Versal ACAP Clocking Resources

Architecture Manual

AM003 (v1.1) December 7, 2020

XILINX®

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
12/07/2020 Version 1.1	
Clocking Resources Features	Updated to add information about RAMs in a full clock region, as well as updated figure to remove CMAC.
Differences from Previous Generations	Updated deskew description for XPLL.
Using the Deskew Logic	Added details about using digital skew with external clock network delay.
Limitations	Added device specific limitations for the DPLL.
Deskew Logic - Additional Deskew Options	Clarified description.
Functioning of Deskew	Added requirements related to clock input source.
Phase Align Selected Clocks in Different ACAPs	Added clock input requirements.
07/16/2020 Version 1.0	
Initial release.	N/A

Table of Contents

XILINX

Chapter 1

Overview

Introduction to Versal ACAP

Versal™ adaptive compute acceleration platforms (ACAPs) combine Scalar Engines, Adaptable Engines, and Intelligent Engines with leading-edge memory and interfacing technologies to deliver powerful heterogeneous acceleration for any application. Most importantly, Versal ACAP hardware and software are targeted for programming and optimization by data scientists and software and hardware developers. Versal ACAPs are enabled by a host of tools, software, libraries, IP, middleware, and frameworks to enable all industry-standard design flows.

Built on the TSMC 7 nm FinFET process technology, the Versal portfolio is the first platform to combine software programmability and domain-specific hardware acceleration with the adaptability necessary to meet today's rapid pace of innovation. The portfolio includes six series of devices uniquely architected to deliver scalability and AI inference capabilities for a host of applications across different markets—from cloud—to networking—to wireless communications—to edge computing and endpoints.

The Versal architecture combines different engine types with a wealth of connectivity and communication capability and a network on chip (NoC) to enable seamless memory-mapped access to the full height and width of the device. Intelligent Engines are SIMD VLIW AI Engines for adaptive inference and advanced signal processing compute, and DSP Engines for fixed point, floating point, and complex MAC operations. Adaptable Engines are a combination of programmable logic blocks and memory, architected for high-compute density. Scalar Engines, including Arm® Cortex™-A72 and Cortex-R5F processors, allow for intensive compute tasks.

The Versal AI Core series delivers breakthrough AI inference acceleration with AI Engines that deliver over 100x greater compute performance than current server-class of CPUs. This series is designed for a breadth of applications, including cloud for dynamic workloads and network for massive bandwidth, all while delivering advanced safety and security features. AI and data scientists, as well as software and hardware developers, can all take advantage of the high-compute density to accelerate the performance of any application.

The Versal Prime series is the foundation and the mid-range of the Versal platform, serving the broadest range of uses across multiple markets. These applications include 100G to 200G networking equipment, network and storage acceleration in the Data Center, communications test equipment, broadcast, and aerospace & defense. The series integrates mainstream 58G transceivers and optimized I/O and DDR connectivity, achieving low-latency acceleration and performance across diverse workloads.

The Versal Premium series provides breakthrough heterogeneous integration, very high-performance compute, connectivity, and security in an adaptable platform with a minimized power and area footprint. The series is designed to exceed the demands of high-bandwidth, compute-intensive applications in wired communications, data center, test & measurement, and other applications. Versal Premium series ACAPs include 112G PAM4 transceivers and integrated blocks for 600G Ethernet, 600G Interlaken, PCI Express® Gen5, and high-speed cryptography.

The Versal architecture documentation suite is available at: https://www.xilinx.com/versal.

Navigating Content by Design Process

Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. This document covers the following design processes:

- **System and Solution Planning:** Identifying the components, performance, I/O, and data transfer requirements at a system level. Includes application mapping for the solution to PS, PL, and AI Engine. Topics in this document that apply to this design process include:
  - Chapter 1: Overview: provides an overview of clock routing and clock distribution resources and includes:
    - Clocking Resources Features
  - Chapter 2: Versal Architecture Clocking Resources: describes the clocking routing resources to support the various clocking schemes.
- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, subsystem functional simulation, and evaluating the Vivado® timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
  - Chapter 3: Clock Buffer Resources: describes the set of clock buffers that drive the routing and distribution resources across the entire device.
  - Chapter 4: Clock Management Functions: describes the clock functions to generate clocking for specific and general purposed tasks in ACAPs.

Clocking Resources Features

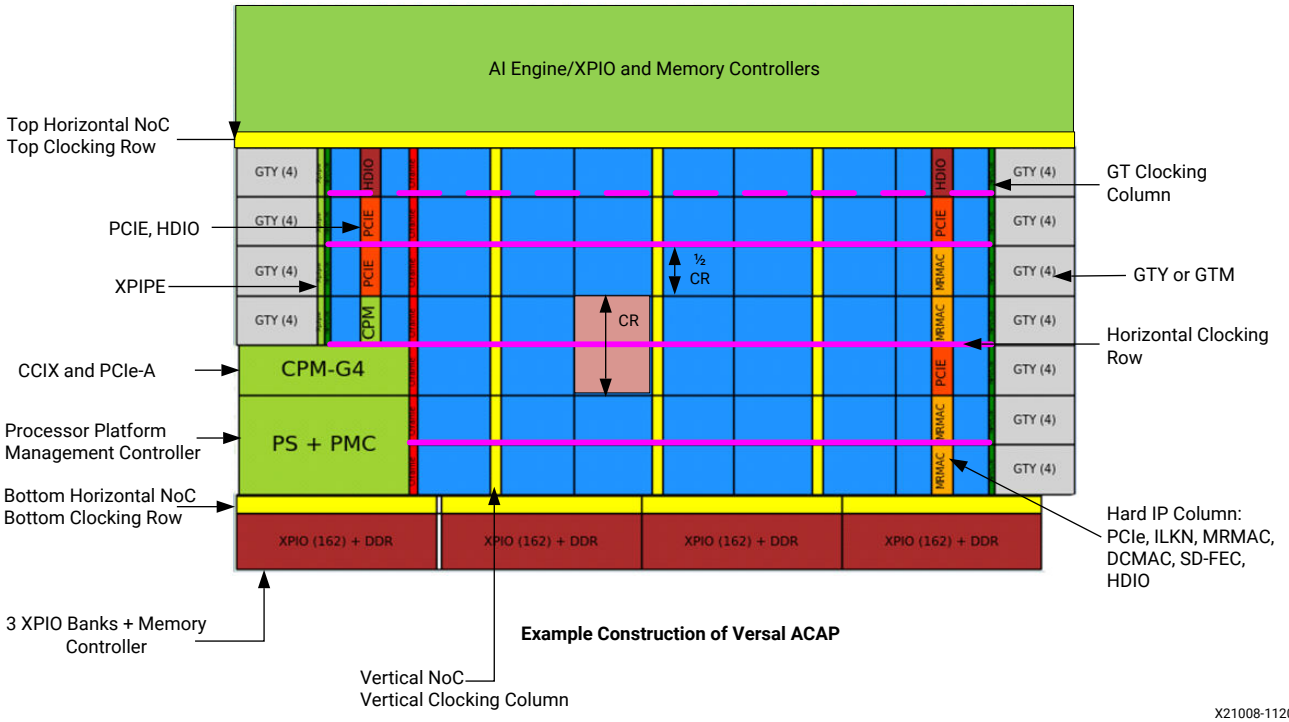
Xilinx® Versal™ architecture clocking resources manage complex and simple clocking requirements with dedicated global clocks distributed on clock routing and clock distribution resources. The global clocking network in Versal architecture is largely similar in structure to the UltraScale™ families. It is a bidirectional, segmented network of clock spines that support many independent clock networks, and allows the same structure to be used for low skew or low propagation clocks.

Most departures from the UltraScale clocking architecture are related to skew and jitter minimization implementation strategies that leave the base architecture intact. The clock management functions (MMCMs, XPLLs, and DPLLs) provide clock frequency synthesis, deskew, and jitter filtering functionality. Non-clock resources such as local routing are not recommended when designing for clock functions.

- The core of the device is subdivided into columns and rows of segmented clock regions (CRs). CRs are arranged in tiles. A CR contains configurable logic blocks (CLBs), DSP Engine, block RAMs, UltraRAMs, interconnect, and associated clocking. Gigabit transceivers (Type Y or M) are placed in the right and left most columns. In the lower left side resides the Arm® based processor system (PS) which also contains a platform management controller (PMC) that manages the overall system such as configuration, health-monitoring (root SYSMON), security, and various other tasks. In some devices, an interconnect for CCIX and PCIe® (CPM) sits on top of the PS+PMC block. Other hard block IPs such as PCIe, Interlaken (ILKN), 100G multirate Ethernet MAC (MRMAC), 600G Ethernet MAC (DCMAC), forward error correction (SD-FEC), and high-density I/Os (HDIOs) are arranged in a separate vertical hardblock IP columns. To support the network on a chip (NoC) concept horizontal NoC columns run on top and bottom of the core and vertical NoC columns are embedded in the core in regular intervals. The top or bottom of the core can have high-performance I/Os (XPIOs) or AI Engine on top only. Hard memory controllers are also located along with the XPIOs in the bottom. The mixture of the above features varies with Xilinx Versal ACAPs.
- Unlike the previous two generations, Versal devices implement full and half CRs. The height of a full CR is 96 CLBs, 48 DSP58s, typically 24 block RAMs, and 24 UltraRAMs with a horizontal clock spine (HCS) at its center. A half CR is half the height of a full CR and the HCS run in the bottom. The HCS contains the horizontal routing and distribution resources, leaf clock buffers, and clock network interconnections. Clock buffers drive into the HCS. If present, there are two banks of 22 HDIO each per CR in the HDIO.
- Vertical logic clock columns exist on the programmable logic interior. A CR is bound by vertical routing on one side and a BRAM column on the other. CRs are placed back-to-back with a vertical clocking column in between each pair. GT clocking columns are next to the GTs. The HCSs are in their traditional location in the center of each CR spanning the entire programmable logic region except in the half CR where they are at the bottom of the CR. The clocking drives vertical and horizontal connectivity through separate clock routing and clock distribution resources through the HCS.

- Two XPLLs are embedded in the center of each XPIO bank and directly connect to XPHY logic/XPIO and the horizontal clock row, thus driving the BUFG resources. XPLLs can be cascaded through a direct connection to support wider interfaces.
- DPLLs are located to the left and right of the GT clock columns. In devices with HDIO columns, DPLLs are also located next to those I/O columns.
- Mixed-mode clock managers (MMCMs) are located adjacent to the XPIO banks in a horizontal clock row with the various clock BUFG buffers such as BUFGCE, BUFGCTRL, and BUFGCE_DIV next to them. Each MMCM site also incorporates a DPLL next to it increasing the total MMCM equivalent count.
- BUFG_GTs reside in the same clock columns next to the GT block and drive horizontal and vertical clock routing.
- BUFG_PSs reside in the clock region next to the PS/PMC block and can drive vertical and horizontal clocking.
- Horizontal clock routing and distribution tracks drive horizontally into the CRs. Vertical routing and distribution tracks drive vertically adjacent CRs. The tracks are segmentable at the CR boundaries in both the horizontal and vertical directions. This allows for the creation of device-wide global clocks or local clocks of variable size.
- The distribution tracks drive the clocking of synchronous elements across the device. Distribution tracks are driven by routing tracks or directly by the clocking structures in the PHY.
- Versal architecture clock input pins drive XPLLs, BUFGs, DPLLs, and MMCMs.

Figure 1: **High-Level Architecture View**



Terms used in the previous figure:

- **NoC:** Network-on-chip.
- **XPIPE:** Physical connection between GTY and PCIe® in the CPM block.
- **HDIO/XPIO:** High-density I/O and high performance I/O.
- **CMAC:** 100GE MAC and PCS.
- **CPM:** Interconnect for CCIX and PCIe features of the PS subsystem.
- **CCIX:** Cache Coherent Interconnect for Accelerators.
- **ILKN:** Interlaken Hard-IP block.
- **MRMAC:** 100G multirate Ethernet MAC and PCS.
- **DCMAC:** 600G Ethernet MAC and PCS.
- **SD-FEC:** Software defined forward error correction.

Clock Buffers and Routing

Each XPIO I/O bank contains four global clock input pins (GCs) to bring user clocks onto the device clock management and routing resources. Every HDIO bank contains two GC pins. The global clock inputs bring user clocks onto:

- XPLLs in the same bank (XPHY) and adjacent banks.
- MMCM and DPLL next to the XPIO banks (XPHY).
- Any of the 24 BUFGCEs, 8 BUFGCTRLs, and 4 BUFGCE_DIVs co-located with the MMCMs/DPLLs.
- Two GC pins per HDIO bank can drive DPLLs and four clock buffers (only BUFGCE) next to them.

Each device has three general purpose global clock buffers: BUFGCTRL, BUFGCE, and BUFGCE_DIV. In addition, there is a local BUFDIV_LEAF clock buffer for driving leaf clocks from horizontal distribution to various blocks in the device. The BUFDIV_LEAF is a physical only buffer and cannot be instantiated by the user. BUFGCTRL has derivative software representations of types BUFGMUX, BUFGMUX1, BUFGMUX_CTRL, and BUFGCE_1. BUFGCE is for improved clock gating and has a software derivative BUFG (BUFGCE with clock enable tied High). The global clock buffers drive routing and distribution tracks into the device logic through the HCS rows. There are 12 routing and 24 distribution tracks in each HCS row. Bottom and top clocking rows have 24 routing and 24 distribution tracks. There is BUFG_PS that drives from PS through horizontal and vertical routing tracks and also BUFG_GT that generates divided clocks for GT clocking. The clock buffers:

- Can be used as a clock enable circuit to enable or disable clocks either globally, locally, or within a CR for fine-grained power control
- Can be used as a glitch-free multiplexer to:
  - Select between two clock sources
  - Switch away from a failed clock source
- Are often driven by an MMCM/XPLL/DPLL to:
  - Eliminate the clock distribution delay
  - Adjust clock delay relative to another clock

See Chapter 2: Versal Architecture Clocking Resources for further details on global clocks, I/O, and GT clocking. It also describes which clock routing resources to use for various applications.

In Versal ACAP, there is a new group of multi-clock buffers (MBUFG) that are similar to BUFGs but with multiple clock outputs. Refer to Multi-Clock Buffers.

Clock Management MMCM, XPLL, and DPLL

Each device has MMCMs, DPLLs, and XPLLs near the PHY next to each of the I/O banks. An MMCM block consists of one MMCM and one DPLL. Other DPLLs reside next to HDIO and GT columns. The MMCM is the primary block for frequency synthesis for a wide range of frequencies, and serves as a jitter filter for either external or internal clocks, and deskews clocks among a wide range of other functions. Similar in basic functionalities and capabilities to the MMCMs and XPLLs, the DPLLs provide clock output to the general interconnect of the Xilinx® Versal adaptive compute acceleration platform (ACAPs). It can also be used for deskewing applications. The primary purpose of the XPLL is to provide clocking to the PHY I/Os, but it can also be used for clocking other resources in the device in a limited fashion. The device clock input connectivity allows multiple resources to provide the reference clock(s) to the MMCM and XPLL.

MMCMs, DPLLs, and XPLLs have infinite fine phase shift capability in either direction and can be used in dynamic phase shift mode. MMCM and DPLL have 2^6 resolution fractional feedback counters to generate output frequencies that are non-integer multiples of the reference clock frequency.

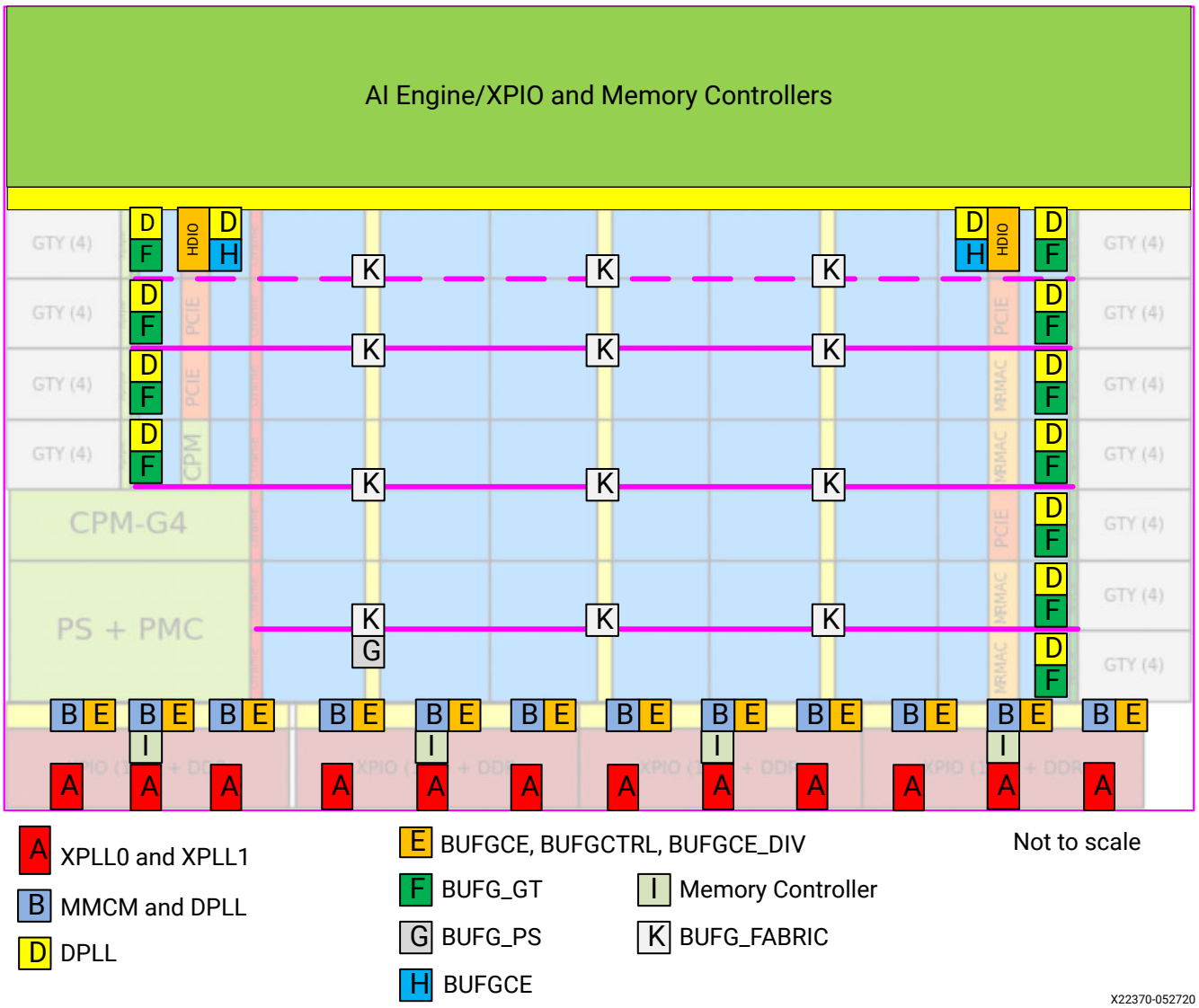
The LogiCORE™ IP clocking wizard is available to assist in using the MMCMs, DPLLs, and XPLLs to create clock networks in Xilinx Versal architecture designs. The GUI interface is used to collect clock network parameters. The clocking wizard chooses the appropriate clock resource and optimally configures the clock management resource and associated clock routing resources.

See Chapter 4: Clock Management Functions for further details on the MMCM, XPLL, and DPLL block features and connectivity.

Clock Trees and Buffers

- Vertical clocking spines are constructed as balanced binary trees. This operation equalizes delays to the horizontal spines for matching delays of the static clock networks.
- Active clock deskews minimize jitter through a calibrated skew compensation to minimize local and global skews both within a die and between two SLRs. Refer to Clock Skew Minimization through Calibrated Mesh Network Deskew. This is achieved through an calibrated deskew mesh using phase detectors that modulate delays at clock region boundaries.
- The leaf clocks no longer have EN pins but have added divide capability.

Figure 2: **Locations of Clock Buffer Resources and Clock Management Functions**



Differences from Previous Generations

The concept of a heterogeneous clock management tile (CMT) no longer exists and its functions are separated in to the XPLL, MMCM/DPLL, and GCLK blocks. The XPLL and MMCM have been separated and are located in the optimal position for the Versal™ architecture. The main function of the XPLL is to clock the XPHY that are embedded within the XPIO banks. The MMCM is still co-located with the clock buffers.

MMCM

- Includes two deskew subsystems each containing a deskew phase detector (PD). The deskew PD can adjust the phase of the output counters through the phase interpolators (PIs) to deskew the delay between two clock inputs connected to CLKIN/CLKFB, connected to the inputs of the same deskew PD. When used to deskew, the PIs cannot be used in the fine phase-shift mode.
- Each output counter of the MMCM has a PI associated with it. The PI is placed between the VCO and the counter and can provide static or dynamic fine phase shifting of the counter inputs. The resolution of phase shift is now 1/32 of the VCO clock.
- The UltraScale™ method of fractional divide has been replaced with a sigma-delta module (SDM) based fractional mode. The MMCM now supports fractional divide ratios with a 6-bit resolution which is much finer than the 0.125 resolution that was supported in UltraScale. This fractional divide is supported using CLKFBOUT. CLKOUT0 no longer has support for fractional divide.
- The ZHOLD compensation mode has been removed from MMCM because XPIO no longer supports the legacy memory interface mode. ZHOLD is supported by the DPLL for the HDIO columns.
- The complementary (inverted) outputs have been removed from the counters.
- The clock divide dynamic change (CDDC) feature whereby you can change the output counter divide ratio using Dynamic Reconfiguration Port (DRP) has been removed.
- The Startup Wait option during configuration has been eliminated.
- Dynamic configuration in MMCM and/or PLL is performed through the Dynamic Reconfiguration Port. Unlike UltraScale devices, an APB3-compatible interface is the only option used as the protocol on DRP signals. Refer to Dynamic Reconfiguration Port (DRP) for more details.
- Wider multiply and divide ranges and other data sheet specification improvements.

XPLL

- The name of the PLL has changed to XPLL. XPLL is used for every DDR interface along with XPHY.
- Specification improvements as detailed in the Versal ACAP data sheets.

- There are four output counters (dividers) instead of two that support programmable logic clocking. The complementary (inverted) outputs have been removed.
- The XPLL functionality is extended by a user-controllable deskew subsystem. This system allows that an output clock can be dynamically deskewed or phase shifted to an input clock other than the required XPLL input clock.
- The XPLL can be set up to deskew the delay between the two clock inputs connected to CLKIN_DESKEW/CLKFB similar to MMCM.
- The XPLL now has a dynamic phase shift capability and dynamic phase shift interface for each output and feedback counter. Resolution of this phase shift is 1/32 of the VCO clock.

DPLL

An MMCM Lite version of the PLL (DPLL) block has been added.

- The DPLL is located next to the HDIO banks and GT banks.
- The functionality and capabilities of the DPLL are similar to that of the MMCM. However, the DPLL does not have fractional clock generation capabilities and, unlike in the MMCM, there is no distinct feedback counter divider. The DPLL is also located next to every MMCM to provide an increased number of clocking managers.
- The primary function is for frequency synthesis. Duty cycle is not programmable.
- All operations are on a clock cycle basis.

XILINX

Chapter 2

Versal Architecture Clocking Resources

Versal™ architecture based devices have several clock routing resources to support various clocking schemes and requirements, including high fanout, short propagation delay, and extremely low skew. To best use the clock routing resources, the designer must understand how to get user clocks from the PCB to the Versal ACAPs, decide which clock routing resources are optimal, and then access those clock routing resources by using the appropriate I/O and clock buffers.

Global Clock Inputs

External global user clocks must be brought into the Versal device on differential clock pin pairs called global clock (GC or GCIO) inputs. There are four GC pin pairs in each XPIO bank and two GC pin in every HDIO bank that have direct access to the global clock buffers, MMCMs, DPLLs, and XPLLs depending on their location on the device. The HDIO GC pins (HDGC pins) can be connected to DPLL directly. They can only be connected to MMCM and XPLL in XPIO through BUFGCE in HDIO. GC inputs provide dedicated, high-speed access to the internal global and regional clock resources. GC inputs use dedicated routing and must be used for clock inputs where the timing of various clocking features is imperative. General-purpose I/O with local interconnects must not be used for clock signals. Each XPIO bank is located in a single clock region and includes 54 I/O pins. Of the 54 I/O pins in each I/O bank in the I/O column, there are four global clock input pin pairs (a total of eight pins). Each global clock input:

- Can be connected to a differential or single-ended clock on the PCB.
- Can be configured for any I/O standard, including differential I/O standards.
- Has a P-side (master) and an N-side (slave).
- Can be cascaded to the MMCMs, DPLLs, and XPLLs in adjacent banks.

Single-ended clock inputs must be assigned to the P (master) side of the GC input pin pair. If a single-ended clock is connected to the P-side of a differential clock pin pair, the N-side cannot be used as another single-ended clock pin—it can only be used as an user I/O.

GC inputs are global clock input pins and can be used as regular I/O if not used as clocks. The global clock input pins can be configured as any single-ended or differential I/O standard. GC inputs can connect to the XPIO bank or adjacent to the same bank they reside in.

XCC inputs are clock capable input pins for XPHY related clocking and capable of accepting a strobe to capture data.

HDGC inputs are clock capable pins in a HDIO bank to source BUFGCE or DPLL.

For more information on how the above input pins are connected, refer to *Versal ACAP SelectIO Resources Architecture Manual* (AM010).

Clock Buffers

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

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

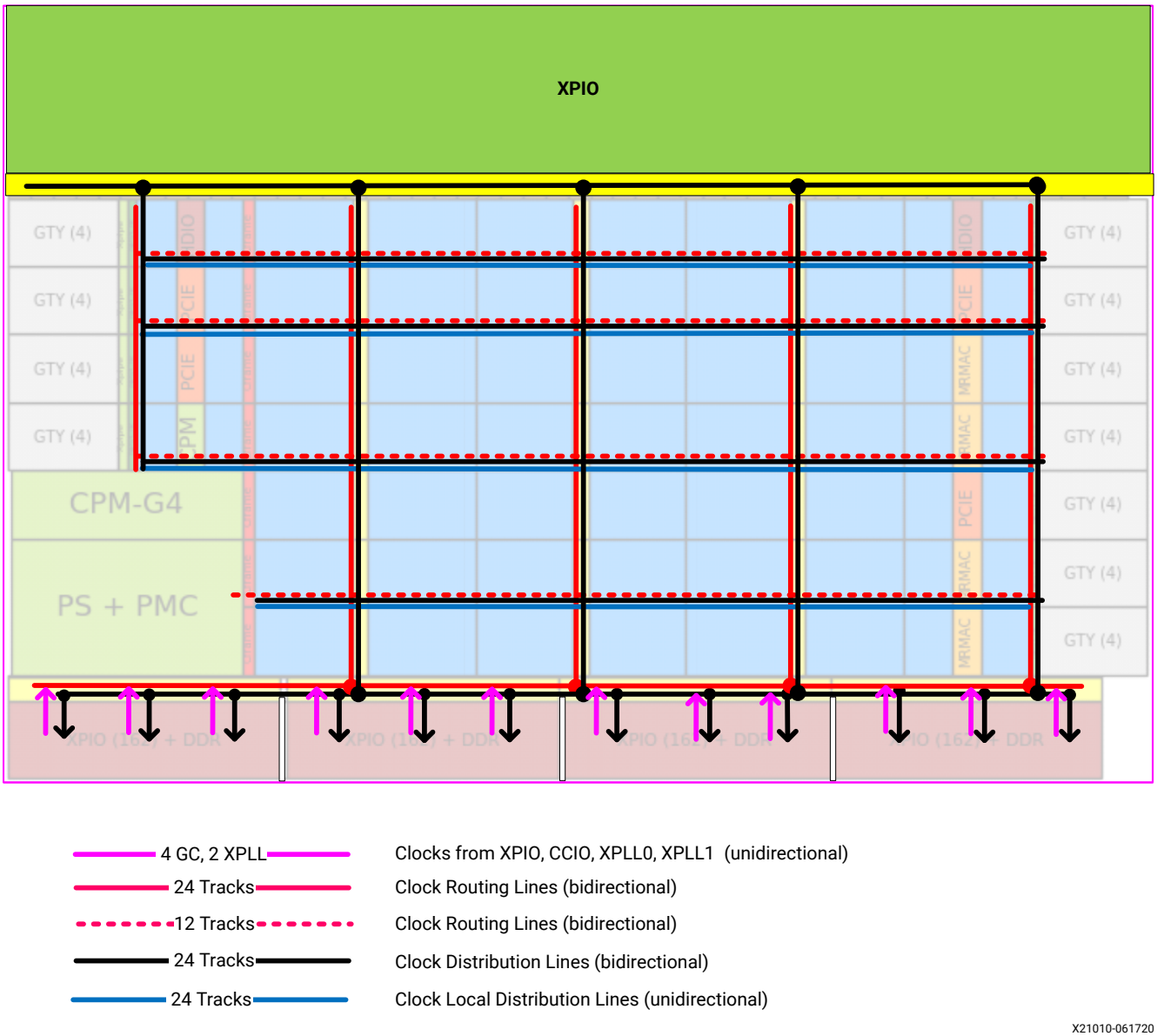
- Clock structure
- Clock buffers
- BUFGCTRL clock buffer primitives
- Additional BUFGCTRL use models
- BUFGCE and BUFGCE_DIV clock buffers
- BUFG clock buffer
- BUFG_GT clock buffer
- BUFG_PS clock buffer
- BUFG_FABRIC buffer (for routing high fanout non-clock nets, not a global clock resource)

All buffers are described later in this chapter.

Clock Routing Structure and Resources

The basic device architecture is composed of blocks of CRs. CRs are organized into tiles and thus build columns and rows. Each CR contains slices (CLBs), DSPs, and 36K block RAM blocks. The mix of slice, DSP, and block RAM columns in each CR can be different, but are always identical when stacked in the vertical direction, thus building columns of those resources for the entire device. I/O and GT columns are then inserted with columns of CRs. In addition, there are one or more columns that contain PCIe®, Interlaken (ILKN), 100G multirate Ethernet MAC (MRMAC), 600G Ethernet MAC (DCMAC), and soft- decision forward error correction (SD-FEC). An HCS runs horizontally through the device in the center of each row of CRs and GTs. The HCS contains the horizontal routing and distribution tracks across the programmable logic region as well as leaf clock buffers. The clocking columns provide interconnects between HCS routing and distribution. Vertical tracks of routing and distribution in the vertical clocking column can connect to all CRs that are to the left and right side of it. There are 12 horizontal routing and 24 horizontal distribution tracks and 24 vertical routing and 24 distribution tracks (see the following figure). The top and bottom clocking rows have 24 routing tracks. 24 local horizontal distribution lines in each individual CR are driven from the clocking column through delay lines for local deskew. The purpose of the clock routing resources is to route a clock from the global clock buffers to a central point from where it is connected to the loads through the distribution resources. This central point of the clock network is called a clock root in the Versal™ architecture. The root can be next to any CR in a device from where it is routed to the loads through the clock distribution resources. This architecture optimized clock skew, routing, and distribution resources can either connect to adjacent CRs or disconnect (isolated) at the border of the CR as needed.

Figure 3: **Clock Routing Architecture**

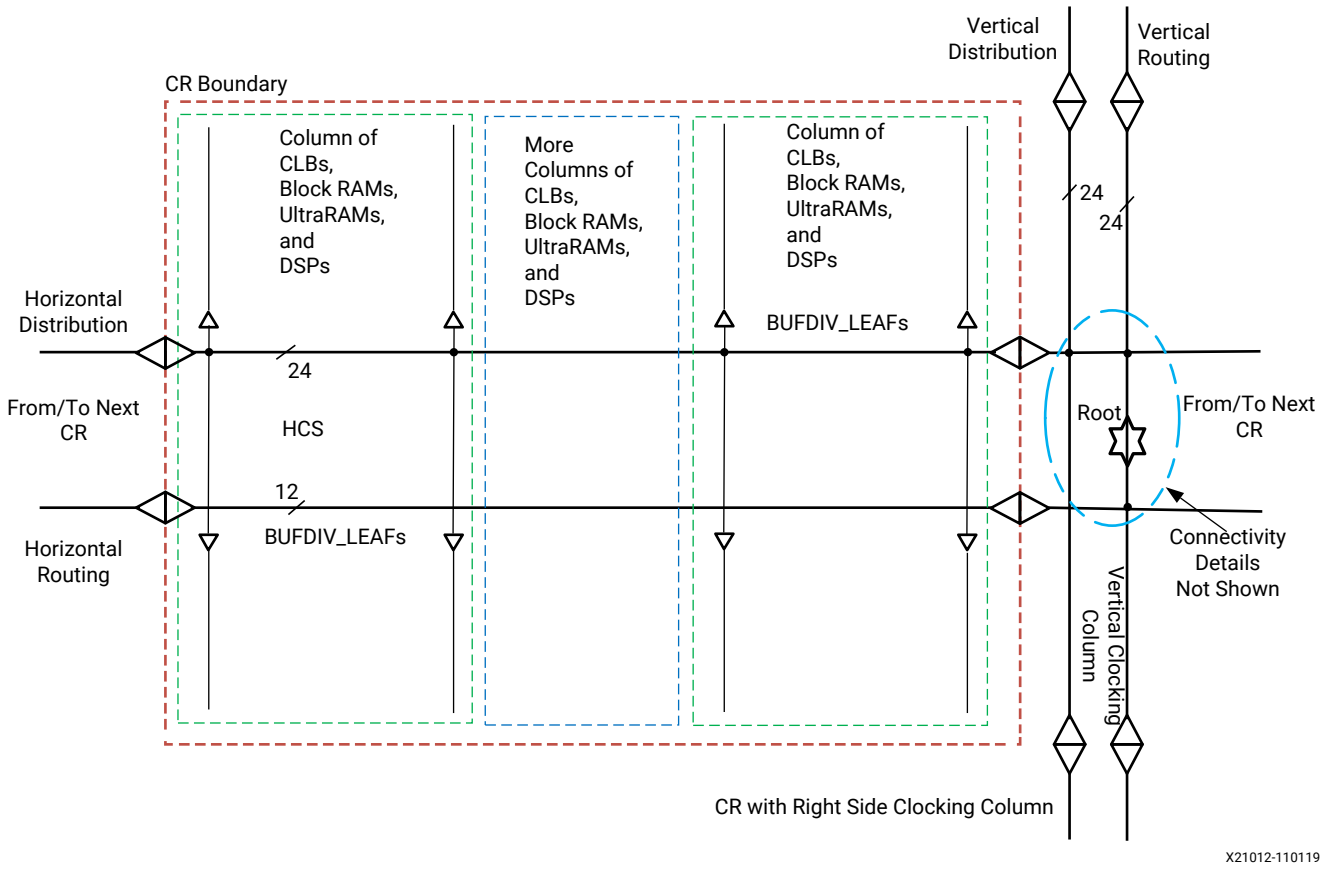


The clocks can be distributed from their sources in one of two ways (see the following figure):

- The clocks can go onto routing tracks that take the clocks to a central point in a clocking column next to a CR without going to any loads. The clocks can then drive the distribution tracks from which the clock networks fan out. In this way, the clock buffers can drive to a specific point next to the CRs from which the clocks travel vertically and then horizontally on the distribution tracks to drive the clocking points. The clocking points are driven through leaf clocks with divide capability in that CR.

- This distribution scheme is used to move the root for all the loads to be at a specific location for improved and localized skew. Furthermore, both routing and distribution tracks can drive into horizontally or vertically adjacent CRs in a segmented fashion. Vertical routing drives a vertical distribution H tree to balance vertical delay to each row. The vertical distribution tree can also feed into the horizontal distribution on top and bottom of the devices. Routing tracks can drive both routing and distribution tracks in the adjacent CRs getting rebuffered while the distribution tracks can drive other horizontal distribution tracks in adjacent CRs through rebuffers as well. The CR boundary segmentation allows construction of either truly global, device-wide clock networks or more local clock networks of variable sizes by reusing clocking tracks.
- Alternatively, clock buffers can drive straight onto the distribution tracks and distribute the clock in that manner. This reduces the clock insertion delay.

Figure 4: **Clock Region**



Where GC inputs are used to drive XPIO corner banks, clock buffers BUFGCTRL and BUFGCE_DIV cannot be placed after MMCM because the sites are reserved. The corner bank affected is below PS (bottom left corner) and does not have full access to programmable logic resources. There is no restriction for the corner bank below the GTs at the bottom right corner. When you need to use an MMCM in the restricted corner bank, it is recommended that you insert a BUFGCE (after the MMCM) which in turn drives the BUFGCTRL and BUFGCE_DIV (and also other BUFGCEs). Refer to the *Versal ACAP Hardware, IP, and Platform Development Methodology Guide* (UG1387) for details on how to work with the restrictions.

Clock Skew Minimization through Calibrated Mesh Network Deskew

Versal architecture uses calibrated skew compensation system to minimize both local and global skew. This is effectively a global clock calibration executed while the part is powered up. A Versal ACAP then automatically adjusts the insertion delay of clocks of each CR to minimize effects of process variations across the die. It samples phase information at all active CR boundaries, and modulates delay lines consisting of coarse and fine delays to each CR to match with all of its neighbors. Delay lines are initialized at configuration time. Prior to the startup of the device, PDs at the CR boundaries measure phase-offset from their neighbors. This information is relayed back to the state machines controlling the delay lines for the two clocks. A simple up/down indication is sent to the surrounding CR's PDs. If information from all PDs agree which way to move (increment/decrement delay), the delay line controller updates the fine delays. This adaptive deskew mesh network then converges on an optimal solution. The adaptive deskew system is bypass-able in the event that it is preferred to route a signal with minimal delay insertion (that is, asynchronous reset, high fanout data signal).

Primitives

- **MMCME5:** A mixed-mode clock manager (MMCM) is a general purpose phase-locked loop (PLL) used for all-purpose clock generation as well as for deskewing applications. In Versal™ ACAPs, the functionalities and capabilities of MMCM are broadly similar to that of UltraScale+™ devices. For more information, see MMCM Primitive.
- **XPLL:** XPLL is a PLL that is situated in between the XPIO banks. It provides clock output to both, the XPHY logic and the programmable logic of the ACAP. It can also be used for deskewing applications. In Versal devices, the functionalities and capabilities of XPLL are broadly similar to that of PLL. For more information, see XPLL Primitive.

- **DPLL:** DPLL is a PLL that is located next high-density I/O (HDIO) banks, and the GT clocking column. It provides clock output to the programmable logic of the ACAP. It can also be used for deskewing applications. The functionalities and capabilities of DPLL are similar to that of MMCM and XPLL. Refer to DPLL Primitive for more information.
- **Clock Buffer Primitives:** The clock buffers in Versal devices are similar to the clock buffers in UltraScale+ devices including:
  - BUFGCTRL block for glitchless clock muxing and gating
  - BUFGCE block for glitchless clock gating
  - BUFGCE_DIV block for clock division
  - BUFG_GT block for clock division of GT clocks
  - BUFG_PS block for PS/PMC

  For more information, see BUFGCTRL Clock Buffer Primitives.

Clock Buffer Resources

Clock buffers are near their intended clock sources. Next to the XPIO, XPHY global clocking contains several sets of BUFGCTRLs, BUFGCEs, and BUFGCE_DIVs. Each set can be driven by four GC pins from the XPIO bank, MMCMs, XPLLs, and DPLLs in the bottom of the part, and interconnect. Four BUFGs next to the HDIO columns can be driven by two GC pins. The clock buffers then drive the routing and distribution resources across the entire device. Each XPIO clocking block contains 24 BUFGCEs, 8 BUFGCTRLs, and 4 BUFGCE_DIVs but only 24 of them can be used at the same time. HDIO clocking block only contains BUFGCEs. The BUFG_GTs can drive on horizontal routing to get to vertical routing in another column or drive the vertical routing in its own column directly.

RECOMMENDED: *It is recommended to only allow the Vivado® Placer to assign all global clock buffers to specific locations. Each CR in the HSR contains 24 BUFGCEs, 8 BUFGCTRLs, and 4 BUFGCE_DIVs. These clock buffers share the 24 routing tracks and therefore collision scan occur resulting in unroutable designs. If the design requires a number of global clock buffers to be in a certain CR then it is recommended to attach the CLOCK_REGION property to these buffers instead of a specific location property.*

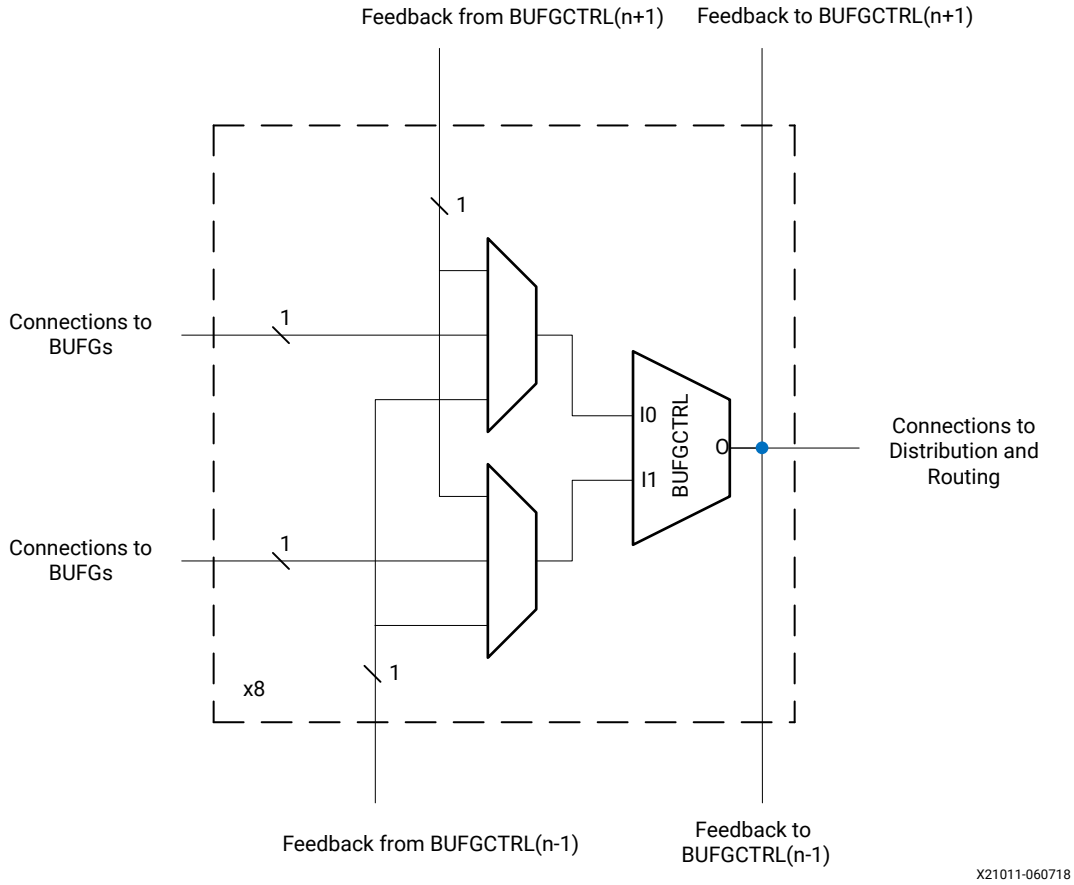
The CLOCK_REGION property allows you to assign a clock buffer to a specific clock region of a Versal™ ACAP, while letting the Vivado Placer assign the clock buffer to the best site within that region. User assignment of the CLOCK_REGION can be performed in XDC as follows:

```
set_property CLOCK_REGION X4Y6 [get_cells {sys_clk_pll/inst/clkf_buf}]
```

In Versal architecture, BUFGCTRL multiplexers and all derivatives can be cascaded to adjacent clock buffers to create larger multiplexers. This feature can be used to create a ring of eight BUFGMUXes (BUFGCTRL multiplexers). For example, BUFGCTRL1 is fed by and feeds BUFGCTRL0 as well as BUFGCTRL2. This wraps around in such a way that BUFGCTRL0 is fed by and feeds BUFGCTRL7 as well as BUFGCTRL1.

The following figure shows a simplified diagram of cascading BUFGCTRLs.

Figure 5: **BUFGCTRL Cascading**



The following subsections detail the various configurations, primitives, and use models of the clock buffers.

BUFGCTRL Clock Buffer Primitives

The primitives shown in the following table are different configurations of the clock BUFGCTRL buffers. The Vivado tools manage the configuration of all these primitives. Refer to the *Vivado Design Suite User Guide: Using Constraints* (UG903) for information on manually placing the BUFGCTRL locations with LOC constraint.

Table 1: **BUFGCTRL Clock Buffer Primitives**

Primitive	Input	Output	Control
BUFGCTRL	I0, I1	O	CE0, CE1, IGNORE0, IGNORE1, S0, S1
BUFGCE_1	I	O	CE
BUFGMUX	I0, I1	O	S

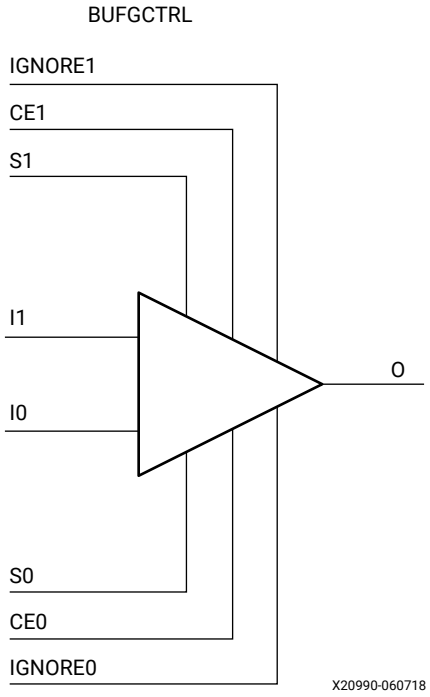
Table 1: **BUFGCTRL Clock Buffer Primitives** *(cont'd)*

Primitive	Input	Output	Control
BUFGMUX_1	I0, I1	O	S
BUFGMUX_CTRL	I0, I1	O	S

BUFGCTRL

The BUFGCTRL primitive shown in the following figure can switch between two asynchronous clocks. All other global clock buffer primitives are derived from certain configurations of BUFGCTRL. BUFGCTRL has four select lines, S0, S1, CE0, and CE1. It also has two additional control lines, IGNORE0 and IGNORE1. These six control lines are used to control the inputs I0 and I1.

Figure 6: **BUFGCTRL Primitive**



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

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

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

Table 2: **Truth Table for BUFGCTRL Clocking Resources**

CE0	S0	CE1	S1	O
1	1	0	X	I0
1	1	X	0	I0
0	X	1	1	I1
X	0	1	1	I1
1	1	1	1	Old Input[1]

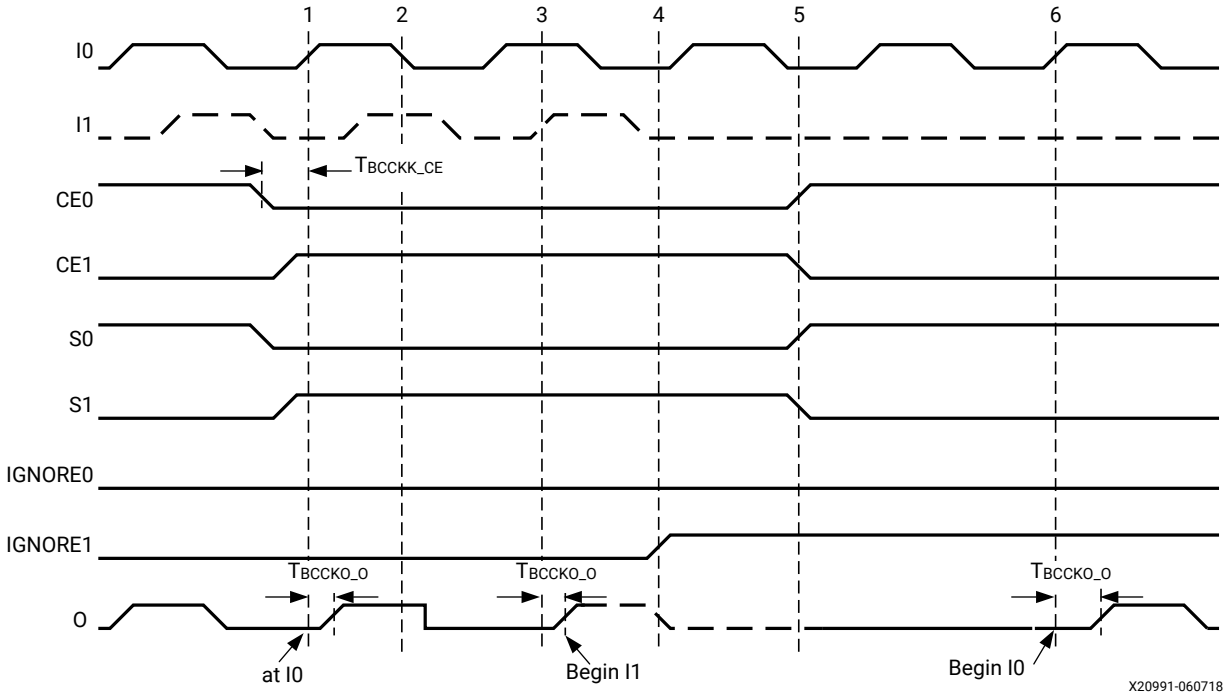
Notes:

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

Although both S and CE are used to select a desired output, only S is suggested for glitch-free switching. This is because when using CE to switch clocks, the change in clock selection can be faster than when using S. A violation in the setup/hold time of the CE pins causes a glitch at the clock output. On the other hand, using the S pins allows the user to switch between the two clock inputs without regard to setup/hold times. As a result, using S to switch clocks does not result in a glitch. See BUFGMUX_CTRL.

The following timing diagram illustrates various clock switching conditions using the BUFGCTRL primitives. Exact timing numbers are best found using the speed specification.

Figure 7: **BUFGCTRL Timing Diagram**



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

Other capabilities of the BUFGCTRL primitive are:

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

The following table summarizes the attributes for the BUFGCTRL primitive.

Table 3: **BUFGCTRL Attributes**

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

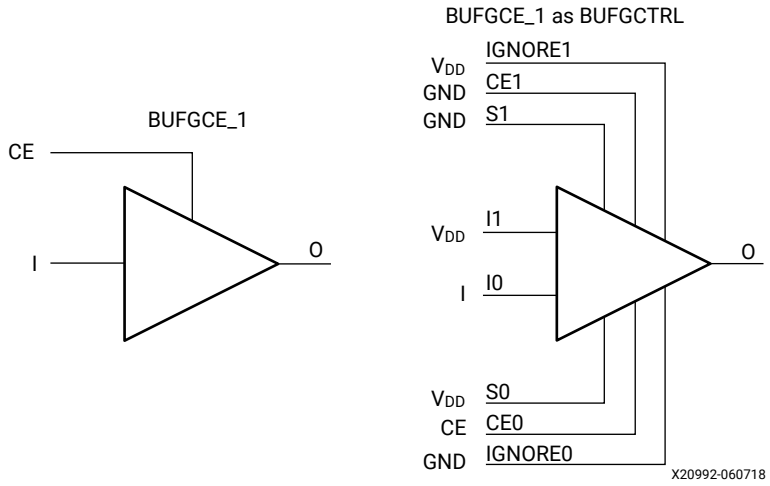
Notes:

1. Both PRESELECT attributes cannot be TRUE at the same time.

BUFGCE_1

The BUFGCE_1 primitive is a clock buffer with one clock input, one clock output, and a clock enable line. This primitive is based on BUFGCTRL with some pins connected to logic High or Low. The following figure illustrates the relationship of BUFGCE_1 and BUFGCTRL. The LOC constraint is available for manually placing the BUFGCE_1 location. See the *Vivado Design Suite User Guide: Using Constraints* (UG903) for more information.

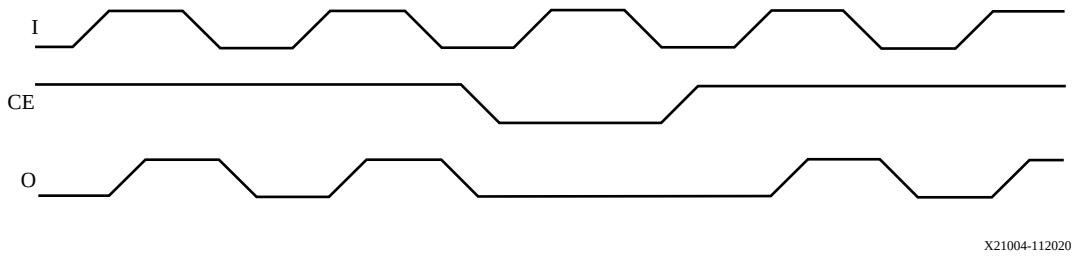
Figure 8: **BUFGCE_1 using BUFGCTRL**



IMPORTANT! *The select signal must meet the setup time requirement because the clock enable line uses the CE pin of the BUFGCTRL. Violating this setup time can result in a glitch.*

The following figure illustrates the timing diagram for BUFGCE_1.

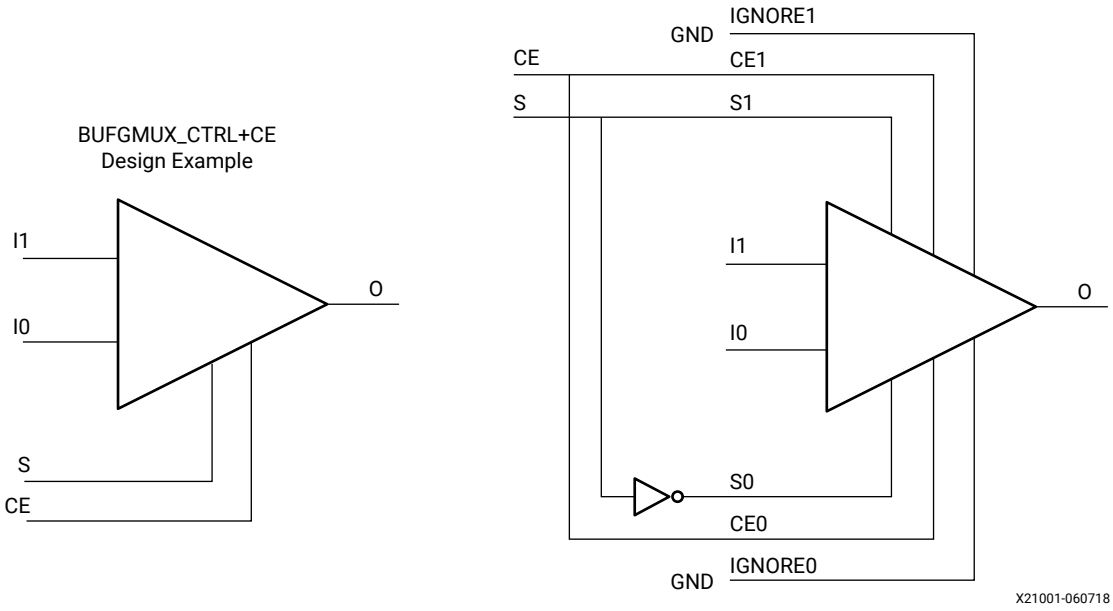
Figure 9: **BUFGCE_1 Timing Diagram**



BUFGMUX and BUFGMUX_1

BUFGMUX is a clock buffer with two clock inputs, one clock output, and a select line. This primitive is based on BUFGCTRL with some pins connected to logic High or Low. The following figure illustrates the relationship of BUFGMUX and BUFGCTRL. The LOC constraint is available for manually placing the BUFGMUX and BUFGCTRL locations. See the *Vivado Design Suite User Guide: Using Constraints* (UG903) for more information.

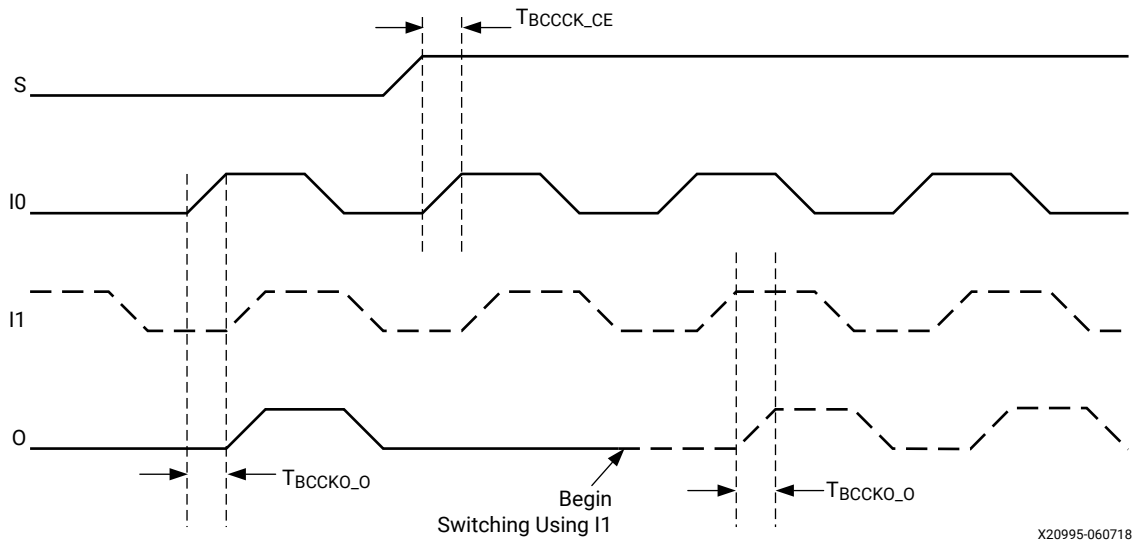
Figure 10: **BUFGMUX using BUFGCTRL**



IMPORTANT! *When using the select, the setup time requirement must be met because BUFGMUX uses the CE pins as select pins. Violating this setup time can result in a glitch.*

Switching conditions for BUFGMUX are the same as the CE pins on BUFGCTRL. The following figure illustrates the timing diagram for BUFGMUX.

Figure 11: **BUFGMUX Timing Diagram**

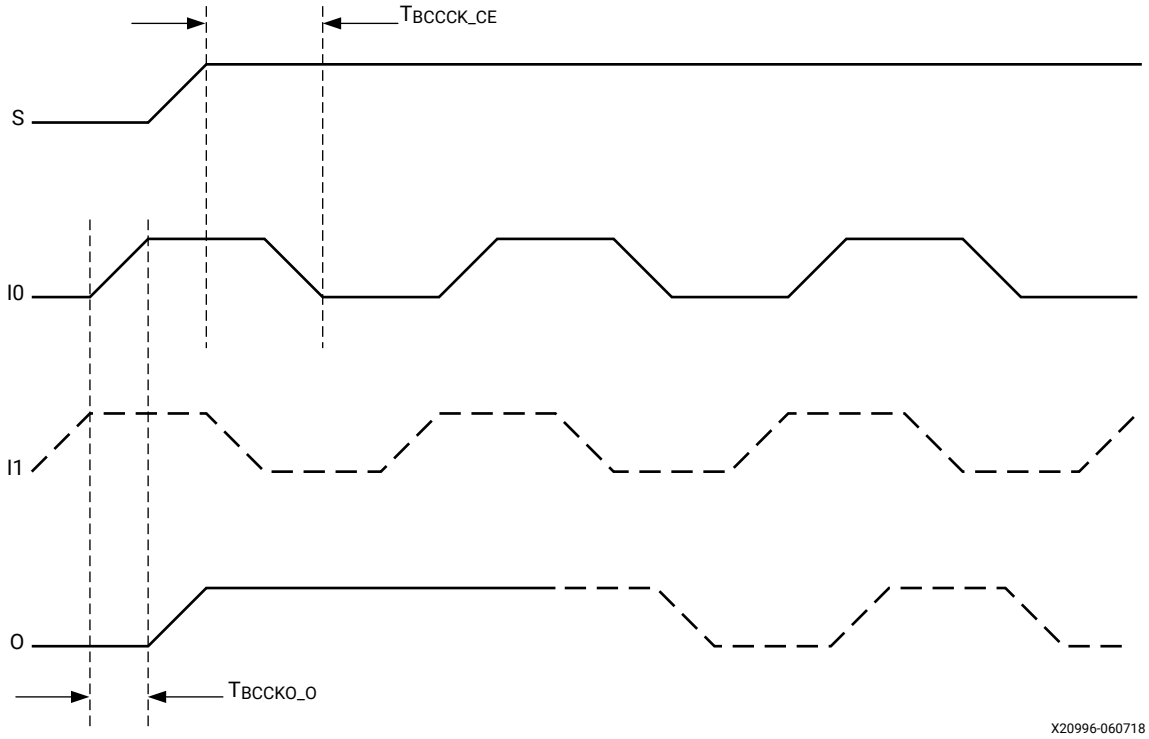


The figure shows:

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

BUFGMUX_1 is rising-edge sensitive and held at High prior to input switch. The following figure illustrates the timing diagram for BUFGMUX_1. The LOC constraint is available for manually placing the BUFGMUX and BUFGMUX_1 locations. See the *Vivado Design Suite User Guide: Using Constraints* (UG903) for more information.

Figure 12: **BUFGMUX_1 Timing Diagram**



The figure shows:

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

The following table summarizes the attributes for the BUFGMUX primitive.

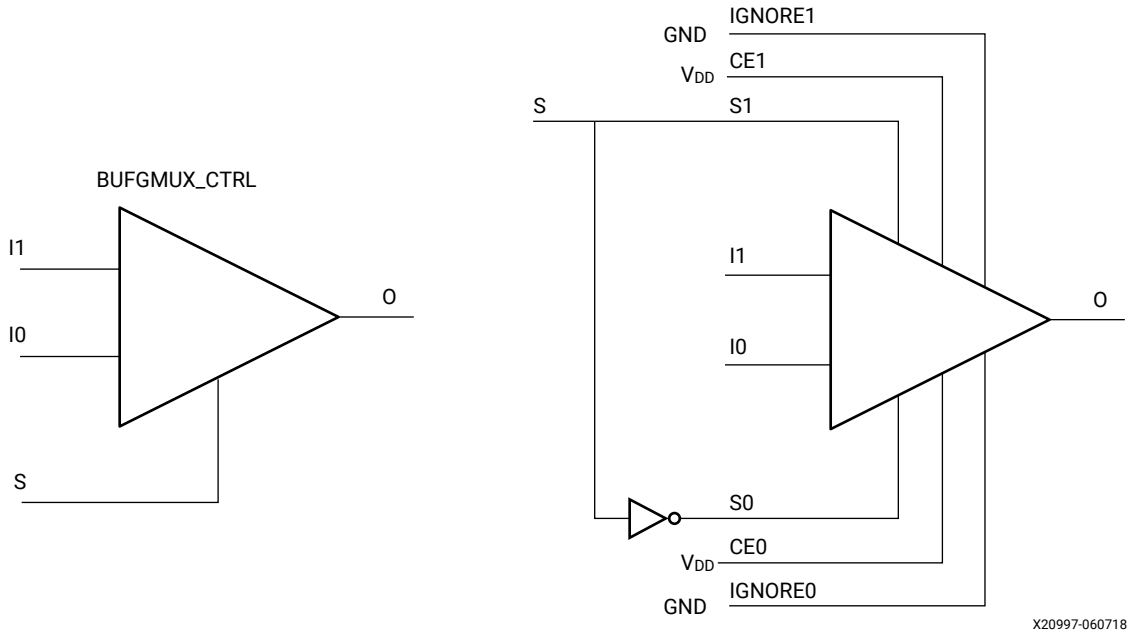
Table 4: **BUFGMUX Attributes**

Attribute Name	Description	Possible Values
CLK_SEL_TYPE	Specifies synchronous or asynchronous clock switching.	SYNC (default), ASYNC

BUFGMUX_CTRL

BUFGMUX_CTRL is a clock buffer with two clock inputs, one clock output, and a select line. This primitive is based on BUFGCTRL with some pins connected to logic High or Low. The following figure illustrates the relationship of BUFGMUX_CTRL and BUFGCTRL.

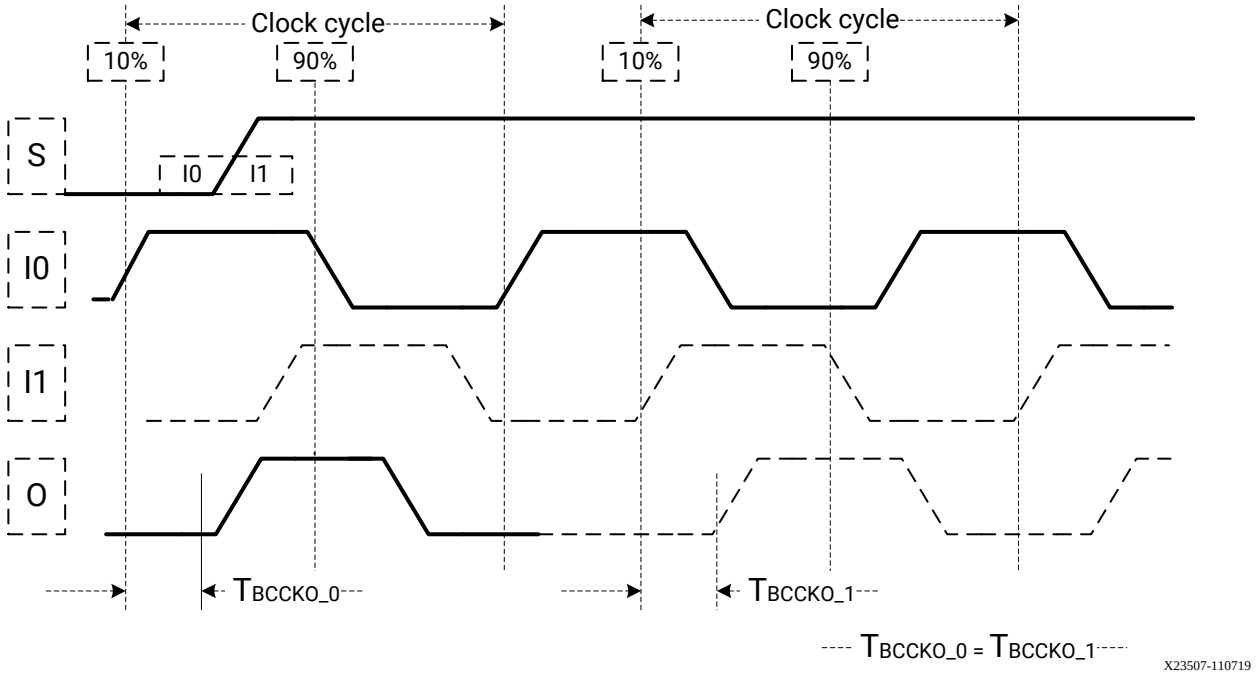
Figure 13: **BUFGMUX_CTRL using BUFGCTRL**



BUFGMUX_CTRL uses the S pins as select pins. S can switch anytime without causing a glitch. The setup/hold time on S is for determining whether the output passes an extra pulse of the previously selected clock before switching to the new clock. If S changes as shown in the following figure prior to the setup time TBCCCK_S and before I0 transitions from High to Low, the output does not pass an extra pulse of I0. If S changes following the hold time for S, the output passes an extra pulse. If S violates the setup/hold requirements, the output might pass the extra pulse but it will not glitch. In any case, the output changes to the new clock within three clock cycles of the slower clock.

The setup/hold requirements for S0 and S1 are with respect to the falling clock edge, not the rising edge as for CE0 and CE1. Switching conditions for BUFGMUX_CTRL are the same as the S pin of BUFGCTRL. The following figure illustrates the timing diagram for BUFGMUX_CTRL.

Figure 14: **BUFGMUX_CTRL Timing Diagram**



Other capabilities of the BUFGMUX_CTRL primitive are:

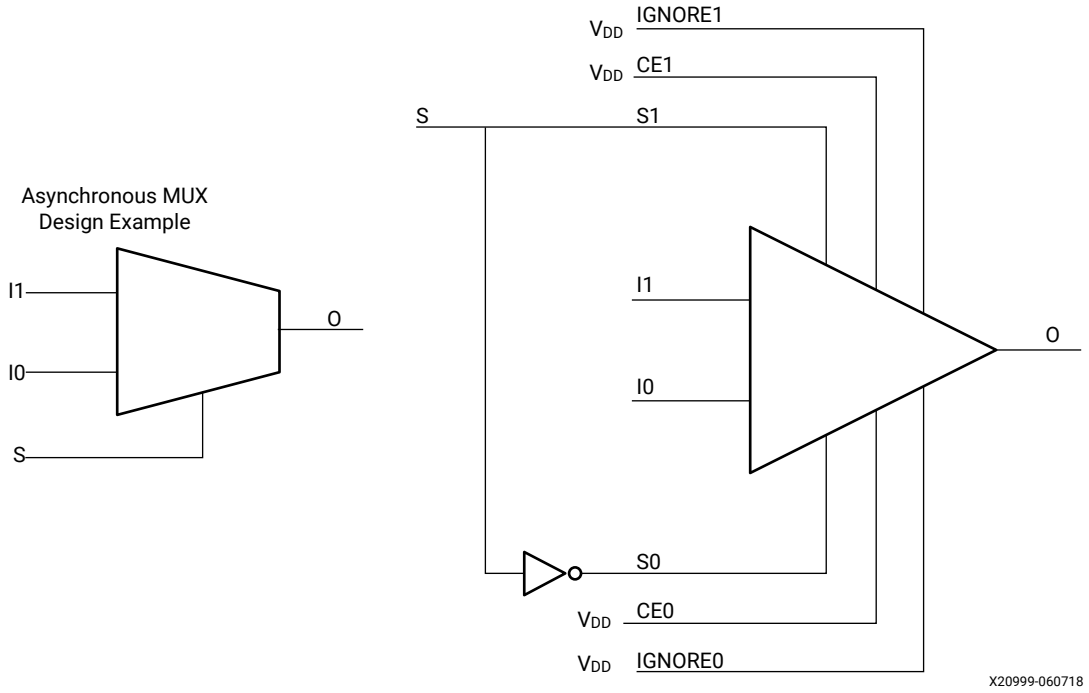
- I0 and I1 inputs can be preselected after configuration
- Initial output can be selected as High or Low after configuration

Additional BUFGCTRL Use Models

Asynchronous MUX Using BUFGCTRL

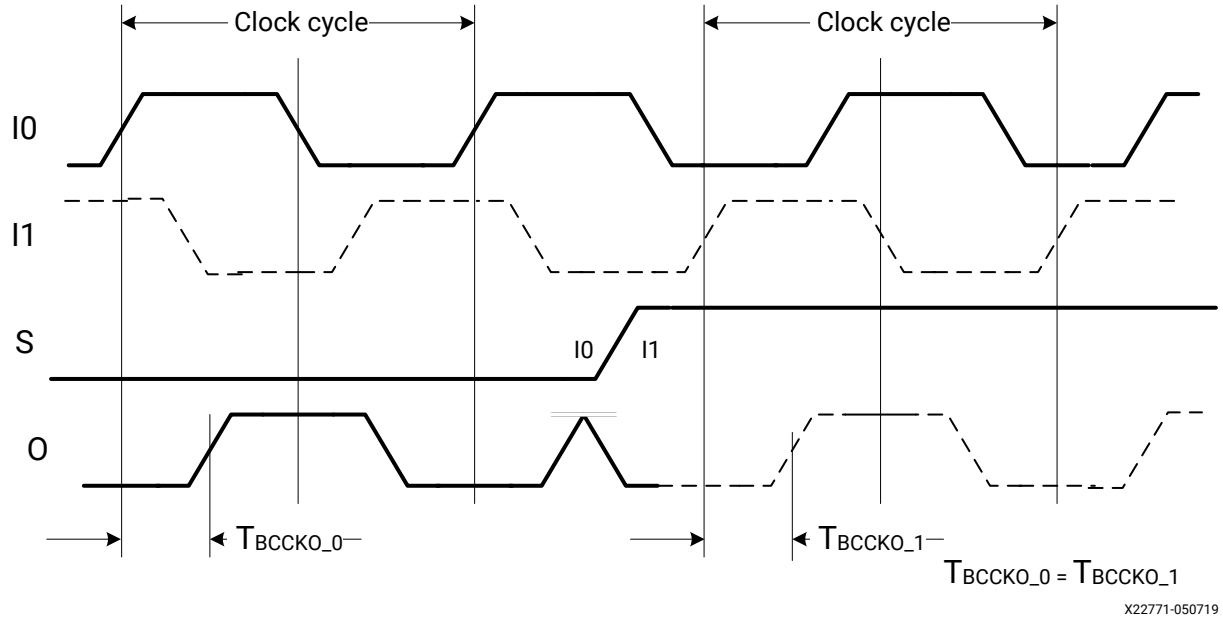
In some cases an application requires immediate switching between clock inputs or bypassing the edge sensitivity of BUFGCTRL. An example is when one of the clock inputs is no longer switching. If this happens, the clock output would not have the proper switching conditions because the BUFGCTRL never detected a clock edge. This case uses the asynchronous MUX. The following figure illustrates an asynchronous MUX with BUFGCTRL design example.

Figure 15: **Asynchronous MUX using BUFGCTRL Design Example**



The following figure shows the asynchronous MUX timing diagram.

Figure 16: **Asynchronous MUX Timing Diagram**



The figure shows:

- The current clock is from I0.

- S is activated High.
- The clock output immediately switches to I1.
- When ignore signals are asserted High, glitch protection is disabled.

BUFGMUX_CTRL with a Clock Enable

A BUFGMUX_CTRL with a clock enable BUFGCTRL configuration allows you to choose between the incoming clock inputs. If needed, the clock enable is used to disable the output. The following figures illustrate the BUFGCTRL usage design example and the timing diagram.

Figure 17: **BUFGMUX_CTRL with a CE Using BUFGCTRL**

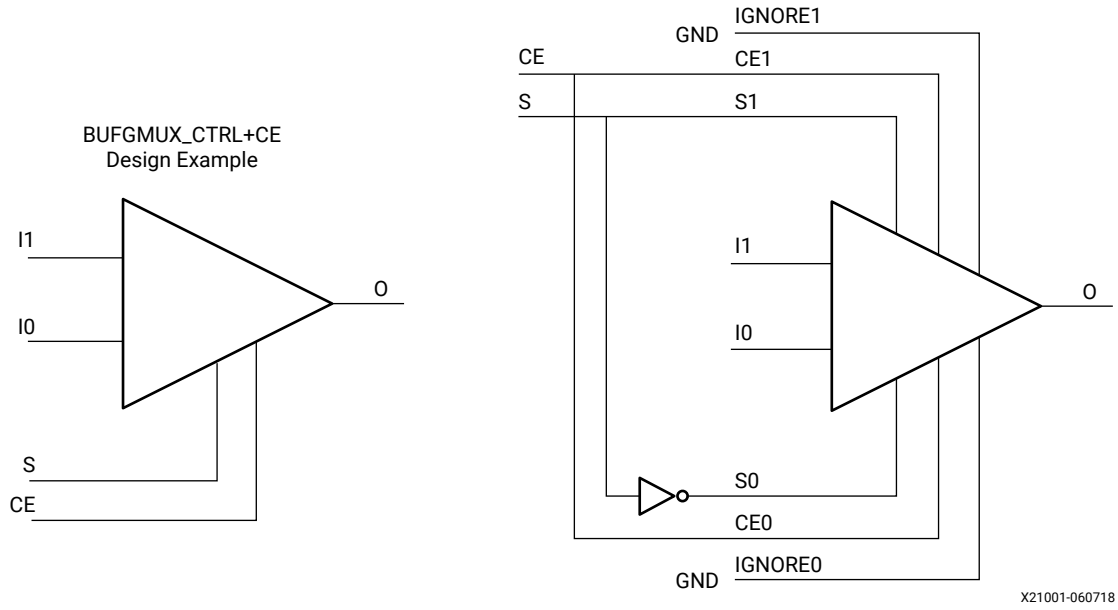
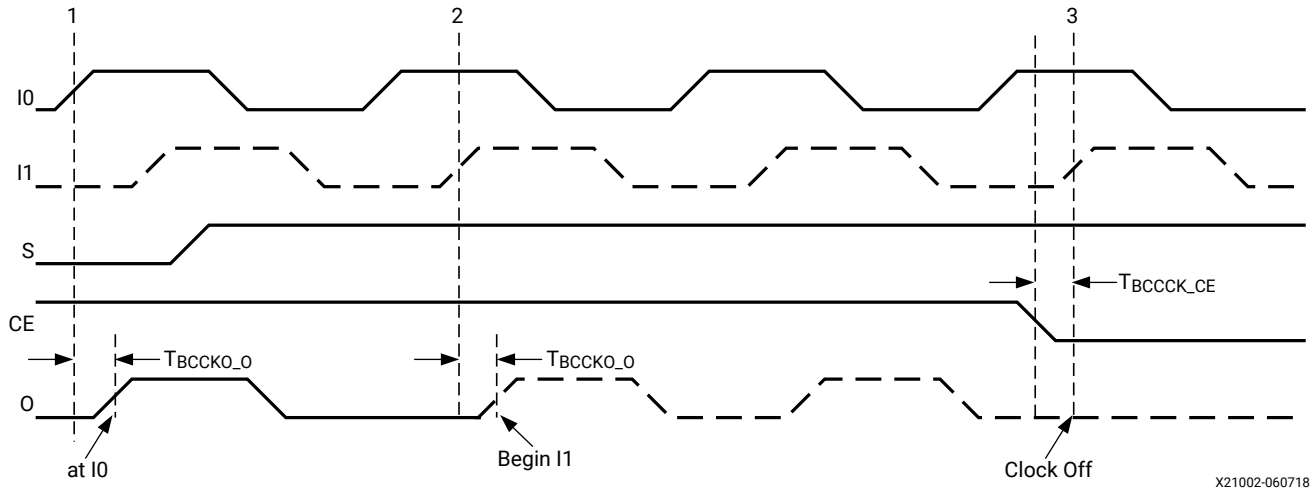


Figure 18: **BUFGMUX_CTRL with a CE Timing Diagram**

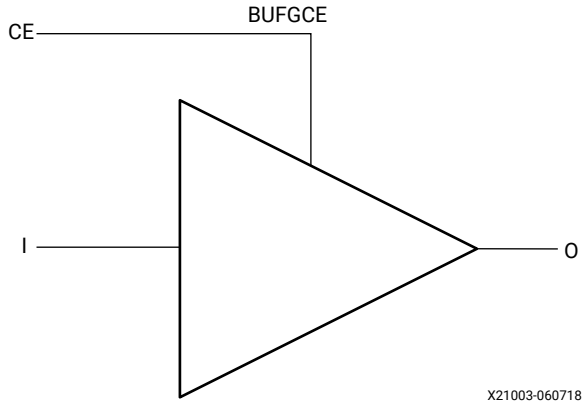


- At time event 1, output O uses input I0.
- Before time event 2, S is asserted High.
- At time TBCCKO_O, after time event 2, output O uses input I1. This occurs after a High-to-Low transition of I0 followed by a High-to-Low transition of I1 is completed.
- At time TBCCCK_CE, before time event 3, CE is asserted Low. To avoid any output clock glitches, the clock output is switched Low and kept at Low until after a High-to-Low transition of I1 is completed.

BUFGCE Clock Buffers

BUFGCE is a clock buffer with one clock input, one clock output, and a clock enable line (see the following figure). This buffer provides glitch less clock gating. BUFGCE can directly drive the routing resources and is a clock buffer with a single gated input. Its O output is 0 when CE is Low (inactive). When CE is High, the I input is transferred to the O output.

Figure 19: **BUFGCE with CE**



The following table shows the BUFGCE attributes.

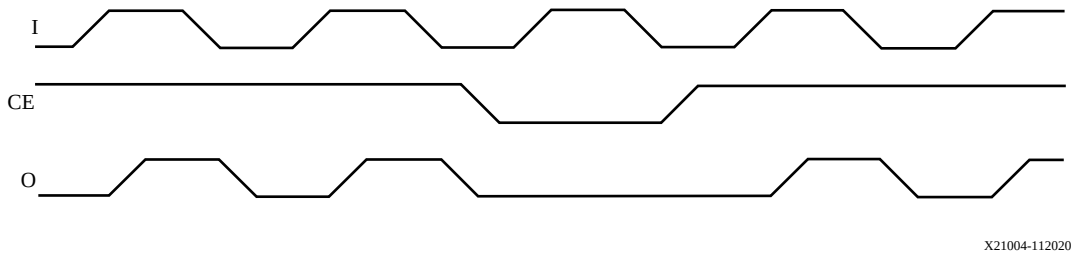
Table 5: **BUFGCE Attributes**

Attribute Name	Values	Default	Type	Description
CE_TYPE	SYNC, ASYNC, HARDSYNC	SYNC	STRING	Sets the clock enable behavior where SYNC allows for glitchless transition while ASYNC allows immediate transition. The SYNC setting times the CE pin in the Vivado tools while the ASYNC setting ignores the timing arc. HARDSYNC turns ON an internal 3-stage synchronizer for maximum performance. However, that results in a latency of either three or four clock cycles.
STARTUP_SYNC	FALSE, TRUE	FALSE	STRING	Defines whether logic to CE input is staged

***Note*:** For additional tool related attributes used by Vivado Design Suite during the opt_design stage, refer to the *Versal Architecture AI Core Series Libraries Guide* (UG1353).

The following figure shows the BUFGCE timing diagram.

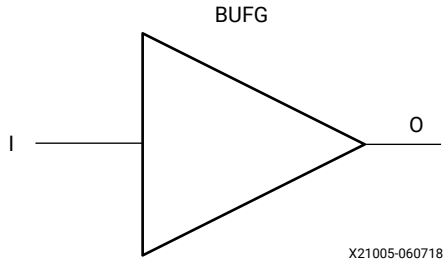
Figure 20: **BUFGCE Timing Diagram**



BUFG Clock Buffer

BUFG is a clock buffer with one clock input and one clock output. This primitive is based on BUFGCE with the CE pin connected to High, as shown in the following figure.

Figure 21: **BUFG Buffer**



BUFDIV_LEAF Clock Buffer

BUFDIV_LEAF is a clock buffer with a static divide capability of 1, 2, 4, and 8 for leaf clocks driving off horizontal HCS row. This buffer is an interconnect leaf clock buffer driving the clocking point of the various blocks with a single gated input. It is a physical only buffer and cannot be instantiated by the user. The following table shows the BUFLEAF_DIVIDE attributes.

Table 6: **BUFDIV_LEAF Attributes**

Attribute Name	Values	Default	Type	Description
BUFLEAF_DIVIDE	1, 2, 4, 8	1	STRING	Factor by which the input clock at the leaf level is divided down.

BUFGCE_DIV

BUFGCE_DIV is a clock buffer with one clock input (I), one clock output (O), one clear input (CLR), and a clock enable (CE) input. BUFGCE_DIV can directly drive the routing and distribution resources and is a clock buffer with a single gated input and a reset. Its O output is 0 when CLR is High (active). When CE is High, the I input is transferred to the O output. CE is synchronous to the clock for glitch-free operation. CLR is an asynchronous reset assertion and synchronous reset deassertion to this buffer. BUFGCE_DIV can also divide the input clock by 1 to 8.

When CLR (reset) is deasserted, the output clock transitions from Low to High on the first edge after the CLR is deasserted, regardless of the divide value. Therefore, BUFGCE_DIV output clocks are always aligned, regardless of the divide value. The output clock then toggles at the divided frequency. When CLR is asserted, the clock stops toggling after some clock-to-out time. For an odd divide, the duty cycle is not 50% because the clock is High one cycle less than it is Low. For example, for a divide value of 7, the clock is High for 3 cycles and Low for 4 cycles.

When CE is deasserted, the output stops at its current state, High or Low. When CE is reasserted, the internal counter restarts from where it stopped. For example, if the divide value is 8 and CE is deasserted two input clock cycles after the last output High transition, the output stays High. Then when CE is reasserted, the output transitions Low after two input clock cycles. If the reset input is used, upon assertion the output transitions Low immediately if the current output is High, otherwise it stays Low.

Because reset is synchronously deasserted, when reset is deasserted in the previous example, the output transitions High at the next input clock edge and transitions Low four input clock cycles later.

The following table shows the BUFGCE_DIV pins.

Table 7: **BUFGCE_DIV Pins**

Pin Name	Type	Invertible	Description
I	Input	FALSE	Clock input
CLR	Input	TRUE	Reset
CE	Input	TRUE	Clock enable
O	Output	FALSE	Clock output

The following table shows the BUFGCE_DIV attributes.

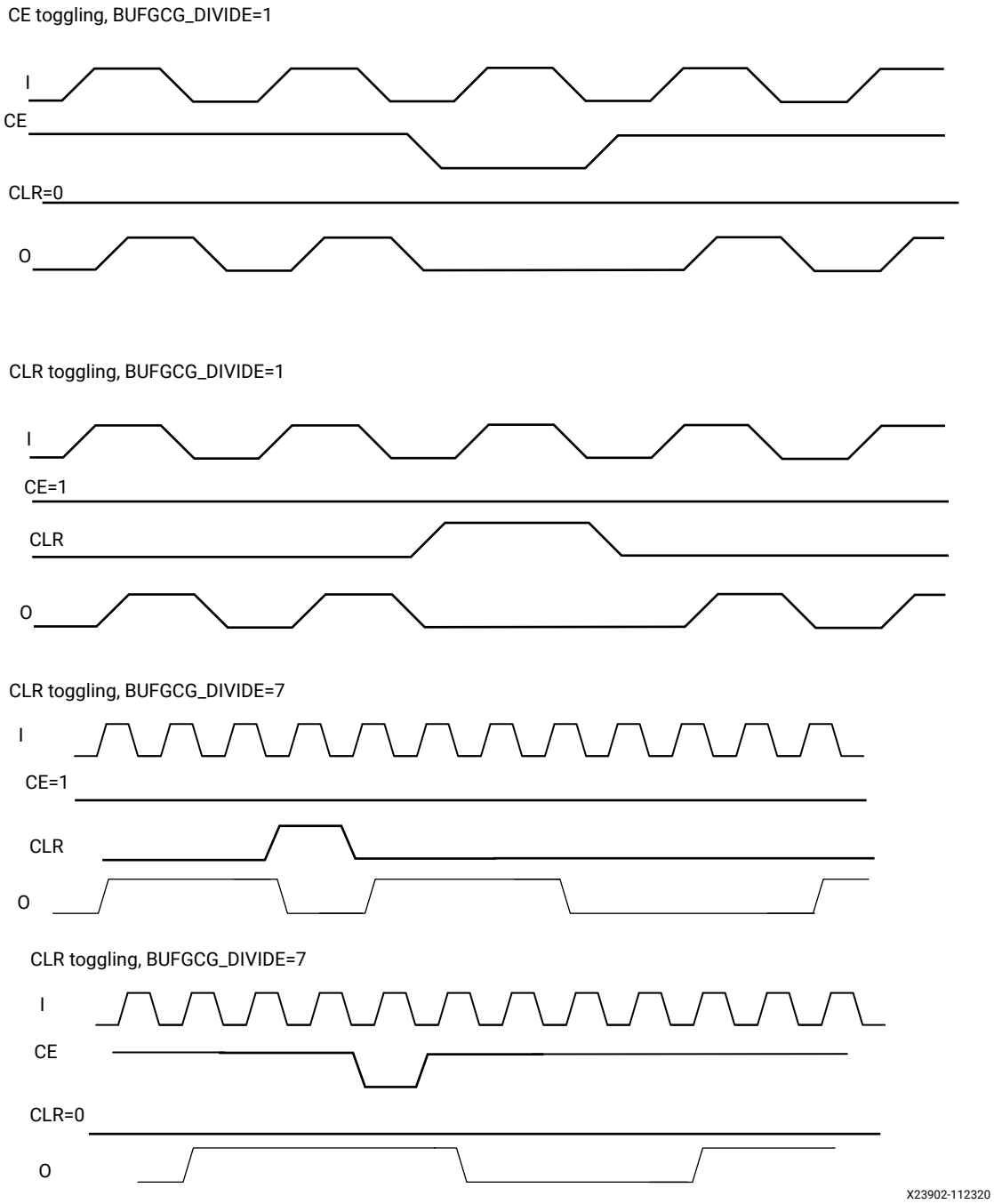
Table 8: **BUFGCE_DIV Attributes**

Attribute Name	Values	Default	Type	Description
BUFGCE_DIVIDE	1, 2, 3, 4, 5, 6, 7, 8	1	Integer	Defines whether the output clock is a divided version of the input clock.
CE_TYPE	SYNC,HARDSYNC	SYNC	STRING	Sets the clock enable behavior where SYNC allows for a glitch-less transition while ASYNC allows an immediate transition. The SYNC setting times the CE pin in the Vivado tools while the ASYNC setting ignores the timing arc. HARDSYNC turns ON an internal 3-stage synchronizer for maximum performance. However, that results in a latency of either three or four clock cycles.
HARDSYNC_CLR	FALSE, TRUE	FALSE	STRING	Asynchronous clear port to control CLR_B signal on BUFDIV_LEAF
STARTUP_SYNC	FALSE, TRUE	FALSE	STRING	Defines whether logic to CE input is staged

***Note*:** For additional tool related attributes used by Vivado Design Suite during the opt_design stage, refer to the *Versal Architecture AI Core Series Libraries Guide* (UG1353).

The following figure shows the BUFGCE_DIV timing diagram.

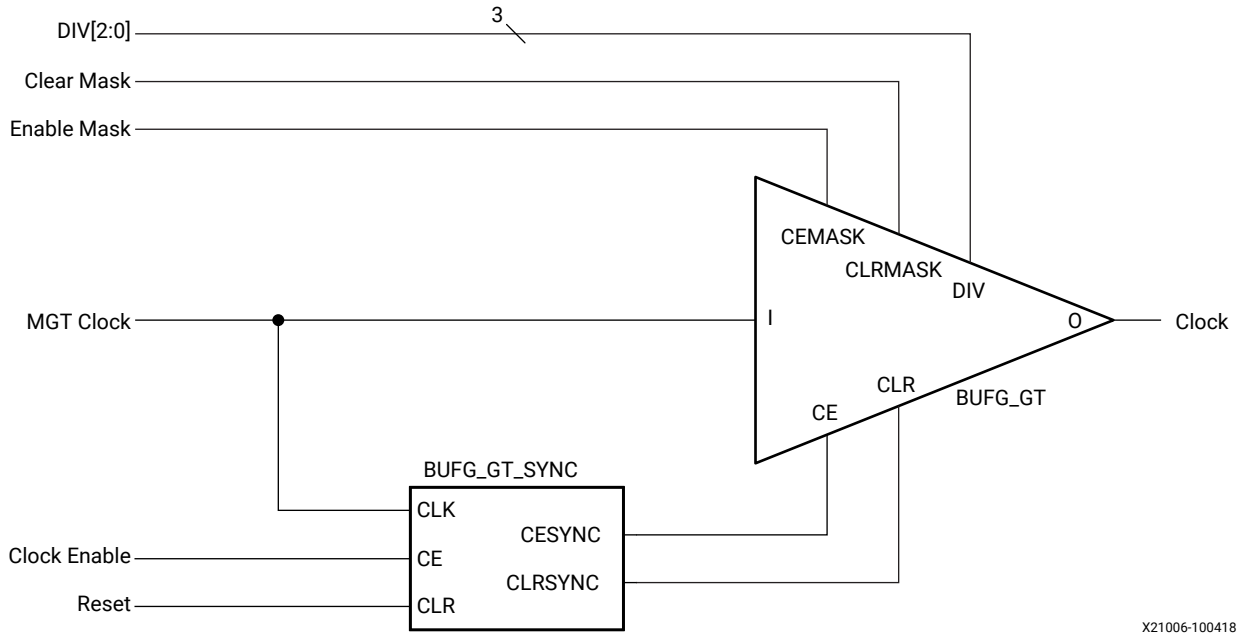
Figure 22: **BUFGCE_DIV Timing Diagram**



BUFG_GT and BUFG_GT_SYNC

The BUFG_GTs are driven by the gigabit transceivers (GTs). BUFG_GT (see the following figure) is a clock buffer with one clock input (I), one clock output (O), one clear input (CLR) with CLR mask input (CLRMASK), a clock enable (CE) input with a CE mask input (CEMASK), and a 3-bit divide (DIV[2:0]) input. BUFG_GT_SYNC is the synchronizer circuit for the BUFG_GTs and is shown here explicitly. The BUFG_GT_SYNC primitive is automatically inserted by the Vivado® tools, if not present in the design. This buffer can directly drive the routing and distribution resources and is a clock buffer with a single gated input and a reset. When CE is deasserted (Low) the output stops at its current state, High or Low. When CE is High, the I input is transferred to the O output. Both edges of CE and the deassertion of CLR are automatically synchronized to the clock for glitch-free operation. The Vivado tools do not support timing for the CE pin, therefore, a deterministic latency cannot be achieved. CLR is an asynchronous reset assertion and synchronous reset deassertion to the BUFG_GTs. The synchronizers have two stages, but the CLR pin does not have a setup/hold timing arc assigned. Therefore, the latency is not deterministic. BUFG_GTs can also divide the input clock by 1 to 8. The DIV[2:0] value is the actual divide minus 1 (that is, `3'b000` corresponds to 1 while `3'b111` corresponds to 8). The divide value (DIV inputs), CEMASK, and CLRMASK must be changed while the buffer is held in reset. The input clock is allowed to change while CE is deasserted or reset is asserted. However, there is a minimum deassertion/assertion time for those control signals.

Figure 23: **BUFG_GT Primitive**



Versal™ ACAPs have 24 BUFG_GTs and 41 BUFG_GT_SYNCs per GT Quad. Any of the GT output clocks in a Quad can be multiplexed to any of the BUFG_GTs. The 41 CE and CLR pins, which correspond to the 41 BUFG_GT_SYNCs, can drive the 24 BUFG_GTs. Each of the BUFG_GT buffers have an individual mask for both CE and CLR (24). All BUFG_GTs driven by the same clock source must also have a common CE and CLR signal. Tying off CE and CLR to a constant signal in this case is not allowed, but a mask can be set to provide the same functionality. The output clocks of the BUFG_GTs connected to the same input clock are synchronized (phase-aligned) to each other when coming out of reset (CLR) or on CE assertion. Individual mask pins can be used to control which BUFG_GT(s) out of the group of 24 respond to CE and CLR and therefore, are synchronized to each other or retain their previous phase and divide value. These clock buffers are located in the HCS and are directly driven by the GT output clocks. Their purpose is to directly drive hard blocks and logic in the CRs through the routing and distribution resources. GTs have no other direct, dedicated connections to other clock resources. However, they can connect to XPLL, MMCM, DPLL, and GCLK blocks through the BUFG_GT and the clock routing resources.

When CLR (reset) is deasserted, the output transitions High at the next input clock edge and transitions Low divide_value/2 input clock cycles later. Because reset is synchronously deasserted, two clock cycles of synchronization latency need to be added to the output to transition it to High. The next transition to Low then occurs four input clock cycles after that (divide by 8). The output transitions to High a number of clock cycles later, determined by the divide value specified, after which the output clock toggles at the divided frequency.

When CLR is asserted, the clock stops toggling at Low after some clock-to-out time. For an odd divide, the duty cycle is not 50% because the clock is High one cycle less than it is Low. For example, for a divide value of 7, the clock is High for 3 cycles and Low for 4 cycles.

When CE is deasserted, the output stops at its current state, High or Low. When CE is reasserted, the internal counter restarts from where it stopped. For example, if the divide value is 8 and CE is deasserted two input clock cycles after the last output High transition, the output stays High. Then, when CE is reasserted, the output transitions Low four input clock cycles later (two for synchronization and two to complete the High time period of the output clock because of being a divide by 8). If the reset input is used, upon assertion the output transitions Low immediately if the current output is High, otherwise it stays Low. Because reset is synchronously deasserted, when reset is deasserted in the previous example, the output transitions High two input clock cycles later due to synchronization and transitions Low four input clock cycles after that (divide by 8).

The mask pins (CEMASK and CLRMASK) control how a specific, single BUFG_GT responds to the CE/CLR control inputs. When a mask pin is deasserted, its respective control pin has their normal function. When a mask pin is asserted, the respective control pin is ignored, in effect allowing the clock to propagate through (that is, CE is effectively High and reset is effectively Low). The internal synchronizers phase align the clock outputs of the BUFG_GTs that are not masked. Both edges of CE are synchronized while only the deassertion of reset is synchronized. Assertion of

reset immediately causes the output of the BUFG_GT to go Low if it was previously High. This can cause a potential glitch or runt pulse. If this is not acceptable, CE should be used to stop the output. A reset should then be asserted after two input clock cycles plus half the *divide value*. This ensures that the output clock High time (if the output clock happened to be disabled High) is no less than normal.

IMPORTANT! *While the synchronizers ensure that all BUFG_GTs driven by the same clock come out of reset in phase, they might not be in phase with BUFG_GTs that have not been reset (i.e., that have their reset mask asserted).*

Multi-Clock Buffers

For high-speed designs in previous architectures, it was common to use several clocks, generated by the same MMCM, with power of 2 ratios. In such cases, users generated clocks with separate MMCM outputs, used BUFGCE cells or the fastest clock with an MMCM, and created the divided clocks with BUFGCE_DIV cells. Those clock buffers were placed close to the clock managers and often caused timing challenges due to the amount of uncertainty introduced in the paths between source and destination clocks. In designs with synchronous paths between clocks, the routing delay matching implementation option (CLOCK_DELAY_GROUP) is needed to minimize the clock-domain-crossing skew, and limit the impact on setup and hold fixing. In larger devices, timing closure between clock domains can turn out to be very challenging and users, therefore, often make the choice to add logic, FIFOs, and treat clock paths as asynchronous paths.

With Versal™ ACAP, you can use a multi-clock buffer (MBUFG) and generate up to four clocks close to the leaf clock pins instead of at the XPLL, MMCM, or DPLL, thereby greatly reducing the clock pessimism impact on synchronous clock-domain-crossing paths and solving design timing more efficiently. Basically there is a single route to the BUFDIV_LEAF cell which does the division at the leaf driver level. Each MBUFG type buffer uses the corresponding BUFG site, for example MBUFGCE will be placed on BUFGCE site and so on.

The different MBUFG clock buffers have the same attributes as their counterparts, but with one extra MODE attribute that can be set for PERFORMANCE or POWER. The type of clocks vary based on the mode PERFORMANCE or POWER as explained below.

When on a MBUFG-type clock buffer and the MODE attribute is set to PERFORMANCE, the outputs of that clock buffer behave as follows:

- Output O1 = Input I
- Output O2 = Input I divided by 2
- Output O3 = Input I divided by 4
- Output O4 =Input I divided by 8

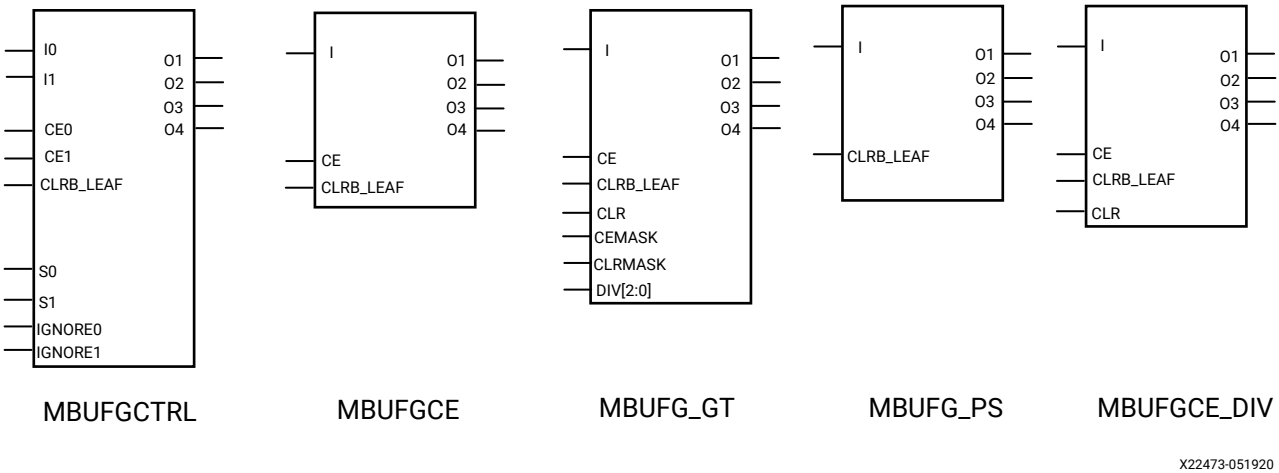
When on a MBUFG-type clock buffer and the MODE attribute is set to POWER, the outputs of that clock buffer behaveas follows:

- Output O1 – Input I multiplied by 2
- Output O2 = Input I
- Output O3 = Input I divided by 2
- Output O4 = Input I divided by 4

There are five types of MBUFG or multi-clock buffers: MBUFGCTRL, MBUFGCE, MBUFGCE_DIV, MBUFG_PS, and MBUFG_GT, as shown in the following figure. The multi-clock buffers have the same input pins as the respective equivalent BUFGCTRL, BUFGCE, BUFGCE_DIV, BUFG_PS, and BUFG_GT clock buffers (also described in this manual), but it has four different outputs.

The multi-clock buffers are logical-only clock buffers that utilize the equivalent BUFGs plus the physical-only BUFDIV_LEAF clock buffers to divide the clock closer to the loads. This removes the insertion delay from the clock buffer to the leaf clocks and reduces clock skew on synchronous clock domain crossing (CDC) paths. Many designs use multiple simple clock buffers divide-by-n clock buffers, where *n* can be 2, 4, or 8, and the MBUFG handles multiple instantiated buffers by offering a single buffer. The CLRB_LEAF input on the MBUFG primitives is used to asynchronously reset the BUFDIV_LEAF dividers. For more information on CLRB_LEAF reset requirements for MBUFG, see the *Versal ACAP Hardware, IP, and Platform Development Methodology Guide* (UG1387).

Figure 24: **MBUFG Type Clock Buffers**



- **MBUFGCTRL:** This primitive is designed as a synchronous/asynchronous glitch-free multiplexer with two clock inputs and multiple outputs. If clock multiplexing is not necessary, one should use the MBUFGCE component.
- **MBUFGCE:** This primitive is designed as a clock buffer with enable/disable possibilities with single clock input and multiple outputs.

- **MBUFG_GT:** This primitive is designed as a clock buffer by the gigabit transceiver devices for the purpose of clock distribution with minimal clock pessimism to other parts of the design.
- **MBUFG_PS:** This primitive is designed as a multi-output high fanout buffer for low skew distribution of the PS clock signals.
- **MBUFGCE_DIV:** This primitive is designed as a multi-output clock buffer with an enable, clear, and divide function.

BUFG_PS

The BUFG_PS is a simple clock buffer with one clock input (I) and one clock output (O). This clock buffer is a resource for the processor system (PS) and provides access to the programmable logic (PL) clock routing resources for clocks from the processor into the PL. Up to 12 PS clocks can drive the BUFG_PS. This clock buffer resides in the vertical clocking column next to the PS.

BUFG_FABRIC

The BUFG_FABRIC is buffer driven programmable logic used for routing high fanout non-clock nets. It allows you to bring a signal from programmable logic routing resources onto the clock network. However, it is not for global clocking.

XILINX

Chapter 4

Clock Management Functions

Overview

In Versal™ ACAPs, clock management includes:

- MMCM (analog PLL with extended functions for general purpose clocking usage)
- XPLL (analog PLL to support XPHY and provide additional general purpose clocking usage)
- DPLL (MMCM lite version of PLL)

All three clock functions are designed to generate clocking for specific and general purpose tasks in Versal ACAP. The purpose of the MMCM is to generate clocks of different frequencies for various general purpose applications. The other two are designed to fill the needs and special requirements of certain blocks like XPIO while maintaining some secondary, general purpose functionality by supporting a limited subset of the MMCM capabilities that can be used for general clocking purposes.

The following figure provides a more detailed view of the location of clock management elements, MMCM and PLL, in the Versal architecture. Refer to Clock Management MMCM, XPLL, and DPLL for a high-level view of where clock management components are placed within the Versal architecture.

Unlike all previous architectures, Versal ACAP is actually a combination of dedicated or specialized hardware components along with standard programmable logic. Clocking management and structures are no longer formatted within a regular structure; they are only placed where required.

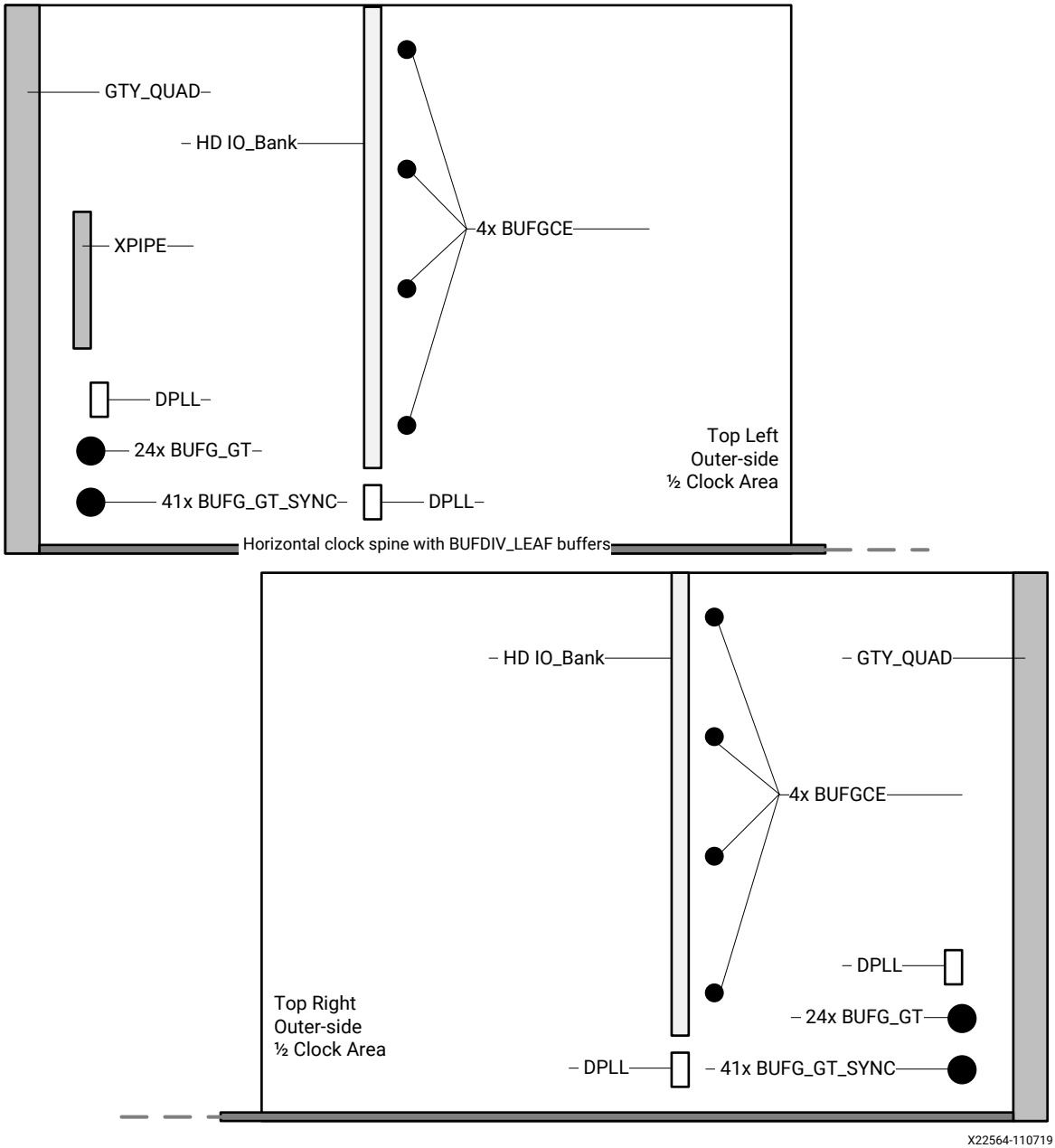
The structure of a device is still divided into clock regions (although not in rigid blocks) which segment the device into functional areas: configurable logic, memory, hard IP, and signal and clock routing. Segmentation in the clock regions makes it possible to model devices to target specified needs and/or markets.

Clock regions with clock management functions are available in a XCVC1902 device. These appear in different compositions in all other Versal ACAPs.

- **Top-left and right-half clock region with high-density I/O bank:**

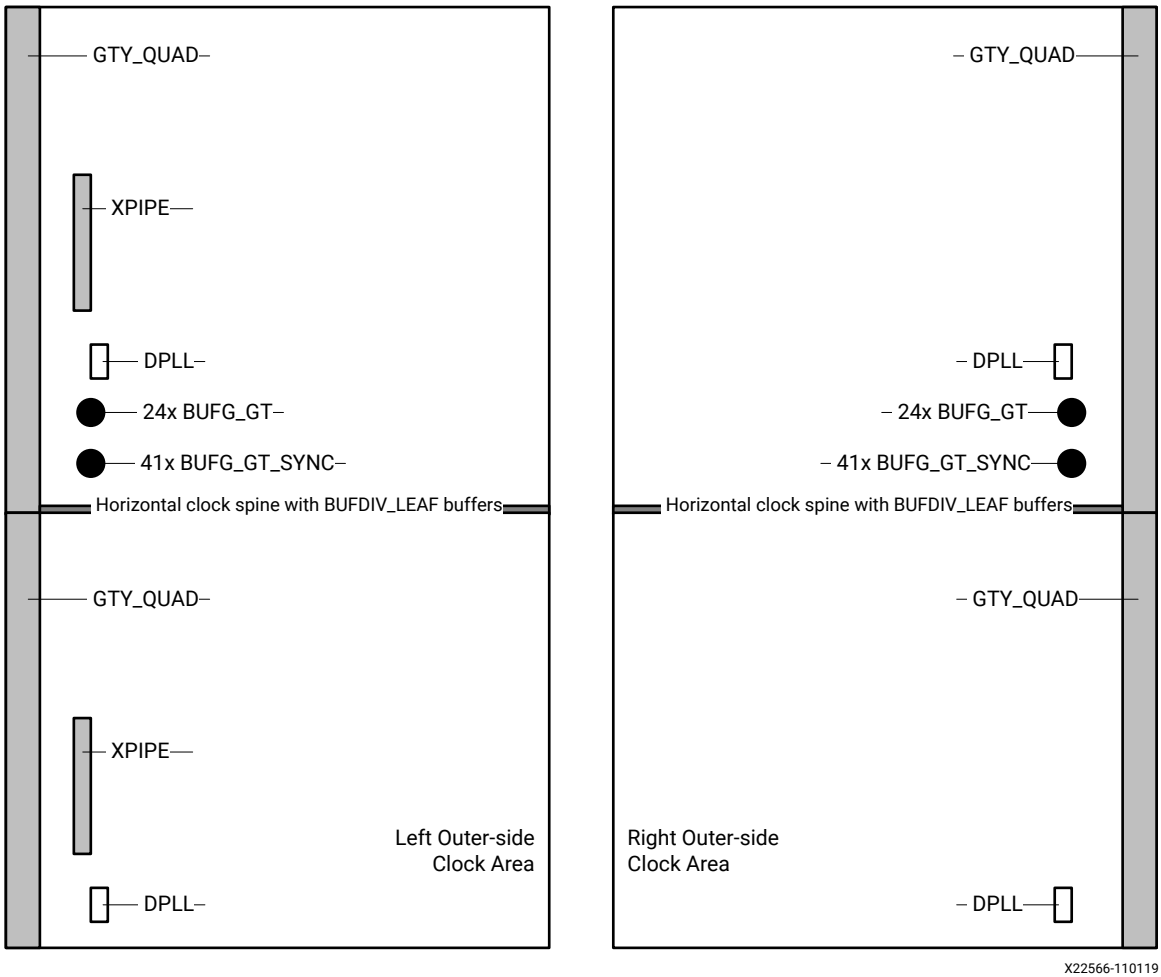
- Half-clock region in this device featuring a high-density I/O bank. This is the top side of a clock region because that includes all clock buffers and the horizontal clock spine.
- A DPLL and clock buffers (24x BUFGT_GT and 41x BUFGT_GT_SYNC) are available to serve the high speed serial GT (GTY, GTH, ...) components and their programmable logic.
- A DPLL and clock buffers (4x BUFGCE) is available to serve all clock needs of the high-density I/O and required programmable logic. The horizontal clock spine with BUFDIV_LEAF buffers runs from left to right and vice versa over the device.

Figure 25: **Top Left and Right Clock Regions**



- **Left and right clock region with hard-IP blocks:**
  - In this case, the clock region is used full and half-sized. The top side of the clock region, containing all clock buffers and horizontal clock spine, is used when half a clock region is implemented. On the left side, on the top of the CPM processor, a top half clock region is used.
  - These clock regions are where the hard-IP blocks are: PCIe, DCMAC, ILKN, and others.
  - A DPLL and clock buffers (24x BUFGT_GT and 41x BUFGT_GT_SYNC) are available to serve the high-speed serial GT (GTY, GTH, ...), hard-IP blocks components and their programmable logic. The horizontal clock spine with BUFDIV_LEAF buffers runs from left to right and vice versa over the device.

Figure 26: **Middle Left and Right Clock Regions**

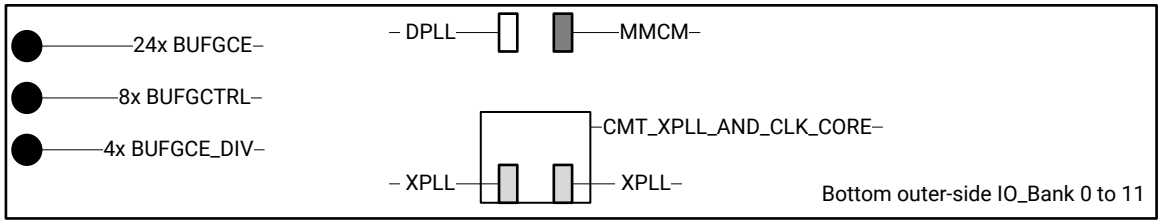


- **Middle buried clock regions:**
  - Clock regions where a vertical NoC spine passes are regions where in the top side of that clock region 24 BUFG_FABRIC clock buffers are available.

- All other clock regions only have the horizontal clock spine in the top side of the region.
- The clock region next to the processor subsystem contain as set of 24 BUFG-PS clock buffers.

- **Bottom clock regions:**
  - These are clock regions housing dedicated memory controller IP and high-speed select I/O components.
  - The clock regions are: DPLL, MMCM, a dual-XPLL supplemented with 24 BUFGCE, 8 BUFGCTRL, and 4 BUFGCE_DIV.

Figure 27: **Bottom Clock Regions**



MMCMs

The MMCMs serve as frequency synthesizers for a wide range of frequencies, and as jitter filters for either external or internal clocks, and deskew clocks.

The clock input connectivity allows multiple resources to provide the reference clock(s) to the MMCM. The number of output counters (dividers) is seven. MMCMs have 32 step phase interpolators feeding the input of the output and feedback counters thus providing infinite fine phase shift capability in either direction and can be used in dynamic phase shift mode. The resolution of the fine phase shift is 1/32 of the VCO frequency. In addition to integer divide output counters, MMCMs can support fractional feedback divide values using a sigma-delta module (SDM). This effectively allows for a clock multiplier to be specified with a resolution of $1/(2^6)$ (6 bits) to support greater clock frequency synthesis capability than before. The new fractional divide support is available on CLKFBOUT counter M. CLKOUT0 no longer has separate fractional divide.

MMCMs also have the capability to generate spread-spectrum clocks that vary the clock frequency very slowly to spread the clock's electromagnetic (EM) energy over a frequency band. This reduces the maximum EM energy at any single frequency. If the MMCM spread-spectrum feature is not used, a spread spectrum on an external input clock is not filtered and thus passed on to the output clock.

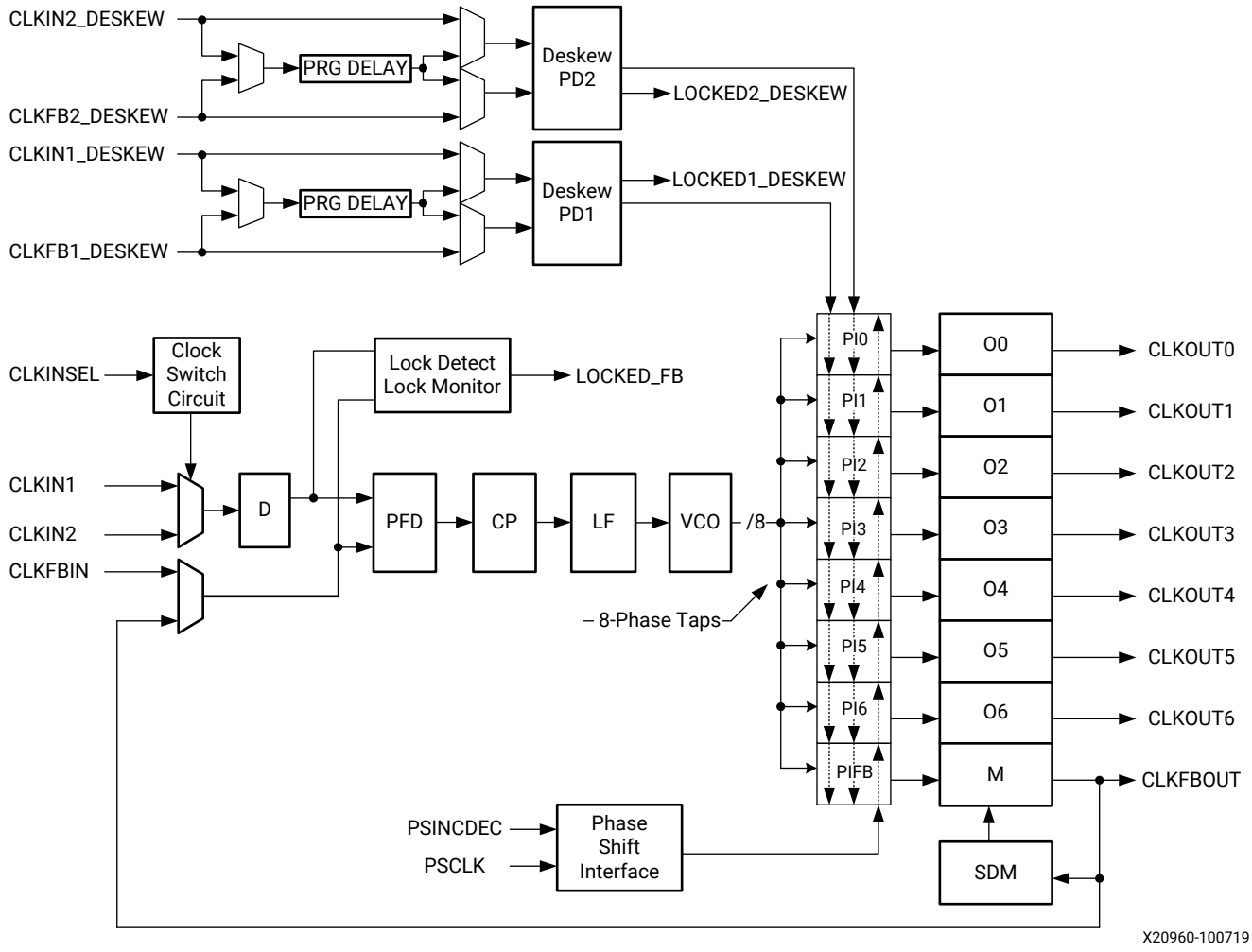
Input multiplexers select the reference and feedback clocks from either the global clock I/Os or the clock routing or distribution routing resources. Each clock input has a programmable divider (D). The phase-frequency detector (PFD) compares both phase and frequency of the rising edges of both the input (reference) clock and the feedback clock. If a minimum high/low pulse is maintained, the duty cycle is ancillary. The PFD is used to generate a signal proportional to the phase and frequency between the two clocks. This signal drives the charge pump (CP) and loop filter (LF) to generate a reference voltage to the VCO. The PFD produces an up or down signal to the charge pump and loop filter to determine whether the VCO should operate at a higher or lower frequency. When VCO operates at a frequency that is too high, the PFD activates a down signal causing the control voltage to be reduced, thus decreasing the VCO operating frequency. When the VCO operates at a frequency that is too low, an up signal increases the voltage. The VCO produces eight 45 degree output phases which in turn feed the eight PIs. The PI can convert the 8 VCO phases into 32 VCO phases. Each PI output is then fed to a corresponding counter (O or M counters). Each output phase can be selected as the reference clock to the counters (see the following figure). 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 MMCM, allowing a wide range of frequency synthesis.

A deskew sub-block consists of two deskew phase detectors (PDs) and phase interpolators (PIs). Each of the PIs can be controlled either by one of the two deskew PDs or by the phase shift interface. Each deskew PD takes in two clock inputs, CLKIN1_DESKEW/CLKIN2_DESKEW and CLKFB1_DESKEW/CLKFB2_DESKEW and adjusts the PIs controlled by the deskew network to drive the delay between the rising edges of the two clocks to zero. For more information about requirements on the sets of clock inputs, see Functioning of Deskew.

IMPORTANT! *The MMCM/XPLL/DPLL output clocks using this deskew scheme are not phase aligned to the other output clocks.*

A phase shift interface system is also present. The PIs can be chosen to be controlled by nothing (not used), either of the two deskew PDs, or by the phase shift interface. Using this interface, the PIs under the control of phase shift interface can be incremented or decremented by 1-step dynamically while the MMCM is locked and operational. This includes the feedback PI (PIFB).

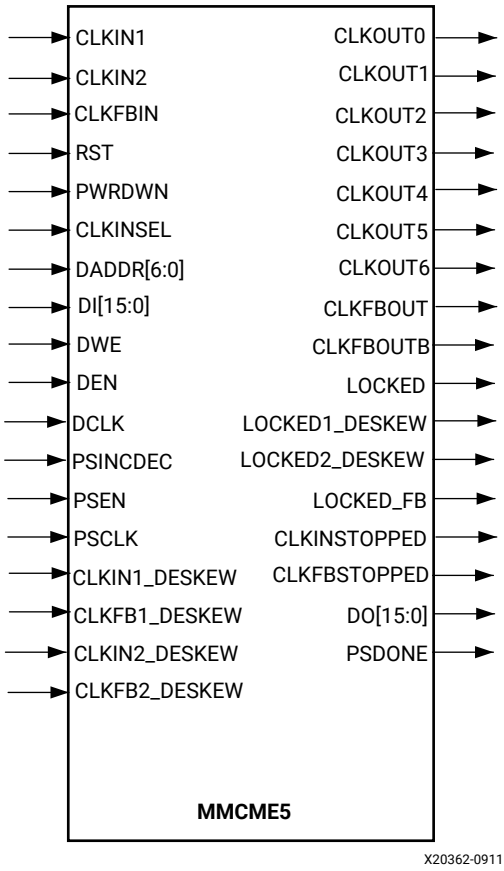
Figure 28: **MMCM Block Diagram**



MMCM Primitive

The Versal device MMCME5 primitive is shown in the following figure.

Figure 29: **MMCME5 Primitive**



The MMCM 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 in this section. The VCO operating frequency can be determined by using the following relationship:

$$F_{vco} = F_{clkin} \times \frac{M}{D}$$

$$F_{clkout} = F_{clkin} \times \frac{M}{D \times O}$$

where the M, D, and O counters are shown in Figure 28. The value of M corresponds to the CLKFBOUT_MULT_F setting, the value of D to the DIVCLK_DIVIDE, and O to the CLKOUT_DIVIDE. The seven 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 because a single VCO drives all the counters.

Frequency Synthesis and Deskew

Frequency synthesis and deskew options are available in Versal ACAP MMCM/PLL primitives. Use the examples provided in the MMCM Use Models chapter for all options, including PLL use models.

Without Deskew

The MMCM can be used for standalone frequency synthesis and is not used to deskew a clock network. The MMCM simply generates output clock frequencies for other blocks. In this mode, the MMCM feedback paths are internal, which keeps all the routing local, minimizing the jitter. Furthermore, no input clock alignment is done in this mode. Deskewing output clock networks to each other is still possible using deskew logic.

With CLKFBOUT Deskew

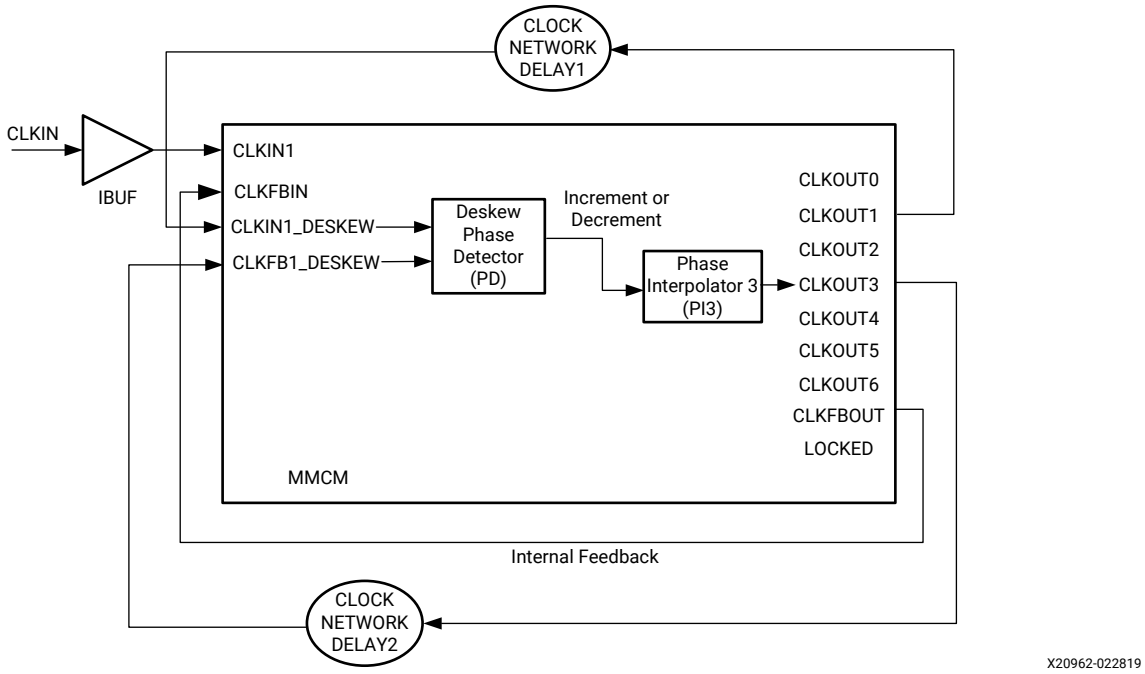
In many cases, designers do not want to incur the delay on a clock network in their I/O timing budget. Therefore, an MMCM is used to compensate for the clock network delay. This feature is supported in Versal ACAPs. A clock output matching the reference clock CLKIN frequency (always CLKFBOUT) is connected to a clock buffer of the same type driving the logic and fed back to the CLKFBIN feedback pin of the MMCM. 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.

Using the Deskew Logic

In addition to the normal deskew options, internal feedback, or clock network delay feedback using CLKFBOUT, deskew logic features are also available in the MMCM and PLL of Versal ACAPs. MMCM and XPLL have two deskew logic units while a DPLL has one. A deskew logic unit consists of CLKIN_DESKEW and CLKFB_DESKEW pins along with a deskew phase detector (PD) and a set of shared phase interpolators (PI). Clock network deskew can be achieved using a deskew phase detector (PD) and with the limitation that the PI associated with the feedback counter cannot be used for deskewing using the deskew PD. An MMCM/XPLL can receive four additional clock inputs (CLKIN1_DESKEW/CLKIN2_DESKEW and CLKFB1_DESKEW/CLKFB2_DESKEW) to help eliminate skew between two pairs of clock networks while with two additional inputs (CLKIN1_DESKEW and CLKFB1_DESKEW) a DPLL can only deskew one clock network. Note that both the CLKIN_DESKEW and CLKFB_DESKEW clock network pairs must have the same frequency. For more information about requirements on the sets of clock inputs, see Functioning of Deskew. The duty cycles of the two inputs are not required to be the same provided that they are within specification and the phases can be different. Each of the seven output counters and the feedback counter receive input from their associated PI. The PIs can be controlled by either one of the two deskew PDs, the phase shift interface, or by neither. It is possible to deskew two different pairs of clock networks using this system. There is also an

option to add programmable delay to either of the two clock inputs of a deskew PD. The Vivado tools calculate the proper delay values for fine tuning the skew compensation. However, it is also possible to manually enable or disable the delays, adjust the delay line settings, and select the multiplexer CLKIN_DESKEW or CLKFB_DESKEW inputs using the attributes DESKEW_DELAY_EN, DESKEW_DELAY, and DESKEW_DELAY_PATH. The analog external feedback (CLKFBOUT) has limitations on the amount of delay that can be added to the external feedback path before making the MMCM unstable. This delay limitation does not apply to the digital deskew. Hence using the digital deskew is more suitable for feedback that is going outside the device and coming back in because the delay tends to be longer in such use cases.

Figure 30: **MMCM Clock Network Deskew Using Deskew Logic**



Using Fractional Divide

Versal ACAPs support fractional (non-integer) divides in the feedback path resulting in a fractional VCO frequency. The method of fractional divide from UltraScale™ devices has been replaced with a sigma-delta module (SDM) based fractional mode. The MMCM now supports fractional divide ratios with a 6-bit resolution which is far finer than the 0.125 resolution supported in the UltraScale devices. The CLKOUT0 output counter does no longer have a fractional capability. The MMCM supports a 6-bit fractional modulus which enables it to synthesize frequency in a much finer granularity.

To use the integer mode, fractional SDM must be disabled. The fractional modulus of the MMCM feedback divide is 6-bit wide, hence the frequency can be set to any fractional value equal to n/64, where n = 1, 2, ..., 63, or expressed differently $1/(2^6)$ $2/(2^6)$, ... , $63/(2^6)$. When in fractional mode, it is advised to choose M value that generates the highest possible VCO frequency within the VCO operating range to achieve the best possible jitter performance. The Clocking Wizard automatically implements all the necessary settings when fractional divide is used.

Jitter Filter

MMCMs like all PLLs reduce the jitter inherent on a reference clock. The MMCM can be instantiated as a stand-alone function to only support filtering jitter from an external clock before it is driven into another block. As a jitter filter, it is usually assumed that the MMCM act as a buffer and regenerate the input frequency on the output (for example, F_{IN} = 100 MHz, F_{OUT} = 100 MHz). In general, greater jitter filtering is possible by using the MMCM attribute BANDWIDTH set to Low. Setting the BANDWIDTH to Low can incur an increase in the static offset of the MMCM. Users can find jitter and offset values for the configuration they have selected in the clocking wizard.

Limitations

The MMCM has some restrictions that must be adhered to. These are summarized in the MMCM electrical specifications in the Versal ACAP data sheets. In general, the major limitations are VCO operation range, input frequency, duty cycle programmability, and phase shift. In addition, there are connectivity limitations to other clocking elements (pins, GTs, and clock buffers). Cascading MMCMs can only occur through the clock routing network.

VCO Operating Range

The minimum and maximum VCO operating frequencies are defined in the electrical specification of the Versal ACAP data sheets. These values can also be extracted from the speed specification.

Minimum and Maximum Input Frequencies

The minimum and maximum CLKIN input frequencies are defined in the electrical specification of the Versal ACAP data sheets.

Duty Cycle Programmability

Only discrete duty cycles are possible given a VCO operating frequency. Depending on the CLKOUT_DIVIDE value, a minimum and maximum range is possible with a step size that is also dependent on the CLKOUT_DIVIDE value. The Clocking Wizard tool gives the possible values for a given CLKOUT_DIVIDE.

The only allowed duty cycle for fractional divides is 0.5 or 50%. For non-fractional divides, the granularity of the CLKOUT duty cycle can be calculated as 0.5/CLKOUT_DIVIDE. For CLKOUT_DIVIDE = 1, the only allowed duty cycle is 0.5 or 50%. For 1 < CLKOUT_DIVIDE ≤ 256, the minimum duty cycle is 1/CLKOUT_DIVIDE and the maximum is (CLKOUT_DIVIDE – 1)/ CLKOUT_DIVIDE. For CLKOUT_DIVIDE > 256, the minimum duty cycle is (CLKOUT_DIVIDE – 255)/CLKOUT_DIVIDE and maximum duty cycle is 255/CLKOUT_DIVIDE.

Phase Shift

In many cases, there needs to be a phase shift between clocks. The MMCM has multiple options to implement phase shifting. Static phase shifting can be achieved by selecting one of the 1/32 VCO phases that are generated by the PI feeding each counter. Dynamic phase shifting can also be performed after the MMCM is locked and operational. The phase shift can be incremented or decremented dynamically in resolution of 1/32 of VCO period. The MMCM phase shifting capabilities are very powerful and can lead to complex scenarios. By using the Clocking Wizard, the allowable phase shift values are determined based on the MMCM configuration settings.

Static Phase Shift Mode Using Counters and VCO Phases (MMCM and XPLL) or DCO Phases (DPLL)

The VCO can provide eight phase-shifted clocks at 45° each; thus always providing possible settings for 0°, 45°, 90°, 135°, 180°, 225°, 270°, and 315° of phase shift. The higher the VCO frequency is, the smaller the phase shift resolution. Because of the distinct operating range of the VCO, it is possible to bound the phase shift resolution using from ($1/8F_{vcomin}$) to ($1/8F_{vcomax}$) ps.

***Note*:** Due to the phase interpolator (PI) in front of every counter, the output and feedback counters can operate on increments of 1/32nd of the VCO period. This applies to the static phase shift in Versal™ ACAPs.

To achieve static phase shift between outputs of the dividers O0-O6 and feedback counter M for MMCM and XPLL or outputs of the dividers O0-O3 for DPLL, the PIs feeding the counters can be chosen to have different phase step values. Each PI has 32 possible step values that choose one of the 32 divided phases of the VCO or DCO cycle. During static phase shift, only the 0,4,8, …,28 step values can be chosen. The minimum phase resolution achieved by the PI is given by (replace VCO by DCO in the equation for DPLL):

$$Minimum\ phase\ step\ resolution\left(ps\right) = \frac{(VCO\ period\ in\ ps)}{4 \times 32}$$

Note that even though the phase difference between the counters in ps remains unchanged by the divide ratio of the counters, the phase difference in degrees varies depending on the divide ratio. Expressed as degrees relative to output, the minimum phase resolution is:

$$Minimum\ phase\ step\ resolution\ (degrees) = \frac{360}{CLKOUT\_DIVIDE} \times \frac{4}{32}$$

For example, by choosing phase step 0 for PI0 and phase step 16 for PI1, a nominal phase difference of 0.5 × VCO period or 0.5 x DCO period between the output of O0 and O1 can be achieved. For undivided clocks, this translates to a phase difference of 180 degrees. However, if both the counters are in divide-by-2 mode, the phase difference between them is 90 degrees. If both the counters are in divide-by-4 mode, the phase difference between them is 45 degrees.

In addition, each counter is capable of adding an additional phase shift that can be multiples of the VCO period or DCO period. Translating this to degrees at the output of the counters, each counter can add an additional phase shift with a resolution of 360/CLKOUT_DIVIDE degrees. The maximum phase shift range is also determined by the CLKOUT_DIVIDE value. The maximum phase shift is 360° when CLKOUT_DIVIDE is less than or equal to 256. When CLKOUT_DIVIDE is > 256, the maximum phase shift is:

$$Maximum\ Phase\ Shift\ (degrees) = \left(255 + \frac{28}{32}\right) \times \frac{360}{CLKOUT\_DIVIDE}$$

The phase shift can be provided as a negative number. This is implemented by just adding 360° to the negative phase to get a positive phase value. Therefore, the negative phase shift range is –360 to maximum positive phase shift 360.

It is possible to phase shift the CLKFBOUT feedback clock. In that case all CLKOUT output clocks are negatively phase shifted with respect to CLKIN. The PI step value and the counter delay value would be automatically determined from the CLKFBOUT_PHASE and CLKOUT[0:6]_PHASE values.

Dynamic Interpolated Fine Phase Shift in MMCM and XPLL (variable phase shift)

Interpolated fine phase shift mode has a linear shift behavior independent of the CLKOUT_DIVIDE/CLKFBOUT_MULT value. There is an individual phase interpolator (PI) for each of the O clock output counters and the M counter in the MMCM. The phase shift resolution only depends on the VCO frequency. The phase shift resolution is 1/32nd of the VCO frequency ((VCO period in ps)/32)) and is based on one of the eight phases out of the VCO selected. In this mode, the output clocks can be rotated 360° round robin. The phase interpolators can be controlled by either of the two deskew phase detectors (PDs) or by the phase shift interface.

$$Phase\ step\ resolution\ \left(period\right) = \frac{(VCO\ period\ in\ ps)}{32}$$

If the VCO runs at 3 GHz, the phase resolution is approximately (rounded) 10 ps, and at 4 GHz it is approximately (rounded) 8 ps.

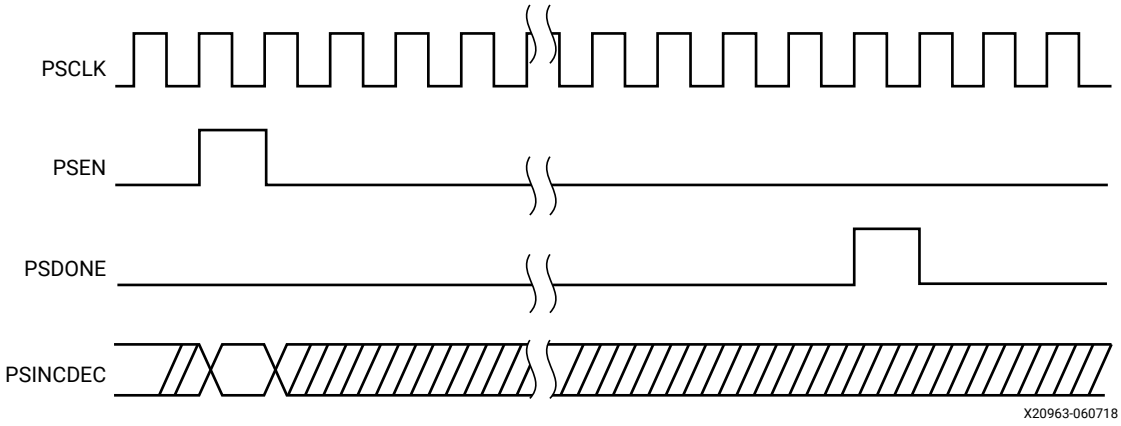
When using fine phase shift, the initial phase step value of every PI can be independently setup. The phase can be dynamically incremented or decremented. The dynamic phase shift is controlled by the PS interface of the MMCME5_ADV. This phase shift mode affects each individual CLKOUT output. In interpolated fine phase shift mode, a clock must always be connected to the PSCLK pin of the MMCM. Fixed or dynamic phase shifting of the feedback path results in a negative phase shift of all output clocks with respect to CLKIN.

Dynamic Phase Shift Interface in the MMCM

The MMCME5_ADV primitive provides three inputs and one output for dynamic fine phase shifting. Each CLKOUT and the CLKFBOUT divider can be individually selected for phase shifting. The dynamic phase shift amount is common to all the output clocks selected.

The variable phase shift is controlled by the PSEN, PSINCDEC, PSCLK, and PSDONE ports (see the following figure). The phase of the MMCM output clock(s) increments/decrements according to the interaction of PSEN, PSINCDEC, PSCLK, and PSDONE from the initial or previously performed dynamic phase shift. PSEN, PSINCDEC, and PSDONE are synchronous to PSCLK. When PSEN is asserted for one PSCLK clock period, a phase shift increment/decrement is initiated. When PSINCDEC is High, an increment is initiated and when PSINCDEC is Low, a decrement is initiated. Each increment adds to the phase shift of the MMCM clock outputs by 1/32nd of the VCO period. Similarly, each decrement decreases the phase shift by 1/32nd of the VCO period. PSEN must be active for one PSCLK period. PSDONE is High for exactly one-clock period when the phase shift is complete. The number of PSCLK cycles (12) is deterministic. After initiating the phase shift by asserting PSEN and the completion of the phase shift signaled by PSDONE, the MMCM output clocks gradually drift from their original phase shift to an increment/decrement phase shift in a linear fashion. The completion of the increment or decrement is signaled when PSDONE asserts High. After PSDONE has pulsed High, another increment/decrement can be initiated. There is no maximum phase shift or phase shift overflow. An entire clock period (360°) can always be phase shifted regardless of frequency. When the end of the period is reached, the phase shift wraps around round-robin style. In the case where there is no additional phase shift initiated (PSEN stays Low), PSDONE continues to repeat a one-cycle High pulse every 32 PSCLK cycles.

Figure 31: **Phase Shift Timing Diagram**



Dynamic Reconfiguration Port (DRP)

In most circumstances, MMCM and/or PLL used in a design are configured using static-calculated values to set up the used outputs. A wizard can be used to calculate all values and generate a instantiable wrapper containing a configured MMCM or PLL or one can instantiate the MMCM and/or PLL primitive and calculate the values separately to make the primitive function correctly using the formula given in this guide.

The DRP port provides the ability to use an MMCM and/or PLL as a dynamic element in a design. The DRP port setup is that of a common microcontroller peripheral and gives the user access to a set of registers in the MMCM or PLL. These registers allow the user to fully control the MMCM or PLL. Inputs pins and the values to define output clocks are turned into register bits making it possible to use the primitives as active elements in a design.

The DRP port connections and port descriptions are shown in the following figure and table.

Figure 32: **DRP Port**

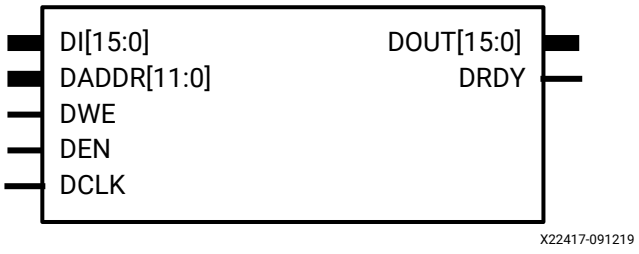


Table 9: **DRP Port Signals**

Port	Size	I/O	Description
DCLK	1	input	The DCLK signal is the reference clock for the dynamic reconfiguration port. This clock is normally about 100 MHz to 200 MHz.

Table 9: **DRP Port Signals** *(cont'd)*

Port	Size	I/O	Description
DEN	1	input	The dynamic reconfiguration enable (DEN) provides the enable control signal to access the dynamic reconfiguration feature. When the dynamic reconfiguration feature is not used, DEN must be tied Low.
DWE	1	input	The dynamic reconfiguration write enable (DWE) input pin provides the write enable control signal to write the data on the DI port into the register selected by the DADDR address. When not used, DWE must be tied Low.
DADDR	7	input	The dynamic reconfiguration address (DADDR) input bus provides a reconfiguration address to access a specific register in the primitive for dynamic reconfiguration. When not used, all bits must be assigned zeros.
DI	16	input	The dynamic reconfiguration data input (DI) bus provides reconfiguration data that writes into a specified address (DADDR) of the register set. When not used, all bits must be set to zero.
DO	16	output	The dynamic reconfiguration output bus provides data output of the register selected by the DADDR bus. This port can be used to read DRP register contents.
DRDY	1	output	The dynamic reconfiguration ready output (DRDY) provides the response to the DEN signal for the PLL's dynamic reconfiguration feature. This signal is pulsed High when a write or read operation is successful.

Unlike previous generations, the protocol on the DRP signals in Versal ACAPs has been changed to be an APB3-compatible interface. The table below shows the mapping between DRP and APB3 signals. Note that the APB3 PSEL signal is not supported, and thus, the MMCM or PLL DRP ports must be the only slave peripheral on an APB3 bus because the DRP interface is always selected and responds to all APB3 bus transactions.

Table 10: **Mapping Between DRP and APB3 Signals**

DRP signals	APB3 signals	I/O	Size
DCLK	PCLK	input	1
DADDR	PADDR	input	7
DWE	PWRITE	input	1
DEN	PENABLE	input	1
DI	PWDATA	input	16
DOUT	PRDATA	output	16
DRDY	PREADY	output	1

It is strongly recommended to use clocking wizard to setup dynamic reconfiguration. When using the clocking wizard, the DRP port (running on the APB3 protocol) can be enabled via an AXI4-Lite controller. To see how the clocking wizard provides an AXI4-Lite interface for the dynamic reconfiguration of the clocking primitives MMCM/XPLL/DPLL, refer to the *Clocking Wizard for Versal ACAP LogiCORE IP Product Guide* (PG321).

The DRP port running on the APB3 protocol is intended to be used over standard AXI connections to simplify interface connections. Contact Xilinx support if the intended application cannot use the AXI protocol.

The APB3-compatible interface is based on the APB3 interface, which is described in the AMBA® APB Protocol Specification v2.0, together with read and write timing waveforms.

MMCM Programming

Programming of the MMCM must follow a set flow to ensure configuration that guarantees stability and performance. This section describes how to program the MMCM based on certain design requirements. A design can be implemented in two ways: 1) directly through the GUI interface (the Clocking Wizard) or 2), implementing the MMCM through instantiation. Regardless of the method selected, the following information is necessary to program the MMCM:

- Reference clock period
- Output clock frequencies (up to seven maximum)
- Output clock duty cycle (default is 50%)
- Output clock phase shift in number of degrees relative to the original 0 phase of the clock
- Desired bandwidth of the MMCM (default is OPTIMIZED and the bandwidth is chosen in software)
- Compensation mode (automatically determined by the software)
- Reference clock jitter in UI (that is, a percentage of the reference clock period)

Using the following equations, the input frequency, D, M, and O values can be determined.

$$F_{vco} = F_{clkin} \times \frac{M}{D}$$

$$F_{clkout} = F_{clkin} \times \frac{M}{D \times O}$$

The best performance can be achieved by observing the following guidelines:

- Choose the highest input/reference clock frequency as possible to keep M Low. This allows for higher PLL bandwidths (which helps to filter reference noise).

Direct instantiation of the MMCM requires that the D, M (M+F) and O (Seven possible O values) are manually calculated. The following formulas help to determine the allowed M and D values:

$$D_{MIN} = roundup\left(\frac{F_{CLKIN}}{F_{PFDMAX}}\right)$$

$$D_{MAX} = roundup\left(\frac{F_{CLKIN}}{F_{PFDMIN}}\right)$$

$$M_{MIN} = roundup\left(\frac{F_{VCOMIN}}{F_{CLKIN}} \times D_{MIN}\right)$$

$$M_{MAX} = roundup\left(\frac{F_{VCOMIN}}{F_{CLKIN}} \times D_{MAX}\right)$$

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.

$$M_{IDEAL} = \frac{D_{MIN} \times F_{VCOMAX}}{F_{CLKIN}}$$

M is the representation of a value M+(F/2^6), where M is the integer value, and F is the integer value to calculate the fraction *n*.

$$n = \frac{F}{2^6}$$

$$F_{rounddown} = 2^6 \times n$$

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.

MMCM Ports

The following table summarizes the MMCM ports.

Table 11: **MMCM Ports**

Port Name	I/O	Description
CLKIN1	Input	General clock input.
CLKIN2	Input	Secondary clock input for the MMCM reference clock.
CLKFBIN	Input	Feedback clock input.
CLKINSEL	Input	This signal controls the state of the clock input MUX, High = CLKIN1, and Low = CLKIN2. CLKINSEL dynamically switches the MMCM reference clock.
CLKOUT[0:6]	Output	User configurable clock outputs (0 through 6) that can be divided versions of the VCO phase outputs (user controllable) from 2 to 512. The output clocks are phase aligned to each other (unless phase shifted) and aligned to the input clock with a proper feedback configuration.
CLKFBOUT	Output	Dedicated MMCM feedback output.
CLKINSTOPPED	Output	Status pin indicating that the input clock has stopped.

Table 11: **MMCM Ports** *(cont'd)*

Port Name	I/O	Description
CLKIN1_DESKEW	Input	Primary clock input to the phase detector 1 block for deskewing clock network delays between two different CLKOUT networks.
CLKFB1_DESKEW	Input	Secondary (feedback) clock input to the phase detector1 block for deskewing clock network delays.
CLKIN2_DESKEW	Input	Primary clock input to the phase detector 2 block for deskewing clock network delays between two different CLKOUT networks.
CLKFB2_DESKEW	Input	Secondary (feedback) clock input to the phase detector2 block for deskewing clock network delays.
CLKFBSTOPPED	Output	Status pin indicating that the feedback clock has stopped.
DADDR[6:0]	Input	The dynamic reconfiguration address (DADDR) input bus provides a reconfiguration address for the dynamic reconfiguration. When not used, all bits must be assigned zeros.
DI[15:0]	Input	The dynamic reconfiguration data input (DI) bus provides reconfiguration data. When not used, all bits must be set to zero.
DO[15:0]	Output	The dynamic reconfiguration output bus provides MMCM data output when using dynamic reconfiguration.
DRDY	Output	The dynamic reconfiguration ready output (DRDY) provides the response to the DEN signal for the MMCM's dynamic reconfiguration feature.
DWE	Input	The dynamic reconfiguration write enable (DWE) input pin provides the write enable control signal to write the DI data into the DADDR address. When not used, it must be tied Low.
DEN	Input	The dynamic reconfiguration enable (DEN) provides the enable control signal to access the dynamic reconfiguration feature. When the dynamic reconfiguration feature is not used, DEN must be tied Low.
DCLK	Input	The DCLK signal is the reference clock for the dynamic reconfiguration port.
LOCKED	Output	The LOCKED signal indicates that all functions requiring a LOCKED signal for the MMCM to operate properly have LOCKED. This LOCKED signal is therefore an AND function of LOCKED_FB and both LOCKED1/2_DESKEWs, if used.
LOCKED_FB	Output	An output from the MMCM that indicates when the MMCM has achieved phase alignment within a predefined window and frequency matching within a predefined PPM range. The MMCM automatically locks after power on. No extra reset is required. LOCKED is deasserted if the input clock stops or the phase alignment is violated (for example, input clock phase shift). The MMCM must be reset after LOCKED is deasserted.
LOCKED1/2_DESKEW	Output	Indicates if the deskew circuit is locked. Applies only to the deskew circuits used in the design. Ignore these outputs for unused deskew circuits.
PSCLK	Input	Phase shift clock.
PSEN	Input	Phase shift enable.
PSINCDEC	Input	Phase shift increment/decrement control.
PSDONE	Output	Phase shift done.
RST	Input	Asynchronous reset signal. The RST signal is an asynchronous reset for the MMCM. The MMCM synchronously re-enables itself when this signal is released (that is, MMCM re-enabled). A reset is required when the input clock conditions change (for example, frequency).
PWRDWN	Input	Powers down instantiated but unused MMCMs.

Notes:

1. All control and status signals except PSINCDEC are active-High.
2. See the Dynamic Reconfiguration Port (DRP) section for further information and recommendations on dynamic reconfiguration ports.

TIP: *The port names generated by the clocking wizard can differ from the port names used on the primitive.*

MMCM Port Descriptions

- **CLKIN1 – Primary Reference Clock Input:** General clock input. Clocks can come from GC pins in adjacent XPIO banks, horizontal routing and vertical distribution. Therefore, all clock buffers can drive the CLKIN1 input.
- **CLKIN2 – Secondary Clock Input:** Secondary clock input to dynamically switch the MMCME5. Connectivity is identical to CLKIN1.
- **CLKFBIN – Feedback Clock Input:** CLKFBIN must be connected either directly to the CLKFBOUT for internal feedback, or to the CLKFBOUT through a BUFG for clock buffer feedback matching, or IBUFG (through a global clock pin for external deskew) or interconnect (not recommended). For clock alignment, the feedback path clock buffer type should match the forward clock buffer type.

  IMPORTANT! *The internal compensation mode setting is determined by a direct connection (wire) from the CLKFBOUT to the CLKFBIN port in the source. However, synthesis optimizes this connection away such that the CLKFBOUT to CLKFBIN connection is removed from all subsequent representations in the Vivado® Design Suite. However, the INTERNAL compensation attribute attached to the MMCM/PLL indicates that the compensation is still internal to the MMCM/PLL.*

- **CLKFBOUT – Dedicated MMCM Feedback Output:** For possible configuration of CLKFBOUT. CLKFBOUT can also drive logic if the feedback path contains a clock buffer.
- **CLKINSEL – Clock Input Select:** The CLKINSEL signal controls the state of the clock input multiplexers. High = CLKIN1, Low = CLKIN2. The MMCM must be held in RESET during clock switchover.
- **CLKOUT[6:0] – Output Clocks:** These user-configurable clock outputs (CLKOUT0 through CLKOUT6) can be divided versions of the VCO phase outputs (user controllable) from 2 to 128. The input clock and output clocks can be phase aligned. For possible configurations, see MMCM Use Models. CLKFBOUT can be used to infer fractional mode for all CLKOUT outputs. The CLKOUT outputs can thus be used in fractional or non-fractional mode depending on the VCO mode. A static or dynamic phase shift can be applied to individual outputs. See the Phase Shift and Using Fractional Divide sections for more information.
- **CLKIN1_DESKEW:** Primary clock input to the phase detector 1 block for deskewing clock network delays between two different CLKOUT networks. Use this pin to select one of the output clock that needs deskewing with respect to another output clock.
- **CLKFB1_DESKEW:** Primary (feedback) clock input to the phase detector 1 block for deskewing clock network delays. Use this pin to select the output clock that needs deskewing with respect to the output clock selected for the CLKIN1_DESKEW.

- **CLKIN2_DESKEW:** Secondary clock input to the phase detector 2 block for deskewing clock network delays between two different CLKOUT networks. Use this pin to select a second output clock that needs deskewing with respect to another output clock.
- **CLKFB2_DESKEW:** Secondary (feedback) clock input to the phase detector 2 block for deskewing clock network delays. Use this pin to select the output clock that needs deskewing with respect to the output clock selected for the CLKIN2_DESKEW.
- **CLKINSTOPPED – Input Clock Status:** This is a status pin indicating that the input clock has stopped. This signal is asserted within two CLKFBOUT clock cycles of clock stoppage. The signal is deasserted after the clock has restarted and LOCKED is achieved, or the clock is switched to the alternate clock input and the MMCM has re-locked.
- **CLKFBSTOPPED – Feedback Clock Status:** This is a status pin indicating that the feedback clock has stopped. CLKFBSTOPPED is asserted within one clock cycle of clock stoppage. The signal is deasserted after the feedback clock has restarted and the MMCM has re-locked.
- **DADDR[6:0] – Dynamic Reconfiguration Address:** The dynamic reconfiguration address (DADDR) input bus provides a reconfiguration address for dynamic reconfiguration. The address value on this bus specifies the 16 configuration bits that are written or read with the next DCLK cycle. When not used, all bits must be assigned zeros.
- **DI[15:0] – Dynamic Reconfiguration Data Input:** The dynamic reconfiguration data input (DI) bus provides reconfiguration data. The value of this bus is written to the configuration cells. The data is presented in the cycle that DEN and DWE are active. The data is captured in a shadow register and written at a later time. DRDY indicates when the DRP port is ready to accept another write. When not used, all bits must be set to zero.
- **DO[15:0] – Dynamic Reconfiguration Output Bus:** The dynamic reconfiguration output bus provides MMCM data output when using dynamic reconfiguration. If DWE is inactive while DEN is active at the rising edge of DCLK, this bus holds the content of the configuration cells addressed by DADDR. The DO bus must be captured on the rising edge of DCLK when DRDY is active. The DO bus value is held until the next DRP operation.
- **DRDY – Dynamic Reconfiguration Ready:** The dynamic reconfiguration ready output (DRDY) provides the response to the DEN signal for the MMCM's dynamic reconfiguration feature. This signal indicates that a DEN/ DCLK operation has completed.
- **DWE – Dynamic Reconfiguration Write Enable:** The dynamic reconfiguration write enable (DWE) input pin provides the write/read enable control signal to write the DI data into or read the DO data from the DADDR address. When not used, it must be tied Low.
- **DEN – Dynamic Reconfiguration Enable Strobe:** The dynamic reconfiguration enable strobe (DEN) provides the enable control signal to access the dynamic reconfiguration feature and enables all DRP port operations. When the dynamic reconfiguration feature is not used, DEN must be tied Low.

- **DCLK – Dynamic Reconfiguration Reference Clock:** The DCLK signal is the reference clock for the dynamic reconfiguration port. The rising edge of this signal is the timing reference for all other port signals. The setup time is specified in the Versal ACAP data sheets. There is no hold time requirement for the other input signals relative to the rising edge of the DCLK. The pin can be driven by an IBUF, IBUFG, BUFGCE, or BUFGCTRL. There are no dedicated connections to this clock input.
- **LOCKED, LOCKED1/2_DESKEW, and LOCKED_FB:** LOCKED_FB is an output from the MMCM used to indicate when the MMCM has achieved phase and frequency alignment of the reference clock and the feedback clock at the input pins. Phase alignment is within a predefined window and frequency matching within a predefined PPM range. The MMCM automatically locks after power on; no extra reset is required. LOCKED_FB is deasserted within one PFD clock cycle if the input clock stops, the phase alignment is violated (for example, input clock phase shift), or the frequency has changed. The LOCKED1/2_DESKEW outputs indicate if the optional deskew circuit is used and has locked. The LOCKED output signals that the LOCKED_FB and LOCKED1/2_DESKEW circuits have achieved LOCK. Only unused LOCKED1/2_DESKEW circuits are considered for this AND function. The MMCM must be reset when LOCKED is deasserted. The clock outputs should not be used prior to the assertion of LOCKED.
- **PSCLK – Phase Shift Clock:** This input pin provides the source clock for the dynamic phase shift interface. All other inputs are synchronous to the positive edge of this clock. The pin can be driven by an IBUF, IBUFG, BUFG, or BUFGCE. There are no dedicated connections to this clock input.
- **PSEN – Phase Shift Enable:** A dynamic (variable) phase shift operation is initiated by synchronously asserting this signal. PSEN must be activated for one cycle of PSCLK. After initiating a phase shift, the phase is gradually shifted until a High pulse on PSDONE indicates that the operation is complete. There are no glitches or sporadic changes during the operation. From the start to the end of the operation, the phase is shifted in a continuous analog manner.
- **PSINCDEC – Phase Shift Increment/Decrement Control:** This input signal synchronously indicates if the dynamic phase shift is an increment or decrement operation (positive or negative phase shift). PSENCDEC is asserted High for increment and Low for decrement. There is no phase shift overflow associated with the dynamic phase shift operation. If 360° or more are shifted, the phase wraps around, starting at the original phase.
- **PSDONE – Phase Shift Done:** The phase shift done output signal is synchronous to the PSCLK. When the current phase shift operation is completed, the PSDONE signal is asserted for one clock cycle indicating that a new phase shift cycle can be initiated. If there is no new phase-shift cycle initiated, PSDONE continues to repeat a one-cycle High pulse every 32 PSCLK cycles.
- **RST – Asynchronous Reset Signal:** The RST signal is an asynchronous reset for the MMCM. The MMCM is synchronously re-enabled when this signal is deasserted.

- **PWRDWN – Power Down:** This signal powers down instantiated but currently unused MMCMs. This mode can be used to save power for temporarily inactive portions of the design and/or MMCMs that are not active in certain system configurations. No MMCM power is consumed in this mode.

MMCM Attributes

The following table lists the attributes for the MMCME5 primitive.

Table 12: **MMCM Attributes**

Attribute	Type	Allowed Values	Default	Description
BANDWIDTH	String	HIGH LOW OPTIMIZED	OPTIMIZED	Specifies the MMCM programming algorithm affecting the jitter, phase margin, and other characteristics of the MMCM.
CLKOUT[0:6]_DIVIDE	Decimal	2 to 511	2	Specifies the amount to divide the associated CLKOUT clock output if a different frequency is desired. This number, in combination with the CLKFBOUT_MULT_F and DIVCLK_DIVIDE values, determines the output frequency.
CLKOUT[0:6]_PHASE	Real	–360.000 to 360.000	0.000	Specifies the output phase relationship of the associated CLKOUT clock output in number of degrees offset (that is, 90 indicates a 90° or ¼ cycle offset phase offset while 180 indicates a 180° offset or ½ cycle phase offset).
CLKOUT[0:6]_DUTY_CYCLE	Real	0.001 to 0.999	0.5	Specifies the duty cycle of the associated CLKOUT clock output as a percentage (that is, 0.50 generates a 50% duty cycle).
CLKFBOUT_MULT	Decimal	4 to 432	42	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, determines the output frequency.
CLKFBOUT_FRACT	Decimal	0 to 63	0	6-bit fractional M feedback divider in increments of 1/63. Generates a fraction of the CLKFBOUT_MULT value and adds it to CLKFBOUT_MULT.
DIVCLK_DIVIDE	Decimal	1 to 123	1	Specifies the division ratio for all output clocks with respect to the input clock. Effectively divides the CLKIN going into the PFD.
CLKFBOUT_PHASE	Real	–360.000 to 360.000	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 MMCM.

Table 12: **MMCM Attributes** *(cont'd)*

Attribute	Type	Allowed Values	Default	Description
REF_JITTER1 REF_JITTER2	Real	0.000 to 0.200	0.010	Allows specification of the expected jitter on the reference clock to better optimize MMCM performance. A bandwidth setting of OPTIMIZED attempts to choose the best parameter for input clocking when unknown. If known, the value provided should be specified in terms of the unit interval (UI) (the maximum peak-to-peak value) of the expected jitter on the input clock.
CLKIN1_PERIOD	Real	0.000 to 100.000	0.000	Specifies the input period in ns to the MMCM CLKIN1 input. Resolution is down to the ps. This information is mandatory and must be supplied.
CLKIN2_PERIOD	Real	0.000 to 100.000	0.000	Specifies the input period in ns to the MMCM CLKIN2 input. Resolution is down to the ps. This information is mandatory and must be supplied.
CLKOUTn_PHASE_CTRL[0:1]	Binary	00 to 11	00	CLKOUT[0:6] counter variable fine phase shift or deskew select. 00: Interpolator is not controlled by either deskew or phase shift interface. 01: Interpolator is controlled by deskew1. 10: Interpolator is controlled by phase shift interface. 11: Interpolator is controlled by deskew2.

Table 12: **MMCM Attributes** *(cont'd)*

Attribute	Type	Allowed Values	Default	Description
CLKOUTFB_PHASE_CTRL	Binary	00 to 11	00	CLKFBOUT counter variable fine phase shift or deskew select. 00: Interpolator is not controlled by either deskew or phase shift interface. 01: Interpolator is controlled by deskew1. 10: Interpolator is controlled by phase shift interface. 11: Interpolator is controlled by deskew2.
DESKEW_DELAY1	Decimal	0 to 63	0	Value of the optional programmable delay in the deskew1 circuit.
DESKEW_DELAY2	Decimal	0 to 63	0	Value of the optional programmable delay in the deskew2 circuit.
DESKEW_DELAY_PATH1	String	TRUE, FALSE	FALSE	Determines if the CLKIN1_DESKEW path or the CLKFB1_DESKEW path is selected for the optional programmable delay. TRUE = CLKIN1_DESKEW, FALSE = CLKFB1_DESKEW.
DESKEW_DELAY_PATH2	String	TRUE, FALSE	FALSE	Determines if the CLKIN2_DESKEW path or the CLKFB2_DESKEW path is selected for the optional programmable delay. TRUE = CLKIN2_DESKEW, FALSE = CLKFB2_DESKEW.
DESKEW_DELAY_EN1	String	FALSE, TRUE	FALSE	Set to TRUE to enable the optional programmable delay in the deskew circuit 1.
DESKEW_DELAY_EN2	String	FALSE, TRUE	FALSE	Set to TRUE to enable the optional programmable delay in the deskew circuit 2.

Table 12: **MMCM Attributes** *(cont'd)*

Attribute	Type	Allowed Values	Default	Description
COMPENSATION	String	AUTO, EXTERNAL, INTERNAL, BUF_IN	AUTO	Clock input compensation. Must be set to AUTO. Defines how the MMCM feedback is configured. EXTERNAL: Indicates a network external to the device is being compensated. INTERNAL: Indicates the MMCM is using its own internal feedback path so no delay is being compensated. BUF_IN: Indicates that the configuration does not match with the other compensation modes. The CLKIN and CLKFBIN pins are aligned in a way that a delay in the feedback path is compensated with respect to CLKIN.
SS_EN	String	FALSE, TRUE	FALSE	Enables spread spectrum generation.
SS_MODE	String	DOWN_LOW, DOWN_HIGH, CENTER_LOW, CENTER_HIGH	CENTER_HIGH	Controls the spread spectrum frequency deviation and the spread type.
SS_MOD_PERIOD	Decimal	4000–40000	10000	Specifies the spread spectrum modulation period (ns).
LOCK_WAIT	String	FALSE, TRUE	FALSE	Wait during the configuration startup for the MMCM to lock.

Notes:

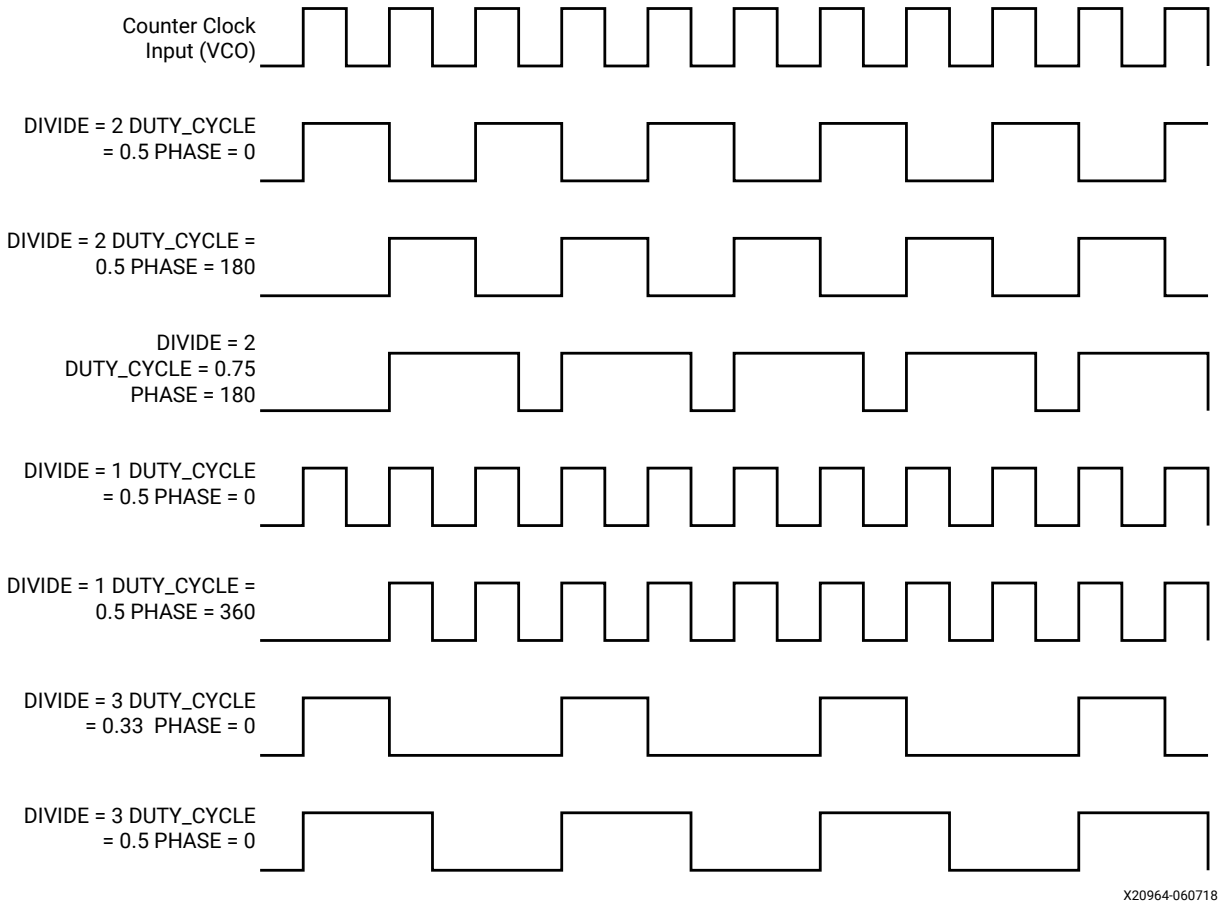
1. The COMPENSATION attribute values are documented for informational purpose only. The Vivado® tools automatically select the appropriate compensation based on circuit topology. Do not manually select a compensation value, leave the attribute at the default value.

TIP: *The port names generated by the clocking wizard can differ from the port names used on the primitive.*

Counter Control

The MMCM output counters provide a wide variety of synthesized clocks using a combination of DIVIDE, DUTY_CYCLE, and PHASE. The following figure illustrates how the counter settings impact the counter output. The top waveform represents the output from the VCO.

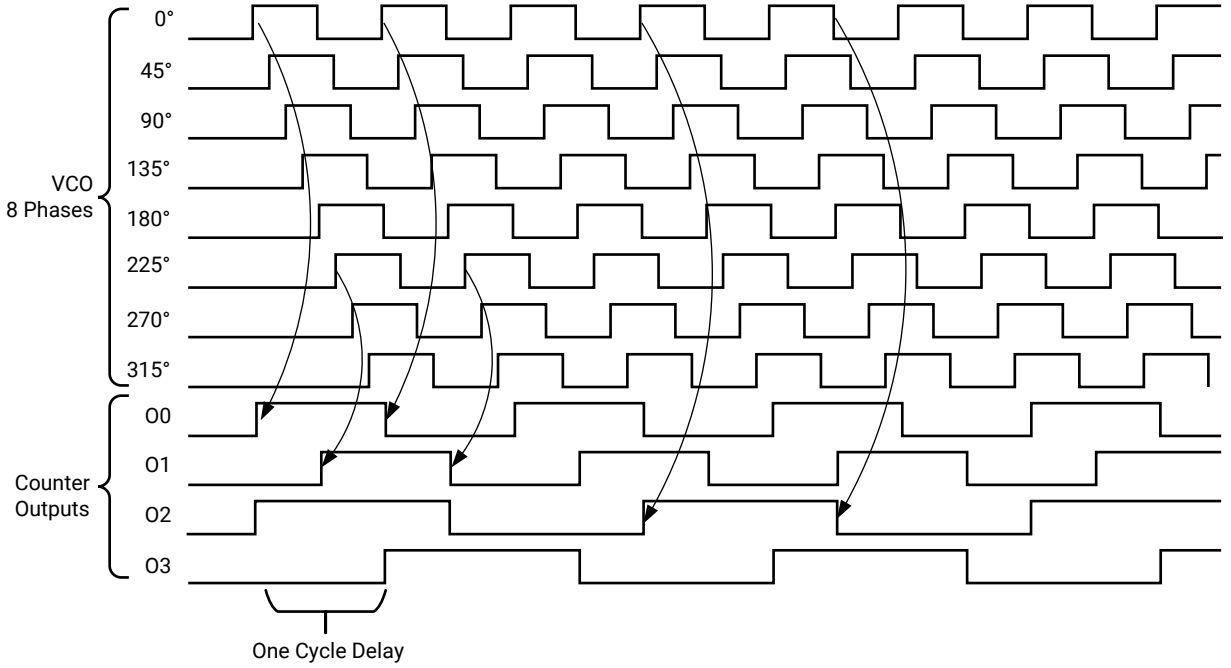
Figure 33: **Output Counter Clock Synthesis Examples**



Detailed VCO and Output Counter Waveforms

The following figure 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 ensure the correct phase is maintained. This means the rising edge of the 0° phase happens 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. This counter setting can 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. Phase shifts greater than one VCO period are possible.

Figure 34: **Selecting VCO Phases**



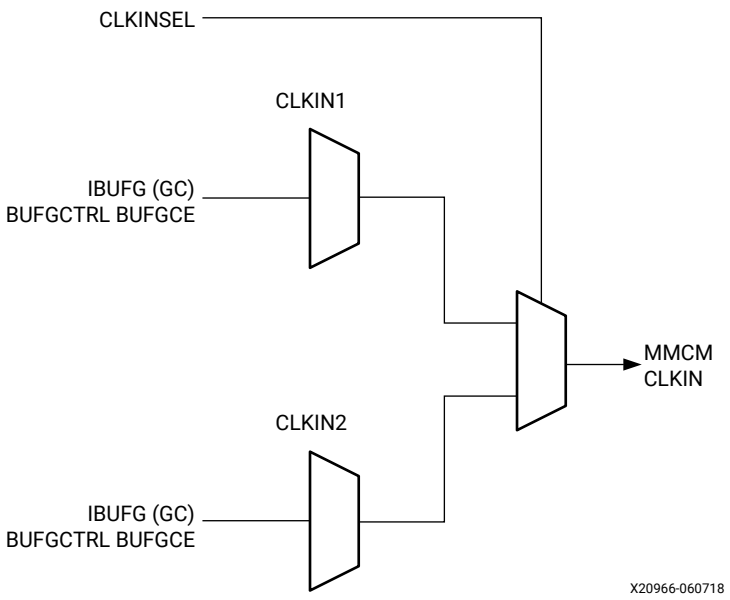
If the MMCM is configured to provide a certain phase relationship and the input frequency is changed, this phase relationship is also changed because the VCO frequency changes and therefore the absolute shift in picoseconds changes. This aspect must be considered when designing with the MMCM. When an important aspect of the design is to maintain a certain phase relationship among various clock outputs, (for example, CLK and CLK90), this relationship is maintained regardless of the input frequency.

All O counters can be equivalent; anything O0 can do, O1 can do. The O0 counter has the additional capability to be used in fractional divide mode. The MMCM outputs are flexible when connecting to the global clock network because they are identical. In most cases, this level of detail is imperceptible because the software and Clocking Wizard determine the proper settings through the MMCM attributes and Wizard inputs.

Reference Clock Switching

The MMCM reference clock can be dynamically switched using the CLKINSEL pin. The switching is done asynchronously. After the clock switches, the MMCM is likely to lose LOCKED and automatically lock onto the new clock. Therefore, after the clock switches, the MMCM must be reset. The MMCM clock MUX switching is shown in the following figure. The CLKINSEL signal directly controls the MUX. No synchronization logic is present.

Figure 35: **Input Clock Switching**



Missing Input Clock or Feedback Clock

When the input clock or feedback clock is lost, the CLKINSTOPPED or CLKFBSTOPPED status signal is asserted. The MMCM deasserts the LOCKED signal. After the clock returns, the CLKINSTOPPED signal is deasserted and a RESET must be applied.

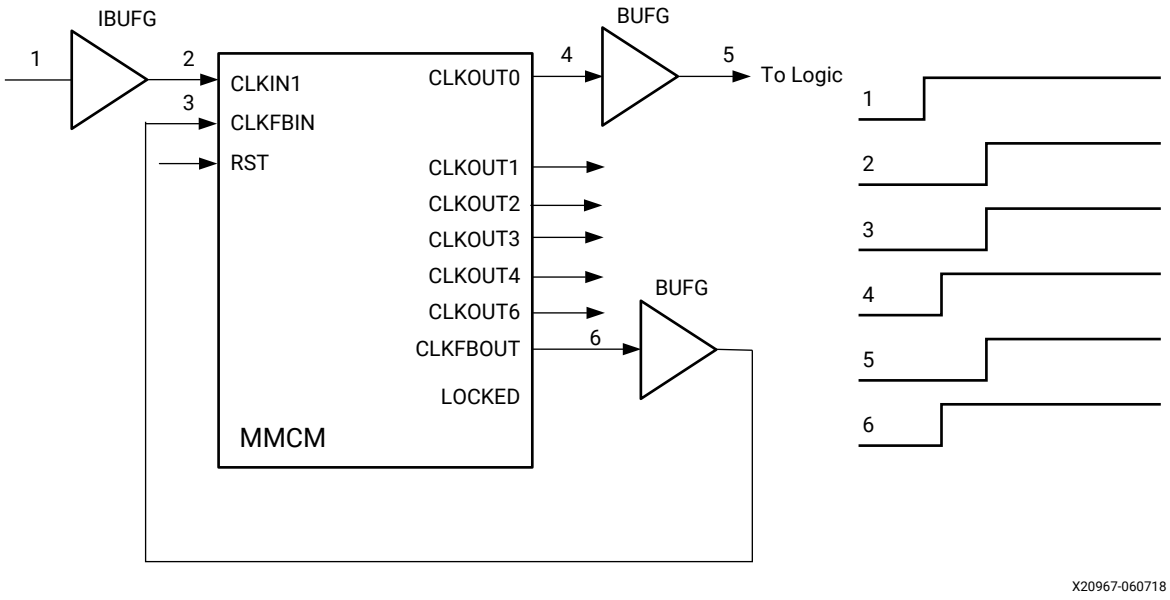
MMCM Use Models

The examples in this section show the MMCM. There are several methods to design with the MMCM. The Clocking Wizard in the Vivado® tools can assist with generating the various MMCM parameters. Additionally, the MMCM can be manually instantiated as a component. It is also possible for the MMCM to be merged with an IP core. The IP core would contain and manage the MMCM.

CLKIN Clock Network Deskew

One of the predominant uses of the MMCM is for clock network deskew. The following figure shows the MMCM in this mode. The clock output from one of the CLKOUT counters is used to drive logic within the device 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. The configuration in the following figure is the most flexible, but it does require two global clock networks.

Figure 36: **Global Clock Network Deskew Using Two BUFGs**



There are certain restrictions on implementing the feedback. The CLKFBOUT output can be used to provide the feedback clock signal. When an MMCM is driving both BUFGs and BUFGCTRL, only one of the clock buffers that is also used in the feedback path is deskewed. The fundamental restriction is that both input frequencies to the PFD must be identical. Therefore, this relationship must be met:

$$\frac{F_{in}}{\mathrm{D}} = F_{fb} = \frac{F_{vco}}{M}$$

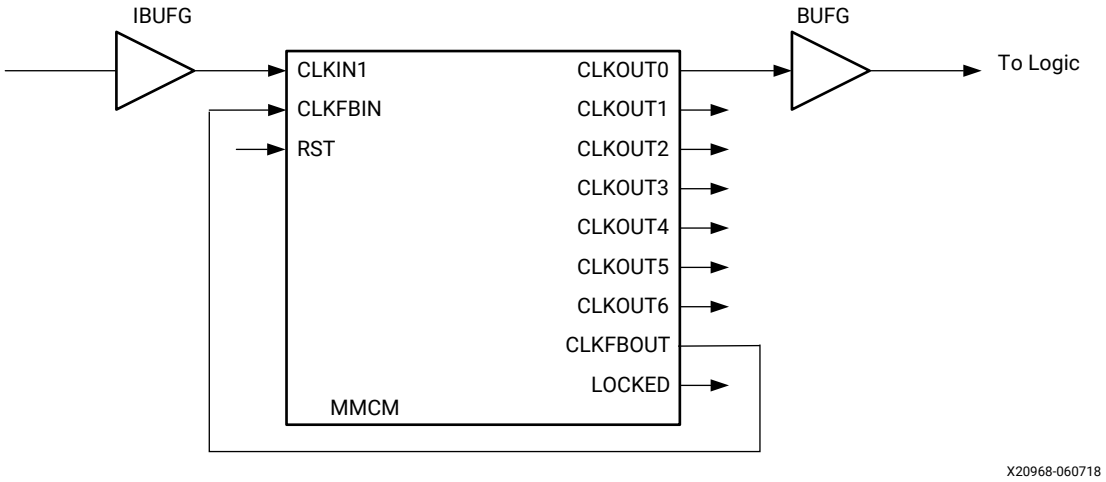
As an example, if F_{in} is 166 MHz, D = 1, M = 15, and O = 20, then VCO is 2490 MHz and the clock output frequency is 124.5 MHz. Because the M value in the feedback path is 15, both input frequencies at the PFD are 166 MHz.

Another more complex scenario has an input frequency of 66.66 MHz, D = 2, M = 100, and O = 8. The VCO frequency in this case is 3333 MHz and the CLKOUT output frequency is 416.625 MHz. Therefore, the feedback frequency at the PFD is 3333/100 or 33.33 MHz, matching the 66.66 MHz/2 input clock frequency at the PFD.

MMCM with Internal Feedback - No Deskew

The MMCM feedback can be internal to the MMCM when the MMCM is used as a synthesizer or jitter filter and there is no required phase relationship between the MMCM input clock and the MMCM output clock. The feedback clock is not subjected to noise on the core supply because it never passes through a block powered by this supply. Thus the MMCM performance increases. However, noise introduced on the CLKIN signal and the BUFG are still present (see the following figure).

Figure 37: **MMCM with Internal Feedback - No Deskew**

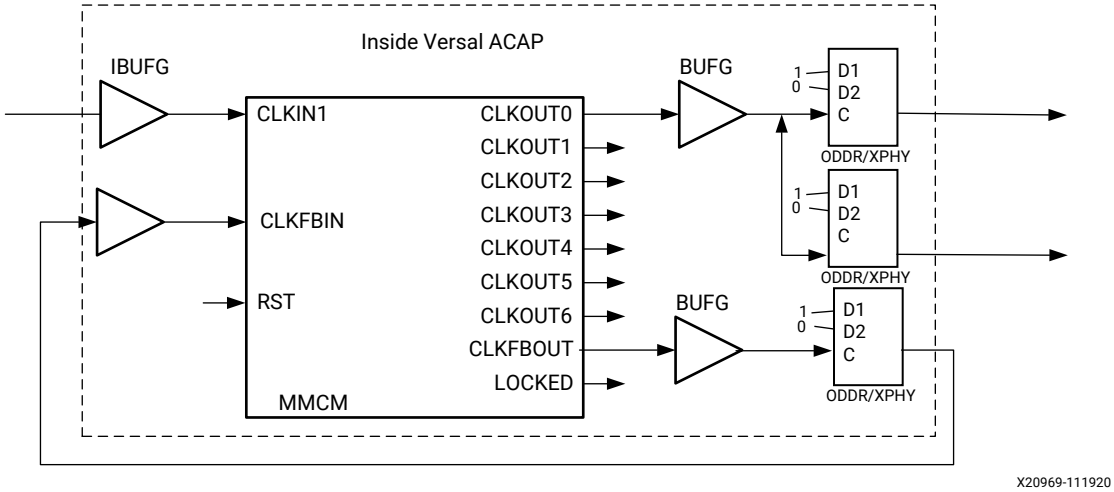


Zero Delay Buffer - CLKFBOUT Deskew

The MMCM 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 fanout to multiple destinations with a low skew between them. This configuration is shown in the following figure.

In this case, the feedback signal drives off chip and the board trace feedback is designed to match the trace to the external components. It is assumed in this configuration that the clock edges are aligned at the input of the Versal ACAP and the input of the external component. The input clock buffers for CLKIN and CLKFBIN must be in the same bank.

Figure 38: **Zero Delay Buffer - CLKFBOUT Deskew**



In some cases, precise alignment cannot occur because of the difference in loading between the input capacitance of the external component and the feedback path capacitance of the Versal device. For example, the external components can have an input capacitance of 1 pF to 4 pF while the part has an input capacitance as specified in the Versal ACAP data sheets. There is a difference in the signal slope, which is basically skew. Designers should be aware of this effect to help ensure timing.

Deskew Logic - Additional Deskew Options

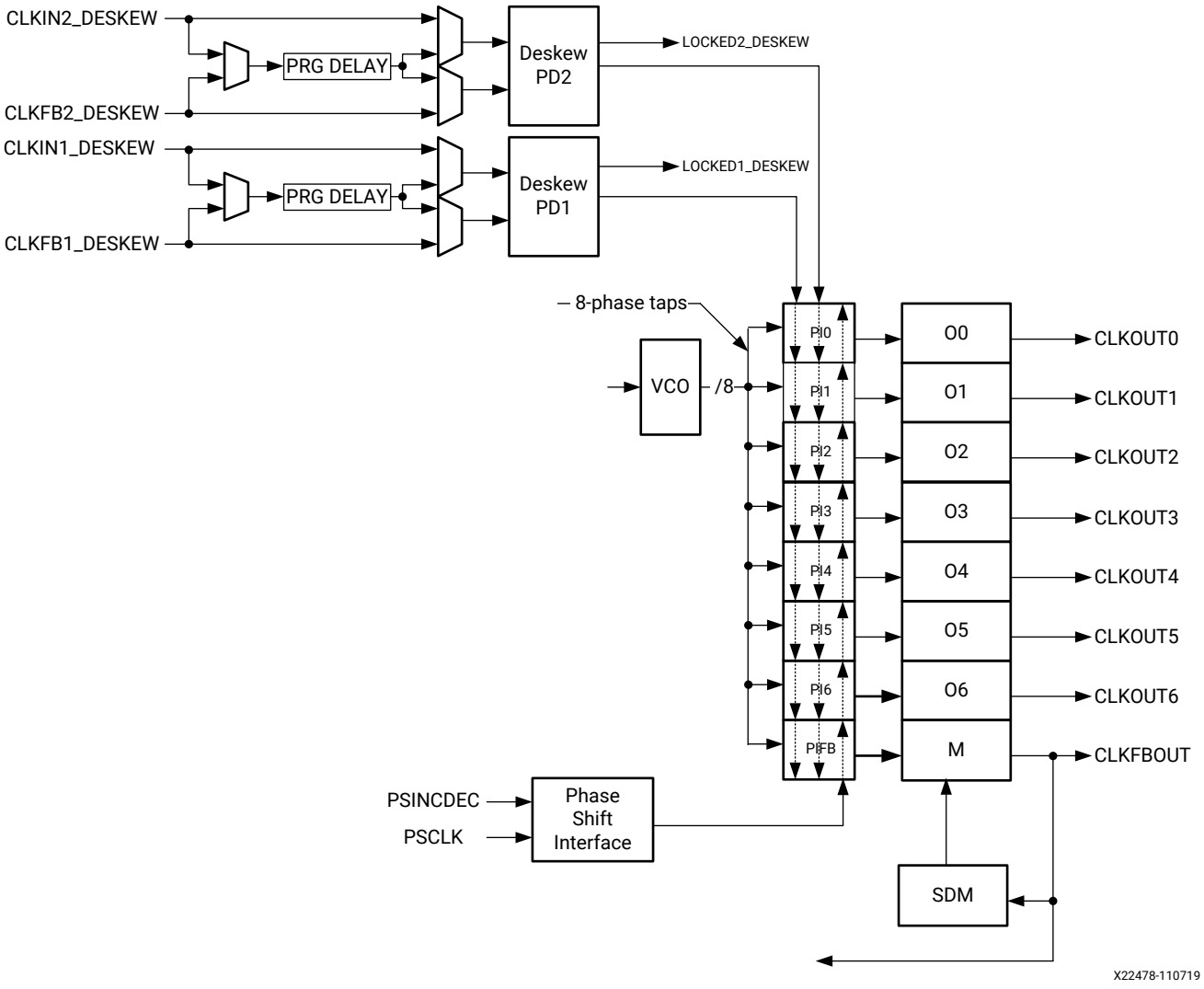
Versal ACAP is equipped with functionality to allow the deskewing of the MMCM, DPLL, and/or XPLL output clocks.

- These deskew options *do not* replace the classic feedback of MMCM, DPLL, and/or XPLL.

  Normal MMCM and PLL feedback uses an M counter with clock feedback output (CLKFBOUT) that needs to be routed/connected to the clock feedback input (CLKFBIN). This feedback route can be direct, pass through a clock buffer, or is internal or external.

- The deskew inputs and logic (Phase Detector (PD) and Phase Interpolator (PI)) help align and deskew an output clock network (CLKOUT*), and to provide an input clock other than the regular input clock (CLKIN). The deskew functionality is aimed at eliminating the skew between two clock networks of the same frequency. See Functioning of Deskew for additional details.
- The deskew function operates by using a PD on two clocks inputs and a set of PIs. There is one PI for each output counter and therefore a clock output as in the figure below. Each PI can be controlled by a PD.
- The PD looks to align the rising edge of the both input clocks (CLKIN_DESKEW and CLKFB_DESKEW). The output of the PD controls the selected PIs to output one of 32 VCO clock phases to the required output counter. A configurable delay line with selectable input (CLKIN_DESKEW or CLKFB_DESKEW) can be inserted in one of the two inputs of the PD. The delay line has a fixed input delay and consists of 64 taps of 40 ps (PVT variable) each.
- The PI converts the eight clock phases coming from the VCO to 32 phases feeding the output counters O and M under control of a PD.
  - The PI in front of an output counter selects one of 32 phases as the clock for the output.
  - Eight VCO phase equals to 45° for each phase, 32 phases equals to 11.25° per phase. Therefore, it is absolutely necessary to have the VCO run as fast as possible to keep the phase jumps of 11.25° as small as possible for lower speed output clocks.
- The fine phase shift option of an MMCM also uses the PI to do its job.

Figure 39: **The Deskew Logic**



Use of the Deskew Logic

All involved settings can be set by using attributes to the MMCM and/or PLL: *CLKOUTn_PHASE_CTRL[0:1]*

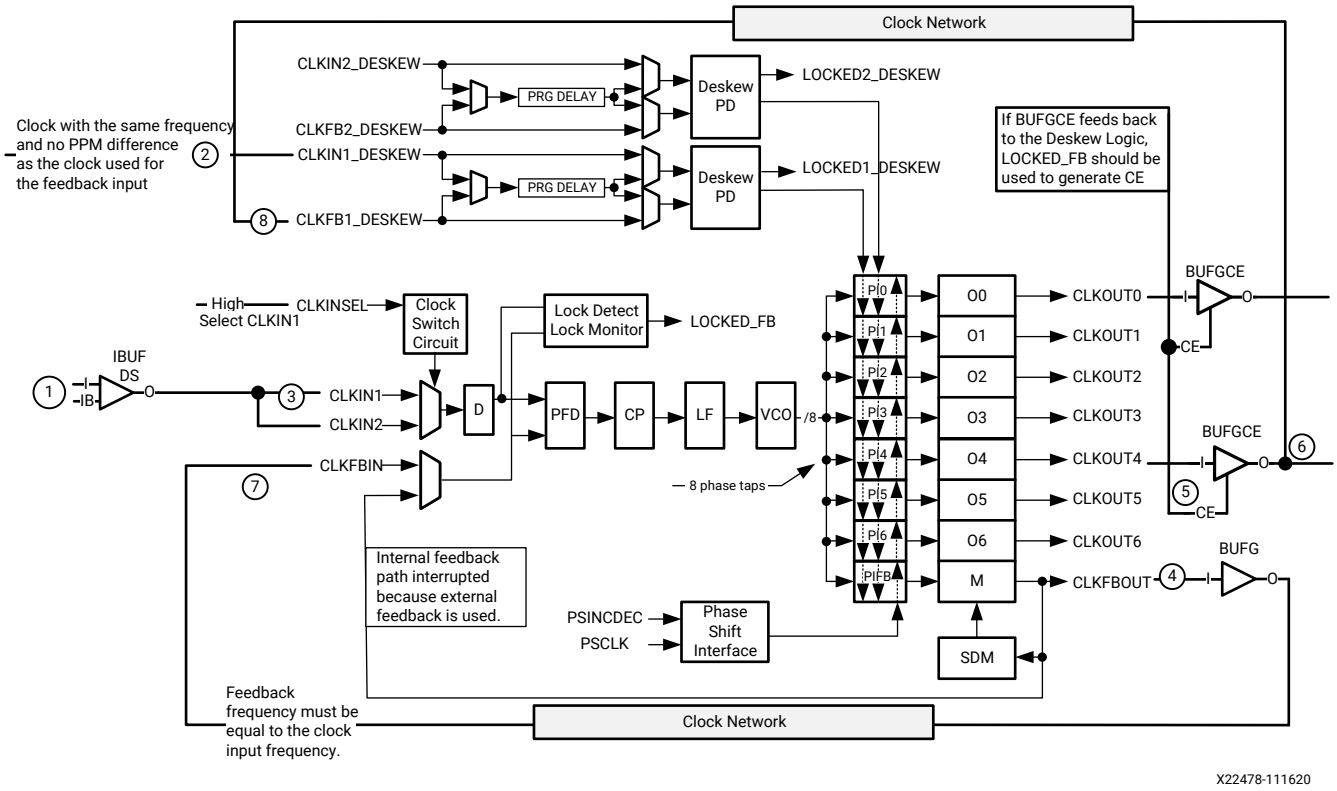
Table 13: **Deskew Logic Settings**

Value	Description
00	No interpolator control
01	Interpolator controlled by deskew 1
10	Interrpolator controlled by fine phase shift
11	Interpolator controlled by deskew 2

- ***DESKEW_DELAY1*:** Optional value to program the delay in the deskew circuit (0 to 63).
- ***DESKEW_DELAY2*:** Optional value to program the delay in the deskew circuit (0 to 63).
- ***DESKEW_DELAY_PATH1*:** Set what input, CLKIN1_DESLEW or CLKFB1_DESKEW passes through the delay line.
- ***DESKEW_DELAY_PATH2*:** Set what input, CLKIN2_DESLEW or CLKFB2_DESKEW passes through the delay line.
- ***DESKEW_DELAY_EN1*:** Enable the optional programmable delay.
- ***DESKEW_DELAY_EN2*:** Enable the optional programmable delay.

Functioning of Deskew

Figure 40: **Example of an MMCM Using DESKEW for a Single Clock Output**

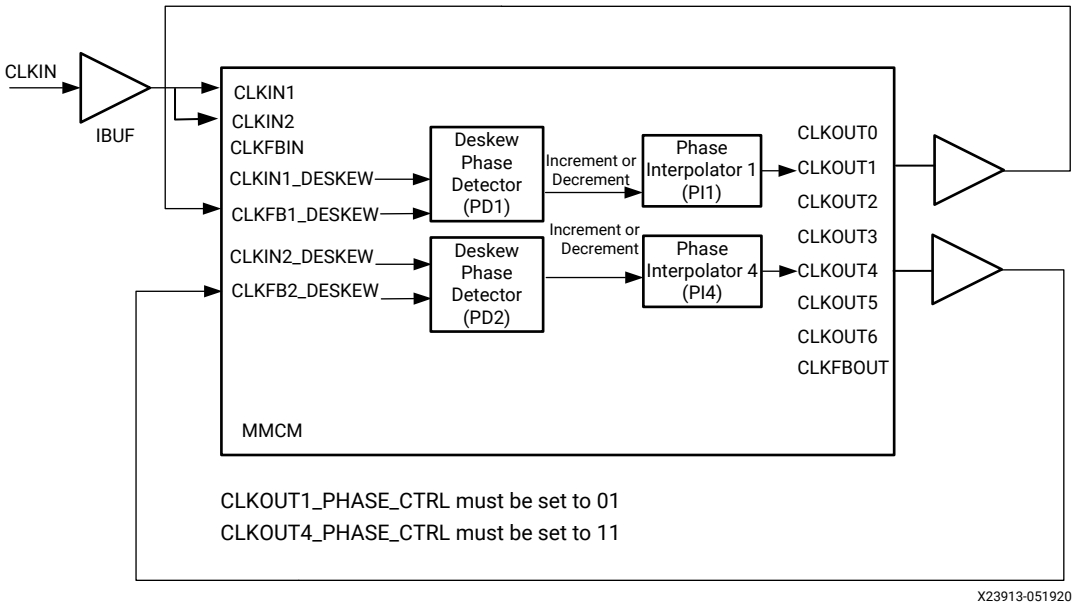


1. Apply a clock of any frequency, obeying the min/max values provided in the Versal ACAP data sheets, to one of the clock inputs on an MMCM (3). The single clock is connected to both clock inputs and the CLKINSEL input being high determines that CLKIN1 is going to be used.

2. Apply clock with the same frequency as CLKFB*_DESKEW, the input clock used for the feedback from CLKOUT, to the CLKIN*_DESKEW input (2). If both clocks are coming from CLKOUTs of the same PLL and have the same frequency, they are guaranteed to have no PPM difference since both CLKOUTs are derived from the same VCO.
3. Route the required feedback network between CLKFBOUT (4) and CLKFBIN (7).
4. M and D values are calculated to make the VCO run at the maximal possible frequency. The M value ensures the feedback counter runs at the same frequency as the MMCM input clock (CLKIN1), and that the following equation is fulfilled: *Fin/D = Ffb = Fvco/M*. One of the 32, 1-of-8 to after PI 1-of-32, output phases of the VCO is chosen as input clock for the output counter creating in this setup CLKOUT4. That counter divides the VCO clock down to the by CLKOUT4 required clock frequency.
5. CLKOUT4 of the MMCM must be fed to a clock buffer (BUFG or its variant) to make sure the clock is distributed over the dedicated clock routing in the programmable logic. In this setup, the clock output (6), after passing through the clock buffer, is also routed back to a CLKFB_DESKEW input (8). One must make sure that the frequency calculated and generated by CLKOUT4 equals the frequency of the clock applied to the CLKINx_DESKEW pin as stated in point 2. The duty cycles of the two clocks are not required to be the same provided they are within specifications.
6. In the normal operation mode of the MMCM, the phase frequency detector (PFD) block will make sure that the input clock (3) will be matched to the feedback clock (7) to make sure that the output clocks are deskewed and in phase with the MMCM input clock.
7. In this example, the DESKEW logic is used and that is going to take precedence over the normal phase alignment. In this case, CLKOUT4 (6) is going to be aligned to the clock applied to CLKIN_DESKEW (2).
    a. This PD of the DESKEW circuit somewhat functions as the regular PFD circuit. Except that it does not alter the VCO frequency and selects with the phase interpolator (PI) one clock out of 32 and uses that as clock input for an output counter.
    b. The CLKIN_DESKEW clock and CLKFB_DESKEW clock, both of the same frequency, are rising edge phase compared to each other. When the phase between both clocks is not equal a signal will influence a, during setup of the MMCM, selected PI. The PI block will select 1-of-32 clock phases of the VCO output (per 11.25 degrees) as input for the attached output counter. This way the clock output will move in one direction or another to phase align the MMCM output clock with the extra foreign clock. The comparison will be done again and, when necessary, the deskew PD will act again on the selected PI. All this operates dynamically and ensures that the output clock, in this case CLKOUT4, will stay phase aligned to the external applied clock.
    c. CLKOUT4 will *not* be phase-aligned to other output clocks of the MMCM and will *not* be in phase with the clock input of the MMCM, but will be phase-aligned to the clock applied to the CLKIN_DESKEW input.

8. In MMCM and XPLL, CLKINx_DESKEW must be of the same clock frequency as the CLKOUTx connected to CLKFBx_DESKEW with no PPM (parts-per-million) difference. To ensure the condition, the CLKINx and CLKINx_DESKEW inputs must be referenced to the same source, such as a crystal. In DPLL, regardless of whether it is in ZHOLD mode, CLKINx_DESKEW must be connected to CLKIN input and not from any CLKOUT feedback.
9. In general, CLKFB_DESKEW is fed back from any CLKOUT through a BUFG or its variant. In DPLL, the ZHOLD mode applies to all CLKOUTs. Refer to the DPLL Attributes section for a description of the ZHOLD attribute.
10. To enable deskew using PDs, the CLKOUT which is routed back to one of the CLKFB_DESKEWs must have the attribute CLKOUTn_PHASE_CTRL set to 01 or 11. For MMCM in deskew using PDs, the attribute must be set to INTERNAL. If it is set to the default value AUTO, you must ensure that CLKFBOUT is not connected to CLKFBIN. The following figure illustrates the connections between CLKOUT and CLKFB_DESKEW and settings for CLKOUTn_PHASE_CTRL.

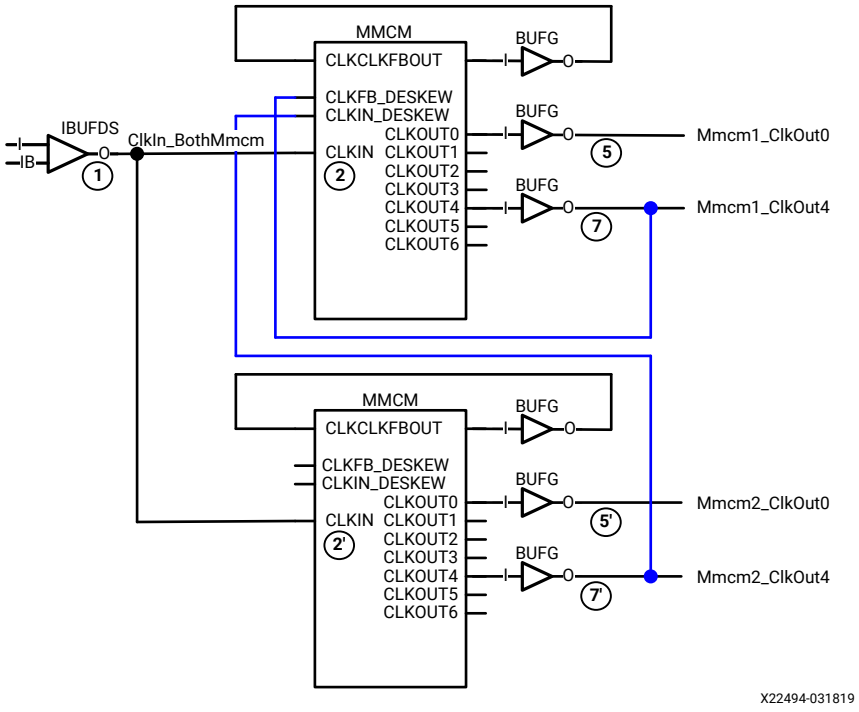
Figure 41: **MMCM Deskew using PDs Connections and CLKOUTn_PHASE_CTRL Settings**



Use Cases for Deskew

Phase Align Selected Clock Outputs of Two MMCMs

Figure 42: **Dual MMCM Using DESKEW**



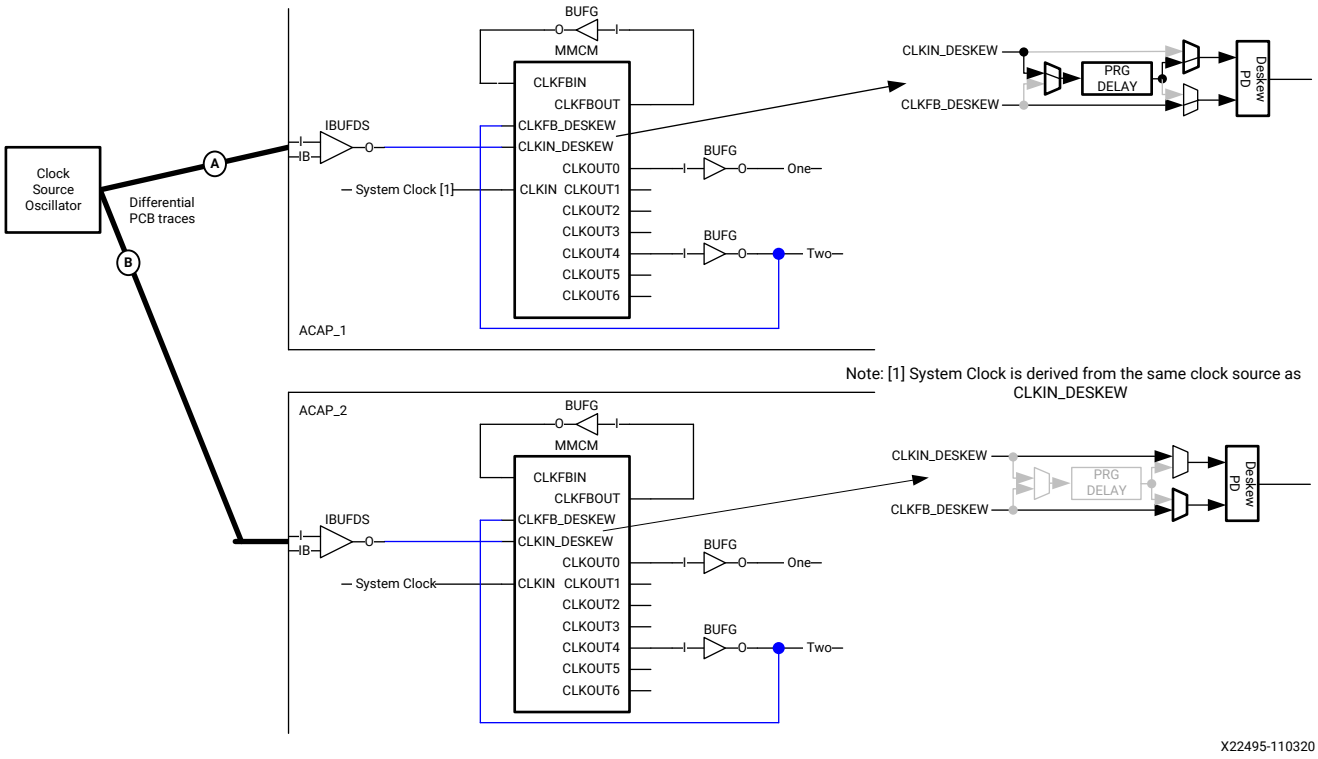
- Each MMCM fits in its own I/O bank / clock area. Both MMCMs get the same input clock (2) from a differential clock input (1).
- Without the DESKEW circuit the output clocks of each MMCM, 5, 7 and 5', 7', will be phase-aligned to the input clock of each MMCM (5, 7 are phase-aligned to 2 and 5', 7' are phase aligned to 2'). Because of the difference in delay routing between the output of the clock input buffer (IBUFDS), and the clock input of each MMCM, the output clocks of both MMCMs will not be phase-aligned to each other.
- When DESKEW unit(s) is/are used, it can help to phase align selected clock outputs of one MMCM to clock outputs of the other MMCM.
- Take one of the outputs of an MMCM and use it as CLKIN_DESKEW input on the second MMCM.
- Take an output clock of the MMCM where the DESKEW unit is used and route that to the CLKFB_DESKEW input.
- The output clock (CLKOUT4 (7) of the MMCM will be phase-aligned to the output clock (CLKOUT4 - 7') of the other MMCM.

Phase Align Selected Clocks in Different ACAPs

When multiple ACAPs on a PCB have to be fed with the same clock, and the design on the board needs to run synchronously, it is common practice to pass the clock source through a multi-clock buffer on the board, and adjust the length of the clock routes/lanes between the clock buffer device, and the different Versal ACAPs on the PCB.

Using a DESKEW unit in an MMCM or PLL can achieve the same result without external multi-zero delay clock buffer and careful timing routed PCB tracks. The following figure shows a two-Versal ACAP example and how it can be achieved.

Figure 43: **Deskew of External Versal ACAP Clocks - System Synchronous Design**



The setup demonstrated here is between two ACAPs.

- The *System Clock* in both ACAPs can be any master clock necessary to generate clocks for logic, hard-IP and/or soft-IP.
- In both devices, CLKOUT4 (Clock Two) is used as clock that needs to be phase aligned between the two ACAPs.
- The PCB traces from Clock Source to ACAP_1 (A) is shorter than the clock traces between the Clock Source and ACAP_2 (B).
- The key requirement is that CLKIN and CLKIN_DESKEW in each ACAP have to have a common reference point, which in this case is the clock source oscillator.

- How does it work?
  - The differential clock input buffers (IBUFDS) are handled as zero delay buffers.
  - The *programmable delay* is handled as if it has no initial delay.
  - In both ACAPs, the setup around the MMCM is identical. CLKOUT4 is used as feedback clock for the DESKEW unit while the clock input of the unit is connected to a clock input. This setup will phase align the output of the clock buffer connected to CLKOUT4 to the CLKIN_DESKEW of the MMCM.
  - The DESKEW unit setup in ACAP_2 is straightforward. Feed the input clock and feedback clock to the DESKEW unit and let it phase align clock two to the input clock.
  - The DESKEW unit setup in ACAP_1 uses the programmable delay to compensate for the longer clock route to ACAP_2. The programmable delay is engaged in the clock input path and is set to a value of: $PRGDLY = Trace_B - Trace_A$.
  - Using the delay line in the DESKEW unit of the MMCM in ACAP_1 allows the clock traces between the clock source and both ACAPs are of equal length. As such, this ensures that both clocks are phase-aligned.

Fractional Divide Frequency Synthesis

The fractional divide in UltraScale™ and UltraScale+™ devices (where the resolution is 0.125) has been replaced in the Versal ACAP by a fractional sigma-delta module (SDM) in the feedback path. The additional 6-bit resolution divider allows generation of finer granularity output clocks than was possible with UltraScale devices. The best jitter performance of an MMCM is obtained when using it in integer mode. When the fractional part of M+F is set to 0, integer mode is automatically chosen. When running the MMCM in fractional mode make sure the VCO runs at the maximum possible frequency for the application. Following examples are provided to highlight the mechanism while the Clock Wizard will automatically deal with this when fractional output frequencies are required. Use attributes for fractional divide:

```
CLKFBOUT_FRACT : integer := 0;
```

```
CLKFBOUT_MULT : integer := 42;
```

To put the focus on the fractional divide, the steps to calculate M, D, and other values are omitted from the following examples:

Example 1

Assume F_{CLKOUT} must be 296.703 MHz for an F_{CLKIN} of 27 MHz.

D = 10 and M = 109

$$F = 57,\ \frac{F}{2^6} = 0.890625$$

$$M + \left(\frac{F}{2^6}\right) = 109.890625$$

F_{VCO} = 2967.046875 MHz from:

$$F_{VCO} = F_{CLKIN} \times \frac{M}{D}$$

F_{CLKOUT} is then 296.7046875 MHz (rounded to 296.705 MHz)

The previous calculation does not use the highest possible VCO frequency to generate the output clock frequency of 296.703 MHz. Recalculate for the highest possible VCO frequency.

D = 14 and M = 153

$$F = 54, \ \frac{F}{2^6} = 0.84375$$

$$M + \left(\frac{F}{2^6}\right) = 153.84375$$

F_{VCO}= 4153.78125 MHz

F_{CLKOUT} is then 296.69866 MHz (rounded to 296.699)

The first calculation can achieve 296.7046875 MHz and the second can achieve 296.69866 MHz. Both settings are slightly off from the target frequency of 296.703 MHz, which is due to the resolution of the fractional mode. To improve the frequency error, higher reference frequencies must be used.

Example 2

Assume F_{CLKOUT} must be 335 MHz for an F_{CLKIN} of 30 MHz

D = 1, M = 100, and O = 9

$$F = 32, \ \frac{F}{2^6} = 0.5$$

$$M + \left(\frac{F}{2^6}\right) = 32.5$$

F_{VCO}= 3015 MHz from:

$$F_{VCO} = F_{CLKIN} \times \frac{M}{D}$$

F_{CLKOUT} is then 335 MHz

Example 3

$$F_{VCO} = F_{CLKIN} \times \frac{M}{D}$$

$$F_{CLKOUT} = F_{CLKIN} \times \frac{M}{D \times O}$$

Desired F_{CLKOUT} is 212.3457 MHz for an F_{CLKIN} of 50 MHz

F_{VCOmax} = 4320 MHz
The highest possible VCO frequency is obtained for a O value of:

$$rounddown\ \frac{F_{VCOMAX}}{F_{CLKOUT}}$$

$_{rounddown}$(3420/212.3457) = 20

$$F_{VCO} = 20 \times F_{CLKOUT}\ \textbf{\textit{or}}\ F_{VCO} = 4246.9140$$

$$M = M + \frac{F}{2^6} = \frac{F_{VCO}}{F_{CLKIN}} = 84.93828$$

Assume D = 1

$$M = 84\ \textbf{\textit{and}}\ \frac{F}{2^6} = n = 0.93828$$

$$F_{IDEAL} = n \times 2^6 = 0.93828 \times 2^6 = 60.0499$$

F must be an integer value, therefore take the $_{rounddown}F_{IDEAL}$ or F = 60

Redo the calculations to get the real F_{CLKOUT}:

$$F_{VCO} = 50\ MHz \times \left[\frac{84 + \frac{60}{2^6}}{1}\right] = 4246.8750\ MHz$$

$$F_{CLKOUT} = 50\ MHz \times \left[\frac{84 + \frac{60}{2^6}}{1 \times 20}\right] = 212.3437\ MHz$$

212.3457 MHz was required.

MMCM/DPLL to MMCM/DPLL Connection

The MMCM and DPLLs can be cascaded through clock buffers and the routing resources only. There is no compensation for routing delays.

Spread-Spectrum Clock Generation

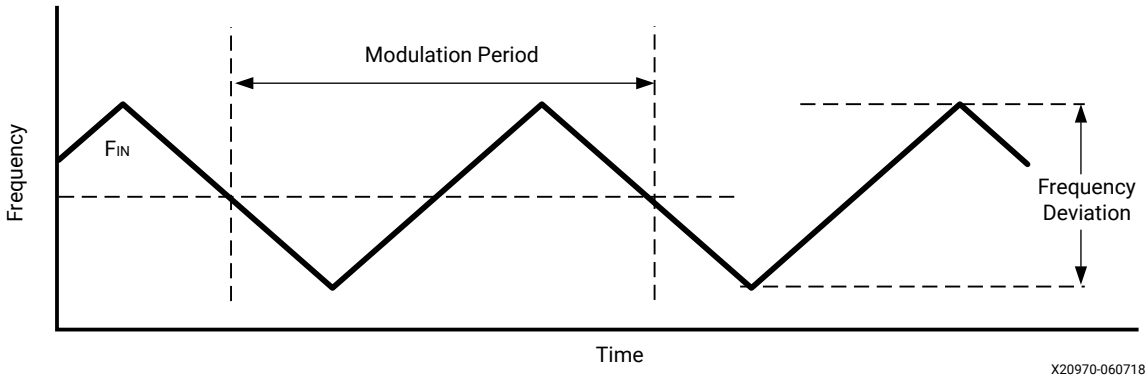
Spread-spectrum clock generation (SSCG) is widely used by manufacturers of electronic devices to reduce the spectral density of the electromagnetic interference (EMI) generated by these devices. Manufacturers must ensure that levels of electromagnetic energy emitted do not interfere with the operation 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.

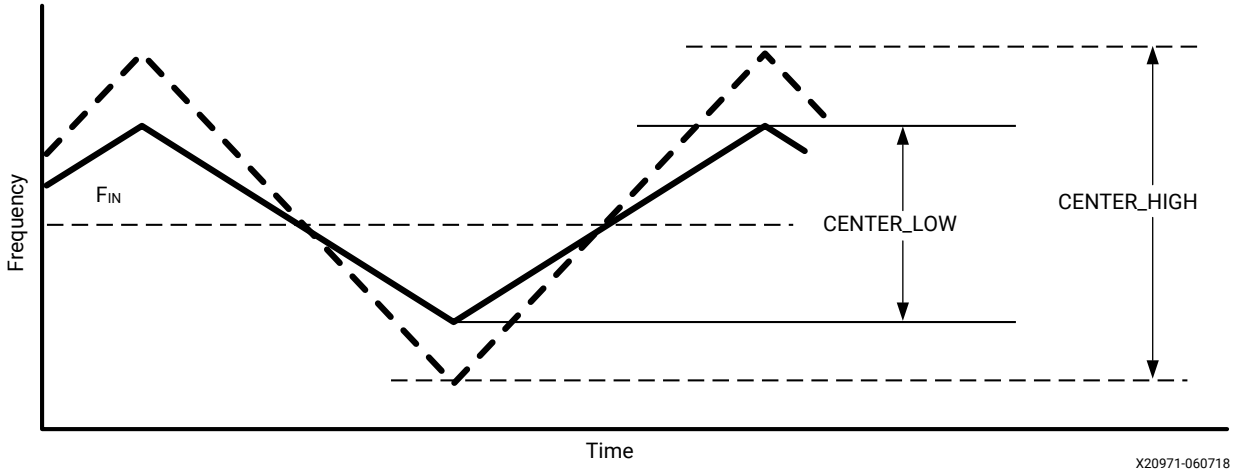
The MMCME5 can generate a spread-spectrum clock from a standard fixed frequency oscillator when SS_EN is set to TRUE (see the following figure). Within the MMCME5, the VCO frequency is modulated along with CLKFBOUT and CLKOUT[6:4,1,0]. Clock outputs CLKOUT[3:2] are used to control the modulation period and are not available for general use. As long as the clock frequency is adjusted slowly, the spread spectrum does not affect the period jitter of the MMCME5. The MMCM offers 4 modes of spread: CENTER_HIGH, CENTER_LOW, DOWN_HIGH, and DOWN_LOW. HIGH and LOW indicate the amount of spread. CENTER spreads equally both above and below the nominal frequency. DOWN only spreads to lower frequencies. These modes are shown in the figures below.

Figure 44: **Center-Spread Modulation**



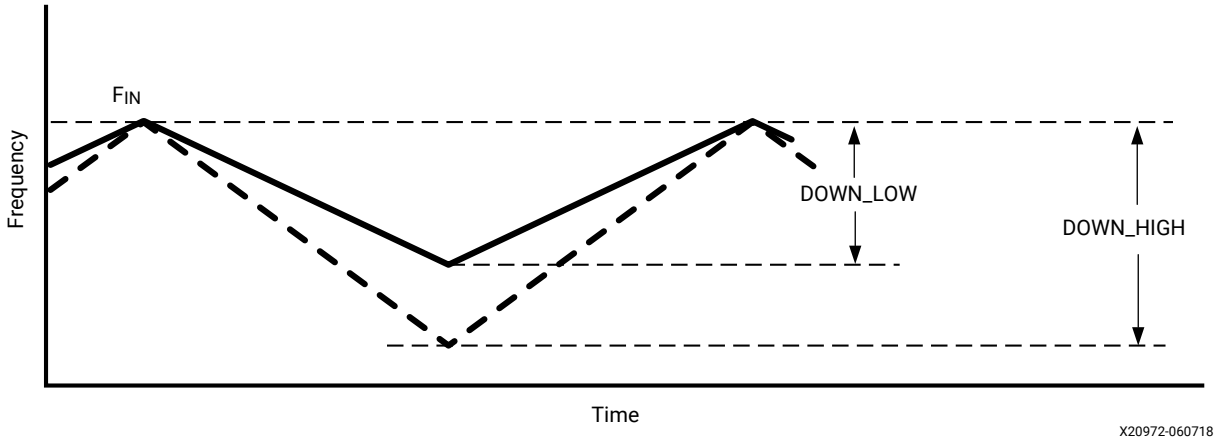
Adjusting the modulation period SS_MOD_PERIOD allows you to direct the tools to select the closest modulation period based on the MMCME5 settings. The spread-spectrum modulation reduces EMI as long as the modulation frequencies are higher than the audible frequency range of 30 KHz. Typically, lower modulation frequencies are preferred to minimize the impact of the introduction of spread spectrum. Increasing the frequency deviation with SS_MODE (CENTER_HIGH or DOWN_HIGH) increases the overall EMI reduction, but care must be taken to ensure that the increased range of frequencies does not affect the overall system operation (see the following figure).

Figure 45: **Center-Spread Modulation (CENTER_LOW vs. CENTER_HIGH)**



Another design trade-off is the decision to use a center spread or down spread. Selecting SS_MODE (DOWN_HIGH, DOWN_LOW) spreads the frequencies to lower frequencies as shown in the following figure. DOWN_HIGH has similar frequency deviation to CENTER_LOW.

Figure 46: **Down-Spread Modulation**



When using center spread-spectrum clocking (CENTER_HIGH, CENTER_LOW), the average output frequency is approximately equal to the nominal frequency. When using down spread-spectrum clocking, the average output frequency is lower than the nominal frequency. This means that for all cases of spread-spectrum clocking the spread spectrum clocks are different for long periods of time (modulation frequency). As a result, it is necessary to apply appropriate clock domain crossing (CDC) measures on all data and non-data signals crossing between crossing clock and spread-spectrum clock domains and vice versa. Asynchronous FIFOs should be used to transfer data between two clock domains. The depth of the FIFO depends on the modulation frequency in the clock. The slower the modulation, the deeper the FIFO needs to be:

$$FIFO\ depth\ is\ proportional\ to\ \frac{frequencyDeviation}{ModulationFrequency}$$

When down spread-spectrum clocking (DOWN_LOW or DOWN_HIGH) is used, the output frequency is lower than the original clock frequency, which means that if no precautions are taken the used FIFO can fill up and over-run. To prevent this, a FIFO with throttle control must be used. Designs using spread-spectrum clocking need to take in account that the implementation of the design needs to meet timing for the highest frequency in the frequency deviation. For example:

- A design with an input clock of 100 MHz using spread-spectrum clocking set as CENTER_LOW creates a clock spread of ±1.15%. This means that the design must meet timing when the clock reaches 100 MHz + 1.15%, or 101.15 MHz.
- A design with the same 100 MHz clock input but set in spread-spectrum mode DOWN_HIGH creates a clock spread of 2.44%. The spread-spectrum clock reaches then a maximum at 100 MHz – (–2.44%), or 102.44 MHz.

It is therefore expected that a design using spread-spectrum clocking requires that the timing constraints of the design are adjusted for the highest frequency in the clocks. And the Vivado software tools take the deviation caused by spread-spectrum clocking automatically into account when producing static timing analysis. For designs using spread-spectrum clocking, the formula leveraged by the static analysis tools is as follows:

$$\frac{\left(\left(TSj^2 - Dj^2\right)^{\frac{1}{2}}\right)}{2} + PE + SS$$

where:

- *TSj* = Total system jitter
- *Dj* = Discrete jitter
- *PE* = Phase error
- *SS* = Spread spectrum

The SS parameter is calculated by the Vivado tools from the information provided by the design input, such as attributes or clock wizard settings.

Logic within the MMCM controls the spread-spectrum modulation based on a given input frequency and SS_MOD_PERIOD. The modulation frequency must be between 25 and 250 kHz and the input frequency must be between 25 and 150 MHz. Only LOW BANDWIDTH is supported. Additionally, only specific M and D settings are allowed for particular input frequencies. These restrictions, the spread and allowable frequencies are listed in the following table.

***Note*:** In actual devices, the center-spread modulation waveform may deviate from an ideal triangular shape. However, there is no impact on nominal frequency.

Spread-Spectrum Frequencies and Spread

Table 14: **Spread-Spectrum Frequencies and Spread**

SS_MODE	Allowed Frequencies	CLKFBOUT_MULT_F	DIVCLK_DIVIDE	Spread
CENTER_HIGH	100 MHz < F_{in} < 135 MHz	28	1	± 1.82%
	135 MHz < F_{in} <185 MHz	21	1	± 2.44%
		22	1	± 2.32%
	185 MHz < F_{in} < 275 MHz	28	2	± 1.82%
	275 MHz < F_{in} < 350 MHz	21	2	± 2.44%
		22	2	± 2.32%
	350 MHz < F_{in} < 550 MHz	21	3	± 2.44%
		22	3	± 2.32%
CENTER_LOW	100 MHz < F_{in} < 135 MHz	56	2	± 0.90%
	135 MHz < F_{in} < 185 MHz	42	2	± 1.20%
		44	2	± 1.15%
	185 MHz < F_{in} < 275 MHz	56	4	± 0.90%
	275 MHz < F_{in} < 350 MHz	42	4	± 1.20%
		44	4	± 1.15%
	350 MHz < F_{in} < 550 MHz	42	6	± 1.20%
		44	6	± 1.15%
DOWN_HIGH	100 MHz < F_{in} < 135 MHz	28	1	– 1.82%
	135 MHz < F_{in} <185 MHz	21	1	– 2.44%
		22	1	– 2.32%
	185 MHz < F_{in} < 275 MHz	28	2	– 1.82%
	275 MHz < F_{in} < 350 MHz	21	2	– 2.44%
		22	2	– 2.32%
	350 MHz < F_{in} < 550 MHz	21	3	– 2.44%
		22	3	– 2.32%
DOWN_LOW	100 MHz < F_{in} < 135 MHz	56	2	– 0.90%
	135 MHz < F_{in} < 185 MHz	42	2	– 1.20%
		44	2	– 1.15%
	185 MHz < F_{in} < 275 MHz	56	4	– 0.90%
	275 MHz < F_{in} < 350 MHz	42	4	– 1.20%
		44	4	– 1.15%
	350 MHz < F_{in} < 550 MHz	42	6	– 1.20%
		44	6	– 1.15%

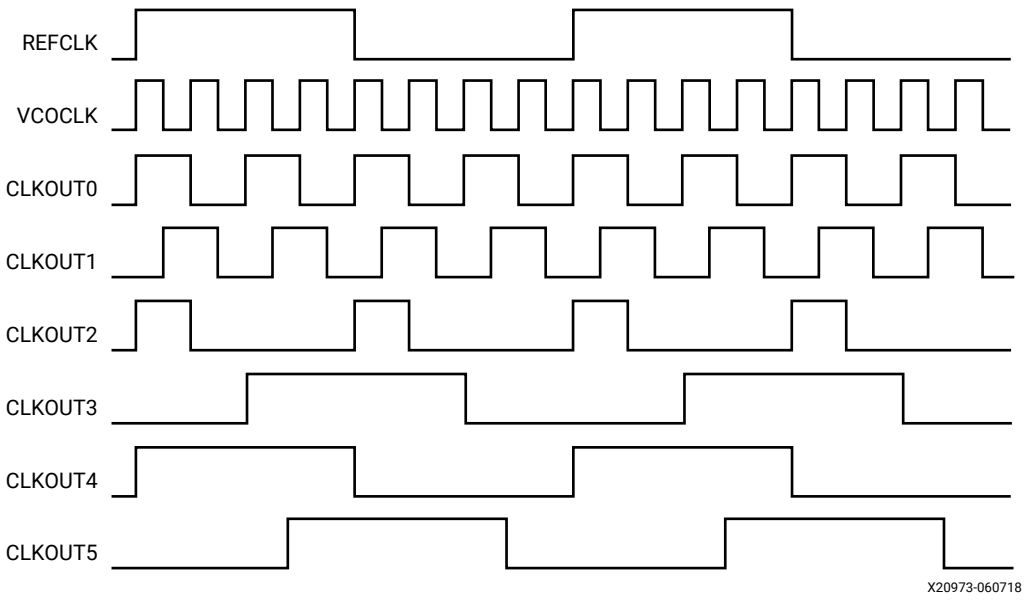
Application Example

These MMCM 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_F = 8;
DIVCLK_DIVIDE = 1;
CLKIN1_PERIOD = 10.0;
```

The following figure displays the resulting waveforms.

Figure 47: **Example Waveform**



DPLLs

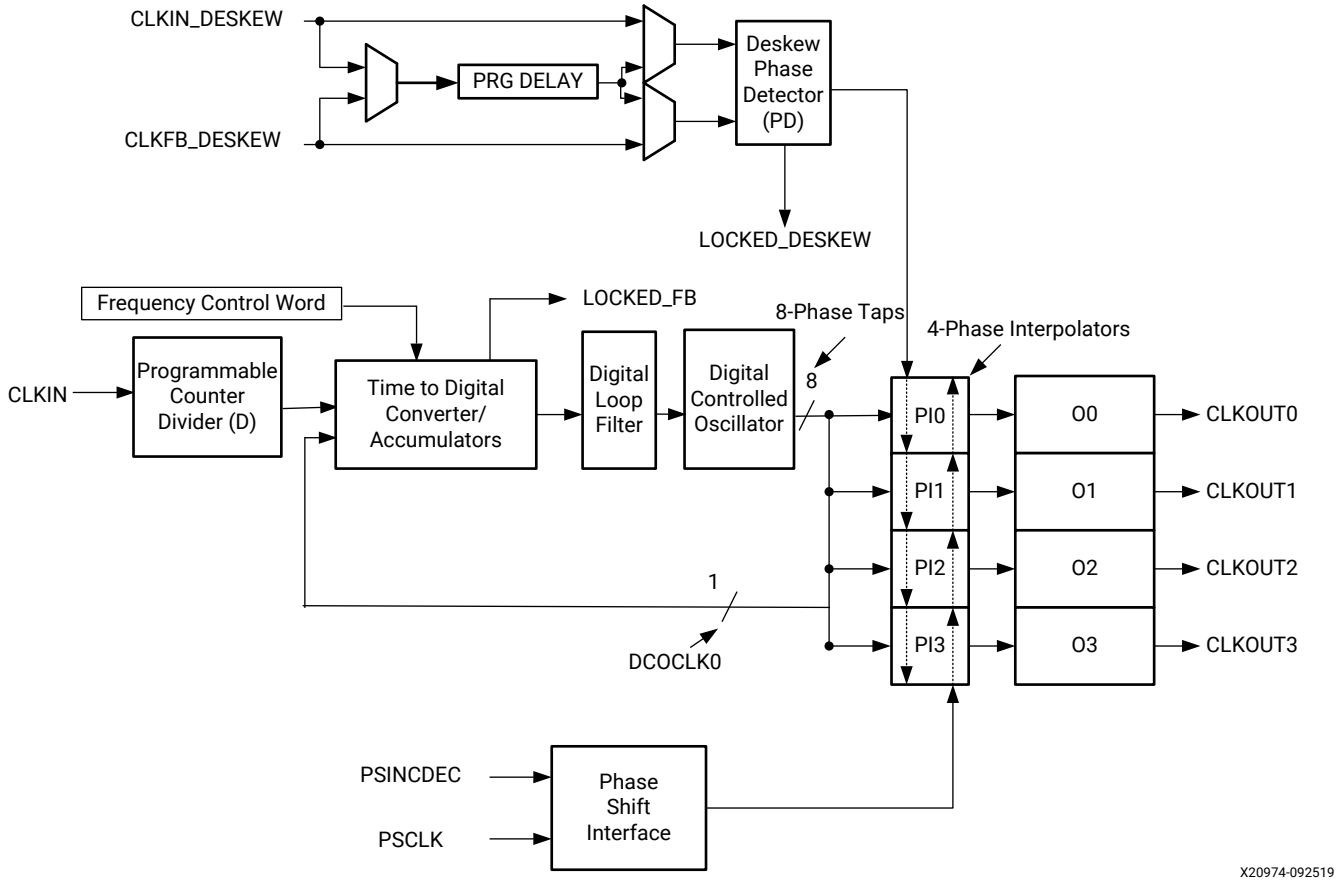
DPLL is an MMCM lite version of the phase locked loop (PLL) that is situated in the clocking column next to the HDIO and GT clocking column. Additional DPLLs are co-located in the same tile with MMCMs. DPLLs serve as a digital version of the MMCM for frequency synthesizers for a wide range of frequencies, and as jitter filters for either external or internal clocks, and deskew clocks. In many respects the functionality and capability of a DPLL are similar to that of MMCM and XPLL. Many of the DPLL functions can be found in the MMCMs section. Some of the differences are frequency specifications and DPLLs do not have fractional clock generation capabilities.

The fundamental, functional operation of the DPLL can be described as follows:

The time to digital converter (TDC) and accumulators measure the phase offset and frequency ratio of the input (reference) clock and the oscillator clock. They create a code that is fed to a digital loop filter (DLF). The output of the DLF determines the frequency of the digital controlled oscillator (DCO). It being a digital implementation loop filter, adjustments only occur every positive clock edge. The DCO produces eight output phases which feed the 4 phase interpolators (PIs). Each PI output generates the input clock for each of the 4 output counters. Each counter can be independently programmed for a given design. A frequency control word (FCW) providing the reference phase, is the division ratio command responsible for generating the multiply value for frequency synthesis. Therefore, unlike in the MMCM, there is no distinct feedback counter/divider.

Unlike the MMCM, the DPLL does not have a SDM module for fractional divide.

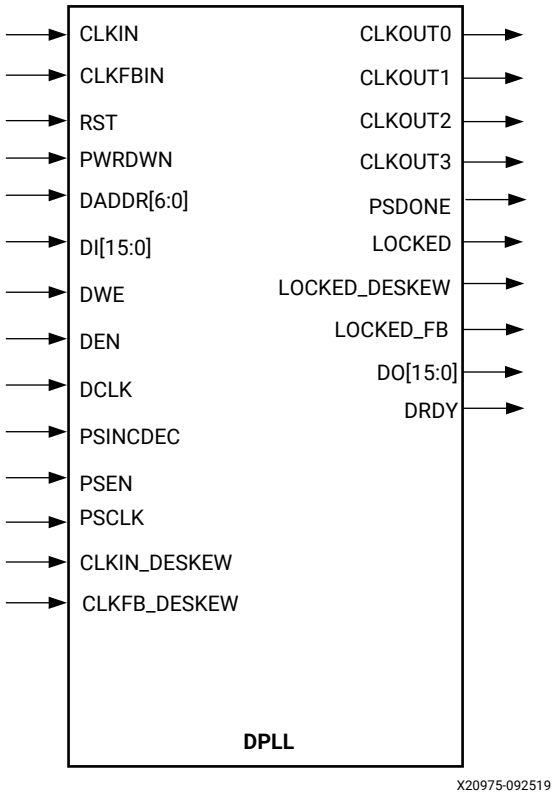
Figure 48: **DPLL Block Diagram**



DPLL Primitive

The Versal™ device DPLL instantiable primitive is shown in the following figure.

Figure 49: **DPLL Primitive**



The DPLL is a digital block designed to support clock network deskew, frequency synthesis, and jitter reduction. These three modes of operation are discussed in more detail in this section. The DCO operating frequency can be determined using the following relationship:

$$F_{dco} = F_{clkin} \times \frac{M}{D}$$

$$F_{clkout} = F_{clkin} \times \frac{M}{D \times O}$$

where the M, D, and O counters are shown in Figure 48. The value of M corresponds to the CLKFBOUT_MULT_F setting, the value of D to the DIVCLK_DIVIDE, and O to the CLKOUT_DIVIDE.

The four 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 because a single VCO drives all the counters.

Frequency Synthesis without Deskew

See Without Deskew.

Frequency Synthesis with CLKFBOUT Deskew

See With CLKFBOUT Deskew.

Clock Network Deskew Using the Deskew Phase Detector and Phase Interpolator

Similar to the MMCM, the DPLL can also deskew two CLKOUT networks chosen by the user. However, unlike the MMCM, only one pair of clocks can be deskewed using this method.

For the functional description, see the MMCMs section of this manual.

Frequency Synthesis Using Integer Divide

See Using Fractional Divide in the MMCM section of this manual.

Jitter Filter

DPLLs like all PLLs reduce the jitter inherent on a reference clock. The DPLL can be instantiated as a stand-alone function to only support filtering jitter from an external clock before it is driven into another block. As a jitter filter, it is usually assumed that the DPLLs act as a buffer and regenerate the input frequency on the output (for example, F_{IN} = 100 MHz, F_{OUT} = 100 MHz). The DPLL being a strictly digital implementation, does not have a BANDWIDTH setting and all adjustments are made on a clock cycle basis.

Limitations

DPLLs have some restrictions that must be adhered to. The major limitations are the DCO operation range, input frequency, and phase shift. In addition, there are connectivity limitations to other clocking elements (pins, GTs, and clock buffers). Duty cycle is not programmable and is always nominally 50/50. Cascading MMCMs can only occur through the clock routing network.

For Versal devices, VC1902, VC1802, and VM1802, the following restrictions related to DPLL apply.

- DPLL PD deskew function and ZHOLD mode are not supported in the listed devices. MMCMs have to be used instead.
- DPLLs in HDIO banks are not supported in the listed devices

For a list of Versal devices, refer to *Versal Architecture and Product Data Sheet: Overview* (DS950) and for switching characteristic specification, see *Versal Prime Series Data Sheet: DC and AC Switching Characteristics* (DS956) and *Versal AI Core Series Data Sheet: DC and AC Switching Characteristics* (DS957).

VCO Operating Range

The minimum and maximum VCO operating frequencies are defined in the electrical specification of the Versal ACAP data sheets. These values can also be extracted from the speed specification.

Minimum and Maximum Input Frequencies

The minimum and maximum CLKIN input frequencies are defined in the electrical specification of the Versal ACAP data sheets.

Duty Cycle Programmability

See the MMCM Duty Cycle Programmability section of this manual.

Phase Shift

See the MMCM Phase Shift sections of this manual.

Dynamic Phase Shift Interface in the DPLL

See the Dynamic Phase Shift Interface in the MMCM section of this manual.

Counter Control

See the MMCM Counter Control section.

Detailed DCO and Output Counter Waveforms

See the MMCM Detailed VCO and Output Counter Waveforms section of this manual.

DPLL Ports

The following table summarizes the DPLL ports.

Table 15: **DPLL Ports**

Port Name	I/O	Description
CLKIN1	Input	General clock input.
RST	Input	Asynchronous reset signal. The RST signal is an asynchronous reset for the DPLL. The DPLL synchronously re-enables itself when this signal is released (that is, DPLL re-enabled). A reset is required when the input clock conditions change (for example, frequency).
PWRDWN	Input	Powers down instantiated but unused DPLLs.

Table 15: **DPLL Ports** *(cont'd)*

Port Name	I/O	Description
CLKOUT[0:3]	Output	User configurable clock outputs (0 through 3) that can be divided versions of the VCO phase outputs (user controllable) from 2 to 512. The output clocks are phase aligned to each other (unless phase shifted) and aligned to the input clock with a proper feedback configuration.
CLKIN_DESKEW	Input	Primary clock input to the phase detector1 block for deskewing clock network delays between two different CLKOUT networks.
CLKFB_DESKEW	Input	Secondary (feedback) clock input to the phase detector1 block for deskewing clock network delays
LOCKED_FB	Output	An output from the DPLL that indicates when the DPLL has achieved phase alignment within a predefined window and frequency matching within a predefined PPM range. The DPLL automatically locks after power on. No extra reset is required. LOCKED is deasserted if the input clock stops or the phase alignment is violated (for example, input clock phase shift). The DPLL must be reset after LOCKED is deasserted.
LOCKED1/2_DESKEW	Output	Indicates if the deskew circuit is locked. Applies only to the deskew circuits used in the design. Ignore these outputs for unused deskew circuits.
LOCKED	Output	The LOCKED signal indicates that all functions requiring a LOCKED signal for the DPLL to operate properly have LOCKED. This LOCKED signal is therefore an AND function of LOCKED_FB and both LOCKED1/2_DESKEWs if used.
DO[15:0]	Output	The dynamic reconfiguration output bus provides DPLL data output when using dynamic reconfiguration.
DI[15:0]	Input	The dynamic reconfiguration data input (DI) bus provides reconfiguration data. When not used, all bits must be set to zero.
DADDR[6:0]	Input	The dynamic reconfiguration address (DADDR) input bus provides a reconfiguration address for the dynamic reconfiguration. When not used, all bits must be assigned zeros.
DRDY	Output	The dynamic reconfiguration ready output (DRDY) provides the response to the DEN signal for the DPLL's dynamic reconfiguration feature.
DWE	Input	The dynamic reconfiguration write enable (DWE) input pin provides the write enable control signal to write the DI data into the DADDR address. When not used, it must be tied Low.
DEN	Input	The dynamic reconfiguration enable (DEN) provides the enable control signal to access the dynamic reconfiguration feature. When the dynamic reconfiguration feature is not used, DEN must be tied Low.
DCLK	Input	The DCLK signal is the reference clock for the dynamic reconfiguration port.
PSCLK	Input	Phase shift clock.
PSEN	Input	Phase shift enable.
PSINCDEC	Input	Phase shift increment/decrement control.
PSDONE	Output	Phase shift done.

Notes:

1. All control and status signals except PSINCDEC are active-High.

TIP: *The port names generated by the clocking wizard can differ from the port names used on the primitive.*

For dynamic reconfiguration port description, see Dynamic Reconfiguration Port (DRP). For all other port descriptions, refer to the MMCM Port Descriptions.

DPLL Attributes

The following table lists the attributes for the DPLL primitive.

Table 16: **DPLL Attributes**

Attribute	Type	Allowed Values	Default	Description
CLKOUT[0:3]_DIVIDE	Decimal	2 to 511	2	Specifies the amount to divide the associated CLKOUT clock output if a different frequency is desired. This number, in combination with the CLKFBOUT_MULT_F and DIVCLK_DIVIDE values, determines the output frequency.
CLKOUT[0:3]_PHASE	Real	–360.000 to 360.000	0.000	Specifies the output phase relationship of the associated CLKOUT clock output in number of degrees offset (that is, 90 indicates a 90° or ¼ cycle offset phase offset while 180 indicates a 180° offset or ½ cycle phase offset).
CLKOUT[0:3]_DUTY_CYCLE	Real	0.001 to 0.999	0.500	Specifies the duty cycle of the associated CLKOUT clock output as a percentage (that is, 0.50 generates a 50% duty cycle).
CLKFBOUT_MULT	Decimal	10 to 400	42	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, determines the output frequency.
CLKFBOUT_FRACT	Decimal	0 to 63	0	6-bit fractional M feedback divider in increments of 1/63. Generates a fraction of the CLKFBOUT_MULT value and adds it to CLKFBOUT_MULT.
DIVCLK_DIVIDE	Decimal	1 to 123	1	Specifies the division ratio for all output clocks with respect to the input clock. Effectively divides the CLKIN going into the PFD.
REF_JITTER	Real	0.000 to 0.200	0.010	Allows specification of the expected jitter on the reference clock to better optimize DPLL performance. If known, the value provided should be specified in terms of the unit interval (UI) (the maximum peak-to-peak value) of the expected jitter on the input clock.
CLKIN_PERIOD	Real	0.000 to 100.000	0.000	Specifies the input period in ns to the DPLL CLKIN1 input. Resolution is down to the ps. This information is mandatory and must be supplied.

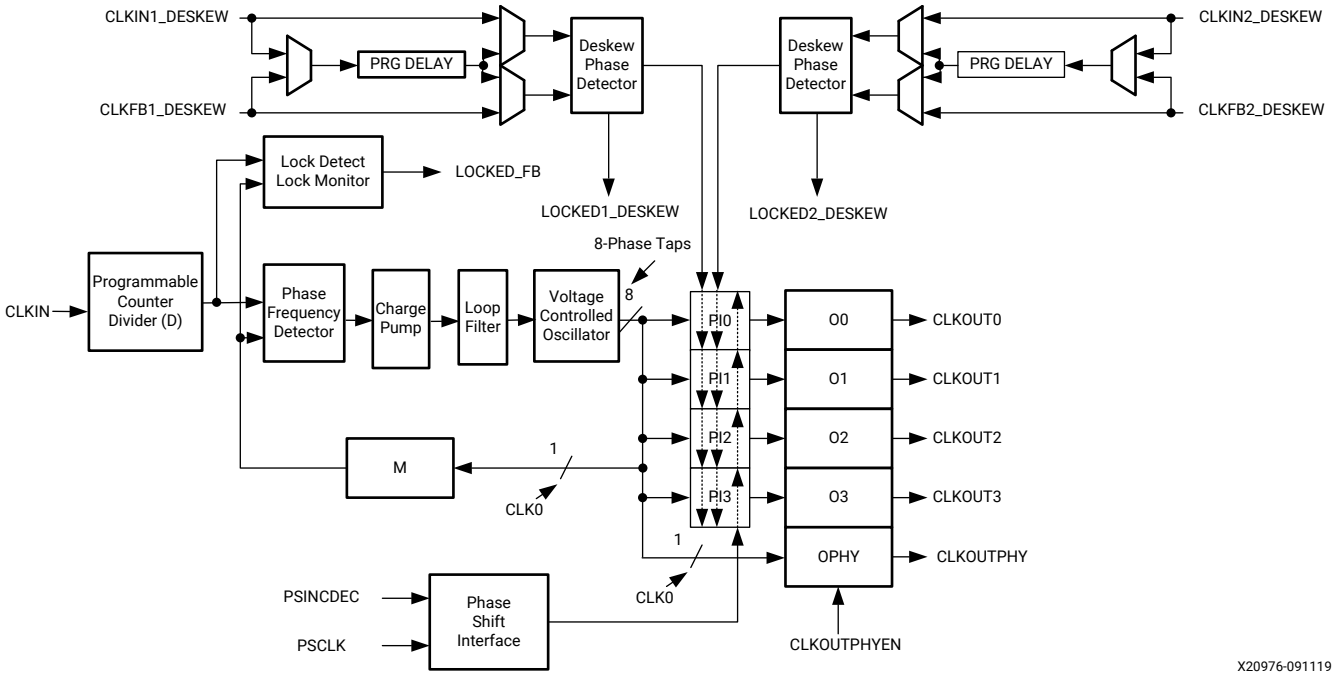
Table 16: **DPLL Attributes** *(cont'd)*

Attribute	Type	Allowed Values	Default	Description
CLKOUTn_PHASE_CTRL[0:1]	Binary	00 to 11	00	CLKOUT[0:3] counter variable fine phase shift or deskew select. 00: Interpolator is not controlled by either deskew or phase shift interface. 01: Interpolator is controlled by Deskew. 10: Interpolator is controlled by phase shift interface. Other binary values are not valid.
DESKEW_DELAY	Decimal	0 to 63	0	Value of the optional programmable delay in the deskew circuit.
DESKEW_DELAY_PATH	String	TRUE, FALSE	FALSE	Determines if the CLKIN_DESKEW path or the CLKFB_DESKEW path is selected for the optional programmable delay. TRUE = CLKFB_DESKEW, FALSE = CLKIN_DESKEW. If ZHOLD = TRUE, DESKEW_DELAY_PATH must be set to TRUE.
DESKEW_DELAY_EN	String	FALSE, TRUE	FALSE	Set to TRUE to enable the optional programmable delay in the deskew circuit. Must be set to TRUE if ZHOLD = TRUE.
ZHOLD	String	TRUE, FALSE	FALSE	Indicates the DPLL is configured to provide a negative hold time only at the HDIO registers.
LOCK_WAIT	String	FALSE, TRUE	FALSE	Wait during the configuration startup for the DPLL to lock.

XPLLs

The XPLL is a mixed-signal block designed to support clock network deskew, frequency synthesis, and jitter reduction. The primary function of the XPLL is to support the clocking needs of the memory interface. It also provides clocks to the programmable logic for various general purpose applications. There are two PLLs per XPIO bank that provide clocking to the XPHY logic and XPIO. In addition, they can be used as general purpose frequency synthesizers for a wide range of frequencies, serve as jitter filters, and provide phase shift capabilities as well as duty cycle programming. The basic functionality and operating functions are mostly identical to the description in MMCM section. The XPLLs differ from the MMCM in number of outputs (4), multipliers, and output dividers which have a lesser value range. However, the XPLL has special hardware features and connections to specifically support high performance direct clocking of the XPHY. XPLLs do not have fractional clock generation.

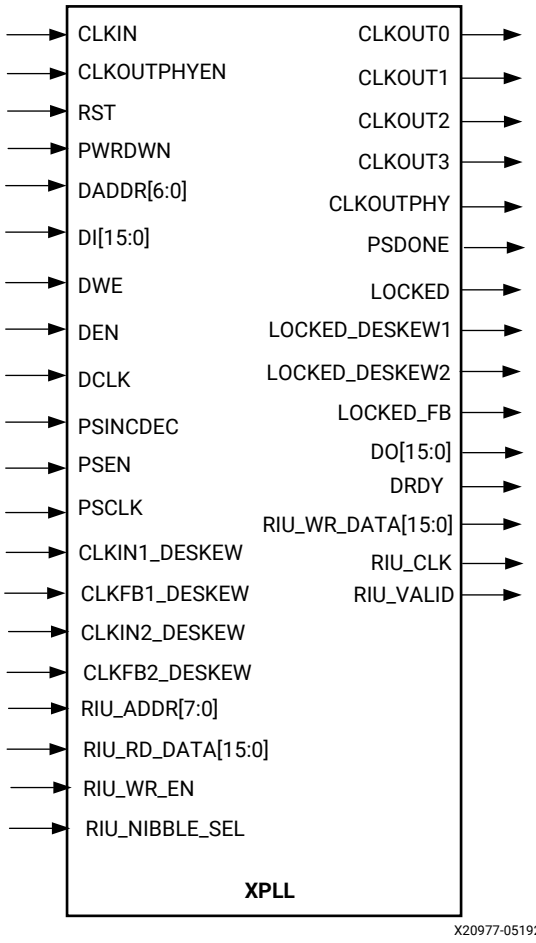
Figure 50: **XPLL Block Diagram**



XPLL Primitive

The Versal™ device XPLL primitive is shown in the following figure.

Figure 51: **XPLL Primitive**



The XPLL 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 in this section. The VCO operating frequency can be determined using the following relationship:

$$F_{vco} = F_{clkin} \times \frac{M}{D}$$

$$F_{clkout} = F_{clkin} \times \frac{M}{D \times O}$$

where the M, D, and O counters are shown in Figure 50. The value of M corresponds to the CLKFBOUT_MULT_F setting, the value of D to the DIVCLK_DIVIDE, and O to the CLKOUT_DIVIDE.

The five 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 because a single VCO drives all the counters.

Clock Network Deskew

See the MMCMs section of this manual.

Frequency Synthesis Using Integer Divide

See Using Fractional Divide in the MMCM section of this manual.

Jitter Filter

See Jitter Filter in the MMCM section of this manual.

Limitations

XPLLs have some restrictions that must be adhered to. The major limitations are the VCO operation range, input frequency, and phase shift. In addition, there are connectivity limitations to other clocking elements (pins, GTs, and clock buffers). Duty cycle is not programmable and is always nominally 50/50. Cascading MMCMs can only occur through the clock routing network.

VCO Operating Range

The minimum and maximum VCO operating frequencies are defined in the electrical specification of the Versal ACAP data sheets. These values can also be extracted from the speed specification.

Minimum and Maximum Input Frequencies

The minimum and maximum CLKIN input frequencies are defined in the electrical specification of the Versal ACAP data sheets.

Duty Cycle Programmability

See the MMCM Duty Cycle Programmability section of this manual. Note that duty cycle programmability is not available on CLKOUTPHY outputs.

Phase Shift

See Phase Shift in the MMCM section of this manual. Note that the phase shift discussion does not apply to the CLKOUTPHY outputs.

Counter Control

See the MMCM Counter Control section.

Detailed DCO and Output Counter Waveforms

See the MMCM Detailed VCO and Output Counter Waveforms section of this manual.

XPLL Ports

The XPLL has a set of ports specifically for the RIU XPHY interface signals (the XPLL-RIU interface). These pins can be directly connected to control the interface. For available pins see the tables below.

Table 17: **XPLL Ports**

Port Name	I/O	Description
CLKIN	Input	General clock input.
RST	Input	Asynchronous reset signal. The RST signal is an asynchronous reset for the XPLL. The XPLL synchronously re-enables itself when this signal is released (that is, XPLL re-enabled). A reset is required when the input clock conditions change (for example, frequency).
PWRDWN	Input	Powers down instantiated but unused PLLs.
CLKOUT[0:3]	Output	User configurable clock outputs (0 through 3) that can be divided versions of the VCO phase outputs (user controllable) from 2 to 128. The output clocks are phase aligned to each other (unless phase shifted).
CLKOUTPHYEN	Input	Asynchronous enable of the XPHY clock.
CLKOUTPHY	Output	Dedicated XPHY clock.
CLKIN1_DESKEW	Input	Primary clock input to the phase detector 1 block for deskewing clock network delays between two different CLKOUT networks.
CLKFB1_DESKEW	Input	Secondary (feedback) clock input to the phase detector 1 block for deskewing clock network delays.
CLKIN2_DESKEW	Input	Primary clock input to the phase detector 2 block for deskewing clock network delays between two different CLKOUT networks.
CLKFB2_DESKEW	Input	Secondary (feedback) clock input to the phase detector2 block for deskewing clock network delays.
LOCKED	Output	The LOCKED signal indicates that all functions requiring a LOCKED signal for the XPLL to operate properly have LOCKED. This LOCKED signal is therefore an AND function of LOCKED_FB and both LOCKED1/2_DESKEWs, if used.
LOCKED_FB	Output	An output from the XPLL that indicates when the XPLL has achieved phase alignment within a predefined window and frequency matching within a predefined PPM range. The XPLL automatically locks after power on. No extra reset is required. LOCKED is deasserted if the input clock stops or the phase alignment is violated (for example, input clock phase shift). The XPLL must be reset after LOCKED is deasserted.
LOCKED1/2_DESKEW	Output	Indicates if the deskew circuit is locked. Applies only to the deskew circuits used in the design. Ignore these outputs for unused deskew circuits.
DO[15:0]	Output	The dynamic reconfiguration output bus provides XPLL data output when using dynamic reconfiguration.
DI[15:0]	Input	The dynamic reconfiguration data input (DI) bus provides reconfiguration data. When not used, all bits must be set to zero.
DADDR[6:0]	Input	The dynamic reconfiguration address (DADDR) input bus provides a reconfiguration address for the dynamic reconfiguration. When not used, all bits must be assigned zeros.

Table 17: **XPLL Ports** *(cont'd)*

Port Name	I/O	Description
DRDY	Output	The dynamic reconfiguration ready output (DRDY) provides the response to the DEN signal for the XPLL's dynamic reconfiguration feature.
DWE	Input	The dynamic reconfiguration write enable (DWE) input pin provides the write enable control signal to write the DI data into the DADDR address. When not used, it must be tied Low.
DEN	Input	The dynamic reconfiguration enable (DEN) provides the enable control signal to access the dynamic reconfiguration feature. When the dynamic reconfiguration feature is not used, DEN must be tied Low.
DCLK	Input	The DCLK signal is the reference clock for the dynamic reconfiguration port.
PSCLK	Input	Dynamic phase shift clock.
PSEN	Input	Dynamic phase shift enable.
PSINCDEC	Input	Dynamic phase shift increment/decrement control.
PSDONE	Output	Dynamic phase shift done.
RIU_CLK	Output	DMC XPHY RIU interface clock.
RIU_ADDR<7:0>	Input	DMC XPHY RIU interface address.
RIU_WR_DATA<15:0>	Output	DMC XPHY RIU interface write data.
RIU_RD_DATA<15:0>	Input	DMC XPHY RIU interface read data.
RIU_WR_EN	Input	DMC XPHY RIU interface write Enable.
RIU_NIBBLE_SEL	Input	DMC XPHY RIU interface XPHY nibble select.
RIU_VALID	Output	DMC XPHY RIU interface valid.

Notes:

1. All control and status signals except PSINCDEC are active-High.

TIP: *The port names generated by the clocking wizard can differ from the port names used on the primitive.*

XPLL Port Descriptions

Many of the XPLL ports are identical to those for the MMCM (see MMCM Port Descriptions). For dynamic reconfiguration port description, see Dynamic Reconfiguration Port (DRP). The additional ports available for the XPLL are described below.

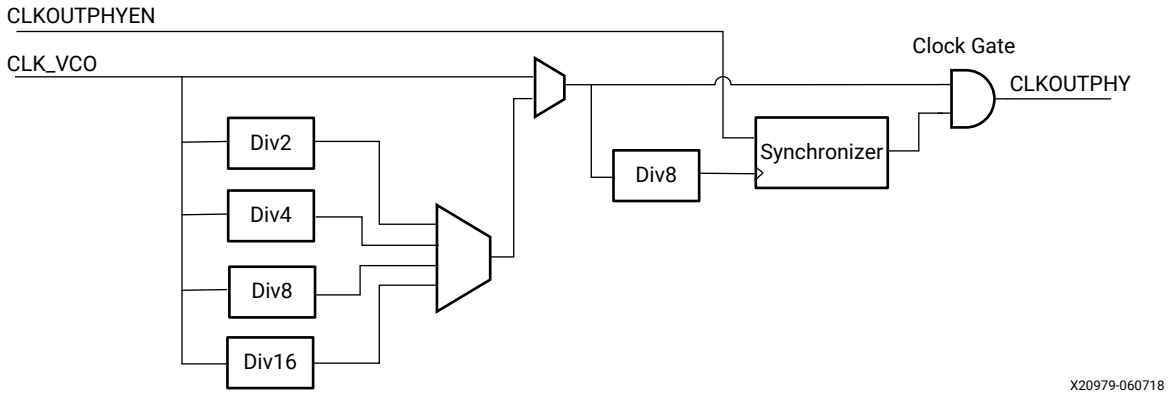
- **CLKOUTPHY and CLKOUTPHYEN:**

  The XPLL provides a dedicated CLKOUTPHY clock output to the XPHY. There is special logic associated with this clock output in order to support the synchronization between the programmable logic clock and the XPHY clock generated by VCO. CLKOUTPHY is gated by CLKOUTPHYEN signal from programmable logic. When CLKOUTPHYEN signal is asserted High from the programmable logic, CLKOUTPHY starts toggling so that the rising edge is aligned with the rising edge of the slowest programmable logic clock. CLKOUTPHYEN signal

is synchronized to a divide by 8 clock divider and gates the CLKOUTPHY clock. CLKOUTx_DIVIDE and CLKOUT_MULT are synchronized to the internal clock. However, phase alignment between multiple XPLL CLKOUTPHY clocks is only assured when both the CLKFBOUT_MULT and CLKOUT[0:1]_DIVIDE values are set to 1, 2, 4, 8, 16. Rising edges do not align for CLKFBOUT = 3, 5, 6, 7, 9,...

The following figure shows how clock division and multiplexing is achieved for enabling CLKOUTPHY clock based on CLKOUTPHYEN signal and CLKOUTPHY_DIVIDE attribute:

Figure 52: **CLKOUTPHY Multiplexer Scheme**



- **XPHY RIU interface pins:** These pins provide a direct connection to the hardened memory controller for highest performance deterministic timing.

XPLL Attributes

Table 18: **XPLL Attributes**

Attribute	Type	Allowed Values	Default	Description
CLKOUT[0:3]_DIVIDE	Integer	2 to 128	2	Specifies the amount to divide the associated CLKOUT clock output if a different frequency is desired. This number, in combination with the CLKFBOUT_MULT values, determines the output frequency.
CLKOUT[0:3]_ DUTY_CYCLE	Real	0.001 to 0.999	0.50	Specifies the duty cycle of the associated CLKOUT clock output in percentages (that is, 0.50 generates a 50% duty cycle).
CLKFBOUT_MULT	Decimal	4 to 43	42	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, determines the output frequency.

Table 18: **XPLL Attributes** *(cont'd)*

Attribute	Type	Allowed Values	Default	Description
CLKOUT[0:1]_PHASE	Decimal	–360.000 to 360.000	0.000	Allows specification of the output phase relationship of the associated CLKOUT clock output in number of degrees offset (that is, 90 indicates a 90° offset or ¼ cycle phase offset while 180 indicates a 180° offset or ½ cycle phase offset). Valid phase shifts are in 360÷CLKOUT[0:1]_DIVIDE degree increments.
DIVCLK_DIVIDE	Decimal	1 to 12	1	Specifies the division ratio for all output clocks with respect to the input clock.
REF_JITTER	Real	0.000 to 0.200	0.010	Allows specification of the expected jitter on the reference clock to better optimize PLL performance. A bandwidth setting of OPTIMIZED attempts to choose the best parameter for input clocking when unknown. If known, the value provided should be specified in terms of the unit interval (UI) (the maximum peak-to-peak value) of the expected jitter on the input clock.
CLKIN_PERIOD	Real	0.000 to 100.000	0.000	Specifies the input period in ns to the PLL CLKIN input. Resolution is down to the ps. This information is mandatory and must be supplied.
CLKOUTn_PHASE_CTRL[0:1]	Binary	00 to 11	00	CLKFBOUT counter variable fine phase shift or deskew select. 00: Interpolator is not controlled by either deskew or phase shift interface. 01: Interpolator is controlled by deskew1. 10: Interpolator is controlled by phase shift interface. 11: Interpolator is controlled by deskew2.
DESKEW_DELAY1	Decimal	0 to 63	0	Value of the optional programmable delay in the deskew1 circuit.
DESKEW_DELAY2	Decimal	0 to 63	0	Value of the optional programmable delay in the deskew2 circuit.
DESKEW_DELAY_PATH1	String	TRUE, FALSE	FALSE	Determines if the CLKIN1_DESKEW path or the CLKFB1_DESKEW path is selected for the optional programmable delay. TRUE = CLKIN1_DESKEW, FALSE = CLKFB1_DESKEW.
DESKEW_DELAY_PATH2	String	TRUE, FALSE	FALSE	Determines if the CLKIN2_DESKEW path or the CLKFB2_DESKEW path is selected for the optional programmable delay. TRUE = CLKIN2_DESKEW, FALSE = CLKFB2_DESKEW.

Table 18: **XPLL Attributes** *(cont'd)*

Attribute	Type	Allowed Values	Default	Description
DESKEW_DELAY_EN1	String	FALSE, TRUE	FALSE	Set to TRUE to enable the optional programmable delay in the deskew circuit 1.
DESKEW_DELAY_EN2	String	FALSE, TRUE	FALSE	Set to TRUE to enable the optional programmable delay in the deskew circuit 2.
CLKOUTPHY_DIVIDE	String	DIV1 (bypass), DIV2, DIV4, DIV8, DIV16	DIV8	Determines the CLKOUTPHYP and CLKOUTPHYN clocks by the VCO output clock.
LOCK_WAIT	String	FALSE, TRUE	FALSE	Wait during the configuration startup for the XPLL to lock.
XPLL_CONNECT_TO_NOCMC	String	NONE, LP4, DDR	NONE	Indicates if XPLL is used to drive the DDRMC.

XILINX

Appendix A

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the Vivado® IDE, select **Help → Documentation and Tutorials**.
- On Windows, select **Start → All Programs → Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on DocNav, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this guide:

1. *Vivado Design Suite User Guide: Using Constraints* (UG903)
2. *Versal ACAP Hardware, IP, and Platform Development Methodology Guide* (UG1387)
3. *Versal Architecture AI Core Series Libraries Guide* (UG1353)
4. *Clocking Wizard for Versal ACAP LogiCORE IP Product Guide* (PG321)
5. *Versal ACAP SelectIO Resources Architecture Manual* (AM010)
6. Versal ACAP data sheets
   - *Versal Architecture and Product Data Sheet: Overview* (DS950)
   - *Versal Prime Series Data Sheet: DC and AC Switching Characteristics* (DS956)
   - *Versal AI Core Series Data Sheet: DC and AC Switching Characteristics* (DS957)

Please Read: Important Legal Notices

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright