

Spartan-6 FPGA Clocking Resources

User Guide

UG382 (v1.8) June 20, 2013



Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.

© Copyright 2009–2013 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Vivado, Virtex, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|----------|---------|--|
| 06/24/09 | 1.0 | Initial Xilinx release. |
| 08/17/09 | 1.1 | <p>In Chapter 1: Deleted Figure 1-1, Overview of Global Clock Connections. Changed the Global Clocking Infrastructure section including revisions in Table 1-1 and Table 1-2, page 15. Revised the Spanning a Full Bank with a Single Global Clock Input With Two I/O Clocks and Clock Inputs discussions and Figure 1-7 and Figure 1-8. Added Figure 1-11, page 28. Removed BUFIN2 from Table 1-10. Removed exception from BUFGMUX_1, page 40. Added Clock Buffers for the High-Speed I/O Clock Region, page 43 including Table 1-16 and Table 1-17. Updated Table 1-18 through Table 1-25 and added or revised Figure 1-32, Figure 1-33, Figure 1-35, and Figure 1-36. Added section: Examples of High-Speed I/O Clock Network Connections, page 32.</p> <p>In Chapter 2: Updated XC6SLX4 resources in Table 2-1 and Table 2-2. Added note to Table 2-3, clarified descriptions in Phase Shift, page 64. Updated CLKIN_PERIOD description in Table 2-7. Updated the DCM_CLKGEN Primitive feature list. Updated Table 2-11 to all frequency ranges.</p> <p>In Chapter 3: Updated CLKIN2 and CLKINSEL descriptions in Table 3-4. Updated discussion leading to Figure 3-5. Typographical edits in Figure 3-16.</p> |

| Date | Version | Revision |
|----------|---------|--|
| 01/04/10 | 1.2 | <p>Added clarification to the Clock Resources section. Updated Table 1-1 and Table 1-2. Added Figure 1-3. Added Table 1-3, Table 1-4, and Table 1-6. Fixed Figure 1-7, Figure 1-8, and Figure 1-9. Added Table 1-8. Added Clocking Structure Guidelines. Moved Examples of High-Speed I/O Clock Network Connections. Added Figure 1-19. Added Table 1-14 and Figure 1-23 and Figure 1-24. Updated I/O Clock Network Inputs for BUFIO2 and BUFIO2_2CLK in Table 1-17. Added GTP_DUAL to Table 1-17. Updated BUFGMUX_1 section. Updated the definition of the input on Table 1-18. Updated Figure 1-32, Figure 1-33, and Figure 1-36. Updated GCLK description in Table 1-22. Added ENABLE_SYNC to Table 1-23 and Table 1-25.</p> <p>Added Table 1-12 and Figure 1-15 and Figure 1-16.</p> <p>In Table 2-9, updated SPREAD_SPECTRUM, added CLKFX_MD_MAX, and Note 1. Updated Spread-Spectrum Clock Generation section.</p> <p>Updated Figure 3-1. Added clarification on BUFIO2FB under Equation 3-1. Updated description of CLKOUT[0:5]_PHASE. Added BUFIO2 to PLL Clock Input Signals.</p> |
| 02/22/10 | 1.3 | <p>Updated the BUFIO2 clocking regions in Table 1-3, Table 1-4, Figure 1-7, Figure 1-9 and Figure 1-11. Removed note 1 from Table 1-8. Revised Figure 1-19 and added Example 7 including Figure 1-20. Added further discussion to Clock Buffers and Multiplexers.</p> <p>Updated the STATUS[7:3] description in Table 2-6 and added STATUS[7:3] to Table 2-8. Added description of reset circuit for lower-power devices to RST Input Behavior section.</p> <p>Updated Figure 3-3 and added Figure 3-4. Adding equations Equation 3-2 through Equation 3-6. Revised Equation 3-7 and Equation 3-9 and added Equation 3-8. Updated the description of EXTERNAL compensation in Table 3-5.</p> |
| 04/28/10 | 1.3.1 | Fixed Figure 1-24 . |
| 08/24/10 | 1.4 | <p>Added BUFH descriptions and Figure 1-2 to Clock Resources. Revised direct connections in Figure 1-3 and Figure 1-4. Added Figure 1-6 and updated the I/O Clocking Infrastructure section with BUFIO2 clocking region descriptions. Moved GCLK input description to Table 1-5. Added Figure 1-10 to describe the XC6SLX25 and XC6SLX25T BUFIO2 clocking regions. Added pin planning considerations when using BUFIO2 clocking regions and V_{CCO} bank restrictions. Added XC6SLX25 and XC6SLX25T BUFIO2 clocking regions to Table 1-8. Updated Clocking Structure Guidelines. Updated Figure 1-20. Added Global Clock Input Buffer Primitives section and Table 1-9. Added BUFH. Updated the I/O clock network discussion and added Figure 1-30. Updated Figure 1-32, Figure 1-33, and Figure 1-36 waveforms and buffers. Updated BUFPLL description and added Figure 1-38. Updated Table 1-24. Clarified descriptions in BUFIO2FB. Removed CLK_REF from Figure 1-40 and Figure 3-5.</p> <p>Updated LOCKED in Table 2-6 and Table 2-8 and STARTUP_WAIT in Table 2-7.</p> <p>Clarification to Figure 3-5 and changed Equation 3-1. Adding DRP ports to PLL_ADV in Figure 3-6. Updated Table 3-3. Added DRP port descriptions to Table 3-4. Adding attributes to Figure 3-11 and Figure 3-12. Updated Zero Delay Buffer section updating Figure 3-13 and adding Figure 3-14 for single-ended and differential solutions. Clarifying edits to Figure 3-15, Figure 3-16, and Figure 3-17.</p> |
| 02/16/11 | 1.5 | <p>Added BUFGMUX routing restrictions for DCM and PLL programming clock and BUFGMUX ASYNC usage to Clock Buffers and Multiplexers. Updated signal O in Figure 1-23. Updated title of Figure 1-24. Clarified BUFPLL LOCKED routing restrictions in BUFPLL. Updated definitions of PLLIN and LOCK in Table 1-22.</p> <p>Added DIVIDE_BYPASS usage to BUFIO2FB. Updated Phase Shift. Updated DCM function for PS in Table 2-5. Updated STATUS[1] and STATUS[2] ports in Table 2-6. Updated description of PHASE_SHIFT in Table 2-7. Updated descriptions of STATUS[1] and STATUS[7:3] in Table 2-8. Added description of low-power reset circuit to RST Input Behavior, including Figure 2-12 and Table 2-14.</p> <p>Added note to CLKOUT[0:5]_DIVIDE in Table 3-5.</p> |

| Date | Version | Revision |
|----------|---------|---|
| 05/12/11 | 1.6 | <p>Updated description of I_INVERT in Table 1-19. Added BUFIO2 clock inputs to Figure 1-32 and Figure 1-33. Updated description of ENABLE_SYNC in Table 1-23. Added Dynamic Reconfiguration Port.</p> |
| 07/20/12 | 1.7 | <p>Added sentence about BUFH not being recommended for the DCM or PLL feedback paths to BUFH. Corrected spelling of BUFIO2FB in Table 1-16 and Table 1-17. Updated first bullet after Table 1-17. Added paragraph about word synchronization after Figure 1-37. Added Skew Adjustment. Updated first paragraphs of Phase Shift and Variable Phase Shift. Updated phase shift values after Table 2-4. Added note 1 to and updated descriptions of STATUS[0], LOCKED, AND PSDONE in Table 2-6. Updated descriptions of DESKEW_ADJUST and STARTUP_WAIT in Table 2-7. Added note 1 to and updated descriptions of LOCKED and PROGDONE in Table 2-8. Updated descriptions of SPREAD_SPECTRUM and STARTUP_WAIT in Table 2-9. Removed paragraph about using two adjacent DCMs in LOCKED Output Behavior. Added note 1 to Table 2-15. Updated first paragraph of PLL_ADV Primitive. Added note 2 and updated LOCKED pin description in Table 3-4. Removed matches from Figure 3-15 and Figure 3-16.</p> |
| 06/20/13 | 1.8 | <p>In Chapter 1, removed DIVIDE(2) setting for BUFIO2 and BUFIO2_2CLK per XCN13018: Spartan-6 FPGA BUFIO2 and BUFIO2_2CLK Primitives DIVIDE Attribute Allowed Values Change. Clarified BUFIO2 alignment in Spanning a Full Bank with a Single Global Clock Input With Two I/O Clocks. Updated Control column in Table 1-16. Removed DIVIDE_BYPASS from Table 1-21. Added note to Table 1-23 and Table 1-25. Added sentence about general interconnect to description of Clock Input Sources in Table 2-2. Removed general interconnect input into CLKIN multiplexer in Figure 3-1. Updated description of LOCKED pin in Table 3-4.</p> |

Table of Contents

| | |
|------------------------|---|
| Revision History | 2 |
|------------------------|---|

Preface: About This Guide

| | |
|--------------------------------|----|
| Guide Contents | 9 |
| Additional Documentation | 9 |
| Additional Resources | 10 |

Chapter 1: Clock Resources

| | |
|---|----|
| Summary | 11 |
| Introduction | 11 |
| Clock Resources..... | 12 |
| Global Clocking Infrastructure | 13 |
| I/O Clocking Infrastructure | 21 |
| Spanning a Full Bank with a Single Global Clock Input With Two I/O Clocks | 22 |
| Clock Inputs | 23 |
| Clocking Structure Guidelines..... | 30 |
| SDR Data Rate (FD Register in IOB, No IOSERDES2)..... | 30 |
| DDR Data Rate (IDDR2, ODDR2, No IOSERDES2) | 31 |
| High-Speed IOSERDES2 Usage for Advanced Serialization..... | 31 |
| Examples of High-Speed I/O Clock Network Connections | 32 |
| Clock Buffers and Multiplexers | 37 |
| Global Clock Input Buffer Primitives | 37 |
| Global Clock Buffer Primitives | 37 |
| BUFGMUX | 38 |
| BUFGMUX_1 | 40 |
| BUFG | 41 |
| BUFGCE and BUFGCE_1 | 42 |
| BUFH | 43 |
| Clock Buffers for the High-Speed I/O Clock Region | 43 |
| BUFIO2 | 45 |
| BUFIO2_2CLK | 49 |
| BUFPLL | 51 |
| BUFPLL_MCB | 54 |
| BUFIO2FB | 55 |

Chapter 2: Clock Management Technology

| | |
|---|----|
| Clock Management Summary | 57 |
| DCM Summary | 58 |
| DCM Introduction | 59 |
| Compatibility and Comparison with Other Xilinx FPGA Families..... | 60 |
| DCM Functional Overview..... | 61 |
| Delay-Locked Loop | 61 |
| Skew Adjustment..... | 62 |

| | |
|--|----|
| Digital Frequency Synthesizer | 64 |
| Phase Shift | 64 |
| Fixed Phase Shift | 65 |
| Variable Phase Shift | 65 |
| Status Logic | 68 |
| DCM Primitives | 69 |
| DCM_SP Primitive | 69 |
| DCM_CLKGEN Primitive | 74 |
| DCM_SP Design Guidelines | 77 |
| Input Clock Frequency Range | 77 |
| Output Clock Frequency Range | 78 |
| Input Clock and Clock Feedback Variation | 78 |
| Cycle-to-Cycle Jitter | 78 |
| Period Jitter | 78 |
| DLL Feedback Delay Variance | 79 |
| Spread Spectrum Clock Reception | 79 |
| Optimal DCM Clock and External Feedback Inputs | 79 |
| LOCKED Output Behavior | 79 |
| Using the LOCKED Signal | 80 |
| RST Input Behavior | 81 |
| DCM_CLKGEN Design Guidelines | 82 |
| Dynamic Frequency Synthesis | 83 |
| Spread-Spectrum Clock Generation | 85 |
| Spread-Spectrum Generation | 86 |
| Fixed Spread Spectrum | 86 |
| Soft Spread Spectrum | 87 |
| Free-Running Oscillator | 88 |

Chapter 3: Phase-Locked Loops

| | |
|--|-----|
| Introduction | 91 |
| Phase Lock Loop (PLL) | 94 |
| Aligning PLL using CLK_FEEDBACK and BUFIO2FB | 96 |
| General Usage Description | 98 |
| PLL Primitives | 98 |
| PLL_BASE Primitive | 98 |
| PLL_ADV Primitive | 99 |
| Clock Network Deskew | 99 |
| Frequency Synthesis Only | 100 |
| Jitter Filter | 100 |
| Limitations | 101 |
| VCO Operating Range | 101 |
| Minimum and Maximum Input Frequency | 101 |
| Duty Cycle Programmability | 101 |
| Phase Shift | 101 |
| PLL Programming | 101 |
| Determine the Input Frequency | 102 |
| Determine the M and D Values | 102 |
| PLL Ports | 103 |
| PLL Attributes | 104 |
| PLL Clock Input Signals | 106 |
| Counter Control | 106 |
| Clock Shifting | 107 |

| | |
|---|-----|
| Detailed VCO and Output Counter Waveforms | 108 |
| Missing Input Clock or Feedback Clock | 108 |
| PLL Use Models | 109 |
| Clock Network Deskew | 109 |
| PLL with Internal Feedback | 110 |
| Zero Delay Buffer | 110 |
| Differential BUFIO2FB Zero Delay Buffer Example (Verilog) | 112 |
| Differential BUFIO2FB Zero Delay Buffer Example (VHDL) | 112 |
| DCM Driving PLL | 112 |
| PLL Driving DCM | 113 |
| PLL to PLL Connection | 114 |
| Dynamic Reconfiguration Port | 115 |
| Application Guidelines | 115 |
| PLL Application Example | 115 |

About This Guide

This document describes Spartan®-6 FPGA clocking. Complete and up-to-date documentation of the Spartan-6 family of FPGAs is available on the Xilinx website at <http://www.xilinx.com/support/documentation/spartan-6.htm>.

Guide Contents

This manual contains the following chapters:

- Chapter 1, Clock Resources
- Chapter 2, Clock Management Technology
- Chapter 3, Phase-Locked Loops

Additional Documentation

The following documents are also available for download at

<http://www.xilinx.com/support/documentation/spartan-6.htm>.

- Spartan-6 Family Overview
This overview outlines the features and product selection of the Spartan-6 family.
- Spartan-6 FPGA Data Sheet: DC and Switching Characteristics
This data sheet contains the DC and switching characteristic specifications for the Spartan-6 family.
- Spartan-6 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.
- Spartan-6 FPGA Configuration User Guide
This all-encompassing configuration guide includes chapters on configuration interfaces (serial and parallel), multi-bitstream management, bitstream encryption, boundary-scan and JTAG configuration, and reconfiguration techniques.
- Spartan-6 FPGA SelectIO Resources User Guide
This guide describes the SelectIO™ resources available in all Spartan-6 devices.
- Spartan-6 FPGA Block RAM Resources User Guide
This guide describes the Spartan-6 device block RAM capabilities.
- Spartan-6 FPGA Configurable Logic Block User Guide

This guide describes the capabilities of the configurable logic blocks (CLBs) available in all Spartan-6 devices.

- Spartan-6 FPGA DSP48A1 Slice User Guide

This guide describes the architecture of the DSP48A1 slice in Spartan-6 FPGAs and provides configuration examples.

- Spartan-6 FPGA GTP Transceiver User Guide

This guide describes the GTP transceivers available in the Spartan-6 LXT FPGAs.

- Spartan-6 FPGA Memory Controller User Guide

This guide describes the Spartan-6 FPGA memory controller block, a dedicated, embedded multi-port memory controller that greatly simplifies interfacing Spartan-6 FPGAs to the most popular memory standards.

- Spartan-6 FPGA PCB Design Guide

This guide provides information on PCB design for Spartan-6 devices, with a focus on strategies for making design decisions at the PCB and interface level.

Additional Resources

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

<http://www.xilinx.com/support>

Clock Resources

Summary

This chapter describes how to take advantage of the Spartan-6 FPGA clock resources, including the dedicated clock inputs, buffers, and routing. The clocking infrastructure provides a series of low-capacitance, low-skew interconnect lines that are well suited to carrying high-frequency signals throughout the FPGA, minimizing clock skew and improving performance, and should be used for all clock signals. Third-party synthesis tools, and Xilinx synthesis and implementation tools, automatically use some of these resources for high fan-out clock signals.

The clock routing can be used in conjunction with the DCMs and PLLs, which are discussed in more detail in [Chapter 2, Clock Management Technology](#) and [Chapter 3, Phase-Locked Loops](#).

Introduction

Each Spartan-6 FPGA device offers 16 high-speed, low-skew global clock resources to optimize performance. These resources are used automatically by the Xilinx tools. Even if the clock rate is relatively slow, it is still important to use the global routing resources to eliminate any potential for timing hazards. It is important to understand how to define and take the best advantage of these resources.

Each Spartan-6 FPGA also provides 40 ultra high-speed, low-skew I/O regional clock resources (32 BUFIO2s and eight BUFPLLs) to serve localized I/O serializer/de-serializer (ISERDES and OSERDES) circuits. For more information on ISERDES and OSERDES, go to the [UG381, Spartan-6 FPGA SelectIO Resources User Guide](#) and [XAPP1064, Source-Synchronous Serialization and Deserialization \(up to 1050 Mb/s\)](#).

Use the ISE® software to check all the design rules and to ensure correct usage of clock resources, SelectIO logic, I/O standard compatibility, and routability. A completed design ensures that all placement and logic restrictions are properly checked.

A design checklist is provided in [UG393, Spartan-6 FPGA PCB Design Guide](#) to assist with pin planning. [UG385, Spartan-6 FPGA Packaging and Pinout Specification](#).

Clock Resources

The Spartan-6 FPGA clock resources consist of four types of connections:

- Global clock input pads (GCLK)
- Global clock multiplexers (BUFG, BUFGMUX)
- I/O clock buffers (BUFIO2, BUFIO2_2CLK, BUFPLL)
- Horizontal clock routing buffers (BUFH)

There are two types of clock networks:

- Global clock network providing low-skew clock routing to the FPGA logic resources
- I/O regional clock networks providing high-performance low-skew clocking to the SelectIO logic resources

BUFGMUX can multiplex between two global clock sources or be used as a simple BUFG clock buffer. The clock buffer can only directly drive the global clock routing resources, which can only drive clock inputs. However, clock inputs on the FPGA logic flip-flops can also come from general-purpose routing, although their use should be limited due to higher skew.

BUFPLL and BUFIO2 are used to drive clocks routed only on the I/O regional clock network with much higher performance than the global clock network. This limits the driving destination only to the input serial-to-parallel logic resources (ISERDES) or output parallel-to-serial logic resources (OSERDES) on each bank of the FPGA.

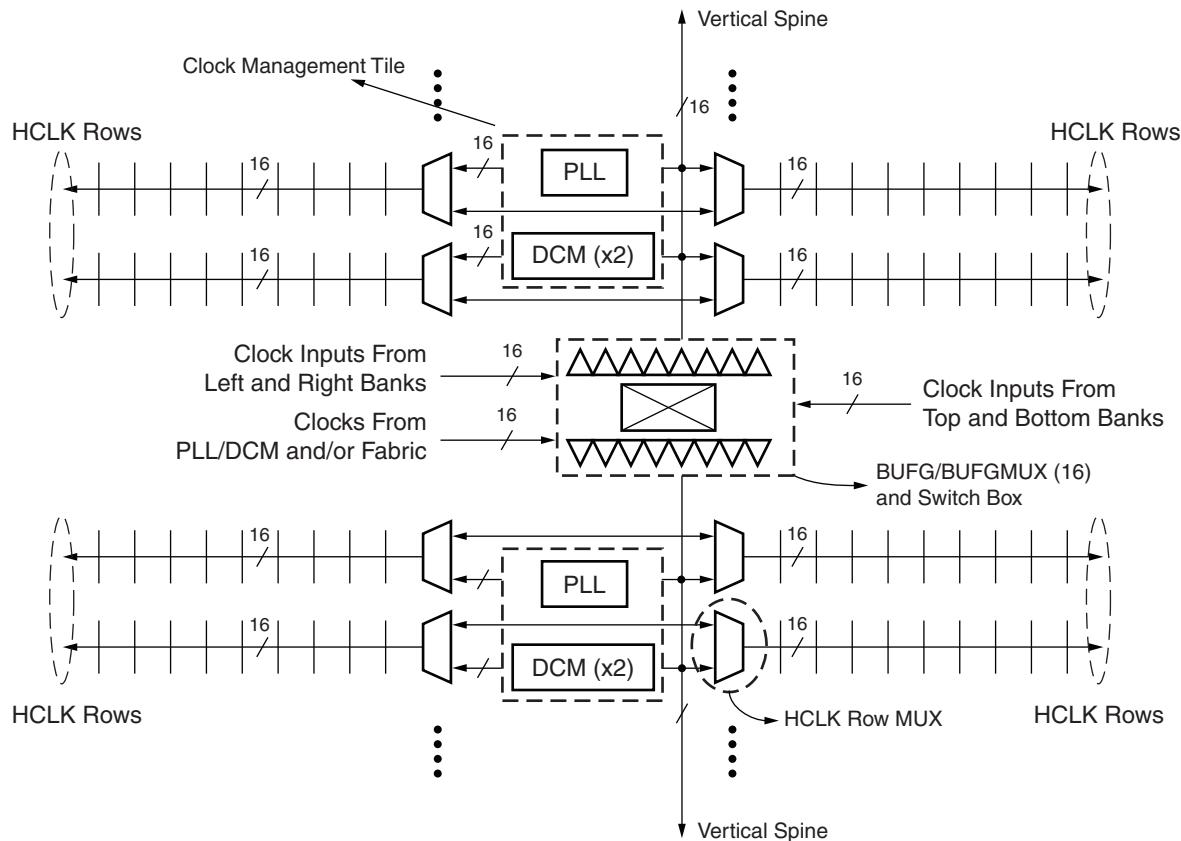
BUFIO2 can drive ISERDES2 and OSERDES2 for either SDR or DDR clocking. BUFIO2 can also route clock inputs from either a GCLK or a GTP_DUAL tile to a BUFG, DCM, or PLL clock input. BUFIO2_2CLK can be used to replace one of the BUFIO2s required for DDR clocking of the ISERDES2 and OSERDES2.

Similarly, the BUFPLL drives clocks routed to the I/O clock network for SDR clocking. The BUFPLL uses a direct connection from the PLL (CLKOUT0 or CLKOUT1) to drive the I/O regional clock network.

BUFH increases the total number of low-skew clock resources available by providing direct access to horizontal sections of the global clock routing.

Global Clocking Infrastructure

The detailed Spartan-6 FPGA global clocking infrastructure is shown in [Figure 1-1](#).

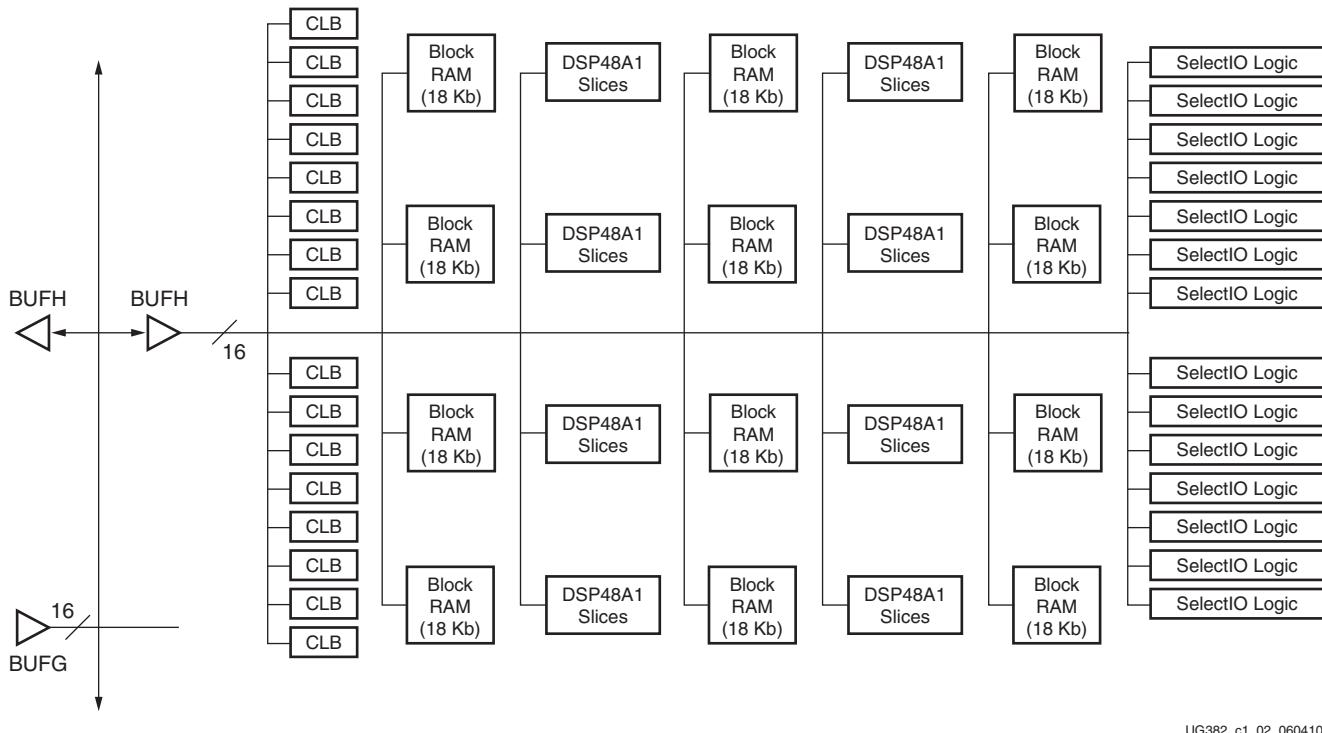


UG382_c1_01_081009

Figure 1-1: Spartan-6 FPGA Global Clock Structure

The global clock network in Spartan-6 FPGAs is driven by 16 BUFGMUXes located in the center of the device. The 16 BUFGMUXes can be fed from three different sources; clock inputs from top and bottom banks, clock inputs from left and right banks, and clocks from FPGA logic interconnect and/or PLL/DCM. These three clock sources are multiplexed using a switch box also located in the center of the device.

The 16 BUFGMUXes then drive a vertical spine to travel north and south. Along the way it horizontally spans toward HCLK row clocks used to provide clock access to regional logic primitives. Each HCLK row has 16 horizontal clock buffers (BUFHs) driving left and 16 BUFHs driving right as shown in [Figure 1-2](#).



UG382_c1_02_060410

Figure 1-2: BUFH Routing

The HCLK row is entered through a dedicated multiplexer switching clocks between the vertical spine and the PLL or DCM outputs. Each HCLK row hosts either one PLL or two DCMs. The PLL or DCM clock outputs can optionally drive a BUFH within the same HCLK row.

Since there are up to 32 GCLK input pins and only 16 global clock buffers in Spartan-6 devices, every global clock buffer can be driven by one of two GCLK pins. When driving a global clock buffer (BUFG or BUFGMUX) directly with a global clock pin (IBUFG or IBUFGDS), the global clock pins from banks 0, 1, and 5 share the same eight global clock buffers as shown in [Table 1-1](#). Similarly, banks 2, 3, and 4 share eight global clock buffers ([Table 1-2](#)).

To illustrate the routing conflict caused by the sharing of BUFGMUX inputs, consider a design using GCLK19 and GCLK11. As [Table 1-1](#) shows, both of the global clocks are connected to BUFGMUX_X2Y1 causing a routing error to be generated.

For more routing flexibility, the BUFIO2 can additionally be used to route to a second global clock buffer when using a BUFIO2 ([Table 1-1](#)). When using a BUFIO2, a nominal delay through the BUFIO2 will be incurred. Using the BUFIO2 can also impact clock routing to the I/O clock network. Additional routing information is shown in [Spanning a Full Bank with a Single Global Clock Input With Two I/O Clocks, page 22](#).

When using a differential global clock, the global clock associated with the master side of the differential pair (P) will determine the global clock resource used.

Table 1-1: Shared Global Clocking Resources for Bank 0 and Bank 1

| BUFGMUX Routing Restrictions | | Bank 0 | Bank 1 |
|--|-----------------|------------------------|------------------------|
| BUFGMUX_X2Y1 (I0) BUFGMUX_X2Y2 (I1) | Direct Routing | GCLK_19 | GCLK_11 |
| | Indirect BUFIO2 | GCLK_19 <BUFIO2_X2Y28> | GCLK_11 <BUFIO2_X4Y20> |
| | Indirect BUFIO2 | GCLK_15 <BUFIO2_X2Y28> | GCLK_7 <BUFIO2_X4Y20> |
| BUFGMUX_X2Y2 (I0) BUFGMUX_X2Y1 (I1) | Direct Routing | GCLK_18 | GCLK_10 |
| | Indirect BUFIO2 | GCLK_18 <BUFIO2_X2Y29> | GCLK_10 <BUFIO2_X4Y21> |
| | Indirect BUFIO2 | GCLK_14 <BUFIO2_X2Y29> | GCLK_6 <BUFIO2_X4Y21> |
| BUFGMUX_X2Y3 (I0) BUFGMUX_X2Y4 (I1) | Direct Routing | GCLK_17 | GCLK_9 |
| | Indirect BUFIO2 | GCLK_17 <BUFIO2_X2Y26> | GCLK_9 <BUFIO2_X4Y18> |
| | Indirect BUFIO2 | GCLK_13 <BUFIO2_X2Y26> | GCLK_5 <BUFIO2_X4Y18> |
| BUFGMUX_X2Y4 (I0) BUFGMUX_X2Y3 (I1) | Direct Routing | GCLK_15 | GCLK_7 |
| | Indirect BUFIO2 | GCLK_15 <BUFIO2_X4Y28> | GCLK_7 <BUFIO2_X3Y12> |
| | Indirect BUFIO2 | GCLK_19 <BUFIO2_X4Y28> | GCLK_11 <BUFIO2_X3Y12> |
| BUFGMUX_X3Y5 (I0) BUFGMUX_X3Y6 (I1) | Direct Routing | GCLK_16 | GCLK_8 |
| | Indirect BUFIO2 | GCLK_16 <BUFIO2_X2Y27> | GCLK_8 <BUFIO2_X4Y19> |
| | Indirect BUFIO2 | GCLK_12 <BUFIO2_X2Y27> | GCLK_4 <BUFIO2_X4Y19> |
| BUFGMUX_X3Y6 (I0) BUFGMUX_X3Y5 (I1) | Direct Routing | GCLK_14 | GCLK_6 |
| | Indirect BUFIO2 | GCLK_14 <BUFIO2_X4Y29> | GCLK_6 <BUFIO2_X3Y13> |
| | Indirect BUFIO2 | GCLK_18 <BUFIO2_X4Y29> | GCLK_10 <BUFIO2_X3Y13> |
| BUFGMUX_X3Y7 (I0) BUFGMUX_X3Y8 (I1) | Direct Routing | GCLK_13 | GCLK_5 |
| | Indirect BUFIO2 | GCLK_13 <BUFIO2_X4Y26> | GCLK_5 <BUFIO2_X3Y10> |
| | Indirect BUFIO2 | GCLK_17 <BUFIO2_X4Y26> | GCLK_9 <BUFIO2_X3Y10> |
| BUFGMUX_X3Y8 (I0) BUFGMUX_X3Y7 (I1) | Direct Routing | GCLK_12 | GCLK_4 |
| | Indirect BUFIO2 | GCLK_12 <BUFIO2_X4Y27> | GCLK_4 <BUFIO2_X3Y11> |
| | Indirect BUFIO2 | GCLK_16 <BUFIO2_X4Y27> | GCLK_8 <BUFIO2_X3Y11> |

Table 1-2: Shared Global Clocking Resources for Bank 2 and Bank 3

| BUFGMUX Routing Restrictions | | Bank 2 | Bank 3 |
|---|-----------------|-----------------------|-----------------------|
| BUFGMUX_X2Y9 (I0) BUFGMUX_X2Y10 (I1) | Direct Routing | GCLK_3 | GCLK_27 |
| | Indirect BUFIO2 | GCLK_3 <BUFIO2_X3Y0> | GCLK_27 <BUFIO2_X1Y8> |
| | Indirect BUFIO2 | GCLK_31 <BUFIO2_X3Y0> | GCLK_23 <BUFIO2_X1Y8> |
| BUFGMUX_X2Y10 (I0) BUFGMUX_X2Y9 (I1) | Direct Routing | GCLK_2 | GCLK_26 |
| | Indirect BUFIO2 | GCLK_2 <BUFIO2_X3Y1> | GCLK_26 <BUFIO2_X1Y9> |
| | Indirect BUFIO2 | GCLK_30 <BUFIO2_X3Y1> | GCLK_22 <BUFIO2_X1Y9> |

Table 1-2: Shared Global Clocking Resources for Bank 2 and Bank 3 (Cont'd)

| BUFGMUX Routing Restrictions | | Bank 2 | Bank 3 |
|--|-----------------|-----------------------|------------------------|
| BUFGMUX_X2Y11 (I0) BUFGMUX_X2Y12 (I1) | Direct Routing | GCLK_1 | GCLK_25 |
| | Indirect BUFIO2 | GCLK_1 <BUFIO2_X3Y6> | GCLK_25 <BUFIO2_X1Y14> |
| | Indirect BUFIO2 | GCLK_29 <BUFIO2_X3Y6> | GCLK_21 <BUFIO2_X1Y14> |
| BUFGMUX_X2Y12 (I0) BUFGMUX_X2Y11 (I1) | Direct Routing | GCLK_31 | GCLK_23 |
| | Indirect BUFIO2 | GCLK_31 <BUFIO2_X1Y0> | GCLK_23 <BUFIO2_X0Y16> |
| | Indirect BUFIO2 | GCLK_3 <BUFIO2_X1Y0> | GCLK_27 <BUFIO2_X0Y16> |
| BUFGMUX_X3Y13 (I0) BUFGMUX_X3Y14 (I1) | Direct Routing | GCLK_0 | GCLK_24 |
| | Indirect BUFIO2 | GCLK_0 <BUFIO2_X3Y7> | GCLK_24 <BUFIO2_X1Y15> |
| | Indirect BUFIO2 | GCLK_28 <BUFIO2_X3Y7> | GCLK_20 <BUFIO2_X1Y15> |
| BUFGMUX_X3Y14 (I0) BUFGMUX_X3Y13 (I1) | Direct Routing | GCLK_30 | GCLK_22 |
| | Indirect BUFIO2 | GCLK_30 <BUFIO2_X1Y1> | GCLK_22 <BUFIO2_X0Y17> |
| | Indirect BUFIO2 | GCLK_2 <BUFIO2_X1Y1> | GCLK_26 <BUFIO2_X0Y17> |
| BUFGMUX_X3Y15 (I0) BUFGMUX_X3Y16 (I1) | Direct Routing | GCLK_29 | GCLK_21 |
| | Indirect BUFIO2 | GCLK_29 <BUFIO2_X1Y6> | GCLK_21 <BUFIO2_X0Y22> |
| | Indirect BUFIO2 | GCLK_1 <BUFIO2_X1Y6> | GCLK_25 <BUFIO2_X0Y22> |
| BUFGMUX_X3Y16 (I0) BUFGMUX_X3Y15 (I1) | Direct Routing | GCLK_28 | GCLK_20 |
| | Indirect BUFIO2 | GCLK_28 <BUFIO2_X1Y7> | GCLK_20 <BUFIO2_X0Y23> |
| | Indirect BUFIO2 | GCLK_0 <BUFIO2_X1Y7> | GCLK_24 <BUFIO2_X0Y23> |

A graphical representation of the conflicting BUFGMUX inputs is shown in [Figure 1-3](#) and [Figure 1-4](#).

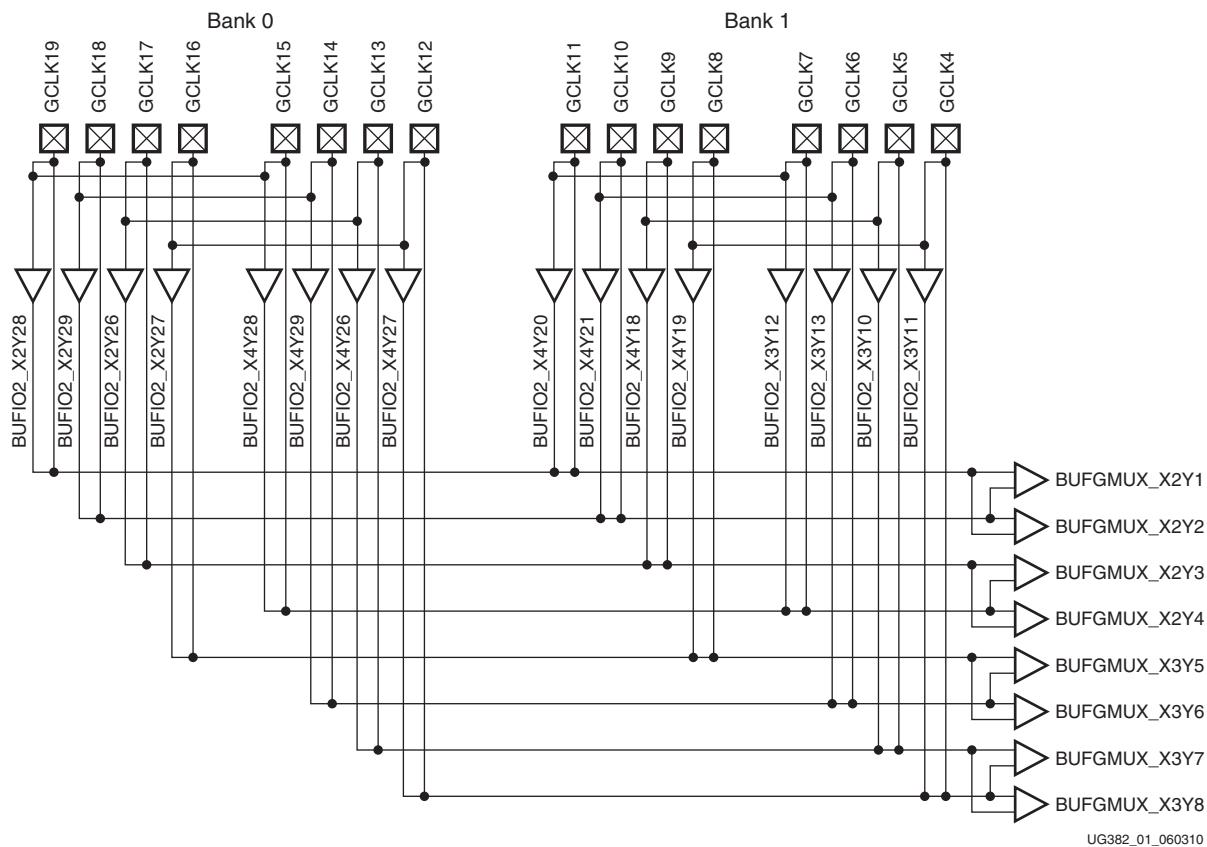


Figure 1-3: BUFGMUX Connections for Bank 0 and Bank 1

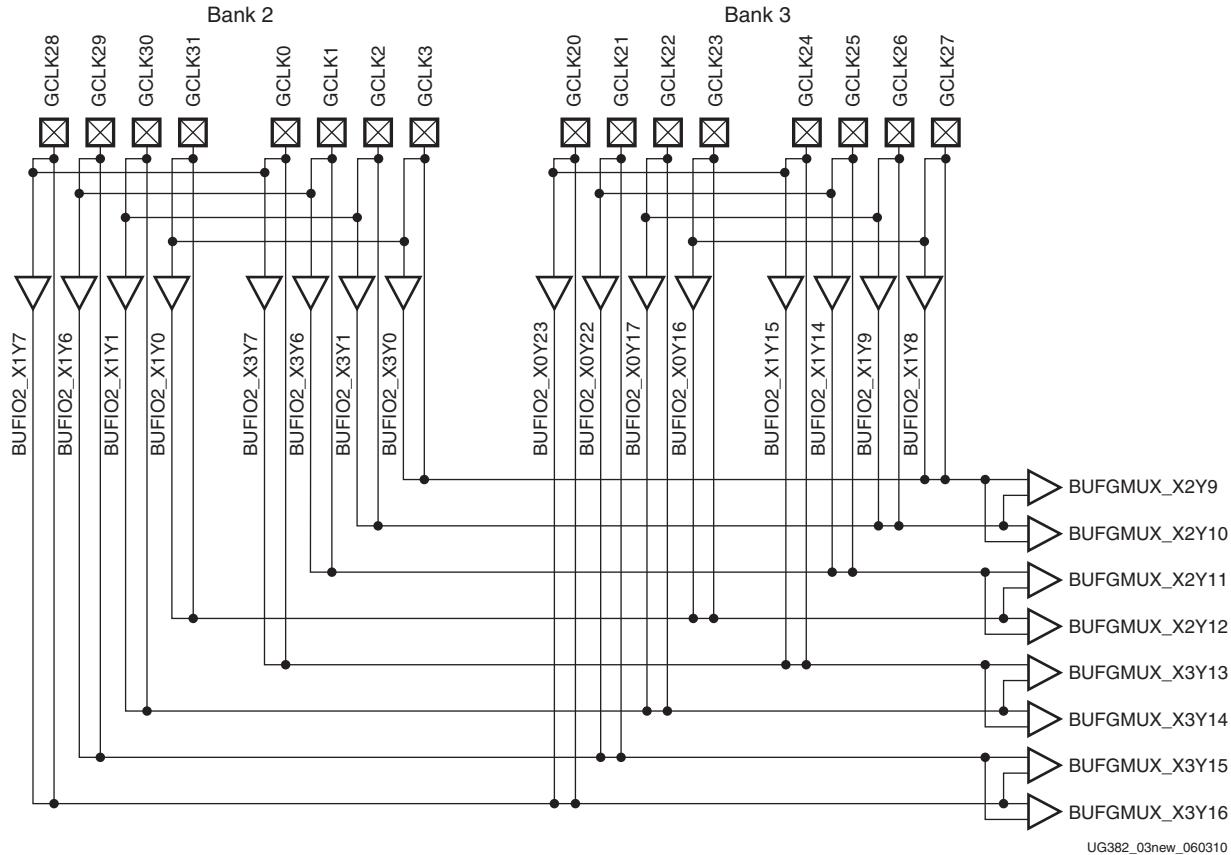


Figure 1-4: BUFGMUX Connections for Bank 2 and Bank 3

For designs using GTP transceivers, each GTP reference clock is associated with a BUFI02. This can affect the global clock pins located in bank 0 or bank 2. For SDR interfaces, [Table 1-3](#) lists the GCLK inputs. For DDR interfaces, a second BUFI02 can be required to invert the clock using the inverting BUFI02 placements as shown in [Table 1-4](#).

For a complete listing GTP_DUAL placement please see the placement diagrams in [UG386, Spartan-6 FPGA GTP Transceivers User Guide](#).

Table 1-3: BUFI02 Input Conflicts for SDR Data Rates (ISERDES2 (SDR), OSERDES2 (SDR))

| Bank | BUFI02 | GCLK Inputs | | GTP Reference Clock for Example Device ⁽¹⁾ | Shared GTPCLKOUT | BUFI02 Clocking Region |
|--------|--------------|-------------|--------|---|------------------|------------------------|
| Bank 0 | BUFI02_X2Y26 | GCLK17 | GCLK13 | GTPA1_DUAL_X0Y1 | GTPCLKOUT1[0] | TL |
| | BUFI02_X2Y27 | GCLK16 | GCLK12 | | GTPCLKOUT1[1] | TL |
| | BUFI02_X2Y28 | GCLK19 | GCLK15 | | GTPCLKOUT0[0] | TL |
| | BUFI02_X2Y29 | GCLK18 | GCLK14 | | GTPCLKOUT0[1] | TL |
| | BUFI02_X4Y26 | GCLK13 | GCLK17 | GTPA1_DUAL_X1Y1 | GTPCLKOUT1[0] | TR |
| | BUFI02_X4Y27 | GCLK12 | GCLK16 | | GTPCLKOUT1[1] | TR |
| | BUFI02_X4Y28 | GCLK15 | GCLK19 | | GTPCLKOUT0[0] | TR |
| | BUFI02_X4Y29 | GCLK14 | GCLK18 | | GTPCLKOUT0[1] | TR |
| Bank 2 | BUFI02_X1Y0 | GCLK31 | GCLK3 | GTPA1_DUAL_X0Y0 | GTPCLKOUT0[0] | BL |
| | BUFI02_X1Y1 | GCLK30 | GCLK2 | | GTPCLKOUT0[1] | BL |
| | BUFI02_X1Y6 | GCLK29 | GCLK1 | | GTPCLKOUT1[0] | BL |
| | BUFI02_X1Y7 | GCLK28 | GCLK0 | | GTPCLKOUT1[1] | BL |
| | BUFI02_X3Y0 | GCLK3 | GCLK31 | GTPA1_DUAL_X1Y0 | GTPCLKOUT0[0] | BR |
| | BUFI02_X3Y1 | GCLK2 | GCLK30 | | GTPCLKOUT0[1] | BR |
| | BUFI02_X3Y6 | GCLK1 | GCLK29 | | GTPCLKOUT1[0] | BR |
| | BUFI02_X3Y7 | GCLK0 | GCLK28 | | GTPCLKOUT1[1] | BR |

Notes:

1. Example using an LX100T/LX150T in an FG(G)900 package.

Table 1-4: BUFI02 Input Conflicts for DDR Data Rates (IDDR2, ODDR2, ISERDES2 (DDR), OSERDES2 (DDR))

| Bank | BUFI02 | GCLK Inputs (Inverting) | | GTP Reference Clock for Example Device ⁽¹⁾ | Shared GTPCLKOUT | BUFI02 Clocking Region |
|--------|-----------------------------------|----------------------------|--------|--|------------------|---------------------------|
| Bank 0 | BUFI02_X2Y26 (I_INVERT = TRUE) | GCLK16 | GCLK12 | GTPA1_DUAL_X0Y1 | GTPCLKOUT1[0] | TL |
| | BUFI02_X2Y27 (I_INVERT = TRUE) | GCLK17 | GCLK13 | | GTPCLKOUT1[1] | TL |
| | BUFI02_X2Y28 (I_INVERT = TRUE) | GCLK18 | GCLK14 | | GTPCLKOUT0[0] | TL |
| | BUFI02_X2Y29 (I_INVERT = TRUE) | GCLK19 | GCLK15 | | GTPCLKOUT0[1] | TL |
| | BUFI02_X4Y26 (I_INVERT = TRUE) | GCLK12 | GCLK16 | GTPA1_DUAL_X1Y1 | GTPCLKOUT1[0] | TR |
| | BUFI02_X4Y27 (I_INVERT = TRUE) | GCLK13 | GCLK17 | | GTPCLKOUT1[1] | TR |
| | BUFI02_X4Y28 (I_INVERT = TRUE) | GCLK14 | GCLK18 | | GTPCLKOUT0[0] | TR |
| | BUFI02_X4Y29 (I_INVERT = TRUE) | GCLK15 | GCLK19 | | GTPCLKOUT0[1] | TR |
| Bank 2 | BUFI02_X1Y0 (I_INVERT = TRUE) | GCLK30 | GCLK2 | GTPA1_DUAL_X0Y0 | GTPCLKOUT0[0] | BL |
| | BUFI02_X1Y1 (I_INVERT = TRUE) | GCLK31 | GCLK3 | | GTPCLKOUT0[1] | BL |
| | BUFI02_X1Y6 (I_INVERT = TRUE) | GCLK28 | GCLK0 | | GTPCLKOUT1[0] | BL |
| | BUFI02_X1Y7 (I_INVERT = TRUE) | GCLK29 | GCLK1 | | GTPCLKOUT1[1] | BL |
| | BUFI02_X3Y0 (I_INVERT = TRUE) | GCLK2 | GCLK30 | GTPA1_DUAL_X1Y0 | GTPCLKOUT0[0] | BR |
| | BUFI02_X3Y1 (I_INVERT = TRUE) | GCLK3 | GCLK31 | | GTPCLKOUT0[1] | BR |
| | BUFI02_X3Y6 (I_INVERT = TRUE) | GCLK0 | GCLK28 | | GTPCLKOUT1[0] | BR |
| | BUFI02_X3Y7 (I_INVERT = TRUE) | GCLK1 | GCLK29 | | GTPCLKOUT1[1] | BR |

Notes:

1. Example using an LX100T/LX150T in an FG(G)900 package.

I/O Clocking Infrastructure

[Figure 1-5](#) illustrates the I/O clocking infrastructure.

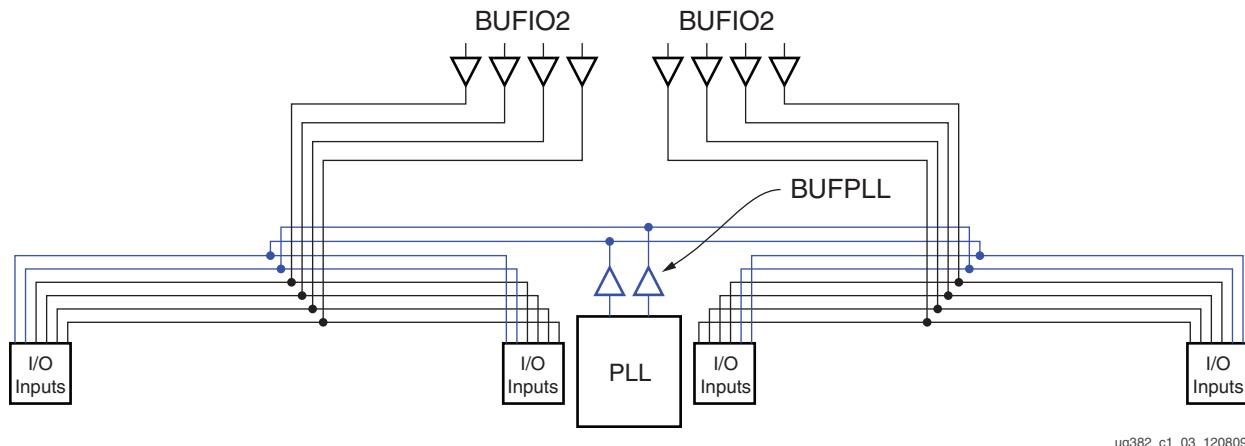


Figure 1-5: Spartan-6 FPGA I/O Clock Structure in an I/O Bank

All SelectIO logic resources (input registers, output registers, IDDR2, ODDR2, ISERDES2, and OSERDES2) must be driven by a clock coming from either a BUFIO2 located within the same BUFIO2 clocking region, one of the BUFP LLs on the same edge of the device, or one of the 16 BUFGs.

It is important to understand the organization of the BUFIO2 clocking regions for designs using the BUFIO2 for clocking SelectIO logic. There are four high-speed I/O clocks in every BUFIO2 clocking region driven by four dedicated BUFIO2 buffers. There are a total of eight BUFIO2 clocking regions for a total of 32 BUFIO2s. As shown in [Figure 1-6](#), each side of the device is split into two BUFIO2 clocking regions. [UG385, Spartan-6 FPGA Packaging and Pinout Specification](#) lists the BUFIO2 clocking region for each pin.

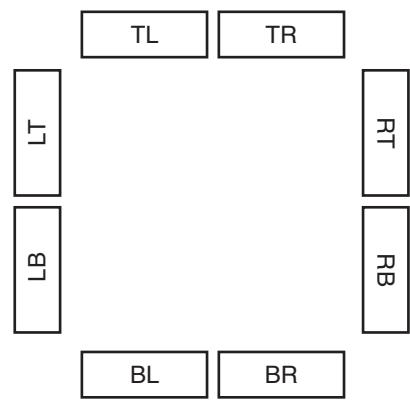


Figure 1-6: BUFIO2 Clocking Regions

Spanning a Full Bank with a Single Global Clock Input With Two I/O Clocks

It is possible to span an entire bank with a single global clock input that is connected to two BUFIO2 buffers in different BUFIO2 clocking regions to span an entire edge of the device. [Figure 1-7](#) illustrates the connection in Bank 0. Two BUFIO2 resources are used as an example for GCLK19, BUFIO2_X2Y28 and BUFIO2_X4Y28. In [Figure 1-7](#), the dashed lines denote I_INVERT paths. Spanning the entire bank by using two BUFIO2 primitives can only be done when the clock input is directly connected to the BUFIO2 primitive. As a result, the BUFIO2 will be aligned to the rising edge of the clock input. See [BUFIO2](#) clock buffer for additional information. For some applications, an IODELAY2 could be needed to delay the input clock. Because the IODELAY2 can only connect to a single BUFIO2, routing of the delayed GCLK input will be restricted to one BUFIO2 clocking region.

Alternatively, to drive the entire bank when using the IODELAY2 primitive, use the PLL with the BUFPPLL primitive.

The I/O clock network can also be driven by a PLL through the BUFPPLL buffers. Each PLL has two associated buffers that each span an entire I/O bank.

Note: Using IODELAY2 clocking with a full bank is not supported.

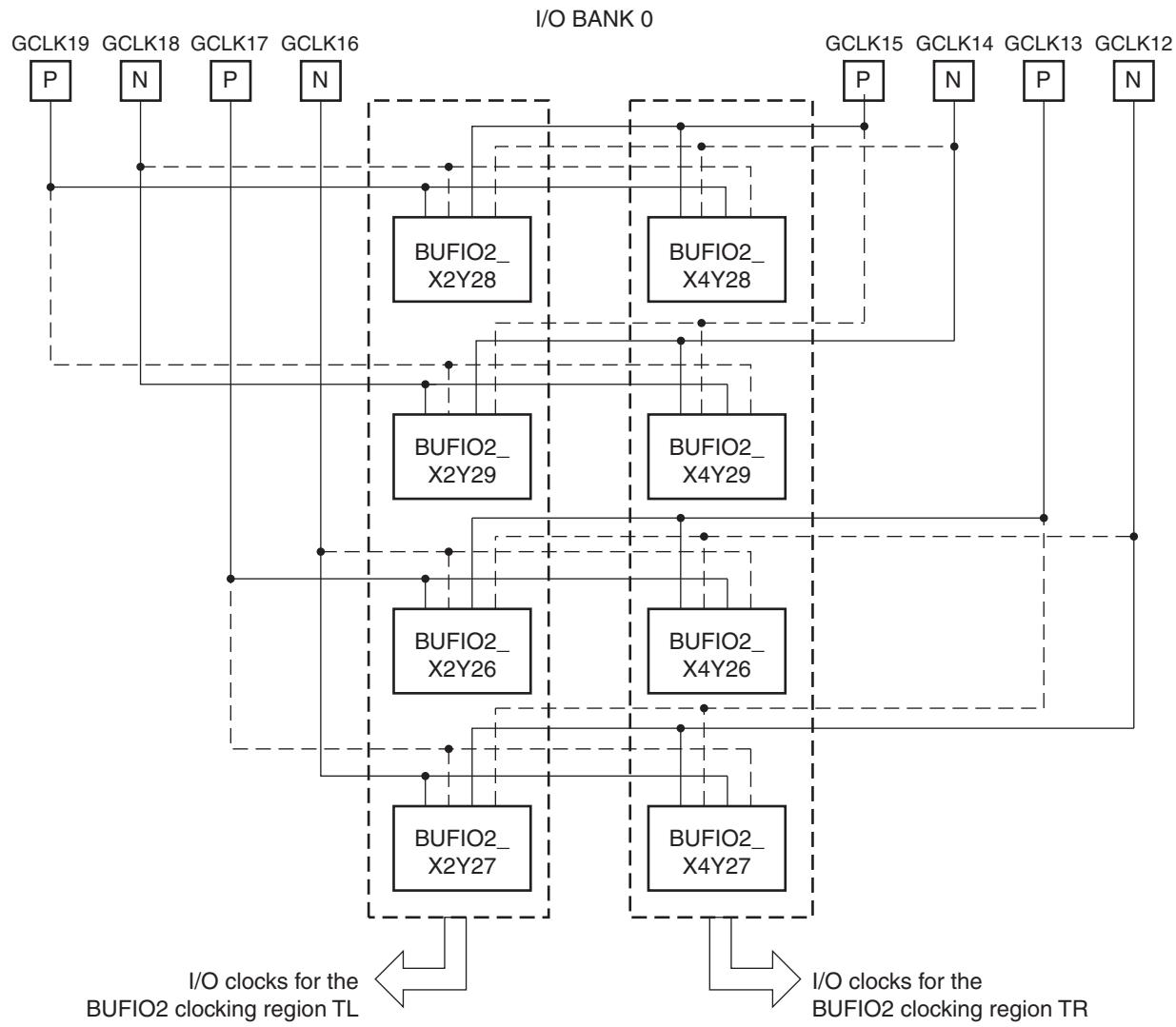


Figure 1-7: I/O Clock Spanning a Full Bank

Clock Inputs

Clock pins accept external clock signals and connect directly to BUFGMUX or BUFIO2 primitives. Clock pins can also be used as general-purpose I/Os. In addition to routing a clock from its input pin onto the I/O clock network, the BUFIO2 also provides a dedicated clock path to PLLs/DCMs and BUFGs. Figure 1-8 illustrates the dedicated clock routes.

Note: Using IODELAY2 clocking with a full bank is not supported.

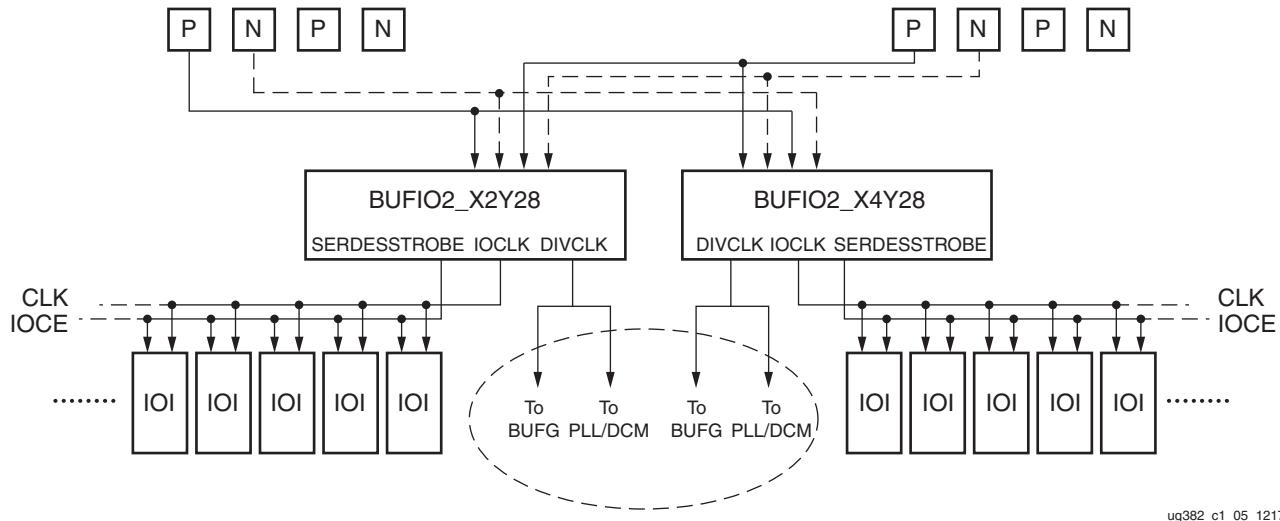


Figure 1-8: Dedicated Clock Inputs Routed by BUFI02

Each Spartan-6 FPGA has:

- Up to 32 global clock inputs located along four edges of the FPGA.
- Eight dedicated clock inputs in the middle of each edge of the device.
- Eight BUFI02 clocking regions

The 32 GCLK input pins are used to drive clock buffers. A differential clock input requires two global clock inputs which allows up to 16 differential global clock inputs. The P and N inputs follow the same configuration as the standard inputs on the clock input pins. The clock inputs that are paired together are consecutive pins in clock number, an even clock number and the next greater odd value. For example, GCLK0 and GCLK1 are a differential pair as are GCLK20 and GCLK21.

All clock input pins can be represented in a design by the IBUFG primitive (or the IBUFGDS primitive for differential clocks). In general, an IBUFG is inferred by the synthesis tool on any top-level clock port. When more control is necessary, an IBUFG can be instantiated by connecting the I port directly to the top-level port and the O port to a DCM, BUFG, or interconnect logic. Most synthesis tools infer the BUFG automatically when connecting an IBUFG to the clock resources of the FPGA.

The IBUFG and IBUFGDS primitives in [Table 1-5](#) are different configurations of the clock input buffer. These two primitives work in conjunction with the Spartan-6 FPGA I/O resource by setting the IOSTANDARD attribute to the desired standard. Refer to *Spartan-6 FPGA SelectIO Resources User Guide* for a complete list of possible I/O standards.

Table 1-5: 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 |

[Table 1-6](#) lists the global clock pin locations where P is the positive and N is the negative side of the differential pairs.

Table 1-6: Global Clock Pin Locations

| GCLK | P/N | TQG144 | CPG196 | CSG225 | FT(G)256 | CSG324 | | CSG484 | | FG(G)484 | | FG(G)676 | | FG(G)900 | |
|--------|-----|--------|--------|--------|----------|--------|-----|--------|------|----------|------|----------|------|----------|------|
| | | | | | | LX | LXT | LX | LXT | LX | LXT | LX | LXT | LX | LXT |
| GCLK0 | N | P55 | P8 | N7 | T8 | V10 | V10 | AB12 | AB12 | AB13 | AB13 | AF13 | AC14 | AK18 | AG16 |
| GCLK1 | P | P56 | N8 | M8 | P8 | U10 | U10 | AA12 | AA12 | Y13 | Y13 | AE13 | AB14 | AJ18 | AF16 |
| GCLK2 | N | none | none | R8 | N8 | T10 | T10 | Y10 | Y10 | Y12 | U12 | AF14 | AF15 | AK19 | AD16 |
| GCLK3 | P | none | none | N8 | M9 | R10 | R10 | W11 | W11 | W12 | T12 | AD14 | AE15 | AH19 | AC16 |
| GCLK4 | N | P84 | H12 | J15 | J16 | H18 | H18 | L22 | L22 | J22 | L22 | U26 | U26 | W30 | W30 |
| GCLK5 | P | P85 | H11 | J14 | J14 | H17 | H17 | L20 | L20 | J20 | L20 | U25 | U25 | W29 | W29 |
| GCLK6 | N | P87 | H14 | H15 | K11 | L16 | L16 | K20 | K20 | L19 | N19 | W24 | W24 | AB30 | AB30 |
| GCLK7 | P | P88 | H13 | H13 | K12 | L15 | L15 | L19 | L19 | M20 | P20 | V23 | V23 | AB28 | AB28 |
| GCLK8 | N | P92 | F14 | G15 | K14 | K16 | K16 | M19 | M19 | H22 | K22 | P22 | R26 | W28 | W28 |
| GCLK9 | P | P93 | F13 | G14 | J13 | K15 | K15 | M18 | M18 | H21 | K21 | P21 | R25 | W27 | W27 |
| GCLK10 | N | P94 | G14 | L12 | J12 | L13 | L13 | K17 | L17 | K19 | M19 | M21 | U24 | V27 | V27 |
| GCLK11 | P | P95 | G13 | K12 | J11 | L12 | L12 | L17 | M17 | K20 | M20 | N20 | U23 | V26 | V26 |
| GCLK12 | N | P123 | A8 | A9 | C10 | A10 | E12 | C12 | F12 | A12 | F16 | A15 | A14 | A18 | D16 |
| GCLK13 | P | P124 | B8 | B9 | E10 | C10 | F12 | D11 | G12 | B12 | E16 | C15 | B14 | C18 | E16 |
| GCLK14 | N | P126 | C8 | A8 | E8 | C11 | G11 | A12 | G11 | C12 | F15 | C14 | A12 | A16 | A16 |
| GCLK15 | P | P127 | D8 | C8 | E7 | D11 | G9 | B12 | H11 | D11 | F14 | D14 | B12 | C16 | C16 |
| GCLK16 | N | P131 | A7 | A7 | A10 | A9 | E8 | A11 | F11 | A11 | G11 | A14 | A13 | A15 | A15 |
| GCLK17 | P | P132 | B7 | B7 | B10 | B9 | G8 | C11 | F10 | C11 | H12 | B14 | C13 | B15 | B15 |
| GCLK18 | N | P133 | A6 | D8 | A9 | C9 | F7 | A10 | G10 | A10 | F10 | A13 | D13 | C15 | G15 |
| GCLK19 | P | P134 | B6 | E7 | C9 | D9 | E6 | B10 | H10 | B10 | G9 | C13 | E13 | D15 | H15 |
| GCLK20 | N | P14 | F1 | G1 | H3 | H3 | H3 | G1 | G1 | J4 | L4 | R1 | R1 | V3 | V3 |
| GCLK21 | P | P15 | F2 | G2 | H4 | H4 | H4 | G3 | G3 | K3 | M3 | R2 | R2 | V4 | V4 |
| GCLK22 | N | P16 | H1 | J3 | H5 | K5 | K5 | P3 | P3 | K4 | M4 | P8 | R6 | W4 | W4 |
| GCLK23 | P | P17 | H2 | K4 | J6 | L5 | L5 | N4 | N4 | K5 | M5 | N8 | R7 | W5 | W5 |
| GCLK24 | N | P21 | G1 | H1 | J4 | K3 | K3 | H1 | H1 | L4 | N4 | W3 | W3 | AB1 | AB1 |
| GCLK25 | P | P22 | G2 | H3 | K3 | K4 | K4 | H2 | H2 | M3 | P3 | V4 | V4 | AB2 | AB2 |
| GCLK26 | N | P23 | J1 | J1 | F1 | H1 | H1 | J1 | J1 | J1 | L1 | T1 | T1 | AA1 | AA1 |
| GCLK27 | P | P24 | J2 | J2 | F2 | H2 | H2 | J3 | J3 | J3 | L3 | T3 | T3 | AA3 | AA3 |
| GCLK28 | N | N/A | N/A | R7 | T7 | V9 | V9 | AB10 | AB10 | AB11 | AB11 | AF12 | AF14 | AK17 | AK17 |
| GCLK29 | P | N/A | N/A | P7 | R7 | T9 | T9 | AA10 | AA10 | Y11 | Y11 | AD12 | AD14 | AH17 | AJ17 |
| GCLK30 | N | P50 | P7 | L8 | M7 | T8 | T8 | AB11 | AB11 | AB12 | AB12 | AD13 | AF13 | AK16 | AK16 |
| GCLK31 | P | P51 | N7 | K8 | P7 | R8 | R8 | Y11 | Y11 | AA12 | AA12 | AC13 | AE13 | AJ16 | AH16 |

Figure 1-9 illustrates a GCLK pin layout for the four bank Spartan-6 devices which include XC6SLX4, XC6SLX9, XC6SLX16, XC6SLX45, XC6SLX45T; and the XC6SLX75/75T, XC6SLX100/100T, and XC6SLX150/150T in the FG(G)484 and CSG484 packages. Input routing to BUFI02, DCM, PLL, and BUFG resources are shown in **Figure 1-9**. The BUFI02 routing to BUFI02 clocking regions, DCM, PLL, and BUFG resources are also indicated. For devices with four V_{CCO} banks, there are two BUFI02 clocking regions on each edge of the device.

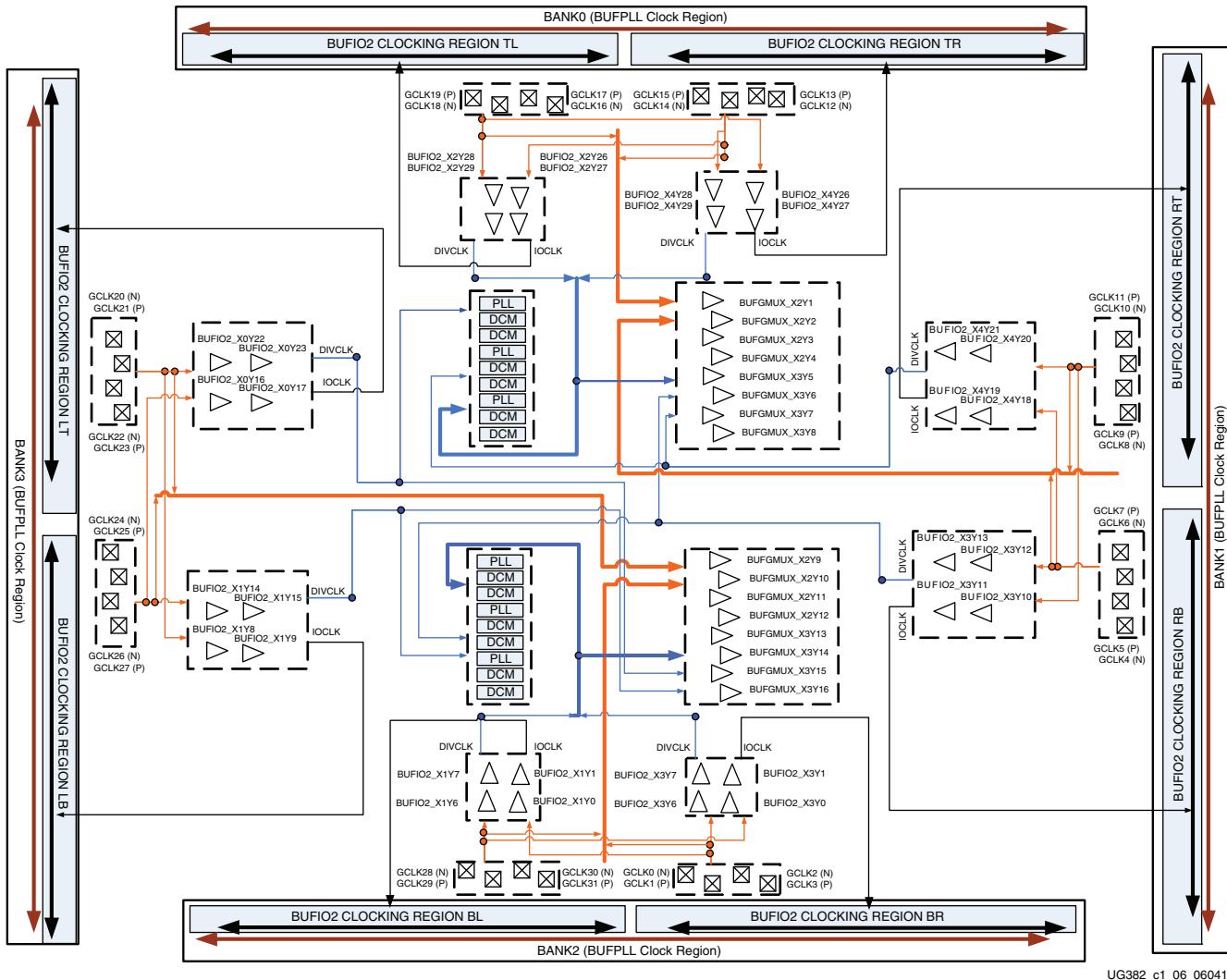


Figure 1-9: Spartan-6 FPGA Clock Pin Layout—Devices with Four Banks

UG382_c1_06_060410

The XC6SLX25 and XC6SLX25T have 12 pins in bank 1 and bank 3 that are associated with a BUFIO2 clocking region that is different from the package pins of other devices. This difference affects package migration for the FT(G)256, CSG324, and FG(G)484 packages.

Eight of the affected pins are GCLK pins. Figure 1-10 shows the BUFIO2 clocking regions for XC6SLX25 and XC6SLX25T. Because every GCLK pin contains SelectIO logic resources, the SelectIO logic resources associated with the GCLK pins must be clocked by a BUFIO2 within the BUFIO2 clocking region. The BUFIO2 clocking regions are LB for GCLK[20:23] and RB for GCLK[8:11]. In all other devices the BUFIO2 clocking regions are LT and RT.

As shown in [Figure 1-8](#), GCLK pins are commonly used to route to clocking resources such as PLL and DCM clock inputs. In the XC6SLX25 and XC6SLX25T, the input routing remains unchanged for GCLK[20:23] and GCLK[8:11]. When using a DCM or PLL, the BUFIO2 typically uses a BUFIO2 located in the LT (GCLK[20:23]) and RT (GCLK[8: 11]) BUFIO2 clocking regions.

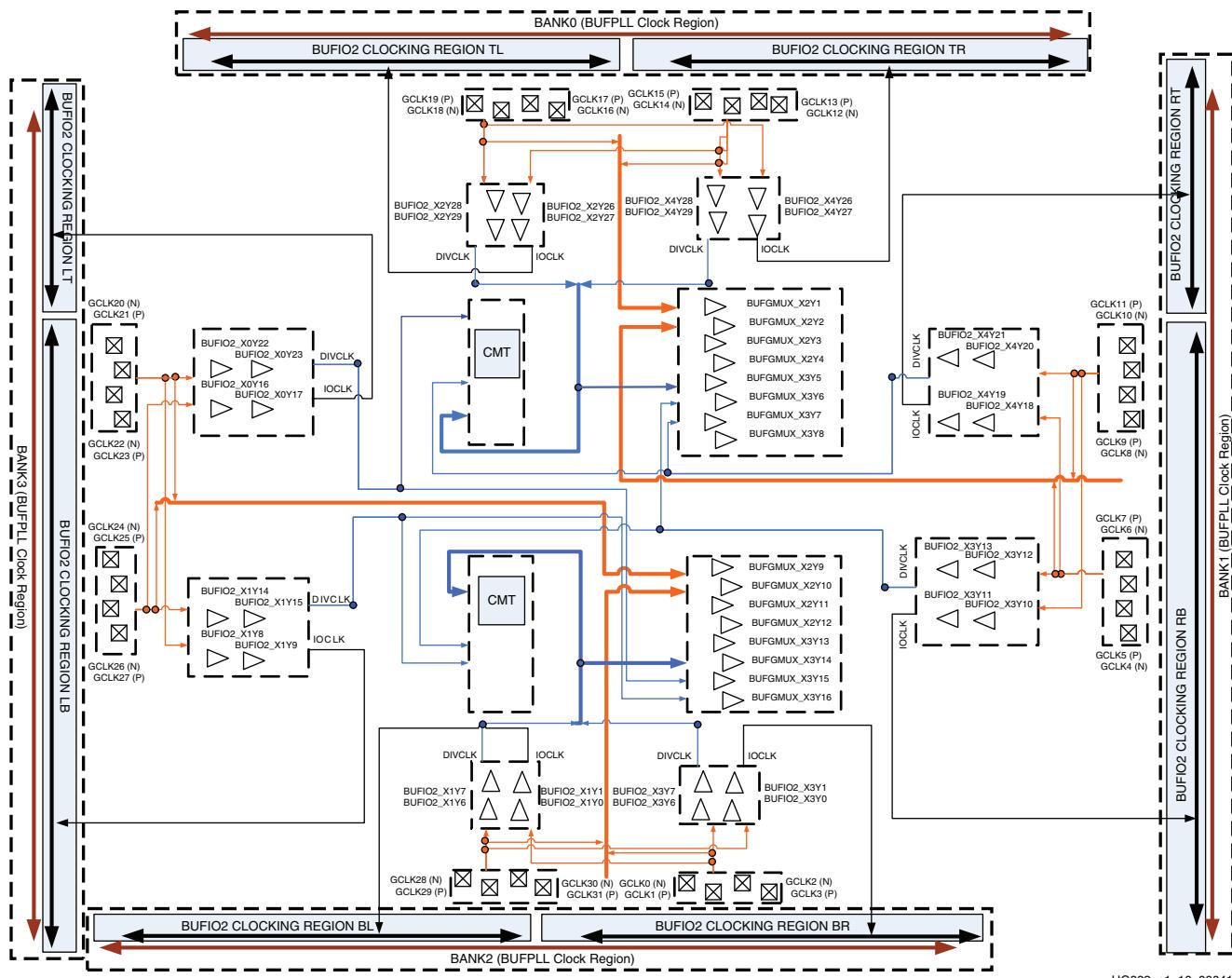


Figure 1-10: Spartan-6 FPGA Clock Pin Layout—XC6SLX25 and XC6SLX25T Only

Figure 1-11 shows the GCLK pin layout for the larger devices with the additional bank 4 and bank 5. The devices include the XC6SLX75, XC6SLX75T, and XC6SLX100 in the FG(G)676 package, and the XC6SLX100T, XC6SLX150, and XC6SLX150T in the FG(G)676 and FG(G)900 packages. GCLK20 through GCLK23 are powered from V_{CCO} bank 3 while the IOCLks connect to BUFIO2 clocking region LT. Similarly, GCLK8 through GCLK11 are powered from V_{CCO} bank 1 while the IOCLks connect to BUFIO2 clocking region RT.

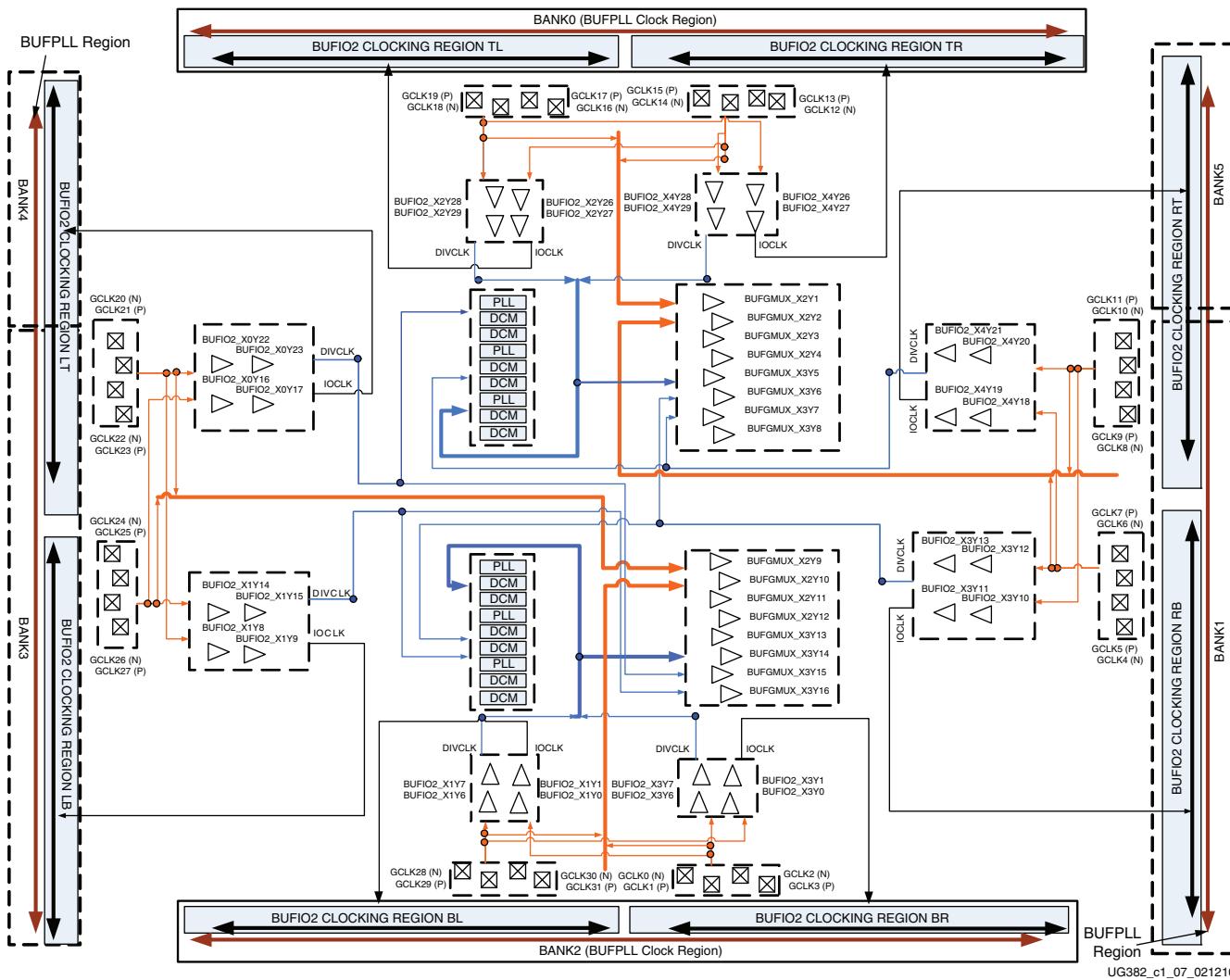


Figure 1-11: Spartan-6 FPGA Clock Pin Layout—Devices with Additional Bank 4 and Bank 5

Designs using BUFIO2s to clock SelectIO logic require additional pin planning considerations. Six bank devices must consider the V_{CCO} IOSTANDARD banking rules and the clock routing of the BUFIO2 clocking regions. [Table 1-7](#) compares the four bank devices with the six bank devices. Pin planning for V_{CCO1} and V_{CCO3} can potentially require clocks to come from two separate BUFIO2 clocking regions. Alternatively, pin planning based on the RT or LT BUFIO2 clocking regions have the potential to place pins in two separate V_{CCO} banks.

Table 1-7: V_{CCO} and BUFIO2 Clocking Regions for Pin Planning

| V_{CCO} Bank | BUFIO2 Clock Region | |
|-------------------|---------------------|------------------|
| | Four Banks | Six Banks |
| 0 | TL, TR | TL, TR |
| 1 | RT, RB | RB, RT (partial) |
| 2 | BL, BR | BL, BR |
| 3 | LT, LB | LB, LT (partial) |
| 4 | | LT |
| 5 | | RT |

In both four bank and six bank devices the global clock inputs are in the same V_{CCO} bank. In six bank devices, GCLK[8:11] is powered by V_{CCO1} . Any SelectIO logic resources associated with GCLK[8:11] require a BUFIO2 associated with the RT BUFIO2 clocking region as summarized in Table 1-8.

Table 1-8: GCLK V_{CCO} Bank Support and BUFIO2 Clocking Region Requirements

| GCLK | V_{CCO} | BUFIO2 (Routing to ILOGIC2 and OLOGIC2) | BUFIO2 (Routing to BUFGMUX, DCM, and PLL) |
|-------------|------------------|---|--|
| GCLK[0:3] | 2 | X3Y[0,1,6,7] | X1Y[0,1,6,7] X3Y[0,1,6,7] |
| GCLK[4:7] | 1 | X3Y[10,11,12,13] | X3Y[10,11,12,13] X4Y[18,19,20,21] |
| GCLK[8:11] | 1 ⁽¹⁾ | X4Y[18,19,20,21] X3Y[10,11,12,13] for XC6SLX25 and XC6SLX25T | X3Y[10,11,12,13] X4Y[18,19,20,21] |
| GCLK[12:15] | 0 | X4Y[26,27,28,29] | X4Y[26,27,28,29] X2Y[26,27,28,29] |
| GCLK[16:19] | 0 | X2Y[26,27,28,29] | X4Y[26,27,28,29] X2Y[26,27,28,29] |
| GCLK[20:23] | 3 ⁽¹⁾ | X0Y[16,17,22,23] X1Y[14,15,18,19] for XC6SLX25 and XC6SLX25T | X0Y[16,17,22,23] X1Y[14,15,18,19] |
| GCLK[24:27] | 3 | X1Y[14,15,18,19] | X0Y[16,17,22,23] X1Y[14,15,18,19] |
| GCLK[28:31] | 2 | X1Y[0,1,6,7] | X1Y[0,1,6,7] X3Y[0,1,6,7] |

Notes:

- Devices with six V_{CCO} banks do not have matching V_{CCO} banks and BUFIO2 clocking regions.

To provide optimal clock routing when using global clock inputs that are directly routed to DCMs and PLLs, the BUFIO2 clock buffers are inferred. Due to this BUFIO2 routing, Bank 0 and Bank 2 global clock inputs can only route to half of the DCMs and PLLs. This limitation is shown in Figure 1-9, Figure 1-10, and Figure 1-11. The BUFIO2 located in the top half of the device (BUFIO2 clocking regions TL, TR, LT, and RT) are limited to routing

to the DCM/PLL in the top half of the device. Similarly, BUFIO2 buffers connected to the bottom half of the device (BUFIO2 clocking regions BL, BR, LB, and RB) are connected to the DCM/PLL in the bottom half of the device. See [Figure 1-9](#), [Figure 1-10](#), and [Figure 1-11](#).

Clocking Structure Guidelines

The advanced features of the Spartan-6 FPGA SelectIO logic can require different clocking structures to support a range of different SelectIO solutions. This section outlines the recommended clocking solutions for optimal performance. A list of possible clocking structures is provided in [High-Speed IOSERDES2 Usage for Advanced Serialization](#).

SDR Data Rate (FD Register in IOB, No IOSERDES2)

Two SelectIO options for registering data into the device.

- [Figure 1-12](#) uses a BUFIO2 (IOCLK) to drive the I/O flip-flop with BUFG (BUFIO2-DIVCLK) to drive FPGA logic registers. There is a routing delay through the BUFG. Works with or without IODELAY2.
- [Figure 1-13](#) uses a BUFG (GCLK) to drive both FPGA logic and I/O. Works with or without IODELAY2.

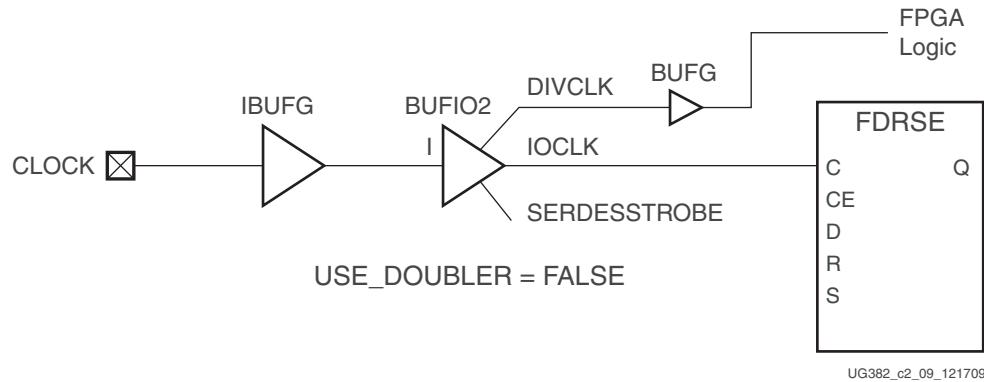


Figure 1-12: I/O Flip-Flop Clocking using BUFIO2

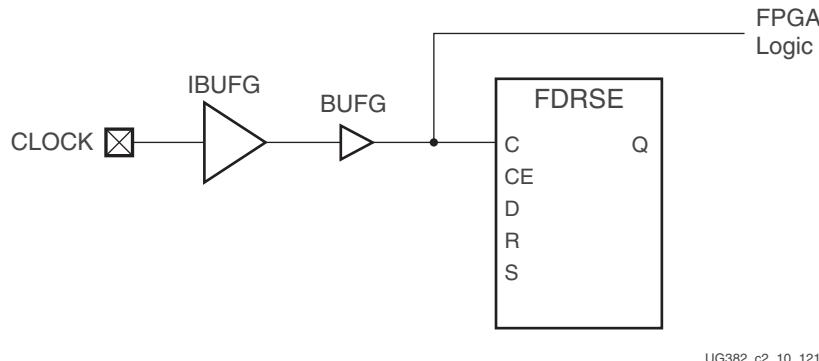


Figure 1-13: I/O Flip-Flop Clocking using BUFG

DDR Data Rate (IDDR2, ODDR2, No IOSERDES2)

The following options can be used for clocking IDDR2 and ODDR2 primitives.

- When performance is not critical, use a single DCM output to drive both clock (C0) and the inverted clock (C1) using local inversion. Works with or without IODELAY2.
- For the highest performance, use two DCM outputs with separate BUFGs with 180° phase difference. Works with or without IODELAY2. See [Figure 1-18](#).
- If not using a DCM, then the GCLK input should directly drive two BUFIO2s. Use two BUFIO2s with the first BUFIO2 (USE_DOUBLER) for C0 and an inverted clock using BUFIO2 (I_INVERT = TRUE) for C1 connected to the same GCLK. FPGA logic is driven by BUFG (C0 BUFIO2-DIVCLK). There is a routing delay through the BUFG.
 - Using an IODELAY2 requires the IBUFGDS_DIFF_OUT. (See [Figure 1-35, page 50](#))
 - A single-ended input with IODELAY2 is not supported.
- For bidirectional interfaces, input and output logic must both use the same data rate (IDDR2 and ODDR2). Mixing SDR and DDR bidirectional I/Os are not permitted.

High-Speed IOSERDES2 Usage for Advanced Serialization

IOSERDES2 (SDR)

IOSERDES2 (SDR) requires one BUFIO2. BUFIO2 (USE_DOUBLER = FALSE and I_INVERT = FALSE) with IOCE driven by BUFIO2-SERDESSTROBE and CLKDIV driven by BUFG (BUFIO2-DIVCLK). FPGA logic driven by BUFG (BUFIO2 - DIVCLK). The SERDESSTROBE resolves the BUFG routing delays. Works with or without IODELAY2. See [Figure 1-14](#).

IOSERDES2 (DDR)

IOSERDES2 (DDR) requires two BUFIO2s. Use first BUFIO2 (USE_DOUBLER = TRUE) with IOCE driven by BUFIO2 (SERDESSTROBE) and CLKDIV driven by BUFG (BUFIO2-DIVCLK). Second BUFIO2 uses (I_INVERT = TRUE, USE_DOUBLER = FALSE) to drive C1 clock input. The SERDESSTROBE resolves the BUFG routing delays. See [Figure 1-15](#).

- Using an IODELAY2 requires the IBUFGDS_DIFF_OUT. (See [Figure 1-31, page 45](#)).
- Single-ended input with IODELAY2 is not supported in this case.

IOSERDES2 with PLL

Only SDR is supported. The GCLK inputs drive the inferred BUFIO2's DIVCLK output, which drives the PLL clock input. The PLL uses two clock outputs to drive the BUFPPLL's PLLIN input and the BUFPPLL's GCLK input from the BUFG output. Connect the BUFPPLL's LOCKED input to the PLL's LOCKED output. Works with or without IODELAY2. See [Figure 1-16](#).

For bidirectional interfaces, input and output logic must set DATA_RATE identically for input logic and output logic. Mixing SDR and DDR bidirectional I/Os is not permitted.

Examples of High-Speed I/O Clock Network Connections

The examples in this section illustrate how the global clock buffers, I/O clock buffers, and I/O tile can be used together for high-speed interfaces.

The example in [Figure 1-14](#) shows the simplest implementation where an input clock is used to clock the ISERDES2. The BUFI02 provides the FPGA logic clock as well as the strobe signals used by ISERDES2. Serial data is clocked using IOCLK on the I/O clock network. Parallel data is clocked out in the FPGA clock domain using the DIVCLK output and the synchronous SERDESSTROBE signal.

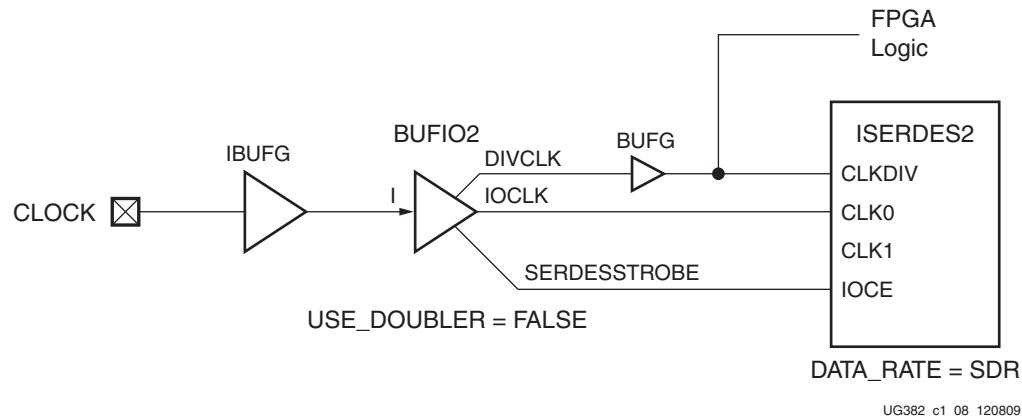


Figure 1-14: Example 1: BUFI02 Driving ISERDES2, DATA_RATE = SDR

In DDR applications where serialized data is clocked in on both the rising and falling edges of the input clock, a second inverted clock is required to drive CLK1 as shown [Figure 1-15](#). Because data is clocked in on both edges of the IOCLK, DIVCLK is divided by a DIVIDE/2 by setting `USE_DOUBLER = TRUE`.

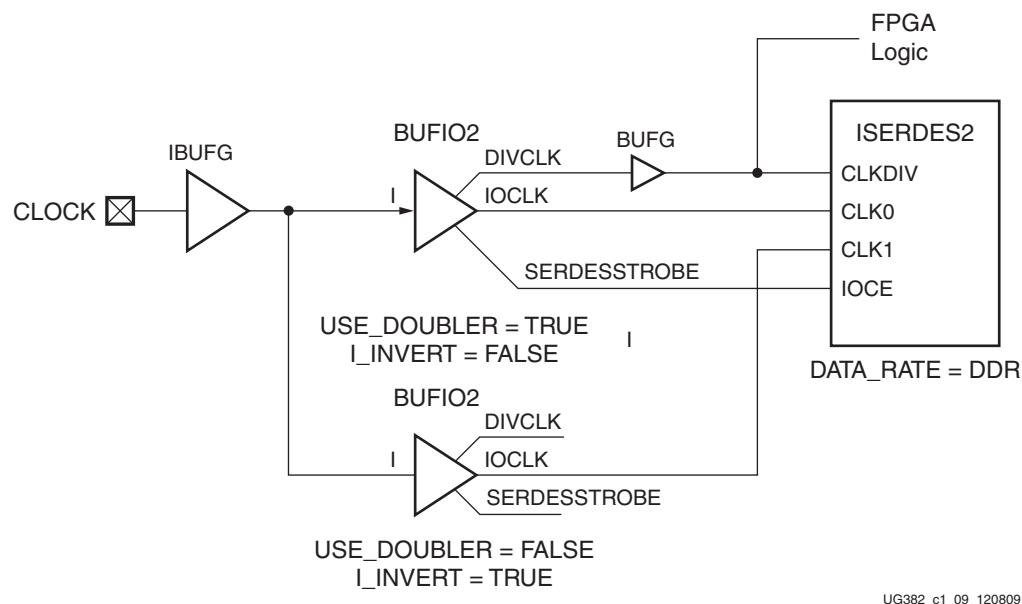


Figure 1-15: Example 2: BUFI02 Driving ISERDES2, DATA_RATE = DDR

In applications where the input clock is run at the frequency of the parallel data and not the serial data, such as the pixel clock for video applications, the input clock must be multiplied up to create a high-speed I/O clock. Figure 1-16 shows a PLL providing the high-speed I/O clock required for the ISERDES.

GCLK clock inputs are automatically routed to the PLL and DCM clock inputs using a BUFIO2. This BUFIO2 routing path allows the input path to be deskewed using the BUFIO2FB (if needed).

The PLL drives the I/O clock network with the CLKOUT0 output. BUFIO2FB is balanced to deskew the input routing delays associated with the primary BUFIO2. The FPGA clock domain is driven by a separate PLL clock output using a BUFG.

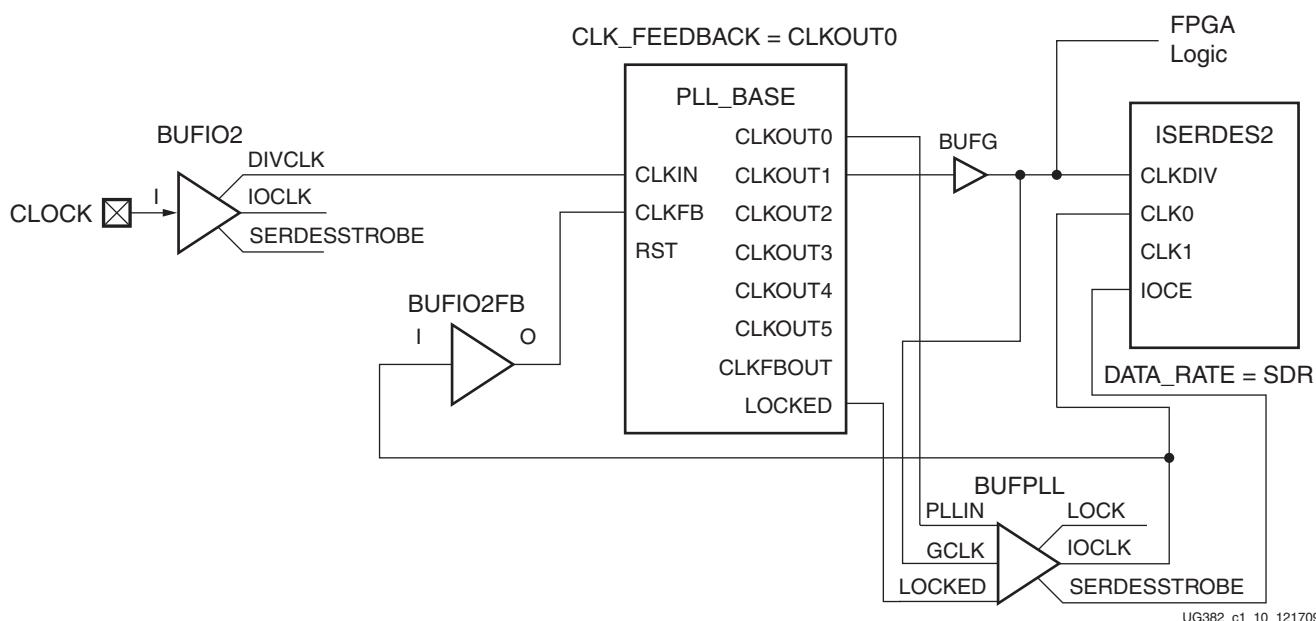


Figure 1-16: Example 3: Basic PLL ISERDES2 (SDR)

Designs with high-speed source synchronous outputs do not require exact timing delays from the GCLK input to the I/O clock domain. When timing alignment is not required, the PLL can use a dedicated feedback from CLKFBOUT to CLKFBIN without the use of any clock buffers (Figure 1-17).

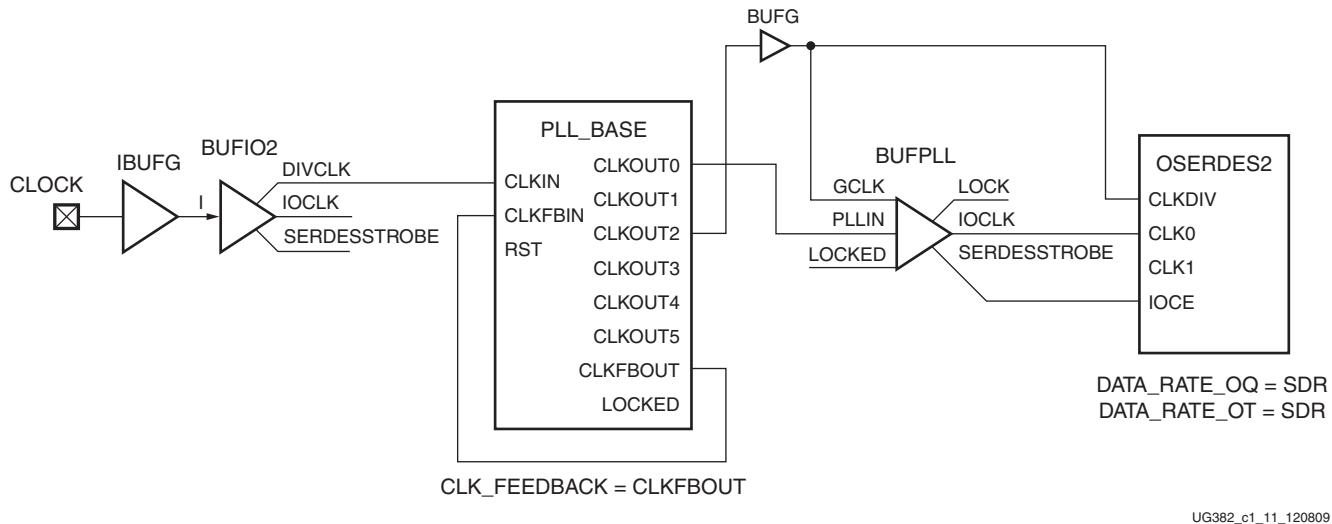


Figure 1-17: Example 4: Basic PLL OSERDES2 (SDR)

In designs where DCM_SP performance is adequate, DCM_SP can be used to drive IOLOGIC (IDDR2) using up to three BUFG clock buffers. To match the input routing delays, either CLK0 or CLK2X must be used to drive the BUFIO2FB as shown in Figure 1-18.

To maintain the best duty-cycle performance using the DCM, use separate DCM clock outputs to drive C0 and C1. Each DCM output drives a separate global buffer. While possible using the BUFG to invert one phase locally within the I/O tile, it is not recommended as locally inverting one of the clocks inserts duty-cycle distortion.

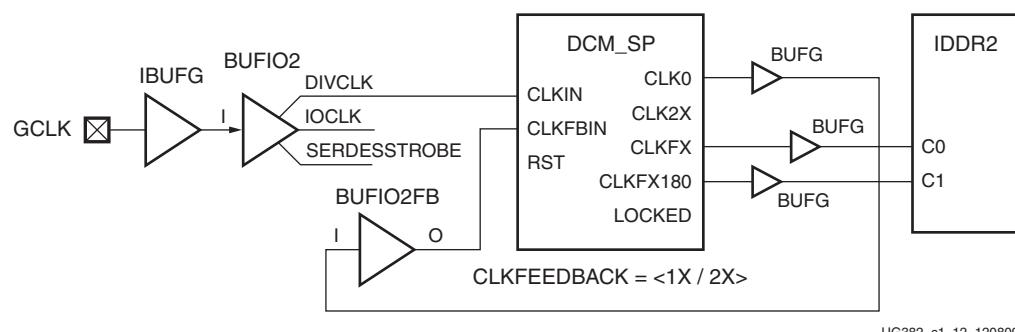


Figure 1-18: Example 5: DCM Deskewed to the Inputs

When using GTP transceivers, GTP reference clocks contain dedicated routing connections to BUFIO2 that can be used to connect to the Clock Management Tiles (CMT).

Example 6 in [Figure 1-19](#) shows a single lane PCI Express design using the PLL_BASE and a feedback path routed through the GTP_DUAL using the BUFIO2FB. See [UG386, Spartan-6 FPGA GTP Transceivers User Guide](#) for more information.

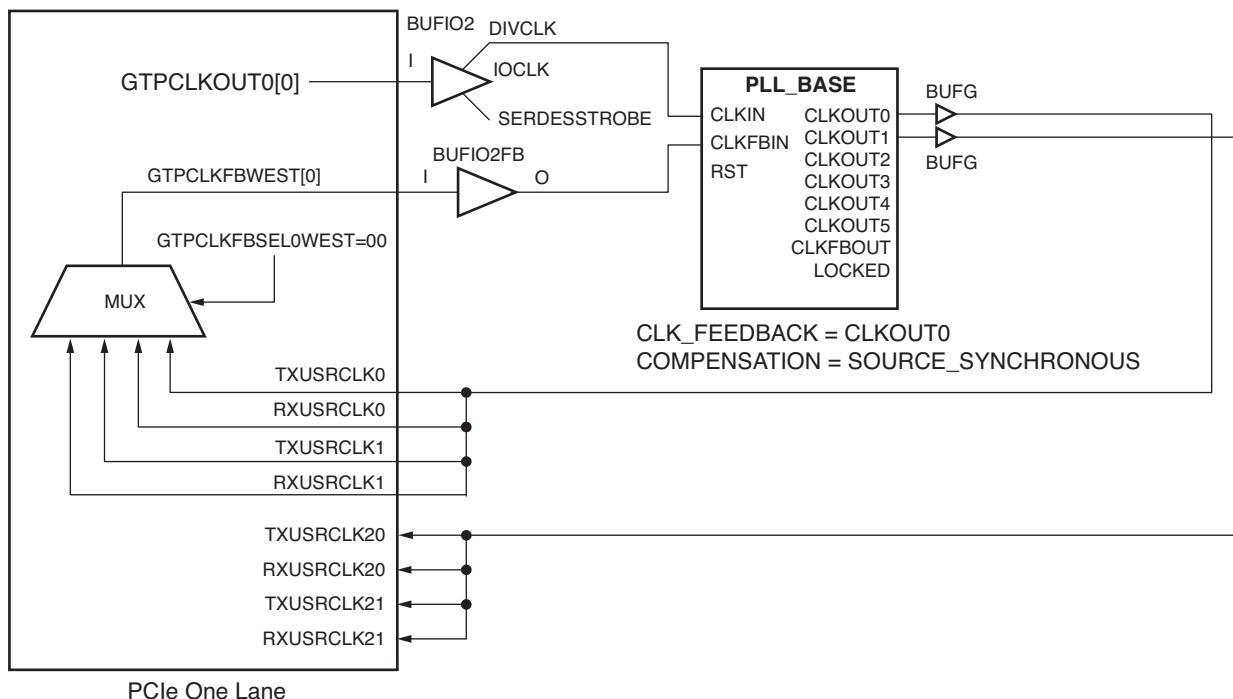
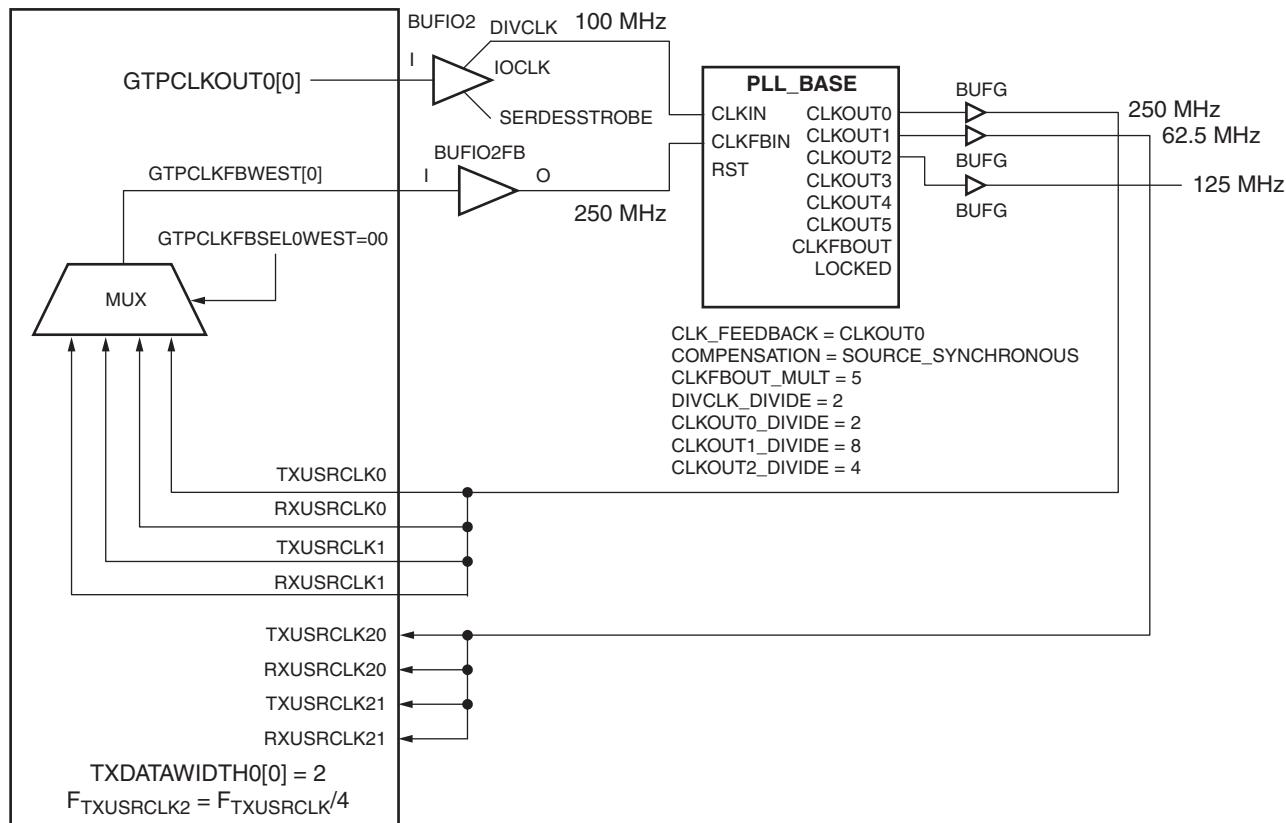


Figure 1-19: Example 6: Single Lane Clocking for PCI Express

Example 7 in [Figure 1-20](#) shows a single lane PCI Express design where the input frequency and feedback frequency are different. Because the input frequency and the feedback frequency do not match, additional analysis of the PLL settings is required. In this example, GTPCLKOUT0[0] is a 100 MHz clock. To create a feedback clock frequency that meets both the GTP_DUAL and PLL requirements, a 250 MHz feedback clock is selected by setting TXDATAWIDTH0[0] = 2.

To allow for more flexibility in matching the Phase-Frequency Detector (PFD) frequencies, the PLL is setup to use CLK_FEEDBACK = CLKOUT0. The PFD operates at 50 MHz and, using the given settings, the VCO is 500 MHz.

Additional instructions for selecting PLL settings are described in [Chapter 3, Phase-Locked Loops](#). See [UG386, Spartan-6 FPGA GTP Transceivers User Guide](#) for more information.



ug382_c1_17_081710

Figure 1-20: Example 7: Single Lane Clocking Example for PCI Express with 100 MHz Clock

Clock Buffers and Multiplexers

The clock buffers and multiplexers either drive clock input signals directly onto a clock line (BUFG or BUFPLL) or optionally provide a multiplexer to switch between two unrelated, possibly asynchronous clock signals (BUFMUX).

Clock buffers are designed to drive clock signals.

- When connected to a reset/set for a slice, PSCLK for a DCM_SP, PROGCLK for DCM_CLKGEN, or DCLK for PLL_ADV, the clock buffer must be located in one of the following locations: BUFMUX_X2Y9, BUFMUX_X2Y10, BUFMUX_X2Y11, BUFMUX_X2Y12, BUFMUX_X3Y13, BUFMUX_X3Y14, BUFMUX_X3Y15, or BUFMUX_X3Y16.
- When connected to a reset/set for a block RAM, a combinatorial input for a slice, clock enable, or a BUFMUX input, the clock buffer must be located in one of the following locations: BUFMUX_X2Y1, BUFMUX_X2Y2, BUFMUX_X2Y3, BUFMUX_X2Y4, BUFMUX_X3Y5, BUFMUX_X3Y6, BUFMUX_X3Y7, BUFMUX_X3Y8.

Global Clock Input Buffer Primitives

Different configurations of the clock input buffer using the IBUFG and IBUFGDS primitives are shown in [Table 1-9](#). These primitives work in conjunction with the SelectIO resource by setting the IOSTANDARD attribute to a chosen standard. Refer to [UG381, Spartan-6 FPGA SelectIO Resources User Guide](#) for a complete list of supported I/O standards.

Table 1-9: Global Clock Input 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 |

Global Clock Buffer Primitives

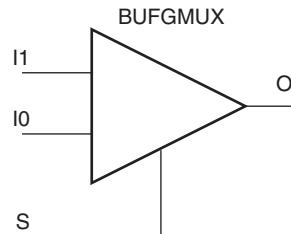
The primitives in [Table 1-10](#) are different configurations of the global clock buffers.

Table 1-10: Global Clock Buffer Primitives

| Primitive | Input | Output | Control |
|-----------|--------|--------|---------|
| BUFMUX | I0, I1 | O | S |
| BUFMUX_1 | I0, I1 | O | S |
| BUFG | I | O | – |
| BUFGCE | I | O | CE |
| BUFGCE_1 | I | O | CE |
| BUFH | I | O | – |

BUFGMUX

Each BUFGMUX primitive, shown in [Figure 1-21](#), is a 2-to-1 multiplexer. S, the select line, chooses from the two inputs, I0 or I1 to drive the BUFGMUX output signal, O, as described in [Table 1-11](#). As specified in the *Spartan-6 FPGA Data Sheet*, the S input has a setup time requirement. It also has programmable polarity.



ug382_c1_14_120809

Figure 1-21: BUFGMUX Primitive

Table 1-11: BUFGMUX Primitive

| S Input | O Output |
|---------|----------|
| 0 | I0 Input |
| 1 | I1 Input |

Table 1-12: BUFGMUX Attribute

| Attribute Name | Description | Possible Values | Default Value |
|----------------|---------------------------------------|-----------------|---------------|
| CLK_SEL_TYPE | Specifies synchronous or asynchronous | SYNC, ASYNC | SYNC |

The BUFGMUX multiplexes two clock signals while eliminating any timing hazards by switching from one clock source to a completely asynchronous clock source without glitches when used with the default CLK_SEL_TYPE (SYNC). The primitive guarantees that when the select line S is toggled to choose the other clock source, the output remains in the inactive state until the next active clock edge on either input. The output can be either High or Low when disabled (when toggling between clock inputs). The default is Low. A cross-coupled register pair ensures the BUFGMUX output does not inadvertently generate a clock edge. The default CLK_SEL_TYPE of SYNC should only be used when both clock sources are actively switching.

When the S input changes, the BUFGMUX does not drive the new input to the output until the previous clock input is Low and the new clock input has a High-to-Low transition ([Table 1-13](#)). By not toggling on the first Low-to-High transition of the input, the output clock pulse is never shorter than the shortest input clock pulse.

The S input selects clock input I0 when Low and I1 when High, but also has built-in programmable polarity, equivalent to swapping I0 and I1. Programmable polarity on the clock signal is available at each slice, which can be rising-edge or falling-edge triggered, avoiding having to generate and propagate two separate clock signals.

Table 1-13: BUFGMUX Functionality

| Inputs | | | Outputs |
|--------|----|---|---------|
| I0 | I1 | S | O |
| I0 | X | 0 | I0 |
| X | I1 | 1 | I1 |
| X | X | ↑ | 0 |
| X | X | ↓ | 0 |

Some applications require immediate switching between clock inputs, which is supported by setting CLK_SEL_TYPE = ASYNC. For example, if either of the clock inputs is no longer switching, the BUFGMUX must use asynchronous switching to disable the glitch filtering. When the clock inputs stop, the BUFGMUX's glitch filtering cannot detect the required clock edges. By using the asynchronous mode, the BUFGMUX switches immediately. Figure Figure 1-22 shows how a short clock pulse can occur when using CLK_SEL_TYPE = ASYNC.

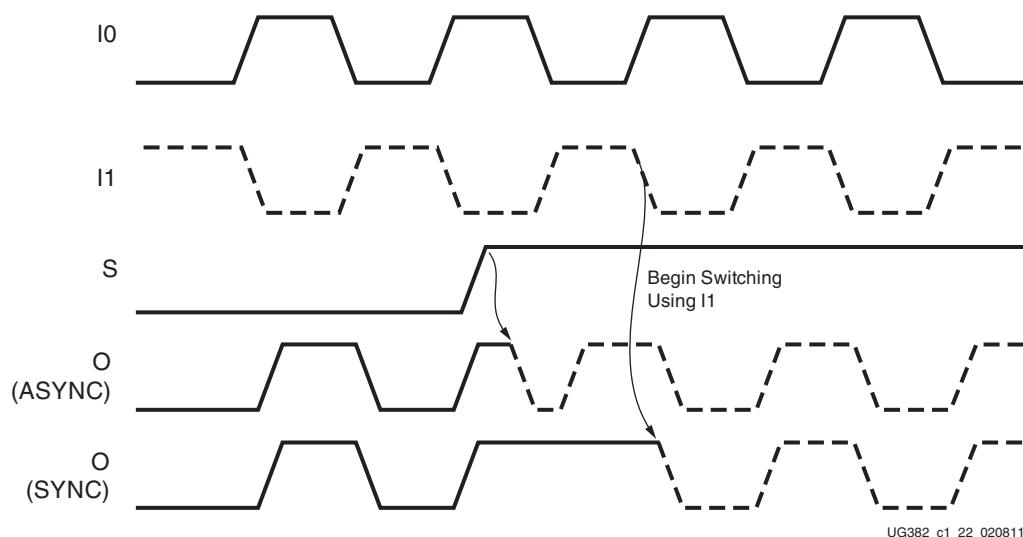


Figure 1-22: BUFGMUX ASYNC vs. SYNC Timing Diagram

In Figure 1-22:

- The current clock is from I0.
- S is activated High.
- The clock output immediately switches to I1 O (CLK_SEL_TYPE = ASYNC).
- The clock output switches with no glitches to I1 O (CLK_SEL_TYPE = SYNC) after I0 is Low and I1 transitions from High to Low.

When only one clock input is needed, a BUFG primitive should be chosen since a second clock input and select lines would not be used.

The BUFGMUX is initialized with I0 selected at power-up and after the assertion of the Global Set/Reset (GSR). Simulation also starts with S = 0 at time 0. If S = 1 at time 0, the output is unknown until the next falling edge of I1.

The select line can change at almost any time, independent of the clock states or transitions. The only exception is a short setup time prior to a Low-to-High transition on the selected clock input, which can result in an undefined runt pulse output.

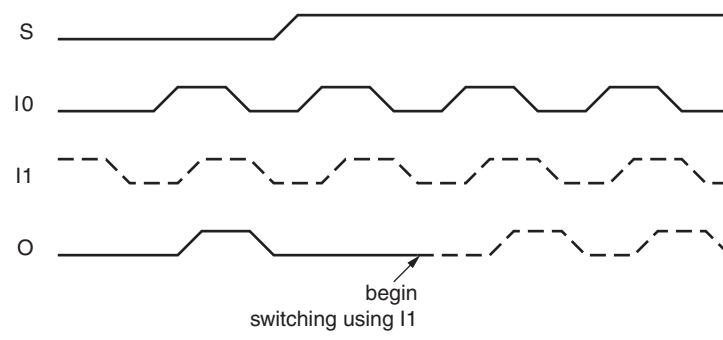
BUFGMUX_1

BUFGMUX and BUFGMUX_1 are distinguished by which state the output assumes when it switches between clocks in response to a change in its select input. BUFGMUX assumes output state 0 and BUFGMUX_1 assumes output state 1 ([Table 1-14](#)).

Table 1-14: BUFGMUX_1 Functionality

| Inputs | | | Outputs |
|--------|----|---|---------|
| I0 | I1 | S | O |
| I0 | X | 0 | I0 |
| X | I1 | 1 | I1 |
| X | X | ↑ | 1 |
| X | X | ↓ | 1 |

To understand the difference between BUFGMUX and BUFGMUX_1, first consider how BUFGMUX operates. [Figure 1-23](#) illustrates the timing diagram for BUFGMUX.



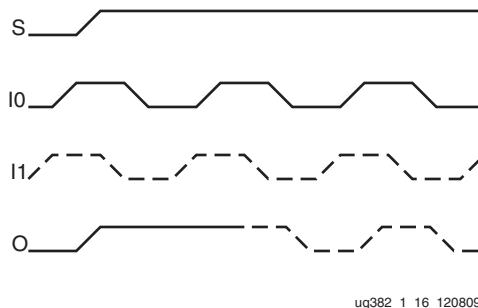
ug382_c1_15_020711

Figure 1-23: BUFGMUX (CLK_SEL_TYPE = SYNC) Timing Diagram

In [Figure 1-23](#):

- 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.
- 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 1-24](#) illustrates the timing diagram for BUFGMUX_1.



ug382_1_16_120809

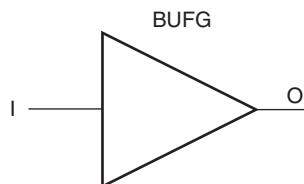
Figure 1-24: BUFGMUX_1 (CLK_SEL_TYPE = SYNC) Timing Diagram

In Figure 1-24:

- 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.
- No glitches or short pulses can appear on the output.

BUFG

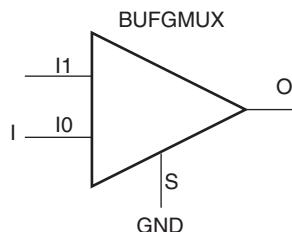
The BUFGMUX is the physical clock buffer in the device, but it can be used as a simple single-input clock buffer. The BUFG clock buffer primitive (see Figure 1-25) drives a single clock signal onto the clock network and is essentially the same as a BUFGMUX, without the clock select mechanism. BUFG is the generic primitive for clock buffers across multiple Virtex and Spartan architectures.



ug382_c1_17_120809

Figure 1-25: BUFG Primitive

The BUFG is built from the BUFGMUX as shown in Figure 1-26.

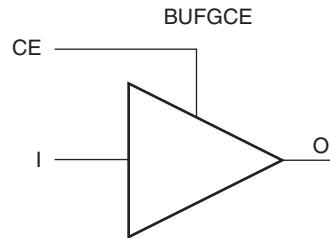


ug382_c1_18_120809

Figure 1-26: BUFG Built from a BUFGMUX

BUFGCE and BUFGCE_1

The BUFGCE primitive creates an enabled clock buffer using the BUFGMUX select mechanism (see [Figure 1-27](#)). BUFGCE is a global clock buffer with a single gated input. Its O output is 0 when clock enable (CE) is Low (inactive). When clock enable (CE) is High, the I input is transferred to the O output. The BUFGCE truth table is shown in [Table 1-15](#).



ug382_c1_19_120809

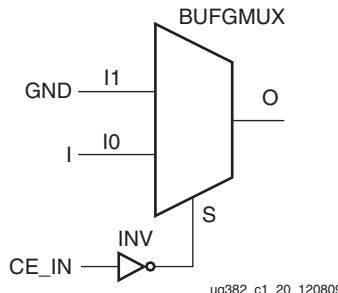
Figure 1-27: BUFGCE Primitive

Table 1-15: BUFGCE Truth Table

| S Input | | O Output |
|---------|----|----------|
| I | CE | O |
| X | 0 | 0 |
| X | 1 | I |

The BUFGCE is built from the BUFGMUX by multiplexing a fixed value for one input. The default value is Low when disabled. The BUFGCE_1 primitive is similar with V_{CC} connected to I1, making the output High when disabled. It also uses the BUFGMUX_1 primitive to provide glitchless operation during the transition between inputs.

[Figure 1-28](#) shows the equivalent functionality, although the library element truly is a primitive. The CE inversion is built into the BUFGMUX functionality. The 0 source can be fed from any convenient unused LUT.



ug382_c1_20_120809

Figure 1-28: Equivalent Functionality of BUFGCE

BUFH

The horizontal clock buffer (BUFH) allows access to one half of an HCLK row clock. The BUFH clock primitive (see [Figure 1-29](#)) drives a single clock signal in the half of the HCLK row. The BUFH is used to clock interconnect logic, SelectIO logic, DSP48A1 tiles, or block RAM resources. The BUFH is accessed using FPGA interconnect logic or directly using any clock output from a DCM, PLL, or GTPA1_DUAL tile in the same HCLK row.

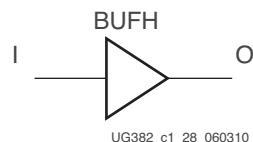


Figure 1-29: BUFH Primitive

Similar to the BUFG restrictions, the BUFH clock buffers are designed to drive clock signals. When a BUFH is used to reset or set interconnect logic, its placement is limited to the top eight BUFH locations in the HCLK row. When used as an asynchronous set or reset for a block RAM, as a clock enable, or as combinatorial logic, the only the lower eight BUFH locations can be used.

BUFH is not recommended for the DCM feedback path or PLL feedback path.

Clock Buffers for the High-Speed I/O Clock Region

To improve the performance of the I/O tile, Spartan-6 FPGAs contain a dedicated I/O clock network for the connections where performance is most critical. I/O clock buffers [Table 1-16](#) work with the IODELAY2, IDDR2, ODDR2, ISERDES2, and OSERDES2 logic in the I/O tile by connecting to both the I/O clock network and the FPGA logic. The I/O clock buffer outputs restricted to the I/O clock network are listed in [Table 1-17](#).

Table 1-16: I/O Clock Buffers

| Primitive | Input | Output | Control |
|-------------|---------------------|--|---------|
| BUFIO2 | I | IOCLK, DIVCLK, SERDESSTROBE | - |
| BUFIO2_2CLK | I, IB | DIVCLK, IOCLK, SERDESSTROBE | - |
| BUFPLL | GCLK, PLLIN, LOCKED | IOCLK, SERDESSTROBE, LOCK | - |
| BUFPLL_MCB | PLLIN0, PLLIN1 | IOCLK0, IOCLK1, SERDESSTROBE0, SERDESSTROBE1 | - |
| BUFIO2FB | I | O | - |

[Table 1-17](#) lists ports that can be used on the I/O clock network.

Table 1-17: I/O Clock Network Signals

| Primitive | I/O Clock Network Inputs | I/O Clock Network Outputs | Resource |
|-------------|--------------------------|---|------------------|
| BUFIO2 | | IOCLK ⁽¹⁾ , SERDESSTROBE ⁽¹⁾ | I/O Clock Buffer |
| BUFIO2_2CLK | | IOCLK ⁽¹⁾ , SERDESSTROBE ⁽¹⁾ | I/O Clock Buffer |
| BUFPLL | | IOCLK ⁽¹⁾ , SERDESSTROBE ⁽¹⁾ | I/O Clock Buffer |
| BUFPLL_MCB | | IOCLK0 ⁽¹⁾ , IOCLK1 ⁽¹⁾ , SERDESSTROBE0 ⁽¹⁾ , SERDESSTROBE1 ⁽¹⁾ | I/O Clock Buffer |
| BUFIO2FB | I | | I/O Clock Buffer |
| ILOGIC2 | C0, C1 | | I/O Tile Logic |
| OLOGIC2 | C0, C1 | | I/O Tile Logic |
| ISERDES2 | CLK0, CLK1, IOCE | | I/O Tile Logic |
| OSERDES2 | CLK0, CLK1, IOCE | | I/O Tile Logic |
| IODELAY2 | IOCLK0, IOCLK1 | DATAOUT | I/O Tile Logic |
| GTP_DUAL | | CLKOUT0[1:0], CLKOUT1[1:0] | GTP_DUAL Tile |

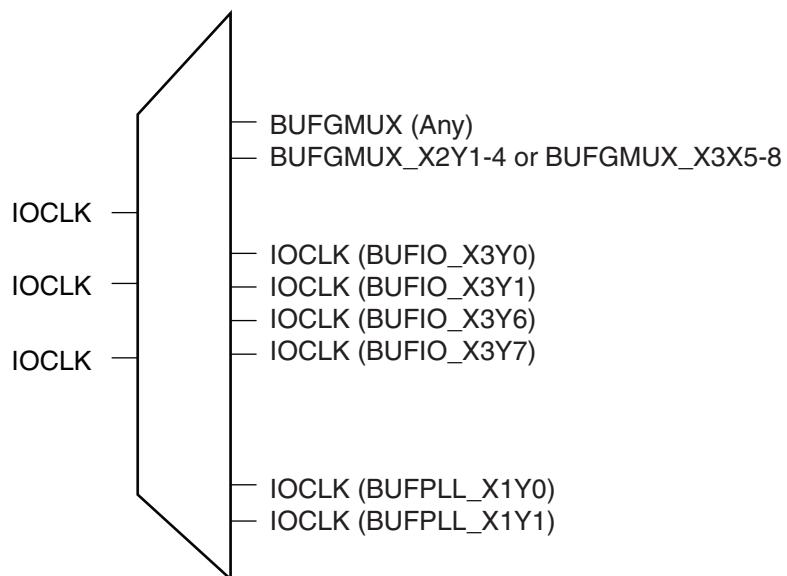
Notes:

1. Outputs only to the I/O clock network. Does not connect to FPGA logic.

Each IOLOGIC supports up to three clocks coming from the I/O clock network. Each IOCLK can be connected to one of eight possible clock sources:

- Two BUFGMUX clock connections. At least one of the clocks must come from BUFGMUX_X2Y[4:1] or BUFGMUX_X3Y[8:5]
- Four BUFIO2 routing connections from a BUFIO2 in the same BUFIO2 clocking region
- Two BUFPLL routing connections from a BUFPLL in the same edge of the device.

An example of the I/O clock network routing for the three I/O clocks is described in [Figure 1-30](#).

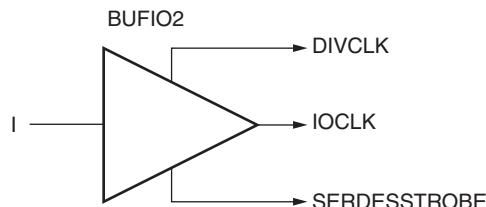


ug382_c1_29_081210

Figure 1-30: Available I/O Clock Network Resources For BUFI02 Clocking Region

BUFI02

The BUFI02 takes a GCLK clock input and generates two clock outputs and a strobe pulse as illustrated in [Figure 1-31](#).



ug382_c1_21_120809

Figure 1-31: BUFI02 Primitive

Table 1-18 lists the BUFIO2 ports. The IOCLK output is a buffered version of the input clock (I). The period and duty-cycle of the DIVCLK output is dependent on the attribute setting listed in [Table 1-19](#).

Table 1-18: BUFIO2 Port List and Definitions

| Port Name | Type | Definition |
|--------------|--------|---|
| I | Input | Clock input. Can optionally be connected to the IODELAY2 (DATAOUT) or the GTP_DUAL tile (GTPCLKOUT0[1:0], GTPCLKOUT1[1:0]). |
| IOCLK | Output | I/O clock network output. Connects to IODDR2 (C0 or C1), IOSERDES2 (CLK0 or CLK1), or IODELAY2 (IOCLK0, IOCLK1). |
| DIVCLK | Output | Divided clock output. Connects to BUFG, PLL_BASE (CLKIN), DCM_SP (CLKIN), and DCM_CLKGEN (CLKIN). |
| SERDESSTROBE | Output | I/O clock network output used to drive IOSERDES2 (IOCE). |

Table 1-19: BUFIO2 Attributes

| Attribute Name | Description | Possible Values | Default Value |
|----------------|--|---------------------|---------------|
| DIVIDE | Sets the DIVCLK and SERDESSTROBE divider divide-by values. FDIVCLK = FIN / DIVIDE <USE_DOUBLE = FALSE> FDIVCLK = (2 * FIN) / DIVIDE <USE_DOUBLE = TRUE> | 1, 3, 4, 5, 6, 7, 8 | 1 |
| DIVIDE_BYPASS | When FALSE, DIVCLK outputs source from divider. When TRUE, DIVCLK outputs source from input I, bypassing the divider. | TRUE, FALSE | TRUE |
| I_INVERT | When set to TRUE, BUFIO2 placement is restricted to the I_INVERT locations of BUFIO2 (see Figure 1-7). Input clock is inverted, shifting the IOCLK output by 180°. DIVCLK and SERDESSTROBE are also affected, as shown in Figure 1-32 and Figure 1-33 . Primarily used for IODDR2 or IOSERDES2 (DATA_RATE = DDR). | TRUE, FALSE | FALSE |
| USE_DOUBLE | Used for ISERDES2/OSERDES2 with DATA_RATE = DDR. When set to TRUE, doubles the DIVCLK and SERDESSTROBE frequencies. FDIVCLK = (2 * FIN) / DIVIDE | TRUE, FALSE | FALSE |

As listed in [Table 1-19](#), when the DIVIDE_BYPASS is set to TRUE, the DIVCLK output is a buffered version of the input clock and the SERDESSTROBE output is driven to 1. When the DIVIDE_BYPASS is set to FALSE then the DIVCLK and SERDESSTROBE outputs will

be the input clock divided by the setting of the divide attribute except when using USE_DOUBLE.

For applications where the I/O clock network is driven by a DDR clock, the USE_DOUBLE must be set to TRUE. Setting USE_DOUBLE = TRUE provides the required DIVCLK and SERDESSTROBE outputs when connected to IOSERDES2 with DATA_RATE = DDR (Figure 1-32). An additional BUFG is required to create the 180° phase shift (see Figure 1-33).

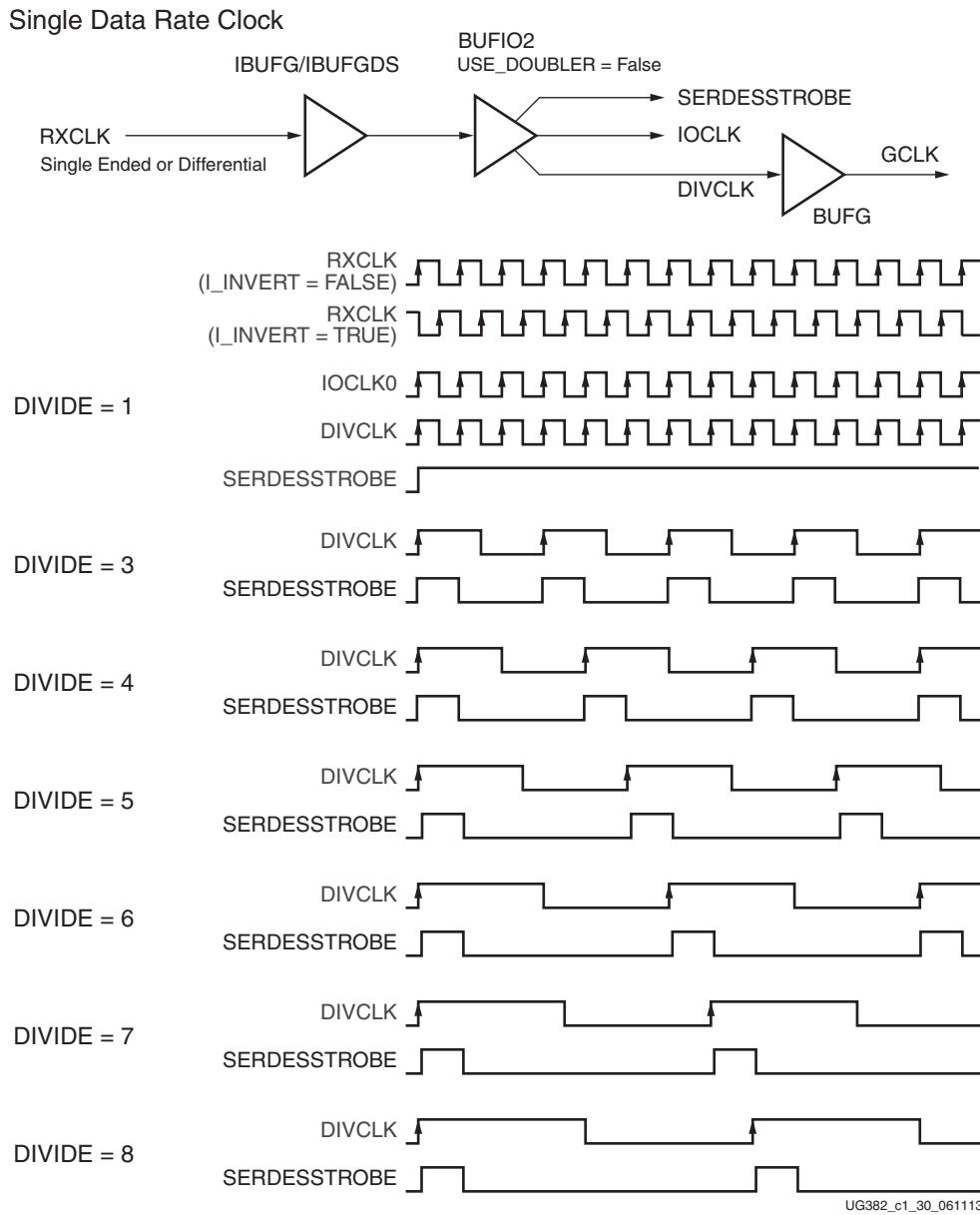
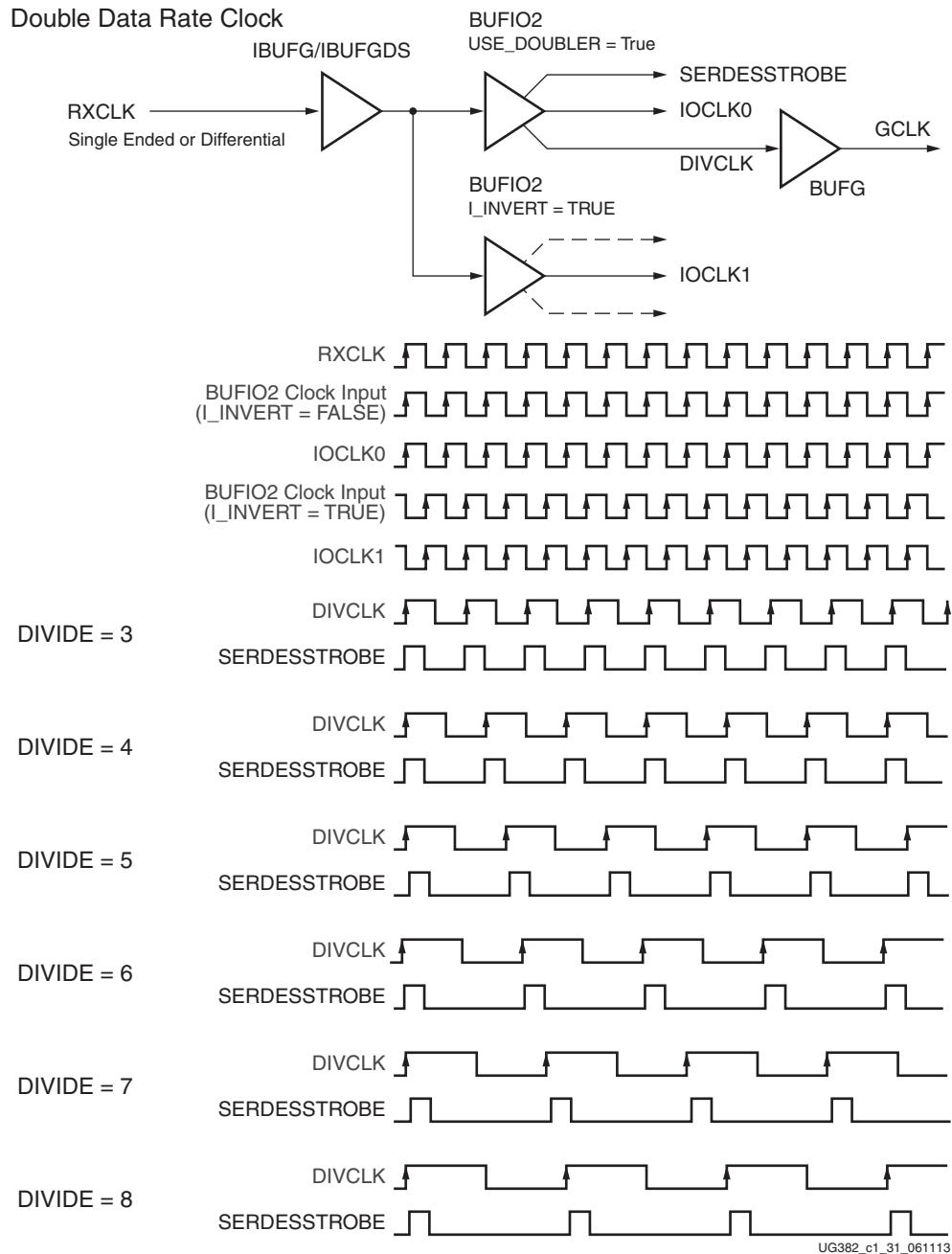


Figure 1-32: BUFI02 Single Data Rate Clock

Figure 1-33: **BUFI02 Double Data Rate Clock**

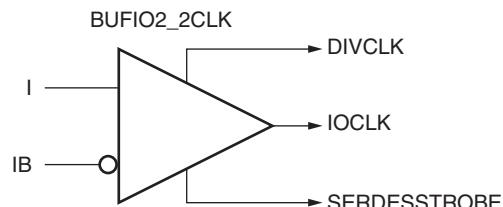
Each BUFIO2 is limited to routing to a single clock for every I/O interface tile. Consequently, when using DDR clocks, a second clock is required with 180° phase shift (Figure 1-33). An example of using two BUFIO2 clock buffers to create the 180° phase shift is shown in Figure 1-15, page 32.

For optimal performance when using a PLL or a DCM, use a BUFIO2. Each GCLK is associated with two BUFIO2 (see Table 1-1 for a complete listing).

A detailed discussion on how to use the strobe and clock outputs is available at [Examples of High-Speed I/O Clock Network Connections, page 32](#).

BUFIO2_2CLK

The BUFIO2_2CLK (Figure 1-34) has almost the same functionality as the BUFIO2 (USE_DBLER = TRUE) except it takes two single-ended clocks or a differential pair (output from IBUFDs_DIFF_OUT). Table 1-20 lists the BUFIO2_2CLK ports. Table 1-21 lists the attributes.



ug382_c1_24_120809

Figure 1-34: **BUFIO2_2CLK Primitive**

Table 1-20: **BUFIO2_2CLK Port List and Definitions**

| Port Name | Type | Definition |
|--------------|--------|---|
| I | Input | GCLK clock input. |
| IB | Input | Inverted GCLK clock input. |
| IOCLK | Output | I/O clock network output. Connects to IODDR2 (C0 or C1), IOSERDES2 (CLK0 or CLK1) or IODELAY2 (IOCLK0, IOCLK1). |
| DIVCLK | Output | Divided clock output. Connects to BUFG, PLL_BASE (CLKIN), DCM (CLKIN), and DCM_CLKGEN (CLKIN). |
| SERDESSTROBE | Output | I/O clock network output used to drive IOSERDES2 (IOCE). |

Table 1-21: **BUFIO2_2CLK Attributes**

| Attribute Name | Description | Possible Values | Default Value |
|----------------|---|------------------|---------------|
| DIVIDE | Data rate setting used for setting DIVCLK and SERDESSTROBE divider divide-by value. $F_{DIVCLK} = (2 * F_{IN}) / DIVIDE$ | 3, 4, 5, 6, 7, 8 | 3 |

As shown Figure 1-35, BUFIO2_2CLK can be connected to a differential GCLK input.

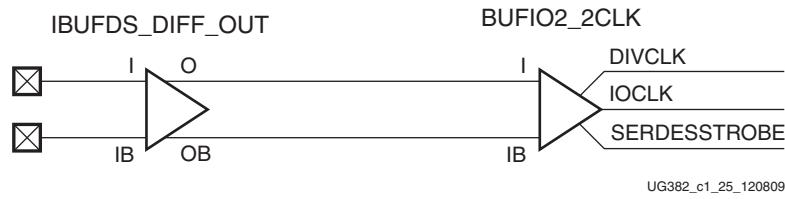


Figure 1-35: BUFIO2_2CLK Driven by Differential GCLK Clock Input

Figure 1-36 shows the timing waveforms when combined with a second BUFIO2 when used for ISERDES2 (DATA_PATH = DDR) or OSERDES2 (DATA_PATH_OQ = DDR).

Double Data Rate Clock

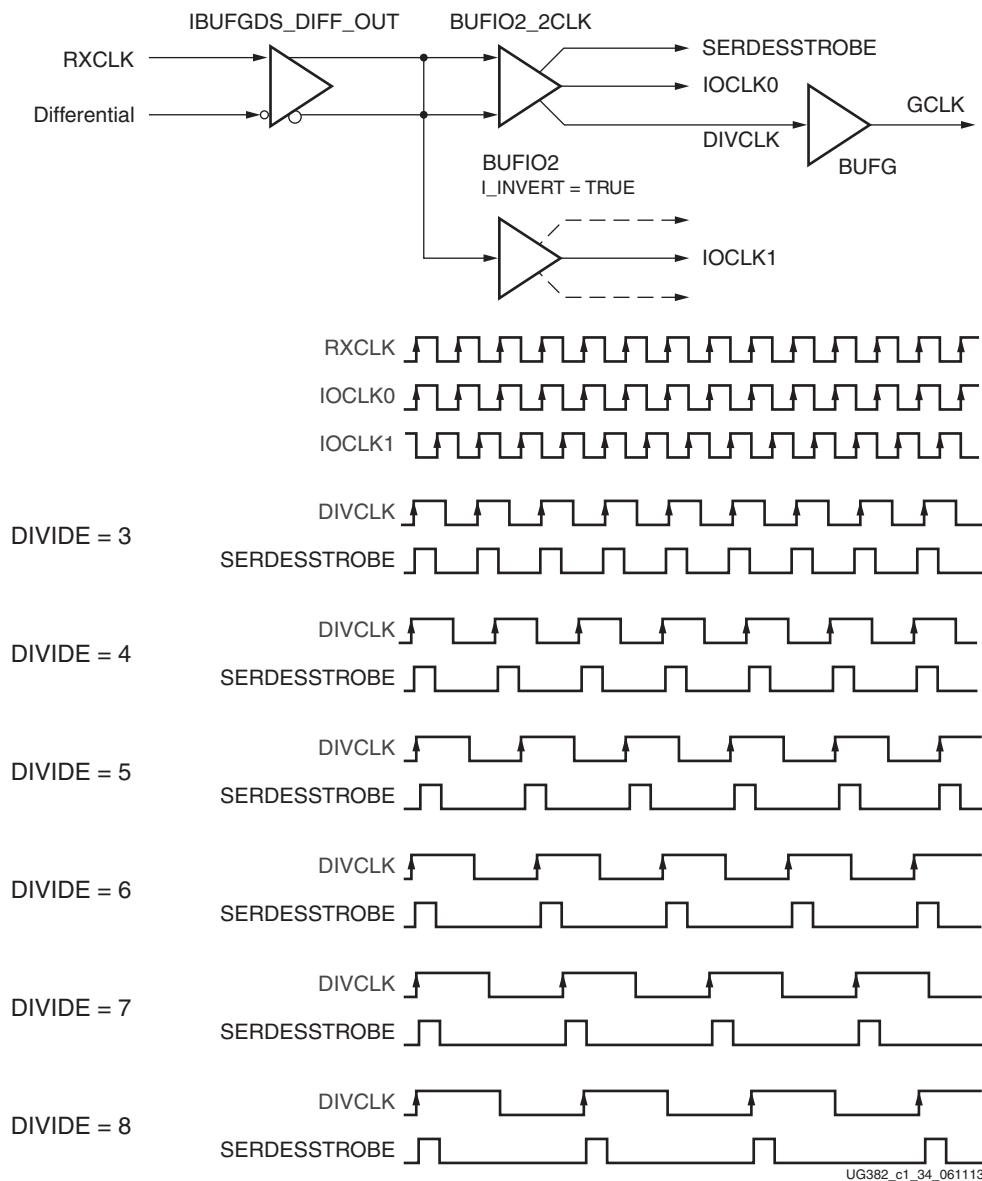


Figure 1-36: BUFIO2_2CLK Double Data Rate Clock

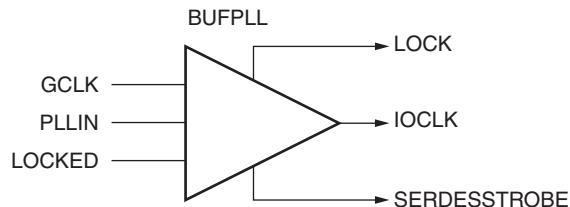
BUFPLL

The BUFPLL (Figure 1-37) is intended for high-speed I/O routing to generate clocks and strobe pulses for the ISERDES2 (SDR) and OSERDES2 (SDR) primitives. The **BUFPLL_MCB** contains two BUFPLLs within the same bank. As a result, the BUFPLL_MCB and BUFPLL cannot be used at the same time.

When using the PLL clock outputs, CLKOUT0 or CLKOUT1 must directly connect to the PLLIN clock input. See [Figure 1-16, Example 3: Basic PLL ISERDES2 \(SDR\)](#).

The BUFPLL can be used with the BUFINO2FB to create the PLL feedback clock. When used with the BUFINO2FB, the IOCLK output must be connected to the BUFINO2FB feedback buffer. Additional information on PLL feedback is discussed in [Aligning PLL using CLK_FEEDBACK and BUFINO2FB in Chapter 3](#).

The BUFPLL also aligns the SERDESSTROBE to the IOCLK. To align the SERDESSTROBE, the GCLK and the LOCKED inputs must be connected. The IOCLK output is a buffered version of the input clock. The LOCK output serves the exact same function as the PLL LOCKED signal except it does not go HIGH until the PLL has locked and the BUFPLL has aligned the SERDESSTROBE signal correctly. For BUFPLLs located in BANK0 (top) and BANK2 (bottom), the BUFPLL LOCKED inputs can connect to any of the PLLs located within the same half of the device. An additional multiplexer allows one additional PLL from the opposite half of the device. This additional multiplexer allows BUFPLLs located in BANK0 to be driven by either one of the PLLs placed in the bottom half of the device. Similarly, BUFPLLs located in BANK2 can be driven by one PLL located within the top half of the device. [Table 1-22](#) lists the BUFPLL ports, [Table 1-23](#) lists the attributes.



ug382_c1_27_120809

Figure 1-37: BUFPLL Primitive

Word synchronization is achieved in the BUFPLL by using three signals from the PLL: the high-speed clock, the low-speed clock, and the LOCKED signal. The BUFPLL uses these signals to set the position of the SERDESSTROBE signal relative to the low-speed clock. Due to the varying placements of the PLL and BUFPLL, the initial SERDESSTROBE can vary up to one bit period. The SERDESSTROBE signal controls when data is moved from the high-speed domain to the low-speed domain in either the ISERDES2 or the OSERDES2. This relationship is established once the LOCKED signal transitions from Low to High and is maintained by a simple counter from that point onwards. The BUFPLL can therefore drive all the input or output SERDES in a single edge of the device, all of which will have the same word synchronization.

Table 1-22: BUFPLL Port List and Definitions

| Port Name | Type | Definition |
|--------------|--------|--|
| PLLIN | Input | Clock input from PLL (CLKOUT0, CLKOUT1) directly connected to the PLL. Banks 1, 3, 4, and 5 can optionally be driven by a BUFG (O) when using ENABLE_SYNC (FALSE). |
| GCLK | Input | Clock input from BUFG or GCLK. The GCLK frequency must match the expected SERDESSTROBE frequency. $F_{GCLK} = F_{PLLIN}/\text{DIVIDE}$ |
| LOCKED | Input | LOCKED signal from PLL. |
| IOCLK | Output | I/O clock network output. Connects to IOSERDES2 (CLK0), BUFIO2FB (I), or IODELAY2 (IOCLK0, IOCLK1). |
| SERDESSTROBE | Output | I/O clock network output used to drive IOSERDES2 (IOCE). |
| LOCK | Output | Synchronized LOCK output directly connected to the PLL. |

Table 1-23: BUFPLL Attributes

| Attribute Name | Description | Possible Values | Default Value |
|----------------|---|--|------------------|
| DIVIDE | Sets the PLLIN divider divide-by value for SERDESSTROBE | 1 ⁽¹⁾ , 2, 3, 4, 5, 6, 7, 8 | 1 ⁽¹⁾ |
| ENABLE_SYNC | When set, synchronizes SERDESSTROBE to GCLK input. When PLLIN is driven by BUFG, set ENABLE_SYNC = FALSE. | TRUE, FALSE | TRUE |

Notes:

- See [UG381, Spartan-6 FPGA SelectIO Resources User Guide](#) for requirements on SelectIO logic resources.

Figure 1-38 shows the timing waveforms for the BUFPLL when used for ISERDES2 (DATA_PATH = SDR) or OSERDES2 (DATA_PATH_OQ = SDR). For the BUFPLL, SERDESSTROBE is aligned to the falling edge of GCLK.

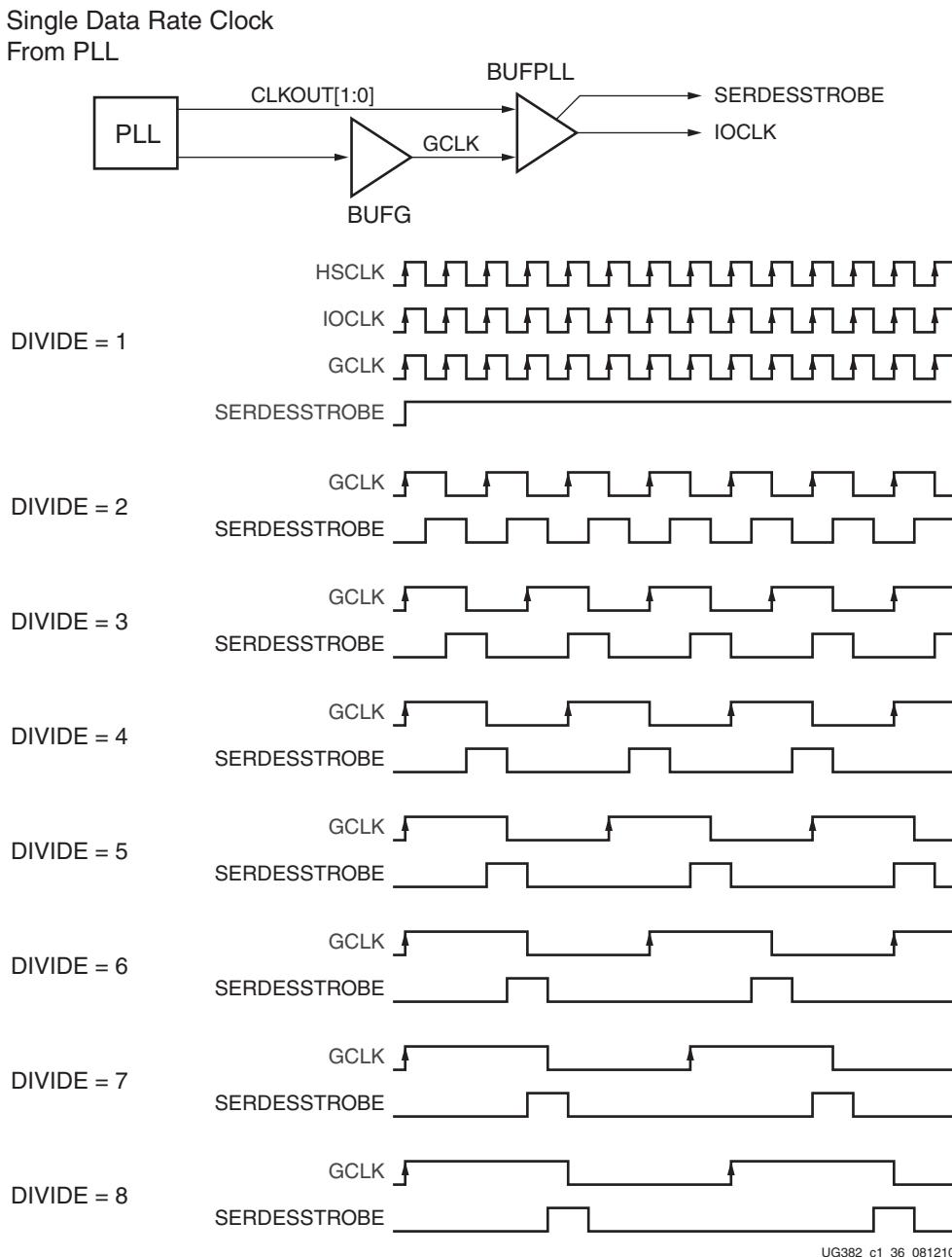


Figure 1-38: BUFPLL Single Data Rate Clock

BUFPLL_MCB

The BUFPLL_MCB primitive is dedicated to supporting the integrated memory controller blocks available in Spartan-6 FPGAs. The BUFPLL_MCB contains two BUFPLLs within the same bank. As a result, the BUFPLL_MCB and BUFPLL cannot be used at the same time.

[Table 1-24](#) lists all the BUFPLL_MCB port names and [Table 1-25](#) lists all the attributes. The BUFPLL_MCB primitive contains two buffers that operate independently when the LOCK_SRC attribute is set to INDEPENDENT. This circuitry synchronizes the SERDESSTROBE0 and SERDESSTROBE1. The synchronization is enabled by setting the LOCK_SRC to LOCK_TO_0 or LOCK_TO_1. When LOCK_SRC is set to LOCK_TO_1, the SERDESSTROBE0 follows the SERDESSTROBE1 by one clock cycle. Similarly, when LOCK_SRC is set to LOCK_TO_0, the SERDESSTROBE1 follows the SERDESSTROBE0 by one clock cycle. A detailed use case discussion is available in the clocking section in Chapter 3 of [UG388, Spartan-6 FPGA Memory Controller User Guide](#).

To ensure correct synchronization of the SERDESSTROBE0 and SERDESSTROBE1, the GCLK and the LOCKED inputs must be connected to the PLL. The LOCK output serves the exact same function as the PLL LOCKED signal except it does not transition High until the PLL is locked and the BUFPLL is correctly aligned to the SERDESSTROBE signal.

Table 1-24: BUFPLL_MCB Port List and Definitions

| Port Name | Type | Definition |
|---------------|--------|--|
| PLLIN0 | Input | Clock input from PLL (CLKOUT0 or CLKOUT1). Banks 1, 3, 4, and 5 can be driven by BUFG (O). |
| PLLIN1 | Input | Clock input from PLL (CLKOUT0 or CLKOUT1). Banks 1, 3, 4, and 5 can be driven by BUFG (O). |
| GCLK | Input | Clock input from BUFG or GCLK. |
| LOCKED | Input | LOCKED signal from PLL. |
| IOCLK0 | Output | I/O clock network output. Connects to IOSERDES2 (CLK0) or BUFI02FB (I). |
| IOCLK1 | Output | I/O clock network output. Connects to IOSERDES2 (CLK0) or BUFI02FB (I). |
| SERDESSTROBE0 | Output | I/O clock network output used to drive IOSERDES2 (IOCE). |
| SERDESSTROBE1 | Output | I/O clock network output used to drive IOSERDES2 (IOCE). |
| LOCK | Output | Synchronized LOCK output. |

Table 1-25: BUFPLL_MCB Attributes

| Attribute Name | Description | Possible Values | Default Value |
|----------------|---|--|---------------|
| LOCK_SRC | Sets whether PLLIN0 and PLLIN1 are independent or locked. If locked, the signal it is locked to PLLIN0 or PLLIN1. | INDEPENDENT, LOCK_TO_0, LOCK_TO_1 | LOCK_TO_0 |
| DIVIDE | Sets the PLLIN0 and PLLIN1 divider divide-by value for SERDESSTROBE0 and SERDESSTROBE1. | 1 ⁽¹⁾ , 2, 3, 4, 5, 6, 7, 8 | 2 |

Table 1-25: BUFPLL_MCB Attributes (Cont'd)

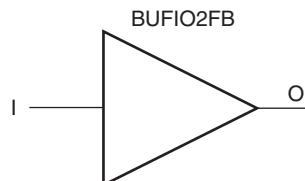
| Attribute Name | Description | Possible Values | Default Value |
|----------------|---|-----------------|---------------|
| ENABLE_SYNC | When set, synchronizes SERDESSTROBE to GCLK input. Only clear if BUFPLL is driven by BUFG | TRUE, FALSE | TRUE |

Notes:

- See [UG381](#), Spartan-6 FPGA SelectIO Resources User Guide for requirements on SelectIO logic resources.

BUFI02FB

The BUFI02FB is a simple buffer that is used to define the feedback paths for the PLL and DCM. It has only one input and one output (Figure 1-39). Table 1-26 lists the BUFI02FB ports. As listed in Table 1-27, when the DIVIDE_BYPASS is set to TRUE then the delays are equivalent to the BUFI02 bypass delays, when set to FALSE the delays will be similar to the DIVCLK output of a BUFI02, keeping the outputs of the BUFI02 and BUFI02FB phase aligned.



ug382_c1_28_120809

Figure 1-39: BUFI02FB Primitive

Table 1-26: BUFI02FB Port List and Definitions

| Port Name | Type | Definition |
|-----------|--------|---|
| I | Input | Feedback clock input. Can be driven by BUFI02 (IOCLK), BUFPLL (IOCLK), IODELAY2 (DATAOUT), GTPCLKFBWEST, GTPCLKFBEAST, or BUFG (O). |
| O | Output | Output feedback clock |

Table 1-27: BUFI02FB Attributes

| Attribute Name | Description | Possible Values | Default Value |
|----------------|---|-----------------|---------------|
| DIVIDE_BYPASS | When FALSE, DIVCLK outputs source from divider. The CLKDIV_DIVIDE must be set to 1. When TRUE, DIVCLK outputs source from input I, bypassing the divider. | TRUE, FALSE | TRUE |

The BUFI02FB buffer matches the clock routing delay between a CMT (PLL or DCM) reference input CLKIN and feedback CLKFB when used as shown in Figure 1-40. The ISE Design Suite automatically inserts matching BUFI02FB and BUFI02 buffers when the CMT feedback path is used.

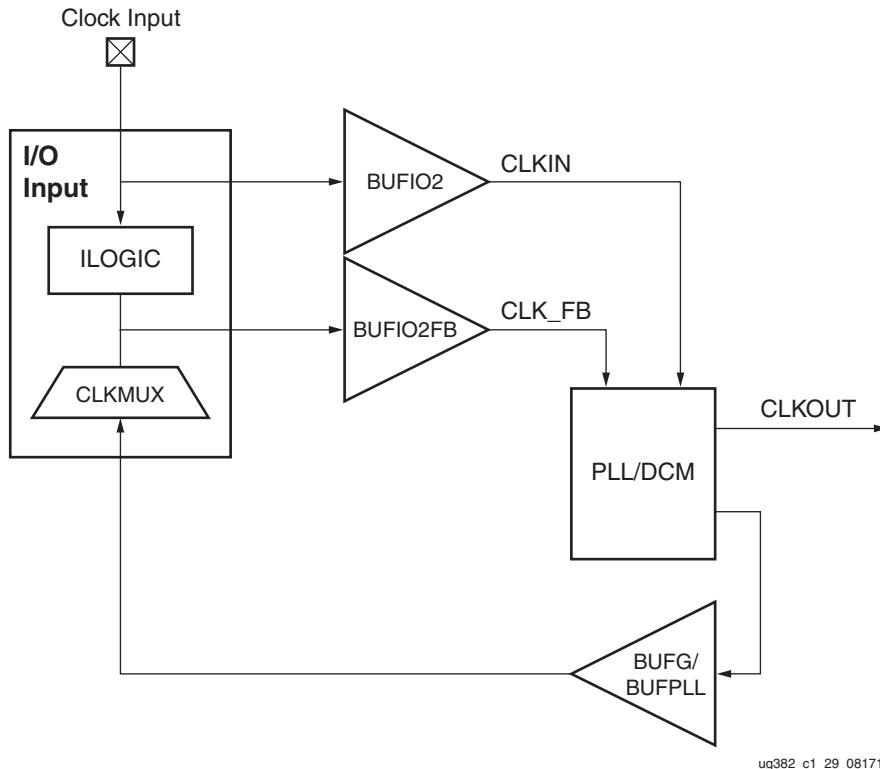


Figure 1-40: BUFIO2FB Buffer Matches Clock Routing Delay

DIVIDE_BYPASS allows the BUFIO2 and BUFIO2FB to optionally divide a clock input using the DIVIDE of the BUFIO2, as shown in [Figure 1-41](#). To ensure the delays are correctly matched, the DIVIDE_BYPASS for BUFIO2FB must match the DIVIDE_BYPASS for BUFIO2.

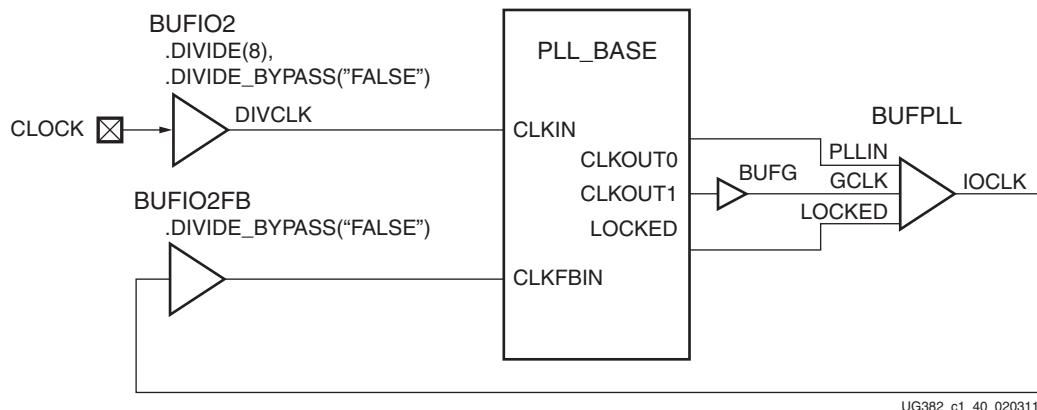


Figure 1-41: Dividing Clocks Using BUFIO2 and BUFIO2FB

Clock Management Technology

Clock Management Summary

The Spartan-6 FPGA Clock Management Tiles (CMTs) provide very flexible, high-performance clocking. The Spartan-6 FPGA CMT blocks (Figure 2-1) are located in the center column along the vertical global clock tree. Each CMT block contains two DCMs and one PLL.

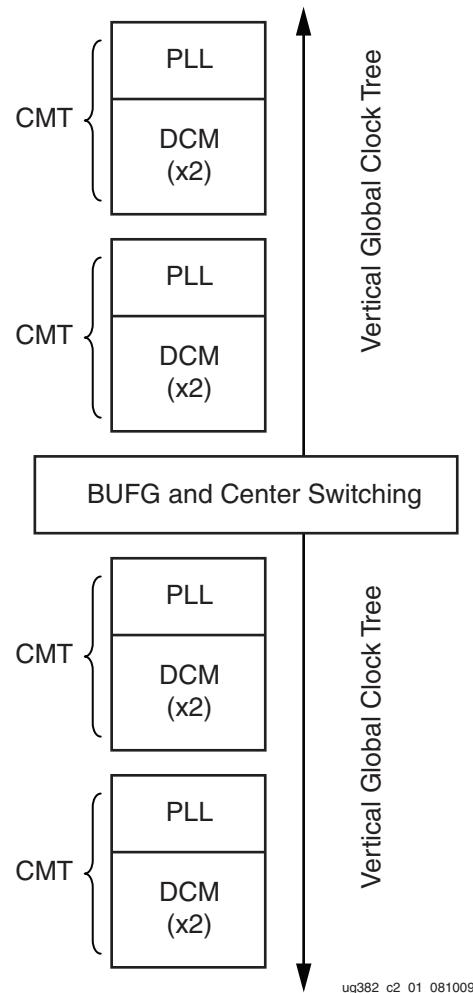


Figure 2-1: Spartan-6 FPGA CMT Location

Table 2-1 summarizes the availability of CMTs, DCMs, and PLLs in each Spartan-6 device.

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

| Device | Number of CMTs | Number of DCMs | Number of PLLs |
|------------|----------------|----------------|----------------|
| XC6SLX4 | 2 | 4 | 2 |
| XC6SLX9 | 2 | 4 | 2 |
| XC6SLX16 | 2 | 4 | 2 |
| XC6SLX25 | 2 | 4 | 2 |
| XC6SLX25T | 2 | 4 | 2 |
| XC6SLX45 | 4 | 8 | 4 |
| XC6SLX45T | 4 | 8 | 4 |
| XC6SLX75 | 6 | 12 | 6 |
| XC6SLX75T | 6 | 12 | 6 |
| XC6SLX100 | 6 | 12 | 6 |
| XC6SLX100T | 6 | 12 | 6 |
| XC6SLX150 | 6 | 12 | 6 |
| XC6SLX150T | 6 | 12 | 6 |

To minimize clock skew, use global clock buffers for the clock outputs from the CMT. When clock buffers are limited, CMT clock outputs can optionally be used without a global clock buffer, but all logic is required to be placed within a clock region. Each of these clock regions is 16 CLBs tall, contains up to four 18 Kb block RAMs, and up to four DSP48A1 slices.

DCM Summary

Digital Clock Managers (DCMs) provide advanced clocking capabilities to Spartan-6 FPGA applications. Primarily, DCMs eliminate clock skew, thereby improving system performance. Similarly, a DCM optionally phase shifts the clock output to delay the incoming clock by a fraction of the clock period. DCMs optionally multiply or divide the incoming clock frequency to synthesize a new clock frequency. The DCMs integrate directly with the global low-skew clock distribution network.

DCM Introduction

DCMs integrate advanced clocking capabilities directly into the global clock distribution network. Consequently, DCMs solve a variety of common clocking issues, especially in high-performance, high-frequency applications:

- Eliminate clock skew, either within the device or to external components, to improve overall system performance and to eliminate clock distribution delays.
- Phase shift a clock signal, either by a fixed fraction of a clock period or by incremental amounts.
- Multiply or divide an incoming clock frequency or synthesize a completely new frequency by a mixture of static or dynamic clock multiplication and division.
- Condition a clock, ensuring a clean output clock with a 50% duty cycle.
- Mirror, forward, or rebuffer a clock signal, often to deskew and convert the incoming clock signal to a different I/O standard. For example, forwarding and converting an incoming LVTTL clock to LVDS.
- Clock input jitter filtering
- Free-running oscillator
- Spread-spectrum clock generation

Table 2-2: DCM Features and Capabilities

| Feature | Description | DCM Signals |
|---|---|--|
| DCMs per device | Four to 12 DCMs, depending on device size. See Table 2-1 . | All |
| Clock input sources | GCLK input, BUFG output, cascaded DCM or PLL output (within the same CMT). General interconnect is allowed but not recommended for optimal performance. | CLKIN |
| Frequency synthesizer output | Multiply CLKIN by the fraction (M/D) where M = {2..256}, D = {1..256} when using the DCM_CLKGEN primitive | CLKFX, CLKFX180 |
| Clock divider output | Divide CLKIN by 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 | CLKDV |
| Clock doubler output | Multiply CLKIN frequency by 2 | CLK2X, CLK2X180 |
| Clock conditioning, duty-cycle correction | Always provided on most outputs. | All |
| Quadrant phase-shift outputs | 0° (no phase shift), 90° (¼ period), 180° (½ period), 270° (¾ period) | CLK0, CLK90, CLK180, CLK270 |
| Half-period phase-shift outputs | Output pairs with 0° and 180° phase shift, ideal for DDR applications | CLK0, CLK180, CLK2X, CLK2X180, CLKFX, CLKFX180 |
| Variable phase-shifting | Allows DCM clock outputs to adjust phase shift during operation | PSEN, PSINCDEC, PSCLK, PSDONE |
| General purpose DCM operation indicators | Number of DCM clock outputs connected to general-purpose interconnect | STATUS, LOCKED |

Compatibility and Comparison with Other Xilinx FPGA Families

The Spartan-6 FPGA includes a very similar DCM design to the Spartan-3E and Extended Spartan-3A family. There are, however, important DCM design differences between the Spartan-6 family and Virtex-5 FPGAs.

Similar to the Spartan-3E and Extended Spartan-3A families, the Spartan-6 FPGA DCMs automatically determine their operating range and are not limited to either a Low or High operating frequency range. Spartan-6 FPGAs implement variable-phase shift operations differently. Spartan-6 FPGAs now include the DCM_CLKGEN primitive to support more advanced features such as jitter reduction, dynamic programming of frequency multiplication, dynamic programming of frequency division, and generation of spread-spectrum clocks. [Table 2-3](#) compares various DCM functions in Xilinx FPGAs.

Table 2-3: DCM Differences between Xilinx FPGAs

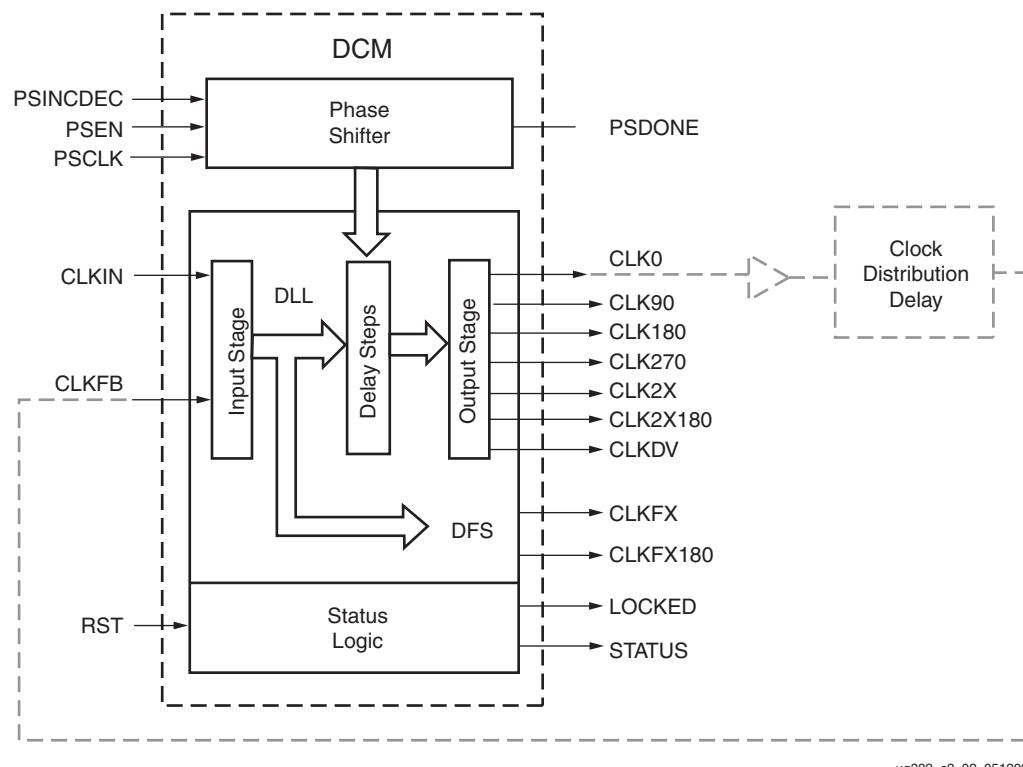
| Function | Virtex-5 FPGAs ⁽¹⁾ | Spartan-3E and Extended Spartan-3A FPGAs | Spartan-6 FPGAs |
|--|--|--|---|
| Design primitive | DCM_BASE DCM_ADV | DCM_SP | DCM_SP DCM_CLKGEN |
| Distinct DLL operating frequency ranges | Two: Low and High | One | One |
| Distinct DFS operating frequency ranges | Two: Low and High | One | One |
| Variable phase-shift increment or decrement unit | $\frac{1}{256}$ th of CLKIN period (degrees) | DCM_DELAY_STEP between 15 to 35 ps (time) | DCM_DELAY_STEP See the Spartan-6 FPGA data sheet |
| DCM V _{CCAUX} voltage supply | 2.5V | 2.5V or 3.3V | 2.5V or 3.3V |
| Jitter reduction with expense of phase alignment | No | No | Yes |
| Dynamic programming of frequency multiplication and division | No | No | Yes |
| Generation of spread-spectrum clocks | No | No | Yes |

Notes:

- When converting legacy designs using the Virtex-5 FPGA primitives DCM_ADV or DCM_BASE, use the Spartan-6 FPGA DCM_SP or DCM_CLKGEN primitives instead.

DCM Functional Overview

The DCM consists of four distinct functions that can operate independently or in tandem. [Figure 2-2](#) shows a simplified block diagram of a DCM.



ug382_c2_02_051209

Figure 2-2: DCM Functional Block Diagram

Delay-Locked Loop

The Delay-Locked Loop (DLL) provides an on-chip digital deskew circuit that effectively generates clock output signals with a net zero delay. The deskew circuit compensates for the delay on the clock routing network by monitoring an output clock, from either the CLK0 or the CLK2X outputs. The DLL effectively eliminates delay from the external clock input port to the individual clock loads within the device. The well-buffered global network minimizes the clock skew on the network caused by loading differences.

To accurately deskew the input routing, the BUFI02FB buffer must be used as shown in [Figure 1-40, page 56](#). The BUFI02FB is matched to the primary BUFI02 buffer limiting the deskewing to a single DCM.

The input signals to the DLL unit are CLKIN and CLKFB. The output signals from the DLL are CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, and CLKDV.

The DLL unit generates the outputs for the clock doubler (CLK2X, CLK2X180), the clock divider (CLKDV) and the quadrant phase-shift functions.

Skew Adjustment

DCM_SP allows for different skew adjustments to help with different interfaces. The feedback circuitry adjusts for placement and routing differences based on the location of the DCM and clock buffers. See [Figure 1-18](#) for an example of the recommended feedback structure. DESKEW_ADJUST allows for a small, fixed amount of delay adjustment.

System synchronous and source synchronous are descriptions of different types of interfaces. [UG612, Timing Closure User Guide](#) contains descriptions of how to constrain these types of interfaces depending on how the clock and data are aligned. Additional source synchronous interfaces are additionally covered in [XAPP1064, Source-Synchronous Serialization and Deserialization \(up to 1050 Mb/s\)](#).

System synchronous interfaces ([Figure 2-3](#)) are the more commonly used type of interface. In this interface, a small fixed amount of skew adjustment is used to reduce the hold times for the input registers. The system synchronous interface advances the clock internal to the FPGA to help ensure zero hold times. Setting DESKEW_ADJUST = SYSTEM_SYNCHRONOUS shifts the clock as shown in [Figure 2-3](#) to help ensure zero hold times.

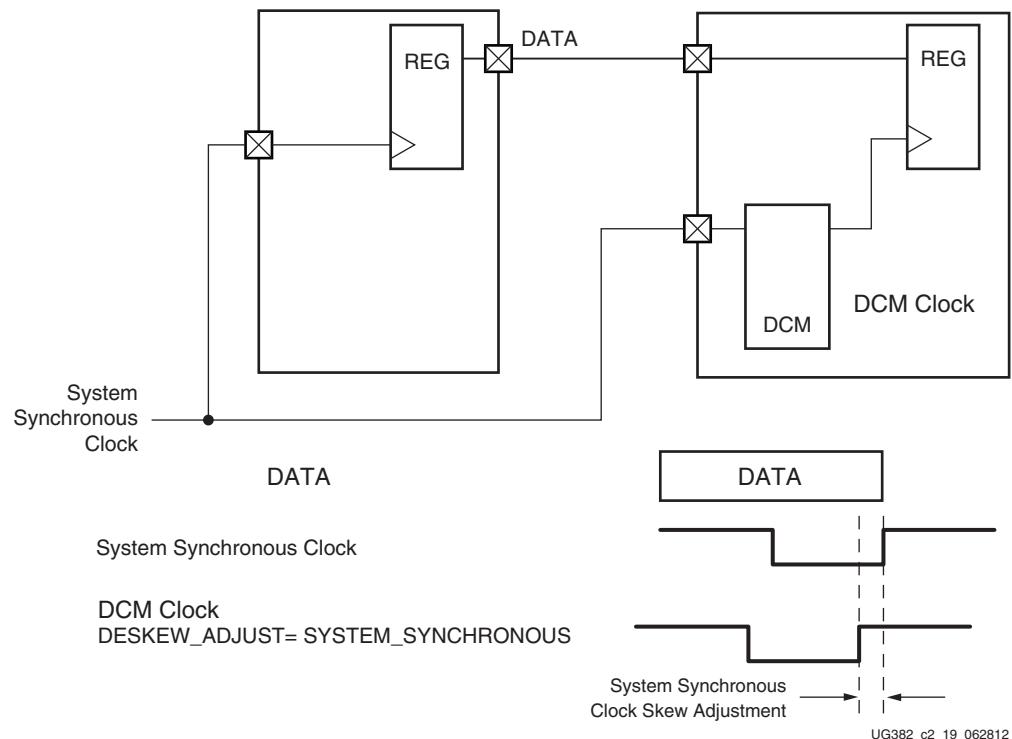


Figure 2-3: System Synchronous

When using the DCM, DESKEW_ADJUST[SYSTEM_SYNCHRONOUS] has no effect on cascaded DCMs, DCMs using external feedback, or DCMs where the clock source does not come from a global clock input routed through the BUFIO2.

As data rates increase, interfaces rely on the clock forwarded with the data. This is commonly called a source synchronous interface. The clock can be either center aligned or edge aligned with the data window. For source synchronous interfaces with the clock center aligned as shown in [Figure 2-4](#), DESKEW_ADJUST(SOURCE_SYNCHRONOUS)

uses the shortest skew adjustment to balance the setup time with the hold time for the input register.

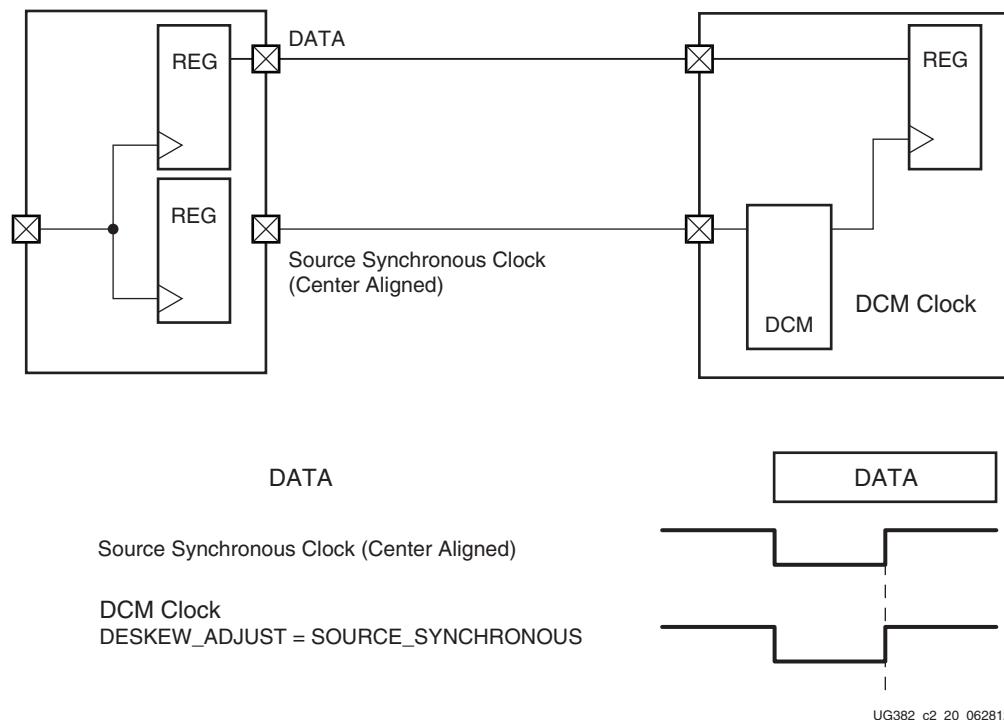


Figure 2-4: Source Synchronous Center Aligned

Some source synchronous interfaces use an edge-aligned clock source as shown in [Figure 2-5](#). Use DESKEW_ADJUST[SOURCE_SYNCHRONOUS] to again balance the setup time with the hold time. Phase shifting of the DCM's clock output can then be used to adjust the phase based on the data rate being used. The DCM clock is shown without phase shift (PHASE_SHIFT[0]) and with the corrected phase shift (PHASE_SHIFT[126]) for the given example.

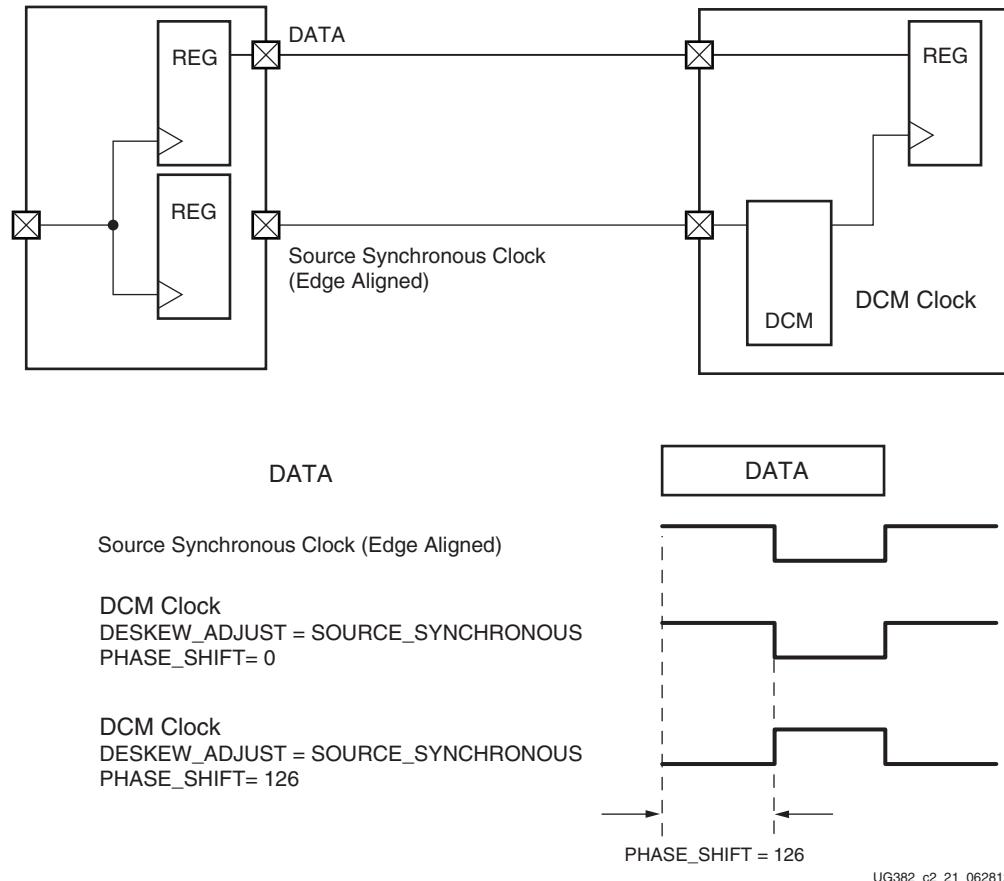


Figure 2-5: Source Synchronous Edge Aligned

Digital Frequency Synthesizer

The Digital Frequency Synthesizer (DFS) provides a wide and flexible range of output frequencies based on the ratio of two user-defined integers, a multiplier (CLKFX_MULTIPLY) and a divisor (CLKFX_DIVIDE). The output frequency is derived from the input clock (CLKIN) by simultaneous frequency division and multiplication. The DFS feature can be used in conjunction with or separately from the DLL feature of the DCM. If the DLL is not used, then there is no phase relationship between CLKIN and the DFS outputs.

The DFS unit generates the frequency synthesizer (CLKFX and CLKFX180) outputs.

Phase Shift

The DCM provides coarse- and fine-grained phase shifting. For coarse-grained phase control, the CLK0, CLK90, CLK180, and CLK270 outputs are each phase-shifted by $\frac{1}{4}$ of the input clock period relative to each other. Similarly, CLK2X180 and CLKFX180 provide a 180° coarse phase shift of CLK2X and CLKFX, respectively.

For fine-grained phase control, the DCM can optionally phase shift all of its clock outputs using the phase shift (PS) controls. The PS controls the phase relations of the DCM clock outputs to the CLKIN input. A phase shift offset, using the PHASE_SHIFT attribute, can be used in either fixed phase-shift (CLKOUT_PHASE_SHIFT=FIXED) or variable phase shift

modes (CLKOUT_PHASE_SHIFT=VARIABLE). For phase shifting to work, the DLL must use an appropriate deskew circuit with either CLK0 or CLK2X being used as a feedback clock connected to CLKFB.

Fixed Phase Shift

In the Fixed Phase Shift mode, the phase of all nine DCM clock output signals are shifted by a fixed fraction of the input clock period. The fixed phase shift value is set at design time. The size of each phase shift unit is 1/256th of the CLKIN clock period or 1.40625° per step (Figure 2-6). PHASE_SHIFT can be any integer from -255 to 255.

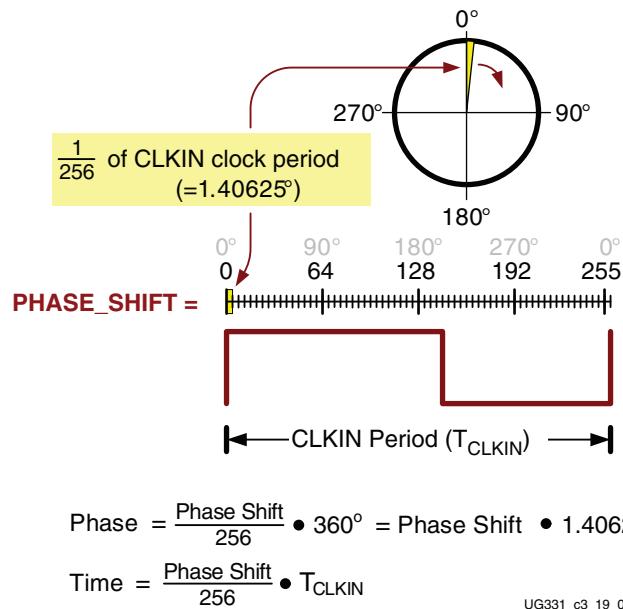


Figure 2-6: **Each PHASE_SHIFT Unit is 1/256th of the CLKIN Period**

Variable Phase Shift

In Variable Phase Shift mode, the initial skew or phase shift is still controlled by the PHASE_SHIFT attribute during configuration, just as it is for Fixed Phase Shift mode. However, the variable phase shift further adjusts the phase shift location after the DCM's LOCKED output goes High. Variable phase shifting changes by one DCM_DELAY_STEP at a time. See the *Spartan-6 FPGA Data Sheet* for DCM_DELAY_STEP values.

When using variable phase shifting, the user is given control of the phase shift adjustments that would otherwise be used by the DCM. As a result, the DCM's phase alignment should be monitored and updated frequently enough to compensate for temperature and voltage variations within the system.

Conversely, the FIXED phase shift setting uses phase shifts to continually update the phase alignment to correct temperature and voltage variations.

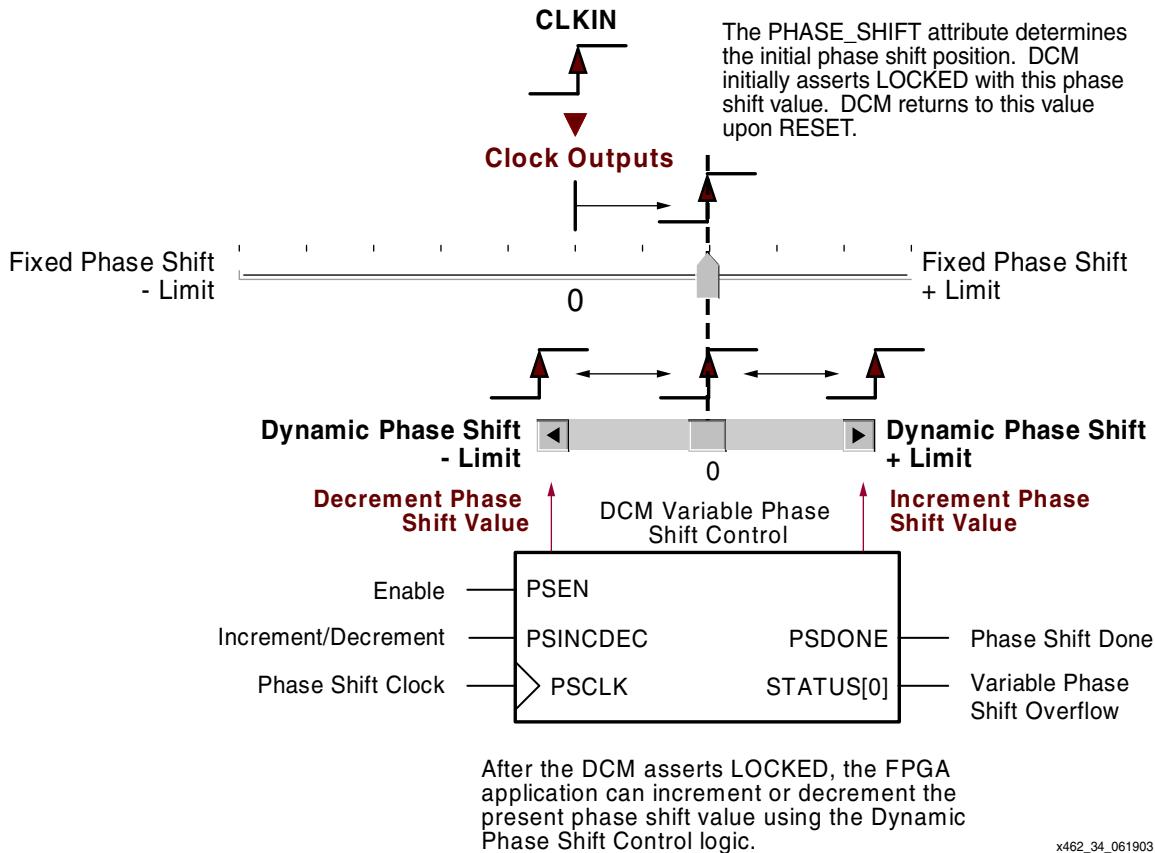


Figure 2-7: Variable Phase Shift With PHASE_SHIFT ≠ 0

The total resulting phase shift is the sum of the initial fixed phase shift plus any variable phase shift adjustments. [Example 1](#) and [Example 2](#) show how the fixed and variable phase shifting adjust the phase shift.

In [Example 1](#), a 100 MHz clock is phase shifted by 90° by setting PHASE_SHIFT = 64. The variable phase shift is then used to further phase shift the clock for a maximum phase shift of 2.900 ns after 10 phase shifts.

Example 1

FCLKIN = 100 MHz

TCLKIN = 10 ns

PHASE_SHIFT = 64

PSINCDEC = High

Initial phase in degrees = 90

Initial phase as a delay = 2.5 ns

Maximum phase after 1 phase shift clock = $2.5 + 1 * \text{DCM_DELAY_STEP} = 2.540 \text{ ns}$

Maximum phase after 10 phase shift clock = $2.5 + 10 * \text{DCM_DELAY_STEP} = 2.900 \text{ ns}$

In [Example 2](#), the same 90° phase shift is applied to a 50 MHz clock. In this example, a 90° phase shift results in a phase shift of 5.0 ns. After 10 variable phase shifts, the amount of

delay associated with the variable phase shift continues to be 0.400 ns. However, given the 90° phase shift, the overall phase shift is now 5.400 ns.

Example 2

FCLKIN = 50 MHz

TCLKIN = 20 ns

PHASE_SHIFT = 64

PSINCDEC = High

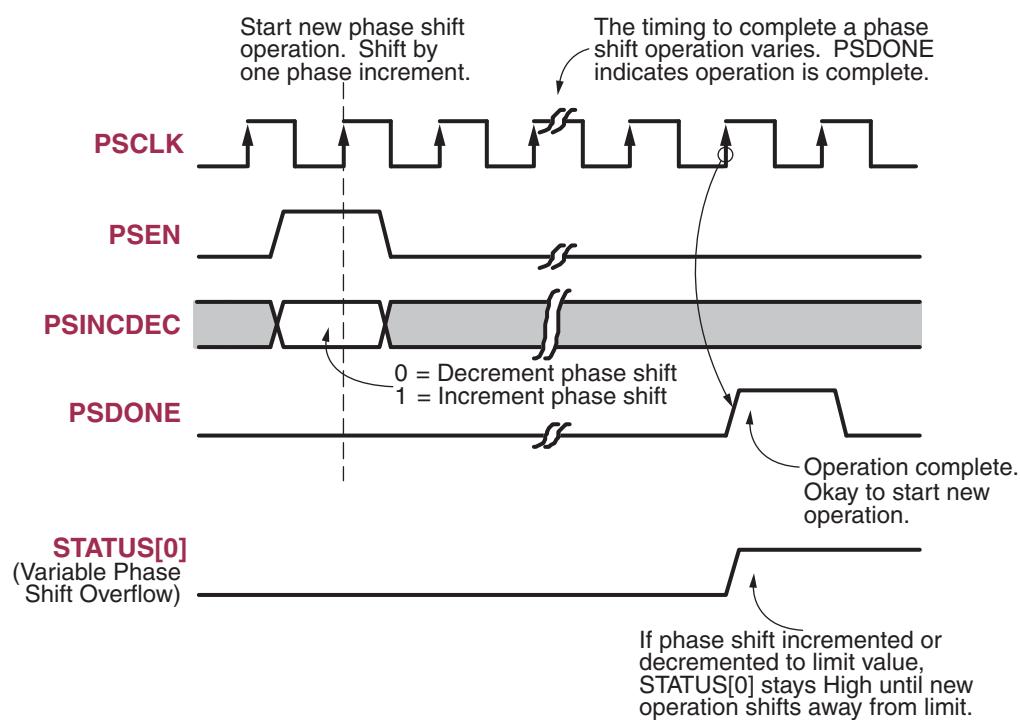
Initial phase in degrees = 90

Initial phase as a delay = 5.0 ns

Maximum phase after 1 phase shift clock = $5 + 0.040 = 5.040$ ns

Maximum phase after 10 phase shift clock = $5 + 10 * 0.040 = 5.400$ ns

The phase shift control inputs adjust the current phase shift value, as shown in Figure 2-8. The rising edge of PSCLK synchronizes all Variable Phase Shift operations. A valid operation starts by asserting the PSEN enable input for one and only one PSCLK clock period. Asserting PSEN for more than one rising PSCLK clock edge might cause undesired behavior.



UG331_c3_29_100509

Figure 2-8: Dynamic Fine Phase Shift Control Interface

The value on the PSINCDEC increment/decrement control input determines the phase shift direction. When PSINCDEC is High, the present Variable Phase Shift value is incremented by one unit. Similarly, when PSINCDEC is Low, the present Variable Phase Shift value is decremented by one unit. The actual phase shift operation timing varies and the operation completes when the DCM asserts the PSDONE output High for a single

PSCLK clock period. Between enabling PSEN until PSDONE is asserted, the DCM output clocks slide, bit by bit, from their original phase shift value to their new phase shift value. During this time, the DCM remains locked on the incoming clock and continues to assert its LOCKED output.

PSDONE indicates that the PS unit completed the previous adjustment and is now ready for the next request. The PHASE_SHIFT attribute value sets the initial phase shift location, established after FPGA configuration. If the DCM is reset, the PHASE_SHIFT value reverts to its initial configuration value.

Variable phase shifting is performed using delay elements. As such, there is a physical maximum for the number of steps, depending on the CLKIN input period (TCLKIN), as shown in [Table 2-4](#).

Table 2-4: Maximum Number of DCM Delay Steps for Variable Phase Shift

| CLKIN Frequency (MHz) | CLKIN period TCLKIN (ns) | Maximum Number of DCM Delay Steps |
|-----------------------|--------------------------|---|
| < 60 | > 16.67 | $\pm[\text{INTEGER}(10^*(\text{TCLKIN} - 3 \text{ ns}))]$ |
| ≥ 60 | ≤ 16.67 | $\pm[\text{INTEGER}(15^*(\text{TCLKIN} - 3 \text{ ns}))]$ |

For example, assume that the CLKIN clock entering the DCM is 100 MHz, which equates to a clock period of TCLKIN = 10 ns. Using the equation in [Table 2-4](#), the Variable Phase Shifter is limited to phase shift operations of ± 105 steps. This equates to a maximum variable phase shift measured in time of up to ± 1.05 ns to ± 4.2 ns. Measured in degrees, this equates to a maximum between $\pm 37.8^\circ$ and $\pm 151.2^\circ$.

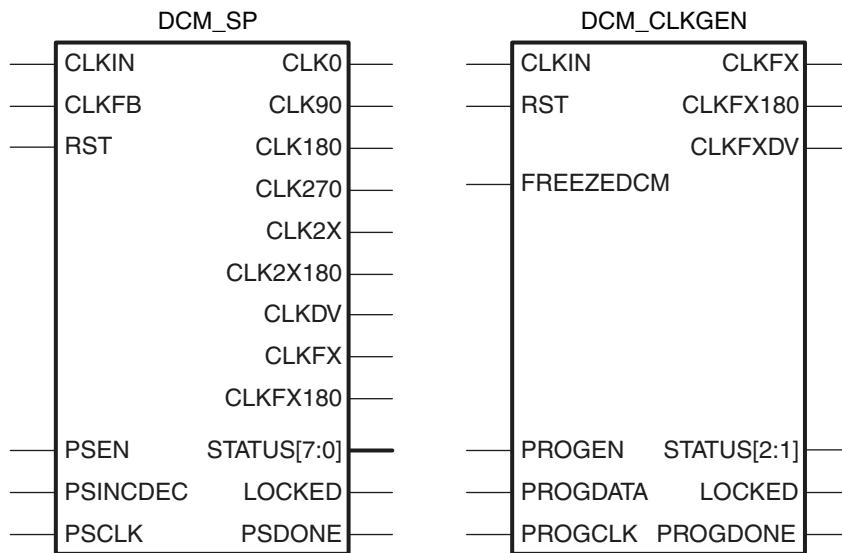
Status Logic

The status logic indicates the current state of the DCM via the LOCKED and STATUS output signals. The LOCKED output signal indicates whether the DCM outputs are in phase with the CLKIN input. The STATUS output signals indicate the state of the DLL and PS operations.

The RST input signal resets the DCM logic and returns it to its post-configuration state. Likewise, a reset forces the DCM to reacquire and lock to the CLKIN input.

DCM Primitives

The DCM design primitives (Figure 2-9), DCM_SP and DCM_CLKGEN, represent all the features within the DCM. STATUS[7:0] is used for simulation when using DCM_CLKGEN for the lower-power Spartan-6 devices (-1L). See [RST Input Behavior, page 81](#) for more information on the low-power reset circuit.



UG382_c2_03_061109

Figure 2-9: DCM Primitives

Each DCM connection port, attributes, properties, and constraints are summarized in this section.

DCM_SP Primitive

The DCM_SP primitive accesses traditional DCM features to provide clock deskew, frequency synthesis, and fixed and variable phase shifting.

Each port description includes the signal direction and notes the DCM functionality that is used when that port is used. [Table 2-5](#) provides the abbreviated name for each function unit used in [Table 2-6](#). [Table 2-6](#) lists the DCM_SP ports.

Table 2-5: Functional Unit Abbreviations for Table 2-5

| Abbreviation | DCM Function |
|--------------|-------------------------------|
| DLL | Delay-locked Loop |
| PS | Variable Phase shift |
| DFS | Digital frequency synthesizer |

Table 2-6: DCM_SP Ports

| Port | Direction | Description | DLL | PS | DFS |
|----------|--------------|--|-----|----|----------|
| CLKIN | Clock Input | Clock input to DCM. Always required. The CLKIN frequency and jitter must fall within the limits specified in the data sheet. | R | R | R |
| CLKFB | Input | Clock feedback input to DCM. The feedback input is required unless the DFS outputs, CLKFX or CLKFX180, are used stand-alone. The source of the CLKFB input must be the CLK0 or CLK2X output from the DCM and the CLK_FEEDBACK must be set to 1X or 2X accordingly. When set to NONE, CLKFB is unused. Ideally, the feedback point includes the delay added by the clock distribution network, either internally or externally. | R | R | Optional |
| RST | Input | Asynchronous reset input. Resets the DCM logic to its post-configuration state. Causes DCM to reacquire and relock to the CLKIN input. Invertible within DCM block. Non-inverted behavior shown below. 0: No effect 1: Reset DCM block. Hold RST pulse High for at least three valid CLKIN cycles | R | R | R |
| PSEN | Input | Variable phase-shift enable. Can be inverted within a DCM block. Non-inverted behavior shown below. 0: Disable variable phase shift. Ignore inputs to phase shifter 1: Enable variable phase shift operations on next rising PSCLK clock edge | | R | |
| PSINCDEC | Input | Increment/decrement variable phase shift. Can be inverted within a DCM block. Non-inverted behavior shown below. 0: Decrement phase shift value on next enabled, rising PSCLK clock edge 1: Increment phase shift value on next enabled, rising PSCLK clock edge | | R | |
| PSCLK | Clock Input | Clock input to variable phase shifter, clocked on rising edge. When using a global clock buffer, only the upper eight BUFGMUXes can drive PSCLK: BUFGMUX_X2Y1, BUFGMUX_X2Y2, BUFGMUX_X2Y3, BUFGMUX_X2Y4, BUFGMUX_X3Y5, BUFGMUX_X3Y6, BUFGMUX_X3Y7, and BUFGMUX_X3Y8. | | R | |
| CLK0 | Clock Output | Same frequency as CLKIN, 0° phase shift (i.e., not phase shifted). Always conditioned to a 50% duty cycle on Spartan-6 FPGAs. CLK_FEEDBACK must be set to 1X or 2X to deskew CLK0. When CLK_FEEDBACK is set to NONE, no phase relationship with CLKIN is present. ⁽¹⁾ | R | | |
| CLK90 | Clock Output | Same frequency as CLKIN, 90° phase shift (quarter period). Always conditioned to a 50% duty cycle on Spartan-6 FPGAs. ⁽¹⁾ | R | | |
| CLK180 | Clock Output | Same frequency as CLKIN, 180° phase shift (half period). Always conditioned to a 50% duty cycle on Spartan-6 FPGAs. ⁽¹⁾ | R | | |

Table 2-6: DCM_SP Ports (Cont'd)

| Port | Direction | Description | DLL | PS | DFS |
|-----------|--------------|--|-----|----|-----|
| CLK270 | Clock Output | Same frequency as CLKIN, 270° phase shift (three-quarters period). Always conditioned to a 50% duty cycle on Spartan-6 FPGAs. ⁽¹⁾ | R | | |
| CLK2X | Clock Output | Double-frequency clock output, 0° phase shift. When available, the CLK2X output always has a 50% duty cycle. Either CLK0 or CLK2X is required as a feedback source for DLL functions. Clock Doubler (CLK2X, CLK2X180) output. ⁽¹⁾ | R | | |
| CLK2X180 | Clock Output | Double-frequency clock output, 180° phase shift. When available, the CLK2X180 output always has a 50% duty cycle. Clock doubler (CLK2X, CLK2X180) output. ⁽¹⁾ | R | | |
| CLKDV | Clock Output | Divided clock output, controlled by the CLKDV_DIVIDE attribute. The CLKDV output has a 50% duty cycle unless the CLKDV_DIVIDE attribute is a non-integer value. ⁽¹⁾ $F_{CLKDV} = \frac{F_{CLKIN}}{CLKDV_DIVIDE}$ | R | | |
| CLKFX | Clock Output | Synthesized clock output, controlled by the CLKFX_MULTIPLY and CLKFX_DIVIDE attributes. Always has a 50% duty cycle. If the CLKFX or CLKFX180 clock outputs are used standalone, then no clock feedback is required. ⁽¹⁾ $F_{CLKFX} = F_{CLKIN} \frac{CLKFX_MULTIPLY}{CLKFX_DIVIDE}$ | | | R |
| CLKFX180 | Clock Output | Synthesized clock output CLKFX, 180° phase shift (appears to be an inverted version of CLKFX). Always has a 50% duty cycle. If only CLKFX or CLKFX180 clock outputs are used on the DCM, then no feedback loop is required. ⁽¹⁾ | | | R |
| STATUS[0] | Output | Variable phase shift overflow. Control output for variable fine phase shifting. The variable phase shifter has reached a minimum or maximum limit value. 0: The phase shift has not yet reached its limit value 1: The phase shift has reached its limited value | | R | |
| STATUS[1] | Output | CLKIN Input Stopped Indicator. Available only when the CLKFB feedback input is connected. Held in reset until the LOCKED output is asserted. Requires at least one CLKIN cycle to become active. Never asserted if CLKIN never toggles. 0: CLKIN input is toggling 1: CLKIN input is not toggling even though the LOCKED output can still be High. | R | R | R |
| STATUS[2] | Output | CLKFX or CLKFX180 output stopped indicator. Held in reset until the LOCKED output is asserted. 0: CLKFX and CLKFX180 outputs are toggling 1: CLKFX and CLKFX180 outputs are not toggling, even though the LOCKED output can still be High. | | | R |

Table 2-6: DCM_SP Ports (Cont'd)

| Port | Direction | Description | DLL | PS | DFS |
|-------------|-----------|--|-----|----|-----|
| STATUS[7:3] | Output | Reserved. Used for simulating reset circuitry in lower-power Spartan-6 devices (-1L). | | | |
| LOCKED | Output | All DCM features have locked onto the CLKIN frequency. Clock outputs are now valid, assuming CLKIN is within specified limits. 0: DCM is attempting to lock onto CLKIN frequency. DCM clock outputs are not valid 1: DCM is locked onto CLKIN frequency. DCM clock outputs are valid. 1-to-0: DCM lost lock. Reset DCM. The FPGA waits for all DCMs and PLLs to be locked when LCK_CYCLE is set to control the startup cycles without setting the STARTUP_WAIT attribute on any DCM_SP port. In the starting configuration sequence, the global 3-state (GTS) must be deasserted for the DCM to lock. | R | R | R |
| PSDONE | Output | Variable phase shift operation complete. 0: No phase shift operation is active, phase shift operation is in progress, or RST has been asserted. 1: Requested phase shift operation is complete. Next variable phase shift operation can commence. | | R | |

Notes:

1. DCM clock outputs must use either a horizontal clock (default) or a global clock buffer.

Table 2-7 lists the DCM_SP attributes. All attributes are set at design time and programmed during configuration. Most, except for the dynamic fine phase shift function, cannot be changed by the FPGA application at run-time. Use <ATTRIBUTE>=<SETTING> as appropriate in the design entry tool to set an attribute.

Table 2-7: DCM_SP Attributes

| Attribute | Allowed Settings and Description |
|--------------------|---|
| DLL_FREQUENCY_MODE | This is a legacy attribute. The DCM is always in the automatic frequency search mode. Setting High or Low makes no effect. |
| DFS_FREQUENCY_MODE | This is a legacy attribute. The DCM is always in the automatic frequency search mode. Setting High or Low makes no effect. |
| CLKIN_PERIOD | Specifies in ns the period of the clock used to drive the CLKIN pin of the DCM. Setting a Spartan-6 FPGA CLKIN_PERIOD helps reduce DFS jitter and results in a faster locking time. |
| CLK_FEEDBACK | Defines the frequency of the feedback clock. 1X: Default. CLK0 feedback. Same frequency as CLKIN 2X: CLK2X feedback. Double the frequency of CLKIN NONE: No feedback. DCM and DLL outputs run without a fixed phase. |

Table 2-7: DCM_SP Attributes (Cont'd)

| Attribute | Allowed Settings and Description |
|--------------------|---|
| CLKDV_DIVIDE | <p>Defines the frequency of the CLKDV output. Allowable values for CLKDV_DIVIDE include 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, 16.</p> <p>The locking time is longer, and there is more output jitter when CLKDV_DIVIDE is a non-integer value.</p> $F_{CLKDV} = \frac{F_{CLKIN}}{CLKDV_DIVIDE}$ |
| CLKFX_MULTIPLY | <p>Defines the multiplication factor for the frequency of the CLKFX and CLKFX180 outputs. Used in conjunction with the CLKFX_DIVIDE attribute. Allowable values for CLKFX_MULTIPLY include integers ranging from 2 to 32. Default value is 4.</p> $F_{CLKFX} = F_{CLKIN} \frac{CLKFX_MULTIPLY}{CLKFX_DIVIDE}$ |
| CLKFX_DIVIDE | <p>Defines the division factor for the frequency of the CLKFX and CLKFX180 outputs. Used in conjunction with the CLKFX_MULTIPLY attribute. Allowable values for CLKFX_DIVIDE include integers ranging from 1 to 32. Default value is 1.</p> $F_{CLKFX} = F_{CLKIN} \frac{CLKFX_MULTIPLY}{CLKFX_DIVIDE}$ |
| PHASE_SHIFT | <p>The PHASE_SHIFT attribute is applicable only if the CLKOUT_PHASE_SHIFT attribute is set to FIXED or VARIABLE. Defines the rising-edge skew between CLKIN and all the DCM clock outputs at configuration and consequently phase shifts the DCM clock outputs.</p> <p>The skew or phase shift value is specified as an integer that represents a fraction of the clock period as expressed in the equations in Phase Shift, page 64. The integer value must range from -255 to 255. The default is 0.</p> |
| CLKOUT_PHASE_SHIFT | <p>Sets the phase shift mode. Together with the PHASE_SHIFT constraint, implements the DPS feature of the DCM. Affects all DCM clock outputs from both the DLL and DFS units.</p> <p>NONE: Default. CLKIN and CLKFB are in phase (no skew) and phase relationship cannot be changed. Equivalent to FIXED setting with a PHASE_SHIFT value of 0.</p> <p>FIXED: Phase relationship is set at configuration by the PHASE_SHIFT attribute value and cannot be changed by the application</p> <p>VARIABLE: Phase relationship is set at configuration by the PHASE_SHIFT attribute value but can be changed by the application using the Variable Phase Shift controls, PSEN, PSCLK, PSINCDEC, and PSDONE.</p> |

Table 2-7: DCM_SP Attributes (Cont'd)

| Attribute | Allowed Settings and Description |
|-------------------|---|
| DESKEW_ADJUST | <p>Controls the clock delay alignment between the FPGA clock input pin and the DCM output clocks.</p> <p>SYSTEM_SYNCHRONOUS: Default. All devices clocked by a common, system-wide clock source.</p> <p>SOURCE_SYNCHRONOUS: Clock is provided by the data source, i.e., source-synchronous applications.</p> <p>Do not use this setting to phase shift DCM clock outputs. Instead, use the CLK and PHASE_SHIFT constraints to achieve accurate phase shifting.</p> <p>DESKEW_ADJUST has no effect for cascaded DCMs, DCMs using external feedback, or DCMs where the clock source does not come from a global clock input routed through the BUFIO2.</p> |
| STARTUP_WAIT | <p>Controls whether the FPGA configuration signal DONE waits for the DCM to assert its LOCKED signal before going High. In the starting configuration sequence, GTS must be deasserted for the DCM to lock.</p> <p>FALSE: Default. DONE is asserted at the end of configuration without waiting for the DCM to assert LOCKED.</p> <p>TRUE: The DONE signal does not transition High until the LOCKED signal transitions High on the associated DCM. STARTUP_WAIT does not prevent LOCKED from transitioning High. The FPGA startup sequence must also be modified to insert a LCK_CYCLE (lock) cycle before the postponed cycle. The DONE cycle or the GWE cycle are typical choices.</p> <p>When more than one DCM is configured with STARTUP_WAIT = TRUE, the FPGA waits until all DCMs are LOCKED. The FPGA waits for all DCMs and PLLs to be locked when LCK_CYCLE is set without STARTUP_WAIT.</p> |
| CLKIN_DIVIDE_BY_2 | <p>Optionally divides in half the CLKIN before entering the DCM block. In some applications, CLKIN_DIVIDE_BY_2 reduces the input clock frequency to acceptable limits.</p> <p>FALSE: Default. CLKIN input directly feeds the DCM.</p> <p>TRUE: Divides CLKIN frequency in half and provides roughly a 50% duty cycle clock before entering the DCM. Helps high frequency clocks meet the DCM input clock frequency or duty cycle requirements. Divides the clock frequency in half when determining operating frequency modes and calculating phase shift limits.</p> |
| LOC | Specifies the physical location of the DCM. |

DCM_CLKGEN Primitive

The DCM_CLKGEN primitive provides access to more advanced DFS features. These features are:

- Lower output jitter on CLKFX and CLKFX180
- Improved jitter tolerance on CLKIN
- Dynamic programming of M and D values, overwrites the CLKFX_MULTIPLY and CLKFX_DIVIDE attributes
- Wider range of the M and D values than the fixed CLKFX_MULTIPLY and CLKFX_DIVIDE attributes
- CLKFXDV provides an additional divided version of the CLKFX output
- Free running oscillator in the event of lost input clock

- Spread-spectrum clock generation

Table 2-8 lists the DCM_CLKGEN ports. Each port lists has a brief description and the signal direction.

Table 2-8: DCM_CLKGEN Ports

| Port | Direction | Description |
|----------------|--------------|---|
| CLKIN | Clock Input | Clock input to DCM. Always required. The CLKIN frequency and jitter must fall within the limits specified in the data sheet. In the case of free-running oscillator mode, running clock needs to be connected until DCM is locked and DCM is frozen, then clock can be removed. In the other modes, a free running clock needs to be provided and remains. |
| RST | Input | Asynchronous reset input. Resets the DCM logic to its post-configuration state. Causes DCM to reacquire and relock to the CLKIN input. Invertible within DCM block. Non-inverted behavior shown below. 0: No effect 1: Reset DCM block. Hold RST pulse High for at least three valid CLKIN cycles |
| FREEZEDCM | Input | Prevents tap adjustment drift in the event of a lost CLKIN input. The DCM is then configured into a free-run mode. |
| CLKFX | Clock Output | Synthesized clock output, controlled by the CLKFX_MULTIPLY and CLKFX_DIVIDE attributes. Can be either statically set or dynamically programmed through a dedicated 4-wire SPI port (PROGDATA, PROGCLK, PROGDONE, and PROGEN). Always has a 50% duty cycle. ⁽¹⁾ $F_{CLKFX} = F_{CLKIN} \frac{CLKFX_MULTIPLY}{CLKFX_DIVIDE}$ |
| CLKFX180 | Clock Output | Synthesized clock output CLKFX, 180° phase shift (appears to be an inverted version of CLKFX). Always has a 50% duty cycle. ⁽¹⁾ |
| LOCKED | Output | Synchronous output indicates whether the DCM is ready for operation. 0: DCM clock outputs are not valid 1: DCM is ready for operation 1-to-0: DCM lost LOCK. Reset DCM. The FPGA waits for all DCMs and PLLs to be locked when LCK_CYCLE is set to control the startup cycles without setting the STARTUP_WAIT attribute on any DCM_SP port. In the starting configuration sequence, GTS must be deasserted for the DCM to lock. |
| STATUS[1] | Output | STATUS[1] port is unused for DCM_CLKGEN. |
| STATUS[2] | Output | CLKFX or CLKFX180 output stopped indicator. 0: CLKFX and CLKFX180 outputs are toggling 1: CLKFX and CLKFX180 outputs are not toggling, even though the LOCKED output can still be High. |
| STATUS[7:3, 0] | Output | Reserved. Used for simulating reset circuitry in lower-power Spartan-6 devices (-1L). SIMPRIMS only. |
| CLKFXDV | Clock Output | Divided CLKFX output clock. Divide value derived from CLKFXDV_DIVIDE attribute. There is no phase alignment between CLKFX and CLKFXDV. ⁽¹⁾ $F_{CLKFXDV} = \frac{F_{CLKFX}}{CLKFXDV_DIVIDE}$ |

Table 2-8: DCM_CLKGEN Ports (Cont'd)

| Port | Direction | Description |
|----------|-----------|--|
| PROGDONE | Output | Indication of M and D programming completeness. 0: No programming operation of M and D is active, the programming is in progress, or RST has been asserted. 1: Requested programming is complete. Next PROGCLK operation can commence. |
| PROGDATA | Input | Serial data input to supply information for the programming of M and/or D values of the DCM. During programming, the value shifted in will be M -1 or D -1 and will begin with the LSB. This input must be applied synchronous to the PROGCLK input. |
| PROGEN | Input | Indication of whether the programming of M and D values is enabled or not. 0: Programming of CLKFX_MULTIPLY and CLKFX_DIVIDE is disabled. 1: Programming of CLKFX_MULTIPLY and CLKFX_DIVIDE is enabled. |
| PROGCLK | Input | Clock input for the programming of M and D values. When using a global clock buffer, only the upper eight BUFMUXs can drive PROGCLK: BUFMUX_X2Y1, BUFMUX_X2Y2, BUFMUX_X2Y3, BUFMUX_X2Y4, BUFMUX_X3Y5, BUFMUX_X3Y6, BUFMUX_X3Y7, and BUFMUX_X3Y8. |

Notes:

1. DCM clock outputs must use either a horizontal clock (defaults) or a global clock buffer.

Table 2-9 lists the DCM_CLKGEN attributes. All attributes can be set at design time and programmed during configuration. Use <ATTRIBUTE>=<SETTING> as appropriate in the design entry tool to set an attribute. The CLKFX_MULTIPLY and CLKFX_DIVIDE are allowed to be changed by the FPGA application at run-time.

Table 2-9: DCM_CLKGEN Attributes

| Attribute | Allowed Settings and Description |
|----------------|--|
| CLKFX_MULTIPLY | Defines the multiplication factor for the frequency of the CLKFX and CLKFX180 outputs. Used in conjunction with the CLKFX_DIVIDE attribute. Allowable values for CLKFX_MULTIPLY include integers ranging from 2 to 256. Default value is 4. $F_{CLKFX} = F_{CLKIN} \frac{CLKFX_MULTIPLY}{CLKFX_DIVIDE}$ |
| CLKFX_DIVIDE | Defines the division factor for the frequency of the CLKFX and CLKFX180 outputs. Used in conjunction with the CLKFX_MULTIPLY attribute. Allowable values for CLKFX_DIVIDE include integers ranging from 1 to 256. Default value is 1. $F_{CLKFX} = F_{CLKIN} \frac{CLKFX_MULTIPLY}{CLKFX_DIVIDE}$ For proper locking with input frequencies below 52 MHz, $CLKFX_DIVIDE < \frac{F_{CLKIN}}{0.500\text{MHz}}$ |

Table 2-9: DCM_CLKGEN Attributes (Cont'd)

| Attribute | Allowed Settings and Description |
|-------------------|---|
| CLKFXDV_DIVIDE | Defines the division factor for the frequency of the CLKFXDV output. Allowable values are 2, 4, 8, 16, 32. Default value is 2. $F_{CLKFXDV} = \frac{F_{CLKFX}}{CLKFXDV_DIVIDE}$ |
| CLKFX_MD_MAX | When dynamic programming of M and D values is engaged, this attribute specifies the maximum ratio of the M and D. CLKFX_MD_MAX is mainly used for static timing analysis. Default value is: $\frac{CLKFX_MULTIPLY}{CLKFX_DIVIDE}$ |
| DFS_BANDWIDTH | Reserved. |
| PROG_MD_BANDWIDTH | Reserved. |
| SPREAD_SPECTRUM | CENTER_LOW_SPREAD, CENTER_HIGH_SPREAD, VIDEO_LINK_M0 ⁽¹⁾ , VIDEO_LINK_M1 ⁽¹⁾ , VIDEO_LINK_M2 ⁽¹⁾ , NONE |
| STARTUP_WAIT | Controls whether the FPGA configuration signal DONE waits for the DCM to assert its LOCKED signal before transitioning High. In the starting configuration sequence, GTS must be deasserted for the DCM to lock. FALSE: Default. DONE is asserted at the end of configuration without waiting for the DCM to assert LOCKED. TRUE: The DONE signal does not transition High until the LOCKED signal transitions High on the associated DCM. STARTUP_WAIT does not prevent LOCKED from transitioning High. The FPGA startup sequence must also be modified to insert a LCK (lock) cycle before the postponed cycle. The DONE cycle or GWE cycle are typical choices. When more than one DCM is configured, the FPGA waits until all DCMs are LOCKED. |
| CLKIN_PERIOD | Specifies the CLKIN period in order to optimize the CLKFX/CLKFX180 outputs. This in turn will result in a faster locking time. Any real number is acceptable. Default value is 0. |
| CLKFX_MD_MAX | When dynamic programming of M and D values is engaged or when SPREAD_SPECTRUM is used, this attribute specifies the maximum ratio of the M and D. CLKFX_MD_MAX is mainly used for static timing analysis. Floating point value. Default value is CLKFX_MULTIPLY/CLKFX_DIVIDE. |

Notes:

1. Soft modes of spread spectrum requires an external state machine for correct operation. PROGCLK and PROGDATA must be connected to an external state machine to create a spread-spectrum clock source.

DCM_SP Design Guidelines

Input Clock Frequency Range

The DCM clock input frequency depends on whether the DLL, the DFS, or both are utilized in the application.

The Spartan-6 FPGA data sheet specifies the clock input, CLKIN, and the frequency range for both the DFS and the DLL. The DFS, when used stand-alone, has a wider frequency

range than the DLL. If the application uses both the DLL and DFS, then the more restrictive DLL requirements apply. [Table 2-10](#) outlines the input clock frequency range specification names used in the Spartan-6 FPGA data sheet.

Table 2-10: Input Clock Frequency Range

| Function | Minimum Frequency | Maximum Frequency |
|----------|--------------------|--------------------|
| DFS | CLKIN_FREQ_FX_MIN | CLKIN_FREQ_FX_MAX |
| DLL | CLKIN_FREQ_DLL_MIN | CLKIN_FREQ_DLL_MAX |

Output Clock Frequency Range

The DCM output clocks also have a specified frequency range.

Input Clock and Clock Feedback Variation

The DCM expects a stable, monotonic clock input. However, for maximum flexibility, the DCM tolerates a certain amount of clock jitter on the CLKIN input and a reasonable amount of frequency variation on both the CLKIN input and the CLKFB clock feedback input.

There are two types of jitter tolerance on the CLKIN input.

- Cycle-to-cycle jitter
- Period jitter

Cycle-to-Cycle Jitter

Cycle-to-cycle jitter indicates how much the CLKIN input period is allowed to change from one cycle to the next. [Table 2-11](#) outlines the maximum allowable cycle-to-cycle change specification names used in the Spartan-6 FPGA data sheet.

Table 2-11: Maximum Allowable Cycle-to-Cycle Jitter

| Function | All Frequency Ranges |
|----------|----------------------|
| DFS | CLKIN_CYC_JITT_FX |
| DLL | CLKIN_CYC_JITT_DLL |

Period Jitter

The other applicable type of jitter is called period jitter. Period jitter indicates the maximum variation in the clock period over millions of clock cycles. Cycle-to-cycle jitter shows the change from one clock cycle to the next while period jitter indicates the maximum range of change over time. [Table 2-12](#) outlines maximum allowable period jitter specification names used in the Spartan-6 FPGA data sheet.

Table 2-12: Maximum Allowable Period Jitter

| Function | Frequency Range | |
|----------|------------------------|------------------------|
| | $\leq 150 \text{ MHz}$ | $\geq 150 \text{ MHz}$ |
| DFS | CLKIN_PER_JITT_FX_LF | CLKIN_PER_JITT_FX_HF |
| DLL | CLKIN_PER_JITT_DLL_LF | CLKIN_PER_JITT_DLL_HF |

DLL Feedback Delay Variance

Another source of stability for the DCM is the clock feedback path used by the DLL. The feedback path delay variance must also be within the limit shown in [Table 2-13](#). This limit only applies to an external feedback path as any on-chip variance is minimal when connected to a global clock line.

Table 2-13: External Feedback Path Delay Variation

| Specification | Description |
|---------------------|---|
| CLKFB_DELAY_VAR_EXT | Maximum allowable variation in off-chip CLKFB feedback path |

Spread Spectrum Clock Reception

The DCMs accept typical spread-spectrum clocks. The DLL portion of the DCM tracks the frequency changes created by a typical spread-spectrum clock, to drive the global clocks to the FPGA internal logic. The spread spectrum clock must meet the DLL input requirements as specified in the device data sheet. See the Input Clock Jitter Tolerance and Delay Path Variation specifications in the Recommended Operating Conditions for the DLL, CLKIN_CYC_JITT_DLL, and CLKIN_PER_JITT_DLL.

The DFS can track a typical spread-spectrum input as long as it meets the input clock specifications.

Optimal DCM Clock and External Feedback Inputs

Each DCM has multiple optimal inputs for an incoming clock signal or external feedback signal.

LOCKED Output Behavior

The LOCKED output of the DCM indicates when all the enabled DCM functions have locked to the CLKIN input.

[Figure 2-10](#) shows the behavior of the LOCKED output. The LOCKED output is Low immediately after the FPGA finishes its configuration process and is Low whenever the RST input is asserted.

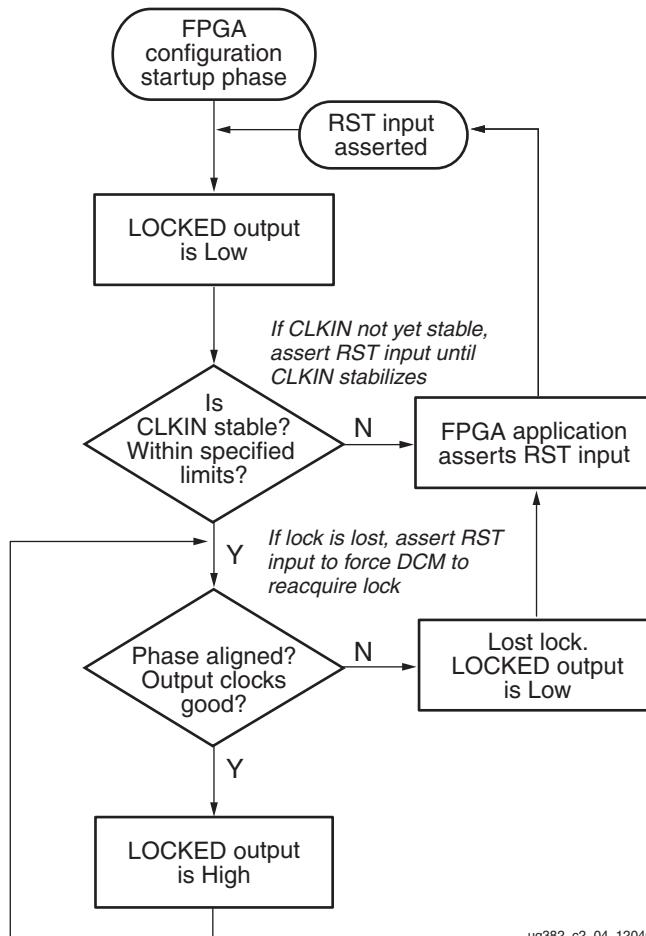
After configuration, the RST input should be asserted until the CLKIN input stabilizes (highly recommended). Once the RST input is released, the DCM locks to the stabilized CLKIN input frequency. The DLL uses both the CLKIN input and the CLKFB feedback input to determine when locking is complete, that is, when the rising edges of CLKIN and CLKFB are phase-aligned. The DFS synthesizes an output clock and locks when a valid frequency is present on the output.

The Spartan-6 FPGA data sheet provides worst-case locking times. In general, the DLL unit outputs lock faster with increasing clock frequency. The DFS lock time depends on many factors including the frequency of operation and the frequency multiply and divide factors. Smaller multiply and divide factors result in faster lock times.

To assure that the system clock is established before the FPGA completes its configuration process, the DCM can optionally delay the completion of the configuration process until after the DCM locks. The STARTUP_WAIT attribute activates this feature.

Until LOCKED is High, the stability of the DCM clock outputs is questionable. The DCM output clocks are not valid until LOCKED is High. Before that time, the DCM clock outputs can exhibit glitches, spikes, or other spurious behavior.

After the DCM has been initially locked, the LOCKED signal can stay High when CLKIN stops. The LOCKED signal can also stay High when CLKIN varies considerably.



ug382_c2_04_120408

Figure 2-10: Functional Behavior of LOCKED Output

Once the DCM loses LOCKED, it does not automatically attempt to reacquire LOCKED. When the DCM loses LOCKED (High transitions Low), the FPGA application must take the appropriate action. For example, once LOCKED is lost, resetting the DCM through the RST input forces the DCM to reacquire LOCKED.

Using the LOCKED Signal

To operate properly, the DCM requires a stable, monotonic clock input. Once locked, the DCM tolerates clock period variations up to the value specified in the Spartan-6 FPGA data sheet. However, it is possible for the clock to stray outside the limits, for the LOCKED output to stay High, and for the DCM outputs to be invalid, for instance when the CLKIN stops toggling. It is a good design practice to monitor both LOCKED and the STATUS signals. Monitoring STATUS[1] is recommended to indicate when the CLKIN has stopped (moved outside the acceptable CLKIN tolerances). STATUS[1] transitions High after one missed CLKIN cycle. STATUS[1] is not a sticky bit; it transitions Low once the CLKIN has

returned. Monitor both the LOCKED and STATUS[1] bits for the most robust indicator of the status of the DCM output clock.

Similarly, the STATUS[2] bit indicates that the DFS output CLKFX is stopped. If LOCKED = 0 and STATUS[2] = 1, then the DCM should be reset. If the FPGA application also resets the DCM, then OR the reset signal from the FPGA application with the monitored output signals.

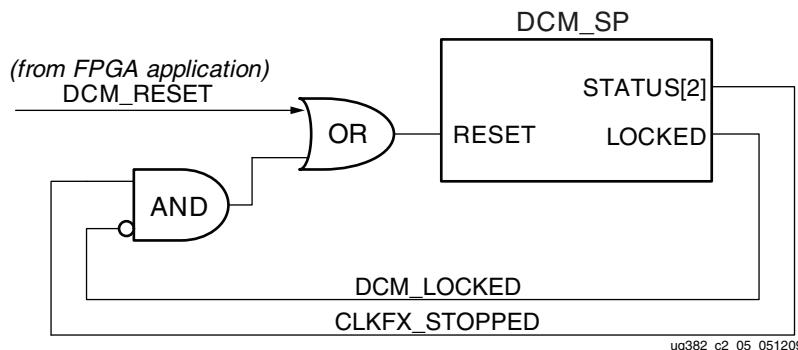


Figure 2-11: Spartan-6 FPGA DCM DFS Lock Logic

RST Input Behavior

The asynchronous RST input forces the DCM to its post-configuration state. Use the RST pin when changing the input clock frequency beyond the allowable range. The active-High RST pin either must connect to a dynamic signal or must be tied to ground. The RST input must be asserted for three valid CLKIN cycles or longer.

If the input clock frequency is not yet stable after configuration, assert RST until the clock stabilizes. When using external feedback, hold the DCM in reset immediately after configuration.

If the DCM loses lock, for instance, the LOCKED output was High then transitioned Low, the FPGA application must assert RST to force the DCM to reacquire the input clock frequency. If the DCM LOCKED output is High, then the LOCKED signal deactivates within four source clock cycles when RST is asserted. Asserting RST forces the DCM to reacquire lock.

Asserting RST also resets the DCM's delay tap position to zero. Due to the tap position changes, glitches can occur on the DCM clock output pins. Similarly, the duty cycle on the clock outputs can be affected when RST is asserted.

Asserting RST also resets the present variable phase shift value back to the value specified by the PHASE_SHIFT attribute.

For lower-power Spartan-6 FPGA (-1L) designs, a small reset circuit is automatically inserted into designs that use the DFS outputs. The additional logic is required to ensure proper operation when using DCM_SP or DCM_CLKGEN with $V_{CCINT} = 1.0V$. Because of the additional circuitry, some signal names will change.

The low-power reset circuit monitors the behavior of the reserved status ports for STATUS[5] and STATUS[7] to determine if a reset is required (Figure 2-12). The low-power reset circuit requires the addition of seven slices for the reset circuitry. The reset circuit requires four shift registers, five slice registers, and nine slice LUTs.

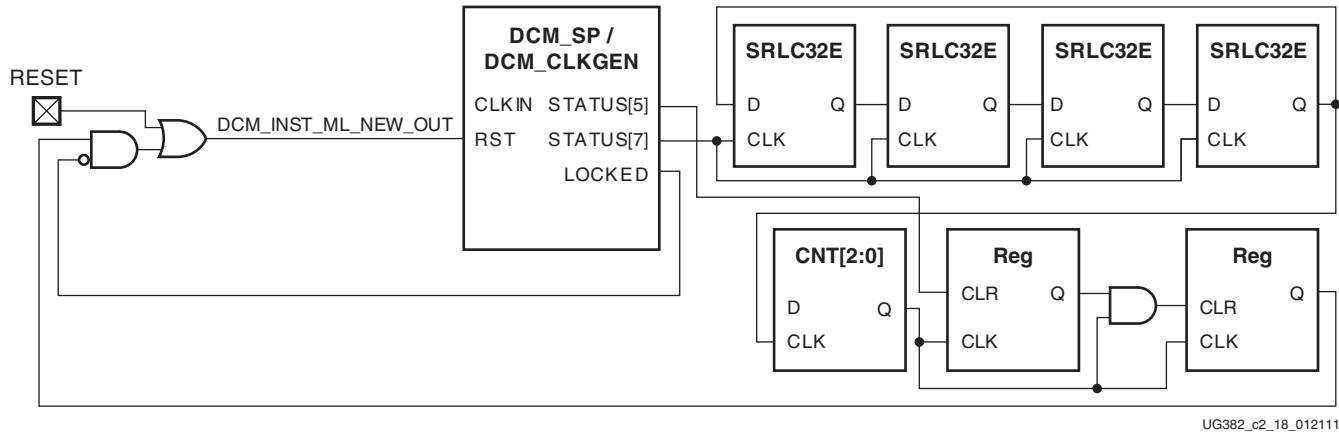


Figure 2-12: Low-Power Reset Circuit

When designing with DCM_CLKGEN, STATUS[2:1] should be used to check whether DCM_CLKGEN is operating correctly. Because STATUS[5] and STATUS[7] are required for the low-power reset circuit, the ISE software and the simulation models show STATUS[7:0] for DCM_CLKGEN.

As shown in [Table 2-14](#), the low-power reset circuitry is automatically inserted during the MAP phase. As a result, XST and TRANSLATE do not show the low-power reset circuit. Post-synthesis simulations and post-translate simulations similarly do not reflect the low-power circuit.

Table 2-14: STATUS For Low-Power Reset Circuit

| | XST | Translate | Map | PAR |
|-------------------------|---------|-----------|----------|----------|
| Low-Power Reset Circuit | No | No | Yes | Yes |
| Simulation Library | UNISIMS | SIMPRIMS | SIMPRIMS | SIMPRIMS |
| STATUS (DCM_CLKGEN) | [2:1] | [7:0] | [7:0] | [7:0] |
| STATUS (DCM_SP) | [7:0] | [7:0] | [7:0] | [7:0] |

The low-power reset circuit is designed to support the full frequency ranges of DCM_SP and DCM_CLKGEN. No other design changes are required.

DCM_CLKGEN Design Guidelines

Although the DCM_SP already provides the comprehensive capability to manage and generate various clocking functions, the Spartan-6 FPGA includes a new primitive (DCM_CLKGEN) to respond to increasing application demands to access more advanced clock synthesis features that are not available in a traditional DCM and PLL. Using this new primitive broadens the FPGA application range, especially in the consumer space:

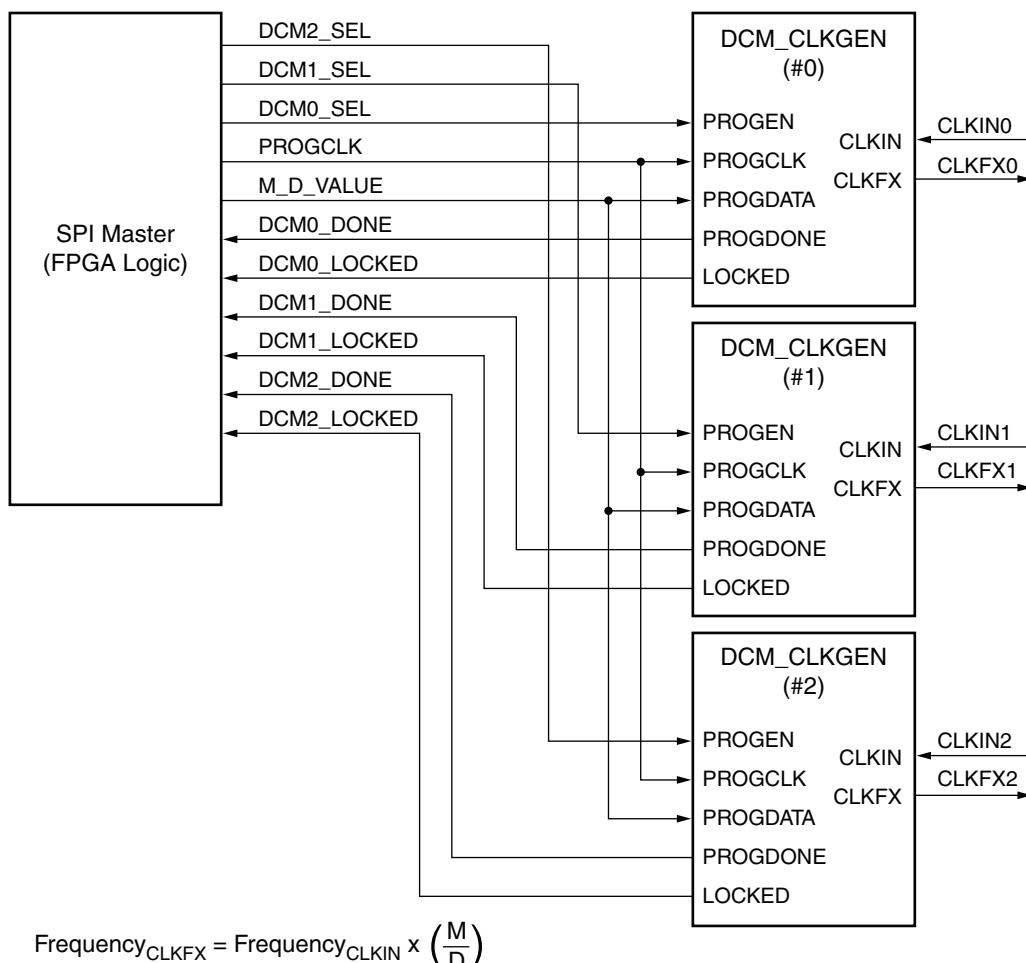
- Some consumer electronics designs provide CLKINs with noise and jitter that traditional DCMs and PLLs can not tolerate.
- Some applications have a CLKIN coming from unsteady or disappearing cable signals. In this case, the DCM output is still expected to continue running.
- Flat-Panel LCD TVs require decent EMI reduction without access to expensive external metal shielding.

- To be compliant with various video timing formats for different screen display modes, a video pixel clock generator requires the ability to perform frequency adjustments during normal operation.
- Complex power management schemes need a way to dynamically scale the clock frequency to tailor power consumption needs.

The following section is a detailed discussion on these new DCM_CLKGEN features.

Dynamic Frequency Synthesis

In contrast to the static clock frequency synthesis described in the Spartan-6 FPGA DCM_SP sections, the DCM_CLKGEN primitive allows the DFS to dynamically synthesize a clock frequency. The M and D values can be programmed to overwrite the corresponding static attributes CLKFX_MULTIPLY and CLKFX_DIVIDE using a dedicated Serial Peripheral Interface (SPI) bus with a programming port using four pins; PROGDATA, PROGEN, PROGCLK, and PROGDONE. The SPI port is always configured as a slave device. The SPI master can be constructed using FPGA logic. Appropriate usage of both the PROGEN and PROGDONE pins allows multiple DCM slaves to be controlled by a single master. [Figure 2-13](#) illustrates the master-slave relationship and their connections.



UG382_c2_06_051209

Figure 2-13: DCM_CLKGEN M and D Programming Interface

An ordered M/D programming sequence can be described as:

- A 2-bit LoadD command 10, followed by 8-bit (D-1), with the LSB first. This consumes exactly 10 cycles when the PROGEN pin must remain a logic High.
- A 2-bit LoadM command 11, followed by 8-bit (M-1), with the LSB first. This consumes exactly 10 cycles when the PROGEN pin must remain a logic High.
- A 1-bit GO command 0. The PROGEN pin must remain a logic High for exactly one cycle.
- The SPI master monitors the PROGDONE pin and waits for it to be asserted to a logic High.
- When the DCM asserts the LOCKED signal High, a new and valid clock frequency should be present on the CLKFX pin.

Figure 2-14 illustrates an example where the M is programmed as 14 and the D as 3. There is a minimal gap requirement between any two commands. The gap between the LoadD and LoadM is at least two cycles. The gap between the LoadM and GO is at least one cycle.

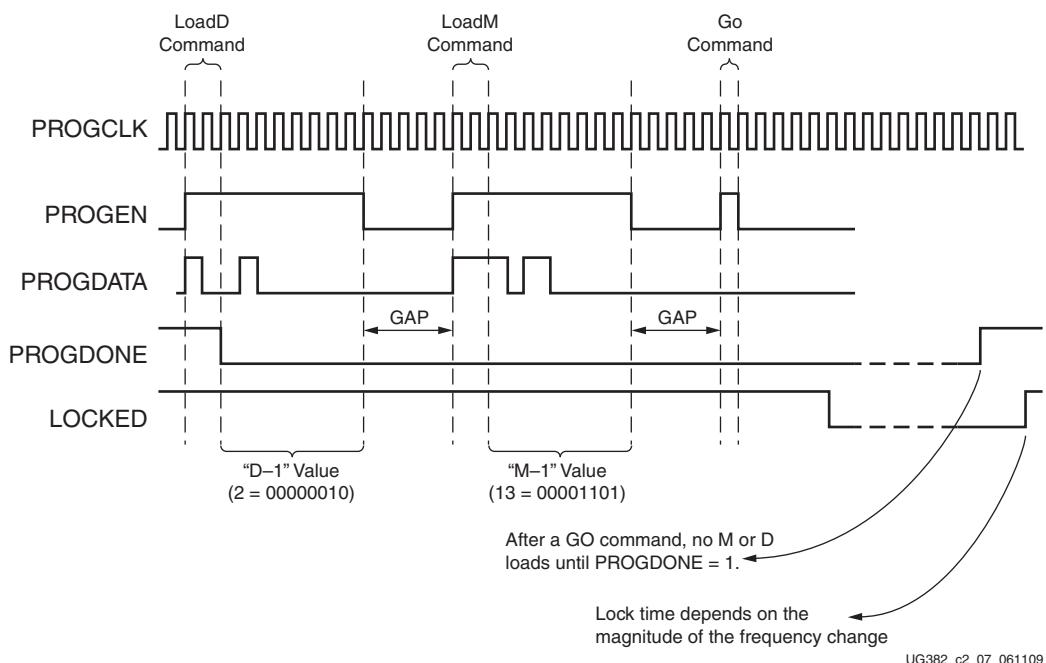


Figure 2-14: DCM_CLKGEN M and D Configuration Timing Waveform

When using DCM_CLKGEN with dynamic frequency synthesis where the output is driving a PLL, the VCO range of the PLL must also be considered. DCM_CLKGEN supports a wide range of output frequencies. All frequencies should be checked to make sure the resultant VCO frequency associated with the PLL setup will not violate the VCO range of the PLL ([Frequency Synthesis Only in Chapter 3](#)).

Spread-Spectrum Clock Generation

Spread-spectrum clock generation (SSCG) is widely used by manufacturers of electronic devices to reduce the spectral density of the electromagnetic interference (EMI) generated by these devices. Manufacturers must ensure that levels of electromagnetic energy emitted do not interfere with the operation of other nearby electronic devices. For example, the clarity of a phone call should not degrade when the phone is next to a video display. In the same way, the display should not be affected when the phone is used.

Electromagnetic Compatibility (EMC) regulations are used to control the noise or EMI that causes these disturbances. Typical solutions for meeting EMC requirements involve adding expensive shielding, ferrite beads, or chokes. These solutions can adversely impact the cost of the final product by complicating PCB routing and forcing longer product development cycles.

SSCG spreads the electromagnetic energy over a large frequency band to effectively reduce the electrical and magnetic field strengths measured within a narrow window of frequencies. The peak electromagnetic energy at any one frequency is reduced by modulating the SSCG output.

The SSCG commonly defines the following parameters:

- Frequency deviation, that is the percentage of the input frequency
- Modulation frequency
- Spread type: up/down/center
- Modulation profile, for example, the shape of the triangle

A typical spread spectrum clock can be defined as: 75 MHz (input frequency), $\pm 2.0\%$ center spread, and 75 KHz triangular modulation. [Figure 2-15](#) illustrates these parameter definitions for center and down spread types.

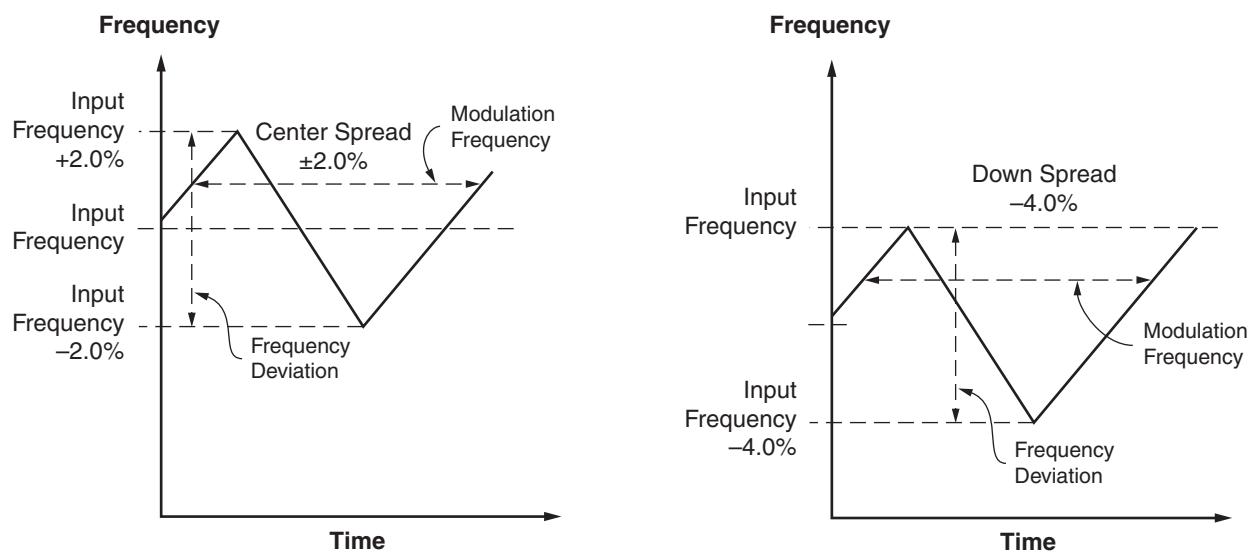


Figure 2-15: Two Types of Spread Spectrum Clocks with Common Parameters Defined

UG382_c2_08_120809

Spread-Spectrum Generation

Spartan-6 FPGAs can generate a spread-spectrum clock source from a standard fixed-frequency oscillator. To generate a spread-spectrum clock source from a fixed-frequency clock source, the DCM_CLKGEN can either use a fixed spread-spectrum solution, providing the simplest implementation, or a soft spread-spectrum solution which adds more flexibility but requires additional logic to continually reprogram DCM_CLKGEN. See [Table 2-15](#).

Table 2-15: DCM_CLKGEN Spread-Spectrum Modes

| Spread Spectrum Values | Fixed Spread-Spectrum Modes | Soft Spread-Spectrum Modes |
|------------------------------|---|--|
| SPREAD_SPECTRUM values | CENTER_LOW_SPREAD CENTER_HIGH_SPREAD | VIDEO_LINK_M0, VIDEO_LINK_M1, VIDEO_LINK_M2 |
| Additional logic | None | SOFT_SS |
| Modulation profile | Triangular | Triangular |
| Spread direction | Center | Down |
| Spread range | Fixed | See Figure |
| F _{MOD} | F _{IN} /1024 | See Figure |
| CLKFX_MULTIPLY | 2–32 ⁽¹⁾ | 7 |
| CLKFX_DIVIDE | 1–4 ⁽¹⁾ | 2, 4 |
| DCM_CLKGEN programming ports | N/A | PROGCLK, PROGEN, PROGDATA, PROGDONE |

Notes:

1. Fixed spread-spectrum CLKFX output frequency must be greater than 50 MHz.

Fixed Spread Spectrum

The simplest way to implement spread spectrum uses a fixed mode ([Figure 2-16](#)) of the DCM_CLKGEN primitive.

`SPREAD_SPECTRUM = < CENTER_LOW_SPREAD, CENTER_HIGH_SPREAD>`

Using this fixed mode, the DCM_CLKGEN automatically creates a spread spectrum clock. The modulation frequency is based on the input frequency. Where:

$$\text{Modulation Frequency} = \text{input frequency}/1024.$$

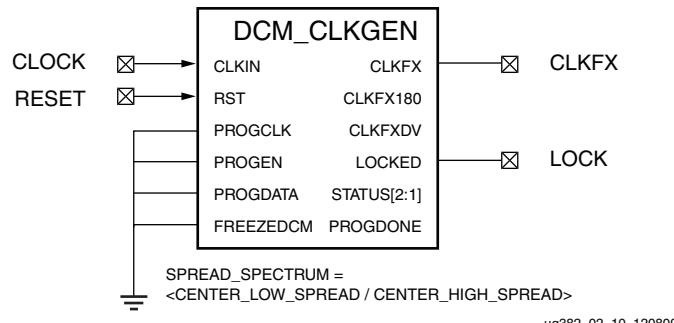
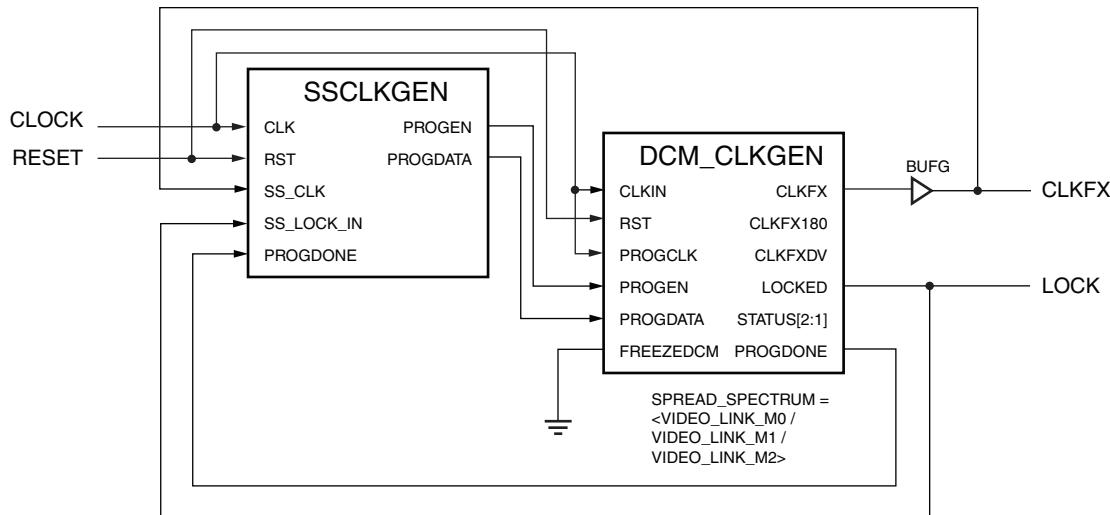


Figure 2-16: Fixed Spread Spectrum

Soft Spread Spectrum

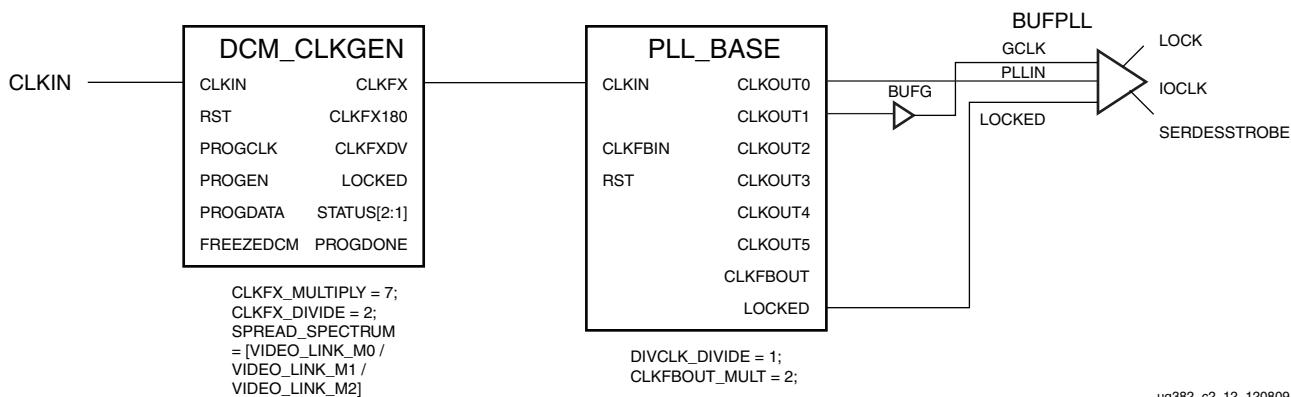
For applications requiring downspread modulation, the DCM_CLKGEN can use one of three spread-spectrum modes to control the rate that the DCM_CLKGEN switches between different programming values. A small counter-based state machine can control DCM_CLKGEN to provide a triangular, spread-spectrum clock source (see [Figure 2-17](#)). Because DCM_CLKGEN requires an external state machine for the soft spread spectrum, failure to connect PROGCLK and PROGDATA to logic will result in a DRC error.



ug382_c2_11_120809

Figure 2-17: Soft Spread Spectrum

Typical video link applications need 7:1 serialization. When using IOSERDES2, clock generation requires using both a DCM_CLKGEN and PLL. DCM_CLKGEN generates the spread-spectrum clock source typically with M = 7 and D = 2. To access high-speed I/O clock networks, the PLL multiplies the clock for up to the full clock frequency required for SDR clocking using the BUFPLL as shown in [Figure 2-18](#).



ug382_c2_12_120809

*Figure 2-18: VIDEO_LINK Setup for Downspread Spread Spectrum 20-107 MHz
(when $F_{IN} \times 3.5 < CLKOUT_FREQ_FX$)*

VIDEO_LINK_M0, VIDEO_LINK_M1, and VIDEO_LINK_M2 are selected to cover typical pixel clock frequencies. VIDEO_LINK_M0 has the fastest modulation frequency response to help compensate for the higher periods associated with 20 MHz pixel clocks. Actual modulation frequency depends on the SSCLKGEN setup. VIDEO_LINK_M1 is slower and VIDEO_LINK_M2 is the slowest.

Pixel clock frequencies of CLKFX_MULTIPLY = 7, CLKFX_DIVIDE = 2 are used when the maximum CLKOUT_FREQ_FX is maintained. For higher pixel clock frequencies, DCM_CLKGEN must be changed to CLKFX_MULTIPLY = 7 and CLKFX_DIVIDE = 4 to compensate for the PLL addition. See [Figure 2-19](#). This slows down the modulation frequency requiring VIDEO_LINK_M1.

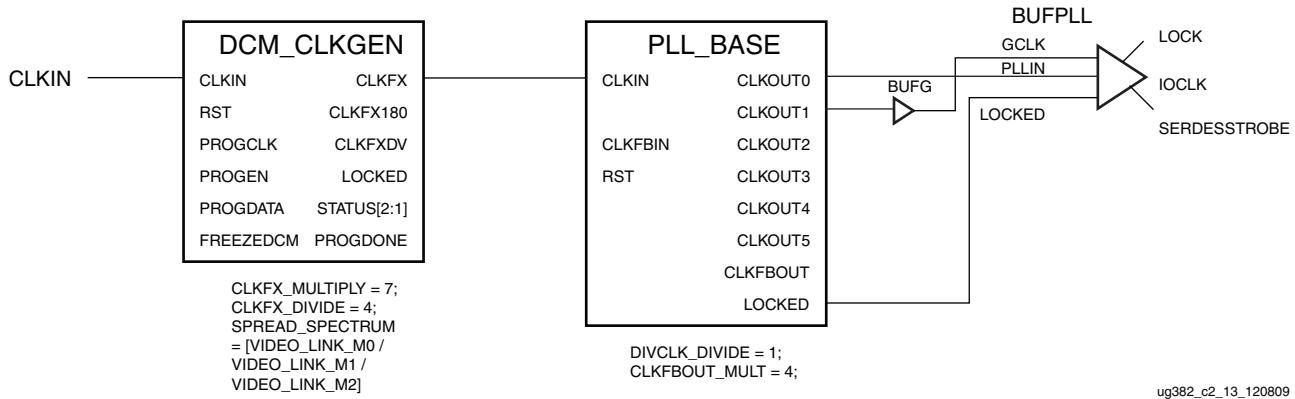


Figure 2-19: VIDEO_LINK Setup for Downspread Spread Spectrum (when $F_{IN} > 107$ MHz)

Free-Running Oscillator

The DCM_CLKGEN can be used to generate a clock source by tying the LOCKED output pin to the FREEZEDCM input pin. The DCM requires a kick-start in this mode; instead of an always running clock, any oscillating signal can be used until the DCM has LOCKED. [Figure 2-20](#) illustrates an example.

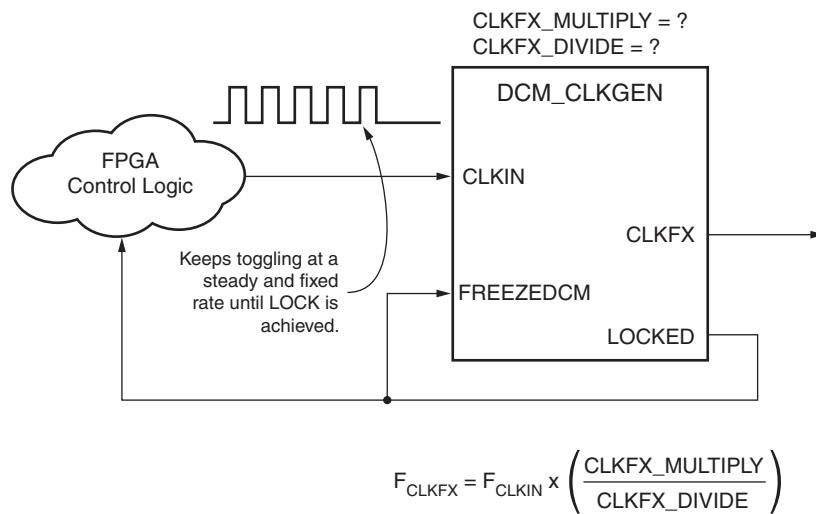


Figure 2-20: Setup of a Free-Running Oscillator

For a deterministic frequency output on the CLKFX pin, the CLKIN must keep toggling at a steady rate until the DCM has achieved LOCKED. After that, the CLKIN no longer needs to be present and can be removed.

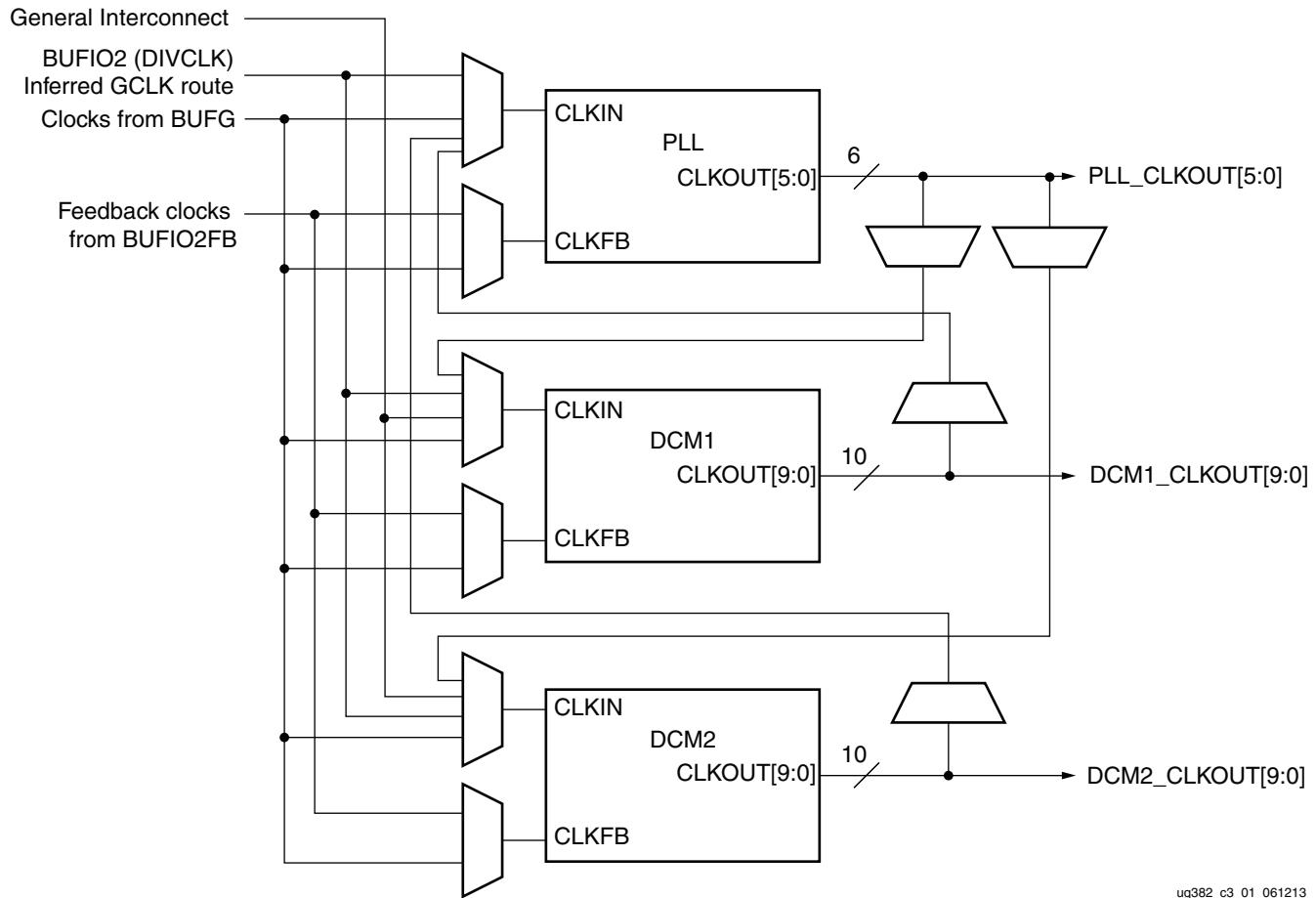
The DCM_CLKGEN will continue to run at the same frequency. If temperature or voltage levels fluctuate, the frequency can vary. Adjust timing constraints to reflect the highest frequency within the frequency drift. When a new clock input is applied, RESET must be asserted.

Phase-Locked Loops

Introduction

The clock management tile (CMT) in Spartan-6 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.



ug382_c3_01_061213

Figure 3-1: Block Diagram of the Spartan-6 FPGA CMT

To support the high-performance SDR clock rates associated with the PLL, each BUFPLL is directly routed to a restricted number of PLL locations. [Table 3-1](#) lists the PLL locations that can be connected to each BUFPLL.

Table 3-1: PLLs with Direct Connections to BUFPLL

| BUFPLL Location | Bank | Valid PLL Locations by Device Type | | |
|-----------------|-------|------------------------------------|--------------|--|
| | | LX4, LX9, LX16, LX25/LX25T | LX45/LX45T | LX75/LX75T LX100/LX100T LX150/LX150T |
| BUFPLL_X1Y5 | 0 | PLL_ADV_X0Y1 | PLL_ADV_X0Y3 | PLL_ADV_X0Y5 |
| BUFPLL_X1Y4 | | PLL_ADV_X0Y0 | PLL_ADV_X0Y2 | PLL_ADV_X0Y3 |
| BUFPLL_MCB_X1Y9 | | | PLL_ADV_X0Y1 | PLL_ADV_X0Y2 |
| BUFPLL_X2Y2 | 1 (5) | PLL_ADV_X0Y1 | PLL_ADV_X0Y3 | PLL_ADV_X0Y5 |
| BUFPLL_X2Y3 | | PLL_ADV_X0Y0 | PLL_ADV_X0Y2 | PLL_ADV_X0Y3 |
| BUFPLL_MCB_X2Y5 | | | PLL_ADV_X0Y1 | PLL_ADV_X0Y2 |
| BUFPLL_X1Y0 | 2 | PLL_ADV_X0Y1 | PLL_ADV_X0Y3 | PLL_ADV_X0Y5 |
| BUFPLL_X1Y1 | | PLL_ADV_X0Y0 | PLL_ADV_X0Y2 | PLL_ADV_X0Y3 |
| BUFPLL_MCB_X1Y5 | | | PLL_ADV_X0Y1 | PLL_ADV_X0Y2 |
| BUFPLL_X0Y2 | 3 (4) | PLL_ADV_X0Y1 | PLL_ADV_X0Y3 | PLL_ADV_X0Y5 |
| BUFPLL_X0Y3 | | PLL_ADV_X0Y0 | PLL_ADV_X0Y2 | PLL_ADV_X0Y3 |
| BUFPLL_MCB_X0Y5 | | | PLL_ADV_X0Y1 | PLL_ADV_X0Y2 |
| | | | PLL_ADV_X0Y0 | PLL_ADV_X0Y0 |

Phase Lock Loop (PLL)

Spartan-6 devices contain up to six CMT tiles. The main purpose of PLLs 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.

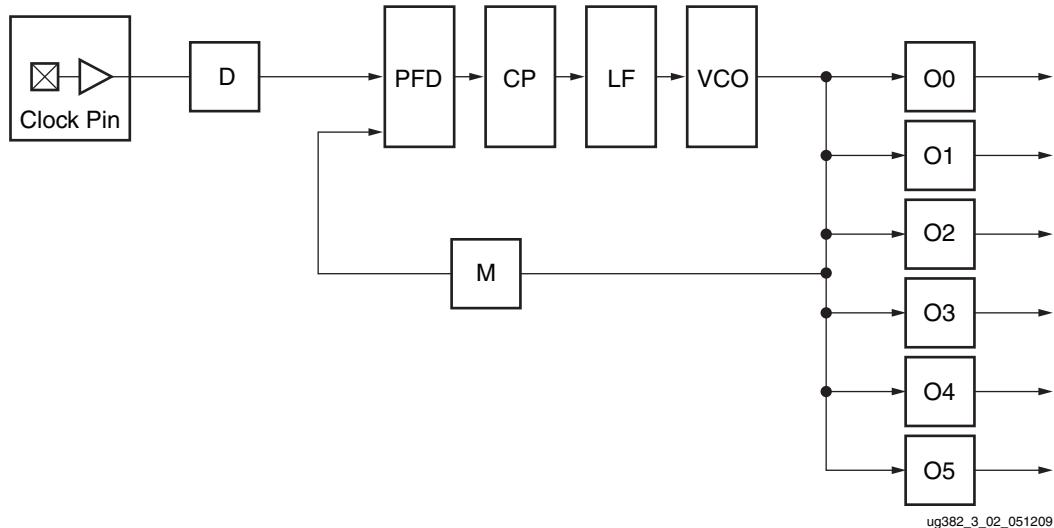
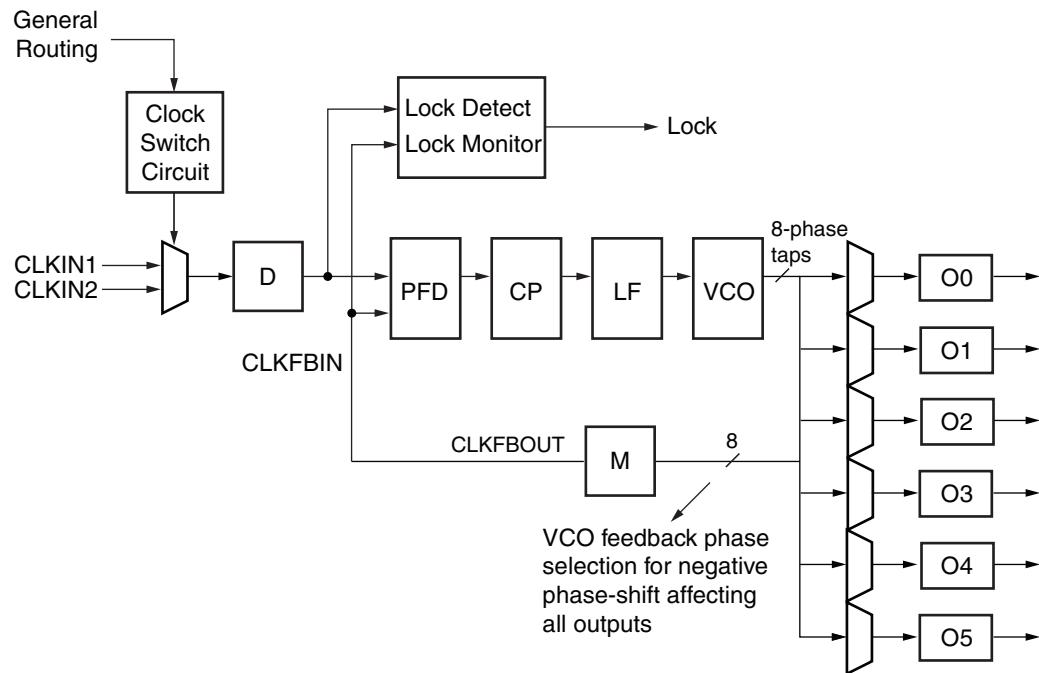
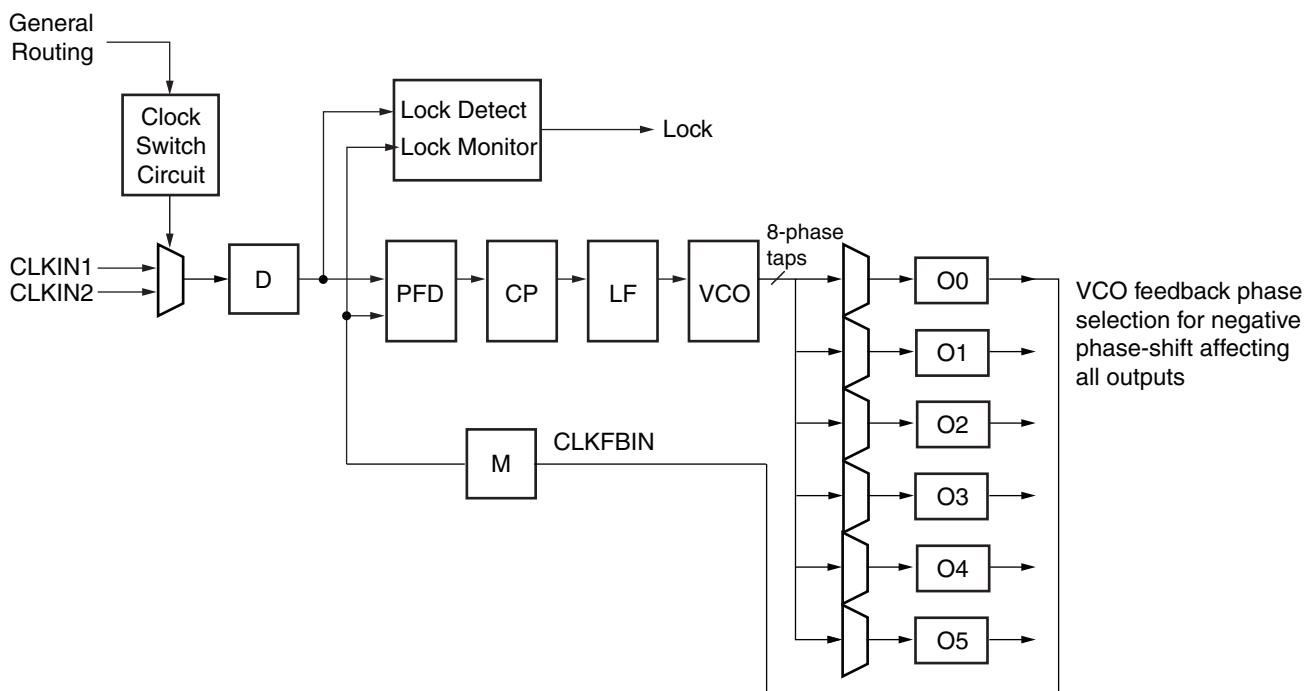


Figure 3-2: Block Diagram of the Spartan-6 FPGA PLL

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 Voltage Controlled Oscillator (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, which decreases 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. See [Figure 3-3](#) and [Figure 3-4](#). 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.



ug382_c3_03a_020510

Figure 3-3: Detailed PLL Block Diagram: CLK_FEEDBACK = CLKFBOUT

ug382_c3_04new_021110

Figure 3-4: Detailed PLL Block Diagram: CLK_FEEDBACK = CLKOUT0

Aligning PLL using CLK_FEEDBACK and BUFI02FB

The Spartan-6 FPGA PLL contains dedicated feedback routing used to minimize phase noise and increase the performance of the PLL clocking beyond the BUFG performance limitations. To use this dedicated PLL routing, CLK_FEEDBACK must be set to CLKOUT0 and use the BUFPPLL and BUFI02FB as shown in Figure 3-5.

Note: BUFPPLL and BUFPPLL_MCB input clocks can be connected to either CLKOUT0 or CLKOUT1 from the PLL.

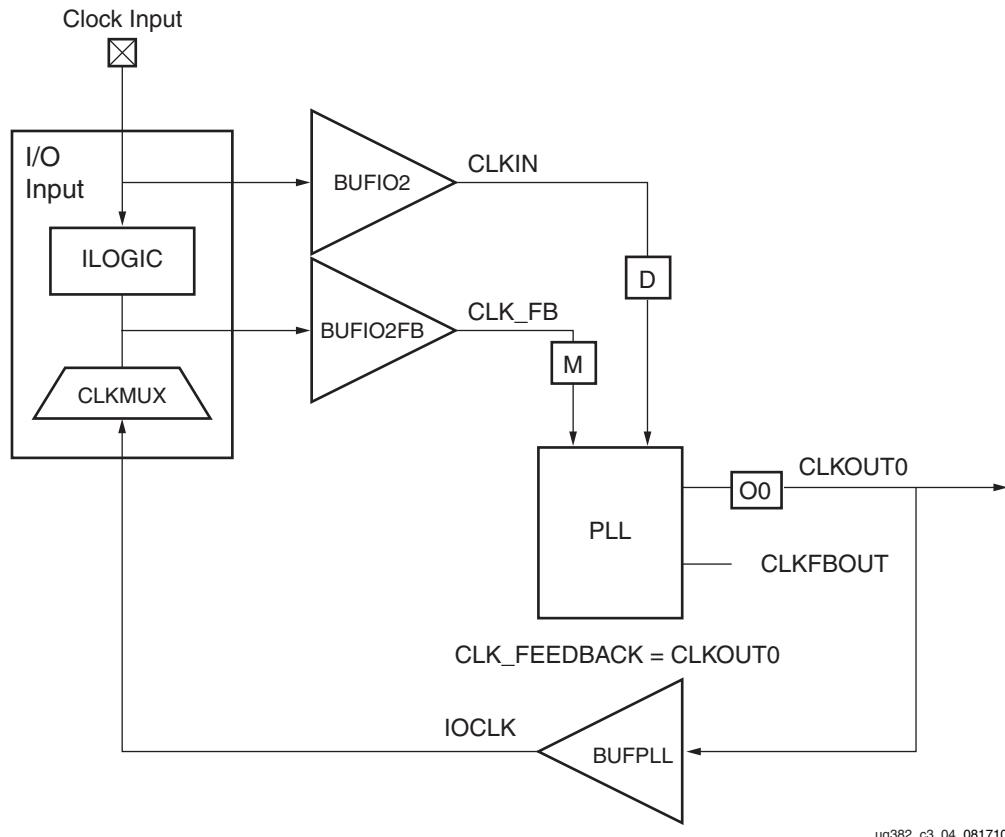


Figure 3-5: PLL Using CLKOUT0 Feedback

In high-speed source-synchronous designs, the feedback clock runs at the IOCLK frequency. To allow the feedback clock to match the input clock, the feedback clock is further divided using the CLKFBOUT_MULT as shown in Figure 3-4. Both CLKOUT0_DIVIDE and CLKFBOUT_MULT affect the VCO frequency.

$$f_{VCO} = \frac{f_{IN} \times CLKFBOUT_MULT \times CLKOUT0_DIVIDE}{DIVCLK_DIVIDE} \quad \text{Equation 3-1}$$

To accurately deskew the input routing, the BUFI02FB buffer must be used as shown in Figure 1-40, page 56. The BUFI02FB is matched to the BUFI02 buffer limiting deskewing to a single PLL.

When using CLK_FEEDBACK = CLKOUT0, the frequency for the CLKFBIN can be different than the frequency at the PFD. Be careful when setting CLK_FEEDBACK = CLKOUT0, because the PFD matches the frequency between the input clock ([Equation 3-2](#)) and the frequency from the feedback clock.

$$F_{PFD_CLKIN} = \frac{F_{CLKIN}}{DIVCLK_DIVIDE} \quad \text{Equation 3-2}$$

When CLK_FEEDBACK = CLKFBOUT, the frequency of CLKFBOUT ([Equation 3-3](#)) matches the feedback frequency for the PFD.

$$F_{CLKFBOUT} = \frac{F_{VCO}}{CLKFBOUT_MULT} \quad \text{Equation 3-3}$$

$$F_{PFD_CLKFBOUT} = F_{CLKFB} \quad \text{Equation 3-4}$$

When CLK_FEEDBACK = CLKOUT0, the output frequency for CLKOUT0 ([Equation 3-5](#)) can be different than the feedback frequency depending on CLKFB_MULT. As shown in [Figure 3-4](#), the frequency for CLKOUT0 is divided by CLKFB_MULT resulting in a PFD frequency that is dependent on CLKFB_MULT ([Equation 3-6](#)).

$$F_{CLKOUT0} = \frac{F_{VCO}}{CLKOUT0_DIVIDE} \quad \text{Equation 3-5}$$

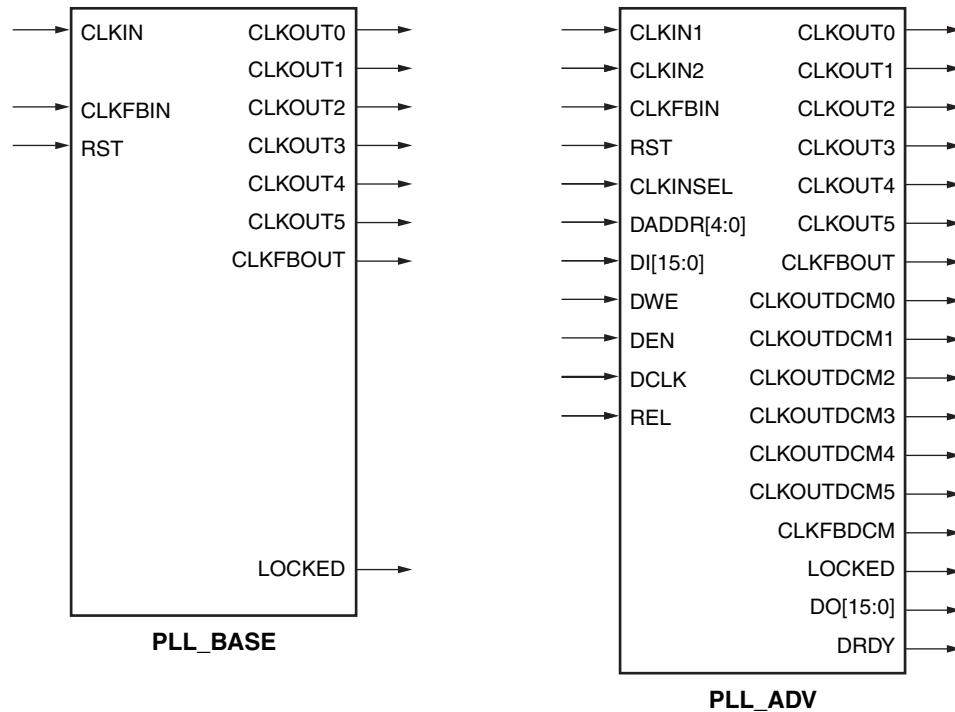
$$F_{PFD_CLKOUT0} = \frac{F_{CLKOUT0}}{CLKFB_MULT} \quad \text{Equation 3-6}$$

From [Equation 3-5](#), the output frequency for CLKOUT0 can be different than the VCO frequency.

General Usage Description

PLL Primitives

The two Spartan-6 FPGA PLL primitives, PLL_BASE and PLL_ADV, are shown in [Figure 3-6](#).



ug382_c3_05_060710

Figure 3-6: 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 3-2](#).

Table 3-2: 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. PLL_ADV is provided for designs that dynamically reconfigure the PLL. For most design situations, use the PLL_BASE primitive or the clocking wizard. The ports are listed in [Table 3-3](#).

Table 3-3: PLL_ADV Ports

| Description | Port |
|------------------------|--|
| Clock Input | CLKIN1, CLKIN2, CLKFBIN, DCLK |
| Control and Data Input | RST, CLKINSEL, (Static 1 or Static 0), DWE, DEN, DADDR, DI |
| Clock Output | CLKOUT0 to CLKOUT5, CLKFBOUT, CLKOUTDCM0 to CLKOUTDCM5, CLKFBDCM |
| Status and Data Output | LOCKED |

The Spartan-6 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 VCO operating frequency can be determined by using the following relationships:

$CLK_FEEDBACK = CLKFBOUT$ is described in [Equation 3-7](#).

$$F_{VCO} = F_{CLKIN} \times \frac{M}{D} \quad \text{Equation 3-7}$$

$CLK_FEEDBACK = CLKOUT0$ is described in [Equation 3-8](#).

$$F_{VCO} = F_{CLKIN} \times \frac{M \times O_0}{D} \quad \text{Equation 3-8}$$

where the M, D, and O counters are shown in [Figure 3-3](#). O0 affects the VCO only when $CLK_FEEDBACK = CLKOUT0$. [Equation 3-9](#) shows the output frequency for $CLKOUT[5:0]$. [Equation 3-10](#) shows the output frequency for $CLKFBOUT$.

$$F_{OUT} = \frac{F_{VCO}}{O} \quad \text{Equation 3-9}$$

$$F_{OUT_CLKFBOUT} = \frac{F_{VCO}}{M} \quad \text{Equation 3-10}$$

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. Spartan-6 FPGA PLLs support this feature. A clock output matching the reference clock CLKIN frequency (usually CLKFBOUT or CLKOUT) 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.

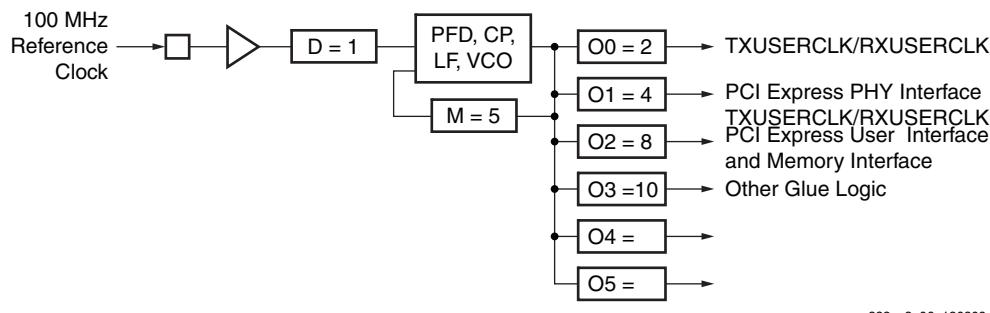
To accurately deskew the input routing, the BUFI02FB buffer must be used as shown in [Figure 1-40](#).

Frequency Synthesis Only

PLLs can also be used for stand-alone frequency synthesis. In this application, a PLL can not be used to deskew a clock network, however, it is used 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 3-7](#) shows the PLL configured as a frequency synthesizer. In this example, a clocking configuration for PCI Express x1 Gen1 is given. A 100 MHz reference clock is fed from the REFCLKOUT of the GTP transceiver. Setting the counters M = 5 and D = 1 makes the VCO oscillate at 500 MHz (100 MHz x 5). Ensure the VCO frequency meets the range specified in the Spartan-6 FPGA data sheet. Four of the six PLL outputs are programmed to provide:

- 250 MHz clock to the GTP transceiver's TXUSRCLK and RXUSRCLK
- 125 MHz clock for the PCI Express PHY interface as well as the GTP transceiver's TXUSRCLK2 and RXUSRCLK2 for data exchange in a 2-byte mode
- 62.5 MHz clock for the PCI Express User Interface and block RAM interface
- 50 MHz clock for other glue logic

In this example, no phase relationships between the reference clock and the output clocks are required, however, there are phase relationships required between the output clocks.



ug382_c3_06_120809

Figure 3-7: 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., $F_{IN} = 100 \text{ MHz}$, $F_{OUT} = 100 \text{ MHz}$). 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 *Spartan-6 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 *Spartan-6 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 *Spartan-6 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 clocking 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 implemented in two ways, directly through the GUI interface (the clocking 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 $128 \times 128 \times 128 = 2,097,152$ 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 19 MHz, then D can only go from 1 to 5. For D = 1, M can only have values from four to 10. If D = 2, M can have values from 8 to 20. In addition, D = 1 M = 4 is a subset of D = 2 M = 8 allowing the D = 2 M = 8 case to be dropped.

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:

$$D_{MIN} = \text{roundup} \frac{f_{IN}}{f_{PFD\ MAX}} \quad \text{Equation 3-11}$$

$$D_{MAX} = \text{rounddown} \frac{f_{IN}}{f_{PFD\ MIN}} \quad \text{Equation 3-12}$$

$$M_{MIN} = \left(\text{roundup} \frac{f_{VCOMIN}}{f_{IN}} \right) \times D_{MIN} \quad \text{Equation 3-13}$$

$$M_{MAX} = \text{rounddown} \frac{D_{MAX} \times f_{VCOMAX}}{f_{IN}} \quad \text{Equation 3-14}$$

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.

$$M_{IDEAL} = \frac{D_{MIN} \times f_{VCOMAX}}{f_{IN}} \quad \text{Equation 3-15}$$

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 3-4 summarizes the PLL ports. Table 3-5 lists the PLL attributes.

Table 3-4: PLL Ports

| Pin Name | I/O | Pin Description |
|-------------------------------|--------|---|
| CLKIN | Input | General clock input. |
| CLKIN1 | Input | PLL_ADV pin used for retargeting. General clock input. |
| CLKIN2 | Input | PLL_ADV pin used for retargeting. Secondary clock input. |
| CLKFBIN | Input | Feedback clock input. |
| CLKINSEL | Input | PLL_ADV pin used for retargeting. Connect to a static High or static Low to control the choice of clock input for PLL_ADV. 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 | PLL_ADV dynamic reconfiguration address (DADDR) input bus. Provides reconfiguration address. When not used, all bits must be assigned zeros. |
| DI[15:0] | Input | PLL_ADV dynamic reconfiguration data input (DI) bus. Provides reconfiguration data. When not used, all bits must be assigned zeros. |
| DWE | Input | PLL_ADV dynamic reconfiguration write enable (DWE) bus. Provides the enable control signal to access the dynamic reconfiguration feature. When not used, DWE must be tied to zero. |
| DEN | Input | PLL_ADV dynamic reconfiguration enable (DEN) input. Provides the enable control signal to access the dynamic reconfiguration feature. When not used, DEN must be tied to zero. |
| DCLK | Input | PLL_ADV dynamic reconfiguration clock (DCLK) input. Provides reference clock for the dynamic reconfiguration port. When using a global clock buffer, only the upper eight BUFGMUXs can drive DCLK: BUFGMUX_X2Y1, BUFGMUX_X2Y2, BUFGMUX_X2Y3, BUFGMUX_X2Y4, BUFGMUX_X3Y5, BUFGMUX_X3Y6, BUFGMUX_X3Y7, and BUFGMUX_X3Y8. |
| REL | Input | PLL_ADV reserved pin. |
| 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. ⁽²⁾ |
| CLKFBOUT | Output | Dedicated PLL feedback output. ⁽²⁾ |
| CLKOUTDCM[0:5] ⁽¹⁾ | Output | PLL_ADV pin used for retargeting. User configurable clocks (0 through 5) that can only connect to the DCM within the same CMT as the PLL. ⁽²⁾ |
| CLKFBDCM | Output | PLL_ADV pin used for retargeting. 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. ⁽²⁾ |

Table 3-4: PLL Ports (Cont'd)

| Pin Name | I/O | Pin Description |
|----------|--------|---|
| LOCKED | Output | Asynchronous 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 within one PFD clock cycle 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. The FPGA waits for all DCMs and PLLs to be locked when LCK_CYCLE is set to control the startup cycles without setting the STARTUP_WAIT attribute on any DCM_SP port. In the starting configuration sequence, GTS must be deasserted for the PLL to lock. |
| DO[15:0] | Output | PLL_ADV dynamic reconfiguration output data bus. |
| DRDY | Output | PLL_ADV dynamic reconfiguration ready output (DRDY). Provides the response to 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.
2. PLL clock outputs must use either a horizontal clock (default) or a global clock buffer.

PLL Attributes

Table 3-5: PLL Attributes

| Attribute | Type | Allowed Values | Default | Description |
|--------------------|---------|--|--------------------|---|
| COMPENSATION | String | SYSTEM_SYNCHRONOUS SOURCE_SYNCHRONOUS EXTERNAL | 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. EXTERNAL is used to compensate by routing the clock external to the FPGA. Additional attributes automatically selected by the ISE software: INTERNAL 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⁾ | 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. |

Table 3-5: PLL Attributes (Cont'd)

| Attribute | Type | Allowed Values | Default | Description |
|------------------------|---------|-----------------|---------|---|
| 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). When setting CLK_FEEDBACK = CLKOUT0, phase shifting results in a negative phase shift of all remaining clock outputs. |
| 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. |
| 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. |

Table 3-5: PLL Attributes (Cont'd)

| Attribute | Type | Allowed Values | Default | Description |
|-----------------------|--------|------------------------------------|----------|--|
| CLK_FEEDBACK | String | CLKFBOUT or CLKOUT0 ⁽¹⁾ | CLKFBOUT | Specifies the clock source to drive CLKFB_IN. See Figure 3-5, page 96 for correct usage of feedback resources and calculating VCO frequency. |
| RESET_ON_LOSS_OF_LOCK | String | FALSE | FALSE | Must be set to FALSE, not supported in silicon. |

Notes:

- When CLK_FEEDBACK = CLKOUT0, CLKOUT0_DIVIDE is further restricted to ensure valid PFD and VCO frequencies. CLKOUT0_DIVIDE * CLKFBOUT_MULT must be 1 to 64.

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.
- BUFG - 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.
- BUFIO2 - When used with IBUFG, the DIVCLK output directly connects to the PLL clock input. The PLL compensates for this delay.

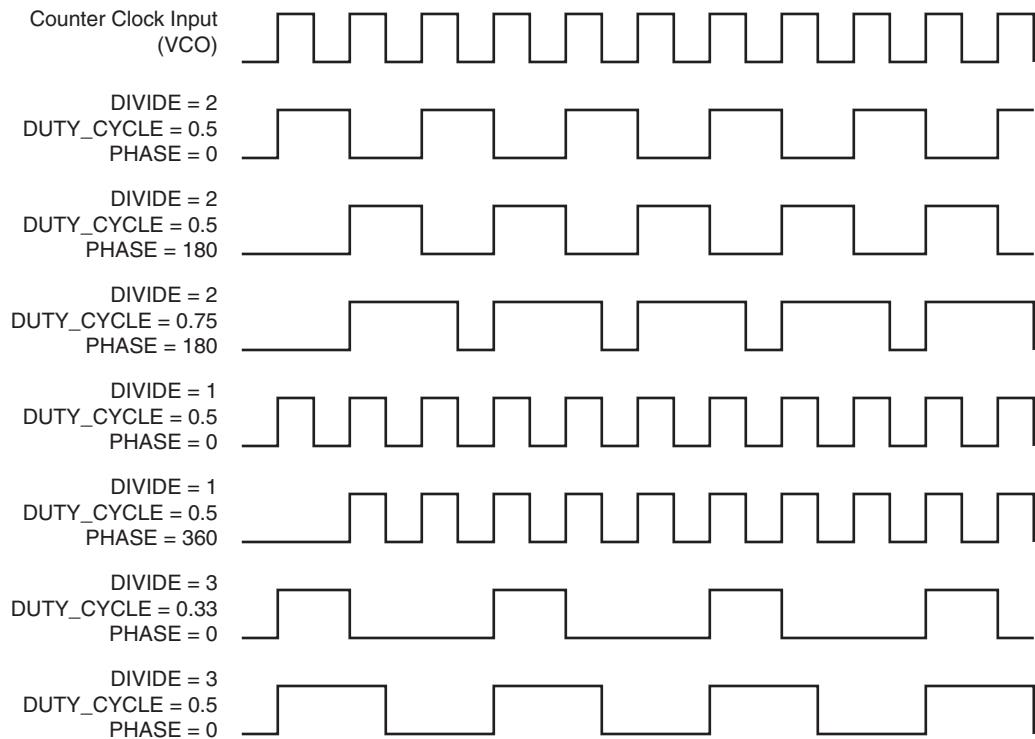
When global clock inputs are used to connect to a PLL, a BUFIO2 clock buffer will be inferred for optimal performance, as shown in [Figure 1-16, page 33](#). Each BUFIO2 either routes to the CMT on the top-half or the bottom-half of the device. The inferred BUFIO2 buffer can restrict routing to ensure proper phase alignment.

BUFIO2 buffers from BUFIO2 clocking regions TL, TR, RT, and LT route to the DCM/PLL on the top half of the device. Similarly, BUFIO2 buffers from BUFIO2 clocking regions BL, BR, RB, and LB connect to the DCM/PLL on the bottom half of the device.

Counter Control

The PLL output counters provide a wide variety of synthesized clocks using a combination of DIVIDE, DUTY_CYCLE, and PHASE. [Figure 3-8](#) illustrates how the counter settings impact the counter output.

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



ug382_c3_07_080309

Figure 3-8: 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. [Figure 3-9](#) 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).

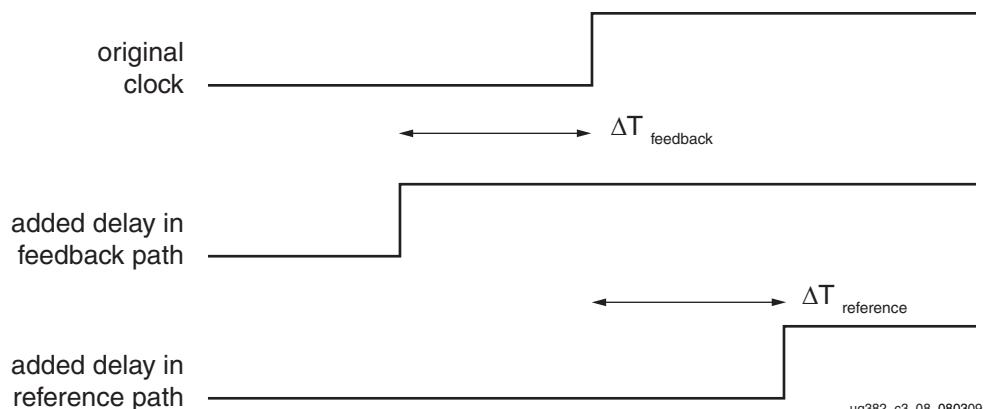
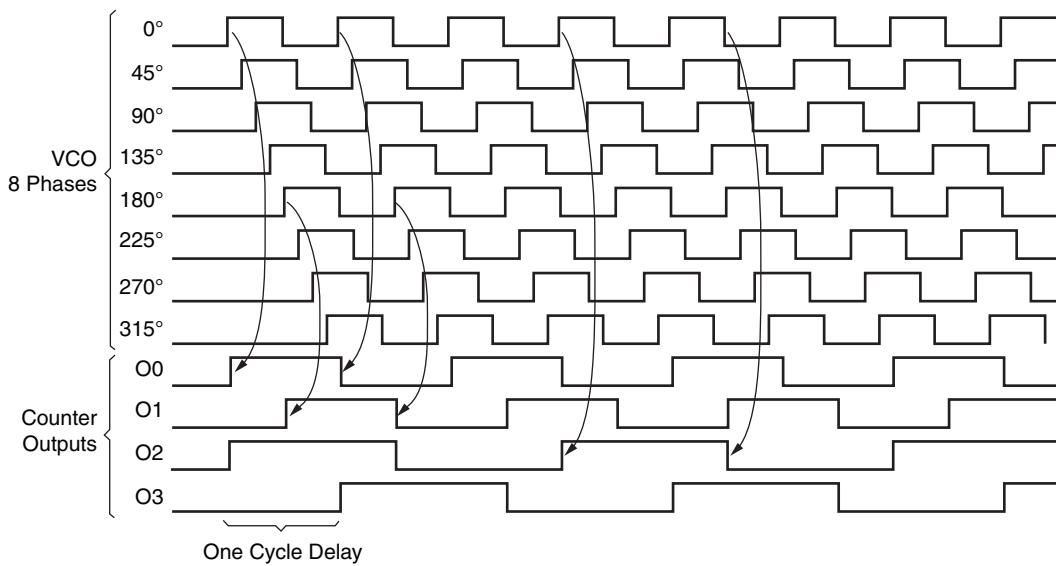


Figure 3-9: Basic Output Clock Shifting

Detailed VCO and Output Counter Waveforms

Figure 3-10 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.



ug382_c3_09_080309

Figure 3-10: 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 clocking wizard determines the proper settings through the PLL attributes and Wizard inputs.

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 clocking 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 merged 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 3-11](#) 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 3-11](#)).

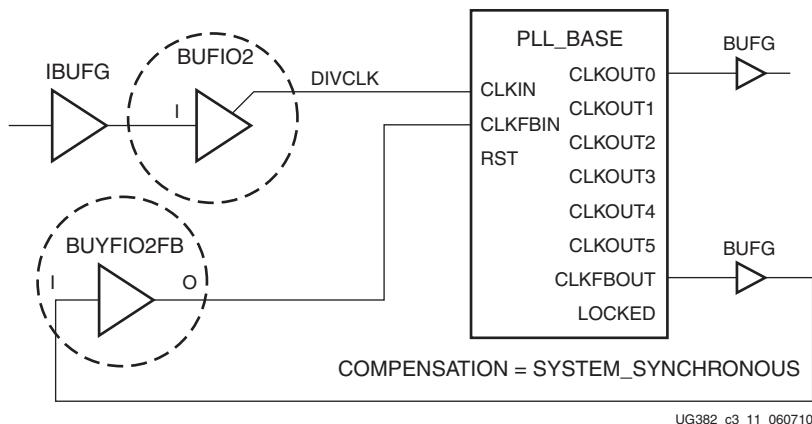


Figure 3-11: Clock Deskew Using Two BUFGs

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:

$$\frac{f_{IN}}{D} = f_{FB} = \frac{f_{VCO}}{M} \quad \text{Equation 3-16}$$

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 3-12).

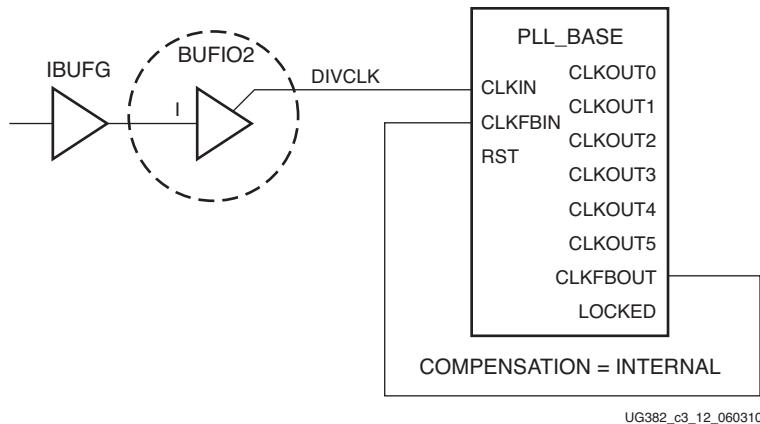


Figure 3-12: **PLL with Internal Feedback**

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 Figure 3-13 for single-ended clocks. In Spartan-6 FPGAs, the BUFIO2FB and BUFIO2 placements must be matched and the feedback clock must be placed on the differential pair of the GCLK pin being used. In the case of a differential clock input where both pins of the differential clock are already being used, Figure 3-14 shows a circuit description using the equivalent GCLK in the neighboring BUFIO2 clocking region.

Note: There is an additional inversion of the OB output of the IBUFGDS_DIFF_OUT. This circuit will directly connect the differential GCLK to the matched BUFIO2FB.

The feedback signal is driven off the devices 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. In this example, CLK_FEEDBACK = CLKFBOUT and the output logic is matched to a BUFG driving output logic with all signals using a single-ended I/O standard.

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

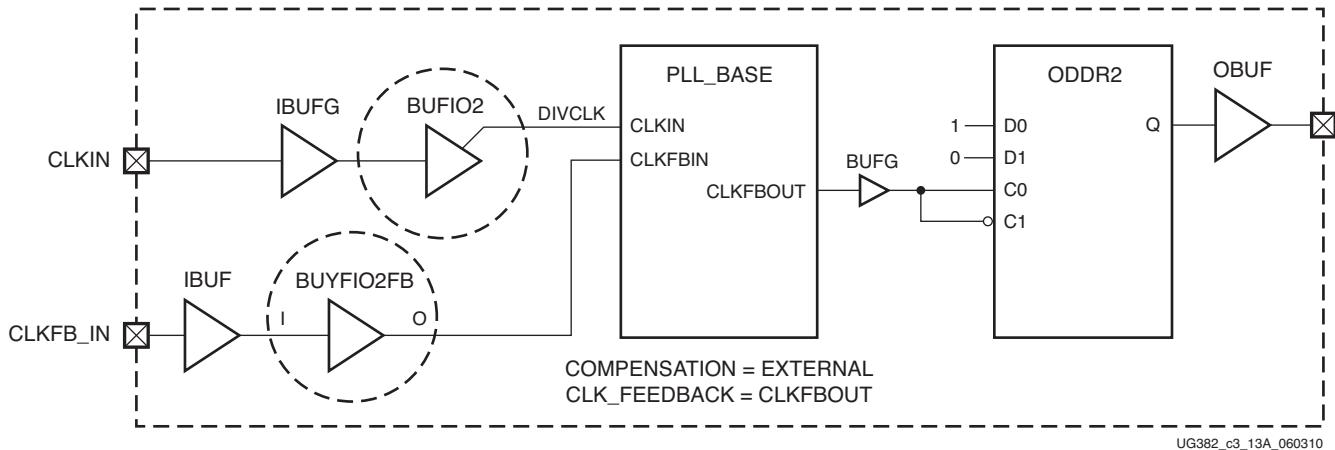


Figure 3-13: Zero Delay Buffer for Single-Ended Clocks

Note: When planning pinouts for a design using external feedback, the tools automatically match the BUFIO2FB and BUFIO2 to ensure optimal phase alignment.

For optimal placement, the input pad for the feedback clock must be placed on the differential pair. For example, if GCLK19 is used for CLKIN, then use GCLK18 for the CLKFB_IN. See [Table 1-6](#) for a complete placement list of global clock input pins.

As shown in [Figure 3-14](#), a BUFIO2 from an adjacent BUFIO2 clocking region is used if either the input clock or the feedback clock are differential clocks. For example, since CLKIN_P and CLKIN_N are connected to GCLK19 and GCLK18 respectively, then CLKFB_IN_P is connected to GCLK15 and CLKFB_IN_N is connected to GCLK14.

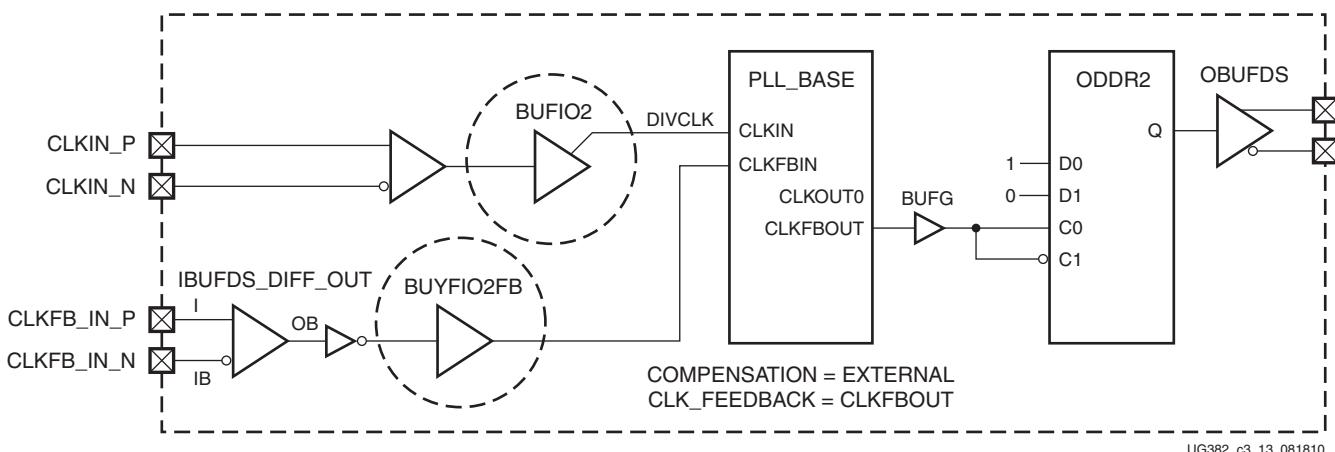


Figure 3-14: Zero Delay Buffer for Differential Clocks

For a differential feedback clock, IBUFGDS_DIFFOUT and an additional logic inversion must be used (as shown in [Figure 3-14](#)). The differential feedback clock pins are directly connected to the BUFI02FB. An additional logic inversion of the BUFI02FB input is used to compensate for the use of the OB output of the IBUFGDS_DIFFOUT. Verilog and VHDL examples are described in this section.

Differential BUFI02FB Zero Delay Buffer Example (Verilog)

```

IBUFDS_DIFF_OUT INST_IBUFDS_DIFF_OUT (
    .I      (CLKFB_IN_P),
    .IB     (CLKFB_IN_N),
    .OB     (CLKFB_IBUFDS_OB)) ;

BUFI02FB INST_BUFI02FB (
    .I   (~ CLKFB_IBUFDS_OB),
    .O   (CLK_FEEDBACK_TO_PLL));

```

Differential BUFI02FB Zero Delay Buffer Example (VHDL)

```

I_IBUFGDS: IBUFGDS_DIFF_OUT PORT MAP(
    I => CLKFB_IN_P, IB =>
    CLKFB_IN_N, OB => CLKFB_OB);

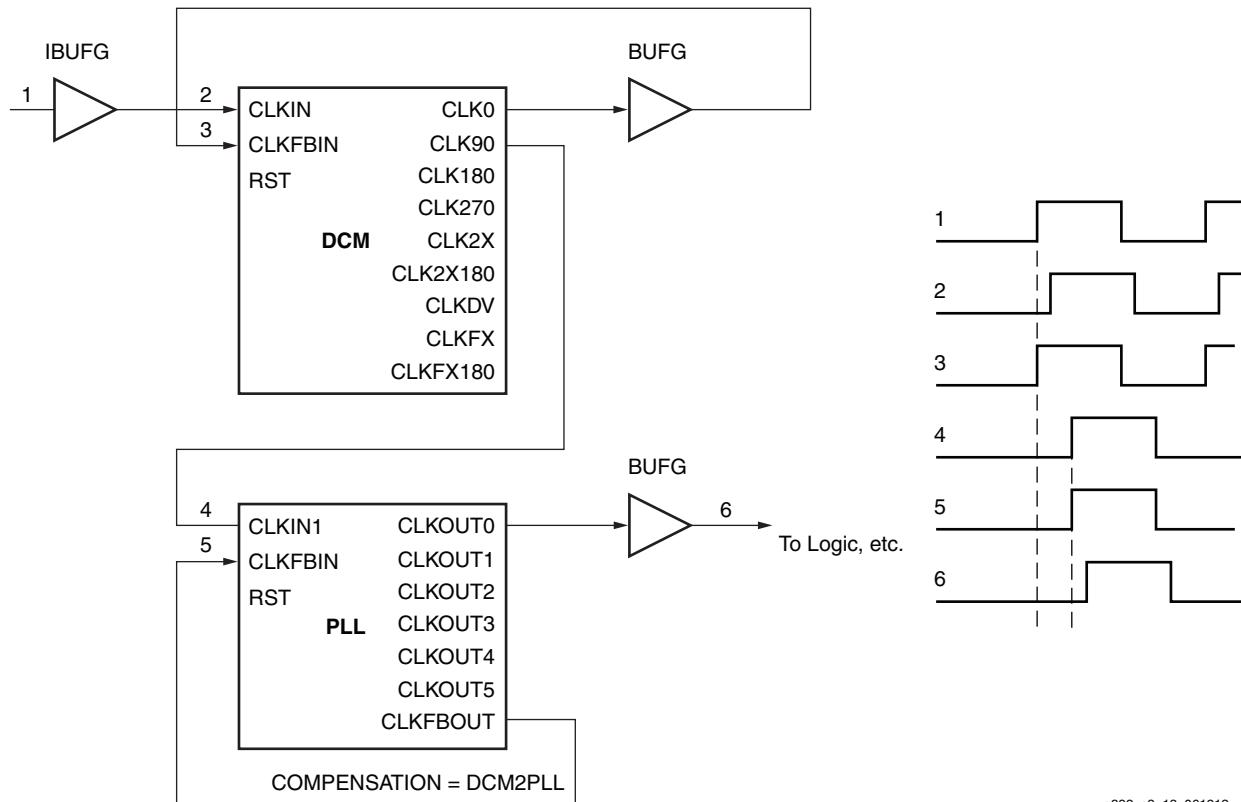
CLKFB_OB_180 <= not CLKFB_OB;

I_BUFI02FB: BUFI02FB PORT MAP (
    I => CLKFB_OB_180,
    O => CLKFB_BUFI02FB);

```

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 3-15](#). 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.



ug382_c3_13_061212

Figure 3-15: 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 3-16, one PLL drives two DCMs.

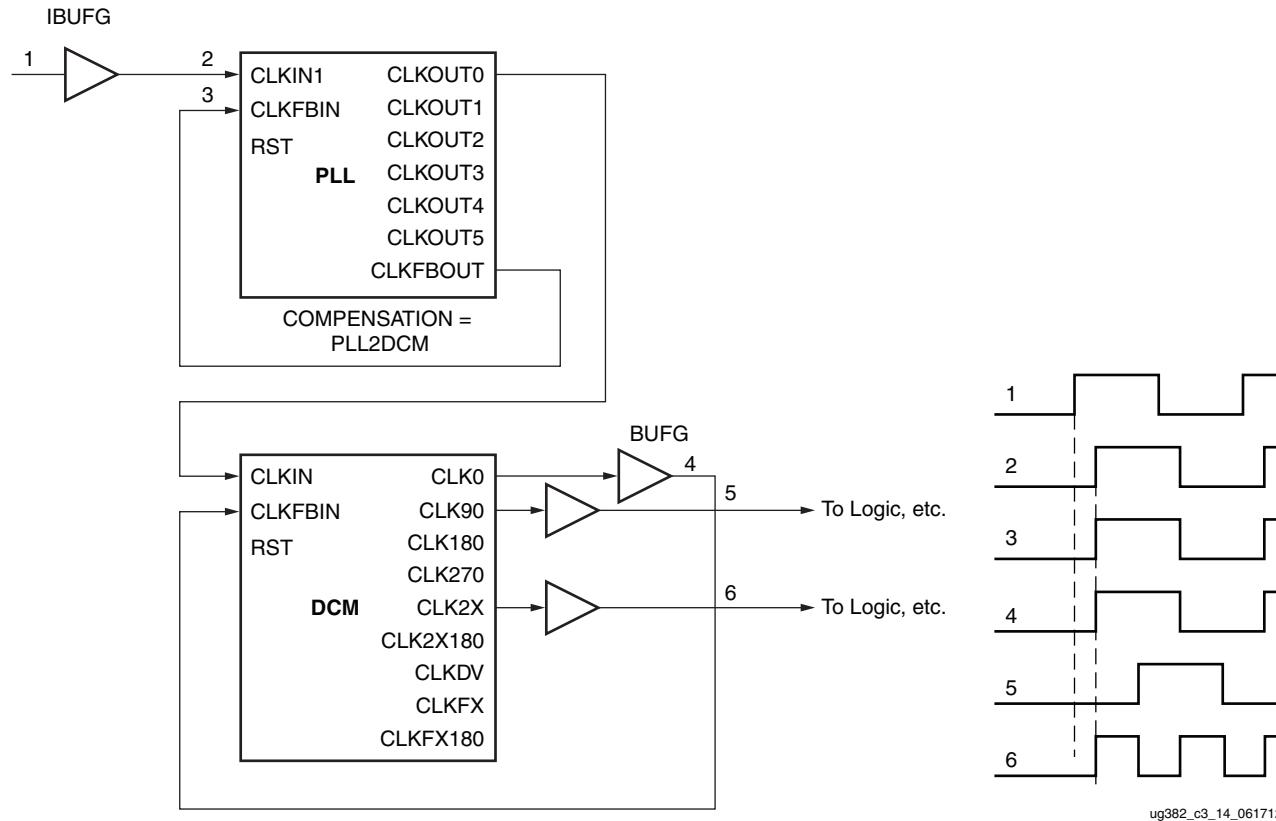


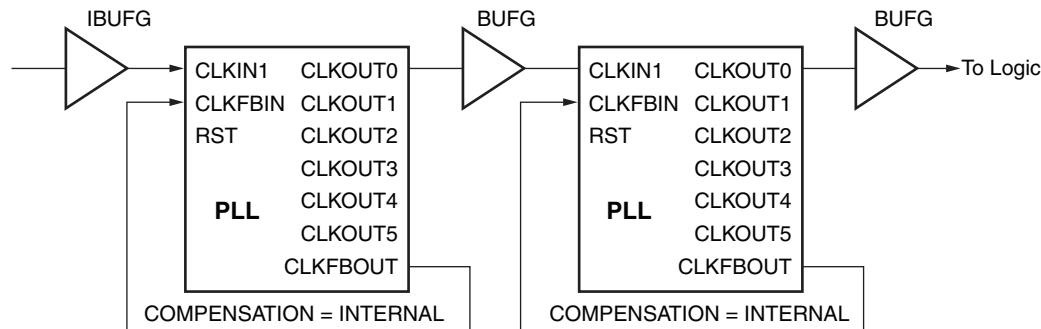
Figure 3-16: PLL Driving a DCM

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 3-17](#) shows the relationship between the final output frequency and the input frequency and counter settings of the two PLLs ([Figure 3-17](#).) 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.

$$f_{OUTPLL2} = f_{OUTPLL1} \frac{M_{PLL2}}{D_{PLL2} \times O_{PLL2}} = f_{IN} \frac{M_{PLL1}}{D_{PLL1} \times O_{PLL1}} \times \frac{M_{PLL2}}{D_{PLL2} \times O_{PLL2}}$$

Equation 3-17



ug382_c3_15_060710

Figure 3-17: Cascading Two PLLs

Dynamic Reconfiguration Port

Details of the supported Spartan-6 FPGAs PLL DRP operations are described in [XAPP879, PLL Dynamic Reconfiguration](#).

Application Guidelines

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

Spartan-6 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:

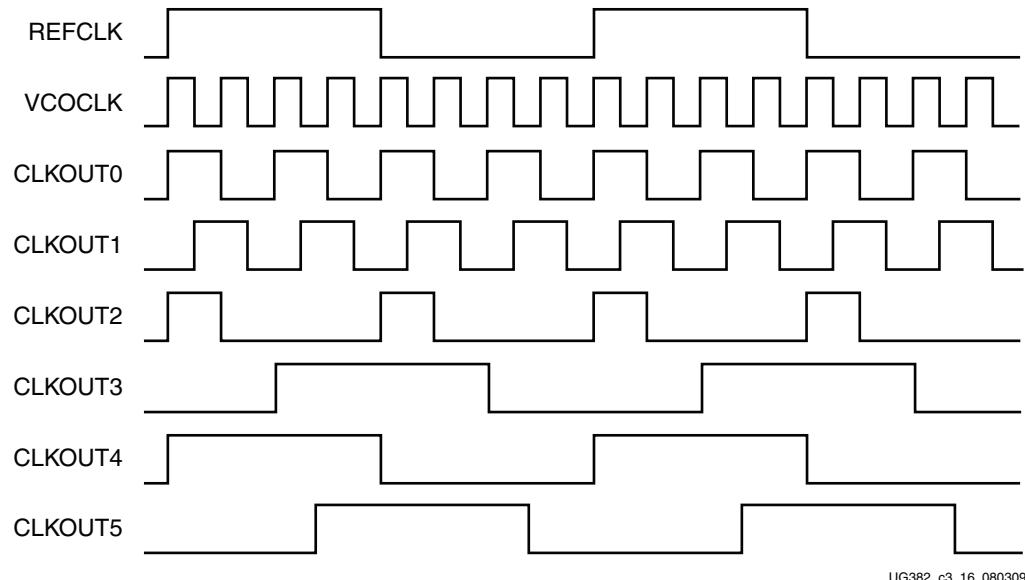
```

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 3-18 displays the resulting waveforms.



UG382_c3_16_080309

Figure 3-18: Example Waveform