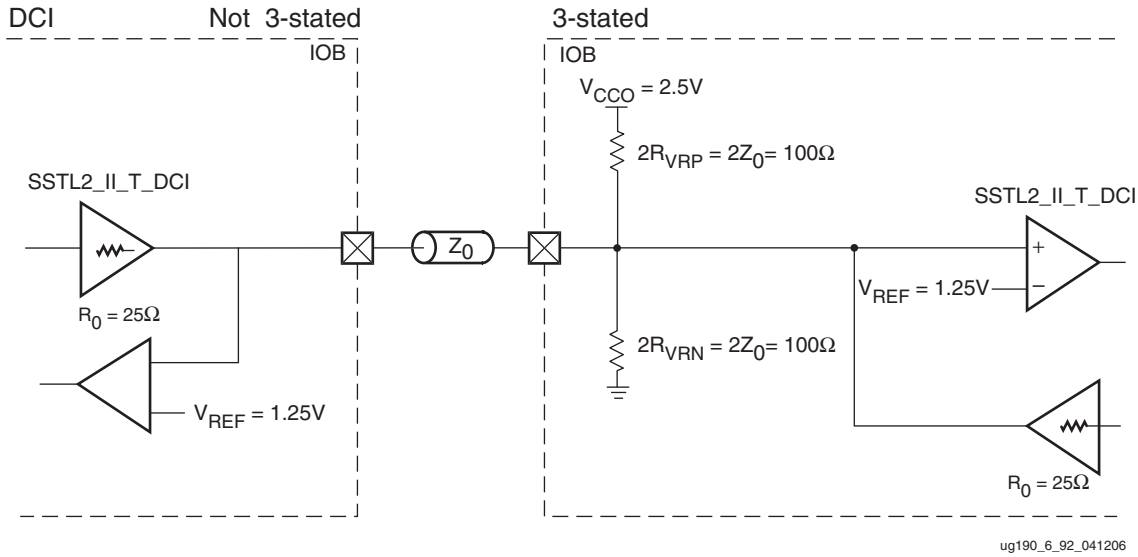


Figure 6-76: **SSTL2_II_T_DCI (2.5V) Split-Thevenin Termination**

SSTL18 Class I (1.8V)

Figure 6-77 shows a sample circuit illustrating a valid termination technique for SSTL Class I (1.8V).

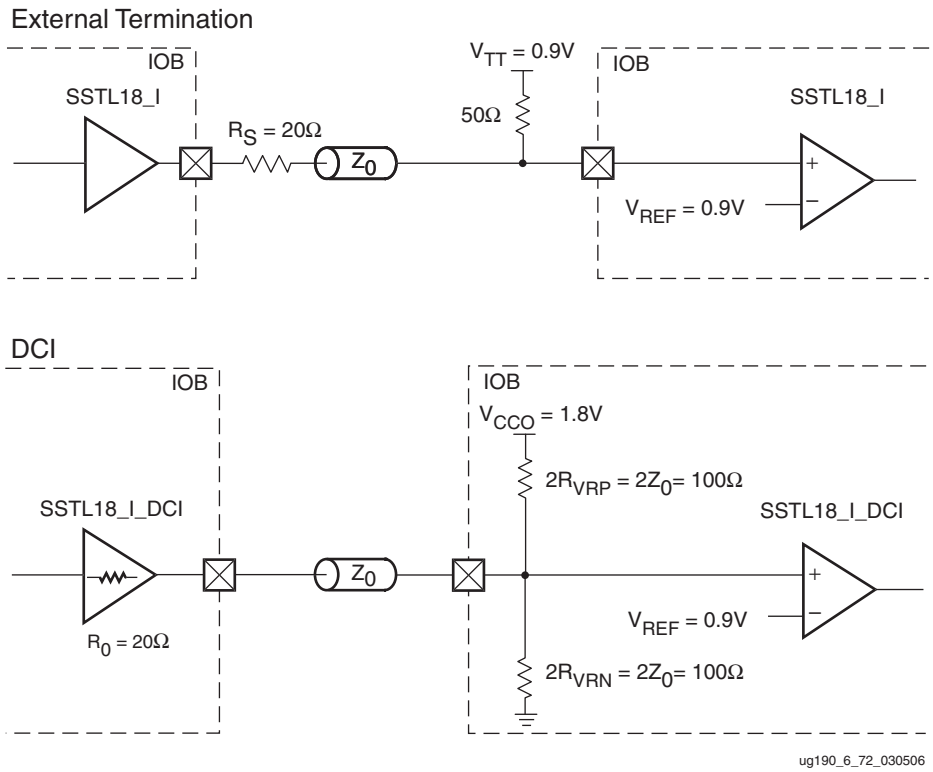


Figure 6-77: **SSTL18 (1.8V) Class I Termination**

Differential SSTL Class I (1.8V)

Figure 6-78 shows a sample circuit illustrating a valid termination technique for differential SSTL Class I (1.8V) with unidirectional termination.

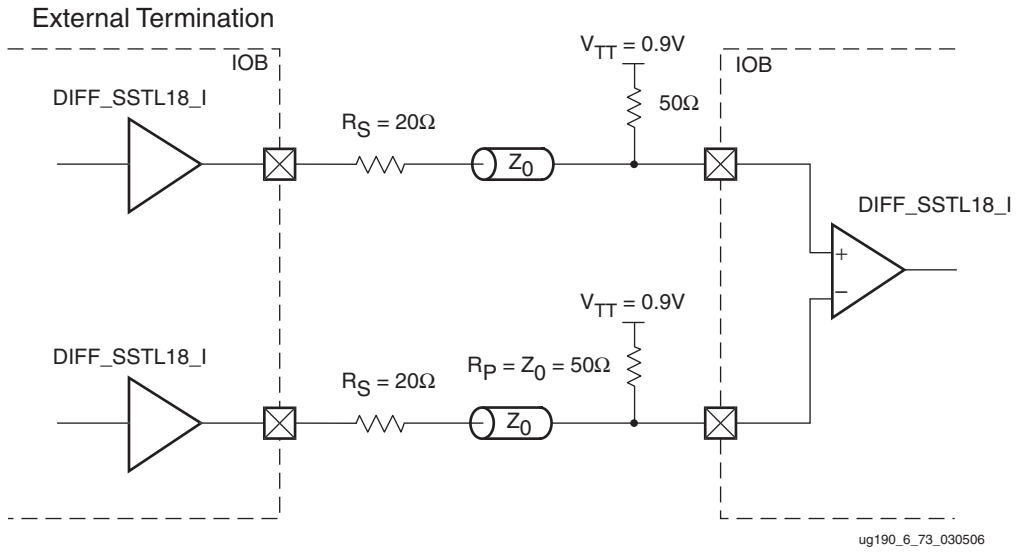


Figure 6-78: **Differential SSTL (1.8V) Class I Unidirectional Termination**

Figure 6-79 shows a sample circuit illustrating a valid termination technique for differential SSTL Class I (1.8V) with unidirectional DCI termination.

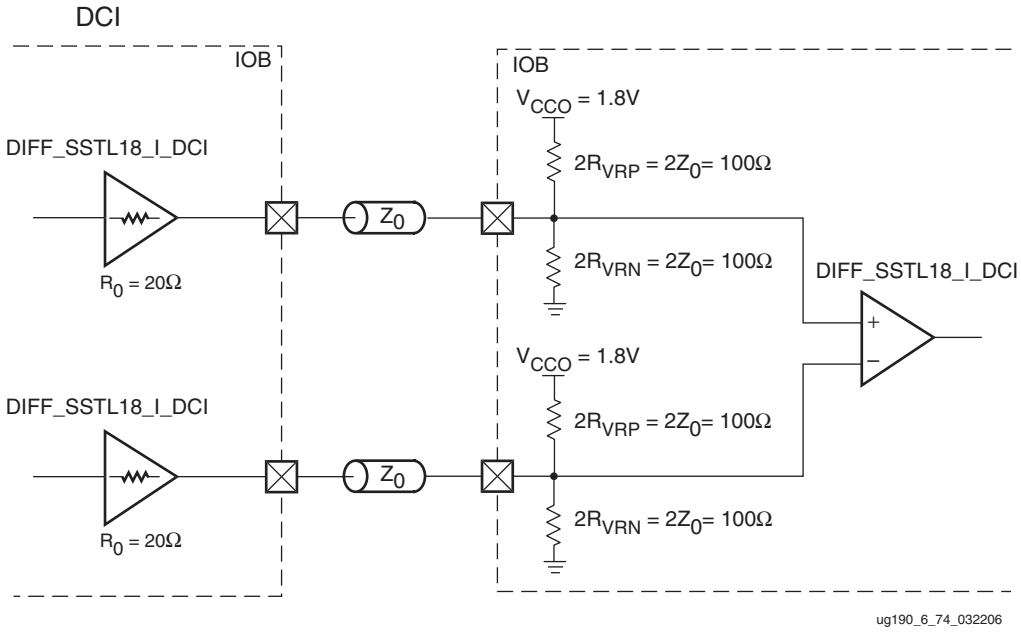


Figure 6-79: **Differential SSTL (1.8V) Class I Unidirectional DCI Termination**

Table 6-32 lists the differential SSTL (1.8V) Class I DC voltage specifications.

Table 6-32: **Differential SSTL (1.8V) Class I and Class II DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	1.7	1.8	1.9
Input Parameters			
V_{TT}	–	$V_{CCO} \times 0.5$	–
V_{IN} (DC)(1)	–0.30	–	V_{CCO} + 0.30
V_{ID} (DC)(3)	0.25	–	V_{CCO} + 0.60
V_{ID} (AC)	0.50	–	V_{CCO} + 0.60
V_{IX} (AC)(4)	0.675	–	1.125
Output Parameters			
V_{OX} (AC)(5)	0.725	–	1.075

Notes:

1. V_{IN} (DC) specifies the allowable DC excursion of each differential input.
2. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."
3. V_{ID} (DC) specifies the input differential voltage required for switching.
4. V_{IX} (AC) indicates the voltage where the differential input signals must cross.
5. V_{OX} (AC) indicates the voltage where the differential output signals must cross.

SSTL18 Class II (1.8V)

Figure 6-80 shows a sample circuit illustrating a valid unidirectional termination technique for SSTL Class II (1.8V).

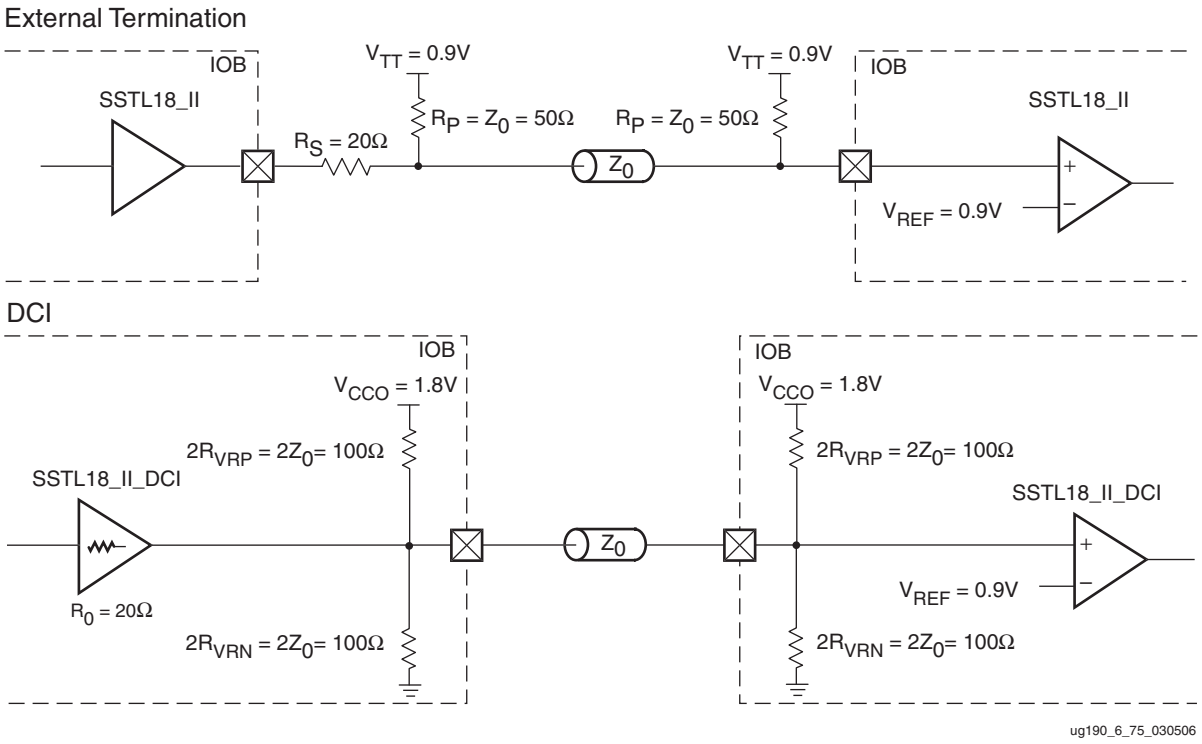


Figure 6-80: **SSTL18 (1.8V) Class II Unidirectional Termination**

Figure 6-81 shows a sample circuit illustrating a valid bidirectional termination technique for SSTL (1.8V) Class II.

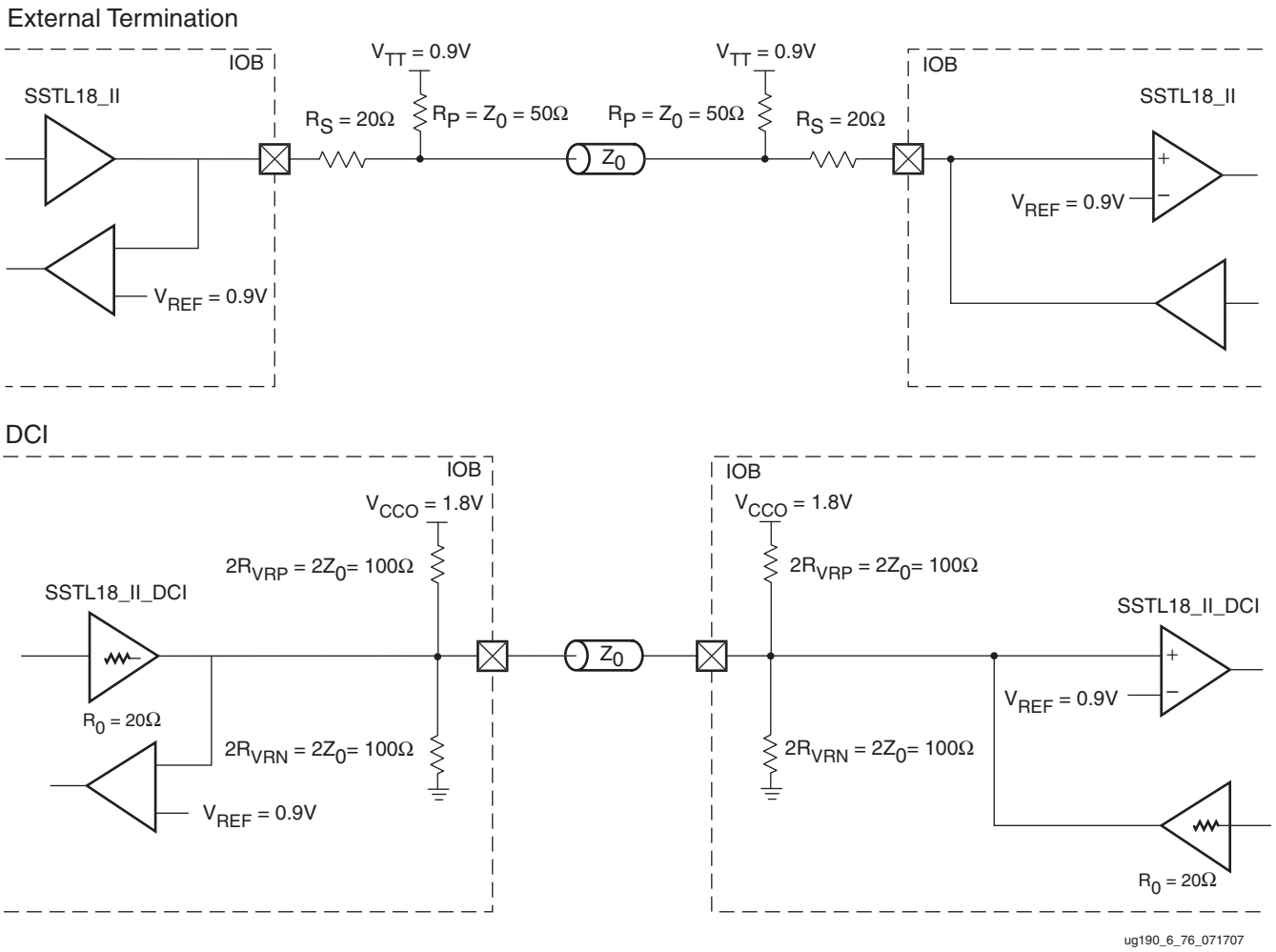


Figure 6-81: **SSTL (1.8V) Class II Termination**

Table 6-33 and Table 6-34 lists the SSTL (1.8V) DC voltage specifications for Class I and Class II respectively.

Table 6-33: **SSTL (1.8V) DC Voltage Specifications Class I**

	Class I		
	Min	**Typ**	**Max**
V_{CCO}	1.7	1.8	1.9
$V_{REF} = 0.5 \times V_{CCO}$	0.833	0.9	0.969
$V_{TT} = V_{REF} + N$(1)	0.793	0.9	1.009
$V_{IH} \geq V_{REF} + 0.125$	0.958	–	$V_{CCO} + 0.3$(2)
$V_{IL} \leq V_{REF} - 0.125$	–0.3(3)	–	0.844
$V_{OH} \geq V_{TT} + 0.47$(4)	1.263	–	–
$V_{OL} \leq V_{TT} - 0.47$(4)	–	–	0.539
I_{OH} at V_{OH} (mA)	–6.7	–	–
I_{OL} at V_{OL} (mA)	6.7	–	–

Notes:

1. N must be greater than or equal to –0.04 and less than or equal to 0.04.
2. V_{IH} maximum is V_{CCO} +0.3.
3. V_{IL} minimum does not conform to the formula.
4. Because SSTL_I_DCI uses a controlled-impedance driver, V_{OH} and V_{OL} are different.

Table 6-34: **SSTL (1.8V) DC Voltage Specifications Class II**

	Class II		
	Min	**Typ**	**Max**
V_{CCO}	1.7	1.8	1.9
$V_{REF} = 0.5 \times V_{CCO}$	0.833	0.9	0.969
$V_{TT} = V_{REF} + N$(1)	0.793	0.9	1.009
$V_{IH} \geq V_{REF} + 0.125$	0.958	–	$V_{CCO} + 0.3$(2)
$V_{IL} \leq V_{REF} - 0.125$	–0.3(3)	–	0.844
$V_{OH} \geq V_{TT} + 0.603$(4)	1.396	–	–
$V_{OL} \leq V_{TT} - 0.603$(4)	–	–	0.406
I_{OH} at V_{OH} (mA)	–13.4	–	–
I_{OL} at V_{OL} (mA)	13.4	–	–

Notes:

1. N must be greater than or equal to –0.04 and less than or equal to 0.04.
2. V_{IH} maximum is V_{CCO} +0.3.
3. V_{IL} minimum does not conform to the formula.
4. Because SSTL_I_DCI uses a controlled-impedance driver, V_{OH} and V_{OL} are different.

Differential SSTL Class II (1.8V)

Figure 6-82 shows a sample circuit illustrating a valid termination technique for differential SSTL Class II (1.8V) with unidirectional termination.

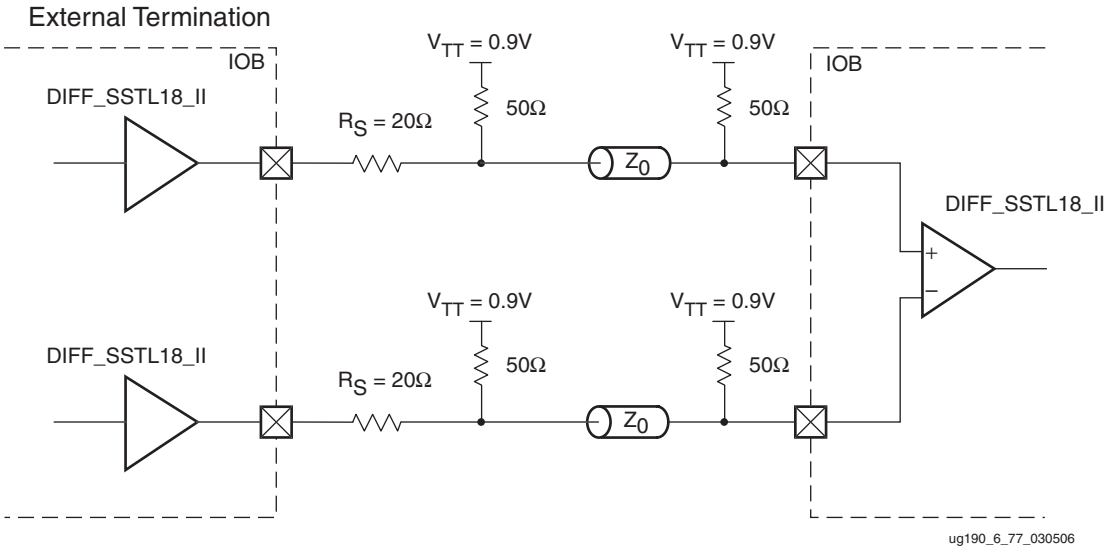


Figure 6-82: **Differential SSTL (1.8V) Class II Unidirectional Termination**

Figure 6-83 shows a sample circuit illustrating a valid termination technique for differential SSTL Class II (1.8V) with unidirectional DCI termination.

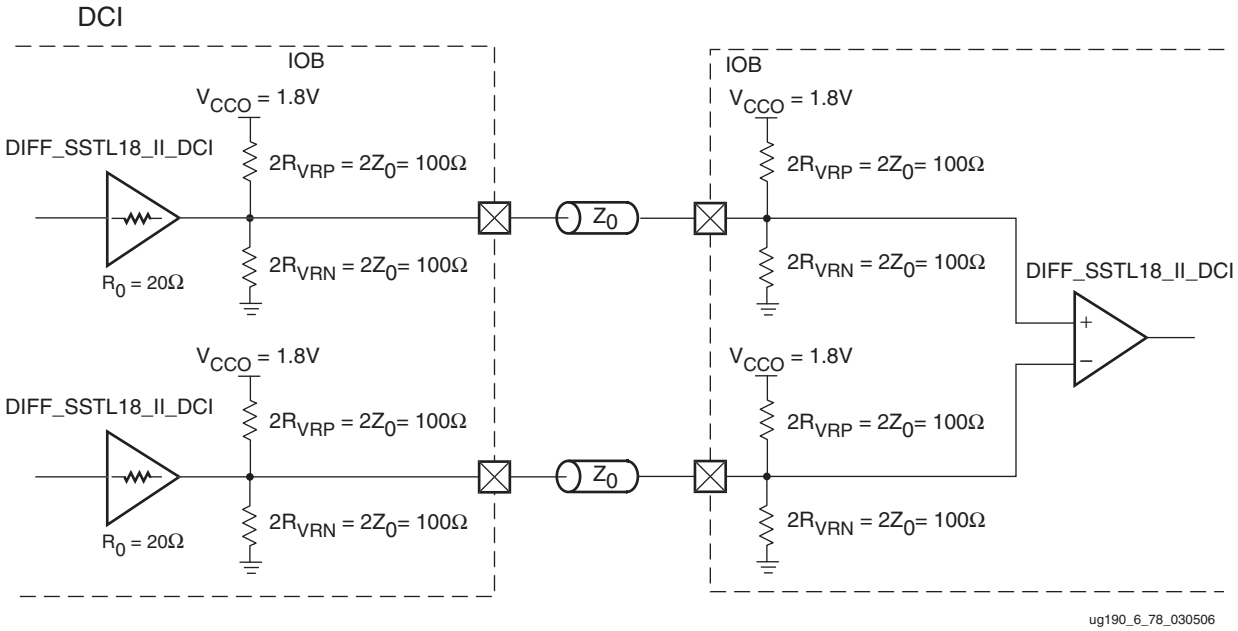


Figure 6-83: **Differential SSTL (1.8V) Class II Unidirectional DCI Termination**

Figure 6-84 shows a sample circuit illustrating a valid termination technique for differential SSTL Class II (1.8V) with bidirectional termination.

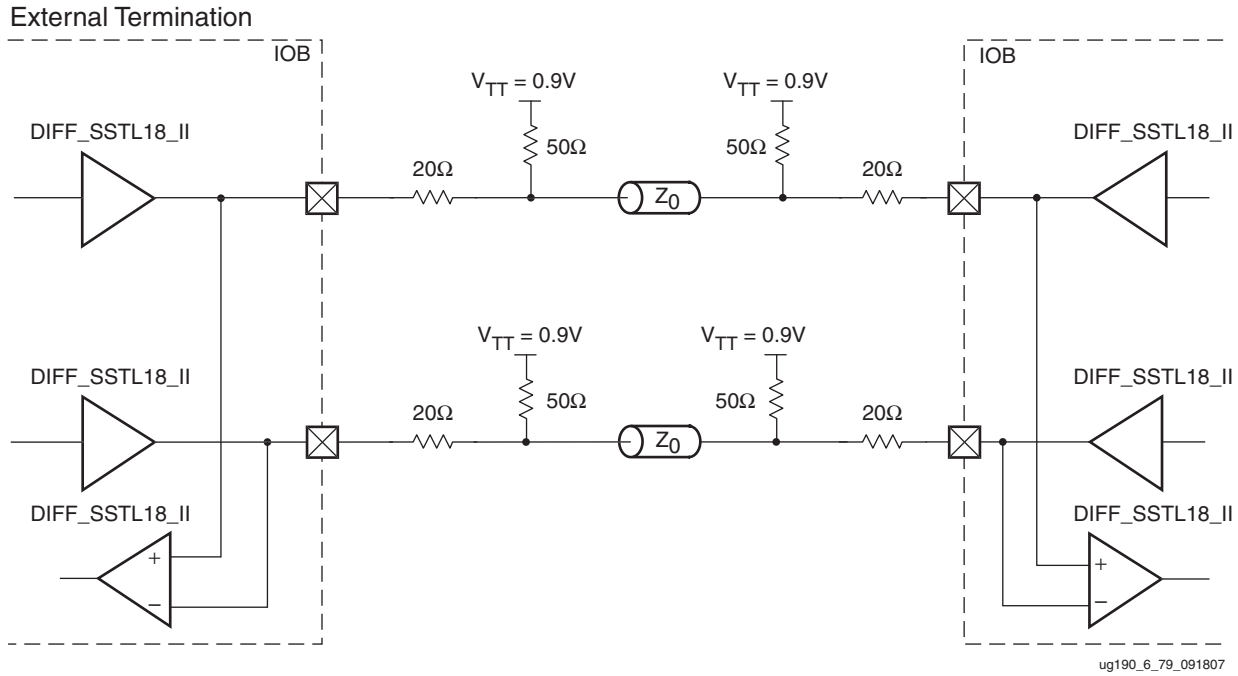


Figure 6-84: **Differential SSTL (1.8V) Class II with Bidirectional Termination**

Figure 6-85 shows a sample circuit illustrating a valid termination technique for differential SSTL Class II (1.8V) with bidirectional DCI termination.

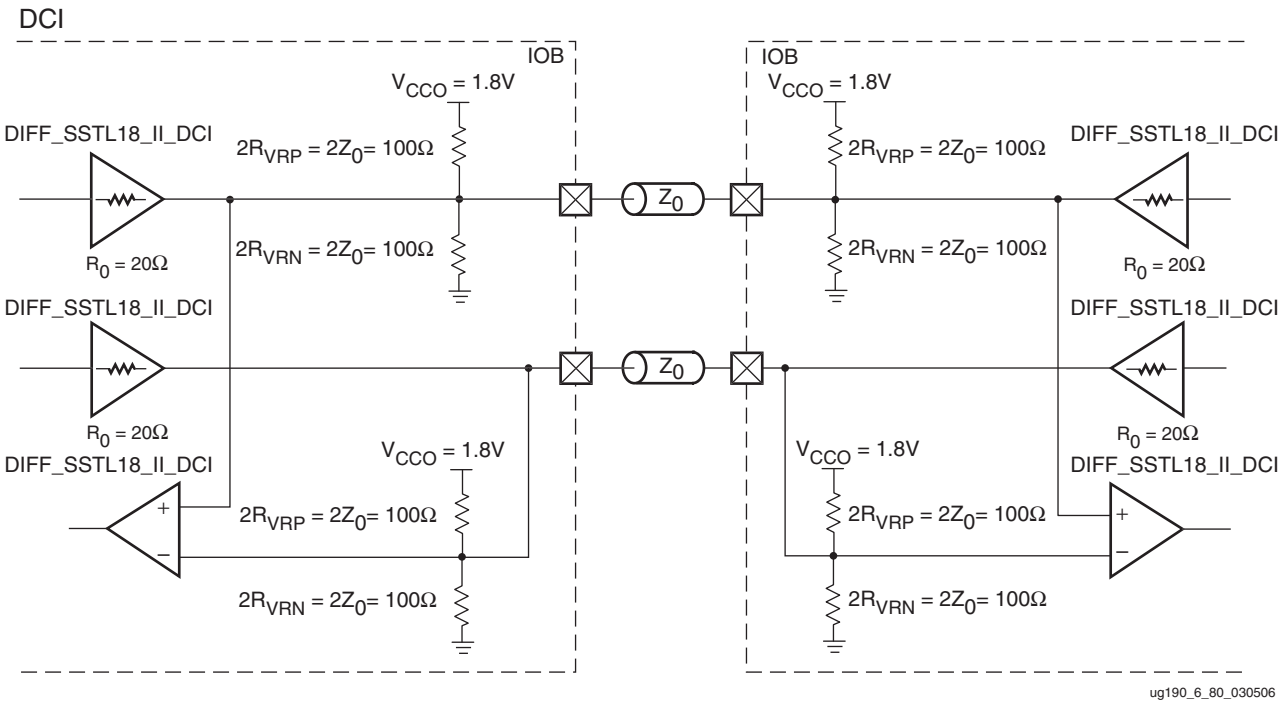


Figure 6-85: **Differential SSTL (1.8V) Class II with DCI Bidirectional Termination**

Table 6-35 lists the differential SSTL (1.8V) Class II DC voltage specifications.

Table 6-35: **Differential SSTL (1.8V) Class II DC Voltage Specifications**

	Min	Typ	Max
V_{CCO}	1.7	1.8	1.9
Input Parameters			
V_{TT}	–	$V_{CCO} \times 0.5$	–
V_{IN} (DC)(1)	–0.30	–	V_{CCO} + 0.30
V_{ID} (DC)(3)	0.25	–	V_{CCO} + 0.60
V_{ID} (AC)	0.50	–	V_{CCO} + 0.60
V_{IX} (AC)(4)	0.675	–	1.125
Output Parameters			
V_{OX} (AC)(5)	0.725	–	1.075

Notes:

1. V_{IN} (DC) specifies the allowable DC excursion of each differential input.
2. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."
3. V_{ID} (DC) specifies the input differential voltage required for switching.
4. V_{IX} (AC) indicates the voltage where the differential input signals must cross.
5. V_{OX} (AC) indicates the voltage where the differential output signals must cross.

SSTL18_II_T_DCI (1.8V) Split-Thevenin Termination

Figure 6-86 shows a sample circuit illustrating a valid termination technique for SSTL18_II_T_DCI (1.8V) with on-chip split-thevenin termination. In this bidirectional I/O standard, when 3-stated, the termination is invoked on the receiver and not on the driver. Because the Thevenin termination on the I/O is disabled for a driving I/O, the line is equivalent to the SSTL18_I termination scheme. This allows the line to be driven by the weaker SSTL class I driver. The SSTL18_II_T_DCI standard behaves like a normal SSTL18_II I/O in a bidirectional environment but has the advantage of lower drive strength and lower power consumption due to the optimized termination circuit.

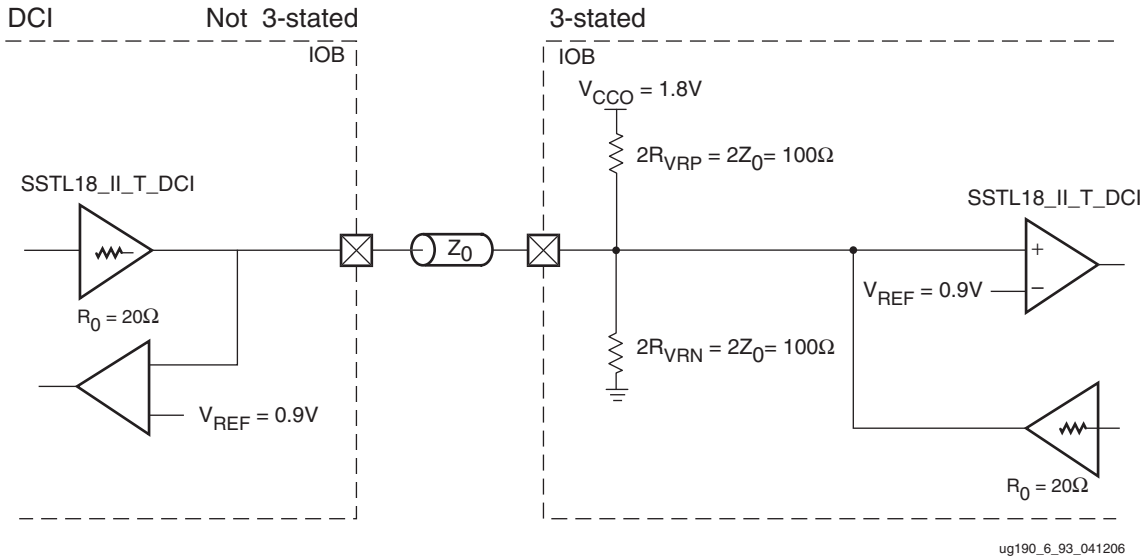


Figure 6-86: **SSTL18_II_T_DCI (1.8V) Split-Thevenin Termination**

Differential Termination: DIFF_TERM Attribute

Virtex-5 FPGA IOBs provide a 100Ω differential termination across the input differential receiver terminals. This attribute is used in conjunction with LVDS_25, LVDSEXT_25, HT_25, and RSDS_25. HT_25 replaces the Virtex-4 FPGA LDT_25 standard.

The on-chip input differential termination in Virtex-5 devices provides major advantages over the external resistor by removing the stub at the receiver completely and therefore greatly improving signal integrity:

- Consumes less power than DCI termination
- Does not use VRP/VRN pins (DCI)

The V_{CCO} of the I/O bank must be connected to 2.5V ±5% to provide 100Ω of effective differential termination. DIFF_TERM is only available for inputs and can *only* be used with a bank voltage of V_{CCO} = 2.5V. The "Differential Termination Attribute" (DIFF_TERM) section outlines using this feature.

LVDS and Extended LVDS (Low Voltage Differential Signaling)

Low Voltage Differential Signaling (LVDS) is a very popular and powerful high-speed interface in many system applications. Virtex-5 FPGA I/Os are designed to comply with the EIA/TIA electrical specifications for LVDS to make system and board design easier. With the use of an LVDS current-mode driver in the IOBs, the need for external source termination in point-to-point applications is eliminated, and with the choice of an extended mode, Virtex-5 devices provide the most flexible solution for doing an LVDS design in an FPGA.

Extended LVDS provides a higher drive capability and voltage swing (350 - 750 mV), making it ideal for long-distance or cable LVDS links. The output AC characteristics of the LVDS extended mode driver are not within the EIA/TIA specifications. The LVDS extended mode driver is intended for situations requiring higher drive capabilities to produce an LVDS signal within the EIA/TIA specification at the receiver.

Transmitter Termination

The Virtex-5 FPGA LVDS transmitter does not require any external termination. Table 6-36 lists the allowed attributes corresponding to the Virtex-5 FPGA LVDS current-mode drivers. Virtex-5 FPGA LVDS current-mode drivers are a true current source and produce the proper (EIA/TIA compliant) LVDS signal.

Receiver Termination

Figure 6-87 is an example of differential termination for an LVDS receiver on a board with 50 Ω transmission lines.

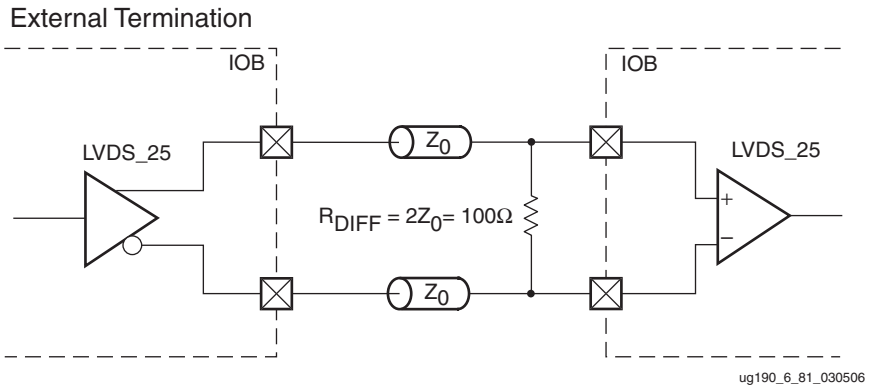


Figure 6-87: **LVDS_25 Receiver Termination**

Figure 6-88 is an example of a differential termination for an LVDS receiver on a board with 50 Ω transmission lines.

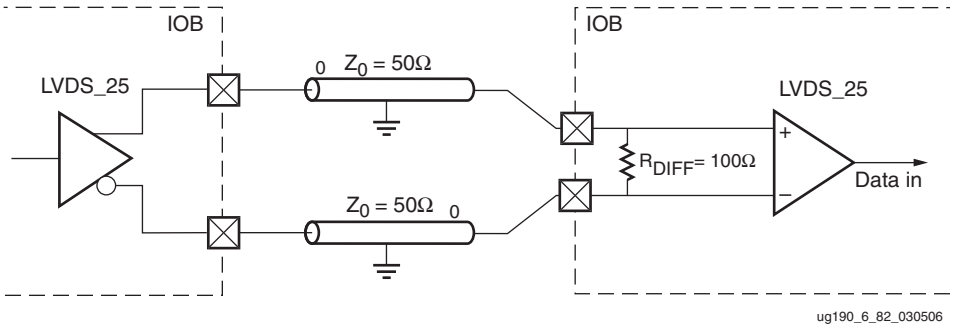


Figure 6-88: **LVDS_25 With DIFF_TERM Receiver Termination**

Table 6-36 lists the available Virtex-5 FPGA LVDS I/O standards and attributes supported.

Table 6-36: **Allowed Attributes of the LVDS I/O Standard**

Attributes	Primitives	
	IBUFDS/IBUFGDS	OBUFDS/OBUFTDS
IOSTANDARD	LVDS_25, LVDSEXT_25	
DIFF_TERM	TRUE, FALSE	N/A

HyperTransport™ Protocol (HT)

The HyperTransport protocol (HT) also known as Lightning Data Transport (LDT), is a low-voltage standard for high speed interfaces. Its differential signaling based interface is very similar to LVDS. Virtex-5 FPGA IOBs are equipped with HT buffers. Table 6-38 summarizes all the possible HT I/O standards and attributes supported.

Table 6-37: **Allowed Attributes of the HT I/O Standard**

Attributes	Primitives	
	IBUFDS/IBUFGDS	OBUFDS/OBUFTDS
IOSTANDARD	HT_25	
DIFF_TERM	TRUE, FALSE	N/A

Reduced Swing Differential Signaling (RSDS)

Reduced Swing Differential Signaling (RSDS) is similar to an LVDS high-speed interface using differential signaling. RSDS has a similar implementation to LVDS in Virtex-5 devices and is only intended for point-to-point applications.

Table 6-38: **Allowed Attributes of the RSDS I/O Standard**

Attributes	Primitives	
	IBUFDS/IBUFGDS	OBUFDS/OBUFTDS
IOSTANDARD	RSDS_25	
DIFF_TERM	TRUE, FALSE	N/A

BLVDS (Bus LVDS)

Since LVDS is intended for point-to-point applications, BLVDS is not an EIA/TIA standard implementation and requires careful adaptation of I/O and PCB layout design rules. The primitive supplied in the software library for bidirectional LVDS does not use the Virtex-5 FPGA LVDS current-mode driver, instead, it uses complementary single-ended differential drivers. Therefore, source termination is required. Figure 6-89 shows the BLVDS transmitter termination.

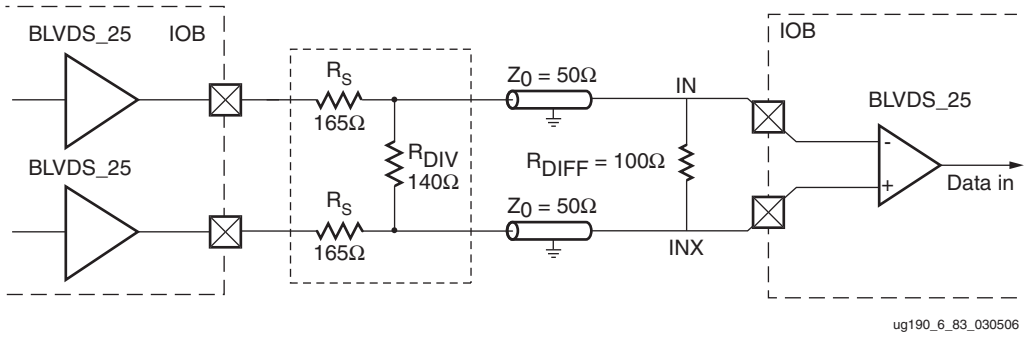


Figure 6-89: **BLVDS Transmitter Termination**

Differential LVPECL (Low-Voltage Positive Emitter-Coupled Logic)

LVPECL is a very popular and powerful high-speed interface in many system applications. Virtex-5 FPGA I/Os are designed to comply with the EIA/TIA electrical specifications for 2.5V LVPECL to make system and board design easier.

LVPECL Transceiver Termination

The Virtex-5 FPGA LVPECL transmitter and receiver requires the termination shown in Figure 6-90, illustrating a Virtex-5 FPGA LVPECL transmitter and receiver on a board with 50 Ω transmission lines. The LVPECL driver is composed of two LVCMOS drivers that form a compliant LVPECL output when combined with the three resistor output termination circuit.

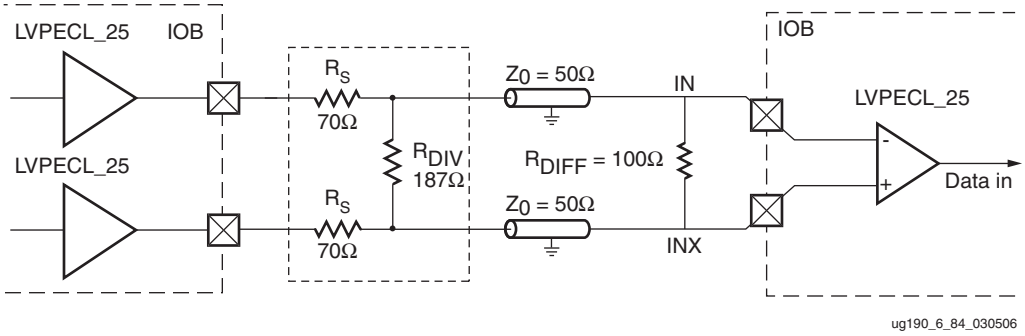


Figure 6-90: **LVPECL Transmitter Termination**

Rules for Combining I/O Standards in the Same Bank

The following rules must be obeyed to combine different input, output, and bidirectional standards in the same bank:

1. **Combining output standards only.** Output standards with the same output V_{CCO} requirement can be combined in the same bank.

   Compatible example:

   SSTL2_I and LVDCI_25 outputs

   Incompatible example:

   SSTL2_I (output V_{CCO} = 2.5V) and
   LVCMOS33 (output V_{CCO} = 3.3V) outputs

2. **Combining input standards only.** Input standards with the same V_{CCO} and V_{REF} requirements can be combined in the same bank.

   Compatible example:

   LVCMOS15 and HSTL_IV inputs

   Incompatible example:

   LVCMOS15 (input V_{CCO} = 1.5V) and
   LVCMOS18 (input V_{CCO} = 1.8V) inputs

   Incompatible example:

   HSTL_I_DCI_18 (V_{REF} = 0.9V) and
   HSTL_IV_DCI_18 (V_{REF} = 1.1V) inputs

3. **Combining input standards and output standards.** Input standards and output standards with the same V_{CCO} requirement can be combined in the same bank.

   Compatible example:

   LVDS_25 output and HSTL_I input

   Incompatible example:

   LVDS_25 output (output V_{CCO} = 2.5V) and
   HSTL_I_DCI_18 input (input V_{CCO} = 1.8V)

4. **Combining bidirectional standards with input or output standards.** When combining bidirectional I/O with other standards, make sure the bidirectional standard can meet the first three rules.

5. **Additional rules for combining DCI I/O standards.**

   a. No more than one Single Termination type (input or output) is allowed in the same bank.

      Incompatible example:

      HSTL_IV_DCI input and HSTL_III_DCI input

   b. No more than one Split Termination type (input or output) is allowed in the same bank.

      Incompatible example:

      HSTL_I_DCI input and HSTL_II_DCI input

The implementation tools enforce these design rules.

Table 6-39, summarizes the Virtex-5 FPGA supported I/O standards.

Table 6-39: **I/O Compatibility**

I/O Standard	V_{CCO}		V_{REF}	Termination Type	
	Output	Input	Input	Output	Input
LVTTL(1)	3.3	3.3	N/R	N/R	N/R
LVCMOS33(1)			N/R	N/R	N/R
LVDCI_33(1)			N/R	Series	N/R
HSLVDCI_33(1)			$V_{CCO}/2$	Series	N/R
PCIX(1)			N/R	N/R	N/R
PCI33_3(1)			N/R	N/R	N/R
PCI66_3(1)			N/R	N/R	N/R
LVDS_25	2.5	Note (2)	N/R	N/R	N/R
LVDSEXT_25			N/R	N/R	N/R
HT_25			N/R	N/R	N/R
RSDS_25(4)			N/R	N/R	N/R
BLVDS_25			N/R	N/R	N/R
LVPECL_25			N/R	N/R	N/R
SSTL2_I			1.25	N/R	N/R
SSTL2_II			1.25	N/R	N/R
DIFF_SSTL2_I			N/R	N/R	N/R
DIFF_SSTL2_II			N/R	N/R	N/R
LVCMOS25		2.5	N/R	N/R	N/R
LVDCI_25			N/R	Series	N/R
HSLVDCI_25			$V_{CCO}/2$	Series	N/R
LVDCI_DV2_25			N/R	Series	N/R
SSTL2_I_DCI			1.25	N/R	Split
SSTL2_II_DCI			1.25	Split	Split
SSTL2_II_T_DCI			1.25	N/R	Split
DIFF_SSTL2_I_DCI			N/R	N/R	Split
DIFF_SSTL2_II_DCI			N/R	Split	Split

Table 6-39: **I/O Compatibility *(Continued)***

I/O Standard	V_{CCO}		V_{REF}	Termination Type	
	Output	Input	Input	Output	Input
HSTL_III_18	1.8	Note (2)	1.08	N/R	N/R
HSTL_IV_18			1.08	N/R	N/R
HSTL_I_18			0.9	N/R	N/R
HSTL_II_18			0.9	N/R	N/R
DIFF_HSTL_I_18			N/R	N/R	N/R
DIFF_HSTL_II_18			N/R	N/R	N/R
SSTL18_I			0.9	N/R	N/R
SSTL18_II			0.9	N/R	N/R
DIFF_SSTL18_I			N/R	N/R	N/R
DIFF_SSTL18_II			N/R	N/R	N/R
LVCMOS18		1.8	N/R	N/R	N/R
LVDCI_18			N/R	Series	N/R
HSLVDCI_18			$V_{CCO}/2$	Series	N/R
LVDCI_DV2_18			N/R	Series	N/R
HSTL_III_DCI_18			1.08	N/R	Single
HSTL_IV_DCI_18			1.08	Single	Single
HSTL_I_DCI_18			0.9	N/R	Split
HSTL_II_DCI_18			0.9	Split	Split
HSTL_II_T_DCI_18			0.9	N/R	Split
DIFF_HSTL_I_DCI_18			N/R	N/R	Split
DIFF_HSTL_II_DCI_18			N/R	Split	Split
SSTL18_I_DCI			0.9	N/R	Split
SSTL18_II_DCI			0.9	Split	Split
SSTL18_II_T_DCI			0.9	N/R	Split
DIFF_SSTL18_I_DCI			N/R	N/R	Split
DIFF_SSTL18_II_DCI			N/R	Split	Split

Table 6-39: **I/O Compatibility *(Continued)***

I/O Standard	V_{CCO}		V_{REF}	Termination Type	
	Output	Input	Input	Output	Input
HSTL_III	1.5	Note (2)	0.9	N/R	N/R
HSTL_IV			0.9	N/R	N/R
HSTL_I			0.75	N/R	N/R
HSTL_II			0.75	N/R	N/R
DIFF_HSTL_I			N/R	N/R	N/R
DIFF_HSTL_II			N/R	N/R	N/R
LVCMOS15		1.5	N/R	N/R	N/R
LVDCI_15			N/R	Series	N/R
HSLVDCI_15			$V_{CCO}/2$	Series	N/R
LVDCI_DV2_15			N/R	Series	N/R
GTLP_DCI			1	Single	Single
HSTL_III_DCI			0.9	N/R	Single
HSTL_IV_DCI			0.9	Single	Single
HSTL_I_DCI			0.75	N/R	Split
HSTL_II_DCI			0.75	Split	Split
HSTL_II_T_DCI			0.75	N/R	Split
DIFF_HSTL_I_DCI			N/R	N/R	Split
DIFF_HSTL_II_DCI			N/R	Split	Split
GTL_DCI	1.2	1.2	0.8	Single	Single
GTLP	N/R	Note (2)	1	N/R	N/R
GTL			0.8	N/R	N/R
LVCMOS12	1.2	1.2	N/R	N/R	N/R
HSTL_I_12			0.6	N/R	N/R

Notes:
1. See "3.3V I/O Design Guidelines" for more detailed information
2. Differential inputs and inputs using V_{REF} are powered from V_{CCAUX}. However, pin voltage must not exceed V_{CCO}, due to the presence of clamp diodes to V_{CCO}.
3. N/R = no requirement.
4. RSDS_25 has the same DC specifications as LVDS_25. All information pertaining to LVDS_25 is applicable to RSDS_25.
5. I/O standard is selected using the IOSTANDARD attribute.

3.3V I/O Design Guidelines

To achieve maximum performance in Virtex-5 devices, several 3.3V I/O design guidelines and techniques are highlighted in this section. This includes managing overshoot/undershoot with termination techniques, regulating V_{CCO} at 3.0V with a voltage regulator, using external bus switches, reviewing configuration methods, and other design considerations.

I/O Standard Design Rules

Overshoot/Undershoot

Undershoot and overshoot voltages on I/Os operating at 3.3V should not exceed the absolute maximum ratings of –0.3V to 4.05V, respectively, when V_{CCO} is 3.75V. These absolute maximum limits are stated in the absolute maximum ratings table in the *Virtex-5 FPGA Data Sheet*. However, the maximum undershoot value is directly affected by the value of V_{CCO}.

The voltage across the gate oxide at any time must not exceed 4.05V. Consider the case in which the I/O is either an input or a 3-stated buffer as shown in Figure 6-91. The gate of the output PMOS transistor P_0 and NMOS transistor N_0 is connected essentially to V_{CCO} and ground, respectively.

The amount of undershoot allowed without overstressing the PMOS transistor P_0 is the gate voltage minus the gate oxide limit, or V_{CCO} – 4.05V.

Similarly, the absolute maximum overshoot allowed without overstressing the NMOS transistor N_0 is the gate voltage plus the gate oxide limit, or Ground + 4.05V.

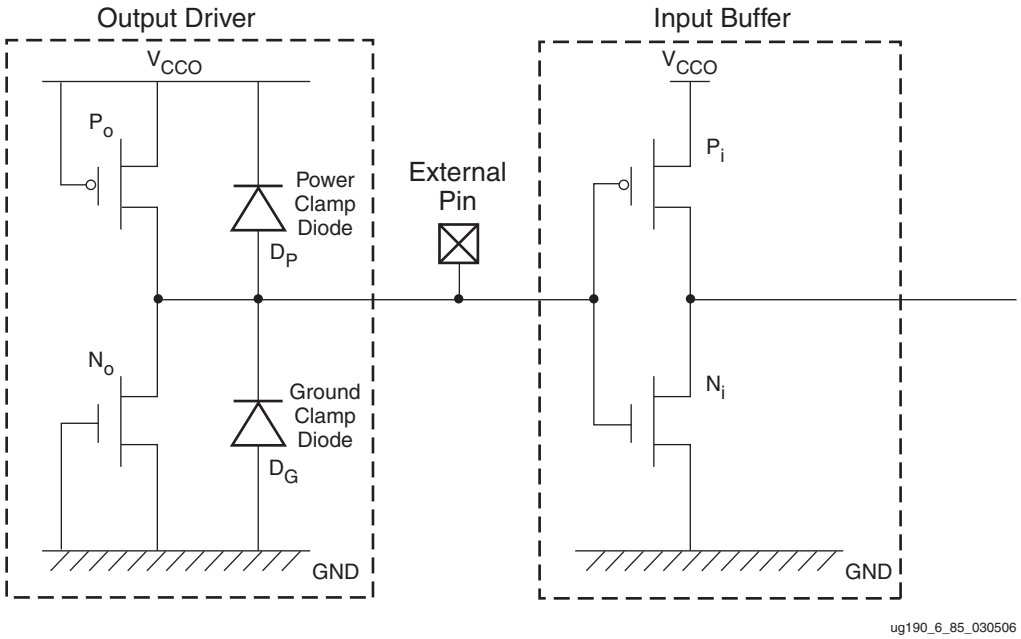


Figure 6-91: **Virtex-5 FPGA I/O: 3-State Output Driver**

The clamp diodes offer protection against transient voltage beyond approximately V_{CCO} + 0.5V and Ground – 0.5V. The voltage across the diode increases proportionally to the current going through it. Therefore the clamped level is not fixed and can vary

depending on the board design. The absolute maximum I/O limits might be exceeded even if the clamp diode is active.

The IBIS models contain the voltage-current characteristics of the I/O drivers and clamp diodes.

To verify overshoot and undershoot are within the I/O absolute maximum specifications, Xilinx recommends proper I/O termination and performing IBIS simulation.

Source Termination and LVDCI_33

In general, the I/O drivers should match the board trace impedance to within ±10% to minimize overshoot and undershoot. Source termination is often used for unidirectional interfaces. The DCI feature has built-in source termination on all user output pins. It compensates for impedance changes due to voltage and/or temperature fluctuations, and can match the reference resistor values. Assuming the reference resistor values are the same as the board trace impedance, the output impedance of the driver will closely match with the board trace.

The LVDCI_33 standard is used to enable the DCI features for 3.3V I/O operations. As shown in Figure 6-92, the OBUF_LVDCI_33 primitive is used to implement the source termination function in Virtex-5 FPGA output drivers. The pull-up resistor connected to VRN and the pull-down resistor connected to VRP determine the output impedance of all the output drivers in the same bank. The "Virtex-5 FPGA Digitally Controlled Impedance (DCI)" section has more details on using DCI.

Since the LVDCI_33 standard does not offer input termination, source termination must be implemented on the driver side. Figure 6-92 shows the recommended external source termination resistors to be incorporated on the external device side.

The total impedance of the LVTTL/LVCMOS driver added to the series termination resistor R_0 must match the board trace impedance ±10 percent to minimize overshoot and undershoot. An IBIS simulation is advised for calculating the exact value needed for R_0.

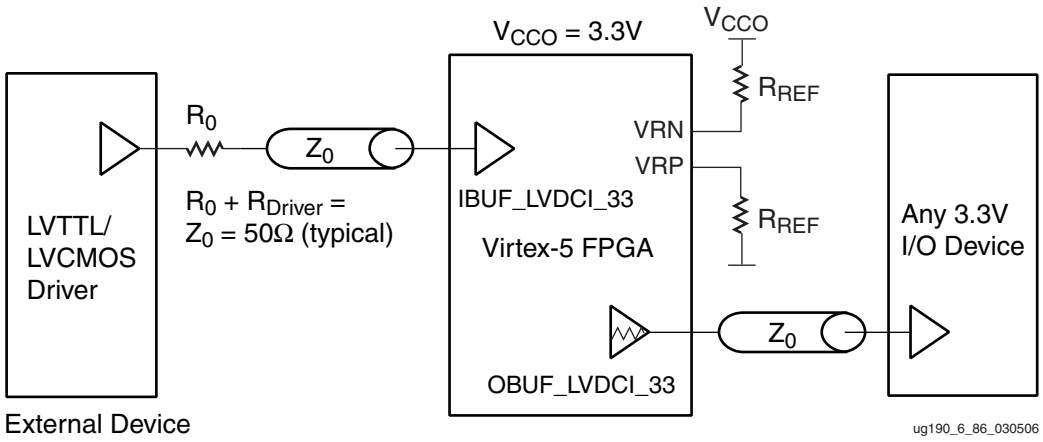


Figure 6-92: **Connecting LVTTL or LVCMOS Using the LVDCI_33 Standard**

The connection scheme shown in Figure 6-93 is for a bidirectional bus scenario. The signal performance may be degraded by R_0. Therefore, it is also recommended to verify the R_0 value and performance with an IBIS simulation.

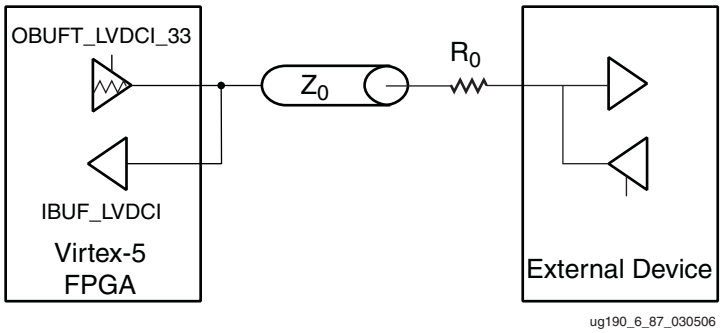


Figure 6-93: **3.3V I/O Configuration**

When designing with the LVDCI_33 standard:

- The output drive strength and slew rates are not programmable. The output impedance references the VRP and VRN resistors, and the output current is determined by the output impedance.
- If only LVDCI_33 inputs are used, it is not necessary to connect VRP and VRN to external reference resistors. The implementation pad report does not record VRP and VRN being used. External reference resistors are required only if LVDCI_33 outputs are present in a bank.
- LVDCI_33 is compatible with LVTTL and LVCMOS standards only.

In addition, changing the slew rate from fast to slow and/or reducing the current drive could significantly reduce overshoot and undershoot.

The *Virtex-5 FPGA PC Board Designers Guide* contains additional design information to assist PCB designers and signal integrity engineers.

Regulating V_{CCO} at 3.0V

The following section discusses alternatives for managing overshoot and undershoot for LVTTL, LVCMOS33, and PCI applications.

When V_{CCO} is lowered to 3.0V, the power clamp diode turns on at about 3.5V. Therefore it limits any overshoot higher than 3.5V before reaching the absolute maximum level of 4.05V. In addition, instead of –0.3V when V_{CCO} = 3.75V, the lower absolute maximum limit corresponding to V_{CCO} = 3.0V is –1.05V. In this case, the ground clamp diode clips undershoot before reaching the lower absolute maximum limit.

As a result, lowering V_{CCO} to 3.0V addresses the overshoot and undershoot specifications for all supported 3.3 V standards, including LVCMOS_33, LVTTL, LVDCI_33, and PCI.

Mixing Techniques

Either using LVDCI_33 standard or lowering the V_{CCO} to 3.0V is a good approach to address overshoot and undershoot. It is also acceptable to combine both methods. When V_{CCO} is lowered to 3.0V, it is not necessary to adjust the reference resistors VRP and VRN. The VRP and VRN values should always be the same as the board trace impedance.

Simultaneous Switching Output Limits

When multiple output drivers change state at the same time, power supply disturbance occurs. These disturbances can cause undesired transient behavior in output drivers, input receivers, or in internal logic. These disturbances are often referred to as Simultaneous-Switching Output (SSO) noise. The SSO limits govern the number and type of I/O output drivers that can be switched simultaneously while maintaining a safe level of SSO noise.

Sparse-Chevron Packages

Virtex-5 FPGA packaging utilizes a sparse-chevron pinout arrangement. The sparse-chevron pinout style is an improvement over previous designs, offering low crosstalk and SSO noise. The pinout is designed to minimize PDS inductance and keep I/O signal return current paths very closely coupled to their associated I/O signal.

The maximum ratio of I/O to reference pins (V_{CCO} and GND) in sparse-chevron packages is 4:1. For every four I/O pins, there is always at least one reference pin.

For boards that do not meet the nominal PCB requirements listed in "Nominal PCB Specifications," the Virtex-5 FPGA SSO calculator is available, containing all SSO limit data for all I/O standards. For designs in nominal PCBs mixing limited and "no limit" I/O standards, the Virtex-5 FPGA SSO calculator must be used to ensure that I/O utilization does not exceed the limit. Information on the calculator is available under the "Full Device SSO Calculator" section.

Unlike devices in previous families, Virtex-5 devices have only two bank sizes: 20 I/O and 40 I/O. With the ratio of signal to reference pins always constant, the SSO capacity of all banks of 20 I/O are the same, and the capacity of all banks of 40 I/O are the same. The SSO limits for Virtex-5 devices are listed on a per-bank basis rather than a limit per V_{CCO}/GND pair.

Nominal PCB Specifications

The nominal SSO table (Table 6-40) contains SSO limits for cases where the PCB parameters meet the following requirements.

Note: In cases where PCB parameters do not meet all requirements listed below, the Virtex-5 FPGA SSO Calculator *must* be used to determine the SSO limit, according to the physical factors of the unique PCB.

PCB Construction

- V_{CCO} and GND vias should have a drill diameter no less than 11 mils (279 µ).
- Total board thickness must be no greater than 62 mils (1575 µ).

Signal Return Current Management

- Traces must be referenced to a plane on an adjacent PCB layer.
- The reference plane must be either GND or the V_{CCO} associated with the output driver.
- The reference layer must remain uninterrupted for its full length from device to device.

Load Traces

- All IOB output buffers must drive controlled impedance traces with characteristic impedance of 50Ω ± 10%.
- Total capacitive loading at the far end of the trace (input capacitance of receiving device) must be no more than 10 pF.

Power Distribution System Design

- Designed according to the *Virtex-5 FPGA PC Board Designers Guide*.
  - Decoupling capacitors per the device guideline
  - Approved solder land patterns
- V_{CCO} and GND planes cannot be separated by more than 5.0 mils (152 µ)

Nominal SSO Limit

Table 6-40 provides the guidelines for the maximum number of simultaneously switching outputs allowed per bank to avoid the effects of ground bounce.

Table 6-40: **Maximum Number of Simultaneously Switching Outputs per Bank**

Voltage	IOSTANDARD	Limit per 20-pin Bank	Limit per 40-pin Bank
1.2V	HSTL_I_12	20	40
	LVCMOS12_2_slow	20	40
	LVCMOS12_4_slow	20	40
	LVCMOS12_6_slow	20	40
	LVCMOS12_8_slow	20	40
	LVCMOS12_2_fast	20	40
	LVCMOS12_4_fast	20	40
	LVCMOS12_6_fast	20	40
	LVCMOS12_8_fast	20	40

Table 6-40: **Maximum Number of Simultaneously Switching Outputs per Bank**

Voltage	IOSTANDARD	Limit per 20-pin Bank	Limit per 40-pin Bank
1.5V	LVCMOS15_2_slow	20	40
	LVCMOS15_4_slow	20	40
	LVCMOS15_6_slow	20	40
	LVCMOS15_8_slow	20	40
	LVCMOS15_12_slow	20	40
	LVCMOS15_16_slow	20	40
	LVCMOS15_2_fast	20	40
	LVCMOS15_4_fast	20	40
	LVCMOS15_6_fast	20	40
	LVCMOS15_8_fast	20	40
	LVCMOS15_12_fast	20	40
	LVCMOS15_16_fast	20	40
	LVDCI_15 50 Ω	20	40
	HSTL_I_15	20	40
	HSTL_I_15_DCI	20	40
	HSTL_II_15	20	40
	HSTL_II_15_DCI	20	40
	HSTL_III_15	20	40
	HSTL_III_15_DCI	20	40
	HSTL_IV_15	12	25
	HSTL_IV_15_DCI	12	25
	HSLVDCI_15 50 Ω	20	40
	DIFF_HSTL_I_15	20	40
	DIFF_HSTL_I_15_DCI	20	40
	DIFF_HSTL_II_15	20	40
	DIFF_HSTL_II_15_DCI	20	40

Table 6-40: **Maximum Number of Simultaneously Switching Outputs per Bank**

Voltage	IOSTANDARD	Limit per 20-pin Bank	Limit per 40-pin Bank
1.8V	LVCMOS18_2_slow	20	40
	LVCMOS18_4_slow	20	40
	LVCMOS18_6_slow	20	40
	LVCMOS18_8_slow	20	40
	LVCMOS18_12_slow	20	40
	LVCMOS18_16_slow	20	40
	LVCMOS18_2_fast	20	40
	LVCMOS18_4_fast	20	40
	LVCMOS18_6_fast	20	40
	LVCMOS18_8_fast	20	40
	LVCMOS18_12_fast	20	40
	LVCMOS18_16_fast	20	40
	LVDCI_18 50 Ω	20	40
	HSTL_I_18	20	40
	HSTL_I_DCI_18	20	40
	HSTL_II_18	20	40
	HSTL_II_DCI_18	20	40
	HSTL_III_18	17	35
	HSTL_III_DCI_18	17	35
	HSTL_IV_18	10	20
	HSTL_IV_DCI_18	10	20
	SSTL18_I	20	40
	SSTL18_I_DCI	20	40
	SSTL18_II	20	40
	SSTL18_II_DCI	20	40
	HSLVDCI_18 50 Ω	20	40
	DIFF_HSTL_I_18	20	40
	DIFF_HSTL_I_DCI_18	20	40
	DIFF_HSTL_II_18	20	40
	DIFF_HSTL_II_DCI_18	20	40
	DIFF_SSTL18_I	20	40
	DIFF_SSTL18_I_DCI	20	40
	DIFF_SSTL18_II	20	40
	DIFF_SSTL18_II_DCI	20	40

Table 6-40: **Maximum Number of Simultaneously Switching Outputs per Bank**

Voltage	IOSTANDARD	Limit per 20-pin Bank	Limit per 40-pin Bank
2.5V	LVCMOS25_2_slow	20	40
	LVCMOS25_4_slow	20	40
	LVCMOS25_6_slow	20	40
	LVCMOS25_8_slow	20	40
	LVCMOS25_12_slow	20	40
	LVCMOS25_16_slow	20	40
	LVCMOS25_24_slow	20	40
	LVCMOS25_2_fast	20	40
	LVCMOS25_4_fast	20	40
	LVCMOS25_6_fast	20	40
	LVCMOS25_8_fast	20	40
	LVCMOS25_12_fast	20	40
	LVCMOS25_16_fast	20	40
	LVCMOS25_24_fast	15	30
	LVDCI_25 50 Ω	20	40
	SSTL2_I	20	40
	SSTL2_I_DCI	20	40
	SSTL2_II	20	40
	SSTL2_II_DCI	20	40
	HSLVDCI_25 50 Ω	20	40
	DIFF_SSTL_I	20	40
	DIFF_SSTL_I_DCI	20	40
	DIFF_SSTL_II	20	40
	DIFF_SSTL_II_DCI	20	40
	LVPECL_25	20	40
	BLVDS_25	20	40
	LVDS_25	20	40
	LVDSEXT_25	20	40
	RSDS_25	20	40
	HT_25	20	40

Table 6-40: **Maximum Number of Simultaneously Switching Outputs per Bank**

Voltage	IOSTANDARD	Limit per 20-pin Bank	Limit per 40-pin Bank
3.3V	LVCMOS33_2_slow	20	40
	LVCMOS33_4_slow	20	40
	LVCMOS33_6_slow	20	40
	LVCMOS33_8_slow	20	40
	LVCMOS33_12_slow	20	40
	LVCMOS33_16_slow	20	40
	LVCMOS33_24_slow	20	40
	LVCMOS33_2_fast	20	40
	LVCMOS33_4_fast	20	40
	LVCMOS33_6_fast	20	40
	LVCMOS33_8_fast	20	40
	LVCMOS33_12_fast	20	40
	LVCMOS33_16_fast	20	40
	LVCMOS33_24_fast	15	30
	LVTTL_2_slow	20	40
	LVTTL_4_slow	20	40
	LVTTL_6_slow	20	40
	LVTTL_8_slow	20	40
	LVTTL_12_slow	20	40
	LVTTL_16_slow	20	40
	LVTTL_24_slow	20	40
	LVTTL_2_fast	20	40
	LVTTL_4_fast	20	40
	LVTTL_6_fast	20	40
	LVTTL_8_fast	20	40
	LVTTL_12_fast	20	40
	LVTTL_16_fast	20	40
	LVTTL_24_fast	15	30
	PCI33_3	20	40
	PCI66_3	20	40
	PCIX	20	40

Table 6-40: **Maximum Number of Simultaneously Switching Outputs per Bank**

Voltage	IOSTANDARD	Limit per 20-pin Bank	Limit per 40-pin Bank
3.3V	GTL	12	25
	GTL_DCI	12	25
	GTLP	12	25
	GTLP_DCI	12	25
	LVDCI_33 50 Ω	20	40
	HSLVDCI_33 50 Ω	20	40

Actual SSO Limits versus Nominal SSO Limits

The Virtex-5 FPGA SSO limits are defined for a set of nominal system conditions in Table 6-40. To compute the actual limits for a specific user's system, the "Parasitic Factors Derating Method (PFDM)" must be used. The PFDM allows the user to account for differences between actual and nominal PCB power systems, receiver capacitive loading, and maximum allowable ground bounce or V_{CC} bounce. A spreadsheet calculator, "Full Device SSO Calculator," automates this process.

Electrical Basis of SSO Noise

SSO noise can manifest as power supply disturbance, in the form of ground bounce or V_{CC} bounce. GND and V_{CC} bounce is a deviation of the die supply voltage (die GND rail or die V_{CC} rail) with respect to the voltage of the associated PCB supply (PCB GND rail or PCB V_{CC} rail). The deviation of die supplies from PCB supplies comes from the voltage induced across power system parasitics by supply current transients. One cause of current transients is output driver switching events. Numerous output switching events occurring at the same time lead to bigger current transients, and therefore bigger induced voltages (ground bounce, V_{CC} bounce, or rail collapse). Relevant transient current paths exist in the die, package, and PCB, therefore, parasitics from all three must be considered. The larger the value of these parasitics, the larger the voltage induced by a current transient (power-supply disturbance).

V_{CC} bounce affects stable high outputs. Ground bounce affects stable low outputs. Ground bounce also affects inputs configured as certain I/O standards because they interpret incoming signals by comparing them to a threshold referenced to the die ground (as opposed to I/O standards with input thresholds referenced to a V_{REF} voltage). If the die voltage disturbance exceeds the instantaneous noise margin for the interface, then a non-changing input or output can be erroneously interpreted as changing.

SSO noise can also manifest in the form of crosstalk between I/Os in close proximity to one another. The sparse chevron pinout of Virtex-5 devices reduces crosstalk in the pinout region to a minimum.

Parasitic Factors Derating Method (PFDM)

This section describes a method to evaluate whether a design is within the SSO limits when taking into account the specific electrical characteristics of the user's unique system.

The SSO limits in Table 6-40 assume nominal values for the parasitic factors of the system. These factors fall into three groups of electrical characteristics:

- PCB PDS parasitics (nominal 1 nH per via)
- Maximum allowable power system disturbance voltage (nominal 600 mV)
- Capacitive loading (nominal 10 pF per load)

When the electrical characteristics of a design differ from the nominal values, the system SSO limit changes. The degree of difference determines the new effective limit for the design. A figure called "SSO Allowance" is used as a single derating factor, taking into account the combined effect of all three groups of system electrical characteristics.

The SSO allowance is a number ranging from 0 to 100% and is a product of three scaling factors:

The *First Scaling Factor* accounts for the PCB PDS parasitic inductance. It is determined by dividing the nominal PCB PDS inductance by the user's PCB PDS inductance, L_{PDS_USR}. The PCB PDS inductance is determined based on a set of board geometries: board thickness, via diameter, breakout trace width and length, and any other additional structures including sockets.

The *Second Scaling Factor* accounts for the maximum allowable power system disturbance. It is determined by dividing the user's maximum allowable power system disturbance, ($V_{DISTURBANCE\_USER}$) by the nominal maximum power system disturbance. $V_{DISTURBANCE\_USER}$ is usually determined by taking the lesser of input undershoot voltage and input logic low threshold.

The *Third Scaling Factor* accounts for the capacitive loading of outputs driven by the FPGA. It is based on the transient current impact of every additional picofarad of load capacitance above the assumed nominal. For every additional 1 pF of load capacitance over the nominal, approximately 9 mV of additional power system disturbance will occur. The additional power system disturbance is compared to the nominal power system disturbance, and a scale factor is derived from the relationship. C_{LOAD_USER} is the user's average load capacitance.

Example calculations show how each scale factor is computed, as well as the SSO allowance. The system parameters used in this example are:

L_{PDS_USER} = 1.1 nH

$V_{DISTURBANCE\_USER}$ = 550 mV

C_{LOAD_USER} = 22 pF

First Scaling Factor (SF1) $= L_{PDS\_NOM}/L_{PDS\_USER}$
$= 1.0\text{ nH}/1.1\text{ nH}$
$= 0.909$

Second Scaling Factor (SF2) $= V_{DISTURBANCE\_USER}/V_{DISTURBANCE\_NOM}$
$= 550\text{ mV}/600\text{ mV}$
$= 0.917$

Third Scaling Factor (SF3)

$= V_{DISTURBANCE\_NOM}/((C_{LOAD\_USER} - C_{LOAD\_NOM}) \times 9\text{ mV/pF}) + V_{DISTURBANCE\_NOM}$

$= 600\text{ mV}/((22\text{ pF} - 15\text{ pF}) \times 9\text{ mV/pF}) + 600\text{ mV}$

$= 600\text{ mV}/663\text{ mV}$

$= 0.905$

SSO Allowance = SF1 × SF2 × SF3 × 100%

= 0.909 × 0.917 × 0.905 × 100%

= 75.4%

Weighted Average Calculation of SSO

This section describes the SSO calculation where the SSO contributions of all I/O in a bank are combined into a single figure.

SSO of an individual bank is calculated by summing the SSO contributions of the individual I/O standards in the bank. The SSO contribution is the percentage of full utilization of any one I/O standard in any one bank. For drivers of each I/O standard, the calculation follows:

SSO Contribution (I/ O group n) = (quantity of drivers)/(Bank SSO limit)

For a bank with drivers of multiple I/O standards, the SSO calculation is:

$$Bank\ SSO = \sum_{(1\ to\ n)} SSO\ Contribution(n)$$

A sample SSO calculation follows. The system parameters used are:

Device: XC5VLX50 FF1153

Bank: 11

I/O Standards, Quantities:

SSTL2_II, 12

LVCMOS25_24 Fast, 6

LVCMOS25_6 Fast, 19

First, SSO limits for each I/O standard are obtained from Table 6-40:

I/O Group	I/O Standard	SSO Limit (Drivers per Bank)
1	SSTL2_II	40
2	LVCMOS25_24 Fast	30
3	LVCMOS25_6 Fast	40

The SSO contribution of each I/O standard is calculated as:

SSO Contribution = (quantity of drivers)/(Bank SSO limit)

SSO Contribution (1) = 12/40 = 30%

SSO Contribution (2) = 6/30 = 20%

SSO Contribution (3) = 19/40 = 48%

Finally, the bank SSO is calculated:

Bank 1 SSO = SSO contribution (1) + SSO contribution (2) + SSO Contribution (3)

= 30% + 20% + 48% = 98%

Full Device SSO Calculator

A Microsoft Excel-based spreadsheet, the Virtex-5 FPGA SSO Calculator, automates all the PFDM and SSO calculations. The Virtex-5 FPGA SSO calculator uses PCB geometry, (board thickness, via diameter, and breakout trace width and length) to determine power system inductance. It determines the smallest undershoot and logic-low threshold voltage among all input devices, calculates the average output capacitance, and determines the SSO allowance by taking into account all of the board-level design parameters mentioned in this document. In addition, the Virtex-5 FPGA SSO calculator checks the adjacent bank and package SSO ensuring the full device design does not exceed the SSO allowance. Since bank-number assignment for Virtex-5 devices is different from package to package due to its columnar architecture (versus the peripheral I/O architecture of previous devices), there is a separate tab at the bottom of the SSO calculator display for each Virtex-5 FPGA package. This customizing allows for the arrangement of physically adjacent banks (as they appear clockwise on each unique package, even though they are not labeled in a contiguous manner), and the hard-coding of the number of V_{CCO}/GND pairs per bank.

The Virtex-5 FPGA SSO Calculator file (ug190_SSO_Calculator.zip) is available at: https://secure.xilinx.com/webreg/clickthrough.do?cid=30154.

Other SSO Assumptions

LVDCI and HSLVDCI Drivers

All limits for controlled impedance DCI I/O standards assume a 50 Ω output impedance. For higher reference resistor (RR) values, less drive strength is needed, and the SSO limit increases linearly. To calculate the SSO limit for a controlled impedance driver with different reference resistors, the following formula is used:

$$User\ SSO = \left(\frac{User\ RR}{50\Omega}\Omega\right)(SSO\ Limit\ for\ \Omega)$$

Example

The designer uses LVDCI_18 driver with 65 Ω reference resistors. The LVDCI_18 SSO limit for 50 Ω impedance is first taken from Table 6-40. The SSO limit for LVDCI_18 at 50 Ω is 11 SSO per V_{CCO}/GND pin pair. Therefore, the SSO limit for LVDCI_18 at 65 Ω is:

SSO Limit LVDCI_18 at 65 Ω = ((65 Ω)/50 Ω) × 11 = 14.3

Bank 0

Bank 0 in all devices contains only configuration and dedicated signals. Since there is no user I/O in Bank 0, no SSO analysis is necessary for this bank.

Chapter 7

SelectIO Logic Resources

Introduction

This chapter describes the logic directly behind the I/O drivers and receivers covered in Chapter 6, "SelectIO Resources."

Virtex-5 FPGAs contain all of the basic I/O logic resources from Virtex-II/Virtex-II Pro FPGAs. These resources include the following:

- Combinatorial input/output
- 3-state output control
- Registered input/output
- Registered 3-state output control
- Double-Data-Rate (DDR) input/output
- DDR output 3-state control

In addition, Virtex-5 FPGAs implement the following architectural features that are also supported in Virtex-4 FPGAs:

- IODELAY provides users control of an adjustable, fine-resolution delay element
- SAME_EDGE output DDR mode
- SAME_EDGE and SAME_EDGE_PIPELINED input DDR mode

ILOGIC Resources

The ILOGIC block shown in Figure 7-1.

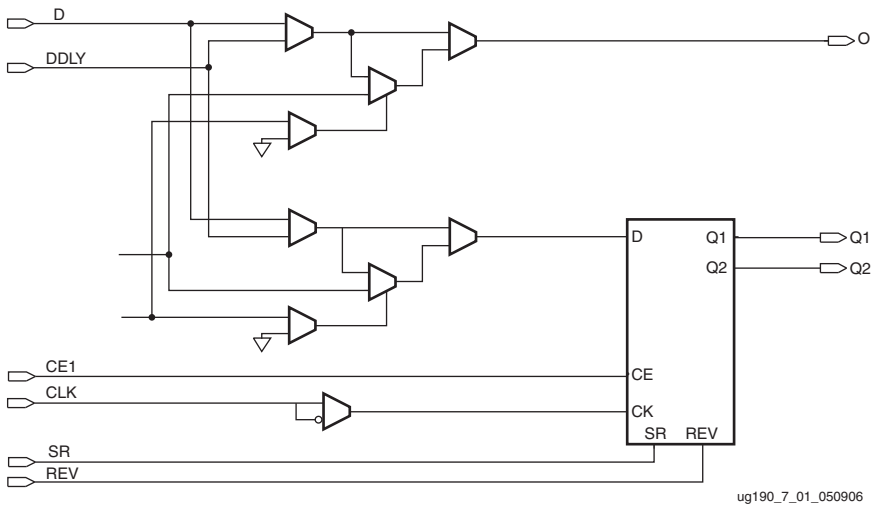


Figure 7-1: **ILOGIC Block Diagram**

ILOGIC can support the following operations:

- Edge-triggered D-type flip-flop
- IDDR mode (OPPOSITE_EDGE or SAME_EDGE or SAME_EDGE_PIPELINED). See "Input DDR Overview (IDDR)," page 317 for further discussion on input DDR.
- Level sensitive latch
- Asynchronous/combinatorial

All ILOGIC block registers have a common clock enable signal (CE1) that is active High by default. If left unconnected, the clock enable pin for any storage element defaults to the active state.

All ILOGIC block registers have a common synchronous or asynchronous set and reset (SR and REV signals). The set/reset input pin, SR forces the storage element into the state specified by the SRVAL attributes. When using SR, a second input, REV forces the storage element into the opposite state. The reset condition predominates over the set condition. Table 7-1 and Table 7-2 describe the operation of SR in conjunction with REV.

Table 7-1: **Truth Table when SRVAL = 0 (Default Condition)**

SR	REV	Function
0	0	NOP
0	1	Reset
1	0	Set
1	1	Reset

Table 7-2: **Truth Table when SRVAL = 1**

SR	REV	Function
0	0	NOP
0	1	Set
1	0	Reset
1	1	Reset

The SRVAL attributes can be set individually for each storage element in the ILOGIC block, but the choice of synchronous or asynchronous set/reset (SRTYPE) can not be set individually for each storage element in the ILOGIC block.

The following sections discuss the various resources within the ILOGIC blocks. All connections between the ILOGIC resources are managed in Xilinx software.

Combinatorial Input Path

The combinatorial input path is used to create a direct connection from the input driver to the FPGA fabric. This path is used by software automatically when:

1. There is a direct (unregistered) connection from input data to logic resources in the FPGA fabric.
2. The "pack I/O register/latches into IOBs" is set to OFF.

Input DDR Overview (IDDR)

Virtex-5 devices have dedicated registers in the ILOGIC to implement input double-data-rate (DDR) registers. This feature is used by instantiating the IDDR primitive.

There is only one clock input to the IDDR primitive. Falling edge data is clocked by a locally inverted version of the input clock. All clocks feeding into the I/O tile are fully multiplexed, i.e., there is no clock sharing between ILOGIC and OLOGIC blocks. The IDDR primitive supports the following modes of operation:

- OPPOSITE_EDGE mode
- SAME_EDGE mode
- SAME_EDGE_PIPELINED mode

The SAME_EDGE and SAME_EDGE_PIPELINED modes are the same as for the Virtex-4 architecture. These modes allow designers to transfer falling edge data to the rising edge domain within the ILOGIC block, saving CLB and clock resources, and increasing performance. These modes are implemented using the DDR_CLK_EDGE attribute. The following sections describe each of the modes in detail.

OPPOSITE_EDGE Mode

A traditional input DDR solution, or OPPOSITE_EDGE mode, is accomplished via a single input in the ILOGIC. The data is presented to the fabric via the output Q1 on the rising edge of the clock and via the output Q2 on the falling edge of the clock. This structure is similar to the Virtex-II, Virtex-II Pro, and Virtex-4 FPGA implementation. Figure 7-2 shows the timing diagram of the input DDR using the OPPOSITE_EDGE mode.

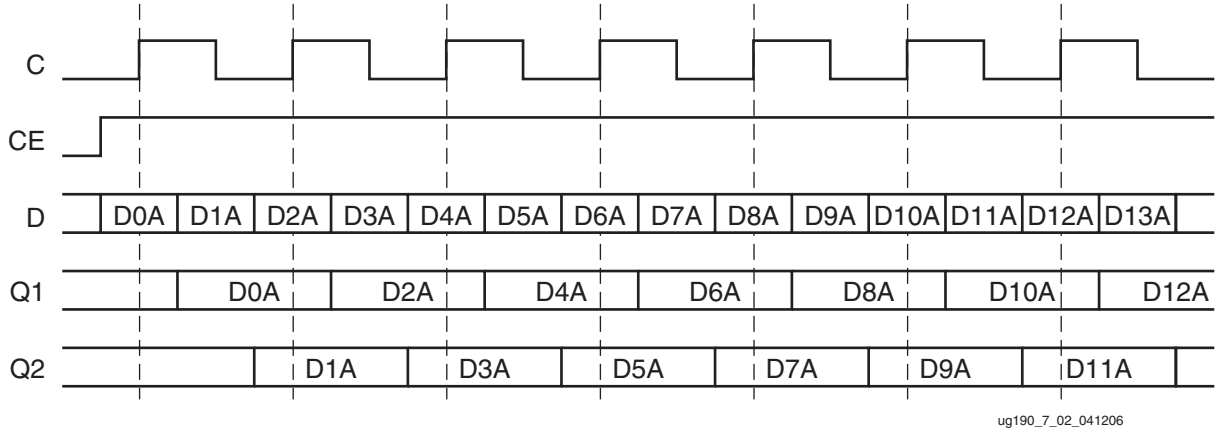


Figure 7-2: **Input DDR Timing in OPPOSITE_EDGE Mode**

SAME_EDGE Mode

In the SAME_EDGE mode, the data is presented into the FPGA fabric on the same clock edge. However, the data pair to be separated by one clock cycle. This structure is similar to the Virtex-II, Virtex-II Pro, and Virtex-4 FPGA implementation.

Figure 7-3 shows the timing diagram of the input DDR using SAME_EDGE mode. In the timing diagram, the output pairs Q1 and Q2 are no longer (0) and (1). Instead, the first pair presented is pair Q1 and Q2 (0) and (don't care) respectively, followed by pair (1) and (2) on the next clock cycle.

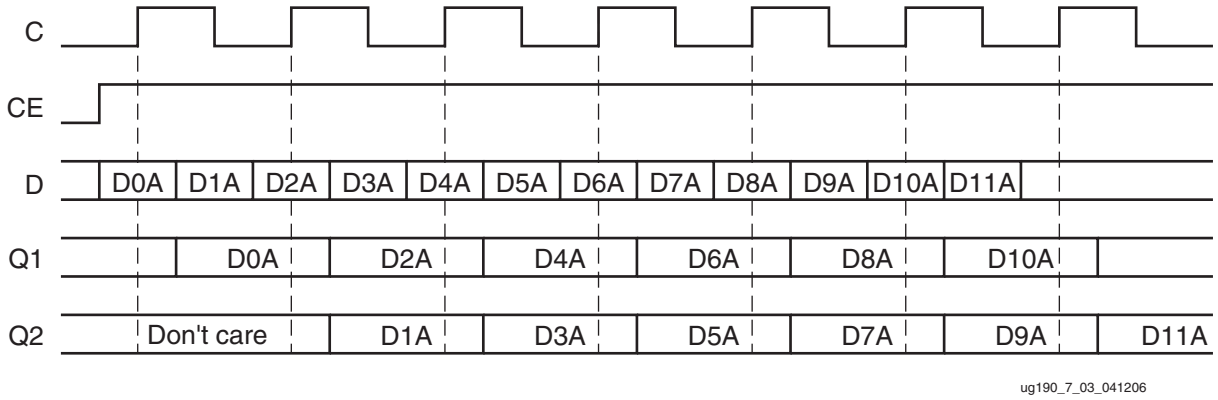


Figure 7-3: **Input DDR Timing in SAME_EDGE Mode**

SAME_EDGE_PIPELINED Mode

In the SAME_EDGE_PIPELINED mode, the data is presented into the FPGA fabric on the same clock edge.

Unlike the SAME_EDGE mode, the data pair is not separated by one clock cycle. However, an additional clock latency is required to remove the separated effect of the SAME_EDGE mode. Figure 7-4 shows the timing diagram of the input DDR using the SAME_EDGE_PIPELINED mode. The output pairs Q1 and Q2 are presented to the FPGA fabric at the same time.

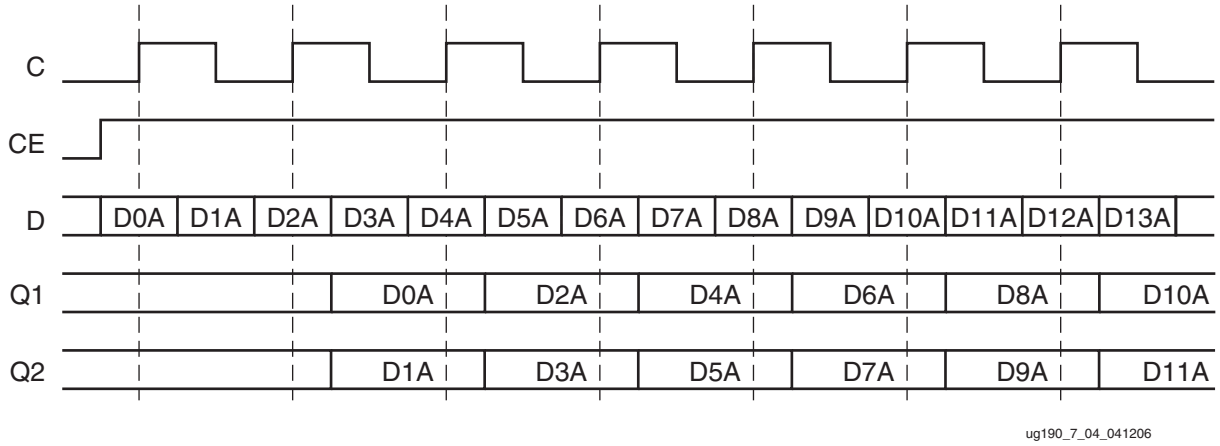


Figure 7-4: **Input DDR Timing in SAME_EDGE_PIPELINED Mode**

Input DDR Primitive (IDDR)

Figure 7-5 shows the block diagram of the IDDR primitive. Table 7-3 lists the IDDR port signals. Table 7-4 describes the various attributes available and default values for the IDDR primitive.

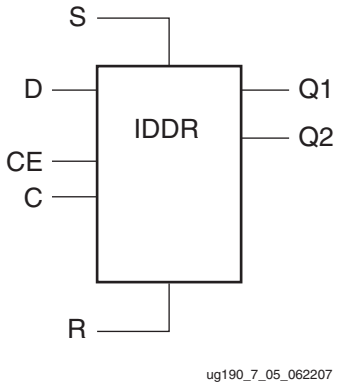


Figure 7-5: **IDDR Primitive Block Diagram**

Table 7-3: **IDDR Port Signals**

Port Name	Function	Description
Q1 and Q2	Data outputs	IDDR register outputs.
C	Clock input port	The C pin represents the clock input pin.
CE	Clock enable port	The enable pin affects the loading of data into the DDR flip-flop. When Low, clock transitions are ignored and new data is not loaded into the DDR flip-flop. CE must be High to load new data into the DDR flip-flop.
D	Data input (DDR)	IDDR register input from IOB.

Table 7-3: **IDDR Port Signals *(Continued)***

Port Name	Function	Description
R	Reset	Synchronous/Asynchronous reset pin. Reset is asserted High.
S	Set	Synchronous/Asynchronous set pin. Set is asserted High.

Table 7-4: **IDDR Attributes**

Attribute Name	Description	Possible Values
DDR_CLK_EDGE	Sets the IDDR mode of operation with respect to clock edge	OPPOSITE_EDGE (default), SAME_EDGE, SAME_EDGE_PIPELINED
INIT_Q1	Sets the initial value for Q1 port	0 (default), 1
INIT_Q2	Sets the initial value for Q2 port	0 (default), 1
SRTYPE	Set/Reset type with respect to clock (C)	ASYNC (default), SYNC

IDDR VHDL and Verilog Templates

The Libraries Guide includes templates for instantiation of the IDDR primitive in VHDL and Verilog.

ILOGIC Timing Models

This section describes the timing associated with the various resources within the ILOGIC block.

ILOGIC Timing Characteristics

Figure 7-6 illustrates ILOGIC register timing. When IDELAY is used, T_{IDOCK} is replaced by T_{IDOCKD}.

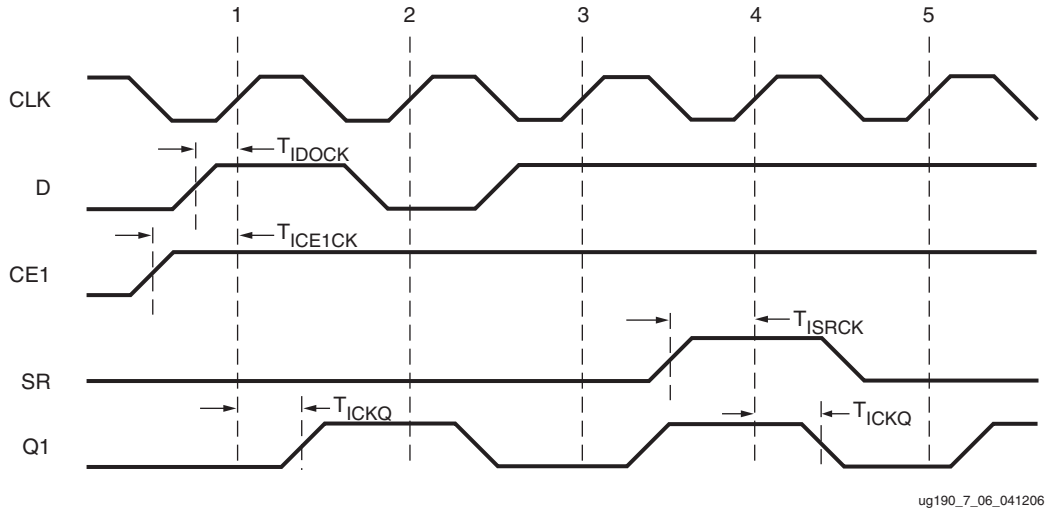


Figure 7-6: **ILOGIC Input Register Timing Characteristics**

Clock Event 1

- At time T_{ICE1CK} before Clock Event 1, the input clock enable signal becomes valid-high at the CE1 input of the input register, enabling the input register for incoming data.
- At time T_{IDOCK} before Clock Event 1, the input signal becomes valid-high at the D input of the input register and is reflected on the Q1 output of the input register at time T_{ICKQ} after Clock Event 1.

Clock Event 4

- At time T_{ISRCK} before Clock Event 4, the SR signal (configured as synchronous reset in this case) becomes valid-high resetting the input register and reflected at the Q1 output of the IOB at time T_{ICKQ} after Clock Event 4.

ILOGIC Timing Characteristics, DDR

Figure 7-7 illustrates the ILOGIC in IDDR mode timing characteristics. When IDELAY is used, T_{IDOCK} is replaced by T_{IDOCKD}. The example shown uses IDDR in OPPOSITE_EDGE mode. For other modes, add the appropriate latencies as shown in Figure 7-4, page 319.

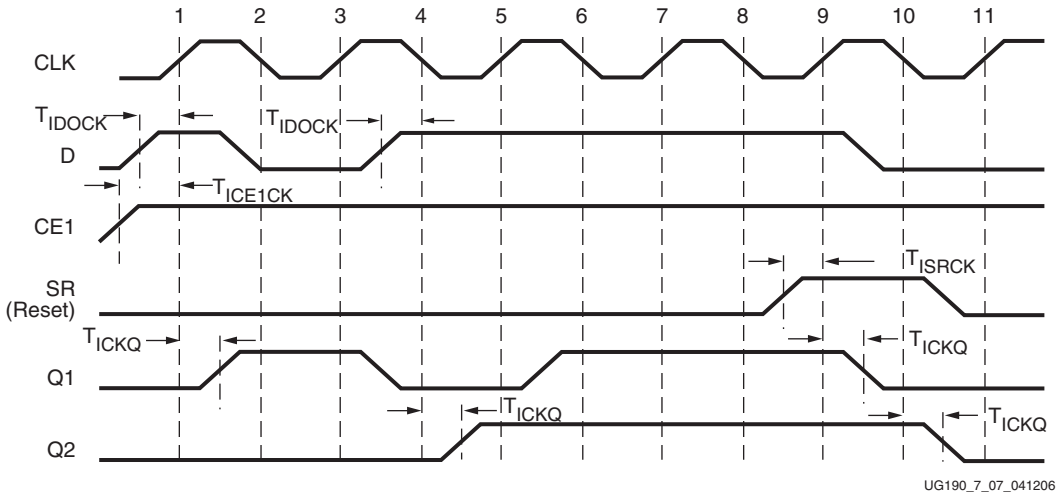


Figure 7-7: **ILOGIC in IDDR Mode Timing Characteristics (OPPOSITE_EDGE Mode)**

Clock Event 1

- At time T_{ICE1CK} before Clock Event 1, the input clock enable signal becomes valid-high at the CE1 input of both of the DDR input registers, enabling them for incoming data. Since the CE1 and D signals are common to both DDR registers, care must be taken to toggle these signals between the rising edges and falling edges of CLK as well as meeting the register setup-time relative to both clocks.
- At time T_{IDOCK} before Clock Event 1 (rising edge of CLK), the input signal becomes valid-high at the D input of both registers and is reflected on the Q1 output of input-register 1 at time T_{ICKQ} after Clock Event 1.

Clock Event 2

- At time T_{IDOCK} before Clock Event 2 (falling edge of CLK), the input signal becomes valid-low at the D input of both registers and is reflected on the Q2 output of input-register 2 at time T_{ICKQ} after Clock Event 2 (no change in this case).

Clock Event 9

- At time T_{ISRCK} before Clock Event 9, the SR signal (configured as synchronous reset in this case) becomes valid-high resetting Q1 at time T_{ICKQ} after Clock Event 9, and Q2 at time T_{ICKQ} after Clock Event 10.

Table 7-5 describes the function and control signals of the ILOGIC switching characteristics in the *Virtex-5 FPGA Data Sheet*.

Table 7-5: **ILOGIC Switching Characteristics**

Symbol	Description
Setup/Hold	
T_{ICE1CK}/T_{ICKCE1}	CE1 pin Setup/Hold with respect to CLK
T_{ISRCK}/T_{ICKSR}	SR/REV pin Setup/Hold with respect to CLK
T_{IDOCK}/T_{IOCKD}	D pin Setup/Hold with respect to CLK
Combinatorial	
T_{IDI}	D pin to O pin propagation delay, no Delay
Sequential Delays	
T_{IDLO}	D pin to Q1 pin using flip-flop as a latch without Delay
T_{ICKQ}	CLK to Q outputs
T_{ICE1Q}	CE1 pin to Q1 using flip-flop as a latch, propagation delay
T_{RQ}	SR/REV pin to OQ/TQ out

Note: The DDLY timing diagrams and parameters are identical to the D timing diagrams and parameters.

Input/Output Delay Element (IODELAY)

Every I/O block contains a programmable absolute delay element called IODELAY. The IODELAY can be connected to an ILOGIC/ISERDES or OLOGIC/OSERDES block or both. IODELAY is a 64-tap, wraparound, delay element with a calibrated tap resolution. See the *Virtex-5 FPGA Data Sheet*. It can be applied to the combinatorial input path, registered input path, combinatorial output path, or registered output path. It can also be accessed directly in the fabric. IODELAY allows incoming signals to be delayed on an individual basis. The tap delay resolution is varied by selecting an IDELAYCTRL reference clock from the range specified in the *Virtex-5 FPGA Data Sheet*. The IODELAY resource can function as IDELAY, ODELAY, or bidirectional delay.

When used as IDELAY, the data input comes from either IBUF or the fabric and the output goes to ILOGIC/ISERDES. There are three modes of operation available:

- Zero-hold time delay mode (IDELAY_TYPE = DEFAULT)

  This mode of operation allows backward compatibility for designs using the zero-hold time delay feature in Virtex-II, Virtex-II Pro and Virtex-4 devices. This delay element is used to provide non-positive hold times when global clocks are used without DCMs to capture data (pin-to-pin parameters). When used in this mode, the IDELAYCTRL primitive does not need to be instantiated. See IDELAYCTRL Usage and Design Guidelines for more details.

- Fixed delay mode (IDELAY_TYPE = FIXED)

  In the fixed delay mode, the delay value is preset at configuration to the tap number determined by the attribute IDELAY_VALUE. Once configured, this value cannot be changed. When used in this mode, the IDELAYCTRL primitive must be instantiated. See IDELAYCTRL Usage and Design Guidelines for more details.

- Variable delay mode (IDELAY_TYPE = VARIABLE)

  In the variable delay mode, the delay value can be changed after configuration by manipulating the control signals CE and INC. When used in this mode, the IDELAYCTRL primitive must be instantiated. See IDELAYCTRL Usage and Design Guidelines for more details.

When used as ODELAY, the data input comes from OLOGIC/OSERDES and the data output goes to OBUF. There is a single mode of operation:

- Fixed delay output mode

  In the fixed delay output mode, the delay value is preset at configuration to the tap number determined by the attribute ODELAY_VALUE. Once configured, this value cannot be changed. When used in this mode, the IDELAYCTRL primitive must be instantiated. See IDELAYCTRL Usage and Design Guidelines for more details.

When used as bidirectional delay, the IOB is configured in bidirectional mode. IODELAY alternately delays the data on the input path and output path. There are two modes of operation:

- Fixed IDELAY (IDELAY_TYPE = FIXED) and fixed ODELAY mode

  In this mode, both the values for IDELAY and ODELAY are preset at configuration and are determined by the IDELAY_VALUE and ODELAY_VALUE attributes. Once configured, this value cannot be changed. When used in this mode, the IDELAYCTRL primitive must be instantiated. See IDELAYCTRL Usage and Design Guidelines for more details.

- Variable IDELAY (IDELAY_TYPE = VARIABLE) and fixed ODELAY mode

In this mode, only the IDELAY value can be dynamically changed after configuration by manipulating the control signals CE and INC. The logic level of the T pin in the IODELAY primitive dynamically determines if the block is in IDELAY or ODELAY mode. When used in this mode, the IDELAYCTRL primitive must be instantiated. See IDELAYCTRL Usage and Design Guidelines for more details.

Figure 7-6 lists the supported IODELAY configurations.

Table 7-6: **IODELAY Configurations Supported**

IODELAY Mode	Direction of IODELAY	Input Pin Used in the IODELAY Element	Source	Destination	Supported Delay Modes
IDELAY	I	IDATAIN	IBUF	ILOGIC/ISERDES/Fabric	Default/Fixed/Variable
		DATAIN	Fabric		Fixed/Variable
ODELAY	O	ODATAIN	OLOGIC/OSERDES	OBUF	Fixed
Bidirectional Delay	I (when T = 1)	IDATAIN	IBUF	ILOGIC/ISERDES/Fabric	Fixed/Variable
	O (when T = 0)	ODATAIN	OLOGIC/OSERDES	OBUF	Fixed

IODELAY Primitive

Figure 7-8 shows the IODELAY primitive.

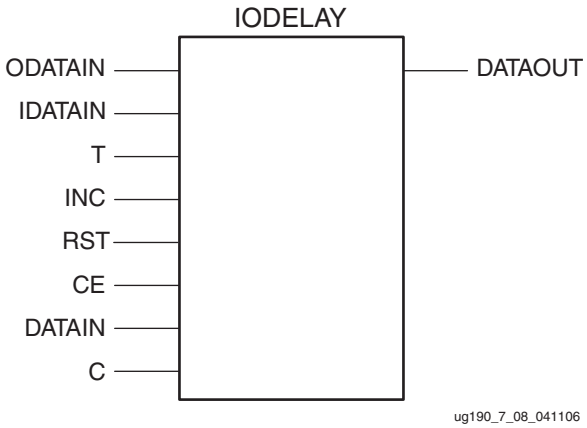


Figure 7-8: **IODELAY Primitive**

Table 7-7 lists the available ports in the IODELAY primitive. All ports are 1-bit wide.

Table 7-7: **IODELAY Primitive Ports**

Port Name	Direction	Function
DATAOUT	Output	Delayed data from one of three data input ports (IDATAIN, ODATAIN, DATAIN)
IDATAIN	Input	Data input for IODELAY from the IOB.
ODATAIN	Input	Data input for IODELAY from the OSERDES/OLOGIC
DATAIN	Input	Data input for IODELAY from the FPGA fabric

Table 7-7: **IODELAY Primitive Ports *(Continued)***

Port Name	Direction	Function
T	Input	3-state input control port. This port determines dynamically if IODELAY is used as IDELAY or ODELAY.
CE	Input	Enable increment/decrement function
INC	Input	Increment/decrement number of tap delays
RST	Input	Reset the IODELAY element to the pre-programmed value
C	Input	Clock input used in variable mode

IODELAY Ports

Data Input from the IOB - IDATAIN

The IDATAIN input is driven by its associated IOB. In IDELAY mode the data can be driven to either an ILOGIC/ISERDES block, directly into the FPGA fabric, or to both through the DATAOUT port with a delay set by the IDELAY_VALUE.

Data Input from the FPGA Fabric - ODATAIN

The ODATAIN input is driven by OLOGIC/OSERDES. In ODELAY mode, the ODATAIN drives the DATAOUT port which is connected to an IOB with a delay set by the ODELAY_VALUE.

Data Input for IODELAY from the FPGA Fabric - DATAIN

The DATAIN input is directly driven by the FPGA fabric providing a fabric logic accessible delay line. The data is driven back into the fabric through the DATAOUT port with a delay set by the IDELAY_VALUE. DATAIN can be locally inverted. The data cannot be driven to an IOB.

Data Output - DATAOUT

Delayed data from the three data input ports. DATAOUT connects to the fabric (IDELAY mode), or an IOB (ODELAY mode) or both (bidirectional delay mode). If used in the bidirectional delay mode, the T port dynamically switches between the IDATAIN and ODATAIN paths providing an alternating input/output delay based on the direction indicated by the 3-state signal T from the OLOGIC block.

3-state Input - T

This is the 3-state input control port. For bidirectional operation, the T pin signal also controls the T pin of the OBUFT.

Clock Input - C

All control inputs to IODELAY primitive (RST, CE, and INC) are synchronous to the clock input (C). A clock must be connected to this port when IODELAY is configured in variable mode. C can be locally inverted, and must be supplied by a global or regional clock buffer. This clock should be connected to the same clock in the SelectIO logic resources (when using ISERDES and OSERDES, C is connected to CLKDIV).

Module Reset - RST

The IODELAY reset signal, RST, resets the delay element to a value set by the IDELAY_VALUE or ODELAY_VALUE attribute. If these attributes are not specified, a value of zero is assumed. The RST signal is an active-High reset and is synchronous to the input clock signal (C).

The control pins are summarized in Table 7-8.

Table 7-8: **Control Pin Descriptions**

Pin	Type	Value	Description
INC	Input	1	Increment/decrement number of tap delays
CE	Input	1	Enable increment/decrement function
RST	Input	1	Reset delay element to pre-programmed value. If no value programmed, reset to 0

Increment/Decrement Signals - CE, INC

The increment/decrement is controlled by the enable signal (CE). This interface is only available for the IDELAY mode, when IDELAY_TYPE = VARIABLE.

As long as CE remains High, IDELAY will increment or decrement by $T_{IDELAYRESOLUTION}$ every clock (C) cycle. The state of INC determines whether IDELAY will increment or decrement; INC = 1 increments, INC = 0 decrements, synchronously to the clock (C). If CE is Low the delay through IDELAY will not change regardless of the state of INC.

When CE goes High, the increment/decrement operation begins on the next positive clock cycle. When CE goes Low, the increment/decrement operation ceases on the next positive clock cycle.

IODELAY is a wrap-around programmable delay element. When the end of the delay element is reached (tap 63) a subsequent increment function will return to tap 0. The same applies to the decrement function: decrementing below zero moves to tap 63. The increment/decrement operation is summarized in Table 7-9.

Table 7-9: **Increment/Decrement Operations**

Operation	RST	CE	INC
Reset to IDELAY_VALUE	1	x	x
Increment tap count	0	1	1
Decrement tap count	0	1	0
No change	0	0	x

Notes:

1. RST takes precedence over CE and INC.

IODELAY Attributes

Table 7-10 summarizes the IODELAY attributes.

Table 7-10: **IODELAY Attribute Summary**

Attribute	Value	Default Value	Description
IDELAY_TYPE	String: DEFAULT, FIXED, or VARIABLE	DEFAULT	Sets the type of tap delay line. Default delay is used to guarantee zero hold times, fixed delay is used to set a static delay value, and variable delay is used to dynamically adjust the delay value.
IDELAY_VALUE	Integer: 0 to 63	0	Specifies the fixed number of delay taps in fixed mode or the initial starting number of taps in variable mode (input path).
ODELAY_VALUE	Integer: 0 to 63	0	Specifies the fixed number of delay taps (output path).
HIGH_PERFORMANCE_MODE	Boolean: FALSE, TRUE	TRUE	When TRUE, this attribute reduces the output jitter. The difference in power consumption is quantified in the Xilinx Power Estimator tool.
SIGNAL_PATTERN	String: DATA, CLOCK	DATA	The SIGNAL_PATTERN attribute causes the timing analyzer to account for the appropriate amount of delay-chain jitter in the data or clock path.
REFCLK_FREQUENCY	Real: 190.0 to 210.0	200	IDELAYCTRL reference clock frequency (MHz).
DELAY_SRC	String: I, O, IO, or DATAIN	DATAIN	I: IODELAY chain input is IDATAIN O: IODELAY chain input is ODATAIN IO: IODELAY chain input is IDATAIN and ODATAIN (controlled by T) DATAIN: IODELAY chain input is DATAIN

IDELAY_TYPE Attribute

The IDELAY_TYPE attribute sets the type of delay used. The attribute values are DEFAULT, FIXED, and VARIABLE. When set to DEFAULT, the zero-hold time delay element is selected. This delay element is used to guarantee non-positive hold times when global clocks are used without DCMs to capture data (pin-to-pin parameters).

When set to FIXED, the tap-delay value is fixed at the number of taps determined by the IDELAY_VALUE attribute setting. This value is preset and cannot be changed after configuration.

When set to VARIABLE, the variable tap delay element is selected. The tap delay can be incremented by setting CE = 1 and INC = 1, or decremented by CE = 1 and INC = 0. The increment/decrement operation is synchronous to C, the input clock signal.

IDELAY_VALUE Attribute

The IDELAY_VALUE attribute specifies the initial number of tap delays. The possible values are any integer from 0 to 63. The default value is zero. The value of the tap delay reverts to IDELAY_VALUE when the tap delay is reset. In variable mode this attribute determines the initial setting of the delay line.

ODELAY_VALUE Attribute

The ODELAY_VALUE attribute specifies tap delays. The possible values are any integer from 0 to 63. The default value is zero. The value of the tap delay reverts to ODELAY_VALUE when the tap delay is reset.

HIGH_PERFORMANCE_MODE Attribute

When TRUE, this attribute reduces the output jitter. This reduction results in a slight increase in power dissipation from the IODELAY element. When set to FALSE the IODELAY element consumes less power.

SIGNAL_PATTERN Attribute

Clock and data signals have different electrical profiles and therefore accumulate different amounts of jitter in the IODELAY chain. By setting the SIGNAL_PATTERN attribute, the user enables timing analyzer to account for jitter appropriately when calculating timing. A clock signal is periodic in nature and does not have long sequences of consecutive ones or zeroes, while data is random in nature and can have long and short sequences of ones and zeroes.

IODELAY Timing

Table 7-11 shows the IODELAY switching characteristics.

Table 7-11: **IODELAY Switching Characteristics**

Symbol	Description
$T_{IDELAYRESOLUTION}$	IDELAY tap resolution
T_{ICECK}/T_{ICKCE}	CE pin Setup/Hold with respect to C
T_{IINCCK}/T_{ICKINC}	INC pin Setup/Hold with respect to C
T_{IRSTCK}/T_{ICKRST}	RST pin Setup/Hold with respect to C

Figure 7-9 shows an IDELAY timing diagram. It is assumed that IDELAY_VALUE = 0.

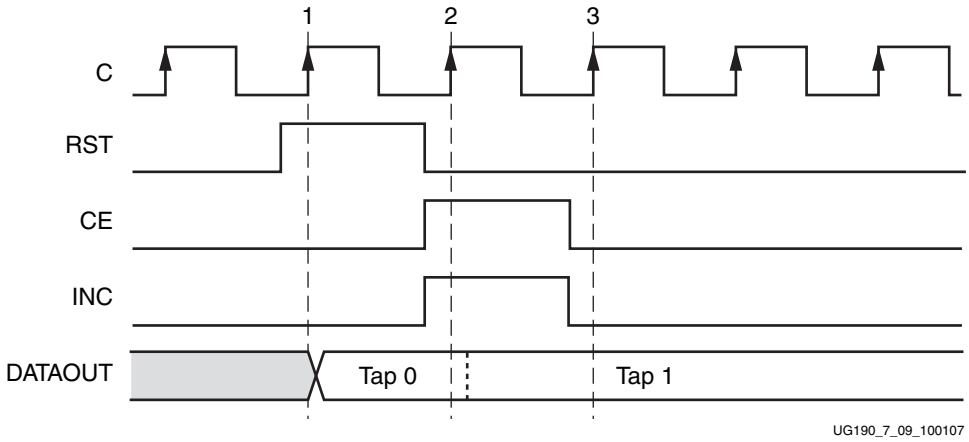


Figure 7-9: **IDELAY Timing Diagram**

Clock Event 1

On the rising edge of C, a reset is detected, causing the output DATAOUT to select tap 0 as the output from the 64-tap chain (assuming IDELAY_VALUE = 0).

Clock Event 2

A pulse on CE and INC is captured on the rising edge of C. This indicates an increment operation. The output changes without glitches from tap 0 to tap 1. See "Stability after an Increment/Decrement Operation."

Clock Event 3

CE and INC are no longer asserted, thus completing the increment operation. The output remains at tap 1 indefinitely until there is further activity on the RST, CE, or INC pins.

Stability after an Increment/Decrement Operation

Figure 7-9 shows a period of instability when the output is changing from one tap to another. Clearly, when the data value at tap 0 is different from the data value at tap 1, the output must change state. However, when the data values at tap 0 and tap 1 are the same (e.g., both 0 or both 1), then the transition from tap 0 to tap 1 causes no glitch or disruption on the output. This concept can be comprehended by imagining the receiver data signal in the IODELAY tap chain. If tap 0 and tap 1 are both near the center of the receiver data eye, then the data sampled at tap 0 should be no different than the data sampled at tap 1. In this case, the transition from tap 0 to tap 1 causes no change to the output. To ensure that this is the case, the increment/decrement operation of IODELAY is designed to be glitchless.

The user can dynamically adjust the IODELAY tap setting in real-time while live user data is passing through the IODELAY element; the adjustments do not disrupt the live user data.

The glitchless behavior also applies when an IODELAY element is used in the path of a clock signal. Adjusting the tap setting does not cause a glitch or disruption on the output. The tap setting of the IODELAY element in the clock path can be adjusted without disrupting state machines that could be running on that clock.

IODELAY VHDL and Verilog Instantiation Template

VHDL and Verilog instantiation templates are available in the Libraries Guide for all primitives and submodules.

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signals names.

Fixed Delay Mode

The Libraries Guide includes templates showing how to instantiate the IODELAY module in fixed delay mode with a tap setting of 31. IDELAYCTRL must also be instantiated when operating in this mode. See "IDELAYCTRL Overview," page 335.

Variable Delay Mode

The Libraries Guide shows how to instantiate the IODELAY module in variable delay mode. IDELAYCTRL must also be instantiated when operating in this mode. See "IDELAYCTRL Overview," page 335.

IODELAY Turnaround Time Usage Model

When using IODELAY in bidirectional mode, the turnaround time needs to be considered. Figure 7-10 shows a simplified block diagram of the IODELAY in the Virtex-5 FPGA IOB that applies to one use of the bidirectional IODELAY functionality.

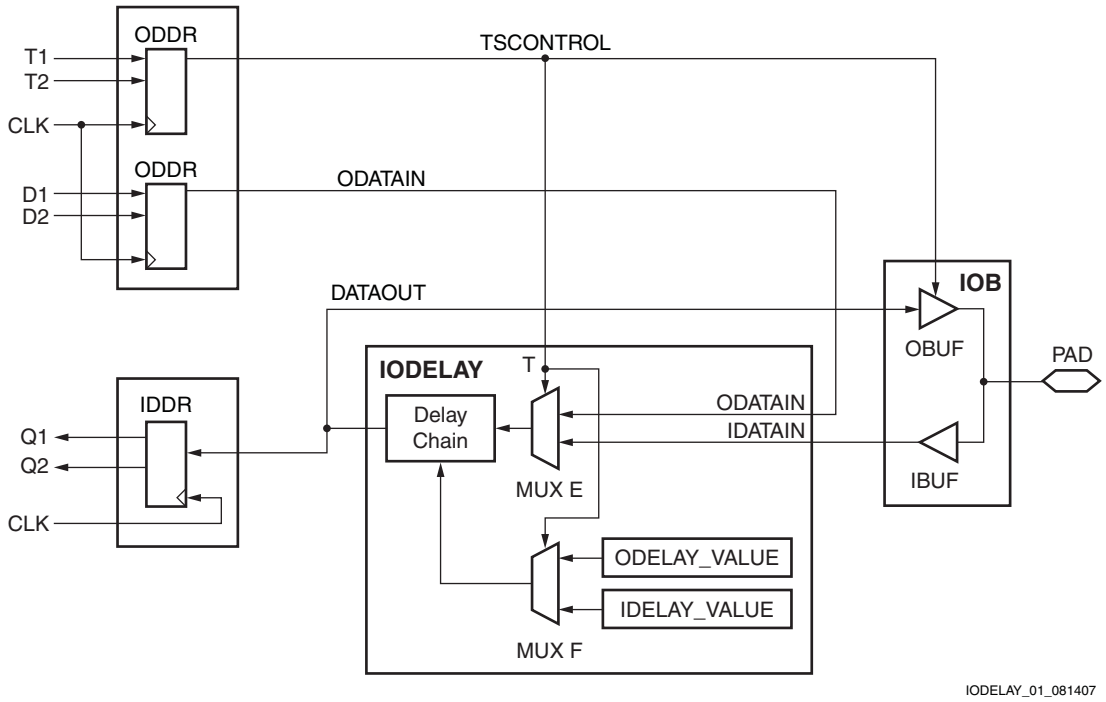


Figure 7-10: **Basic Sections of Blocks Related to IODELAY Turnaround with Pertinent Paths Shown**

When DELAY_SRC = IO, MUXE and MUXF dynamically selects ODATAIN or IDATAIN and ODELAY_VALUE or IDELAY_VALUE inside the IODELAY block.

The following Verilog code segment is used for demonstrating bidirectional IODELAY:

```
IDDR #(
   .DDR_CLK_EDGE ("SAME_EDGE"),
   .INIT_Q1 (1'b0),
   .INIT_Q2 (1'b0),
   .SRTYPE ("SYNC")
)IDDR_INST (
   .C(clk),
   .CE(1'b1),
   .D(DATAOUT),
   .R(1'b0),
   .S(1'b0),
   .Q1(Q1),
   .Q2(Q2)
);
IOBUF #(
   .IOSTANDARD ("LVCMOS25")
)IOBUF_INST (
   .I(DATAOUT),
   .T(TSCONTROL),
   .O(IDATAIN),
   .IO(IOPAD_DATA)
);
```

```
IODELAY #(
   .DELAY_SRC ("IO"),
   .IDELAY_TYPE ("FIXED"),
   .IDELAY_VALUE (12),
   .ODELAY_VALUE (12),
   .REFCLK_FREQUENCY (200.0)
)IODELAY_INST (
   .C(1'b0),
   .CE(1'b0),
   .DATAIN(1'b0),
   .IDATAIN(IDATAIN),
   .INC(1'b0),
   .ODATAIN(ODATAIN),
   .RST(1'b0),
   .T(TSCONTROL),
   .DATAOUT(DATAOUT)
);
ODDR #(
    .DDR_CLK_EDGE ("SAME_EDGE"),
    .INIT (1'b0),
    .SRTYPE ("SYNC")
)ODDR_INST (
   .C(clk),
   .CE(1'b1),
   .D1(D1),
   .D2(D2),
   .R(1'b0),
   .S(1'b0),
   .Q(ODATAIN)
);
ODDR #(
    .DDR_CLK_EDGE ("SAME_EDGE"),
    .INIT (1'b0),
    .SRTYPE ("SYNC")
)TRI_ODDR_INST (
   .C(clk),
   .CE(1'b1),
   .D1(T1),
   .D2(T2),
   .R(1'b0),
   .S(1'b0),
   .Q(TSCONTROL)
);
IDELAYCTRL IDELAYCTRL_INST (
   .REFCLK(refclk),
   .RST(RST),
   .RDY()
);
```

Two cases that use the bidirectional IODELAY functionality are important for a given I/O pin. The first case uses bidirectional IODELAY when the I/O is an output being switched to an input. Figure 7-11 shows the IOB and IODELAY moving toward the input mode as set by the TSCONTROL net coming from the ODDR flip-flop. This controls the selection of MUXes E and F for the IOB input path and IDELAY_VALUE respectively. Additionally, the OBUF is 3-stated.

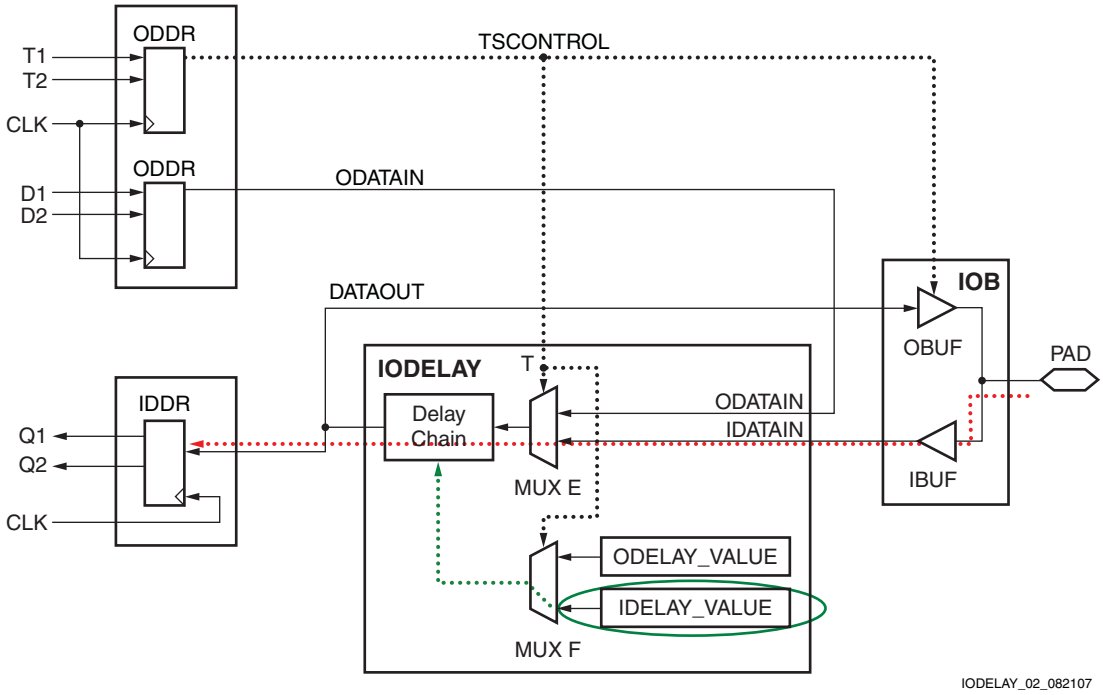


Figure 7-11: **IODELAY and IOB in Input Mode when 3-state is Disabled**

The timing diagram in Figure 7-12 shows the relevant signal timing for the case when the I/O is an output switching to an input using 3-state control. The switching characteristics shown in the diagram are specified in the *Virtex-5 FPGA Data Sheet*.

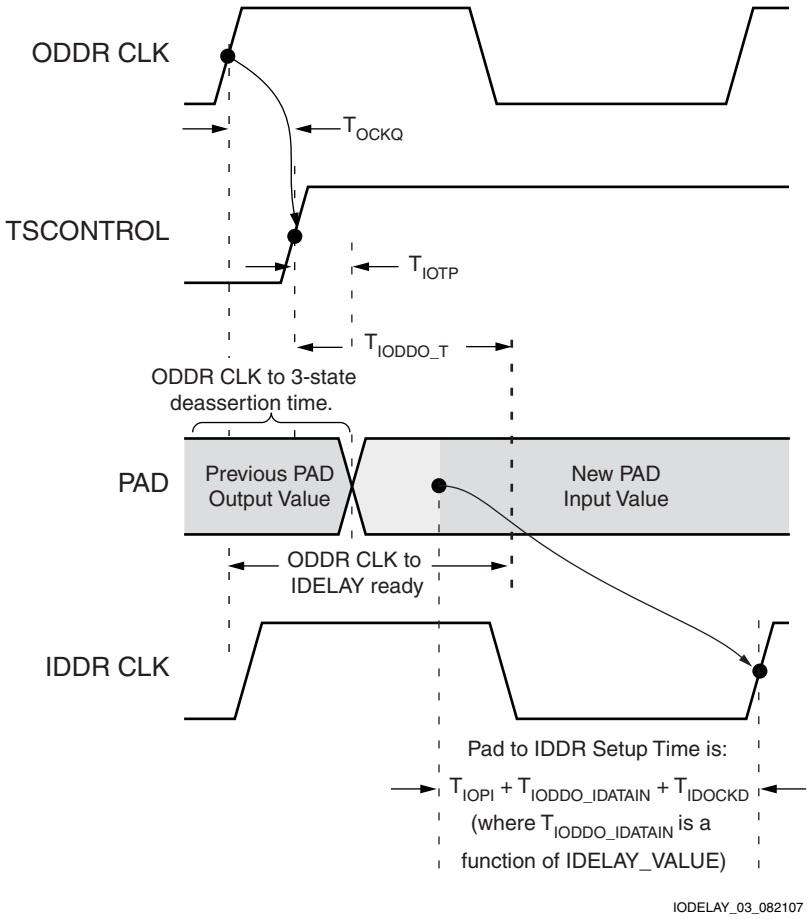


Figure 7-12: **Relevant Timing Signals to Examine IODELAY Timing when the IOB Switches From an Output to an Input**

The activities of the OBUFT pin are controlled by the propagation and state of the TSCONTROL signal from the ODDR flip-flop. The 3-state control data receipt on the OBUF and IDDR flip-flop from a PAD are in parallel with each other, depending on the IDELAY_VALUE setting the final value at the IDDR flip-flop input in response to a clock edge is valid before or after the pad is driven from the 3-state control. After the 3-state control propagates through to the PAD and the IODELAY has been switched to an input, the IDDR setup time is the sole determiner of timing based on the IDELAY_VALUE and other timing parameters defined in the Xilinx speed specification and represented in the ISE tools.

The second case uses bidirectional IODELAY when the I/O is an input switching to an output. Figure 7-13 shows the IOB and IODELAY moving toward the output mode as set by the 3-state TSCONTROL signal coming from the ODDR T flip-flop. This controls the selection of MUXes E and F for the output path and ODELAY_VALUE respectively. Additionally, the OBUF changes to not being 3-stated and starts to drive the PAD.

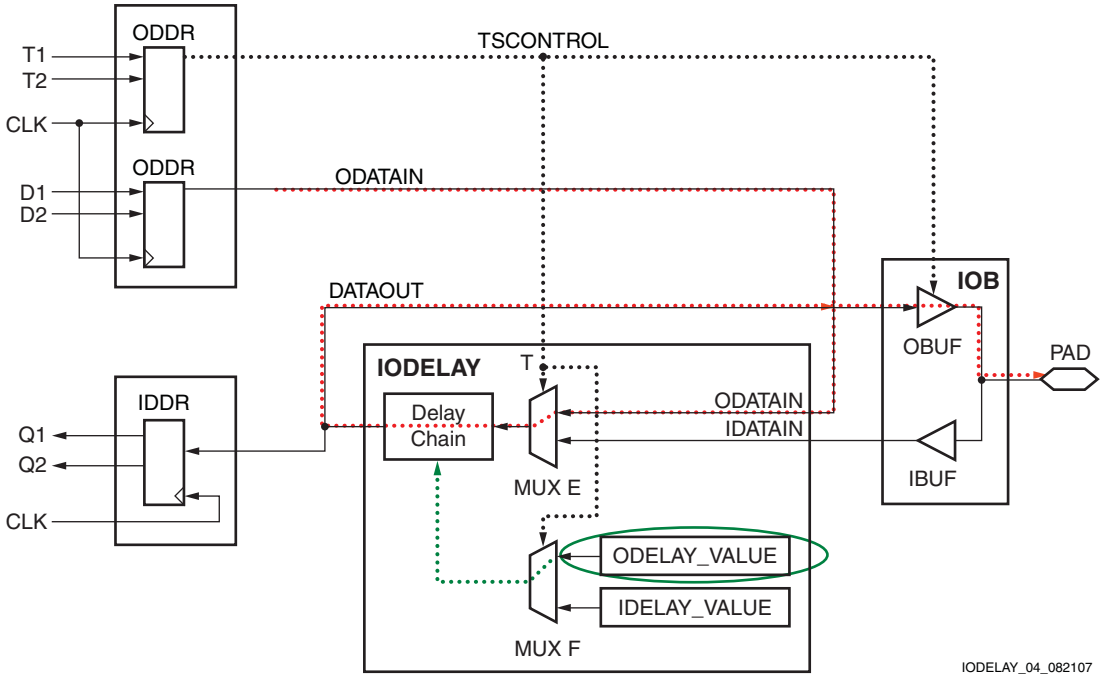


Figure 7-13: **IODELAY and IOB in Output Mode when 3-state is Enabled**

The timing diagram in Figure 7-14 shows the relevant signal timing for the case where the I/O switches from input to an output using 3-state control. The switching characteristics shown in the diagram are specified in the *Virtex-5 FPGA Data Sheet*.

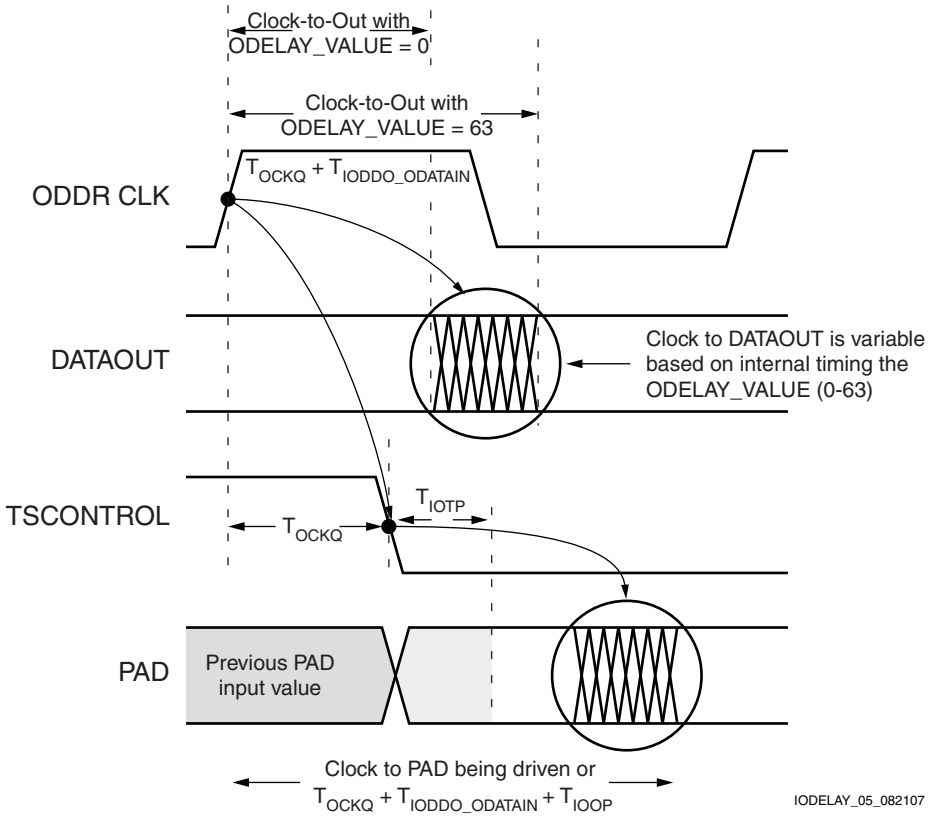


Figure 7-14: **Relevant Timing Signals used to Examine IODELAY Timing when an IOB Changes from an Input to an Output**

3-state control activities on the OBUF of the IOB and ODDR flip-flop to PAD timing are in parallel with each other, depending on the ODELAY_VALUE setting the final output value in response to a clock edge at the ODDR CLK pin is valid before or after the pad is driven from the 3-state control. After the 3-state control propagates through to the PAD and the IODELAY is turned around, the clock-to-output time of the ODDR flip-flop through the IODELAY element (with the ODELAY_VALUE setting) solely determines the clock-to-output time to the pad.

IDELAYCTRL Overview

If the IODELAY or ISERDES primitive is instantiated with the IOBDELAY_TYPE attribute set to FIXED or VARIABLE, the IDELAYCTRL module must be instantiated. The IDELAYCTRL module continuously calibrates the individual delay elements (IODELAY) in its region (see Figure 7-17, page 338), to reduce the effects of process, voltage, and temperature variations. The IDELAYCTRL module calibrates IODELAY using the user supplied REFCLK.

IDELAYCTRL Primitive

Figure 7-15 shows the IDELAYCTRL primitive.

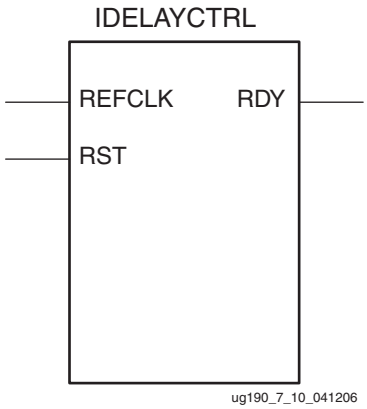


Figure 7-15: **IDELAYCTRL Primitive**

IDELAYCTRL Ports

RST - Reset

The reset input pin (RST) is an active-High asynchronous reset. IDELAYCTRL must be reset after configuration (and the REFCLK signal has stabilized) to ensure proper IODELAY operation. A reset pulse width $T_{IDELAYCTRL\_RPW}$ is required. IDELAYCTRL must be reset after configuration.

REFCLK - Reference Clock

The reference clock (REFCLK) provides a time reference to IDELAYCTRL to calibrate all IODELAY modules in the same region. This clock must be driven by a global clock buffer (BUFGCTRL). REFCLK must be $F_{IDELAYCTRL\_REF}$ ± the specified ppm tolerance (IDELAYCTRL_REF_PRECISION) to guarantee a specified IODELAY resolution ($T_{IDELAYRESOLUTION}$). REFCLK can be supplied directly from a user-supplied source, the PLL, or from the DCM, and must be routed on a global clock buffer.

RDY - Ready

The ready (RDY) signal indicates when the IODELAY modules in the specific region are calibrated. The RDY signal is deasserted if REFCLK is held High or Low for one clock period or more. If RDY is deasserted Low, the IDELAYCTRL module must be reset. The implementation tools allow RDY to be unconnected/ignored. Figure 7-16 illustrates the timing relationship between RDY and RST.

IDELAYCTRL Timing

Table 7-12 shows the IDELAYCTRL switching characteristics.

Table 7-12: **IDELAYCTRL Switching Characteristics**

Symbol	Description
$F_{IDELAYCTRL\_REF}$	REFCLK frequency
IDELAYCTRL_REF_PRECISION	REFCLK precision
$T_{IDELAYCTRLCO\_RDY}$	Reset/Startup to Ready for IDELAYCTRL

As shown in Figure 7-16, the Virtex-5 FPGA RST is an edge-triggered signal.

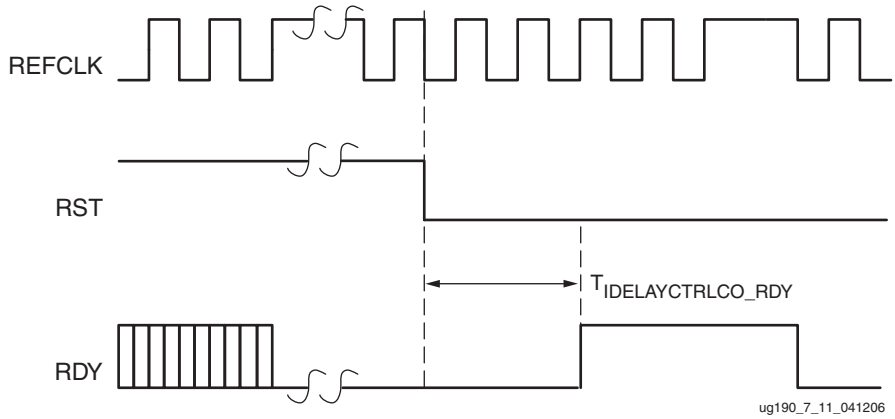


Figure 7-16: **Timing Relationship Between RST and RDY**

IDELAYCTRL Locations

IDELAYCTRL modules exist in every I/O column in every clock region. An IDELAYCTRL module calibrates all the IDELAY modules within its clock region. See "Global and Regional Clocks" in Chapter 1 for the definition of a clock region.

Figure 7-17 illustrates the relative locations of the IDELAYCTRL modules.

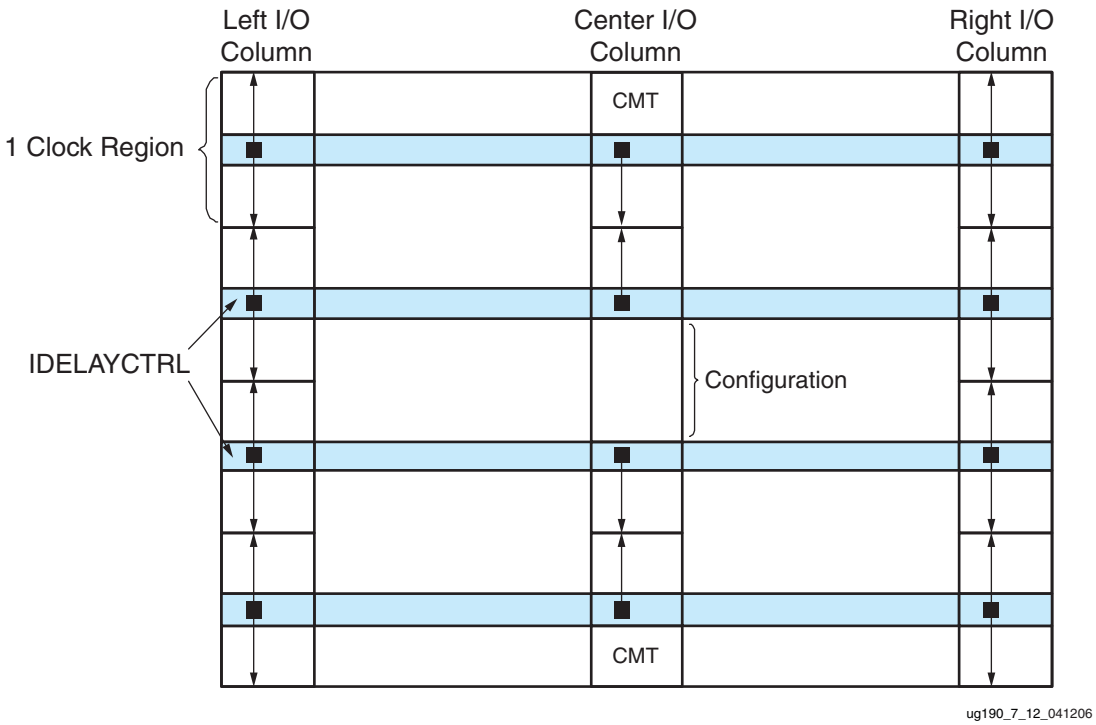


Figure 7-17: **Relative Locations of IDELAYCTRL Modules**

IDELAYCTRL Usage and Design Guidelines

This section describes using the IDELAYCTRL modules, design guidelines, and recommended usage in Virtex-5 devices.

Instantiating IDELAYCTRL Without LOC Constraints

When instantiating IDELAYCTRL without LOC constraints, the user must instantiate only one instance of IDELAYCTRL in the HDL design code. The implementation tools auto-replicate IDELAYCTRL instances throughout the entire device. When IDELAYCTRL instances are replicated to clock regions but not used, the extra instances are trimmed out of the design automatically by the ISE software. The signals connected to the RST and REFCLK input ports of the instantiated IDELAYCTRL instance are connected to the corresponding input ports of the replicated IDELAYCTRL instances.

There are two special cases:

1. When the RDY port is ignored, the RDY signals of all the replacement IDELAYCTRL instances are left unconnected.

   The VHDL and Verilog use models for instantiating an IDELAYCTRL primitive without LOC constraints leaving the RDY output port unconnected are provided in the Libraries Guide.

   The resulting circuitry after instantiating the IDELAYCTRL components is illustrated in Figure 7-18.

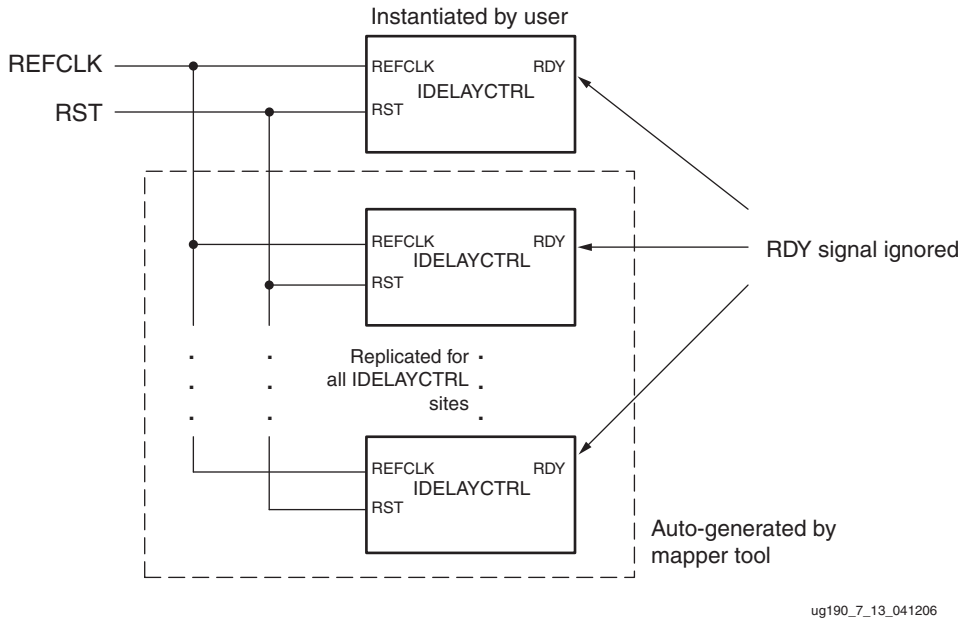


Figure 7-18: **Instantiate IDELAYCTRL Without LOC Constraints - RDY Unconnected**

2. When RDY port is connected, an AND gate of width equal to the number of clock regions is instantiated and the RDY output ports from the instantiated and replicated IDELAYCTRL instances are connected to the inputs of the AND gate. The tools assign the signal name connected to the RDY port of the instantiated IDELAYCTRL instance to the output of the AND gate.

   The VHDL and Verilog use models for instantiating an IDELAYCTRL primitive without LOC constraints with the RDY port connected are provided in the Libraries Guide.

   The resulting circuitry after instantiating the IDELAYCTRL components is illustrated in Figure 7-19.

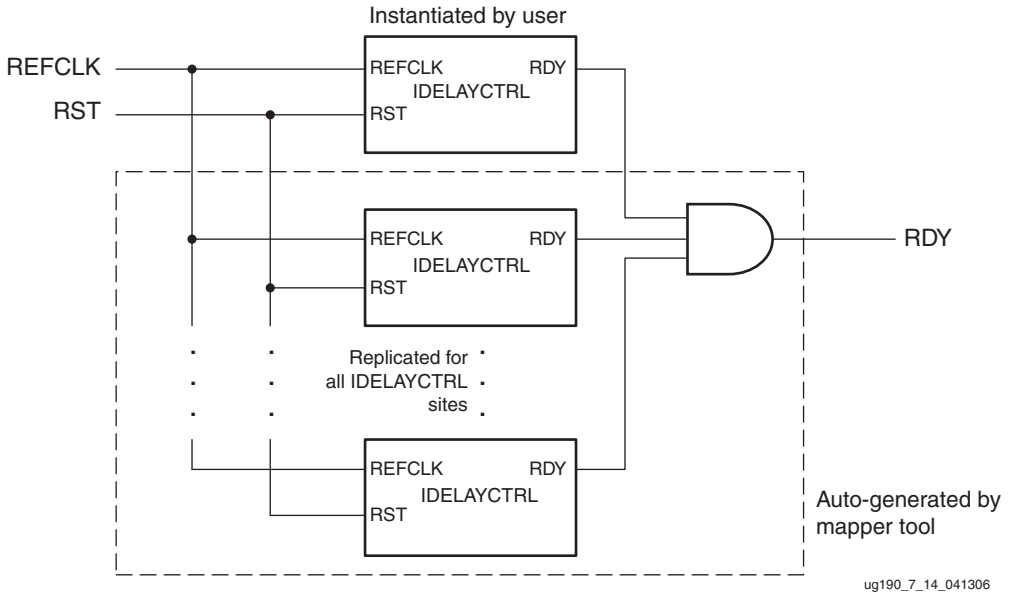


Figure 7-19: **Instantiate IDELAYCTRL Without LOC Constraints - RDY Connected**

Instantiating IDELAYCTRL with Location (LOC) Constraints

The most efficient way to use the IDELAYCTRL module is to define and lock down the placement of every IDELAYCTRL instance used in a design. This is done by instantiating the IDELAYCTRL instances with location (LOC) constraints. The user must define and lock placement of all ISERDES and IDELAY components using the delay element. (IDELAY_TYPE attribute set to FIXED or VARIABLE). Once completed, IDELAYCTRL sites can be chosen and LOC constraints assigned. Xilinx strongly recommends using IDELAYCTRL with a LOC constraint. When not using an IDELAY (with IDELAY_TYPE in FIXED or VARIABLE mode) do not assign a LOC constraint to the IDELAYCTRL for that clock region.

Location Constraints

Each IDELAYCTRL module has XY location coordinates (X:row, Y:column). To constrain placement, IDELAYCTRL instances can have LOC properties attached to them. The naming convention for IDELAYCTRL placement coordinates is different from the convention used in naming CLB locations. This allows LOC properties to transfer easily from array to array.

There are two methods of attaching LOC properties to IDELAYCTRL instances.

1. Insert LOC constraints in a UCF file
2. Embed LOC constraints directly into HDL design files

Inserting LOC Constraints in a UCF File

The following syntax is used for inserting LOC constraints in a UCF file.

```
INST "instance_name" LOC=IDELAYCTRL_X#Y#;
```

Embedding LOC Constraints Directly into HDL Design Files

The following syntax is used to embed LOC constraints into a Verilog design file.

```
// synthesis attribute loc of instance_name is "IDELAYCTRL_X#Y0#";
```

In VHDL code, the LOC constraint is described with VHDL attributes. Before it can be used, the constraint must be declared with the following syntax:

```
attribute loc : string;
```

Once declared, the LOC constraint can be specified as:

```
attribute loc of instance_name:label is "IDELAYCTRL_X#Y0#";
```

The Libraries Guide includes VHDL and Verilog use model templates for instantiating IDELAYCTRL primitives with LOC constraints.

The circuitry that results from instantiating the IDELAYCTRL components is shown in Figure 7-20.

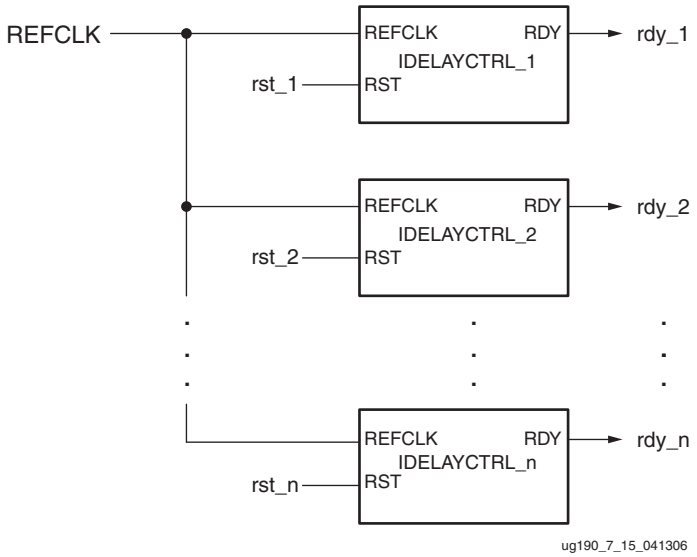


Figure 7-20: **Instantiate IDELAYCTRL with LOC Constraint**

Instantiating IDELAYCTRL With and Without LOC Constraints

There are cases where the user instantiates an IDELAYCTRL module with a LOC constraint but also instantiates an IDELAYCTRL module without a LOC constraint. In the case where an IP Core is instantiated with a non-location constrained IDELAYCTRL module and also wants to instantiate an IDELAYCTRL module without a LOC constraint for another part of the design, the implementation tools will perform the following:

- Instantiate the LOC IDELAYCTRL instances as described in the section Instantiating IDELAYCTRL with Location (LOC) Constraints.
- Replicate the non-location constrained IDELAYCTRL instance to populate with an IDELAYCTRL instance in every clock region without a location constrained IDELAYCTRL instance in place.
- The signals connected to the RST and REFCLK input ports of the non-location constrained IDELAYCTRL instance are connected to the corresponding input ports of the replicated IDELAYCTRL instances.
- If the RDY port of the non-location constrained IDELAYCTRL instance is ignored, then all the RDY signals of the replicated IDELAYCTRL instances are also ignored.
- If the RDY port of the non-location constrained IDELAYCTRL instance is connected, then the RDY port of the non-location constrained instance plus the RDY ports of the replicated instances are connected to an auto-generated AND gate. The implementation tools assign the signal name connected to the RDY port of the non-location constrained instance to the output of the AND gate.
- All the ports of the location constrained instances (RST, REFCLK, and RDY) are independent from each other and from the replicated instances.

The VHDL and Verilog use models for instantiating a mixed usage model are provided in the Libraries Guide. In the example, a user is instantiating a non-location constrained IDELAYCTRL instance with the RDY signal connected. This discussion is also valid when the RDY signal is ignored.

The circuitry that results from instantiating the IDELAYCTRL components is illustrated in Figure 7-21.

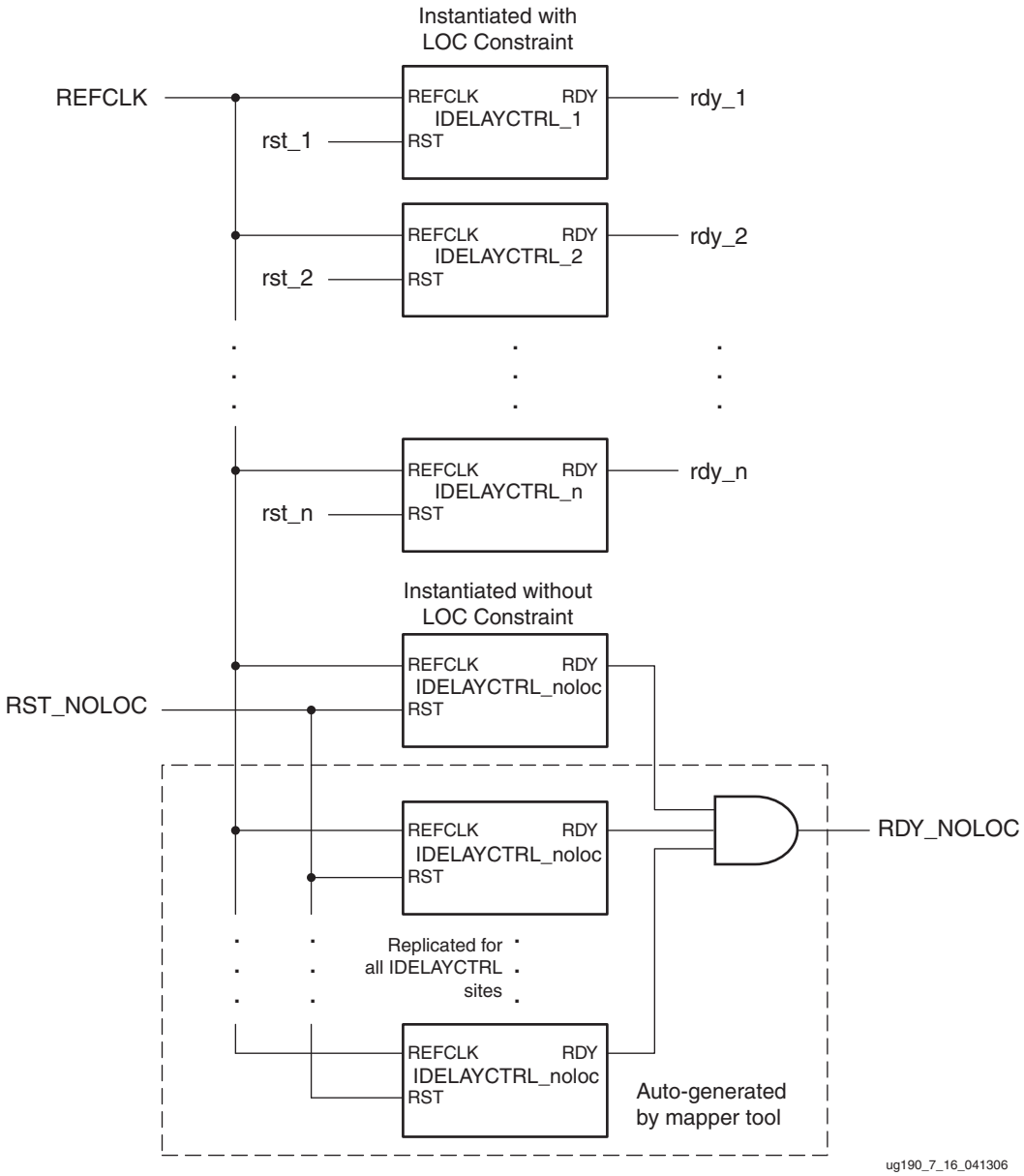


Figure 7-21: **Mixed Instantiation of IDELAYCTRL Elements**

Instantiating Multiple IDELAYCTRLs Without LOC Constraints

Instantiating multiple IDELAYCTRL instances without LOC properties is prohibited. If this occurs, an error is issued by the implementation tools.

OLOGIC Resources

OLOGIC consists of two major blocks, one to configure the output data path and the other to configure the 3-state control path. These two blocks have a common clock (CLK) but different enable signals, OCE and TCE. Both have asynchronous and synchronous set and reset (SR and REV signals) controlled by an independent SRVAL attribute as described in the Table 7-1 and Table 7-2.

The Output and the 3-State paths can be configured in one of the following modes independently.

- Edge triggered D type flip-flop
- DDR mode (SAME_EDGE or OPPOSITE_EDGE)
- Level Sensitive Latch
- Asynchronous/combinatorial

Figure 7-22 illustrates the various logic resources in the OLOGIC block.

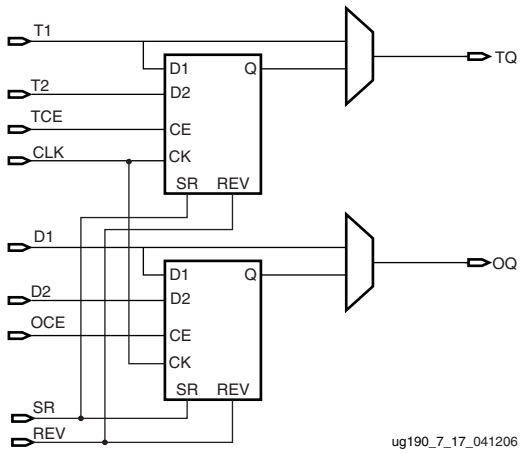


Figure 7-22: **OLOGIC Block Diagram**

This section of the documentation discusses the various features available using the OLOGIC resources. All connections between the OLOGIC resources are managed in Xilinx software.

Combinatorial Output Data and 3-State Control Path

The combinatorial output paths create a direct connection from the FPGA fabric to the output driver or output driver control. These paths is used when:

1. There is direct (unregistered) connection from logic resources in the FPGA fabric to the output data or 3-state control.
2. The "pack I/O register/latches into IOBs" is set to OFF.

Output DDR Overview (ODDR)

Virtex-5 devices have dedicated registers in the OLOGIC to implement output DDR registers. This feature is accessed when instantiating the ODDR primitive. DDR multiplexing is automatic when using OLOGIC. No manual control of the mux-select is needed. This control is generated from the clock.

There is only one clock input to the ODDR primitive. Falling edge data is clocked by a locally inverted version of the input clock. All clocks feeding into the I/O tile are fully multiplexed, i.e., there is no clock sharing between ILOGIC or OLOGIC blocks. The ODDR primitive supports the following modes of operation:

- OPPOSITE_EDGE mode
- SAME_EDGE mode

The SAME_EDGE mode is the same as for the Virtex-4 architecture. This mode allows designers to present both data inputs to the ODDR primitive on the rising-edge of the ODDR clock, saving CLB and clock resources, and increasing performance. This mode is implemented using the DDR_CLK_EDGE attribute. It is supported for 3-state control as well. The following sections describe each of the modes in detail.

OPPOSITE_EDGE Mode

In OPPOSITE_EDGE mode, both the edges of the clock (CLK) are used to capture the data from the FPGA fabric at twice the throughput. This structure is similar to the Virtex-II Virtex-II Pro, and Virtex-4 FPGA implementation. Both outputs are presented to the data input or 3-state control input of the IOB. The timing diagram of the output DDR using the OPPOSITE_EDGE mode is shown in Figure 7-23.

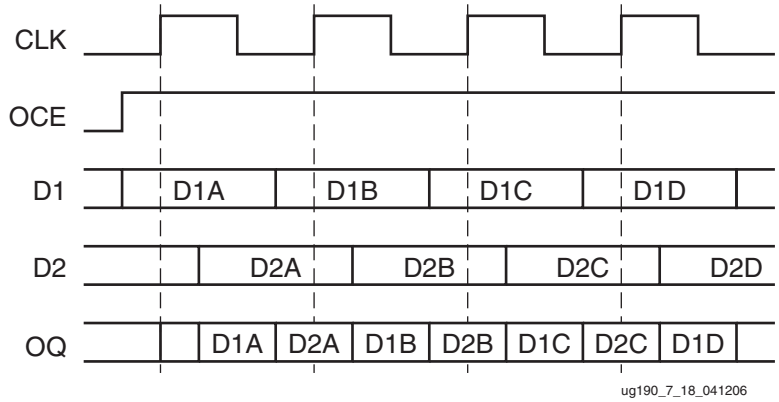


Figure 7-23: **Output DDR Timing in OPPOSITE_EDGE Mode**

SAME_EDGE Mode

In SAME_EDGE mode, data can be presented to the IOB on the same clock edge. Presenting the data to the IOB on the same clock edge avoids setup time violations and allows the user to perform higher DDR frequency with minimal register to register delay, as opposed to using the CLB registers. Figure 7-24 shows the timing diagram of the output DDR using the SAME_EDGE mode.

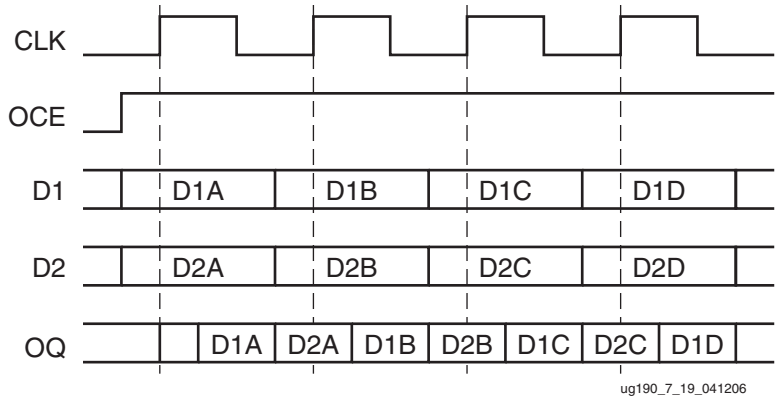


Figure 7-24: **Output DDR Timing in SAME_EDGE Mode**

Clock Forwarding

Output DDR can forward a copy of the clock to the output. This is useful for propagating a clock and DDR data with identical delays, and for multiple clock generation, where every clock load has a unique clock driver. This is accomplished by tying the D1 input of the ODDR primitive High, and the D2 input Low. Xilinx recommends using this scheme to forward clocks from the FPGA fabric to the output pins.

Output DDR Primitive (ODDR)

Figure 7-25 shows the ODDR primitive block diagram. Table 7-13 lists the ODDR port signals. Table 7-14 describes the various attributes available and default values for the ODDR primitive.

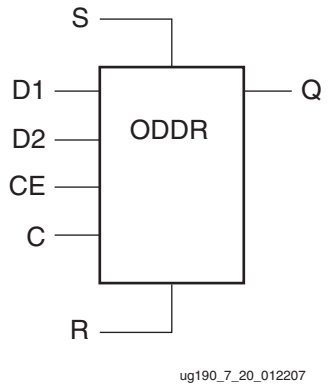


Figure 7-25: **ODDR Primitive Block Diagram**

Table 7-13: **ODDR Port Signals**

Port Name	Function	Description
Q	Data output (DDR)	ODDR register output.
C	Clock input port	The CLK pin represents the clock input pin.
CE	Clock enable port	CE represents the clock enable pin. When asserted Low, this port disables the output clock on port Q.
D1 and D2	Data inputs	ODDR register inputs.
R	Reset	Synchronous/Asynchronous reset pin. Reset is asserted High.
S	Set	Synchronous/Asynchronous set pin. Set is asserted High.

Table 7-14: **ODDR Attributes**

Attribute Name	Description	Possible Values
DDR_CLK_EDGE	Sets the ODDR mode of operation with respect to clock edge	OPPOSITE_EDGE (default), SAME_EDGE
INIT	Sets the initial value for Q port	0 (default), 1
SRTYPE	Set/Reset type with respect to clock (C)	ASYNC, SYNC (default)

ODDR VHDL and Verilog Templates

The Libraries Guide includes templates for instantiation of the ODDR module in VHDL and Verilog.

OLOGIC Timing Models

This section discusses all timing models associated with the OLOGIC block. Table 7-15 describes the function and control signals of the OLOGIC switching characteristics in the *Virtex-5 FPGA Data Sheet*.

Table 7-15: **OLOGIC Switching Characteristics**

Symbol	Description
Setup/Hold	
T_{ODCK}/T_{OCKD}	D1/D2 pins Setup/Hold with respect to CLK
T_{OOCECK}/T_{OCKOCE}	OCE pin Setup/Hold with respect to CLK
T_{OSRCK}/T_{OCKSR}	SR/REV pin Setup/Hold with respect to CLK
T_{OTCK}/T_{OCKT}	T1/T2 pins Setup/Hold with respect to CLK
T_{OTCECK}/T_{OCKTCE}	TCE pin Setup/Hold with respect to CLK
Clock to Out	
T_{OCKQ}	CLK to OQ/TQ out
T_{RQ}	SR/REV pin to OQ/TQ out

Timing Characteristics

Figure 7-26 illustrates the OLOGIC output register timing.

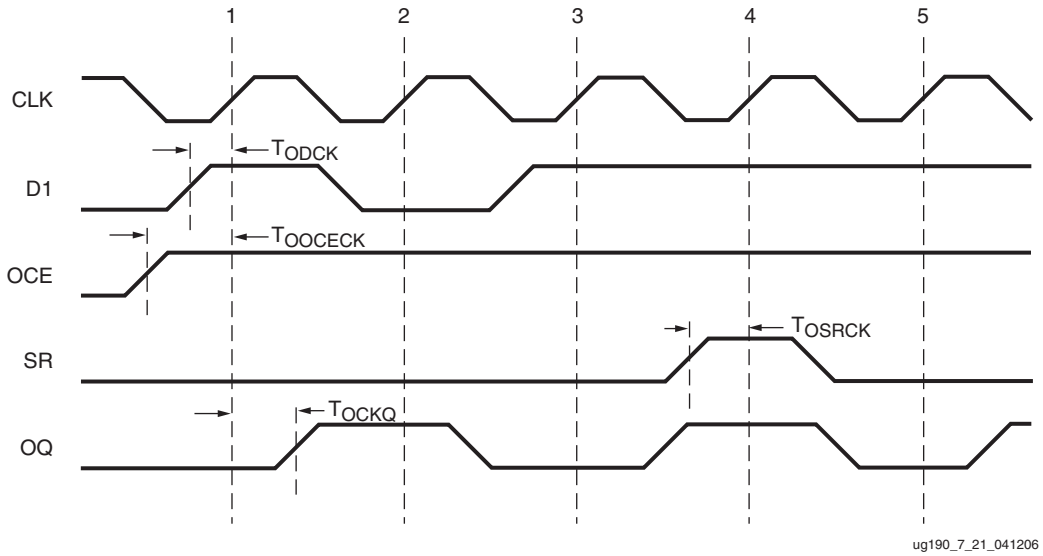


Figure 7-26: **OLOGIC Output Register Timing Characteristics**

Clock Event 1

- At time T_{OOCECK} before Clock Event 1, the output clock enable signal becomes valid-high at the OCE input of the output register, enabling the output register for incoming data.
- At time T_{ODCK} before Clock Event 1, the output signal becomes valid-high at the D1 input of the output register and is reflected at the OQ output at time T_{OCKQ} after Clock Event 1.

Clock Event 4

At time T_{OSRCK} before Clock Event 4, the SR signal (configured as synchronous reset in this case) becomes valid-high, resetting the output register and reflected at the OQ output at time T_{RQ} after Clock Event 4.

Figure 7-27 illustrates the OLOGIC ODDR register timing.

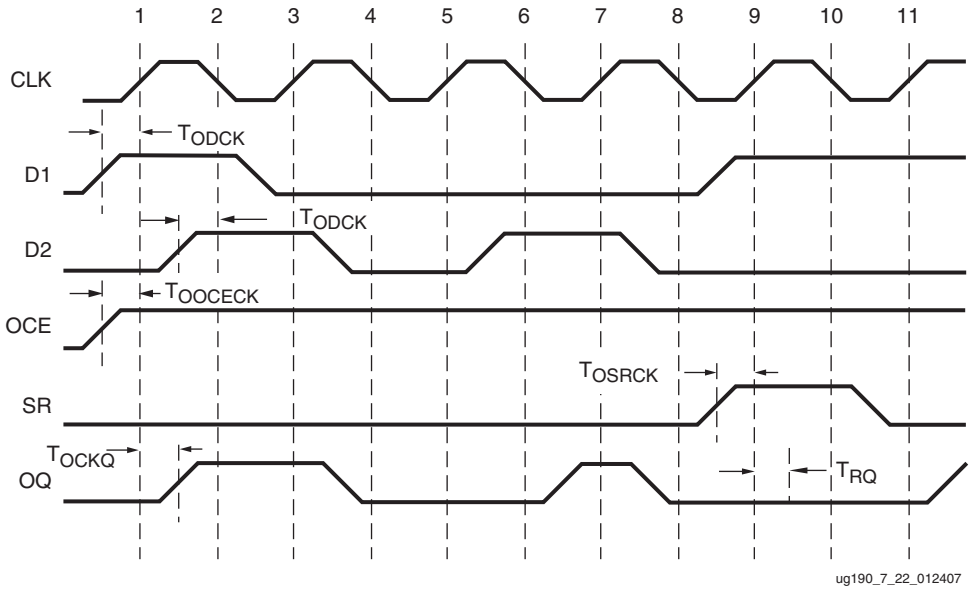


Figure 7-27: **OLOGIC ODDR Register Timing Characteristics**

Clock Event 1

- At time T_{OOCECK} before Clock Event 1, the ODDR clock enable signal becomes valid-High at the OCE input of the ODDR, enabling ODDR for incoming data. Care must be taken to toggle the OCE signal of the ODDR register between the rising edges and falling edges of CLK as well as meeting the register setup-time relative to both clock edges.
- At time T_{ODCK} before Clock Event 1 (rising edge of CLK), the data signal D1 becomes valid-high at the D1 input of ODDR register and is reflected on the OQ output at time T_{OCKQ} after Clock Event 1.

Clock Event 2

- At time T_{ODCK} before Clock Event 2 (falling edge of CLK), the data signal D2 becomes valid-high at the D2 input of ODDR register and is reflected on the OQ output at time T_{OCKQ} after Clock Event 2 (no change at the OQ output in this case).

Clock Event 9

At time T_{OSRCK} before Clock Event 9 (rising edge of CLK), the SR signal (configured as synchronous reset in this case) becomes valid-high resetting ODDR register, reflected at the OQ output at time T_{RQ} after Clock Event 9 (no change at the OQ output in this case) and resetting ODDR register, reflected at the OQ output at time T_{RQ} after Clock Event 10 (no change at the OQ output in this case).

Figure 7-28 illustrates the OLOGIC 3-state register timing.

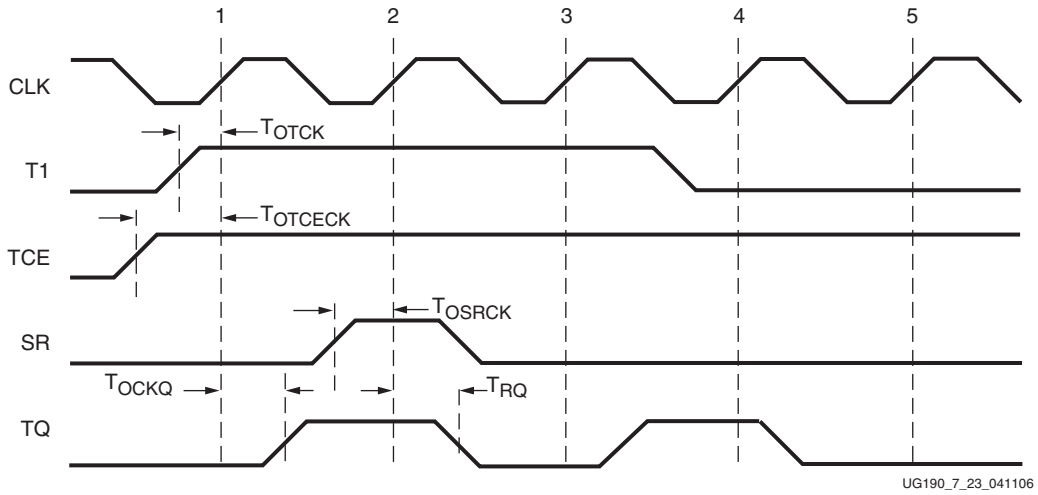


Figure 7-28: **OLOGIC 3-State Register Timing Characteristics**

Clock Event 1

- At time T_{OTCECK} before Clock Event 1, the 3-state clock enable signal becomes valid-high at the TCE input of the 3-state register, enabling the 3-state register for incoming data.
- At time T_{OTCK} before Clock Event 1 the 3-state signal becomes valid-high at the T input of the 3-state register, returning the pad to high-impedance at time T_{OCKQ} after Clock Event 1.

Clock Event 2

- At time T_{OSRCK} before Clock Event 2, the SR signal (configured as synchronous reset in this case) becomes valid-high, resetting the 3-state register at time T_{RQ} after Clock Event 2.

Figure 7-29 illustrates IOB DDR 3-state register timing. This example is shown using DDR in opposite edge mode. For other modes add the appropriate latencies as shown in Figure 7-4, page 319.

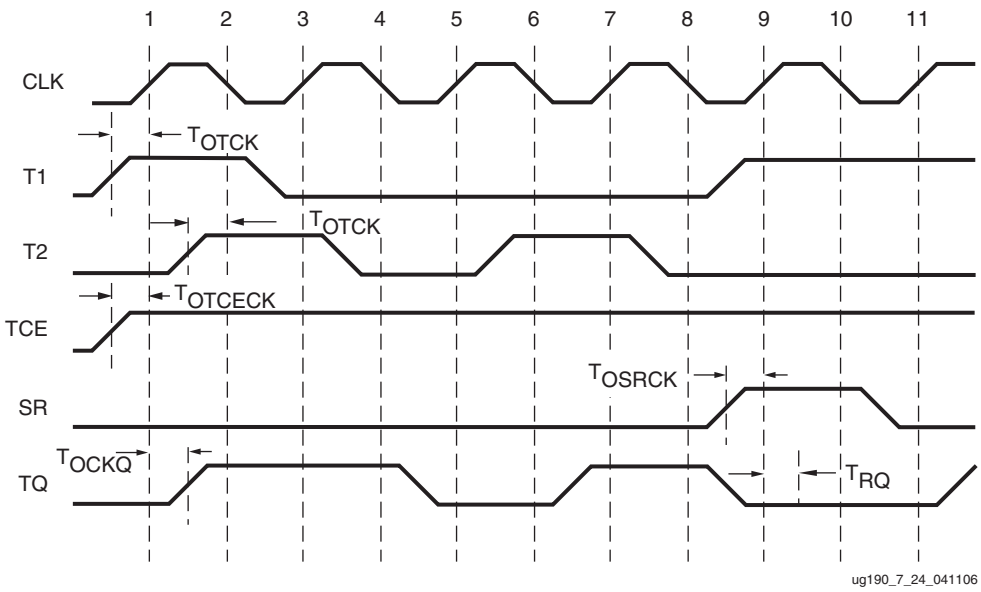


Figure 7-29: **OLOGIC ODDR 3-State Register Timing Characteristics**

Clock Event 1

- At time T_{OTCECK} before Clock Event 1, the 3-state clock enable signal becomes valid-High at the TCE input of the 3-state ODDR register, enabling them for incoming data. Care must be taken to toggle the TCE signal of the 3-state ODDR between the rising edges and falling edges of CLK as well as meeting the register setup-time relative to both clock edges.
- At time T_{OTCK} before Clock Event 1 (rising edge of CLK), the 3-state signal T1 becomes valid-high at the T1 input of 3-state register and is reflected on the TQ output at time T_{OCKQ} after Clock Event 1.

Clock Event 2

- At time T_{OTCK} before Clock Event 2 (falling edge of CLK), the 3-state signal T2 becomes valid-high at the T2 input of 3-state register and is reflected on the TQ output at time T_{OCKQ} after Clock Event 2 (no change at the TQ output in this case).

Clock Event 9

- At time T_{OSRCK} before Clock Event 9 (rising edge of CLK), the SR signal (configured as synchronous reset in this case) becomes valid-high resetting 3-state Register, reflected at the TQ output at time T_{RQ} after Clock Event 9 (no change at the TQ output in this case) and resetting 3-state Register, reflected at the TQ output at time T_{RQ} after Clock Event 10 (no change at the TQ output in this case).

XILINX®

Chapter 8

Advanced SelectIO Logic Resources

Introduction

The I/O functionality in Virtex-5 FPGAs is described in Chapter 6 through Chapter 8 of this user guide.

- Chapter 6 covers the electrical characteristics of input receivers and output drivers, and their compliance with many industry standards.
- Chapter 7 describes the register structures dedicated for sending and receiving SDR or DDR data.
- This chapter covers additional resources:
  - Input serial-to-parallel converters (ISERDES) and output parallel-to-serial converters (OSERDES) support very fast I/O data rates, and allow the internal logic to run up to 10 times slower than the I/O.
  - The Bitslip submodule can re-align data to word boundaries, detected with the help of a training pattern.

Input Serial-to-Parallel Logic Resources (ISERDES)

The ISERDES in Virtex-5 FPGAs is a dedicated serial-to-parallel converter with specific clocking and logic features designed to facilitate the implementation of high-speed source-synchronous applications. The ISERDES avoids the additional timing complexities encountered when designing deserializers in the FPGA fabric.

ISERDES features include:

- Dedicated Deserializer/Serial-to-Parallel Converter

  The ISERDES deserializer enables high-speed data transfer without requiring the FPGA fabric to match the input data frequency. This converter supports both single data rate (SDR) and double data rate (DDR) modes. In SDR mode, the serial-to-parallel converter creates a 2-, 3-, 4-, 5-, 6-, 7-, or 8-bit wide parallel word. In DDR mode, the serial-to-parallel converter creates a 4-, 6-, 8-, or 10-bit-wide parallel word.

- Bitslip Submodule

  The Bitslip submodule allows designers to reorder the sequence of the parallel data stream going into the FPGA fabric. This can be used for training source-synchronous interfaces that include a training pattern.

- Dedicated Support for Strobe-based Memory Interfaces

  ISERDES contains dedicated circuitry (including the OCLK input pin) to handle the strobe-to-FPGA clock domain crossover entirely within the ISERDES block. This allows for higher performance and a simplified implementation.

- Dedicated support for Networking interfaces.

Figure 8-1 shows the block diagram of the ISERDES, highlighting all the major components and features of the block.

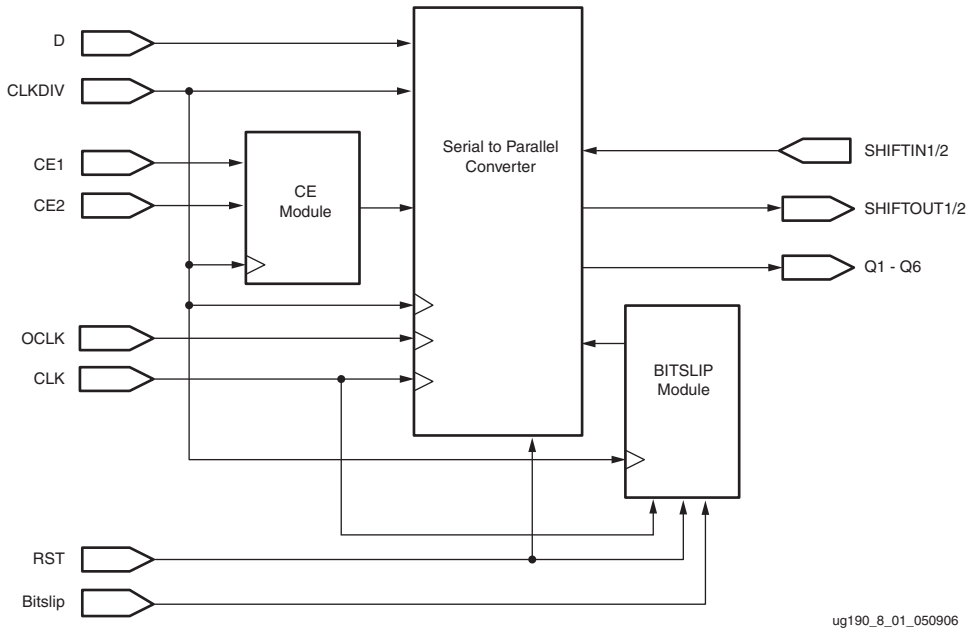


Figure 8-1: **ISERDES Block Diagram**

ISERDES Primitive (ISERDES_NODELAY)

The ISERDES primitive in Virtex-5 devices (shown in Figure 8-2) is ISERDES_NODELAY.

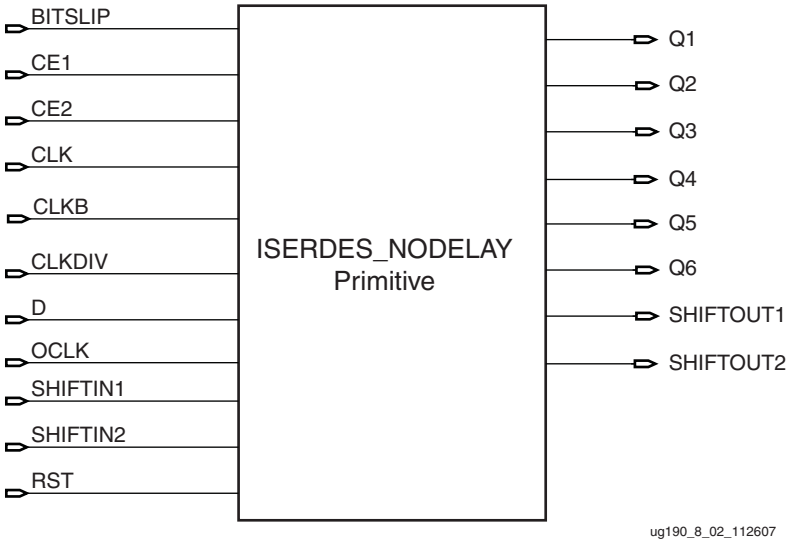


Figure 8-2: **ISERDES_NODELAY Primitive**

Table 8-1 lists the available ports in the ISERDES_NODELAY primitive.

Table 8-1: **ISERDES_NODELAY Port List and Definitions**

Port Name	Type	Width	Description
Q1 – Q6	Output	1 (each)	Registered outputs. See "Registered Outputs - Q1 to Q6."
SHIFTOUT1	Output	1	Carry out for data width expansion. Connect to SHIFTIN1 of slave IOB. See "ISERDES Width Expansion."
SHIFTOUT2	Output	1	Carry out for data width expansion. Connect to SHIFTIN2 of slave IOB. See "ISERDES Width Expansion."
BITSLIP	Input	1	Invokes the Bitslip operation. See "Bitslip Operation - BITSLIP."
CE1 CE2	Input	1 (each)	Clock enable inputs. See "Clock Enable Inputs - CE1 and CE2."
CLK	Input	1	High-speed clock input. Clocks serial input data stream. See "High-Speed Clock Input - CLK."
CLKB	Input	1	High-speed secondary clock input. Clocks serial input data stream. Always connect this $\overline{\text{CLK}}$.
CLKDIV	Input	1	Divided clock input. Clocks delay element, deserialized data, Bitslip submodule, and CE unit. See "Divided Clock Input - CLKDIV."
D	Input	1	Serial input data from IOB. See "Serial Input Data from IOB - D."
OCLK	Input	1	High-speed clock input for memory applications. See "High-Speed Clock for Strobe-Based Memory Interfaces - OCLK."
SHIFTIN1	Input	1	Carry input for data width expansion. Connect to SHIFTOUT1 of master IOB. See "ISERDES Width Expansion."
SHIFTIN2	Input	1	Carry input for data width expansion. Connect to SHIFTOUT2 of master IOB. See "ISERDES Width Expansion."
RST	Input	1	Active High reset. See "Reset Input - RST."

ISERDES_NODELAY Ports

Registered Outputs - Q1 to Q6

The output ports Q1 to Q6 are the registered outputs of the ISERDES_NODELAY module. One ISERDES_NODELAY block can support up to six bits (i.e., a 1:6 deserialization). Bit widths greater than six (up to 10) can be supported. See "ISERDES Width Expansion." The first data bit received appears on the highest order Q output.

The bit ordering at the input of an OSERDES is the opposite of the bit ordering at the output of an ISERDES_NODELAY block, as shown in Figure 8-3. For example, the least significant bit A of the word FEDCBA is placed at the D1 input of an OSERDES, but the same bit A emerges from the ISERDES_NODELAY block at the Q6 output. In other words, D1 is the least significant input to the OSERDES, while Q6 is the least significant output of the ISERDES_NODELAY block. When width expansion is used, D1 of the master OSERDES is the least significant input, while Q4 of the slave ISERDES_NODELAY block is the least significant output.

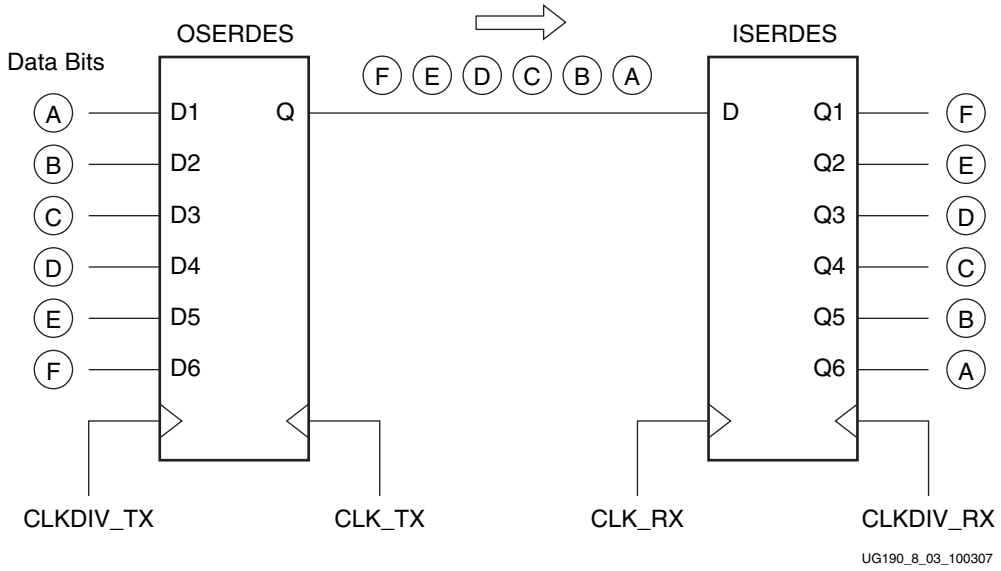


Figure 8-3: **Bit Ordering on Q1–Q6 Outputs of ISERDES_NODELAY Ports**

Bitslip Operation - BITSLIP

The BITSLIP pin performs a Bitslip operation synchronous to CLKDIV when asserted (active High). Subsequently, the data seen on the Q1 to Q6 output ports will shift, as in a barrel-shifter operation, one position every time Bitslip is invoked (DDR operation is different from SDR). See "BITSLIP Submodule" for more details.

Clock Enable Inputs - CE1 and CE2

Each ISERDES_NODELAY block contains an input clock enable module (Figure 8-4).

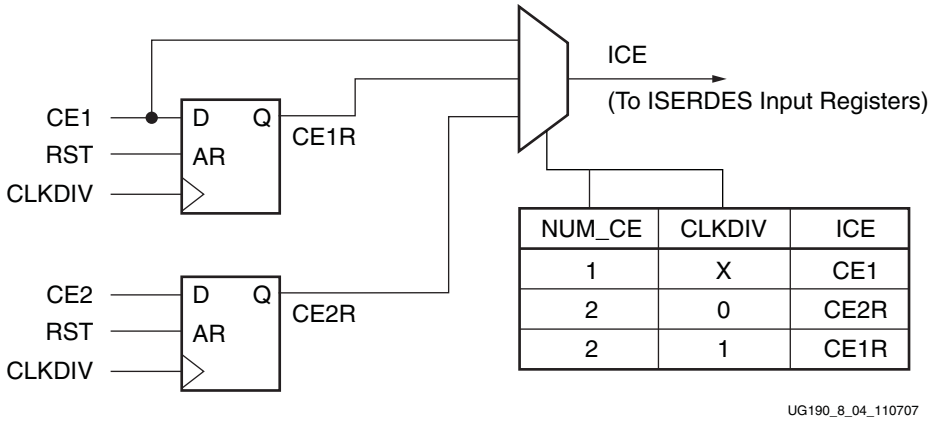


Figure 8-4: **Input Clock Enable Module**

When NUM_CE = 1, the CE2 input is not used, and the CE1 input is an active High clock enable connected directly to the input registers in the ISERDES_NODELAY. When NUM_CE = 2, the CE1 and CE2 inputs are both used, with CE1 enabling the ISERDES_NODELAY for ½ of a CLKDIV cycle, and CE2 enabling the ISERDES_NODELAY for the other ½. The internal clock enable signal ICE shown in Figure 8-4 is derived from the CE1 and CE2 inputs. ICE drives the clock enable inputs of

registers FF0, FF1, FF2, and FF3 shown in Figure 8-12, page 366. The remaining registers in Figure 8-13, page 367 do not have clock enable inputs.

The clock enable module functions as a 2:1 serial-to-parallel converter, clocked by CLKDIV. The clock enable module is needed specifically for bidirectional memory interfaces when ISERDES_NODELAY is configured for 1:4 deserialization in DDR mode. When the attribute NUM_CE = 2, the clock enable module is enabled and both CE1 and CE2 ports are available. When NUM_CE = 1, only CE1 is available and functions as a regular clock enable.

High-Speed Clock Input - CLK

The high-speed clock input (CLK) is used to clock in the input serial data stream.

High-Speed Clock Input - CLKB

The high-speed secondary clock input (CLKB) is used to clock in the input serial data stream. CLKB should be connected to $\overline{CLK}$ in both SDR and DDR mode.

Divided Clock Input - CLKDIV

The divided clock input (CLKDIV) is typically a divided version of CLK (depending on the width of the implemented deserialization). It drives the output of the serial-to-parallel converter, the Bitslip submodule, and the CE module.

Serial Input Data from IOB - D

The serial input data port (D) is the serial (high-speed) data input port of the ISERDES_NODELAY. This port works in conjunction with all the Virtex-5 FPGA I/O resources to accommodate the desired I/O standards.

High-Speed Clock for Strobe-Based Memory Interfaces - OCLK

The OCLK clock input synchronizes data transfer in strobe-based memory interfaces. The OCLK of the ISERDES_NODELAY shares the same routing as the CLK port of the OSERDES.

The OCLK clock input is used to transfer strobe-based memory data onto a free-running clock domain. OCLK is a free-running FPGA clock at the same frequency as the strobe on the CLK input. The domain transfer from CLK to OCLK is shown in the Figure 8-5 block diagram. The timing of the domain transfer is set by the user by adjusting the delay of the strobe signal to the CLK input (e.g., using IDELAY). Examples of setting the timing of this domain transfer are given in several memory-related application notes, including XAPP858: *High-Performance DDR2 SDRAM Interface in Virtex-5 Devices*. When INTERFACE_TYPE is NETWORKING, this port is unused.

Reset Input - RST

The reset input causes the outputs of all data flip-flops in the CLK and CLKDIV domains to be driven Low asynchronously. ISERDES_NODELAY circuits running in the CLK domain where timing is critical use an internal, dedicated circuit to retime the RST input to produce a reset signal synchronous to the CLK domain. Similarly, there is a dedicated circuit to retime the RST input to produce a reset signal synchronous to the CLKDIV domain. Because there are ISERDES_NODELAY circuits that retime the RST input, the user is only required to provide a reset pulse to the RST input that meets timing on the CLKDIV

frequency domain. Therefore, RST should be driven High for a minimum of one CLKDIV cycle.

When building an interface consisting of multiple ISERDES_NODELAY ports, all ISERDES_NODELAY ports in the interface must be synchronized. The internal retiming of the RST input is designed so that all ISERDES_NODELAY blocks that receive the same reset pulse come out of reset synchronized with one another. The reset timing of multiple ISERDES_NODELAY ports is shown in Figure 8-9, page 363.

ISERDES_NODELAY Attributes

Table 8-2 summarizes all the applicable ISERDES_NODELAY attributes. A detailed description of each attribute follows the table. For more information on applying these attributes in UCF, VHDL, or Verilog code, refer to the Xilinx ISE Software Manual.

Table 8-2: **ISERDES_NODELAY Attributes**

Attribute Name	Description	Value	Default Value
BITSLIP_ENABLE	Allows the user to use the Bitslip submodule or bypass it. See "BITSLIP_ENABLE Attribute."	Boolean: "TRUE" or "FALSE"	FALSE
DATA_RATE	Enables incoming data stream to be processed as SDR or DDR data. See "DATA_RATE Attribute."	String: "SDR" or "DDR"	DDR
DATA_WIDTH	Defines the width of the serial-to-parallel converter. The legal value depends on the DATA_RATE attribute (SDR or DDR). See "DATA_WIDTH Attribute."	Integer: 2, 3, 4, 5, 6, 7, 8, or 10. If DATA_RATE = DDR, value is limited to 4, 6, 8, or 10. If DATA_RATE = SDR, value is limited to 2, 3, 4, 5, 6, 7, or 8.	4
INTERFACE_TYPE	Chooses the ISERDES_NODELAY use model. See "INTERFACE_TYPE Attribute."	String: "MEMORY" or "NETWORKING"	MEMORY
NUM_CE	Defines the number of clock enables. See "NUM_CE Attribute."	Integer: 1 or 2	2
SERDES_MODE	Defines whether the ISERDES_NODELAY module is a master or slave when using width expansion. See "SERDES_MODE Attribute."	String: "MASTER" or "SLAVE"	MASTER

BITSLIP_ENABLE Attribute

The BITSLIP_ENABLE attribute enables the Bitslip submodule. The possible values are TRUE and FALSE (default). BITSLIP_ENABLE must be set to TRUE when INTERFACE_TYPE is NETWORKING and FALSE when INTERFACE_TYPE is MEMORY. When set to TRUE, the Bitslip submodule responds to the BITSLIP signal. When set to FALSE, the Bitslip submodule is bypassed. See "BITSLIP Submodule."

DATA_RATE Attribute

The DATA_RATE attribute defines whether the incoming data stream is processed as single data rate (SDR) or double data rate (DDR). The allowed values for this attribute are SDR and DDR. The default value is DDR.

DATA_WIDTH Attribute

The DATA_WIDTH attribute defines the parallel data output width of the serial-to-parallel converter. The possible values for this attribute depend on the INTERFACE_TYPE and DATA_RATE attributes. See Table 8-3 for recommended data widths.

Table 8-3: **Recommended Data Widths**

INTERFACE_TYPE	DATA_RATE	Recommended Data Widths
NETWORKING	SDR	2, 3, 4, 5, 6, 7, 8
	DDR	4, 6, 8, 10
MEMORY	SDR	None
	DDR	4

When the DATA_WIDTH is set to widths larger than six, a pair of ISERDES_NODELAY must be configured into a master-slave configuration. See "ISERDES Width Expansion." Width expansion is not allowed in memory mode.

INTERFACE_TYPE Attribute

The INTERFACE_TYPE attribute determines whether the ISERDES_NODELAY is configured in memory or networking mode. The allowed values for this attribute are MEMORY or NETWORKING. The default mode is MEMORY.

When INTERFACE_TYPE is set to NETWORKING, the Bitslip submodule is available and the OCLK port is unused. BITSLIP_ENABLE must be set to TRUE, and the Bitslip port tied Low to disable Bitslip operation when the Bitslip module is not used in networking mode. When set to MEMORY, the Bitslip submodule is not available (BITSLIP_ENABLE must be set to FALSE), and the OCLK port can be used.

Figure 8-5 illustrates the ISERDES_NODELAY internal connections when in Memory mode.

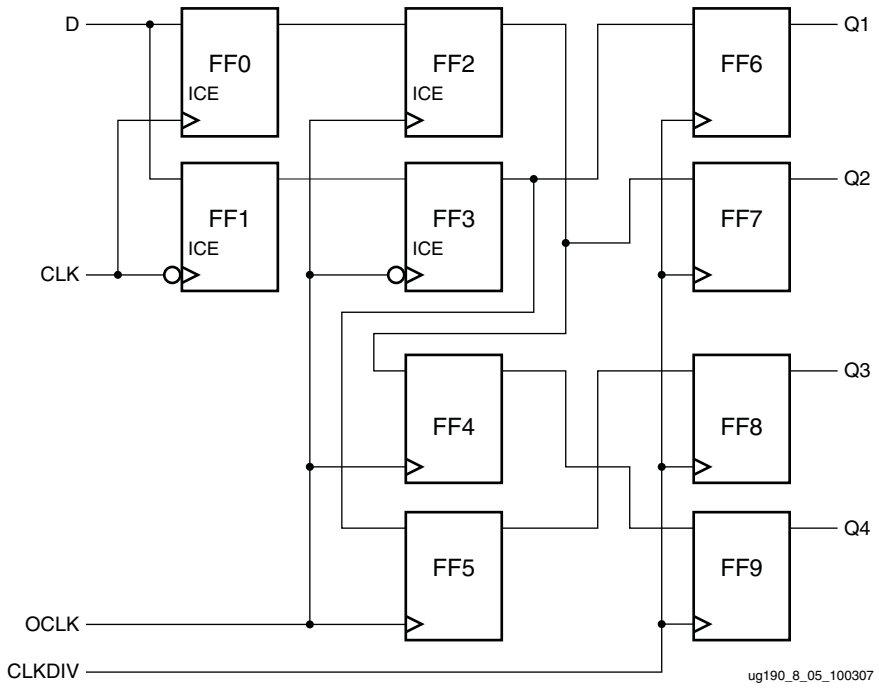


Figure 8-5: **Internal Connections of ISERDES_NODELAY When in Memory Mode**

NUM_CE Attribute

The NUM_CE attribute defines the number of clock enables (CE1 and CE2) used. The possible values are 1 and 2 (default = 2).

SERDES_MODE Attribute

The SERDES_MODE attribute defines whether the ISERDES_NODELAY module is a master or slave when using width expansion. The possible values are MASTER and SLAVE. The default value is MASTER. See "ISERDES Width Expansion."

ISERDES_NODELAY Clocking Methods

Networking Interface Type

The phase relationship of CLK and CLKDIV is important in the serial-to-parallel conversion process. CLK and CLKDIV are (ideally) phase-aligned within a tolerance. There are several clocking arrangements within the FPGA to help the design meet the phase relationship requirements of CLK and CLKDIV. The only valid clocking arrangements for the ISERDES_NODELAY block using the networking interface type are:

- CLK driven by BUFIO, CLKDIV driven by BUFR
- CLK driven by DCM, CLKDIV driven by the CLKDV output of the same DCM
- CLK driven by PLL, CLKDIV driven by CLKOUT[0:5] of same PLL

Memory Interface Type

The only valid clocking arrangements for the ISERDES_NODELAY block using the memory interface type are:

- CLK driven by BUFIO or BUFG
- OCLK driven by DCM and CLKDIV driven by CLKDV output of same DCM
- OCLK driven by PLL and CLKDIV driven by CLKOUT[0:5] of same PLL

The clocking arrangement using BUFIO and BUFR is shown in Figure 8-6. The CLK and CLKDIV inputs must be nominally phase-aligned. For example, if CLK and CLKDIV in Figure 8-6 were inverted by the designer at the ISERDES inputs, then although the clocking arrangement is a legal BUFIO/BUFR configuration, the clocks would still be out of phase. No phase relationship between CLK and OCLK is expected. Calibration must be performed for reliable data transfer from CLK to OCLK domain. "High-Speed Clock for Strobe-Based Memory Interfaces - OCLK" gives further information about transferring data between CLK and OCLK.

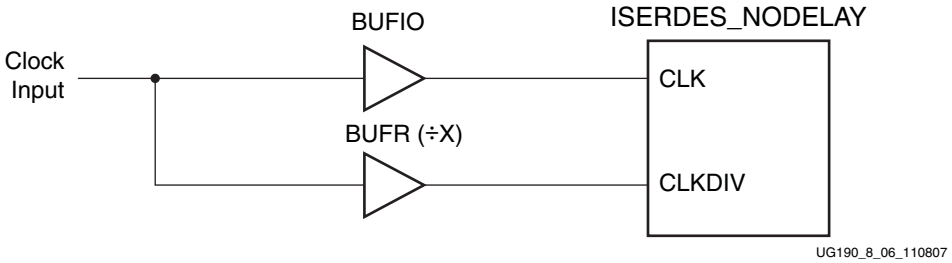


Figure 8-6: **Clocking Arrangement Using BUFIO and BUFR**

ISERDES Width Expansion

Two ISERDES modules are used to build a serial-to-parallel converter larger than 1:6. In every I/O tile there are two ISERDES modules; one master and one slave. By connecting the SHIFTOUT ports of the master ISERDES to the SHIFTIN ports of the slave ISERDES the serial-to-parallel converter can be expanded to up to 1:10 (DDR) and 1:8 (SDR).

Figure 8-7 illustrates a block diagram of a 1:10 DDR serial-to-parallel converter using the master and slave ISERDES modules. Ports Q3 - Q6 are used for the last four bits of the parallel interface on the slave ISERDES.

For a differential input, the master ISERDES must be on the positive side of the differential input pair. When the input is not differential, the input buffer associated with the slave ISERDES is not available and can not be used.

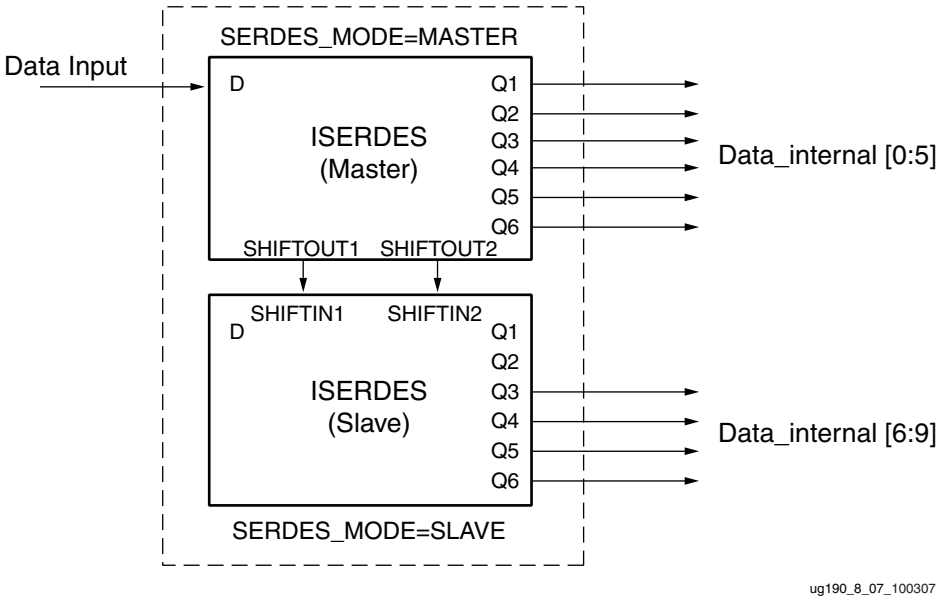


Figure 8-7: **Block Diagram of ISERDES Width Expansion**

Guidelines for Expanding the Serial-to-Parallel Converter Bit Width

1. Both ISERDES modules must be adjacent master and slave pairs. Both ISERDES modules must be in NETWORKING mode because width expansion is not available in MEMORY mode.
2. Set the SERDES_MODE attribute for the master ISERDES to MASTER and the slave ISERDES to SLAVE. See "SERDES_MODE Attribute."
3. The user must connect the SHIFTIN ports of the SLAVE to the SHIFTOUT ports of the MASTER.
4. The SLAVE only uses the ports Q3 to Q6 as an input.
5. DATA_WIDTH applies to both MASTER and SLAVE in Figure 8-7.

ISERDES Latencies

When the ISERDES interface type is MEMORY, the latency through the OCLK stage is one CLKDIV cycle. However, the total latency through the ISERDES depends on the phase relationship between the CLK and the OCLK clock inputs. When the ISERDES interface type is NETWORKING, the latency is two CLKDIV cycles. See Figure 8-12, page 366 and Figure 8-13, page 367 for a visualization of latency in networking mode. The extra CLKDIV cycle of latency in networking mode (compared to memory mode) is due to the Bitslip submodule.

ISERDES Timing Model and Parameters

Table 8-4 describes the function and control signals of the ISERDES switching characteristics in the *Virtex-5 FPGA Data Sheet*.

Table 8-4: **ISERDES Switching Characteristics**

Symbol	Description
Setup/Hold for Control Lines	
$T_{ISCCK\_BITSLIP}$/ $T_{ISCKC\_BITSLIP}$	BITSLIP pin Setup/Hold with respect to CLKDIV
T_{ISCCK_CE} /T_{ISCKC_CE}	CE pin Setup/Hold with respect to CLK (for CE1)
T_{ISCCK_CE} /T_{ISCKC_CE}	CE pin Setup/Hold with respect to CLKDIV (for CE2)
Setup/Hold for Data Lines	
T_{ISDCK_D} / T_{ISCKD_D}	D pin Setup/Hold with respect to CLK
	D pin Setup/Hold with respect to CLK
	D pin Setup/Hold with respect to CLK
T_{ISDCK_DDR} / T_{ISCKD_DDR}	D pin Setup/Hold with respect to CLK at DDR mode
	D pin Setup/Hold with respect to CLK at DDR mode
	D pin Setup/Hold with respect to CLK at DDR mode
Sequential Delay	
T_{ISCKO_Q}	CLKDIV to Out at Q pins

Timing Characteristics

Figure 8-8 illustrates an ISERDES timing diagram for the input data to the ISERDES. The timing parameter names change for different modes (SDR/DDR). However, the names do not change when a different bus input width, including when two ISERDES are cascaded together to form 10 bits. In DDR mode, the data input (D) switches at every CLK edge (rising and falling).

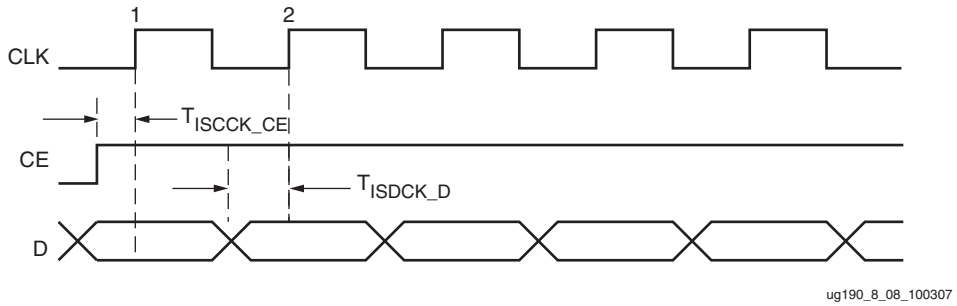


Figure 8-8: **ISERDES Input Data Timing Diagram**

Clock Event 1

- At time T_{ISCCK_CE}, before Clock Event 1, the clock enable signal becomes valid-High and the ISERDES can sample data.

Clock Event 2

- At time T_{ISDCK_D}, before Clock Event 2, the input data pin (D) becomes valid and is sampled at the next positive clock edge.

Reset Input Timing

Clock Event 1

As shown in Figure 8-9, the reset pulse is generated on the rising edge of CLKDIV. Because the pulse must take two different routes to get to ISERDES0 and ISERDES1, there are different propagation delays for both paths. The difference in propagation delay is emphasized. The path to ISERDES0 is very long and the path to ISERDES1 is very short, such that each ISERDES receives the reset pulse in a different CLK cycle. The internal resets for both CLK and CLKDIV are reset asynchronously when the RST input is asserted.

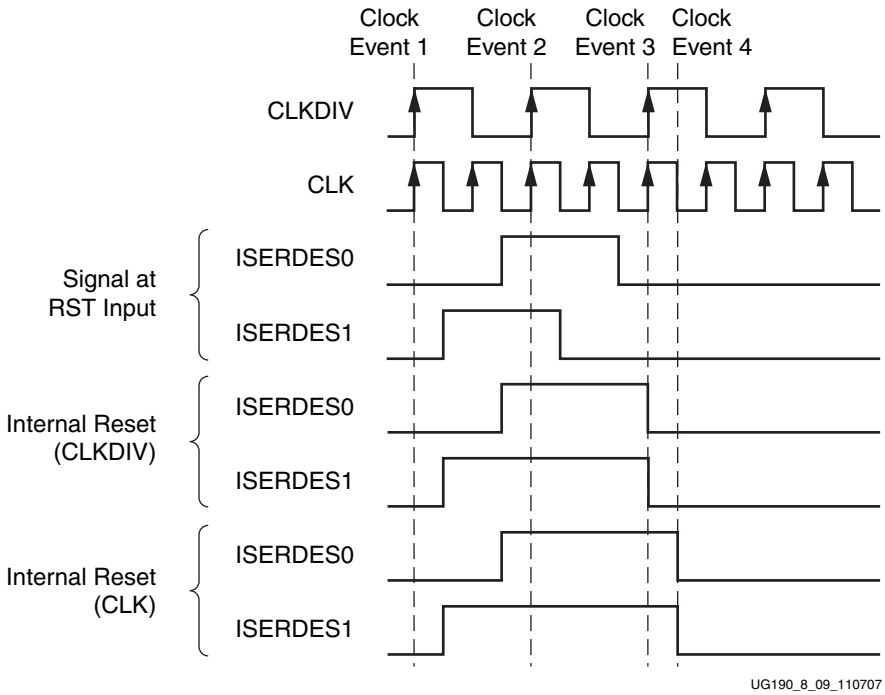


Figure 8-9: **Two ISERDES Coming Out of Reset Synchronously with One Another**

Clock Event 2

The reset pulse is deasserted on the rising edge of CLKDIV. The difference in propagation delay between the two ISERDES causes the RST input to come out of reset on two different CLK cycles. Without internal retiming, ISERDES1 finishes reset one CLK cycle before ISERDES0 and both ISERDES are asynchronous.

Clock Event 3

The release of the reset signal at the RST input is retimed internally to CLKDIV. This synchronizes ISERDES0 and ISERDES1.

Clock Event 4

The release of the reset signal at the RST input is retimed internally to CLK.

ISERDES VHDL and Verilog Instantiation Template

VHDL and Verilog instantiation templates are available in the Libraries Guide for all primitives and submodules.

In VHDL, each template has a component declaration section and an architecture section.

Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

BITSLIP Submodule

All ISERDES blocks in Virtex-5 devices contain a Bitslip submodule. This submodule is used for word-alignment purposes in source-synchronous networking-type applications. Bitslip reorders the parallel data in the ISERDES block, allowing every combination of a repeating serial pattern received by the deserializer to be presented to the FPGA fabric. This repeating serial pattern is typically called a training pattern (training patterns are supported by many networking and telecom standards).

Bitslip Operation

By asserting the Bitslip pin of the ISERDES block, the incoming serial data stream is reordered at the parallel side. This operation is repeated until the training pattern is seen. The tables in Figure 8-10 illustrate the effects of a Bitslip operation in SDR and DDR mode. For illustrative purposes the data width is eight. The Bitslip operation is synchronous to CLKDIV. In SDR mode, every Bitslip operation causes the output pattern to shift left by one. In DDR mode, every Bitslip operation causes the output pattern to alternate between a shift right by one and shift left by three. In this example, on the eighth Bitslip operation, the output pattern reverts to the initial pattern. This assumes that serial data is an eight bit repeating pattern.

Bitslip Operation in SDR Mode

Bitslip Operations Executed	Output Pattern (8:1)
Initial	10010011
1	00100111
2	01001110
3	10011100
4	00111001
5	01110010
6	11100100
7	11001001

Bitslip Operation in DDR Mode

Bitslip Operations Executed	Output Pattern (8:1)
Initial	00100111
1	10010011
2	10011100
3	01001110
4	01110010
5	00111001
6	11001001
7	11100100

ug190_8_10_100307

Figure 8-10: **Bitslip Operation Examples**

Figure 8-11 illustrates the ISERDES configured in 1:8 SDR mode with Bitslip_ENABLE set to TRUE. Two ISERDES modules are in a master-slave configuration for a data width of eight.

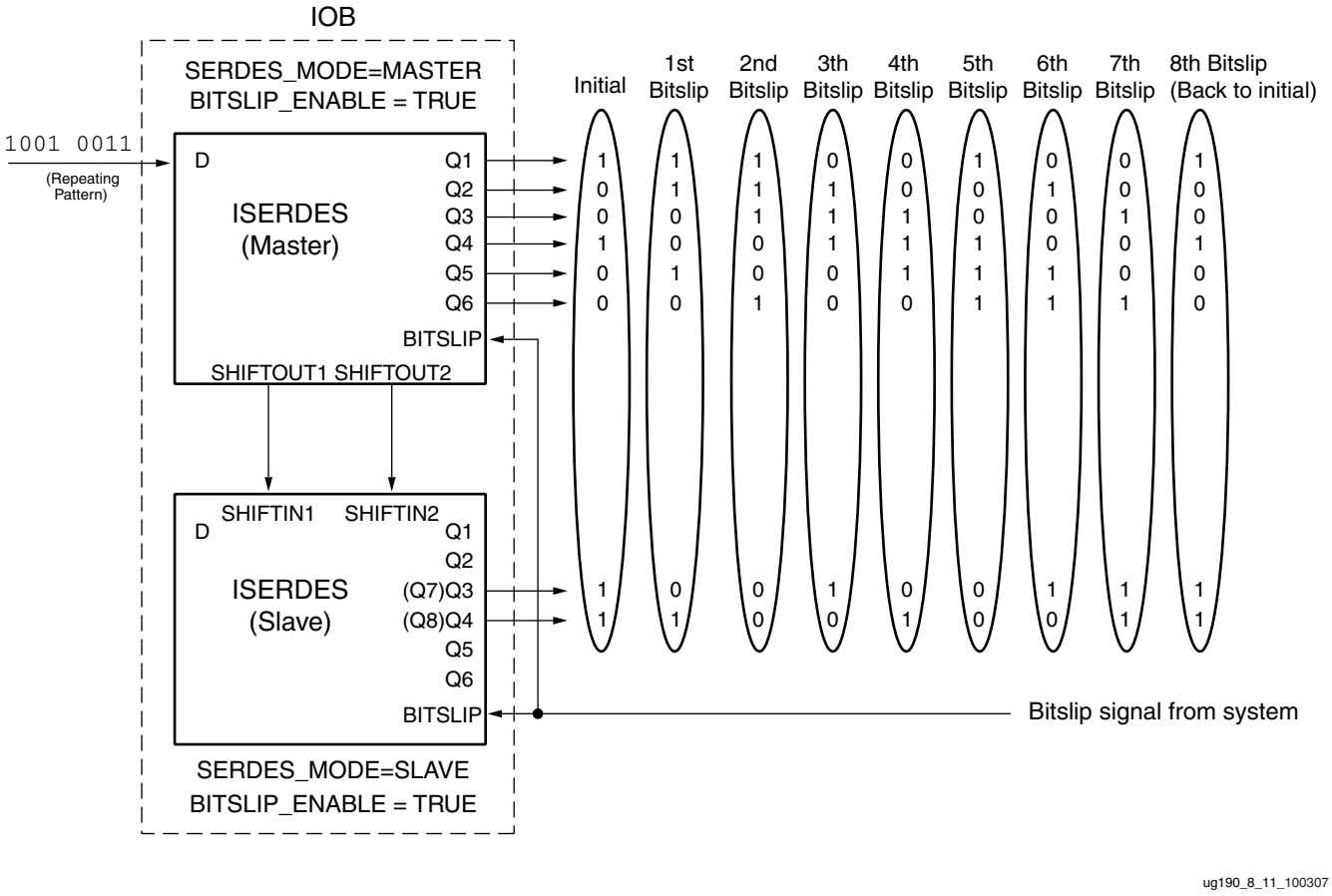


Figure 8-11: **Circuit Diagram for Bitslip Configuration in 1:8 SDR Mode**

Guidelines for Using the Bitslip Submodule

Set the BITSLIP_ENABLE attribute to TRUE. When BITSLIP_ENABLE is set to FALSE, the Bitslip pin has no effect. In a master-slave configuration, the BITSLIP_ENABLE attribute in both modules must be set to TRUE.

To invoke a Bitslip operation, the BITSLIP port must be asserted High for one CLKDIV cycle. In SDR mode, Bitslip cannot be asserted for two consecutive CLKDIV cycles; Bitslip must be deasserted for at least one CLKDIV cycle between two Bitslip assertions. In both SDR and DDR mode, the total latency from when the ISERDES captures the asserted Bitslip input to when the "bit-slipped" ISERDES outputs Q1–Q6 are sampled into the FPGA logic by CLKDIV is two CLKDIV cycles.

Bitslip Timing Model and Parameters

This section discusses the timing models associated with the Bitslip controller in a 1:4 DDR configuration. Data (D) is a repeating, 4-bit training pattern ABCD. ABCD could appear at the parallel outputs Q1–Q4 of the ISERDES in four possible ways: ABCD, BCDA, CDAB, and DABC. Only one of these four alignments of the parallel word makes sense to the user's downstream logic that reads the data from the Q1–Q4 outputs of the ISERDES. In this case, ABCD is assumed to be the word alignment that makes sense. Asserting Bitslip allows the user to see all possible configurations of ABCD and then choose the expected alignment (ABCD). Figure 8-12 shows the timing of two Bitslip operations and the corresponding re-alignments of the ISERDES parallel outputs Q1–Q4.

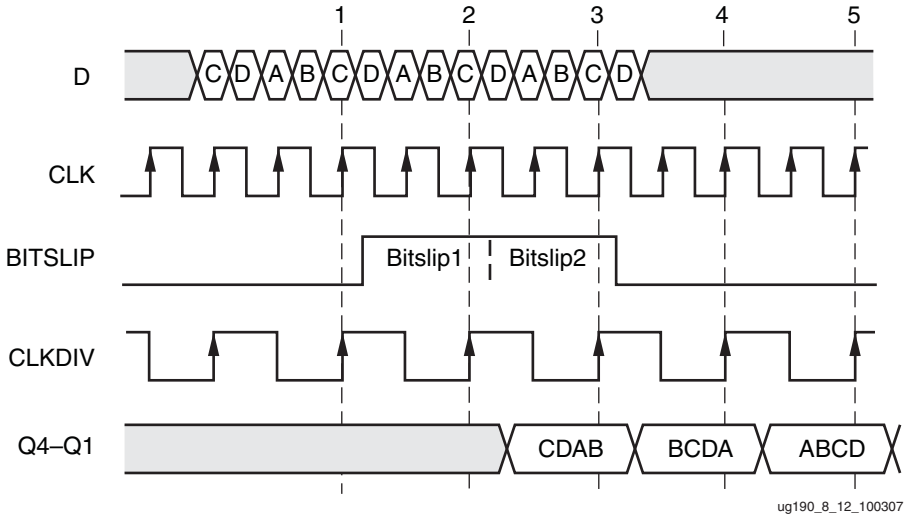


Figure 8-12: **Bitslip Timing Diagram**

Clock Event 1

The entire first word CDAB has been sampled into the input side registers of the ISERDES. The Bitslip pin is not asserted; the word propagates through the ISERDES without any realignment.

Clock Event 2

The second word CDAB has been sampled into the input side registers of the ISERDES. The Bitslip pin is asserted, which causes the Bitslip controller to shift all bits internally by one bit to the right.

Clock Event 3

The third word CDAB has been sampled into the input side registers of the ISERDES. The Bitslip pin is asserted for a second time, which causes the Bitslip controller to shift all bits internally by three bits to the left.

On this same edge of CLKDIV, the first word sampled is presented to Q1–Q4 without any realignment. The actual bits from the input stream that appear at the Q1–Q4 outputs during this cycle are shown in A of Figure 8-13.

Figure 8-13: **Bits from Data Input Stream (D) of Figure 8-12**

Clock Event 4

The first two bits of the fourth word CD have been sampled into the input side registers of the ISERDES. On this same edge of CLKDIV, the second word sampled is presented to Q1–Q4 with one bit shifted to the right. The actual bits from the input stream that appear at the Q1–Q4 outputs during this cycle are shown in B of Figure 8-13.

The realigned bits on Q1–Q4 are sampled into the FPGA logic on the CLKDIV domain. The total latency from when the ISERDES captures the asserted Bitslip input to when the realigned ISERDES outputs Q1–Q4 are sampled by CLKDIV is two CLKDIV cycles.

Clock Event 5

The third word sampled is presented to Q1–Q4 with three bits shifted to the left. The actual bits from the input stream that appear at the Q1–Q4 outputs during this cycle are shown in C of Figure 8-13.

Output Parallel-to-Serial Logic Resources (OSERDES)

The OSERDES in Virtex-5 devices is a dedicated parallel-to-serial converter with specific clocking and logic resources designed to facilitate the implementation of high-speed source-synchronous interfaces. Every OSERDES module includes a dedicated serializer for data and 3-state control. Both Data and 3-state serializers can be configured in SDR and DDR mode. Data serialization can be up to 6:1 (10:1 if using "OSERDES Width Expansion"). 3-state serialization can be up to 4:1.

Figure 8-14 shows a block diagram of the OSERDES, highlighting all the major components and features of the block.

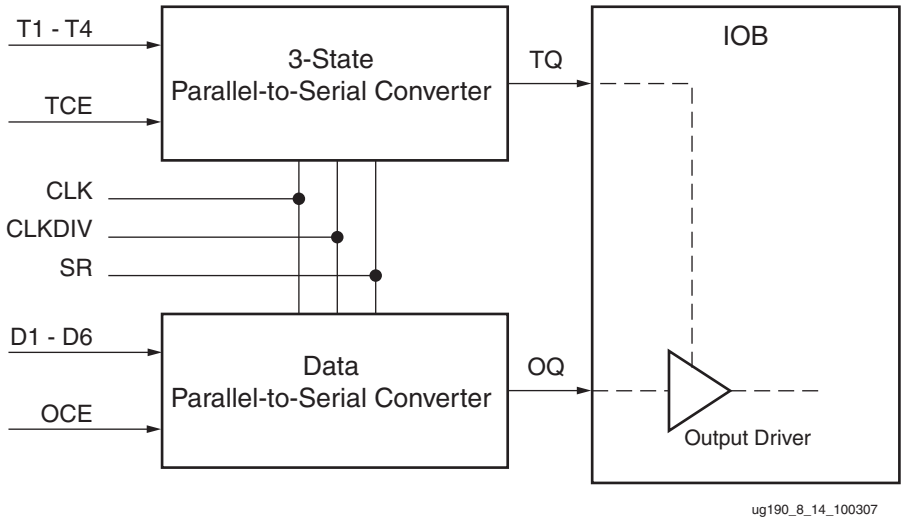


Figure 8-14: **OSERDES Block Diagram**

Data Parallel-to-Serial Converter

The data parallel-to-serial converter in one OSERDES blocks receives two to six bits of parallel data from the fabric (10:1 if using "OSERDES Width Expansion"), serializes the data, and presents it to the IOB via the OQ outputs. Parallel data is serialized from lowest order data input pin to highest (i.e., data on the D1 input pin is the first bit transmitted at the OQ pins). The data parallel-to-serial converter is available in two modes: single-data rate (SDR) and double-data rate (DDR).

The OSERDES uses two clocks, CLK and CLKDIV, for data rate conversion. CLK is the high-speed serial clock, CLKDIV is the divided parallel clock. It is assumed that CLK and CLKDIV are phase aligned.

Prior to use, a reset must be applied to the OSERDES. The OSERDES contains an internal counter that controls dataflow. Failure to synchronize the reset with the CLKDIV will produce an unexpected output. Table 8-5 describes the relationship between CLK and CLKDIV in all modes.

Table 8-5: **CLK/CLKDIV Relationship of the Data Parallel-to-Serial Converter**

Input Data Width Output in SDR Mode	Input Data Width Output in DDR Mode	CLK	CLKDIV
2	4	2X	X
3	6	3X	X
4	8	4X	X
5	10	5X	X
6	–	6X	X
7	–	7X	X
8	–	8X	X

3-State Parallel-to-Serial Conversion

In addition to parallel-to-serial conversion of data, an OSERDES module also contains a parallel-to-serial converter for 3-state control of the IOB. Unlike data conversion, the 3-state converter can only serialize up to four bits of parallel 3-state signals. The 3-state converter cannot be cascaded.

OSERDES Primitive

The OSERDES primitive is shown in Figure 8-15.

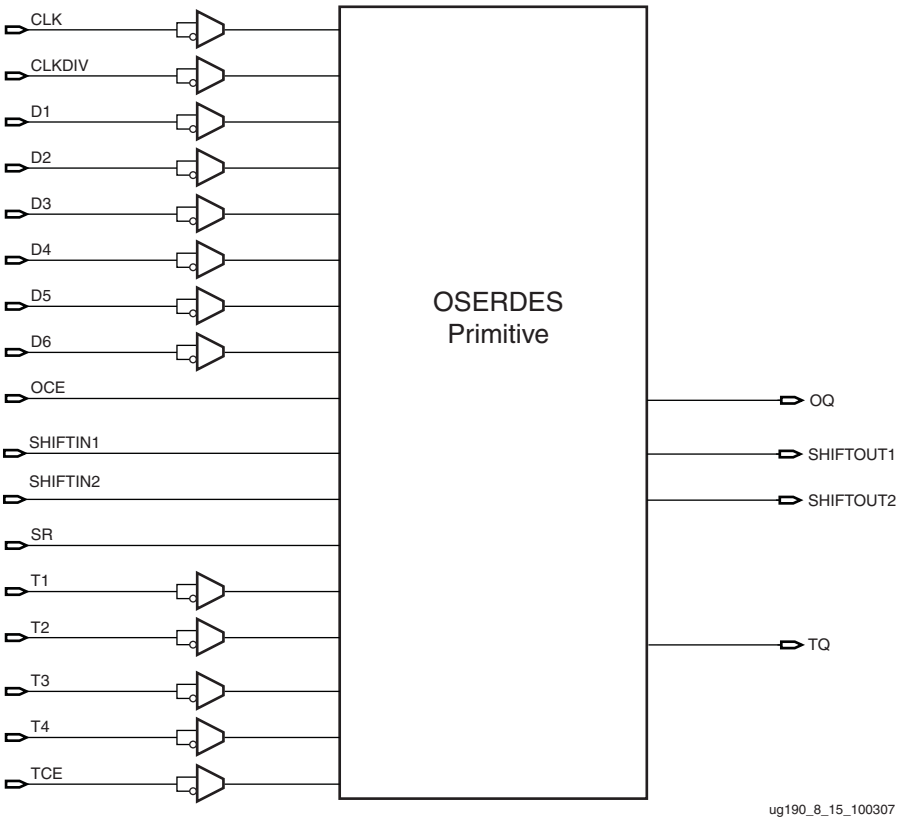


Figure 8-15: **OSERDES Primitive**

OSERDES Ports

Table 8-6 lists the available ports in the OSERDES primitive.

Table 8-6: **OSERDES Port List and Definitions**

Port Name	Type	Width	Description
OQ	Output	1	Data path output. See "Data Path Output - OQ."
SHIFTOUT1	Output	1	Carry out for data width expansion. Connect to SHIFTIN1 of master OSERDES. See "OSERDES Width Expansion."
SHIFTOUT2	Output	1	Carry out for data width expansion. Connect to SHIFTIN2 of master OSERDES. See "OSERDES Width Expansion."
TQ	Output	1	3-state control output. See "3-state Control Output - TQ."
CLK	Input	1	High-speed clock input. See "High-Speed Clock Input - CLK."
CLKDIV	Input	1	Divided clock input. Clocks delay element, deserialized data, Bitslip submodule, and CE unit. See "Divided Clock Input - CLKDIV."
D1 – D6	Input	1 (each)	Parallel data inputs. See "Parallel Data Inputs - D1 to D6."
OCE	Input	1	Output data clock enable. See "Output Data Clock Enable - OCE."
REV	Input	1	Reverse SR pin. Not available in the OSERDES block.
SHIFTIN1	Input	1	Carry input for data width expansion. Connect to SHIFTOUT1 of slave OSERDES. See "OSERDES Width Expansion."
SHIFTIN2	Input	1	Carry input for data width expansion. Connect to SHIFTOUT2 of slave OSERDES. See "OSERDES Width Expansion."
SR	Input	1	Active High reset.
T1 to T4	Input	1 (each)	Parallel 3-state inputs. See "Parallel 3-state Inputs - T1 to T4."
TCE	Input	1	3-state clock enable. See "3-state Signal Clock Enable - TCE."

Data Path Output - OQ

The OQ port is the data output port of the OSERDES module. Data at the input port D1 will appear first at OQ. This port connects the output of the data parallel-to-serial converter to the data input of the IOB.

3-state Control Output - TQ

This port is the 3-state control output of the OSERDES module. When used, this port connects the output of the 3-state parallel-to-serial converter to the control/3-state input of the IOB.

High-Speed Clock Input - CLK

This high speed clock input drives the serial side of the parallel-to-serial converters.

Divided Clock Input - CLKDIV

This divided high-speed clock input drives the parallel side of the parallel-to-serial converters. This clock is the divided version of the clock connected to the CLK port.

Parallel Data Inputs - D1 to D6

All incoming parallel data enters the OSERDES module through ports D1 to D6. These ports are connected to the FPGA fabric, and can be configured from two to six bits (i.e., a 6:1 serialization). Bit widths greater than six (up to 10) can be supported by using a second OSERDES in SLAVE mode. See "OSERDES Width Expansion." Refer to Figure 8-3, page 354 for bit ordering at the inputs and output of the OSERDES along with the corresponding bit order of the ISERDES_NODELAY.

Output Data Clock Enable - OCE

OCE is an active High clock enable for the data path.

Parallel 3-state Inputs - T1 to T4

All parallel 3-state signals enter the OSERDES module through ports T1 to T4. The ports are connected to the FPGA fabric, and can be configured as one, two, or four bits.

3-state Signal Clock Enable - TCE

TCE is an active High clock enable for the 3-state control path.

Reset Input - SR

The reset input causes the outputs of all data flip-flops in the CLK and CLKDIV domains to be driven Low asynchronously. OSERDES circuits running in the CLK domain where timing is critical use an internal, dedicated circuit to retime the SR input to produce a reset signal synchronous to the CLK domain. Similarly, there is a dedicated circuit to retime the SR input to produce a reset signal synchronous to the CLKDIV domain. Because there are OSERDES circuits that retime the SR input, the user is only required to provide a reset pulse to the SR input that meets timing on the CLKDIV frequency domain (synchronous to CLKDIV). Therefore, SR should be driven High for a minimum of one CLKDIV cycle.

When building an interface consisting of multiple OSERDES ports, all OSERDES ports must be synchronized. The internal retiming of the SR input is designed so that all OSERDES blocks that receive the same reset pulse come out of reset synchronized with one another. The reset timing of multiple OSERDES ports is shown in Figure 8-20, page 379.

OSERDES Attributes

Table 8-7 lists and describes the various attributes that are available for the OSERDES primitive. The table includes the default values.

Table 8-7: **OSERDES Attribute Summary**

OSERDES Attribute	Description	Value	Default Value
DATA_RATE_OQ	Defines whether data (OQ) changes at every clock edge or every positive clock edge with respect to CLK.	String: SDR or DDR	DDR
DATA_RATE_TQ	Defines whether the 3-state (TQ) changes at every clock edge, every positive clock edge with respect to clock, or is set to buffer configuration.	String: BUF, SDR, or DDR	DDR
DATA_WIDTH	Defines the parallel-to-serial data converter width. This value also depends on the DATA_RATE_OQ value.	Integer: 2, 3, 4, 5, 6, 7, 8, or 10. If DATA_RATE_OQ = DDR, value is limited to 4, 6, 8, or 10. If DATA_RATE_OQ = SDR, value is limited to 2, 3, 4, 5, 6, 7, or 8.	4
SERDES_MODE	Defines whether the OSERDES module is a master or slave when using width expansion.	String: MASTER or SLAVE	MASTER
TRISTATE_WIDTH	Defines the parallel to serial 3-state converter width.	Integer: 1 or 4 If DATA_RATE_TQ = DDR, DATA_WIDTH = 4, and DATA_RATE_OQ = DDR, value is limited to 4. For all other settings of DATA_RATE_TQ, DATA_WIDTH, and DATA_RATE_OQ, value is limited to 1.	4

DATA_RATE_OQ Attribute

The DATA_RATE_OQ attribute defines whether data is processed as single data rate (SDR) or double data rate (DDR). The allowed values for this attribute are SDR and DDR. The default value is DDR.

DATA_RATE_TQ Attribute

The DATA_RATE_TQ attribute defines whether 3-state control is to be processed as single data rate (SDR) or double data rate (DDR). The allowed values for this attribute are SDR and DDR. The default value is DDR.

DATA_WIDTH Attribute

The DATA_WIDTH attribute defines the parallel data input width of the parallel-to-serial converter. The possible values for this attribute depend on the DATA_RATE_OQ attribute. When DATA_RATE_OQ is set to SDR, the possible values for the DATA_WIDTH attribute are 2, 3, 4, 5, 6, 7, and 8. When DATA_RATE_OQ is set to DDR, the possible values for the DATA_WIDTH attribute are 4, 6, 8, and 10.

When the DATA_WIDTH is set to widths larger than six, a pair of OSERDES must be configured into a master-slave configuration. See "OSERDES Width Expansion."

SERDES_MODE Attribute

The SERDES_MODE attribute defines whether the OSERDES module is a master or slave when using width expansion. The possible values are MASTER and SLAVE. The default value is MASTER. See "OSERDES Width Expansion."

TRISTATE_WIDTH Attribute

The TRISTATE_WIDTH attribute defines the parallel 3-state input width of the 3-state control parallel-to-serial converter. The possible values for this attribute depend on the DATA_RATE_TQ attribute. When DATA_RATE_TQ is set to SDR or BUF, the TRISTATE_WIDTH attribute can only be set to 1. When DATA_RATE_TQ is set to DDR, the possible values for the TRISTATE_WIDTH attribute is 4.

TRISTATE_WIDTH cannot be set to widths larger than 4. When a DATA_WIDTH is larger than four, set the TRISTATE_WIDTH to 1.

OSERDES Clocking Methods

The phase relationship of CLK and CLKDIV is important in the parallel-to-serial conversion process. CLK and CLKDIV are (ideally) phase-aligned within a tolerance.

There are several clocking arrangements within the FPGA to help the design meet the phase relationship requirements of CLK and CLKDIV. The only valid clocking arrangements for the OSERDES are:

- CLK driven by BUFIO, CLKDIV driven by BUFR
- CLK driven by DCM, CLKDIV driven by the CLKDV output of the same DCM
- CLK driven by PLL, CLKDIV driven by CLKOUT[0:5] of same PLL

OSERDES Width Expansion

Two OSERDES modules are used to build a parallel-to-serial converter larger than 6:1. In every I/O tile there are two OSERDES modules; one master and one slave. By connecting the SHIFTIN ports of the master OSERDES to the SHIFTOUT ports of the slave OSERDES, the parallel-to-serial converter can be expanded to up to 10:1(DDR) and 8:1 (SDR). For a differential output, the master OSERDES must be on the positive side of the differential output pair. When the output is not differential, the output buffer associated with the slave OSERDES is not available and can not be used.

When using the OSERDES with width expansion, complementary single-ended standards (e.g., DIFF_HSTL and DIFF_SSTL) cannot be used. This is because both OLOGIC blocks in an I/O tile are used by the complementary single-ended standards to transmit both legs of the signal, leaving no OLOGIC blocks available for width expansion.

Figure 8-16 illustrates a block diagram of a 10:1 DDR parallel-to-serial converter using the master and slave OSERDES modules. Ports Q3-Q6 are used for the last four bits of the parallel interface on the slave OSERDES (LSB to MSB).

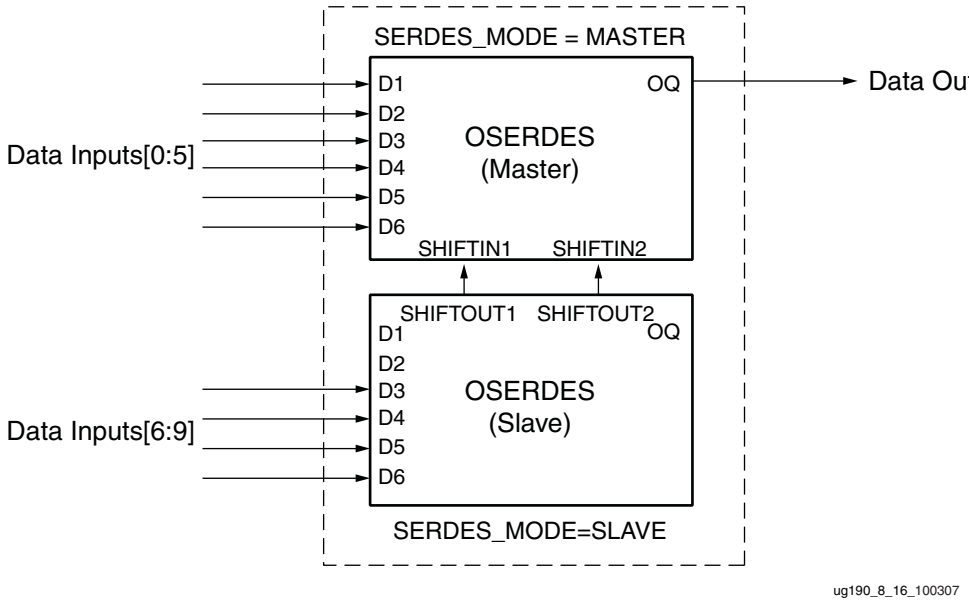


Figure 8-16: **Block Diagram of OSERDES Width Expansion**

Table 8-8 lists the data width availability for SDR and DDR mode.

Table 8-8: **OSERDES SDR/DDR Data Width Availability**

SDR Data Widths	2, 3, 4, 5, 6, 7, 8
DDR Data Widths	4, 6, 8, 10

Guidelines for Expanding the Parallel-to-Serial Converter Bit Width

1. Both the OSERDES modules must be adjacent master and slave pairs.
2. Set the SERDES_MODE attribute for the master OSERDES to MASTER and the slave OSERDES to SLAVE. See "SERDES_MODE Attribute."
3. The user must connect the SHIFTIN ports of the MASTER to the SHIFTOUT ports of the SLAVE.
4. The SLAVE only uses the ports D3 to D6 as an input.
5. DATA_WIDTH for Master and Slave are equal. See "DATA_WIDTH Attribute."

The slave inputs used for data widths requiring width expansion are listed in Table 8-9.

Table 8-9: **Slave Inputs Used for Data Width Expansion**

Data Width	Slave Inputs Used
7	D3
8	D3–D4
10	D3–D6

OSERDES Latencies

The input to output latencies of OSERDES blocks depend on the DATA_RATE and DATA_WIDTH attributes. Latency is defined as a period of time between the following two events: (a) when the rising edge of CLKDIV clocks the data at inputs D1–D6 into the OSERDES, and (b) when the first bit of the serial stream appears at OQ. Table 8-10 summarizes the various OSERDES latency values.

Table 8-10: **OSERDES Latencies**

DATA_RATE	DATA_WIDTH	Latency
SDR	2:1	1 CLK cycle
	3:1	3 CLK cycles
	4:1	4 CLK cycles
	5:1	4 CLK cycles
	6:1	5 CLK cycles
	7:1	5 CLK cycles
	8:1	6 CLK cycles
DDR	4:1	1 CLK cycle
	6:1	3 CLK cycles
	8:1	4 CLK cycles
	10:1	4 CLK cycles

OSERDES Timing Model and Parameters

This section discusses all timing models associated with the OSERDES primitive. Table 8-11 describes the function and control signals of the OSERDES switching characteristics in the *Virtex-5 FPGA Data Sheet*.

Table 8-11: **OSERDES Switching Characteristics**

Symbol	Description
Setup/Hold	
T_{OSDCK_D}/T_{OSCKD_D}	D input Setup/Hold with respect to CLKDIV
T_{OSDCK_T}/T_{OSCKD_T}	T input Setup/Hold with respect to CLK
T_{OSDCK_T}/T_{OSCKD_T}	T input Setup/Hold with respect to CLKDIV
$T_{OSCCK\_OCE}/T_{OSCKC\_OCE}$	OCE input Setup/Hold with respect to CLK
$T_{OSCCK\_TCE}/T_{OSCKC\_TCE}$	TCE input Setup/Hold with respect to CLK
Sequential Delays	
T_{OSCKO_OQ}	Clock to Out from CLK to OQ
T_{OSCKO_TQ}	Clock to Out from CLK to TQ

Table 8-11: **OSERDES Switching Characteristics *(Continued)***

Symbol	Description
Combinatorial	
T_{OSCO_OQ}	Asynchronous Reset to OQ
T_{OSCO_TQ}	Asynchronous Reset to TQ

Timing Characteristics of 2:1 SDR Serialization

In Figure 8-17, the timing of a 2:1 SDR data serialization is illustrated.

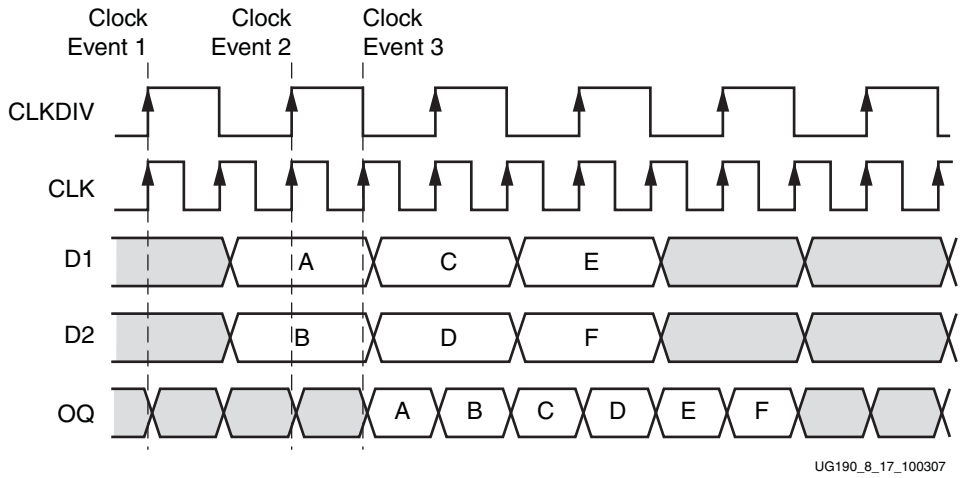


Figure 8-17: **OSERDES Data Flow and Latency in 2:1 SDR Mode**

Clock Event 1

On the rising edge of CLKDIV, the word *AB* is driven from the FPGA logic to the D1 and D2 inputs of the OSERDES (after some propagation delay).

Clock Event 2

On the rising edge of CLKDIV, the word *AB* is sampled into the OSERDES from the D1 and D2 inputs.

Clock Event 3

The data bit A appears at OQ one CLK cycle after *AB* is sampled into the OSERDES. This latency is consistent with the Table 8-10 listing of a 2:1 SDR mode OSERDES latency of one CLK cycle.

Timing Characteristics of 8:1 DDR Serialization

Figure 8-18 illustrates the timing of an 8:1 DDR data serialization. In contrast to the 2:1 SDR example, a second OSERDES is required to achieve an 8:1 serialization. The two OSERDES are connected and configured using the methods described in "OSERDES Width Expansion," page 373. Six of the eight bits are connected to D1–D6 of the master OSERDES while the remaining two bits are connected to D3–D4 of the slave OSERDES.

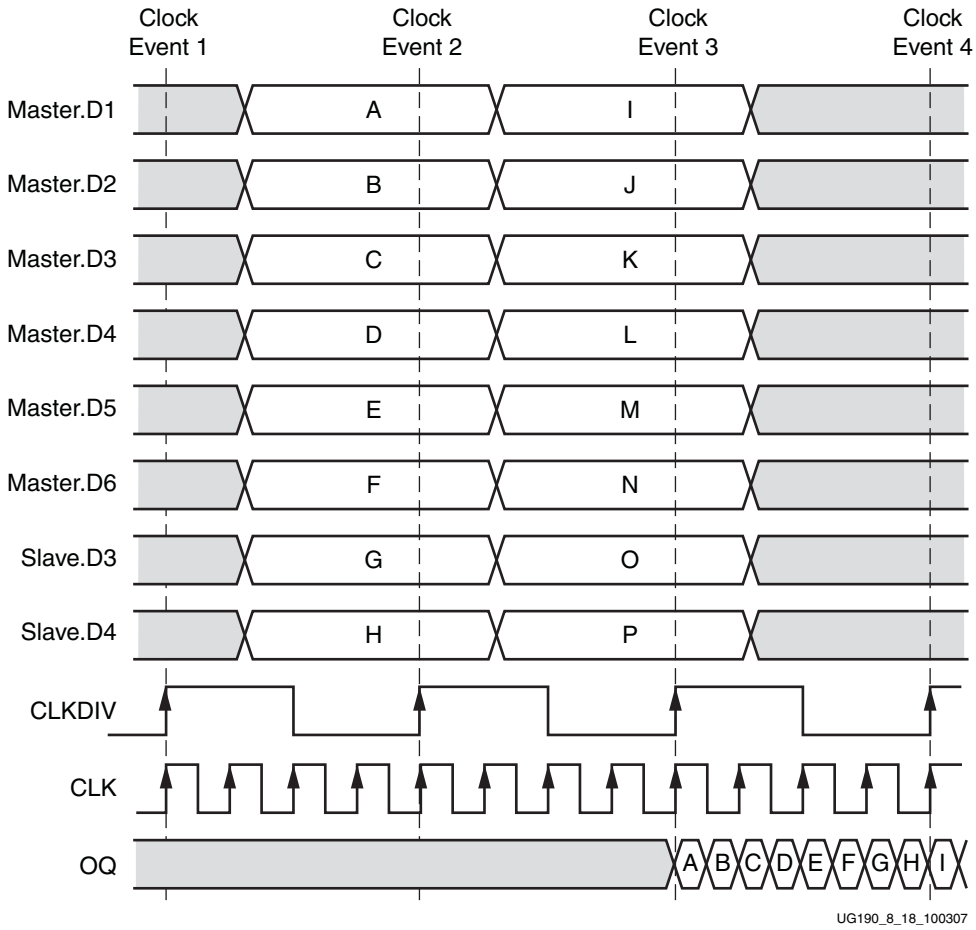


Figure 8-18: **OSERDES Data Flow and Latency in 8:1 DDR Mode**

Clock Event 1

On the rising edge of CLKDIV, the word *ABCDEFGH* is driven from the FPGA logic to the D1–D6 inputs of the master OSERDES and D3–D4 of the slave OSERDES (after some propagation delay).

Clock Event 2

On the rising edge of CLKDIV, the word *ABCDEFGH* is sampled into the master and slave OSERDES from the D1–D6 and D3–D4 inputs, respectively.

Clock Event 3

The data bit *A* appears at OQ four CLK cycles after *ABCDEFGH* is sampled into the OSERDES. This latency is consistent with the Table 8-10 listing of a 8:1 DDR mode OSERDES latency of four CLK cycles.

The second word *IJKLMNOP* is sampled into the master and slave OSERDES from the D1–D6 and D3–D4 inputs, respectively.

Clock Event 4

Between Clock Events 3 and 4, the entire word *ABCDEFGH* is transmitted serially on OQ, a total of eight bits transmitted in one CLKDIV cycle.

The data bit *I* appears at OQ four CLK cycles after *IJKLMNOP* is sampled into the OSERDES. This latency is consistent with the Table 8-10 listing of a 8:1 DDR mode OSERDES latency of four CLK cycles.

Timing Characteristics of 4:1 DDR 3-State Controller Serialization

The operation of a 3-State Controller is illustrated in Figure 8-19. The example is a 4:1 DDR case shown in a bidirectional system where the IOB must be frequently 3-stated.

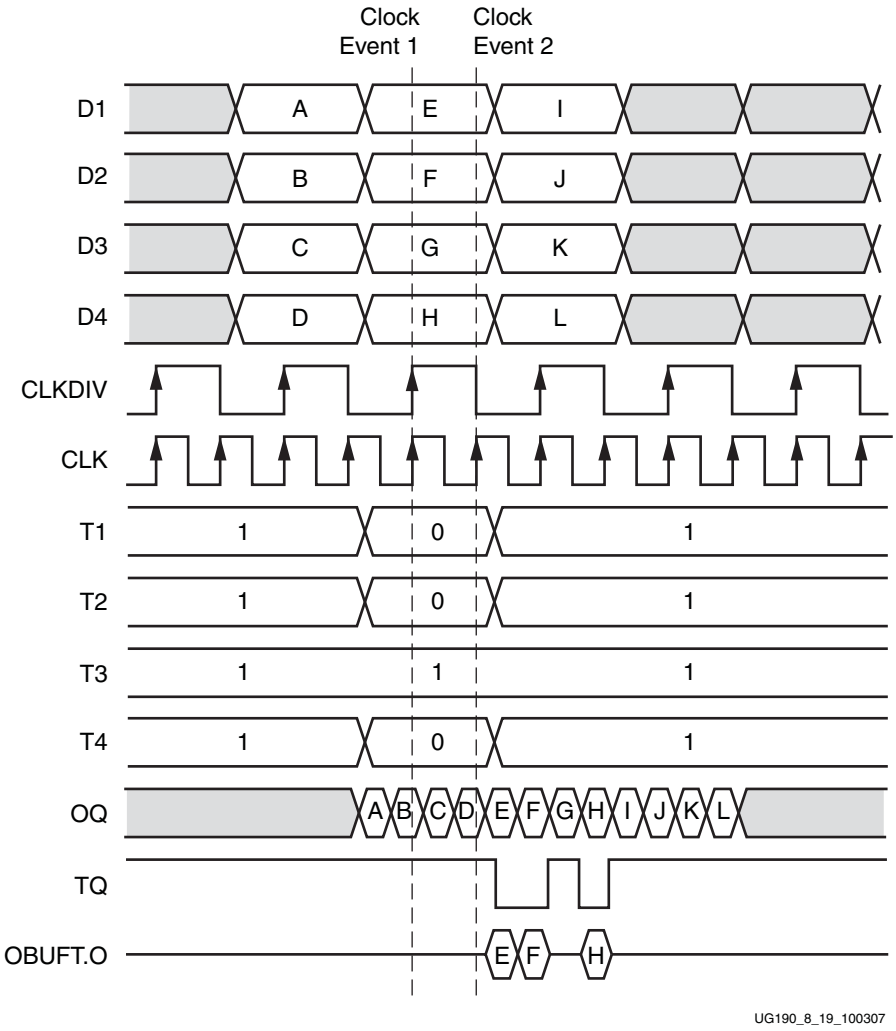


Figure 8-19: **OSERDES Data Flow and Latency in 4:1 DDR Mode**

Clock Event 1

T1, T2, and T4 are driven Low to release the 3-state condition. The serialization paths of T1–T4 and D1–D4 in the OSERDES are identical (including latency), such that the bits *EFGH* are always aligned with the `0010` presented at the T1–T4 pins during Clock Event 1.

Clock Event 2

The data bit *E* appears at OQ one CLK cycle after *EFGH* is sampled into the OSERDES. This latency is consistent with the Table 8-10 listing of a 4:1 DDR mode OSERDES latency of one CLK cycle.

The 3-state bit 0 at T1 during Clock Event 1 appears at TQ one CLK cycle after `0010` is sampled into the OSERDES 3-state block. This latency is consistent with the Table 8-10 listing of a 4:1 DDR mode OSERDES latency of one CLK cycle.

Reset Output Timing

Clock Event 1

A reset pulse is generated on the rising edge of CLKDIV. Because the pulse must take two different routes to get to OSERDES0 and OSERDES1, there are different propagation delays for both paths. The difference in propagation delay is emphasized in Figure 8-20. The path to OSERDES0 is very long and the path to OSERDES1 is very short, such that each OSERDES receives the reset pulse in a different CLK cycle. The internal resets for both CLK and CLKDIV go into reset asynchronously when the SR input is asserted.

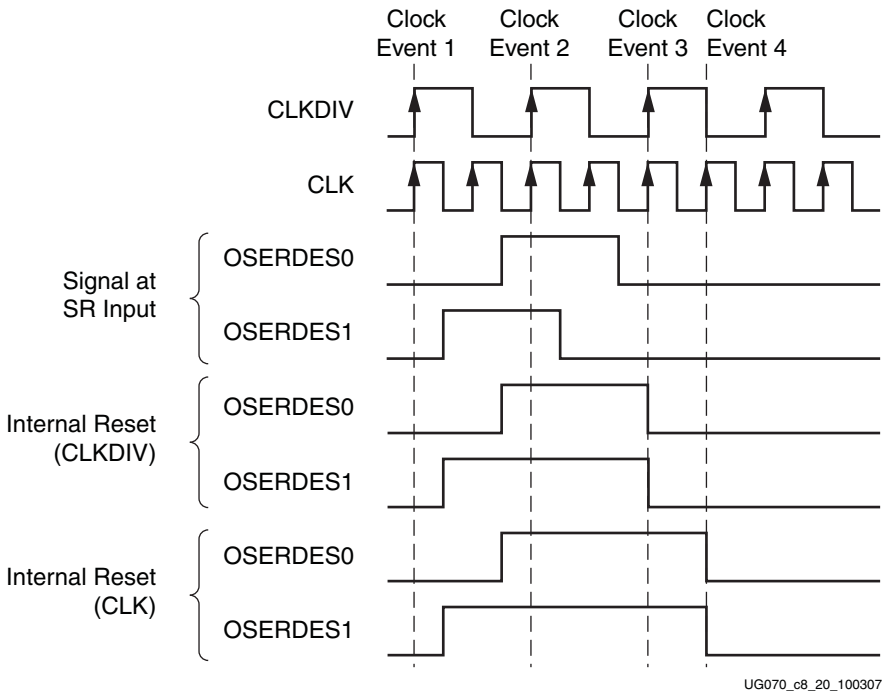


Figure 8-20: **Two OSERDES Coming Out of Reset Synchronously with One Another**

Clock Event 2

The reset pulse is deasserted on the rising edge of CLKDIV. The difference in propagation delay between the two OSERDES causes the SR input to come out of reset on two different CLK cycles. Without internal retiming, OSERDES1 finishes reset one CLK cycle before OSERDES0 and both OSERDES are asynchronous.

Clock Event 3

The release of the reset signal at the SR input is retimed internally to CLKDIV. This synchronizes OSERDES0 and OSERDES1.

Clock Event 4

The release of the reset signal at the SR input is retimed internally to CLK.

OSERDES VHDL and Verilog Instantiation Templates

The Libraries Guide includes instantiation templates of the OSERDES module in VHDL and Verilog.

Index

S

W