

Versal ACAP SelectIO Resources

Architecture Manual

AM010 (v1.1) November 24, 2020



Revision History

The following table shows the revision history for this document.

Section	Revision Summary
11/24/2020 Version 1.1	
XP XPHY	<ul style="list-style-type: none"> Removed bullet about XPHY UNISIM primitive. Updated bullet about QBC and DBC functionality.
Bidirectional Datapath and Controlling IBUF_DISABLE and DYN_DCI	Added new sections.
Table 4	Updated Connection (RX) column for CLK_TO_LOWER, CLK_TO_UPPER.
Delays	<ul style="list-style-type: none"> Removed description of tap value from note. Added sentence that CRSE delay cannot be controlled through the PL.
Controlling Delays	Updated section and added Figure 8 .
Controlling FIFO Modes	Added note about not registering FIFO_EMPTY as part of the FIFO_RDEN = !FIFO_EMPTY connection.
Tristate Control	Rewrote section.
Controlling Tristate Control	Rewrote first sentence.
Table 14	Updated second bullet in Controlled By column for VTC.
Controlling IBUF_DISABLE and DYN_DCI	Added PHY SM description and sequence.
Register Interface Unit	<ul style="list-style-type: none"> Added references to Ports and Attributes sections in most tables. Updated description of bit [2] in Table 19.
Table 19	Updated description of bits [0], [1], and [2].
Table 20	Updated description of bit [13].
Table 38	Added new table.
Table 41	Updated description of bits [9:0].
Table 44	Updated description of PHY_RDEN, PHY_WREN, DYN_DCI, RST, TX_RST, and IBUF_DISABLE.
Table 45	Updated description of CONTINUOUS_DQS, ODT_SRC_<0-5>, RX_GATING, TX_INIT_<0-5>, and TX_INIT_TRI.
Reset Sequence	Updated section, including both figures.
XPHY Usage	Added Tcl commands to change time value of input and output delays of a NIBBLESlice.
Table 61, Table 62, Table 63, Table 64, Table 65, Table 66, Table 73, Table 74, Table 75, Table 76, Table 77, Table 78, and Table 79	Removed OFFSET_CNTRL attribute.
Table 68 and Table 70	Updated description of OSC[3:0], OSC_EN[1:0], and VREF.
Table 73	Added VOH (DIFF_SSTL15 ONLY) attribute.
Table 76	Added note about bidirectional configuration on I/O standards.

Section	Revision Summary
Receiver Offset Control	Removed section.
Differential Termination Attribute	Added VCCO requirement for differential impedance block.
DQS_BIAS, DC_BIAS, and AC Coupling	Updated DC_BIAS and AC Coupling Recommendations sections.
XP IOB Pre-emphasis and Equalization	Updated first paragraph and Table 92 .
HD IOL Features	Updated DPLL section.
HD IOB Primitives	Removed IBUF_IBUFDISABLE, IBUFDS_IBUFDISABLE, and IBUFDS_DIFF_OUT_IBUFDISABLE.
Single-Ended Input Buffer Primitives	Removed IBUF_IBUFDISABLE throughout.
Figure 66	Added new figure.
Differential Input Buffer Primitives	Replaced IBUFDISABLE with INTERMDISABLE throughout.
Differential Bidirectional Buffer Primitives	Corrected primitive names throughout.
07/29/2020 Version 1.0	
Initial release.	N/A

Table of Contents

Revision History	2
Chapter 1: Overview	6
Introduction to Versal ACAP.....	6
SelectIO Resources Features.....	8
SelectIO Resources Architecture.....	9
Differences from Previous Generations.....	12
Chapter 2: XPHY Architecture	14
XPHY Nibble.....	14
Clocking.....	22
XPHY Features.....	28
XPHY Primitive.....	74
XPHY Usage.....	92
Chapter 3: XP IOL Resources	94
XP IOL Features.....	94
Chapter 4: XP IOB Resources	106
XP IOB Banking Structure.....	106
XP IOB Supported Standards.....	108
XP IOB Internal V_{REF}	127
XP IOB Driver Control, Internal Termination, and Internal Bias.....	129
XP IOB IBUFDISABLE.....	138
XP IOB Pre-emphasis and Equalization.....	138
Chapter 5: XP Bank Supporting Resources and Corner Banks	140
Clocking Resources.....	140
Boundary Logic Interface.....	141
Corner Banks.....	142
Chapter 6: HD IOL Resources	143
HD IOL Features.....	143

HD IOL Primitives.....	144
Chapter 7: HD IOB Resources.....	150
HD IOB Banking Structure.....	150
HD IOB Features.....	152
HD IOB Supported Standards.....	154
Appendix A: Additional Resources and Legal Notices.....	164
Xilinx Resources.....	164
Documentation Navigator and Design Hubs.....	164
References.....	164
Please Read: Important Legal Notices.....	165

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 2: XPHY Architecture](#)
 - [XPHY Usage](#)
 - [Chapter 4: XP IOB Resources](#)
 - [Chapter 7: HD IOB Resources](#)
- **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 2: XPHY Architecture](#)
- **System Integration and Validation:** Integrating and validating the system functional performance, including timing, resource use, and power closure. Topics in this document that apply to this design process include:
 - [Chapter 2: XPHY Architecture](#)
 - [Chapter 3: XP IOL Resources](#)
 - [Chapter 6: HD IOL Resources](#)

- **Board System Design:** Designing a PCB through schematics and board layout. Also involves power, thermal, and signal integrity considerations. Topics in this document that apply to this design process include:
 - [Chapter 2: XPHY Architecture](#)
 - [Clocking](#)
 - [Chapter 4: XP IOB Resources](#)
 - [Chapter 7: HD IOB Resources](#)

SelectIO Resources Features

The two primary types of I/O in Versal ACAPs are high-performance XP I/O (XPIO) and high-density HD I/O (HDIO). The XPIO includes dedicated logic to support high-speed interfaces with voltage ranges between 1.0V and 1.5V. HDIO and XPIO banks do not have overlapping voltages or I/O standards. The HDIO supports interfaces with voltages ranging from 1.8V to 3.3V. The HDIO provides logic for both single data rate (SDR) and double data rate (DDR) interfaces at reduced clocking speeds.

XPIO Features

The XPIO are grouped into 54-pin banks with supporting resources for both high-performance and low-speed interfaces. Each XPIO can use the XPHY to align, serialize, and de-serialize a data stream. Each XPIO has I/O interconnect logic (IOL) resources to support low-speed SDR and DDR interfaces and coarse data alignment resources. The XPIO input and output buffers support a wide range of single-ended and differential I/O standards along with resources to support a high level of signal quality.

- 1.0V, 1.2V, 1.35V, and 1.5V bank voltage standards
- XPHY logic resources to align and serialize/de-serialize high-speed data streams
- IOL logic resources to provide simplified lower-bandwidth SDR and DDR logic support
- Internally generated V_{REF} support shared across nibble boundaries
- Calibrated output drive support
- Calibrated internal termination
- Internal differential termination
- Internal bias support
- Transmitter pre-emphasis and receiver equalization
- Native support for MIPI D-PHY interfaces

- Supports serialization/deserialization ratios of 1:8, 1:4, and 1:2

HDIO Features

The HDIOs are grouped into 22-pin banks with supporting resources for low-performance interfaces. Each HDIO has IOL resources to simplify the support for low-speed SDR and DDR interfaces and coarse data alignment resources. In addition to logic, the HDIO output buffers provide resources to drive single-ended and pseudo-differential standards. The HDIO input buffers can receive several single-ended and differential standards. HDIOs are optimized for single-ended, voltage-referenced, and pseudo-differential I/O standards operating at low data rates.

- IOL logic resources support low-speed interfaces with SDR and DDR logic
- IODELAY can provide up to 1.8 ns uncalibrated output delay
- IODELAY can be cascaded with output delay to provide up to 3.6 ns uncalibrated input delay
- 1.8V, 2.5V, and 3.3V bank voltage I/O standard support
- Uncalibrated output drive and slew control
- Internal V_{REF} on a bank-wide resolution
- LVDS and LVPECL input support with external termination

SelectIO Resources Architecture

All Versal™ devices have configurable SelectIO interface drivers and receivers, supporting a wide variety of standard interfaces. The robust feature set includes programmable control of output strength and slew rate, on-chip termination, and an internally generate a reference voltage (INTERNAL_VREF). Each Versal device contain XPIO banks that contain 54 SelectIO pins and can implement both single-ended and differential I/O standards. XPIO banks support the highest speed interfaces powered at or below 1.5V. Some Versal devices contain HDIO banks that can interface with voltage levels between 1.8V and 3.3V. The HDIO banks contain 22 SelectIO pins that can implement both single-ended I/O standards and differential I/O standards. Every SelectIO IOB resource contains input, output, and tristate drivers. The SelectIO pins can be configured to various I/O standards, both single-ended and differential.

- Single-ended I/O standards are, for example, LVCMOS, LVTTTL, HSTL, SSTL, HSUL, LVSTL, and POD
- Pseudo-differential standards are, for example, differential HSTL, POD, HSUL, LVSTL, and SSTL
- LVDS is the true differential standard.

Related Information

[XP IOB Resources](#)

[HD IOB Resources](#)

Supply Voltages and Dedicated SelectIO Pins

V_{CC0}

The V_{CC0} supply is the primary power supply for drivers and termination. The [XP IOB Supported Standards](#) section includes tables that outline the V_{CC0} requirements for each of the supported I/O standards, and illustrate the V_{CC0} requirements for inputs and outputs including the optional internal differential termination circuit. All V_{CC0} pins for a given XP or HD I/O bank must be connected to the same external voltage supply on the board, and as a result, all of the I/O within a given I/O bank must be compatible with the same V_{CC0} level. The V_{CC0} voltage must match the requirements for the I/O standards that have been assigned to the I/O bank.



CAUTION! Incorrect V_{CC0} voltages can result in loss of functionality or damage the device.

V_{CCAUX}

The global auxiliary (V_{CCAUX}) supply rail primarily provides power to the receive circuitry. In the I/O banks, V_{CCAUX} is also used to power input buffer circuits for some of the I/O standards. Additionally, the V_{CCAUX} rail provides power to the differential input buffer circuits used for most of the differential and V_{REF} I/O standards.

$V_{CC_{IO}}$

$V_{CC_{IO}}$ is an internal supply for I/O banks. It supplies the digital portions and supporting logic SelectIO resources.

IO_VR_700 / IO_VR_800

In XP I/O bank 700 and bank 800 (not present in all devices), there is an additional bank pin that is used as a reference to calibrate internal on-die termination. The IO_VR pin must be externally connected to a 240 Ω resistor on the PCB and pulled up to the bank V_{CC0} voltage. See [Calibrated Termination \(Digitally Controlled Impedance\)](#). HD I/O banks do not support calibrated termination and thus no equivalent pin or reference resistor is required on HD I/O banks.



IMPORTANT! The IO_VR_700 and IO_VR_800 pins (not available on all devices) must have an external 240 Ω resistor tied to VCCO_700 and VCCO_800 respectively. These pins are dedicated and can not be used as user I/O. All designs MUST populate these pins appropriately, regardless of the I/O standards used in a design.

Power Supply Sequencing Requirements

The power supply requirements, including power-on and power-off sequencing, are described in [Versal ACAP data sheets](#).

State of I/Os During and After Configuration

During configuration, I/O drivers are tristated in all banks. During configuration (until the applications settings take over), all XP I/O banks use the default IOSTANDARD = LVCMOS15, SLEW = FAST, and DRIVE = 12 mA setting. The corresponding setting in HD I/O banks is IOSTANDARD = LVCMOS25, SLEW = FAST, and DRIVE = 12 mA. The PUDC_B input pin can be used to enable internal pull-ups during configuration. After configuration, the unused I/Os have tristated drivers, and the pads are weakly pulled-down.

Note: This manual applies only to PL-based HD and XP SelectIO resources. Multiplexed I/O are described in *Versal ACAP Technical Reference Manual* ([AM011](#)).

I/O Banking Rules

In the Versal architecture, XP and HD IOBs use the bank V_{CCO} supply for drivers, on-die biasing, on-die termination, and the receive block. Because of the dependency on V_{CCO} , all outputs and many inputs must operate at a specific V_{CCO} level making it a dominant factor in determining the IOSTANDARDS that can reside in the same bank.

Rules for Combining Standards Different Standards in the Same Bank

- V_{CCO} levels must be compatible for all inputs and outputs in the same HD or XP I/O bank
- INTERNAL_VREF levels must be compatible for all inputs in the same HD or XP I/O bank

Note: In Versal devices, all single-ended input and all output (single-ended and differential) IOSTANDARDS have a required V_{CCO} level. Only the differential inputs that do not use ODT, PULLTYPE, or DIFF_TERM can reside in multiple V_{CCO} domains. Although a differential input may be compatible with multiple V_{CCO} domains, it is important to note that data sheet input specifications are impacted by the V_{CCO} level and compatibility should be verified when selecting a V_{CCO} level.

The supported standards and their associated V_{CCO} and INTERNAL_VREF requirements are described in the XP and HD IOB supported sections.

Related Information

[XP IOB Supported Standards](#)

[HD IOB Supported Standards](#)

Differences from Previous Generations

Versal™ ACAPs have several important feature enhancements as well as updates to existing features.

XP XPHY

The following table summarizes the key differences between the UltraScale™ architecture PHY and the Versal™ architecture XPHY.

Table 1: UltraScale Architecture PHY and Versal Architecture XPHY Key Differences

Function	Versal Architecture XPHY	UltraScale Architecture PHY
NIBBLESLICEs per nibble	6	6 or 7
Nibbles per bank	9 (54 pins)	8 (52 pins)
Serialization	8:1, 4:1, 2:1	8:1, 4:1
Deserialization	1:8, 1:4, 1:2	1:8, 1:4
Wizard required to access interface	Yes	No
Input and output delays	625 ps (512 taps)	UltraScale devices: 1250 ps (512 taps) UltraScale+ devices: 1100 ps (512 taps)

Some of the other differences between the PHY architectures of UltraScale™ and Versal devices include the following:

- Receive FIFO bypass support for low-latency applications
- No NIBBLESLICE 0 (formerly called BITSlice 0) instantiation requirements
- The IDELAYCTRL, ISERDES, OSERDES, RXTX_BITSLICE, RX_BITSLICE, TX_BITSLICE, BITSlice_CONTROL, and RIU_OR UNISIM primitives are not supported
- The XP IOL features are independent of the XPHY
- Programmable logic control ports are shared between input and output delays through a delay select port
- Some XPIO banks (typically located on the corner of the device) have pins that have limited function and can only be used for DDR memory controller functionality. See the *Versal ACAP Packaging and Pinouts Architecture Manual (AM013)* for specific pin information.
- QBC and DBC functionality has been split into two parts: Strobes now enter on XCC pins, while inter-nibble and inter-byte clocking capabilities are determined by the nibble rather than the pin itself.
- The PHY can only be constructed by using the Advanced IO Wizard together with the Advanced I/O Planner (see *Advanced I/O Wizard LogiCORE IP Product Guide (PG320)*).

XP IOL

- The XP IOL block provides SDR and DDR logic
- IODELAY provides internal delay support of up to 3.6 ns on input (when cascaded)
- IODELAY provides internal delay support of up to 1.8 ns on output
- Unused XPHY I/O within the nibble are available to the XP IOL
- IODELAY must use the input ports to dynamically select the delay tap. DELAY_VALUE is not supported.

XP IOB

- Banks located on the top and/or bottom periphery of the device
- Banks are groups of 54 IOBs. Each IOB is capable of both single-ended and differential signaling.
- V_{REF} is internally generated (only)
- DCI reference resistors are no longer required on a per bank basis (a maximum of two resistors per device)
- Output drive and termination are calibrated against the DCI reference resistor
- No support for 1.8V bank voltages. The supported bank voltages are 1.0V, 1.2V 1.35V, and 1.5V
- Output drive strength support of 4 mA, 8 mA, and 12 mA
- LVDS supported in 1.5V banks

HD IOL

- Internal input delay support of up to 3.6 ns when cascaded with the output delay or 1.8 ns when not cascaded
- Internal delay support of up to 1.8 ns on output
- Each bank has a digital PLL (DPLL)
- Static clock insertion delay compensation block through DLL (ZHOLD)

HD IOB

- Supported bank voltages of 1.8V, 2.5V, and 3.3V
- Output drive strength support of 4 mA, 8 mA, and 12 mA
- 22 pins per bank
- No class II support for SSTL standards

XPHY Architecture

XPHY is the high-performance I/O interface on the Versal™ ACAP XPIOs. There are nine XPHY nibbles in an XPIO bank, with each XPHY nibble containing six XPHY NIBBLESLICEs that transmit and/or receive data from six individual I/O pins, for a total of 54 pins per bank. Each XPHY NIBBLESLICE is composed of a serializer, deserializer, I/O delays, and a receiver FIFO. The Versal device XPHY is equipped with voltage and temperature compensation (VTC) and a mechanism for automatic delay adjustment for optimal data eye centering through the built-in self-calibration (BISC) feature in each XPHY nibble. I/O delays can also be controlled through the programmable logic. Control of the XPHY features is available through the register interface unit (RIU) in each nibble.

XPHY is used to support the following applications:

- DDR4 and LPDDR4 integrated memory controllers supported through the IP catalog in the Vivado® tools
- DDR4, QDR IV, and RLDRAM3 memory controllers supported through the IP catalog in the Vivado tools
- MIPI D-PHY v1.2
- Gigabit Ethernet 1000Base-X and SGMII
- Toggle NAND flash
- High-speed source-synchronous and asynchronous I/O interfaces supported through the Advanced IO Wizard in the IP catalog of the Vivado tools
- IOB feed-through to programmable logic

XPHY Nibble

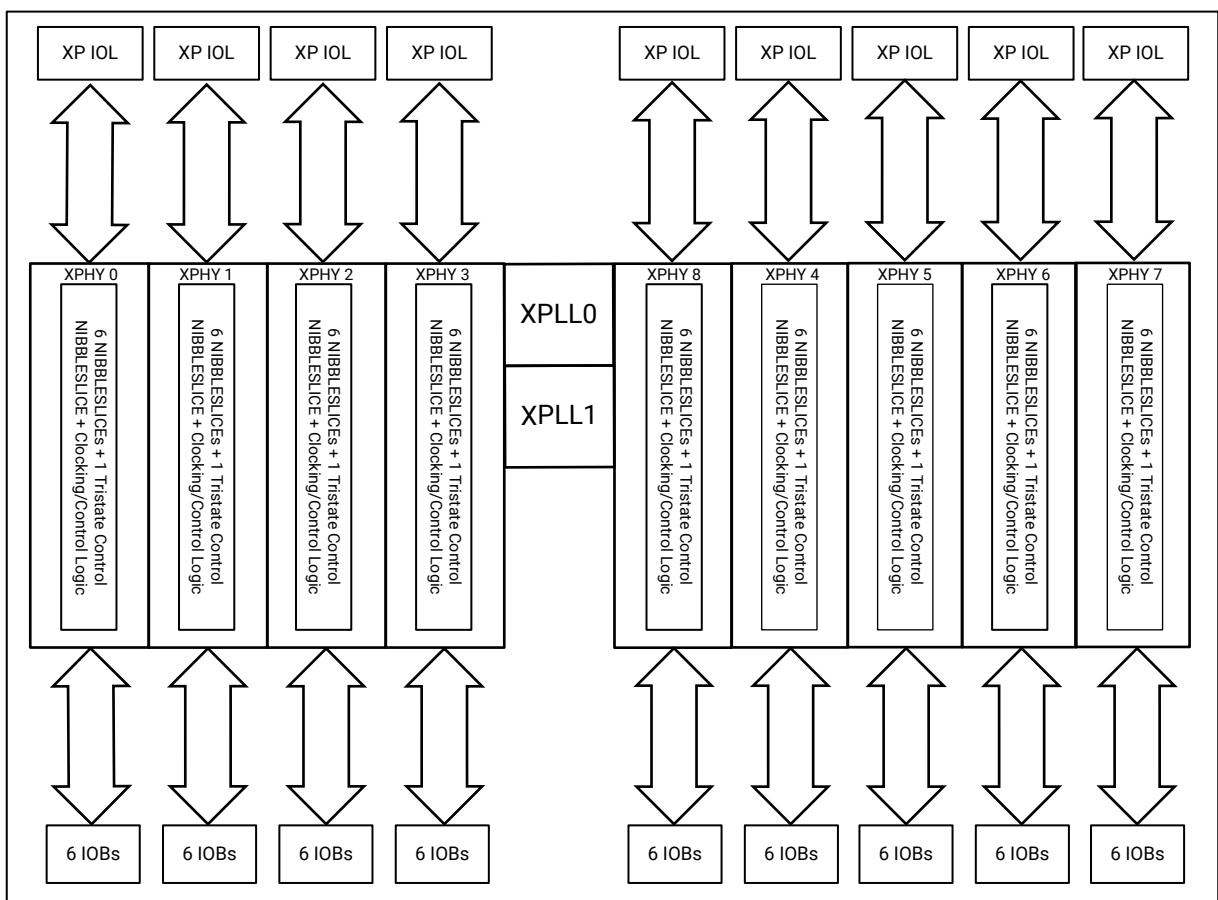
XPHY is the high-performance I/O interface for an XPIO bank. Every XPIO bank has nine XPHY nibbles. Each XPHY nibble is defined as six XPHY NIBBLESLICEs and its associated features. XPHY NIBBLESLICEs contain input and output logic, composed of a serializer, deserializer, I/O delays, and a receiver FIFO. XPHY NIBBLESLICEs can operate as a transmitter or receiver. An XPHY nibble also performs the following functions/features:

- Built-in self-calibration (BISC) aids in alignment and uses voltage and temperature compensation (VTC) to adjust delay lines

- Generates clocks for the receiver and transmitter functions in the XPHY NIBBLESLICES
- Gives access to the register interface unit (RIU) that provides access to all features of an XPHY nibble
- Tristate control
- TX to RX loopback
- Serial mode, which supports receiver interfaces where the clock and data phase relationship is unknown (any interface that is not source-synchronous)

The layout of XPHY nibbles in an XPIO bank is represented in the following figure.

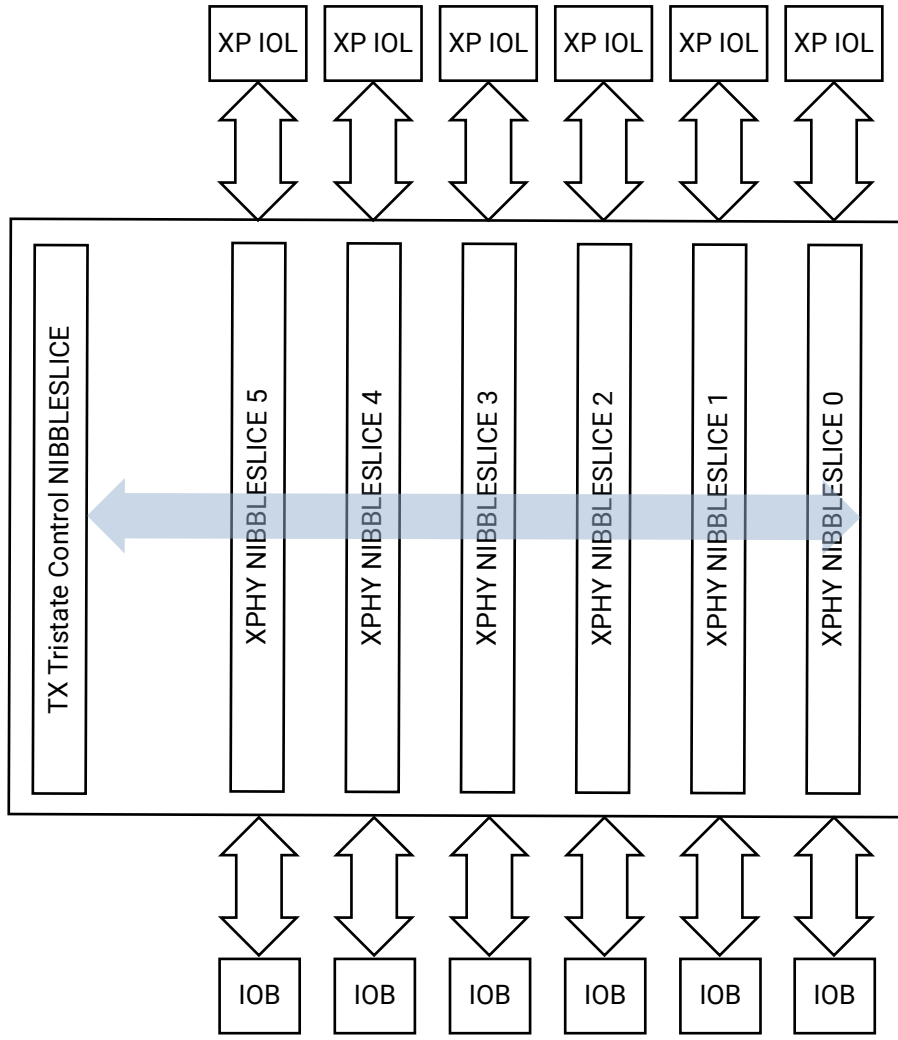
Figure 1: Relationship of XPHY Nibble, XP IOL, and IOB within an XPIO Bank



X21591-042319

The following figure is a detailed view of the previous figure, representing the relationship between a single XPHY nibble, XP IOL, and IOB within an XPIO bank.

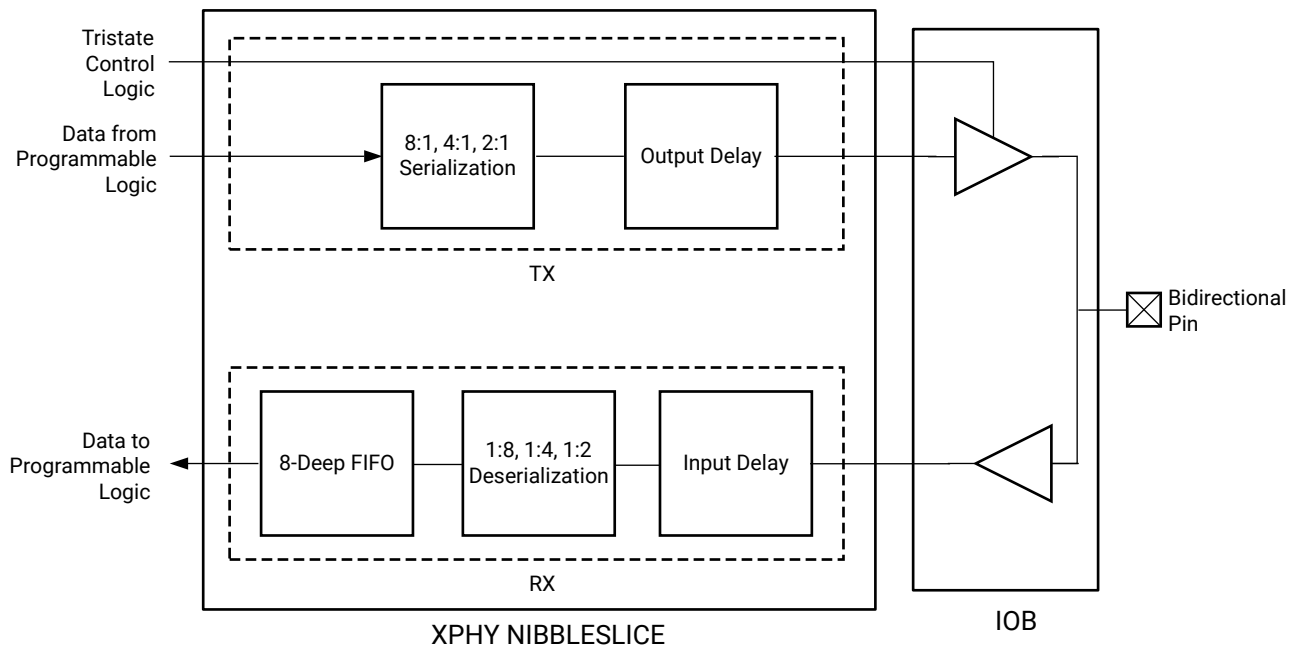
Figure 2: Relationship Between a Single XPHY Nibble, XP IOL, and IOB



X21592-042319

The following figure shows an XPHY NIBBLESlice.

Figure 3: XPHY NIBBLESLICE with TX and RX Datapaths



X21593-121219

TX Datapath

The TX datapath is composed of the following:

- **Serializer:** The serializer supports 8:1, 4:1, and 2:1 serialization. This is set by the TX_DATA_WIDTH attribute.
- **Output Delay:** Output delays can delay outgoing serialized data up to 512 taps (0–511 taps), with a minimum of 625 ps of available delay.

Refer to the [Controlling Tristate Control](#) section for latencies with and without the TX datapath using tristate control.

Related Information

[Controlling Delays](#)

[Controlling Tristate Control](#)

RX Datapath

The RX datapath is composed of:

- **Input delay:** Input delays can delay incoming serialized data up to 512 taps (0–511 taps), with a minimum of 625 ps of available delay. Input delays can be increased to 1024 taps (0–1023 taps) for a minimum of 1250 ps of available delay by cascading the output delay of an XPHY NIBBLESlice to the end of its input delay. For more information on cascading, see the `CASCADE_<0-5>` attribute.
- **Deserializer:** The deserializer supports 1:8, 1:4, and 1:2 deserialization. This is determined by the `RX_DATA_WIDTH` attribute.
- **FIFO:** The receiver of an XPHY NIBBLESlice has an 8-deep FIFO. The parallel data written to the FIFO is synchronized to the programmable logic clock domain of choice before passing to the programmable logic.

RX datapath latency changes depending on the data width (`RX_DATA_WIDTH`) and `FIFO_MODE_x` attribute. Refer to [Controlling FIFO Modes](#) for RX datapath latencies.

★ **IMPORTANT!** *Because each NIBBLESlice routes to a specific pin, receiving a differential signal (regardless of whether clock or data) consumes the pins and RX datapaths of both NIBBLESlices.*

★ **IMPORTANT!** *If receiving a strobe and `RX_GATING = ENABLE`, bit slip is not needed. For all other cases, bit slip is needed for word alignment.*

Related Information

[FIFO](#)

Bidirectional Datapath


The TX and RX datapaths within each XPHY NIBBLESlice can be used together to form a bidirectional datapath. As shown in [Figure 3](#), each TX datapath drives both to the pad and the RX datapath. As such, care must be taken when using the bidirectional datapath so as to tristate the buffer or gate the datapaths without data loss.

The XPHY offers transmit gating, receive gating, and tristating as mechanisms to control the bidirectional datapath. See the table below for a summary of how to enable these controls through the XPHY attributes.

Table 2: Enabling Bidirectional Datapath Control

Control Mechanism	Related Attributes
Gating	<ul style="list-style-type: none"> RX datapath gating:: The RX_GATING attribute enables gating of the RX datapath based on the PHY_RDEN port. While RX_GATING enables gating, the CONTINUOUS_DQS attribute lets users choose between PHY_RDEN operating based in the PLL_CLK or the strobe clock domain. TX datapath gating:: The TX_GATING attribute enables gating of the TX datapath based on the PHY_WREN port (which is serialized but not inverted when used for gating) and PLL_CLK ports. NIBBESLICE[1] cannot be gated.
Tristating	<ul style="list-style-type: none"> Tristating:: TBYTE_CTL_# determines whether tristating is controlled by the T (combinatorial) port or an inverted and serialized PHY_WREN port (which is in the PLL_CLK domain).

T_OUT[5:0] is the tristate control output from the XPHY. Each bit of T_OUT is associated with a NIBBESLICE, and TBYTE_CTL_# allows each NIBBESLICE to select its corresponding T_OUT bit to be controlled by either T or PHY_WREN. In other words for a NIBBESLICE[x], T_OUT[x] reflects the tristate control input selected by TBYTE_CTL_x. If TBYTE_CTL_x = T, T_OUT[x] (associated with NIBBESLICE[x]) is controlled via the T[x] input. Because this is a combinatorial route, T_OUT[x] is not aligned to the data. If TBYTE_CTL_x = PHY_WREN, T_OUT[x] (associated with NIBBESLICE[x]) is controlled through the PHY_WREN input. This input is inverted, serialized, and output synchronously (through T_OUT[x]) with the TX data when used for tristating. For more information, see [Controlling Tristate Control](#).

 **IMPORTANT!** When using 2:1 serialization, each NIBBESLICE tristate buffer can only be controlled through the combinatorial T input (TBYTE_CTL_<0-5> = T). Tristate control through the PHY_WREN input (TBYTE_CTL_x = PHY_WREN) is only possible for 8:1 and 4:1 serialization.

PHY_RDEN is set up and used to control RX datapath gating is as follows:

- PHY_RDEN controls accepting or rejecting the strobe entering on NIBBESLICE[0] or from inter-byte clocking (inter-nibble clocking does not support gating through PHY_RDEN), depending upon the settings of CONTINUOUS_DQS, RX_GATING, and RX_DATA_WIDTH. Always ensure the strobe has stabilized and BISC has completed before asserting PHY_RDEN. Refer to [Controlling Built-in Self-Calibration](#) for when BISC is considered completed.
- When RX_DATA_WIDTH = don't care, RX_GATING = ENABLE, and CONTINUOUS_DQS = TRUE, then the four bits of PHY_RDEN are OR'd together and that output is used to control the gate. If the result of the OR operation is 1, the strobe is accepted. If it is 0, then the strobe is rejected. PHY_RDEN is synchronized to the strobe for this attribute combination. When CONTINUOUS_DQS = TRUE, send 3 strobe cycles before sending data.

- When `RX_DATA_WIDTH = 4` or `8`, `RX_GATING = ENABLE`, and `CONTINUOUS_DQS = FALSE`, set the following bits of `PHY_RDEN` to `1` to accept the strobe or `0` to reject the strobe. `PHY_RDEN` is synchronized to `PLL_CLK` for this attribute combination. Each bit of `PHY_RDEN` controls two UI worth of data:
 - If `RX_DATA_WIDTH = 8`: `[3:0]`
 - If `RX_DATA_WIDTH = 4`: `[2][0]`
 - If `RX_DATA_WIDTH = 2`: not supported
- When `RX_GATING = DISABLE` the gate is always open, regardless of the value of `RX_DATA_WIDTH`, `CONTINUOUS_DQS`, or `PHY_RDEN`. In this scenario (`RX_GATING = DISABLE`), the strobe starts the deserialization in the RX datapath. Because of this, the strobe must be stable to ensure XPHY alignment.
- When `SERIAL_MODE = TRUE`, tie all four bits High

`PHY_WREN` is set up and used to control TX datapath gating as follows:

- When `TX_GATING = ENABLE`, `PHY_WREN` gates the TX datapath of `NIBBLESlice[0]`, `NIBBLESlice[2]`, `NIBBLESlice[3]`, `NIBBLESlice[4]`, and `NIBBLESlice[5]`. `NIBBLESlice[1]` cannot be gated. Set the following bits of `PHY_WREN` to `0` to gate transmit data or `1` to not gate transmit data:
 - If `TX_DATA_WIDTH = 8`: `[3:0]`
 - If `TX_DATA_WIDTH = 4`: `[2][0]`
 - If `TX_DATA_WIDTH = 2`: not supported
- Note that `PHY_WREN` can be used to control both TX datapath gating (if `TX_GATING = ENABLE`) and tristating (if `TBYTE_CTL_# = PHY_WREN`). However, only when `PHY_WREN` is used for tristating is it inverted and serialized prior to its use. When used for gating, `PHY_WREN` is serialized but is not inverted. Thus, when used for gating, `PHY_WREN` should be set to `1` to open the gate and `0` to close the gate. When used for tristating, `PHY_WREN` should be set to `0`, which is then inverted to `1` to tristate the buffer. It follows that setting `PHY_WREN` to `1` for tristating results in the buffer not being tristated. See [Controlling Tristate Control](#) for more information on tristating.

Other important points to keep in mind:

- When turning the bus around, toggle the `BS_RESET_CTRL.clr_gate` bit then toggle the `BS_RESET_CTRL.bs_reset` bit. Toggling `BS_RESET_CTRL.clr_gate` clears the strobe path gating logic, helping to ensure proper alignment when combined with the `NIBBLESlice` reset performed through the toggling of `BS_RESET_CTRL.bs_reset`. See [Register Interface Unit](#) for more information on `BS_RESET_CTRL`. After the write to `bs_reset` is completed, data can be transmitted immediately. For receivers, however, the first `FIFO_EMPTY` deassertion should be used to know when receiving valid data.
- Before performing a `bs_reset`, set `PHY_WREN` to `0` regardless of the `TX_GATING` setting.
- Setting `CONTINUOUS_DQS = TRUE` requires that three strobe cycles be received prior to receiving data to prevent data loss.

- If the TX-only interface data and clock, as well as bidirectional interface data, exist in the same nibble then TBYTE_CTL_# must be set to T for all pins in either interface, regardless of if they are part of the TX-only interface or bidirectional interface, and TX_GATING must be set to DISABLE
 - If the TX-only interface clock is placed in NIBBLESlice[1], TX_GATING can be set to ENABLE because NIBBLESlice[1] cannot be gated. In this scenario, TBYTE_CTL_# should be set to PHY_WREN for the bidirectional pins in the nibble, and TBYTE_CTL_# should be set to T for the TX-only pins in the nibble. If the TX-only interface clock is not placed on NIBBLESlice[1], TX_GATING must be set to DISABLE, and TBYTE_CTL_# must be set to T for all pins in the interfaces, regardless of whether they are TX-only or bidirectional.
- When TX_DATA_WIDTH = 2 or RX_DATA_WIDTH = 2, bidirectional support is limited to:
 - TX_GATING must be set to DISABLE.
 - RX_GATING can be set to ENABLE, but only when CONTINUOUS_DQS is also set to TRUE.
 - Tristating is only supported through the T port (TBYTE_CTL_# = T).

To perform a clr_gate and bs_reset sequence to turn the bus around, do the following:

1. Assert BS_RESET_CTRL.clr_gate through the RIU.
2. Deassert BS_RESET_CTRL.clr_gate through the RIU. The strobe path gating logic is now clear.
3. If PHY_WREN has not already been set to 0, it must be set to 0 before continuing with this step. Assert BS_RESET_CTRL.bs_reset, which resets NIBBLESlices not masked by BS_RST_MASK.bs_reset_mask. While bs_reset is asserted, the TX IOBs of NIBBLESlices not masked by BS_RST_MASK.bs_reset_mask are set to the value of their associated TX_INIT_# attribute. Keep BS_RESET_CTRL.bs_reset asserted for a minimum number of clock cycles based on the TX_DATA_WIDTH and RX_DATA_WIDTH attributes:
 - For data width of 8: 1 CTRL_CLK cycle + 72 PLL_CLK cycles
 - For data width of 4: 1 CTRL_CLK cycle + 40 PLL_CLK cycles
 - For data width of 2: 1 CTRL_CLK cycle + 24 PLL_CLK cycles
4. Deassert BS_RESET_CTRL.bs_reset. After the write to bs_reset is completed, data can be transmitted immediately. For receivers, however, the first FIFO_EMPTY deassertion should be used to know when receiving valid data.



IMPORTANT! *If receiving a strobe and RX_GATING = ENABLE, bitslip is not needed. For all other cases, bitslip is needed for word alignment.*

Related Information

[Controlling Tristate Control](#)

Clocking

Each XPIO bank has two XPLLs. Each XPLL has four user-controlled clock outputs (XPLL.CLKOUT<0-3>) to the programmable logic (PL) and a dedicated, high-speed clock connection (XPLL.CLKOUTPHY) to all XPHY nibbles in an XPIO bank. For more information on XPLLs, see the *Versal ACAP Clocking Resources Architecture Manual (AM003)*. The following tables summarize XPHY clocking ports and attributes. For more complete descriptions, see [Ports](#) and [Attributes](#).

Table 3: XPHY Clocks


Clock	I/O	Description
PLL_CLK	Input	Clocks the XPHY interface
CTRL_CLK	Input	RIU/delay line/BISC clock
FIFO_RD_CLK	Input	The FIFO read clock
CLK_FROM_OTHER_XPHY	Input	Inter-byte clock input
NCLK_NIBBLE_IN	Input	N-clk input for inter-nibble clocking
PCLK_NIBBLE_IN	Input	P-clk input for inter-nibble clocking
FIFO_WR_CLK	Output	The FIFO write clock. Generated internally.
CLK_TO_LOWER	Output	Inter-byte clock output to certain numerically lower nibbles (with one exception to a numerically higher nibble).
CLK_TO_UPPER	Output	Inter-byte clock output to certain numerically higher nibbles.
NCLK_NIBBLE_OUT	Output	N-clk output for inter-nibble clocking
PCLK_NIBBLE_OUT	Output	P-clk output for inter-nibble clocking
Strobe/Capture clock	Input (RX) Output (TX)	Strobes/capture clocks can be received through the IOB to NIBBLESlice[0] or through inter-nibble/inter-byte clocking. Within the XPHY the strobe is separated into a p-clk and n-clk, which then can be used for inter-nibble clocking and data capture. For source-synchronous receive interfaces (implying SERIAL_MODE = FALSE), the strobe/capture clock is received with the data with a known phase relationship. For other receive interfaces (implying SERIAL_MODE = TRUE), the capture clock is generated within the XPHY from PLL_CLK. Inter-nibble and inter-byte clocking aren't supported when SERIAL_MODE = TRUE

Table 4: Clocking Connections

Clock	I/O	Connection (TX)	Connection (RX)
PLL_CLK	Input	XPLL.CLKOUTPHY	XPLL.CLKOUTPHY
CTRL_CLK	Input	Does not need to come from a specific clock source	Does not need to come from a specific clock source.
FIFO_RD_CLK	Input	-	Depends on FIFO_MODE_x. Refer to Controlling FIFO Modes .
CLK_FROM_OTHER_XPHY	Input	-	If receiving an inter-byte clock, connect to the applicable CLK_TO_LOWER or CLK_TO_UPPER of the source nibble sending the inter-byte clock.
PCLK_NIBBLE_IN, NCLK_NIBBLE_IN	Input	-	If receiving inter-nibble clocks, connect PCLK_NIBBLE_OUT of the source nibble to PCLK_NIBBLE_IN of the destination nibble. Do the same for NCLK_NIBBLE_OUT and NCLK_NIBBLE_IN.
FIFO_WR_CLK	Output	-	Generated internally from the strobe or in the case of SERIAL_MODE = TRUE, from PLL_CLK.
CLK_TO_LOWER, CLK_TO_UPPER	Output	-	If sending an inter-byte clock, connect the applicable CLK_TO_LOWER or CLK_TO_UPPER of the source nibble to CLK_FROM_OTHER_XPHY of the destination nibble receiving the inter-byte clock.
PCLK_NIBBLE_OUT, NCLK_NIBBLE_OUT	Output	-	If sending inter-nibble clocks, connect PCLK_NIBBLE_OUT of the source nibble to PCLK_NIBBLE_IN of the destination nibble. Do the same for NCLK_NIBBLE_OUT and NCLK_NIBBLE_IN.
Strobe/Capture clock	Output or input, depending on which perspective- RX (input) or TX (output).	Send through one of the D<0-5> inputs, after which it will be output to the IOB by the corresponding O0[x] bit.	For source-synchronous interfaces, a strobe/capture clock must be received by NIBBLESlice[0] (DATAin[0]), inter-nibble clocking (see Table 3 for the ports), or inter-byte clocking (see Table 3 for the ports). If a strobe/capture clock is received on NIBBLESlice[0], regardless of whether it is single-ended or differential, DELAY_VALUE_0 (and only DELAY_VALUE_0) must be set to 0. If a source-synchronous interface spans multiple nibbles, inter-nibble and/or inter-byte clocking can be used to forward the strobe. If SERIAL_MODE = TRUE, the capture clock is generated from the PLL_CLK input for each nibble in the interface. Inter-nibble and inter-byte clocking are not supported when SERIAL_MODE = TRUE.

Table 5: XPHY Clocking Attributes

Attribute	Description
CONTINUOUS_DQS	Along with RX_GATING, determines if and how the strobe is gated.
DQS_SRC	Determines where the strobe is being received from (NIBBLESLICE[0], inter-nibble clocking, inter-byte clocking, or PLL_CLK for serial mode).
EN_CLK_TO_LOWER	Enables outputting the strobe to certain numerically lower nibbles (inter-byte clocking).
EN_CLK_TO_UPPER	Enables outputting the strobe to certain numerically higher nibbles (inter-byte clocking).
EN_OTHER_NCLK	Enables sourcing the n-side of the strobe from inter-nibble clocking.
EN_OTHER_PCLK	Enables sourcing the p-side of the strobe from inter-nibble clocking.
INV_RXCLK	Inverts the incoming strobe to NIBBLESLICE[0].
REFCLK_FREQUENCY	Set to the frequency of PLL_CLK.
RX_CLK_PHASE_P, RX_CLK_PHASE_N	Controls strobe (p-clk and n-clk in this case) centering for source-synchronous interfaces.
RX_GATING	Along with CONTINUOUS_DQS, determines if and how the strobe is gated.
TX_GATING	Uses PHY_WREN to gate the transmit data and/or outgoing strobe/capture clock. NIBBLESLICE[1] cannot be affected by TX_GATING.
TX_OUTPUT_PHASE_90_<0-5>	Used to center the strobe/capture clock to data on the transmitter side. Applied to the NIBBLESLICE(s) sending the strobe/capture clock, causing it to be center-aligned to the data.

 **IMPORTANT!** *The phase detector in the XPLL should be used for all PL-XPHY TX interfaces. If the interface spans multiple banks, only the phase detector in the XPLL providing the clock to the PL should be used. Refer to Versal ACAP Clocking Resources Architecture Manual (AM003) for more information.*

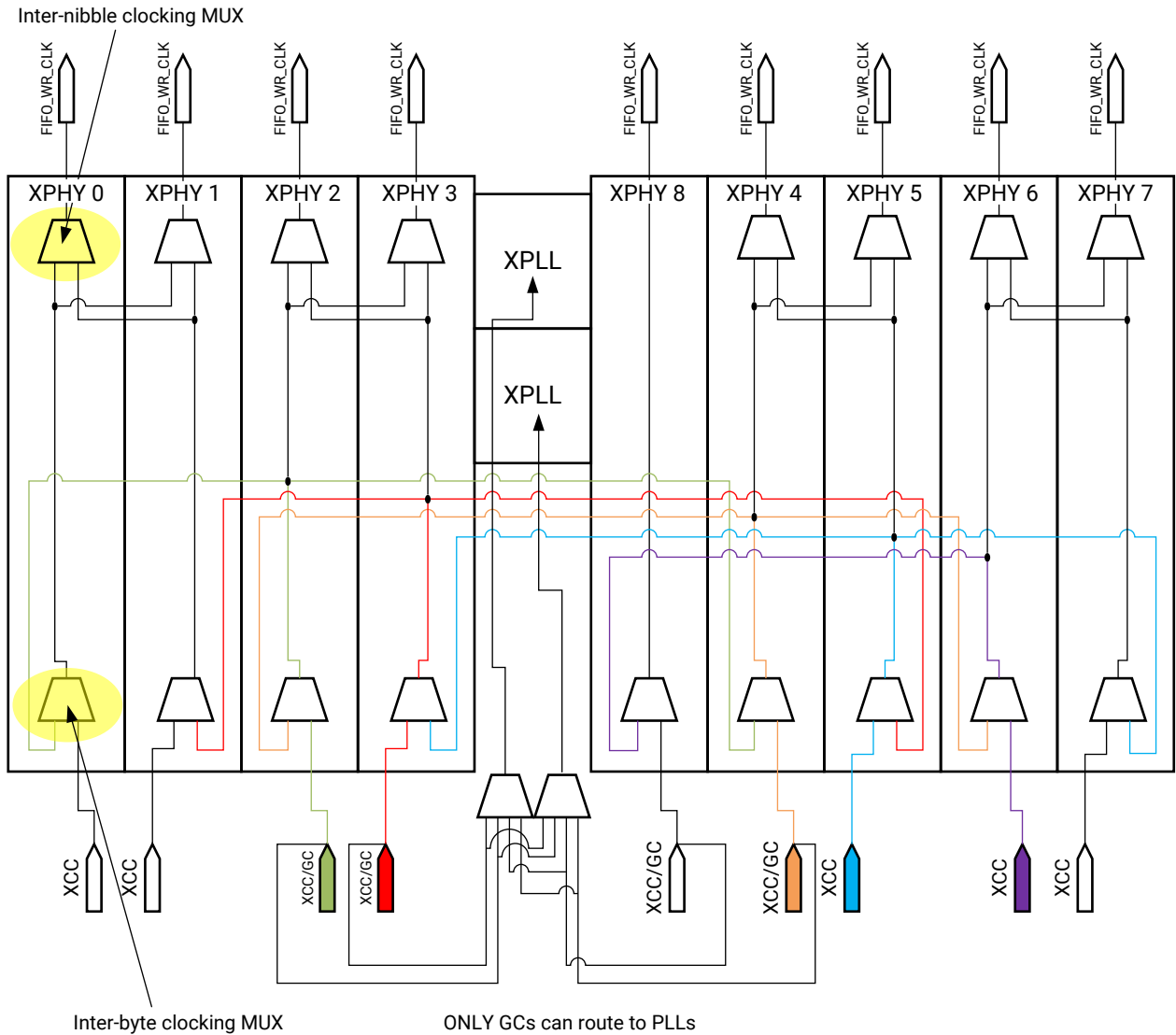
While transmit interface clocking is sourced from XPLLs, receiver clocking for source-synchronous interfaces can take advantage of inter-nibble and inter-byte clocking for forwarding the strobe. If an interface spans more than one nibble, then inter-nibble and/or inter-byte clocking can be used to route the strobe to other nibbles:

- Inter-nibble clocking is supported within nibble-pairs, as shown in the following figure. Nibble-pairs are XPHY nibbles (0,1), (2,3), (4,5), and (6,7)
- Inter-byte clocking is supported between specific XPHY nibbles within an XPIO bank, as shown in the table below. While inter-nibble clocking can only occur between two nibbles, inter-byte clocking can forward a strobe further by using each nibble receiving the inter-byte clock to also forward it. Nibbles receiving a clock through inter-byte clocking can generate an inter-nibble clock, though nibbles receiving an inter-nibble clock cannot use it to generate an inter-byte clock. CLK_FROM_OTHER_XPHY of the source nibble starting the inter-byte clocking should be set to 1'b1.
- XPHY nibble 8 is not part of a nibble-pair and thus is not capable of inter-nibble clocking, but can receive an inter-byte clock from XPHY nibble 6. Use CLK_TO_LOWER when performing inter-byte clocking from XPHY nibble 6 to XPHY nibble 8. This is the exception to the CLK_TO_LOWER/CLK_TO_UPPER naming scheme.
- When SERIAL_MODE = TRUE, inter-nibble/byte clocking is not supported. Instead, each nibble generates its own capture clock from the PLL_CLK input.

- Inter-nibble and inter-byte clocking can only occur between nibbles in the same bank and cannot be connected to the programmable logic (PL).
- Inter-byte clocking is received/sent before the coarse and quarter delays. Thus, an inter-byte clock passes through both delay blocks in the destination nibble, and neither of them in the source nibble.
- Inter-nibble clocking is received/sent after the coarse delay, but before the quarter delay. Thus, an inter-nibble clock passes through the coarse delay in the source nibble, and the quarter delay in the destination nibble.
- NIBBLESLICEs are aligned to each other (assuming BISC is being used), but word alignment is not guaranteed. Using inter-byte clocking can further negatively affect word alignment.

XPLL placement in the following figure is representative, the XPLL locations varies (with respect to nibbles).

Figure 4: Inter-nibble and Inter-byte Clocking Within an XPIO Bank



X21607-102319

The figure above has been translated into a table, as follows.

Table 6: Inter-byte and Inter-nibble Clocking in an XPIO Bank

XPHY Nibble	Can Route To (Through Inter-nibble Clocking)	Can Route To (Through Inter-byte Clocking)	Can Route To (Through Inter-byte, Inter-nibble, or a Combination of the Two)
0	1	-	1
1	0	-	0
2	3	0, 4	0, 1, 3, 4, 5, 6, 7, 8

Table 6: Inter-byte and Inter-nibble Clocking in an XPIO Bank (cont'd)

XPHY Nibble	Can Route To (Through Inter-nibble Clocking)	Can Route To (Through Inter-byte Clocking)	Can Route To (Through Inter-byte, Inter-nibble, or a Combination of the Two)
3	2	1, 5	0, 1, 2, 4, 5, 6, 7
4	5	2, 6	0, 1, 2, 3, 5, 6, 7, 8
5	4	3, 7	0, 1, 2, 3, 4, 6, 7
6	7	8	7, 8
7	6	-	6
8	-	-	-

As shown in the clocking figures, within a bank there are two types of clock inputs that serve two different purposes:

- Global Clock (GC): Clock input with dedicated clock routing designed to have low skew, low duty cycle distortion, and improved jitter resistance. As such, it is recommended for external clocks to enter through GC pins. For interfaces that use XPHY, the GC pins are typically used as the clock source for the XPLLs, which in turn clock the XPHY. GCs can reach all XPLLs in an XPIO bank as well as XPLLs in the adjacent banks.
- XCC: Strobe input for XPHY receive interfaces
- Both GC and XCC: These pins can act as GCs and/or XCCs

Note: If a GC, XCC, or GC/XCC input is not used to receive a clock or strobe, it can be used as a regular I/O pin.

Refer to *Versal ACAP Clocking Resources Architecture Manual* ([AM003](#)) for a more detailed explanation of GC and XCC pins.

The following figure shows the XCC and GC pins that can accept a clock and the NIBBLESLICES with which they are associated. Clocks entering on GC or XCC inputs (as opposed to data entering on those pins), regardless of whether single-ended or differential, must enter NIBBLESLICE[0]. If the clock is differential, the complementary side of the clock can be sent to NIBBLESLICE[1], though it is not routed to the GC or XCC circuitry and instead goes through the normal RX datapath. Recall that because each NIBBLESLICE routes to a specific pin, receiving a differential signal (regardless of whether clock or data) consumes the pins and RX datapaths of both NIBBLESLICES. When NIBBLESLICE[0] receives a clock, it is routed through the GC and/or XCC circuitry (depending on its usage and the capabilities of the pin) and the normal RX datapath.

Figure 5: Relationship of XCC and GC Pins to XPHY NIBBLESLICES

Nibble	0	1	2	3	XPLLs	8	4	5	6	7
NIBBLESlice	0 1 2 3 4 5	0 1 2 3 4 5	0 1 2 3 4 5	0 1 2 3 4 5		0 1 2 3 4 5	0 1 2 3 4 5	0 1 2 3 4 5	0 1 2 3 4 5	0 1 2 3 4 5
	XCC	XCC	XCC	XCC		XCC	XCC	XCC	XCC	XCC
			GC	GC		GC	GC			

X21609-071520

Because there are two XPLLs per bank, one bank can support two data rates. Source-synchronous (implying SERIAL_MODE = FALSE) and SERIAL_MODE = TRUE interfaces can exist in the same bank, but each nibble in its respective interface must be configured as only source-synchronous or only SERIAL_MODE = TRUE.

XPHY Features

Delays

Four types of delays exist within an XPHY. However, only input and output delays can be changed through the programmable logic (PL), using CE, INC, LD, CNTVALUEIN, and RXTX_SEL to do so.

Note: A tap is the smallest amount of delay that a delay line can produce. For information on tap specifications, see *Versal AI Core Series Data Sheet: DC and AC Switching Characteristics (DS957)*.

- **Input delays:** Input delays can delay incoming serialized data up to 512 taps (0–511 taps), with a minimum of 625 ps of available delay. Input delays can be increased to 1024 taps (0–1023 taps) for a minimum of 1250 ps of available delay by cascading the output delay of an XPHY NIBBLESlice to the end of its input delay. For more information on cascading, see the CASCADE_<0–5> attribute.
- **Output delays:** Output delays can delay outgoing serialized data up to 512 taps (0–511 taps), with a minimum of 625 ps of available delay.
- **Coarse (CRSE) delay:** The CRSE delay is only used in low frequency (200 MHz - 1 GHz PLL_CLK), source-synchronous receive interfaces and only applied to the strobe. It cannot be controlled through the PL.
 - Intended for edge-aligned source-synchronous interfaces
 - Enabled through CRSE_DLY_EN
 - Requires SELF_CALIBRATE = ENABLE
- **Quarter (QTR) delays:** QTR delays are applied to the p-clk and n-clk. This is in contrast to CRSE delays, which are applied to the strobe. Cannot be controlled through the PL. For more information on the p-clk and n-clk, see [Clocking](#).
 - Requires SELF_CALIBRATE = ENABLE

Controlling Delays

Input and output delays can be changed through the PL, as shown in the following table.

Table 7: Controlling Input and Output Delays

RX_EN_VTC TX_EN_VTC	LD	CE	INC	Effect on Delay Line
0	0	0	0	Stays the same
	0	0	1	Stays the same
	0	1	0	Decrement by 1 tap
	0	1	1	Increment by 1 tap
	1	0	0	Load value from CNTVALUEIN
	1	0	1	Load value from CNTVALUEIN
	1	1	0	Not supported
	1	1	1	Add value from CNTVALUEIN to the current CNTVALUEOUT value

The following table shows how the delay-related signals are mapped to each NIBBLESLICE.

Table 8: Delay Control Signals NIBBLESLICE Mapping

Port	NIBBLESLIC E[5]	NIBBLESLIC E[4]	NIBBLESLIC E[3]	NIBBLESLIC E[2]	NIBBLESLIC E[1]	NIBBLESLIC E[0]	Description
CE[5:0]	CE[5]	CE[4]	CE[3]	CE[2]	CE[1]	CE[0]	Control signal
INC[5:0]	INC[5]	INC[4]	INC[3]	INC[2]	INC[1]	INC[0]	Control signal
LD[5:0]	LD[5]	LD[4]	LD[3]	LD[2]	LD[1]	LD[0]	Control signal
RXTX_SEL[5:0]	RXTX_SEL[5]	RXTX_SEL[4]	RXTX_SEL[3]	RXTX_SEL[2]	RXTX_SEL[1]	RXTX_SEL[0]	Selects between the input and output delay line to apply delay line updates or report their tap-value through CNTVALUEOUT
CNTVALUEIN [53:0]	CNTVALUEIN [53:45]	CNTVALUEIN [44:36]	CNTVALUEIN [35:27]	CNTVALUEIN [26:18]	CNTVALUEIN [17:9]	CNTVALUEIN [8:0]	Number of taps to be applied to the delay line
CNTVALUEOUT [53:0]	CNTVALUEOUT [53:45]	CNTVALUEOUT [44:36]	CNTVALUEOUT [35:27]	CNTVALUEOUT [26:18]	CNTVALUEOUT [17:9]	CNTVALUEOUT [8:0]	Number of taps currently used by delay line

Table 9: Delay Attributes

Attribute	Description
CASCADE_<0-5>	Doubles the available delay in a NIBBLESLICE by cascading the input and output delays in the NIBBLESLICE. Only applicable for RX.
CRSE_DLY_EN	Enables coarse delays
DELAY_VALUE_<0-5>	Sets the initial delay for the input and output delays in a NIBBLESLICE. If CASCADE_x = TRUE, the max delay available in DELAY_VALUE_x doubles.


DELAY_VALUE_x sets the initial delay (in ps) of both the input and output delay in NIBBLESLICE[x]. While DELAY_VALUE_x is set in terms of time, the delay is ultimately applied to the delay lines in terms of taps. This makes DELAY_VALUE_x unique in that it is the only opportunity to set a time value through an attribute for any of the delay lines.


When changing the value of input or output delays, consider the following:


- DLY_RDY must be 1.
- To avoid glitches, input and output delays can only be changed once every three CTRL_CLK cycles.
- Input and output delay changes take effect one CTRL_CLK cycle after being reflected in CNTVALUEOUT.
- Delays, regardless of the type, are always manifested in terms of taps.
- If a nonzero DELAY_VALUE_x is set, the following equation can be used to estimate the number of taps required for a new time-based value for an input or output delay. Delay_old is the previous delay in terms of time (ps) whereas delay_new is the desired delay in terms of time (ps). If DELAY_VALUE_x were set to zero, this equation would not be valid. While CNTVALUEIN can still be used to load taps, the approximate time value of each tap will not be known and thus cannot be used to calculate a new time delay value. Align_delay is used by BISC to compensate for the internal skew between clock and data insertion delays of input paths to the first capture flip-flops. More information on align_delay can be found beneath the following waveforms and in [Built-in Self-Calibration](#).

$$\text{CNTVALUEIN}[\text{NIBBLESLICE}[x]] = \text{delay_new} * ((\text{CNTVALUEOUT}[\text{NIBBLESLICE}[x]] - \text{align_delay}) / \text{delay_old})$$

- Updating input and output delays through the register interface unit (RIU) takes one additional CTRL_CLK cycle compared to updating delays through the PL. When updating input or output delays through the RIU, RX_EN_VTC and TX_EN_VTC must be set to 1, LD must be set to 1, CE must be set to 0, and INC is a don't care. Note that only input and output delays can be updated through the PL.
- If TBYTE_CTRL_# = PHY_WREN, the tristate NIBBLESLICE is capable of applying a delay to the tristate signal. To change the amount of delay applied in the tristate NIBBLESLICE, use the TRISTATE_ODLY register within the RIU.

 **IMPORTANT!** *Jumps greater than 8 taps might result in data glitching.*

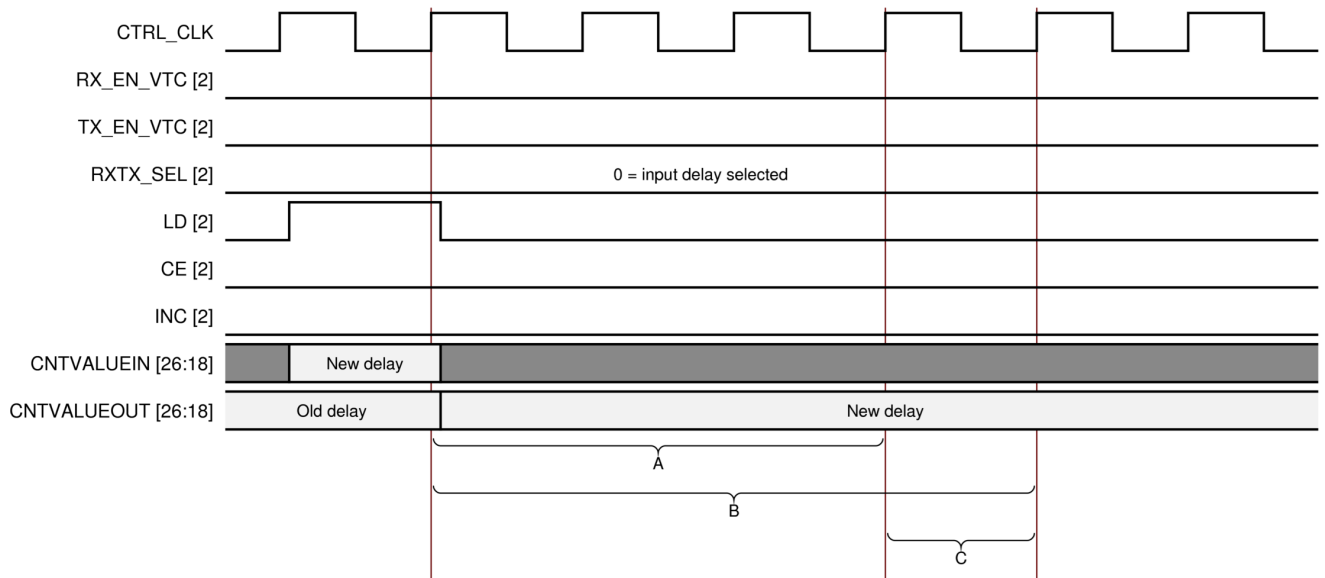
 **IMPORTANT!** *If voltage and temperature compensation (VTC) is desired on an output delay, the input delay in the same NIBBLESLICE as the output delay must be set to the same tap value as the output delay.*

 **IMPORTANT!** *Bit alignment is not guaranteed on output delays if the delays are equal to or greater than 1.5 UI.*

 **IMPORTANT!** *For interfaces with REFCLK_FREQUENCY below 500 MHz, DELAY_VALUE_<0-5> is not supported.*

The following waveform shows how to update a single input or output delay. The waveform updates NIBBLESlice[2] but applies to any NIBBLESlice (note that the bus for each signal, other than CTRL_CLK, would change too).

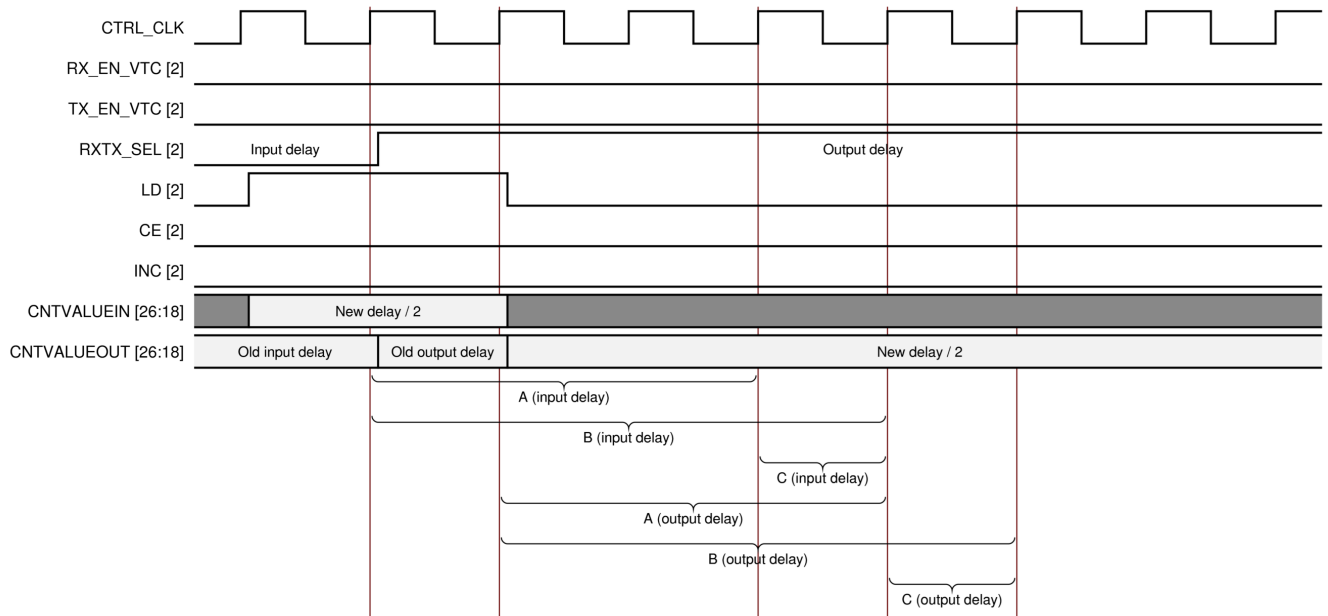
Figure 6: Updating a Single Input or Output Delay Line



- **Start:** The goal of this waveform is to update the input delay, implying $RXTX_SEL[2] = 0$, through $CNTVALUEIN[26:18]$. $\{LD[2], CE[2], INC[2]\} = 10x$ tells the delay line to use $CNTVALUEIN$ to update the delay line, which is the starting point of this waveform.
- **A:** A three-cycle wait is required between updates to the same delay line (in this case, the input delay of $NIBBLESlice[2]$) to prevent glitching.
- **B:** The reported delay might not match the actual delay being applied. This only affects a delay line that is changed, so the other delay lines in the nibble are still reported accurately.
- **C:** The three-cycle wait from A has been met, so from C onward the input delay (of $NIBBLESlice[2]$) can be changed without glitching. Note that B overlaps with C for one cycle.
- **After C:** The input delay (of $NIBBLESlice[2]$) is now reported accurately, and the delay line update is considered complete.

The following waveform shows how to update cascaded delays. The waveform updates $NIBBLESlice[2]$, but applies to any $NIBBLESlice$ (note that the bus for each signal, other than $CTRL_CLK$, would change too).

Figure 7: Updating a Cascaded Delay Line



- Start:** The goal of this waveform is to update a cascaded delay line, which is composed of the input delay and output delay of a NIBBLESlice. To update a cascaded delay line, update both the input and output delays that compose the cascaded delay. Each delay line should be loaded with half of the total desired delay (New delay/2). In this case, the input delay (RXTX_SEL[2] = 0) is updated first, followed by the output delay (RXTX_SEL[2] = 1). Like the waveform in Figure 6, this waveform updates the delay through CNTVALUEIN[26:18]. {LD[2], CE[2], INC[2]} = 10x tells the delay line to use CNTVALUEIN to update the delay line, which is the starting point of this waveform.
- A (input delay/output delay):** A three-cycle wait is required between updates to the same delay line (in this case the <input/output> delay of NIBBLESlice[2]) to prevent glitching.
- B (input delay/output delay):** The reported <input/output> delay might not match the actual delay being applied. This only affects a delay line that is changed, so the other delay lines in the nibble are still reported accurately. Note, however, the three cycles shared between A (input delay) and B (output delay) where both the input and output delays of NIBBLESlice[2] can report a delay that doesn't match the actual delay being applied.
- C (input delay/output delay):** The three-cycle wait from A has been met, so from C onward, the <input/output> delay (of NIBBLESlice[2]) can be changed without glitching.
- After C (input delay):** The input delay (of NIBBLESlice[2]) is now reported accurately, and its delay update is complete. However, the output delay is not finished updating yet.
- After C (output delay):** The input and output delays (of NIBBLESlice[2]) are now reported accurately, and the cascaded delay line update is considered complete.

Align_delay, the result of the first step of BISC, can be estimated by setting DELAY_VALUE_x to zero. Reading CNTVALUEOUT of the input delay (RXTX_SEL[x] = 0) reports align_delay for NIBBLESLICE[x]. This can be used as an estimate of align_delay for other NIBBLESLICES within the nibble. A few other points on align_delay:

- It does not appear in simulation.
- It is calculated on a per-NIBBLESLICE basis.
- It is calculated once and does not change unless the XPHY is reset. Updating an input delay does not change the value of align_delay.
- It only exists for input delays and can be considered zero for output delays or when BISC is not used (SELF_CALIBRATE = DISABLE).
- When an input delay is reported through CNTVALUEOUT, it includes align_delay. For example, if DELAY_VALUE_x ends up being 100 taps and align_delay is 10 taps, CNTVALUEOUT would report 110 taps for that NIBBLESLICE.
- It is not compensated for by VTC because align_delay tracks the strobe path. As VT conditions change, the strobe propagation delay also changes.
- If VTC is used, only the taps above align_delay and below the mark set by DELAY_VALUE_# are compensated for. For example, if DELAY_VALUE_x ends up being 100 taps and align_delay is 10 taps, only 90 taps in the input delay of that NIBBLESLICE are compensated for by VTC. CNTVALUEOUT would still report 110 taps, but 20 of those taps would not be compensated for by VTC (the 10 taps from align_delay and the 10 taps above the 100 tap mark set by DELAY_VALUE# would not be compensated for by VTC). A visual representation of this example can be found in the following figure.
- VTC will not compensate below align_delay. For example, if align_delay is 10 taps, VTC will never compensate for less than 10 taps. In this sense, align_delay acts as a "floor" for VTC.
- In some cases, particularly full-bank designs, align_delay might reach its ceiling and not properly operate. To remedy this, have the strobe/capture clock enter on XPHY 4.


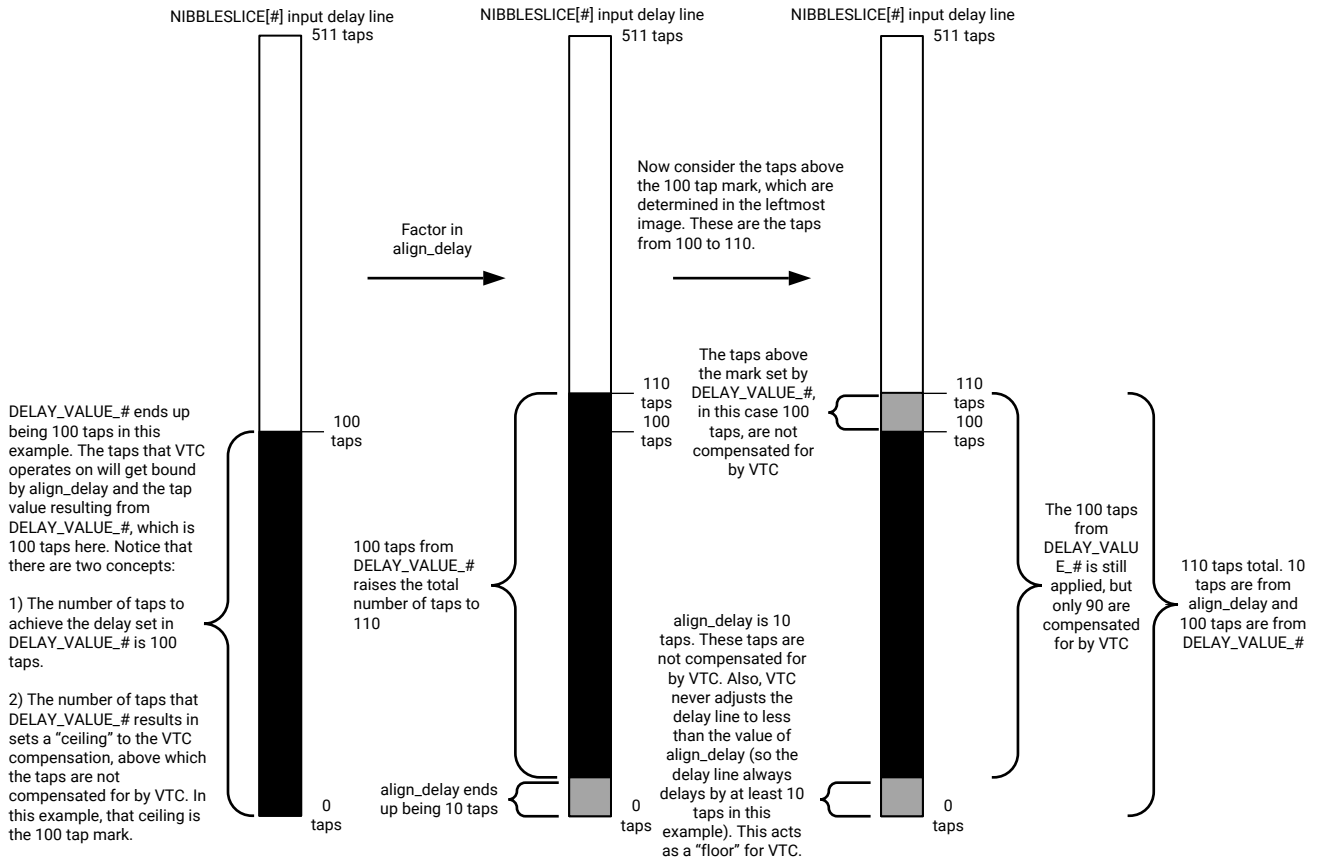
 **IMPORTANT!** *If all XPHY nibbles in a bank comprise an interface (a "full-bank design"), align_delay can be as large as 300 taps. Because align_delay is stored in the input delay of each NIBBLESLICE, align_delay reduces the available delay of input delays.*

Figure 8: Align Delay Value VTC Relationship



X24827-111620

To find the approximate time value (ps) of a single tap for a given nibble using one NIBBLESlice:

1. Set DELAY_VALUE_x to a nonzero value.
2. CNTVALUEOUT of the output delay (RXTX_SEL[x] = 1) of NIBBLESlice[x] will only be the delay implemented by DELAY_VALUE_x. The approximate value of a single tap (ps/tap) for that nibble is then:

$$\text{Time value of a single tap} = \text{DELAY_VALUE_x} / \text{Output delay value of NIBBLESlice[x]}$$

Another way to find the approximate time value (ps) of a single tap for a given nibble is:

1. Set DELAY_VALUE_a to zero and DELAY_VALUE_b to a nonzero value. CNTVALUEOUT of the input delays (RXTX_SEL[a,b] = 0) will be:
 - CNTVALUEOUT[NIBBLESlice[a]] = align_delay
 - CNTVALUEOUT[NIBBLESlice[b]] = DELAY_VALUE_b + align_delay
2. The approximate value of a single tap (ps/tap) for that nibble is then:

$$\text{Time value of a single tap} = (\text{CNTVALUEOUT}[\text{NIBBLESlice}[b]] - \text{CNTVALUEOUT}[\text{NIBBLESlice}[a]]) / \text{DELAY_VALUE_b}$$

Cascaded delays, implying `CASCADE_x = TRUE`, render the TX datapath of `NIBBLESlice[x]` inoperable. When using cascaded delays, consider the following:

- Xilinx recommends storing half of the total delay in the input delay and the other half in the output delay.
- The insertion delay between clock and data is not fully compensated for in `NIBBLESlices` with `CASCADE_x = TRUE`. This typically results in a ~65 ps difference between the clock and data. To account for this, take ~65 ps/2 and add that value to both the p-quarter and n-quarter delays.

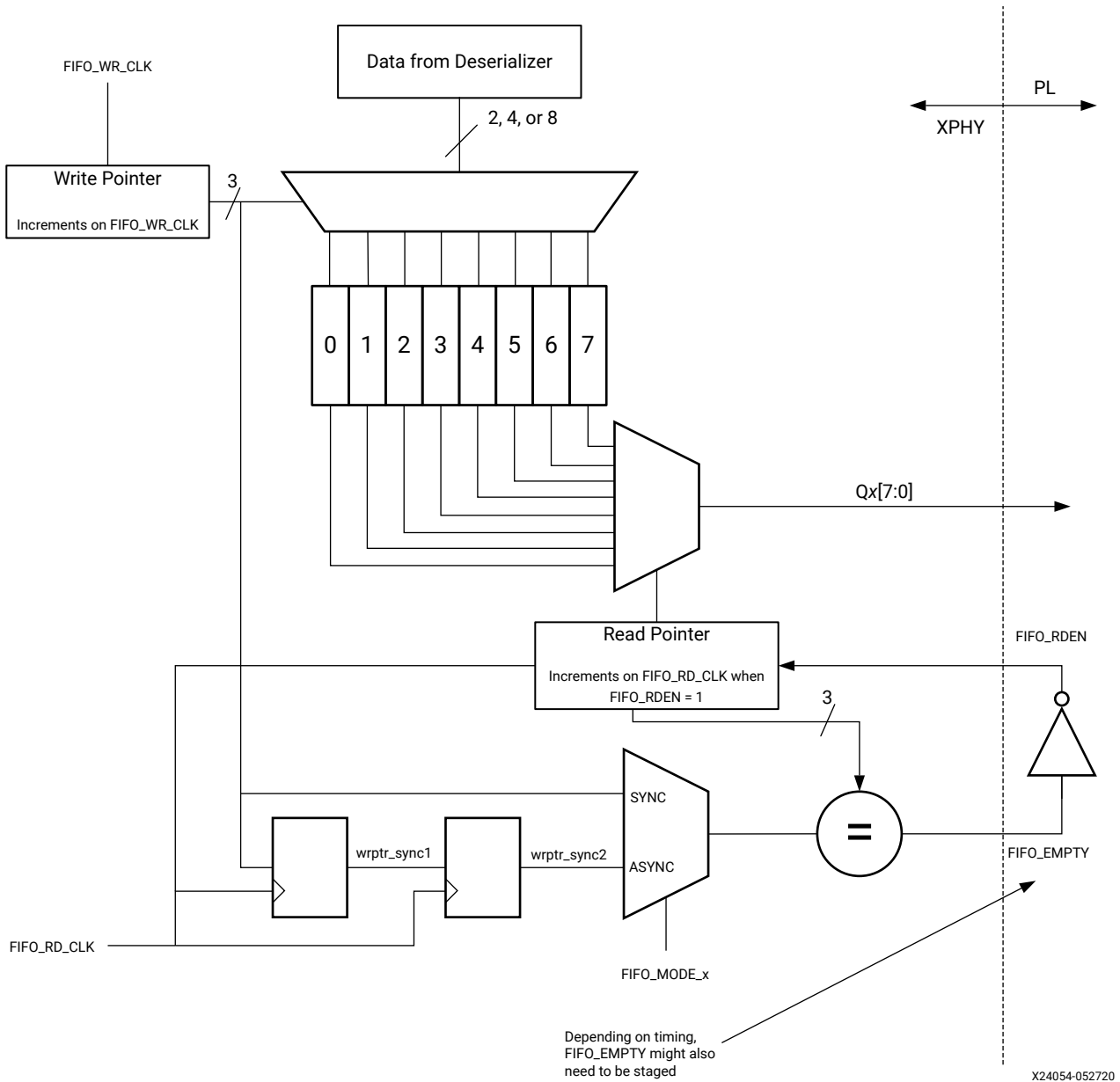
FIFO

This section refers to the FIFO in the RX datapath. The FIFO only supports `ASYNC` mode:

- **FIFO_MODE_x = ASYNC:** The read and write clocks of the FIFO in `NIBBLESlice[x]` share the same frequency, but can be phase independent.

The following figure is representative of the FIFO operation.

Figure 9: FIFO in RX Datapath



Controlling FIFO Modes

FIFO modes are controlled on a per-NIBBLESlice basis, as determined by the FIFO_MODE_<0-5> attribute.

When the write and read pointers equal each other, FIFO_EMPTY is asserted. This is the starting point for the following ASYNC latency waveforms. If the read pointer is locked at one location (for example, FIFO_RDEN is deasserted) but the write pointer is still incrementing, after eight FIFO_WR_CLK cycles the write pointer wraps around to the read pointer location. When this occurs, new data appears at the Q<0-5> pin and FIFO_EMPTY is asserted again.



RECOMMENDED: Do not rely on FIFO_EMPTY asserting every eight FIFO_WR_CLK cycles for bit and word alignment. The first deassertion of FIFO_EMPTY should be used for controlling FIFO_RDEN.



IMPORTANT! Do not register FIFO_EMPTY as part of the FIFO_RDEN = !FIFO_EMPTY connection when receiving a strobe (as opposed to a capture clock), regardless of if the flop is located in the BLI or PL. See [Figure 9](#).

The following table describes FIFO-related attributes in a simplified way. For a complete description, refer to [Attributes](#).

Table 10: FIFO-Related Attributes

Attribute	Description
FIFO_MODE_<0-5>	Determines the clocking topology of the FIFOs within the RX datapath.
RX_DATA_WIDTH	Determines the deserialization for the RX datapath, which affects the DATAIN to Qx mapping.

Mapping of DATAIN to Qx is shown in the following table, where DATAIN[x] corresponds to NIBBLESlice[x]. Refer to the latency waveforms below for the context of P0, N0, ..., P3, N3.

Table 11: DATAIN[x] to Qx Mapping

Deserialization (RX_DATA_WIDTH)	DATAIN[x]							
	N3	P3	N2	P2	N1	P1	N0	P0
1:8	Qx[3]	Qx[7]	Qx[2]	Qx[6]	Qx[1]	Qx[5]	Qx[0]	Qx[4]
1:4	-	-	-	-	Qx[3]	Qx[7]	Qx[2]	Qx[6]

Table 11: DATAIN[x] to Qx Mapping (cont'd)

Deserialization (RX_DATA_WIDTH)	DATAIN[x]							
	N3	P3	N2	P2	N1	P1	N0	P0
1:2	-	-	-	-	-	-	Qx[3]	Qx[7]

When FIFO_MODE_x = ASYNC:

- FIFO_EMPTY can take two to three cycles to change values
- The two to three cycle latency ensures that the write pointer will always be two to three locations ahead of the read pointer during operation
- FIFO_RD_CLK must be the same frequency as FIFO_WR_CLK, but is phase independent
- FIFO_EMPTY and Qx are in the FIFO_RD_CLK domain

DATAIN and Q in the following figures refer to a single NIBBLESlice. DATAIN[x] maps to one of the Q_x outputs. The waveforms show NIBBLESlice[2], but applies to any NIBBLESlice.

Figure 10: 8-bit RX Datapath Latency when FIFO_MODE_x = ASYNC

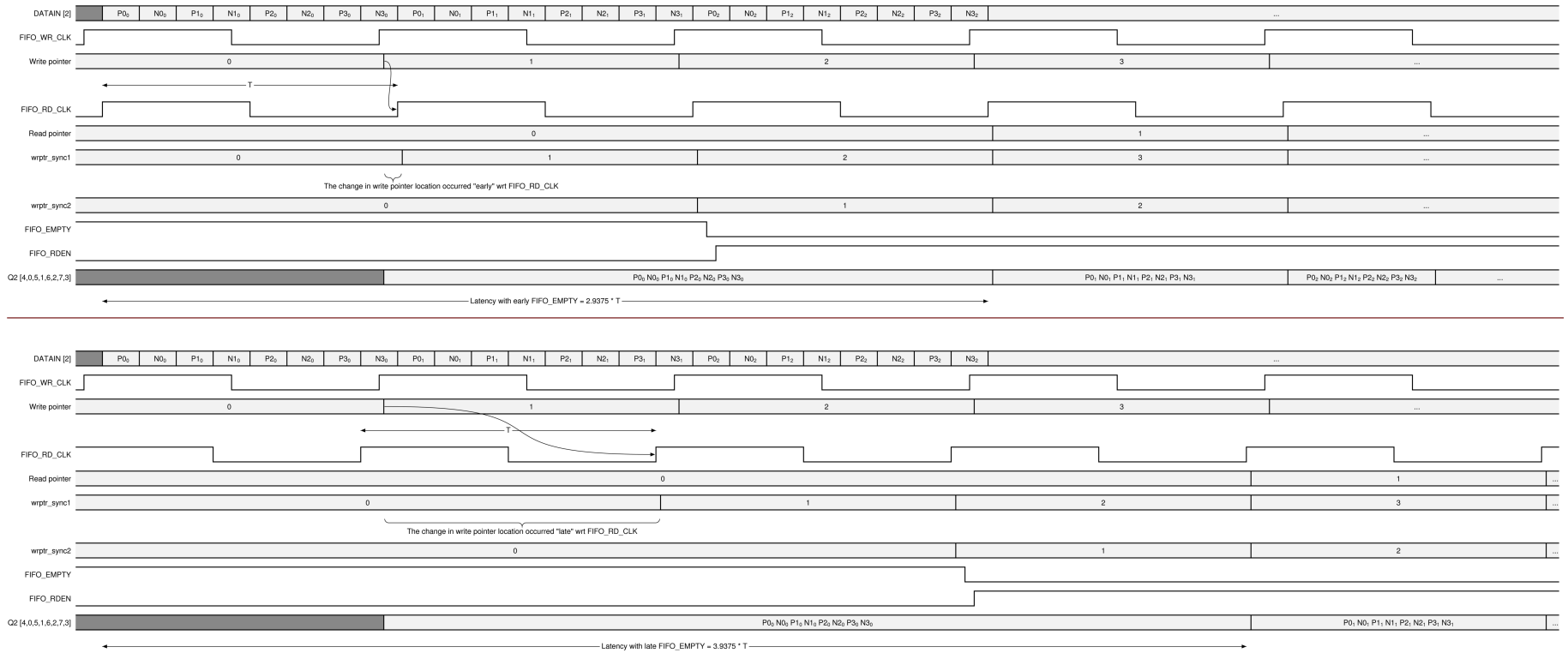


Figure 11: 4-bit RX Datapath Latency when FIFO_MODE_x = ASYNC

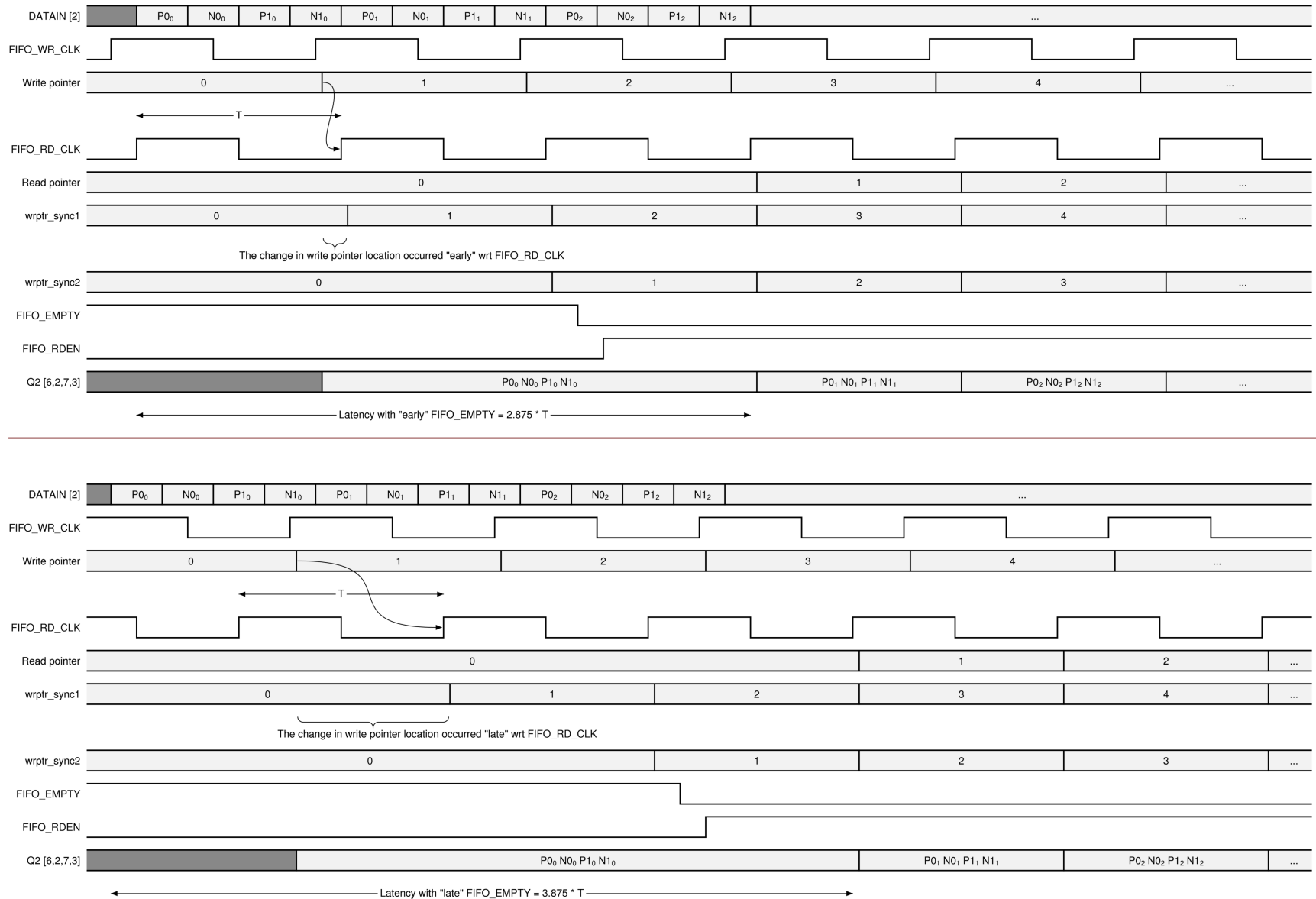
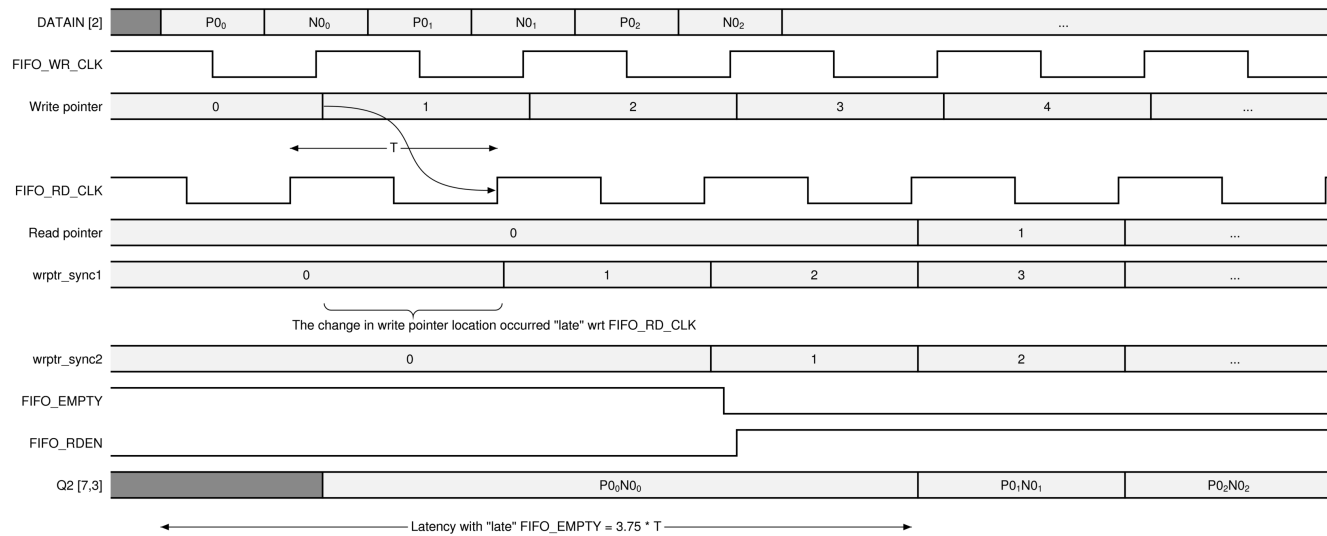
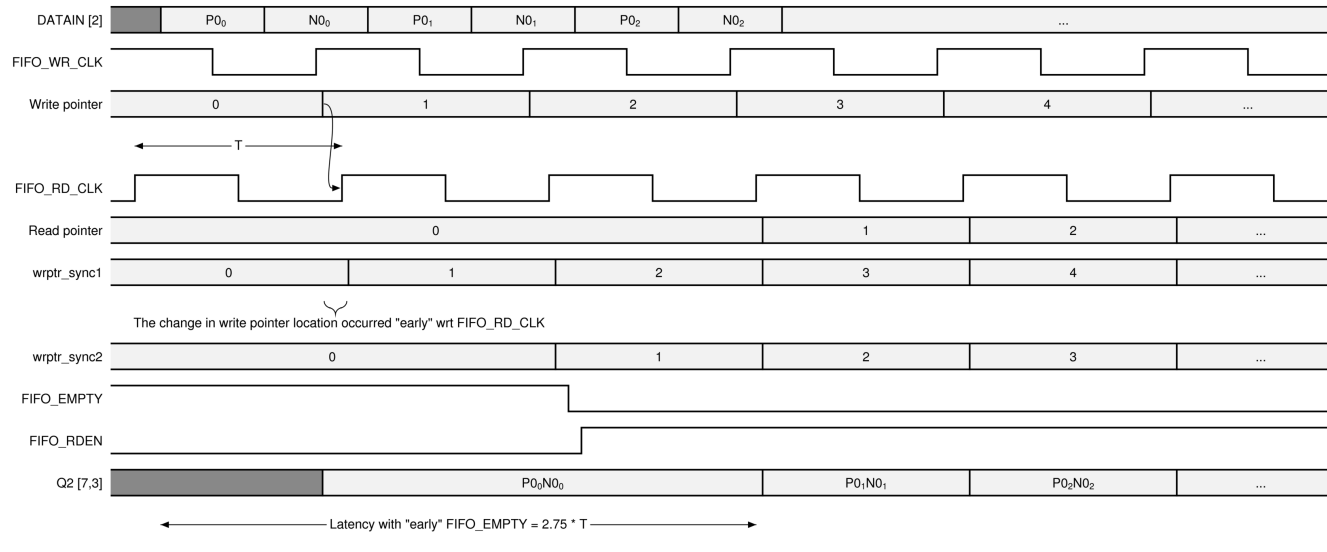


Figure 12: 2-bit RX Datapath Latency when FIFO_MODE_x = ASYNC



Tristate Control

The XPHY can control buffers with tristate capability. The tristating can be performed on a per-NIBBLESlice basis, as determined through the `TBYTE_CTL_<0-5>` attribute.

Controlling Tristate Control

Buffers with tristate capability can be controlled through the XPHY on a per-NIBBLESlice basis, as determined through the `TBYTE_CTL_<0-5>` attribute. The `<0-5>` suffix of `TBYTE_CTL_<0-5>` corresponds to the NIBBLESlice it is applied to. So `TBYTE_CTL_0` is the tristate control setting for NIBBLESlice[0], `TBYTE_CTL_1` corresponds to NIBBLESlice[1], and so on.

- **`TBYTE_CTL_x = T`:** Uses the `T[x]` input of the XPHY to drive the tristate control signal to the IOB of NIBBLESlice[x]. This is a combinatorial path from the PL and thus is not aligned to TX data. When `TX_DATA_WIDTH = 2`, this is the only `TBYTE_CTL_x` setting supported.
- **`TBYTE_CTL_x = PHY_WREN`:** Inverts and serializes the `PHY_WREN` input of the XPHY to drive (broadcast) the tristate control signal to the IOB of each NIBBLESlice. Each bit of `PHY_WREN` acts as the tristate control signal for two UIs worth of data. The serialized and inverted `PHY_WREN` signal is aligned to the serialized output of the TX datapath, `00`. `PHY_WREN` cannot be used when `TX_DATA_WIDTH = 2`.

`TBYTE_CTL_x` determines which signal, `T` or `PHY_WREN`, is accepted by NIBBLESlice[x]. For example, if NIBBLESlice[0] receives both a `PHY_WREN` and `T` stimulus, only the one matching `TBYTE_CTL_0` is accepted. `T_OUT[x]` is then the output to the IOB.

The latency through the TX datapath is shown for `TBYTE_CTL_x = PHY_WREN`. `PHY_WREN` takes one cycle longer than the data to propagate through the XPHY. Due to this, `PHY_WREN` should be applied one cycle before TX data is presented to the XPHY from the PL.

Independent of tristating, each bit of `OO` maps to one of the `D<0-5>[7:0]` inputs. Generalized, this means `Dx` maps to `OO[x]`. The following tables shows how `Dx` maps to `OO[x]` for different data widths. Refer to the latency waveforms below for the context of `P0`, `N0`, ..., `P3`, `N3`.

Table 12: `Dx` to `OO[x]` Mapping

Serialization (<code>TX_DATA_WIDTH</code>)	<code>D_x</code>							
	<code>Dx[7]</code>	<code>Dx[6]</code>	<code>Dx[5]</code>	<code>Dx[4]</code>	<code>Dx[3]</code>	<code>Dx[2]</code>	<code>Dx[1]</code>	<code>Dx[0]</code>
8:1	The eighth bit serialized and transmitted through <code>OO[x]</code>	The seventh bit serialized and transmitted through <code>OO[x]</code>	The sixth bit serialized and transmitted through <code>OO[x]</code>	The fifth bit serialized and transmitted through <code>OO[x]</code>	The fourth bit serialized and transmitted through <code>OO[x]</code>	The third bit serialized and transmitted through <code>OO[x]</code>	The second bit serialized and transmitted through <code>OO[x]</code>	The first bit serialized and transmitted through <code>OO[x]</code>

Table 12: D_x to $O0[x]$ Mapping (cont'd)

Serialization (TX_DATA_WIDTH)	D_x							
	$D_x[7]$	$D_x[6]$	$D_x[5]$	$D_x[4]$	$D_x[3]$	$D_x[2]$	$D_x[1]$	$D_x[0]$
4:1	-	-	-	-	The fourth bit serialized and transmitted through $O0[x]$	The third bit serialized and transmitted through $O0[x]$	The second bit serialized and transmitted through $O0[x]$	The first bit serialized and transmitted through $O0[x]$
2:1	-	-	-	The second bit serialized and transmitted through $O0[x]$	-	-	-	The first bit serialized and transmitted through $O0[x]$

D_x and $O0$ in the following figures refer to a single NIBBLESlice. D_x maps to one of the $O0[x]$ outputs.

Figure 13: 8-Bit TX Latency with Tristate Control (TBYTE_CTL_x = PHY_WREN)

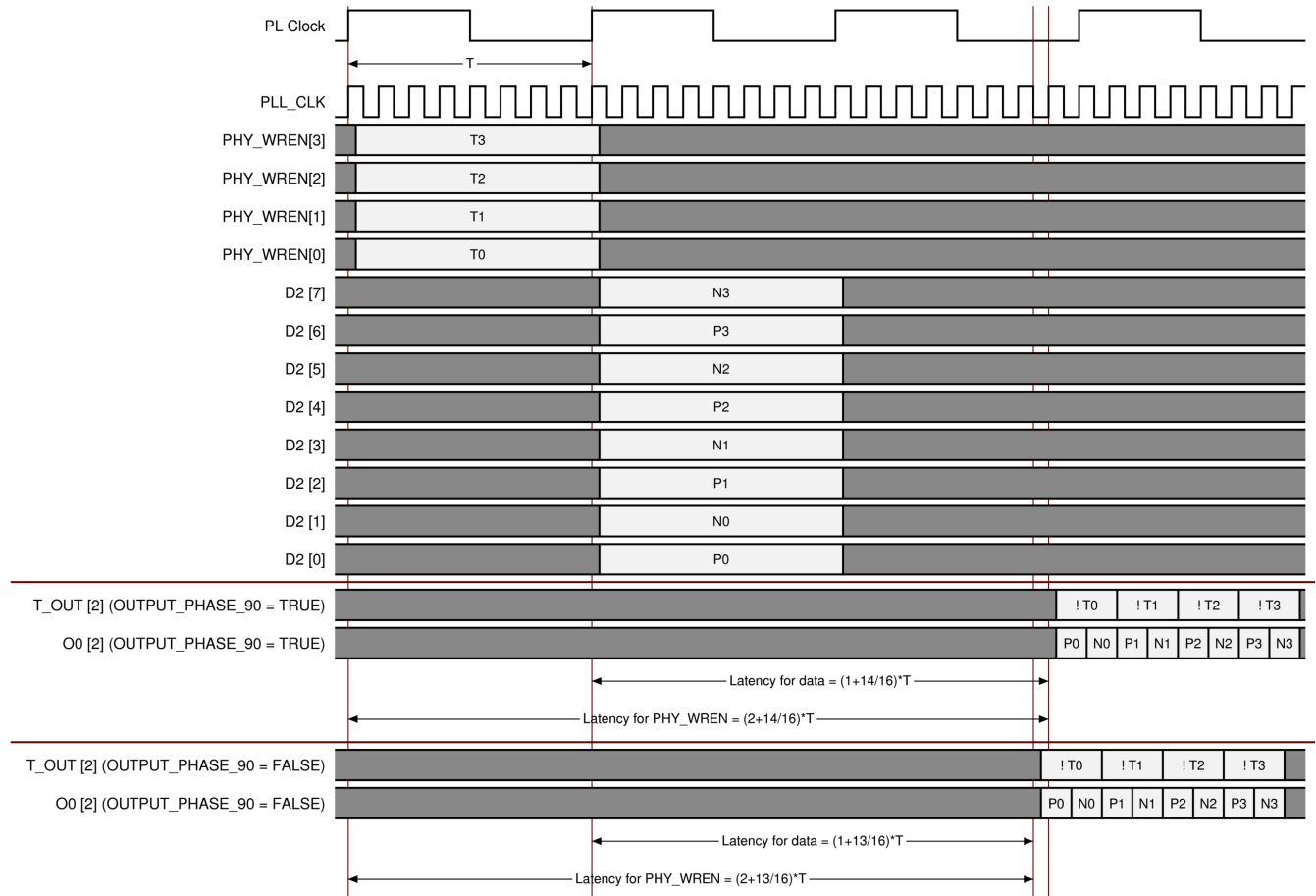
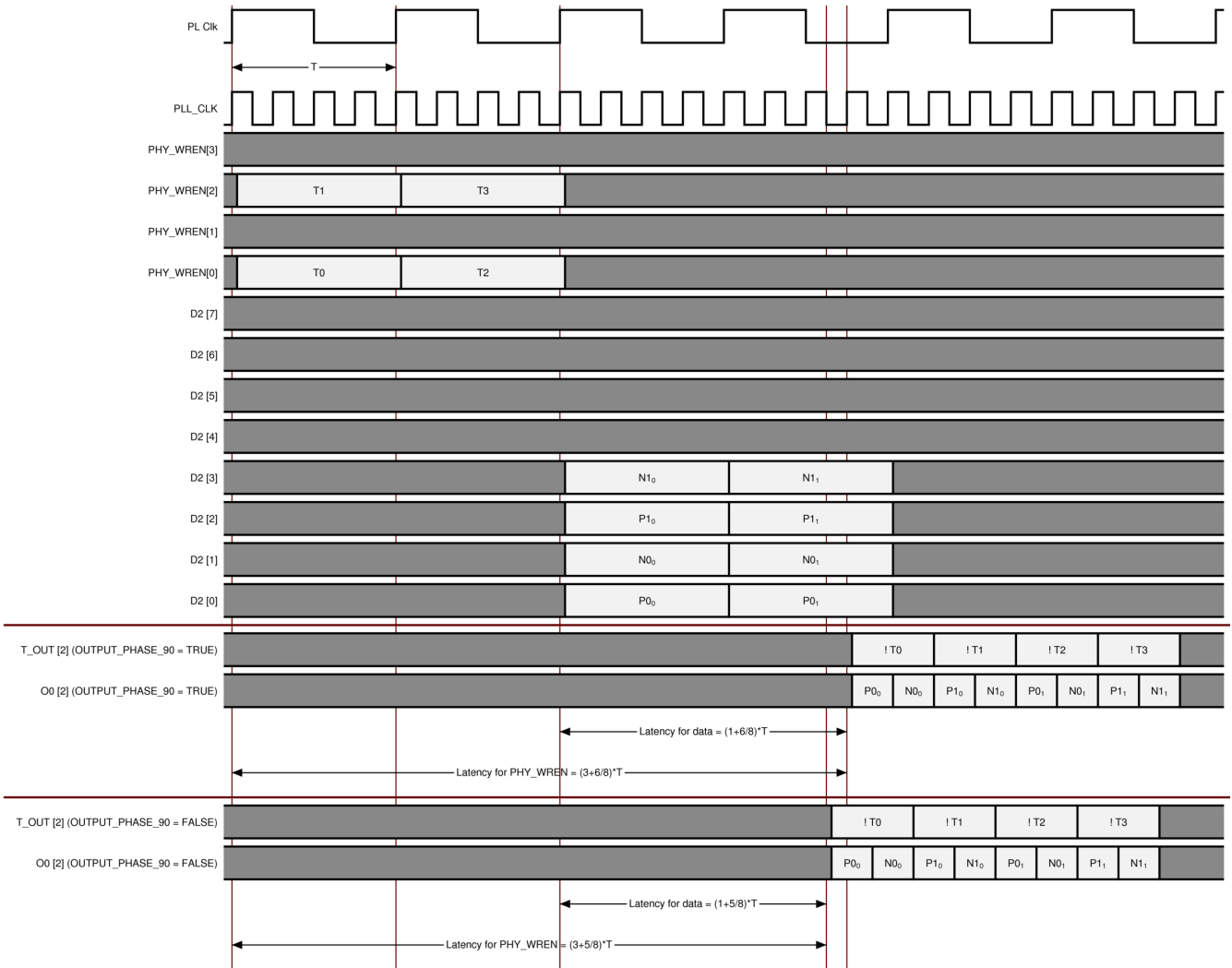


Figure 14: 4-Bit TX Latency with Tristate Control (TBYTE_CTL_x = PHY_WREN)


The latency through the TX datapath when NOT using tristate control or when TBYTE_CTL_x = T is shown in the following figures.

Figure 15: 8-Bit TX Latency

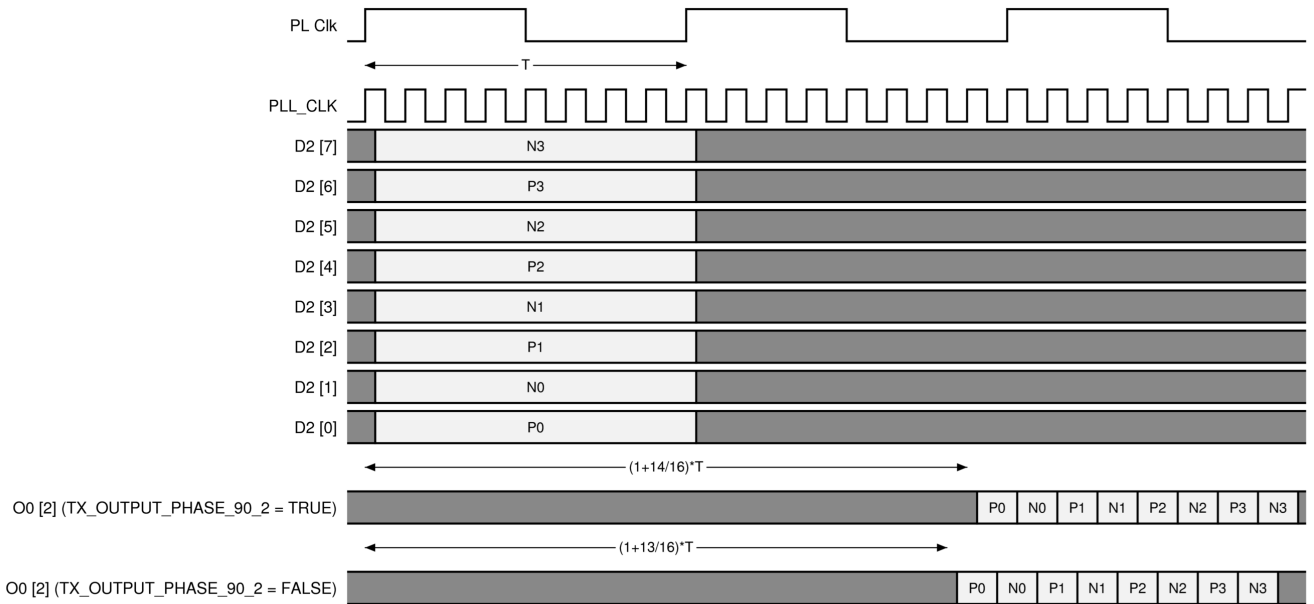


Figure 16: 4-Bit TX Latency

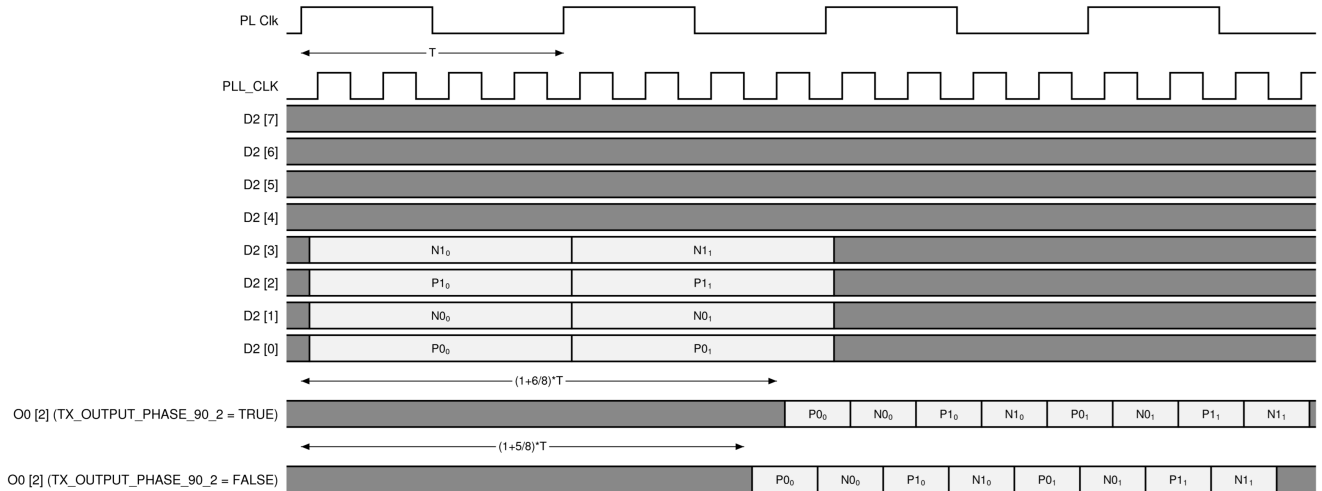
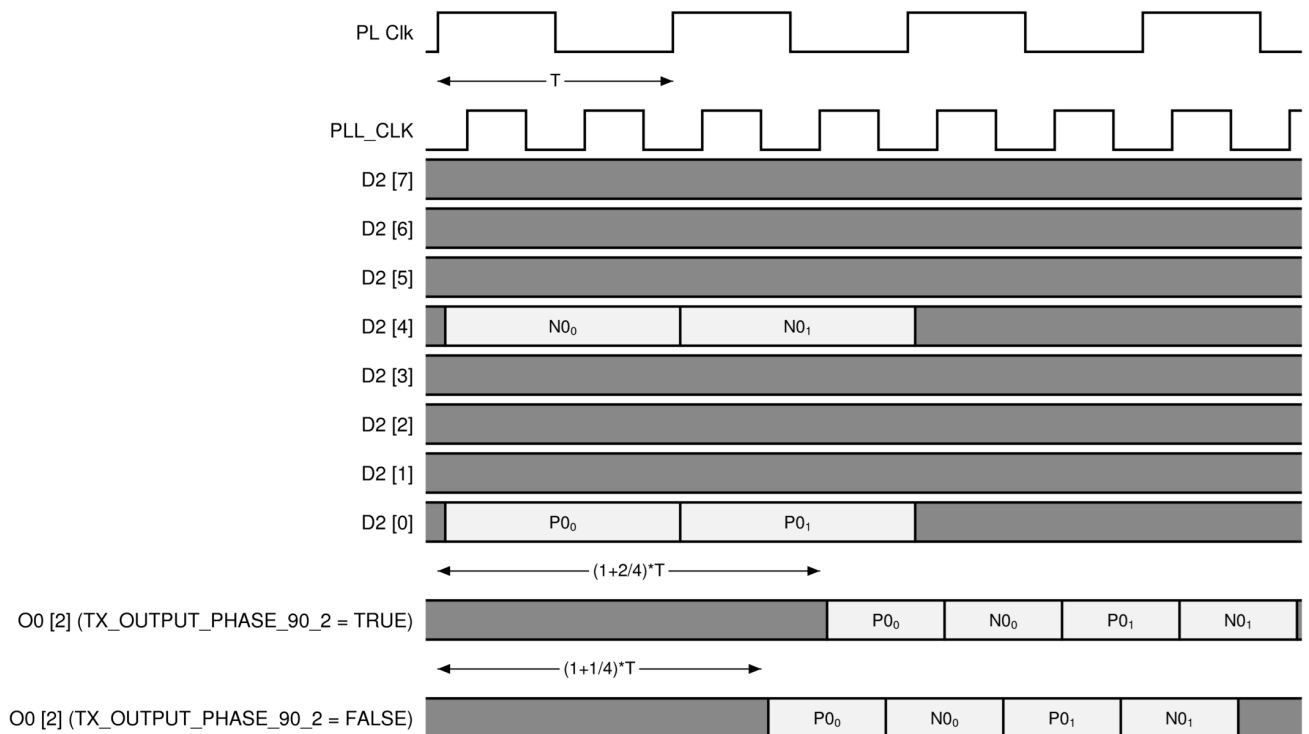


Figure 17: 2-Bit TX Latency



Built-in Self-Calibration

Built-in self-calibration (BISC) calculates delay values for all delay lines. The last step of BISC, voltage and temperature compensation (VTC), automatically tracks and adjusts the delays as voltage and temperature conditions change.

BISC is a three step process:

- Alignment (RX Datapath Only):** Alignment maximizes the data eye by removing internal on-die skew between data inputs and compensating for the internal skew between clock and data insertion delays of input paths to the first capture flip-flops (this includes skew caused by inter-nibble and/or inter-byte clocking). These delays, as a singular group, are called *align_delay*.



IMPORTANT! BISC does not compensate for external delays, such as differences in trace lengths and package skews, or deskewing outputs signals.

- Delay Calibration:** Calculates the taps required for delay lines. For each XPHY NIBBLESlice, delay calibration calculates the number of taps required to provide the delay requested using the `DELAY_VALUE_x` attribute for input and output delays, and 90° shifts for QTR delays, for a given process, voltage, and temperature condition. *align_delay* is not included in any calculations.

- Voltage and Temperature Compensation:** Compensates for voltage and temperature changes. Voltage and temperature compensation (VTC) uses round-robin scheduling to automatically update delays lines based on voltage and temperature drift without interrupting normal operation of the associated XPHY NIBBLESlice.



TIP: When using inter-byte or inter-nibble clocking, each nibble can require a different amount of time to complete the same BISC step. In this case, the next BISC step cannot be started until all nibbles finish the current step.

Controlling Built-in Self-Calibration

The only attribute required to run BISC is `SELF_CALIBRATE = ENABLE`. The following table shows BISC-related attributes and how they are overridden if BISC is not used (`SELF_CALIBRATE = DISABLE`). For more complete attribute descriptions, see [Attributes](#).

Table 13: BISC-Related Attributes

Attribute	Description	Effect From <code>SELF_CALIBRATE = DISABLE</code>
<code>CRSE_DLY_EN</code>	Enables CRSE delays	Coarse delays are not used.
<code>DELAY_VALUE_<0-5></code>	Sets the initial input and output delay line value in each NIBBLESlice.	<code>DELAY_VALUE_<0-5></code> loads a zero delay to input and output delays. However, delays can still be loaded from the PL.
<code>DIS_IDLY_VT_TRACK</code>	Disables VTC on input delays	Disables VTC on input delays
<code>DIS_ODLY_VT_TRACK</code>	Disables VTC on output delays	Disables VTC on output delays
<code>DIS_QDLY_VT_TRACK</code>	Disables VTC on QTR delays	Disables VTC on QTR delays
<code>RX_CLK_PHASE_N</code> , <code>RX_CLK_PHASE_P</code>	Controls strobe (p-clk and n-clk in this case) centering for source-synchronous interfaces	<code>RX_CLK_PHASE_N</code> and <code>RX_CLK_PHASE_P</code> cannot be set to <code>SHIFT_90</code>

The following table shows how to control BISC.

Table 14: Controlling BISC Steps Summary

BISC Step	Controlled by	Other Considerations	Common to All
Alignment	<ul style="list-style-type: none"> Assert RX_EN_VTC and TX_EN_VTC during reset to perform alignment. Refer to Reset Sequence. 	<ul style="list-style-type: none"> Alignment is only performed once upon completion of the reset sequence. To re-perform alignment, reset the XPHY. 	<ul style="list-style-type: none"> SELF_CALIBRATE must be set to TRUE for any BISC steps to occur BISC is considered completed when: <ul style="list-style-type: none"> If the interface is using all steps of BISC, when PHY_RDY asserts If the interface is using BISC without VTC, when DLY_RDY asserts If the interface is not using BISC, then DLY_RDY asserting indicates that delays can be changed, but not that alignment and delay calibration are complete. Because BISC is not used in this scenario, it never starts or completes. When simulating, some of the BISC control ports (BISC_START_IN, BISC_STOP_IN, BISC_START_OUT, and BISC_STOP_OUT) must be daisy chained with other nibbles for BISC to be supported. The daisy chain is agnostic to the order in which nibbles are connected, and unused nibbles do not need to be part of the daisy chain. For more information, see Ports.
Delay Calibration	<ul style="list-style-type: none"> Assert RX_EN_VTC and TX_EN_VTC during reset to perform delay calibration. Refer to Reset Sequence. When DLY_RDY asserts, both BISC alignment and delay calibration are complete. From this point forward delays can be changed. If multiple nibbles comprise an interface, the assertion time for DLY_RDY can vary for each nibble. Within simulation, the assertion time of DLY_RDY does not vary for each nibble in an interface, but varies as the XPHY configuration and connections change. 		
VTC	<ul style="list-style-type: none"> After PHY_RDY asserts, the interface is ready to undergo VTC When EN_VTC = 1, QTR delays and the delay within the tristate NIBBLESlice undergo VTC. This is not dependent upon RX/TX_EN_VTC. VTC on the tristate signal is not supported on NIBBLESlices affected by TBYTE_CTRL_# = T, only for TBYTE_CTRL_# = PHY_WREN. When EN_VTC = 1 and the relevant RX/TX_EN_VTC = 1, input and output delays also undergo VTC VTC compensates for the value in the delay line. For input delays, VTC compensates for the taps above align_delay If external calibration is not required and VTC is being used, tie off EN_VTC = 1 	<ul style="list-style-type: none"> Applied to input, output, and quarter delays Can be disabled through the DIS_IDLY_VT_TRACK (input delays), DIS_ODLY_VT_TRACK (output delays), and DIS_QDLY_VT_TRACK (QTR delays) attributes Coarse delays cannot undergo VTC 	



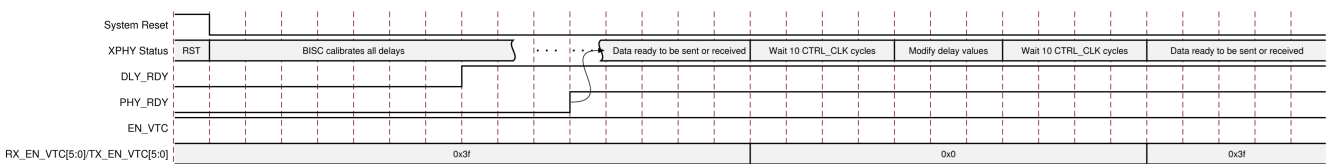
IMPORTANT! The DELAY_VALUE_x attribute and VTC are not supported if REFCLK_FREQUENCY is less than 500 MHz. In this scenario, EN_VTC should be tied to 0.

The steps before/after changing delays differ if PHY_RDY was asserted for the first time, as described in the following sequences. If not using VTC, refer to the [Controlling Delays](#) section for how to change delay values.

The following sequence and figure show the before/after steps of changing delay values on NIBBLESlice[x] *after* PHY_RDY is asserted for the first time:

1. Start with EN_VTC, RX_EN_VTC, and TX_EN_VTC asserted.
2. Deassert RX_EN_VTC and TX_EN_VTC.
3. After RX_EN_VTC and TX_EN_VTC have been deasserted, wait ten CTRL_CLK cycles.
4. Modify delay values (see [Controlling Delays](#)).
5. Wait another ten CTRL_CLK cycles, then reassert RX_EN_VTC and TX_EN_VTC.
6. The XPHY is ready to undergo VTC and can be operated normally.

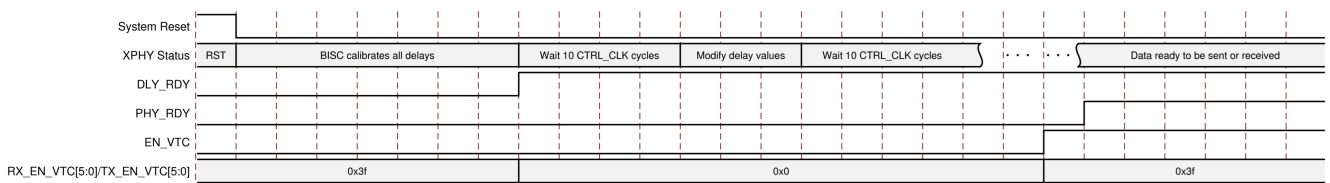
Figure 18: Changing Delay Values After PHY_RDY is Asserted for the First Time



The following sequence and figure show the before/after steps of changing delay values on NIBBLESlice[x] *before* PHY_RDY is asserted for the first time:

1. Start with EN_VTC deasserted, and RX_EN_VTC and TX_EN_VTC asserted.
2. After DLY_RDY asserts, deassert RX_EN_VTC and TX_EN_VTC.
3. After the relevant RX_EN_VTC and TX_EN_VTC is deasserted, wait ten CTRL_CLK cycles.
4. Modify delay values (see [Controlling Delays](#)).
5. Wait another ten CTRL_CLK cycles, then assert EN_VTC, RX_EN_VTC, and TX_EN_VTC.
6. After PHY_RDY asserts, the XPHY is ready to undergo VTC and can be operated normally.

Figure 19: Changing Delay Values Before PHY_RDY is Asserted for the First Time



Serial Mode

Serial mode supports receiver interfaces where the clock and data phase relationship is unknown (any interface that is not source-synchronous).

Controlling Serial Mode

To use serial mode, set `SERIAL_MODE = TRUE`. In addition:

- Nibbles with `SERIAL_MODE = TRUE` can be used as RX-only, TX-only, or RX + TX. Serial mode does not supported mixed mode and bidirectional interfaces. Note that the TX functionality is unaffected by using serial mode.
- `EN_VTC` must be grounded
- If `RX_DATA_WIDTH = 2` (1:2 deserialization): BISC must be disabled by setting `SELF_CALIBRATE = DISABLE`
- Because no strobe is accompanying the incoming data, a different clock must be used to capture data. Serial mode takes the `PLL_CLK` input and generates its own capture clock. Because `PLL_CLK` is not phase-related to the data, the application must have a way of determining the clock-to-data phase relationship.
- Inter-nibble and inter-byte clocking are not supported. The associated attributes should be set accordingly (`EN_CLK_TO_LOWER = DISABLE`, `EN_CLK_TO_UPPER = DISABLE`, `EN_OTHER_PCLK = FALSE`, `EN_OTHER_NCLK = FALSE`)
- Set `DQS_SRC = LOCAL`

TX to RX Loopback

Each `NIBBLESlice` can loop back its TX data (or strobe) to its RX datapath.

Controlling TX to RX Loopback

Each `NIBBLESlice` can loop back the TX data or strobe to its RX. While the loopback occurs prior to the IOB, the TX clock and data will still reach the IOB as normal. Thus, care must be taken to prevent contention with other incoming signals at the pad. The following sequence describes how to perform loopback of a source-synchronous interface:

1. Set `TXRX_LOOPBACK_x = TRUE` on each `NIBBLESlice[x]` loopback is desired
2. If the interface spans multiple nibbles, set `DQS_SRC = EXTERN` on nibbles receiving the strobe through inter-byte clocking. If the strobe does not enter through inter-byte clocking, set `DQS_SRC = LOCAL` on the nibble(s).
3. Set `SELF_CALIBRATE = ENABLE`
4. Set `RX_GATING = DISABLE`

5. A buffer (IBUF, OBUF, IOBUF, or a variant of one of the three) must be used for loopback to work. Connect the net from XPHY.OO[x] to the input of an OBUF, IOBUF, or variant of one of the two. Alternatively, connect the output net from an IBUF, IOBUF, or a variant of one of the two to XPHY.DATAIN[x]. Regardless of which net is used, only use one buffer (IBUF, OBUF, or variant) in this path.
6. Choose one of the following:
 - Transmit the clock as edge-aligned with data by setting TX_OUTPUT_PHASE_90_0 = FALSE. Also set RX_CLK_PHASE_P/N = SHIFT_90 so that the receiver can center the clock to data.
 - Transmit the clock as center-aligned with data by setting TX_OUTPUT_PHASE_90_0 = TRUE. Also set RX_CLK_PHASE_P/N = SHIFT_0 because no centering is necessary and the clock is received as center aligned to the data.
7. Because looping back a source-synchronous interface requires data and strobe be transmitted, a minimum of two NIBBLESLICEs must be used: one for data, one for the strobe. Only NIBBLESLICE[0] can accept a strobe. Use inter-nibble/byte clocking to forward the strobe to other nibbles, if necessary.

Controlling IBUF_DISABLE and DYN_DCI

IBUF_DISABLE and DYN_DCI are signals used for power saving. By connecting these signals to an applicable buffer, the buffer and/or its DCI can be turned off, resulting in power savings. The behavior of IBUF_DISABLE and DYN_DCI is dependent upon three factors:

1. The ODT_SRC_# attribute.
2. The IBUF_DIS_SRC_# attribute.
3. The odt_# bus of the NIBBLE_CTRL2 register. (Denoted by NIBBLE_CTRL2.odt_# below. See [Register Interface Unit](#) for more information on the NIBBLE_CTRL2 register.)

The following table shows the behavior of IBUF_DISABLE and DYN_DCI with respect to the three factors above. Note that DYN_DCI is expected to be connected to DCITERMDISABLE of the buffer (so DYN_DCI being 1 turns off ODT), and IBUF_DISABLE is expected to be connected to IBUFDISABLE of the buffer (so IBUF_DISABLE being 1 turns off the receiver). When *Controlled from the PL* is listed, this just means the PL signal should be connected to the applicable buffer input, instead of the XPHY signal being connected to the buffer. When *PHY SM* is listed, this refers to a state machine (SM) within the XPHY that controls DYN_DCI and IBUF_DISABLE. The state machine operates as follows:

1. Transitions from 1 to 0 after two cycles (with frequency equal to PLL_CLK/RX_DATA_WIDTH) after PHY_RDEN is deasserted.
2. Stays 0 for a programmable number of cycles (determined by the RD_IDLE_COUNT register and the clock being the same as the aforementioned clock).
3. After the wait, transitions from 0 to 1.

Table 15: IBUF_DISABLE and DYN_DCI Control


ODT_SRC_#	IBUF_DIS_SRC_#	NIBBLE_CTRL2.odt_#	DYN_DCI	IBUF_DISABLE
X	X	X	1 ¹	1 ¹
INTERNAL	INTERNAL	0	PHY SM controlled	PHY SM controlled
INTERNAL ³	INTERNAL	1	0	0
INTERNAL	EXTERNAL	0	PHY SM controlled	Controlled from the PL
INTERNAL	EXTERNAL	1	0	Controlled from the PL
EXTERNAL ⁴	INTERNAL	0	Controlled from the PL	PHY SM controlled
EXTERNAL	INTERNAL	1	Controlled from the PL	0
EXTERNAL	EXTERNAL	X	Controlled from the PL	Controlled from the PL


Notes:


1. This row represents the power-on state (the receiver is on and ODT is not active).
2. PHY state machine (SM): transitions 1 → 0 after two cycles (with frequency equal to PLL_CLK/RX_DATA_WIDTH) after PHY_RDEN is deasserted, then stays 0 for a programmable number of cycles (determined by the RD_IDLE_COUNT register and the clock being the same as the aforementioned clock). After the wait, it goes to 1.
3. For applications where no PL power exists, this configuration should be used.
4. For applications using the PL, use this configuration to save receiver power while the bus is idle.

Register Interface Unit

The register interface unit (RIU) is a set of 74 read/write, 16-bit registers that dynamically provides access to XPHY features. Each XPHY nibble has its own RIU, allowing all nine nibbles in a bank to be accessed simultaneously.

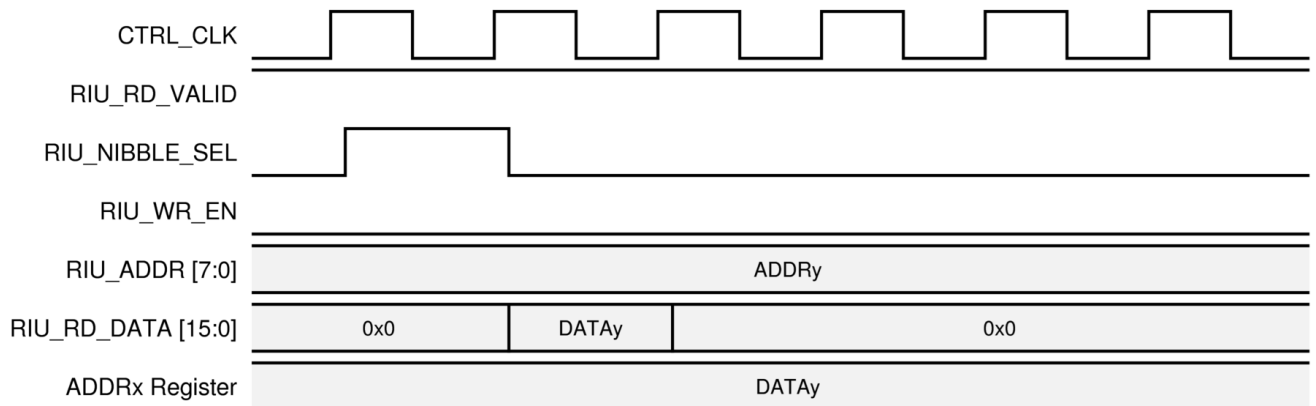
 **IMPORTANT!** When writing to the RIU, always preserve the value of any bits marked as RESERVED.

 **IMPORTANT!** Any mention of crse with respect to input, output, or quarter delays should only be thought of as part of an arbitrary bus name and does not indicate functionality. The coarse delay block is separate from the other delay blocks.

 **IMPORTANT!** RIU writes by BISC take precedence over RIU writes from the PL. If an RIU write from the PL collides with one from BISC, RIU_RD_VALID will deassert and the PL RIU write from the cycle before RIU_RD_VALID deasserted will be stored. After BISC finishes its write(s), RIU_RD_VALID will assert and the stored PL RIU write will be executed. Any writes from the PL while RIU_RD_VALID is Low will be discarded.

To read from the RIU, assert RIU_NIBBLE_SEL as shown in the following waveform. RIU_RD_VALID being High indicates the user has access to the RIU (as opposed to BISC).

Figure 20: RIU Read Operation



To write to the RIU, assert RIU_NIBBLE_SEL and RIU_WR_EN as shown in the following waveform. Notice the latency difference between reading (1 cycle) and writing (2 cycles). RIU_RD_VALID also applies to RIU writes from the PL. Assertion of RIU_RD_VALID is necessary for the writes to be successful.

Figure 21: RIU Write Operation

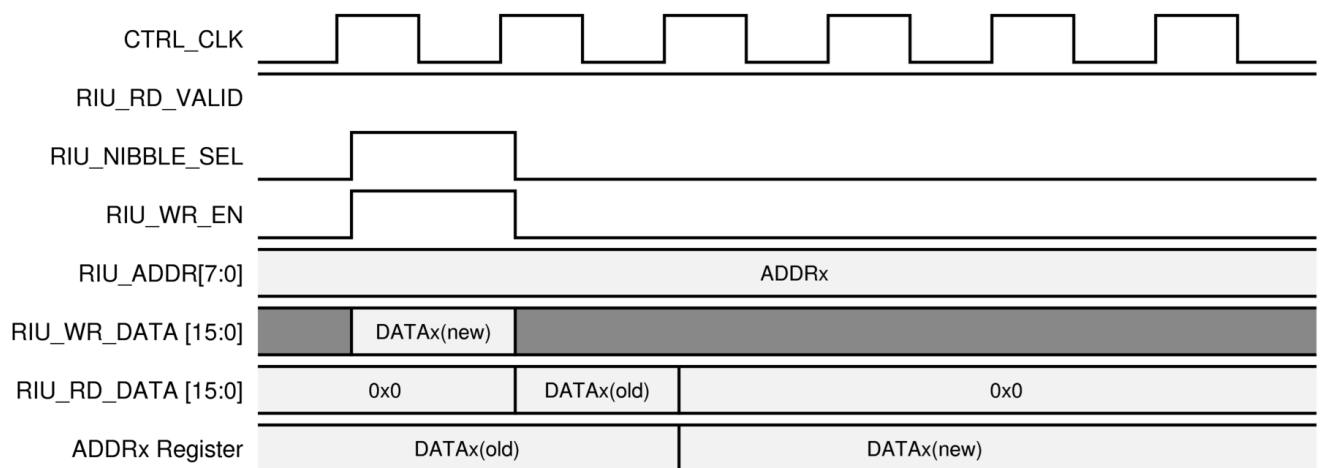


Table 16: Register Description (NIBBLE_CTRL0)

NIBBLE_CTRL0				ADDR: 0x00
Bits	Access Type	Reset Value	Name	Description
[0]	rw	EN_OTHER_PCLK	en_other_pclk	See the description of the EN_OTHER_PCLK attribute in Attributes
[1]	rw	EN_OTHER_NCLK	en_other_nclk	See the description of the EN_OTHER_NCLK attribute in Attributes

Table 16: Register Description (NIBBLE_CTRL0) (cont'd)

NIBBLE_CTRL0				ADDR: 0x00
Bits	Access Type	Reset Value	Name	Description
[2]	rw	INV_RXCLK	inv_rxclk	See the description of the INV_RXCLK attribute in Attributes
[3]	rw	SERIAL_MODE	serial_mode	See the description of the SERIAL_MODE attribute in Attributes . If the RIU (as opposed to the SERIAL_MODE attribute) is used to set this bit, BISC clears this bit as part of calibration. Set this bit again after BISC completes.
[4]	rw	TX_GATING	tx_gating	See the description of the TX_GATING attribute in Attributes
[5]	rw	RX_GATING	rx_gating	See the description of the RX_GATING attribute in Attributes
[6]	rw	CONTINUOUS_DQS	rxgating_extend	See the description of the CONTINUOUS_DQS attribute in Attributes
[15:7]	RESERVED			Reserved

Table 17: Register Description (NIBBLE_CTRL1)

NIBBLE_CTRL1				ADDR: 0x01
Bits	Access Type	Reset Value	Name	Description
[0]	rw	RX_CLK_PHASE_P	rx_clk_phase_p	See the description of the RX_CLK_PHASE_P attribute in Attributes
[1]	rw	RX_CLK_PHASE_N	rx_clk_phase_n	See the description of the RX_CLK_PHASE_N attribute in Attributes
[2]	rw	TX_OUTPUT_PHASE_90_0	tx_out_phase_90_0	See the description of the TX_OUTPUT_PHASE_90_<0-5> attribute in Attributes
[3]	rw	TX_OUTPUT_PHASE_90_1	tx_out_phase_90_1	See the description of the TX_OUTPUT_PHASE_90_<0-5> attribute in Attributes
[4]	rw	TX_OUTPUT_PHASE_90_2	tx_out_phase_90_2	See the description of the TX_OUTPUT_PHASE_90_<0-5> attribute in Attributes
[5]	rw	TX_OUTPUT_PHASE_90_3	tx_out_phase_90_3	See the description of the TX_OUTPUT_PHASE_90_<0-5> attribute in Attributes
[6]	rw	TX_OUTPUT_PHASE_90_4	tx_out_phase_90_4	See the description of the TX_OUTPUT_PHASE_90_x attribute in Attributes
[7]	rw	TX_OUTPUT_PHASE_90_5	tx_out_phase_90_5	See the description of the TX_OUTPUT_PHASE_90_<0-5> attribute in Attributes
[8]	RESERVED			Reserved
[9]	rw	TX_OUTPUT_PHASE_90_TRI	tx_out_phase_90_tri	See the description of the TX_OUTPUT_PHASE_90_TRI attribute in Attributes
[15:10]	RESERVED			Reserved

Table 18: Register Description (CALIB_CTRL)

CALIB_CTRL				ADDR: 0x02
Bits	Access Type	Reset Value	Name	Description
[0]	rw	SELF_CALIBRATE	self_calibrate	See the description of the SELF_CALIBRATE attribute in Attributes .
[1]	rw	DIS_IDLY_VT_TRACK	dis_vttrack_ibit	See the description of the DIS_IDLY_VT_TRACK attribute in Attributes .
[2]	rw	DIS_ODLY_VT_TRACK	dis_vttrack_obit	See the description of the DIS_ODLY_VT_TRACK attribute in Attributes .
[10:3]	RESERVED			Reserved.
[11]	ro	0x0	fixdly_rdy	When 1, fixdly_rdy indicates that calibration is complete (equivalent to DLY_RDY asserting on the XPHY). If 0, calibration is not complete yet. See Ports for a description of DLY_RDY.
[12]	ro	0x0	phy_rdy	The behavior of phy_rdy matches that of the PHY_RDY port. See the description of the PHY_RDY port in Ports for more information.
[13]	RESERVED			Reserved.
[14]	rw	DIS_QDLY_VT_TRACK	dis_vttrack_qdly	See the description of the DIS_QDLY_VT_TRACK attribute in Ports .
[15]	ro	0x0	pause_rdy	Indicates that BISC is paused. See WL_TRAIN.bisc_pause for more information on pausing BISC.

Table 19: Register Description (BS_RESET_CTRL)

BS_RESET_CTRL				ADDR: 0x03
Bits	Access Type	Reset Value	Name	Description
[0]	rw	0x0	clr_gate	toggling this bit clears the strobe path gating logic.
[1]	rw	0x0	bs_reset	<p>NIBBLESlice reset. When set, NIBBLESlices not masked by BS_RST_MASK.bs_reset_mask are reset. Prior to asserting this bit, set PHY_WREN to 0 if it's not already tied off to 0, regardless of the value of the TX_GATING attribute. The bs_reset must be asserted for a minimum amount of time, defined by its data width (the TX_DATA_WIDTH and RX_DATA_WIDTH attributes):</p> <ul style="list-style-type: none"> • For data width of 8: 1 CTRL_CLK cycle + 72 PLL_CLK cycles • For data width of 4: 1 CTRL_CLK cycle + 40 PLL_CLK cycles • For data width of 2: 1 CTRL_CLK cycle + 24 PLL_CLK cycles <p>While bs_reset is asserted, the TX IOBs of NIBBLESlices not masked by BS_RST_MASK.bs_reset_mask are set to the value of their associated TX_INIT_# attribute.</p>

Table 19: Register Description (BS_RESET_CTRL) (cont'd)

BS_RESET_CTRL				ADDR: 0x03
Bits	Access Type	Reset Value	Name	Description
[2]	rw	0x0	bs_reset_tri	Tristate NIBBLESlice reset. When asserted, the tristate NIBBLESlice is set to the value of the TX_INIT_TRI attribute if not masked by BS_RST_MASK.bs_reset_tri_mask. bs_reset_tri must be asserted for a minimum amount of time, defined by the interface's data width (the TX_DATA_WIDTH and RX_DATA_WIDTH attributes): <ul style="list-style-type: none"> For data width of 8: 1 CTRL_CLK cycle + 72 PLL_CLK cycles For data width of 4: 1 CTRL_CLK cycle + 40 PLL_CLK cycles For data width of 2: 1 CTRL_CLK cycle + 24 PLL_CLK cycles
[15:3]	RESERVED			Reserved.

Table 20: Register Description (PQTR)

PQTR				ADDR: 0x07
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	pqtr_dly	P-clk quarter delay. See pqtr_crse for the different ways to update pqtr_dly.
[12:9]	RESERVED			Reserved.
[13]	wo	0x0	pqtr_crse	Along with pqtr_dec and pqtr_inc, determines how pqtr_dly will change. <p>{pqtr_inc, pqtr_dec, pqtr_crse}:</p> 000, 001, 110, 111: This allows for pqtr_dly to be written to directly. 100: Increment pqtr_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at pqtr_dly = 0x1ff. 010: Decrement pqtr_dly by one tap. Wrap around to 0x1ff will happen when this decrement occurs at pqtr_dly = 0x0. 101: Increment pqtr_dly by INCDEC_CRSE.incdec_crse. When pqtr_dly is close to 0x1ff, this increment may wrap it around to above 0x0. To prevent misalignment, all NIBBLESlices affected by this update should be reset through BS_RESET_CTRL.bs_reset. 011: Decrement pqtr_dly by INCDEC_CRSE.incdec_crse. When pqtr_dly is close to 0x0, this decrement may wrap it around to below 0x1ff. To prevent misalignment, all NIBBLESlices affected by this update should be reset through BS_RESET_CTRL.bs_reset.

Table 20: Register Description (PQTR) (cont'd)

PQTR				ADDR: 0x07
Bits	Access Type	Reset Value	Name	Description
[14]	wo	0x0	pqtr_dec	P-clk quarter delay decrement. See the description for pqtr_crse.
[15]	wo	0x0	pqtr_inc	P-clk quarter delay increment. See the description for pqtr_crse.

Table 21: Register Description (NQTR)

NQTR				ADDR: 0x08
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	nqtr_dly	N-clk quarter delay. See nqtr_crse for the different ways to update nqtr_dly.
[12:9]	RESERVED			Reserved.
[13]	wo	0x0	nqtr_crse	<p>Along with nqtr_dec and nqtr_inc, determines how nqtr_dly will change.</p> <p>{nqtr_inc, nqtr_dec, nqtr_crse}: 000, 001, 110, 111: This allows for nqtr_dly to be written to directly. 100: Increment nqtr_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at nqtr_dly = 0x1ff. 010: Decrement nqtr_dly by one tap. Wrap around to 0x1ff will happen when this decrement occurs at nqtr_dly = 0x0. 101: Increment nqtr_dly by INCDEC_CRSE.incdec_crse. When nqtr_dly is close to 0x1ff, this increment may wrap it around to above 0x0. To prevent misalignment, all NIBBLESLICES affected by this update should be reset through BS_RESET_CTRL.bs_reset. 011: Decrement nqtr_dly by INCDEC_CRSE.incdec_crse. When nqtr_dly is close to 0x0, this decrement may wrap it around to below 0x1ff. To prevent misalignment, all NIBBLESLICES affected by this update should be reset through BS_RESET_CTRL.bs_reset.</p>
[14]	wo	0x0	nqtr_dec	N-clk quarter delay decrement. See the description for nqtr_crse.
[15]	wo	0x0	nqtr_inc	N-clk quarter delay increment. See the description for nqtr_crse.

Table 22: Register Description (TRISTATE_ODLY)

TRISTATE_ODLY				ADDR: 0x0a
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	tristate_dly	Tristate NIBBLESlice delay value. See tristate_crse for the different ways to update tristate_dly.
[12:9]	RESERVED			Reserved
[13]	wo	0x0	tristate_crse	Along with tristate_dec and tristate_inc, determines how tristate_dly will change. {tristate_inc, tristate_dec, tristate_crse}: 000, 001, 110, 111: This allows for tristate_dly to be written to. 100: Increment tristate_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at tristate_dly = 0x1ff 010: Decrement tristate_dly by one tap. Wrap around to 0x1ff will happen when this decrement occurs at tristate_dly = 0x0 101: Increment tristate_dly by INCDEC_CRSE.incdec_crse. When tristate_dly is close to 0x1ff, this increment may wrap it around to above 0x0 011: Decrement tristate_dly by INCDEC_CRSE.incdec_crse. When tristate_dly is close to 0x0, this decrement may wrap it around to below 0x1ff
[14]	wo	0x0	tristate_dec	Tristate NIBBLESlice delay decrement. See the description for tristate_crse.
[15]	wo	0x0	tristate_inc	Tristate NIBBLESlice delay increment. See the description for tristate_crse.

Table 23: Register Description (ODLY0)

ODLY0				ADDR: 0x0b
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	odly0_dly	NIBBLESlice[0] output delay value. Reading odly0_dly returns the output delay value in NIBBLESlice[0], similar to reading CNTVALUEOUT[8:0] from the PL. See odly0_crse for the different ways to update odly0_dly.
[12:9]	RESERVED			Reserved

Table 23: Register Description (ODLY0) (cont'd)

ODLY0				ADDR: 0x0b
Bits	Access Type	Reset Value	Name	Description
[13]	wo	0x0	odly0_crse	Along with odly0_dec and odly0_inc, determines how odly0_dly will change. {odly0_crse, odly0_dec, odly0_inc}: 000, 011, 100, 111: This allows for odly0_dly to be written to directly, similar to loading a delay from the PL to CNTVALUEIN[8:0]. 001: Increment odly0_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at odly0_dly = 0x1ff 010: Decrement odly0_dly by one tap. Wrap around to 0x1ff will happen when this decrement occurs at odly0_dly = 0x0 101: Increment odly0_dly by INCDEC_CRSE.incdec_crse. When odly0_dly is close to 0x1ff, this increment may wrap it around to above 0x0 110: Decrement odly0_dly by INCDEC_CRSE.incdec_crse. When odly0_dly is close to 0x0, this decrement may wrap it around to below 0x1ff
[14]	wo	0x0	odly0_dec	NIBBLESLICE[0] output delay decrement. See the description for odly0_crse.
[15]	wo	0x0	odly0_inc	NIBBLESLICE[0] output delay increment. See the description for odly0_crse.

Table 24: Register Description (ODLY1)

ODLY1				ADDR: 0x0c
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	odly1_dly	NIBBLESLICE[1] output delay value. Reading odly1_dly returns the output delay value in NIBBLESLICE[1], similar to reading CNTVALUEOUT[17:9] from the PL. See odly1_crse for the different ways to update odly1_dly.
[12:9]	RESERVED			Reserved

Table 24: Register Description (ODLY1) (cont'd)

ODLY1				ADDR: 0x0c
Bits	Access Type	Reset Value	Name	Description
[13]	wo	0x0	odly1_crse	Along with odly1_dec and odly1_inc, determines how odly1_dly will change. {odly1_crse, odly1_dec, odly1_inc}: 000, 011, 100, 111: This allows for odly1_dly to be written to directly, similar to loading a delay from the PL to CNTVALUEIN[17:9]. 001: Increment odly1_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at odly1_dly = 0x1ff 010: Decrement odly1_dly by one tap. Wrap around to 0x1ff will happen when this decrement occurs at odly1_dly = 0x0 101: Increment odly1_dly by INCDEC_CRSE.incdec_crse. When odly1_dly is close to 0x1ff, this increment may wrap it around to above 0x0 110: Decrement odly1_dly by INCDEC_CRSE.incdec_crse. When odly1_dly is close to 0x0, this decrement may wrap it around to below 0x1ff
[14]	wo	0x0	odly1_dec	NIBBLESLICE[1] output delay decrement. See the description for odly1_crse.
[15]	wo	0x0	odly1_inc	NIBBLESLICE[1] output delay increment. See the description for odly1_crse.

Table 25: Register Description (ODLY2)

ODLY2				ADDR: 0x0d
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	odly2_dly	NIBBLESLICE[2] output delay value. Reading odly2_dly returns the output delay value in NIBBLESLICE[2], similar to reading CNTVALUEOUT[26:18] from the PL. See odly2_crse for the different ways to update odly2_dly.
[12:9]	RESERVED			Reserved

Table 25: Register Description (ODLY2) (cont'd)

ODLY2				ADDR: 0x0d
Bits	Access Type	Reset Value	Name	Description
[13]	wo	0x0	odly2_crse	Along with odly2_dec and odly2_inc, determines how odly2_dly will change. {odly2_crse, odly2_dec, odly2_inc}: 000, 011, 100, 111: This allows for odly2_dly to be written to directly, similar to loading a delay from the PL to CNTVALUEIN[26:18]. 001: Increment odly2_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at odly2_dly = 0x1ff 010: Decrement odly2_dly by one tap. Wrap around to 0x1ff will happen when this decrement occurs at odly2_dly = 0x0 101: Increment odly2_dly by INCDEC_CRSE.incdec_crse. When odly2_dly is close to 0x1ff, this increment may wrap it around to above 0x0 110: Decrement odly2_dly by INCDEC_CRSE.incdec_crse. When odly2_dly is close to 0x0, this decrement may wrap it around to below 0x1ff
[14]	wo	0x0	odly2_dec	NIBBLESlice[2] output delay decrement. See the description for odly2_crse.
[15]	wo	0x0	odly2_inc	NIBBLESlice[2] output delay increment. See the description for odly2_crse.

Table 26: Register Description (ODLY3)

ODLY3				ADDR: 0x0e
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	odly3_dly	NIBBLESlice[3] output delay value. Reading odly3_dly returns the output delay value in NIBBLESlice[3], similar to reading CNTVALUEOUT[35:27] from the PL. See odly3_crse for the different ways to update odly3_dly.
[12:9]	RESERVED			Reserved

Table 26: Register Description (ODLY3) (cont'd)

ODLY3				ADDR: 0x0e
Bits	Access Type	Reset Value	Name	Description
[13]	wo	0x0	odly3_crse	Along with odly3_dec and odly3_inc, determines how odly3_dly will change. {odly3_crse, odly3_dec, odly3_inc}: 000, 011, 100, 111: This allows for odly3_dly to be written to directly, similar to loading a delay from the PL to CNTVALUEIN[35:27]. 001: Increment odly3_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at odly3_dly = 0x1ff 010: Decrement odly3_dly by one tap. Wrap around to 0x1ff will happen when this decrement occurs at odly3_dly = 0x0 101: Increment odly3_dly by INCDEC_CRSE.incdec_crse. When odly3_dly is close to 0x1ff, this increment may wrap it around to above 0x0 110: Decrement odly3_dly by INCDEC_CRSE.incdec_crse. When odly3_dly is close to 0x0, this decrement may wrap it around to below 0x1ff
[14]	wo	0x0	odly3_dec	NIBBLESLICE[3] output delay decrement. See the description for odly3_crse.
[15]	wo	0x0	odly3_inc	NIBBLESLICE[3] output delay increment. See the description for odly3_crse.

Table 27: Register Description (ODLY4)

ODLY4				ADDR: 0x0f
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	odly4_dly	NIBBLESLICE[4] output delay value. Reading odly4_dly returns the output delay value in NIBBLESLICE[4], similar to reading CNTVALUEOUT[44:36] from the PL. See odly4_crse for the different ways to update odly4_dly.
[12:9]	RESERVED			Reserved

Table 27: Register Description (ODLY4) (cont'd)

ODLY4				ADDR: 0x0f
Bits	Access Type	Reset Value	Name	Description
[13]	wo	0x0	odly4_crse	Along with odly4_dec and odly4_inc, determines how odly4_dly will change. {odly4_crse, odly4_dec, odly4_inc}: 000, 011, 100, 111: This allows for odly4_dly to be written to directly, similar to loading a delay from the PL to CNTVALUEIN[44:36]. 001: Increment odly4_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at odly4_dly = 0x1ff 010: Decrement odly4_dly by one tap. Wrap around to 0x1ff will happen when this decrement occurs at odly4_dly = 0x0 101: Increment odly4_dly by INCDEC_CRSE.incdec_crse. When odly4_dly is close to 0x1ff, this increment may wrap it around to above 0x0 110: Decrement odly4_dly by INCDEC_CRSE.incdec_crse. When odly4_dly is close to 0x0, this decrement may wrap it around to below 0x1ff
[14]	wo	0x0	odly4_dec	NIBBLESLICE[4] output delay decrement. See the description for odly4_crse.
[15]	wo	0x0	odly4_inc	NIBBLESLICE[4] output delay increment. See the description for odly4_crse.

Table 28: Register Description (ODLY5)

ODLY5				ADDR: 0x10
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	odly5_dly	NIBBLESLICE[5] output delay value. Reading odly5_dly returns the output delay value in NIBBLESLICE[5], similar to reading CNTVALUEOUT[53:45] from the PL. See odly5_crse for the different ways to update odly5_dly.
[12:9]	RESERVED			Reserved

Table 28: Register Description (ODLY5) (cont'd)

ODLY5				ADDR: 0x10
Bits	Access Type	Reset Value	Name	Description
[13]	wo	0x0	odly5_crse	Along with odly5_dec and odly5_inc, determines how odly5_dly will change. {odly5_crse, odly5_dec, odly5_inc}: 000, 011, 100, 111: This allows for odly5_dly to be written to directly, similar to loading a delay from the PL to CNTVALUEIN[53:45]. 001: Increment odly5_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at odly5_dly = 0x1ff 010: Decrement odly5_dly by one tap. Wrap around to 0x1ff will happen when this decrement occurs at odly5_dly = 0x0 101: Increment odly5_dly by INCDEC_CRSE.incdec_crse. When odly5_dly is close to 0x1ff, this increment may wrap it around to above 0x0 110: Decrement odly5_dly by INCDEC_CRSE.incdec_crse. When odly5_dly is close to 0x0, this decrement may wrap it around to below 0x1ff
[14]	wo	0x0	odly5_dec	NIBBLESlice[5] output delay decrement. See the description for odly5_crse.
[15]	wo	0x0	odly5_inc	NIBBLESlice[5] output delay increment. See the description for odly5_crse.

Table 29: Register Description (BS_RST_MASK)

BS_RST_MASK				ADDR: 0x11
Bits	Access Type	Reset Value	Name	Description
[5:0]	rw	0x0	bs_reset_mask	NIBBLESlice reset mask. When bit x is set, the corresponding NIBBLESlice[x] will not get reset when a 1 is written to BS_RESET_CTRL.bs_reset.
[6]	rw	0x0	bs_reset_tri_mask	Tristate NIBBLESlice reset mask. When set, the tristate NIBBLESlice will not get reset when a 1 is written to BS_RESET_CTRL.bs_reset_tri.
[15:7]	RESERVED			Reserved

Table 30: Register Description (IDLY0)

IDLY0				ADDR: 0x12
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	idly0_dly	NIBBLESLICE[0] input delay value. idly0_dly does include align_delay, but writes or updates to idly0_dly do not change align_delay. Reading idly0_dly returns the input delay value in NIBBLESLICE[0], similar to reading CNTVALUEOUT[8:0] from the PL. See idly0_crse for the different ways to update idly0_dly.
[12:9]	RESERVED			Reserved
[13]	wo	0x0	idly0_crse	Along with idly0_dec and idly0_inc, determines how idly0_dly will change. {idly0_crse, idly0_dec, idly0_inc}: 000, 011, 100, 111: This allows for idly0_dly to be written to directly, similar to loading a delay from the PL to CNTVALUEIN[8:0]. 001: Increment idly0_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at idly0_dly = 0x1ff 010: Decrement idly0_dly by one tap. Wrap around to 0x1ff will happen when this decrement occurs at idly0_dly = 0x0 101: Increment idly0_dly by INCDEC_CRSE.incdec_crse. When idly0_dly is close to 0x1ff, this increment may wrap it around to above 0x0 110: Decrement idly0_dly by INCDEC_CRSE.incdec_crse. When idly0_dly is close to 0x0, this decrement may wrap it around to below 0x1ff
[14]	wo	0x0	idly0_dec	NIBBLESLICE[0] input delay decrement. See the description for idly0_crse.
[15]	wo	0x0	idly0_inc	NIBBLESLICE[0] input delay increment. See the description for idly0_crse.

Table 31: Register Description (IDLY1)

IDLY1				ADDR: 0x13
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	idly1_dly	NIBBLESLICE[1] input delay value. idly1_dly does include align_delay, but writes or updates to idly1_dly do not change align_delay. Reading idly1_dly returns the input delay value in NIBBLESLICE[1], similar to reading CNTVALUEOUT[17:9] from the PL. See idly1_crse for the different ways to update idly1_dly.
[12:9]	RESERVED			Reserved

Table 31: Register Description (IDLY1) (cont'd)

IDLY1				ADDR: 0x13
Bits	Access Type	Reset Value	Name	Description
[13]	wo	0x0	idly1_crse	Along with idly1_dec and idly1_inc, determines how idly1_dly will change. {idly1_crse, idly1_dec, idly1_inc}: 000, 011, 100, 111: This allows for idly1_dly to be written to directly, similar to loading a delay from the PL to CNTVALUEIN[17:9]. 001: Increment idly1_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at idly1_dly = 0x1ff 010: Decrement idly1_dly by one tap. Wrap around to 0x1ff will happen when this decrement occurs at idly1_dly = 0x0 101: Increment idly1_dly by INCDEC_CRSE.incdec_crse. When idly1_dly is close to 0x1ff, this increment may wrap it around to above 0x0 110: Decrement idly1_dly by INCDEC_CRSE.incdec_crse. When idly1_dly is close to 0x0, this decrement may wrap it around to below 0x1ff
[14]	wo	0x0	idly1_dec	NIBBLES_LICE[1] input delay decrement. See the description for idly1_crse.
[15]	wo	0x0	idly1_inc	NIBBLES_LICE[1] input delay increment. See the description for idly1_crse.

Table 32: Register Description (IDLY2)

IDLY2				ADDR: 0x14
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	idly2_dly	NIBBLES_LICE[2] input delay value. idly2_dly does include align_delay, but writes or updates to idly2_dly do not change align_delay. Reading idly2_dly returns the input delay value in NIBBLES_LICE[2], similar to reading CNTVALUEOUT[26:18] from the PL. See idly2_crse for the different ways to update idly2_dly.
[12:9]	RESERVED			Reserved

Table 32: Register Description (IDLY2) (cont'd)

IDLY2				ADDR: 0x14
Bits	Access Type	Reset Value	Name	Description
[13]	wo	0x0	idly2_crse	Along with idly2_dec and idly2_inc, determines how idly2_dly will change. {idly2_crse, idly2_dec, idly2_inc}: 000, 011, 100, 111: This allows for idly2_dly to be written to directly, similar to loading a delay from the PL to CNTVALUEIN[26:18]. 001: Increment idly2_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at idly2_dly = 0x1ff 010: Decrement idly2_dly by one tap. Wrap around to 0x1ff will happen when this decrement occurs at idly2_dly = 0x0 101: Increment idly2_dly by INCDEC_CRSE.incdec_crse. When idly2_dly is close to 0x1ff, this increment may wrap it around to above 0x0 110: Decrement idly2_dly by INCDEC_CRSE.incdec_crse. When idly2_dly is close to 0x0, this decrement may wrap it around to below 0x1ff
[14]	wo	0x0	idly2_dec	NIBBLES_LICE[2] input delay decrement. See the description for idly2_crse.
[15]	wo	0x0	idly2_inc	NIBBLES_LICE[2] input delay increment. See the description for idly2_crse.

Table 33: Register Description (IDLY3)

IDLY3				ADDR: 0x15
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	idly3_dly	NIBBLES_LICE[3] input delay value. idly3_dly does include align_delay, but writes or updates to idly3_dly do not change align_delay. Reading idly3_dly returns the input delay value in NIBBLES_LICE[3], similar to reading CNTVALUEOUT[35:27] from the PL. See idly3_crse for the different ways to update idly3_dly.
[12:9]	RESERVED			Reserved

Table 33: Register Description (IDLY3) (cont'd)

IDLY3				ADDR: 0x15
Bits	Access Type	Reset Value	Name	Description
[13]	wo	0x0	idly3_crse	Along with idly3_dec and idly3_inc, determines how idly3_dly will change. {idly3_crse, idly3_dec, idly3_inc}: 000, 011, 100, 111: This allows for idly3_dly to be written to directly, similar to loading a delay from the PL to CNTVALUEIN[35:27]. 001: Increment idly3_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at idly3_dly = 0x1ff 010: Decrement idly3_dly by one tap. Wrap around to 0x1ff will happen when this decrement occurs at idly3_dly = 0x0 101: Increment idly3_dly by INCDEC_CRSE.incdec_crse. When idly3_dly is close to 0x1ff, this increment may wrap it around to above 0x0 110: Decrement idly3_dly by INCDEC_CRSE.incdec_crse. When idly3_dly is close to 0x0, this decrement may wrap it around to below 0x1ff
[14]	wo	0x0	idly3_dec	NIBBLESlice[3] input delay decrement. See the description for idly3_crse.
[15]	wo	0x0	idly3_inc	NIBBLESlice[3] input delay increment. See the description for idly3_crse.

Table 34: Register Description (IDLY4)

IDLY4				ADDR: 0x16
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	idly4_dly	NIBBLESlice[4] input delay value. idly4_dly does include align_delay, but writes or updates to idly4_dly do not change align_delay. Reading idly4_dly returns the input delay value in NIBBLESlice[4], similar to reading CNTVALUEOUT[44:36] from the PL. See idly4_crse for the different ways to update idly4_dly.
[12:9]	RESERVED			Reserved

Table 34: Register Description (IDLY4) (cont'd)

IDLY4				ADDR: 0x16
Bits	Access Type	Reset Value	Name	Description
[13]	wo	0x0	idly4_crse	Along with idly4_dec and idly4_inc, determines how idly4_dly will change. {idly4_crse, idly4_dec, idly4_inc}: 000, 011, 100, 111: This allows for idly4_dly to be written to directly, similar to loading a delay from the PL to CNTVALUEIN[44:36]. 001: Increment idly4_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at idly4_dly = 0x1ff 010: Decrement idly4_dly by one tap. Wrap around to 0x1ff will happen when this decrement occurs at idly4_dly = 0x0 101: Increment idly4_dly by INCDEC_CRSE.incdec_crse. When idly4_dly is close to 0x1ff, this increment may wrap it around to above 0x0 110: Decrement idly4_dly by INCDEC_CRSE.incdec_crse. When idly4_dly is close to 0x0, this decrement may wrap it around to below 0x1ff
[14]	wo	0x0	idly4_dec	NIBBLESlice[4] input delay decrement. See the description for idly4_crse.
[15]	wo	0x0	idly4_inc	NIBBLESlice[4] input delay increment. See the description for idly4_crse.

Table 35: Register Description (IDLY5)

IDLY5				ADDR: 0x17
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	idly5_dly	NIBBLESlice[5] input delay value. idly5_dly does include align_delay, but writes or updates to idly5_dly do not change align_delay. Reading idly5_dly returns the input delay value in NIBBLESlice[5], similar to reading CNTVALUEOUT[53:45] from the PL. See idly5_crse for the different ways to update idly5_dly.
[12:9]	RESERVED			Reserved

Table 35: Register Description (IDLY5) (cont'd)

IDLY5				ADDR: 0x17
Bits	Access Type	Reset Value	Name	Description
[13]	wo	0x0	idly5_crse	Along with idly5_dec and idly5_inc, determines how idly5_dly will change. {idly5_crse, idly5_dec, idly5_inc}: 000, 011, 100, 111: This allows for idly5_dly to be written to directly, similar to loading a delay from the PL to CNTVALUEIN[53:45]. 001: Increment idly5_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at idly5_dly = 0x1ff 010: Decrement idly5_dly by one tap. Wrap around to 0x1ff will happen when this decrement occurs at idly5_dly = 0x0 101: Increment idly5_dly by INCDEC_CRSE.incdec_crse. When idly5_dly is close to 0x1ff, this increment may wrap it around to above 0x0 110: Decrement idly5_dly by INCDEC_CRSE.incdec_crse. When idly5_dly is close to 0x0, this decrement may wrap it around to below 0x1ff
[14]	wo	0x0	idly5_dec	NIBBLES_LICE[5] input delay decrement. See the description for idly5_crse.
[15]	wo	0x0	idly5_inc	NIBBLES_LICE[5] input delay increment. See the description for idly5_crse.

Table 36: Register Description (CRSE_DLY)

CRSE_DLY				ADDR: 0x18
Bits	Access Type	Reset Value	Name	Description
[4:0]	rw	0x0	crse_dly	Coarse delay
[13:5]	RESERVED			Reserved
[14]	wo	0x0	crse_dec	Along with crse_inc, determines how crse_dly will change. {crse_inc, crse_dec}: 00, 11: This allows for crse_dly to be written to directly. 10: Increment crse_dly by one tap. Wrap around to 0x0 will happen when this increment occurs at crse_dly = 0x1f 01: Decrement by one tap. Wrap around to 0x1ff will happen when this decrement occurs at 0x0
[15]	wo	0x0	crse_inc	Coarse delay increment. See the description for crse_dec.

Table 37: Register Description (WL_TRAIN)

WL_TRAIN				ADDR: 0x2b
Bits	Access Type	Reset Value	Name	Description
[0]		RESERVED		Reserved
[1]	rw	0x0	en_vtc	This operates in the same way as the EN_VTC port. See Controlling Built-in Self-Calibration for more details on VTC.
[6:2]		RESERVED		Reserved
[7]	rw	0x0	bisc_pause	When set, pauses BISC
[15:8]		RESERVED		Reserved

Table 38: Register Description (RD_IDLE_COUNT)

RD_IDLE_COUNT				ADDR: 0x34
Bits	Access Type	Reset Value	Name	Description
[5:0]	rw	0x10	rd_idle_count	The number of cycles with frequency equal to PLL_CLK/RX_DATA_WIDTH that must occur before ODT is disabled. See Controlling IBUF_DISABLE and DYN_DCI for more information.
[15:7]		RESERVED		Reserved

Table 39: Register Description (RL_DLY_QTR)

RL_DLY_QTR				ADDR: 0x36
Bits	Access Type	Reset Value	Name	Description
[8:0]	rw	0x0	rl_dly_qtr	Read quarter cycle delay. Used to know the number of taps equivalent to 90° to detect quarter cycle rollover.
[15:9]		RESERVED		Reserved

Table 40: Register Description (INCDEC_CRSE)

INCDEC_CRSE				ADDR: 0x3b
Bits	Access Type	Reset Value	Name	Description
[7:0]	rw	0x08	incdec_crse	Represents the number of taps used for certain delay line updates (input delays, output delays, tristate output delays, and quarter (QTR) delays) when updated through the RIU.
[15:8]		RESERVED		Reserved

Table 41: Register Description (NIBBLE_CTRL2)

NIBBLE_CTRL2				ADDR: 0x3e
Bits	Access Type	Reset Value	Name	Description
[0]	rw	0x0	odt_0	Contributes to NIBBLESLICE[0]'s on-die termination (ODT) control. See Controlling IBUF_DISABLE and DYN_DCI for more information.
[1]	rw	0x0	odt_1	Contributes to NIBBLESLICE[1]'s ODT control. See Controlling IBUF_DISABLE and DYN_DCI for more information.
[2]	rw	0x0	odt_2	Contributes to NIBBLESLICE[2]'s ODT control. See Controlling IBUF_DISABLE and DYN_DCI for more information.
[3]	rw	0x0	odt_3	Contributes to NIBBLESLICE[3]'s ODT control. See Controlling IBUF_DISABLE and DYN_DCI for more information.
[4]	rw	0x0	odt_4	Contributes to NIBBLESLICE[4]'s ODT control. See Controlling IBUF_DISABLE and DYN_DCI for more information.
[5]	rw	0x0	odt_5	Contributes to NIBBLESLICE[5]'s ODT control. See Controlling IBUF_DISABLE and DYN_DCI for more information.
[9:6]	RESERVED			Reserved
[10]	rw	CRSE_DLY_EN	crse_dly_enable	See the description of the CRSE_DLY_EN attribute in Attributes .
[13:11]	rw	DQS_MODE	dqs_mode	See the description of the DQS_MODE attribute in Attributes .
[15:14]	RESERVED			Reserved

Table 42: Register Description (TXRX_LOOPBACK)

TXRX_LOOPBACK				ADDR: 0x3f
Bits	Access Type	Reset Value	Name	Description
[5:0]	rw	{TXRX_LOOPBACK_5, TXRX_LOOPBACK_4, TXRX_LOOPBACK_3, TXRX_LOOPBACK_2, TXRX_LOOPBACK_1, TXRX_LOOPBACK_0}	txrx_loopback	See the description of the TXRX_LOOPBACK_<0-5> attribute in Attributes .
[15:6]	RESERVED			Reserved

Table 43: Register Description (ODELAY_BYPASS)

ODELAY_BYPASS				ADDR: 0x40
Bits	Access Type	Reset Value	Name	Description
[5:0]	rw	0x0	odelay_bypass	When bit x is set, the output delay of NIBBLESLICE[x] is bypassed

Table 43: Register Description (ODELAY_BYPASS) (cont'd)

ODELAY_BYPASS				ADDR: 0x40
Bits	Access Type	Reset Value	Name	Description
[15:6]		RESERVED		Reserved

XPHY Primitive

Ports

The following table lists ports and their descriptions. For ports with width [5:0], each element maps to the NIBBLESlice with the matching number. For example, DATAIN[0] is the incoming data to NIBBLESlice[0], DATAIN[1] is the incoming data to NIBBLESlice[1], and so on.

Some ports can be connected through the Boundary Logic Interface (BLI). For more information on which ports can be connected, see [Boundary Logic Interface](#).

Table 44: XPHY Ports

Port Name	Width	Input/Output	Clock Domain	Description
BISC_START_IN		Input	-	This is a simulation-only port required for BISC. If an interface is composed of multiple nibbles, connect this port to BISC_START_OUT of the next nibble in the daisy chain. If an interface is composed of only one nibble or this port is on the last nibble in the daisy chain, connect this port to BISC_STOP_OUT of the same nibble.
BISC_STOP_IN		Input	-	This is a simulation-only port required for BISC. If an interface is composed of multiple nibbles, connect this port to BISC_STOP_OUT from the previous nibble in the daisy chain. If an interface is composed of only one nibble or this port is on the first nibble in the daisy chain, tie this port High.
CE	[5:0]	Input	CTRL_CLK	Used with INC, LD, and CNTVALUEIN to change delay values on a per-NIBBLESlice basis. Refer to Controlling Delays for how to set CE, LD, INC, and CNTVALUEIN to get the desired delay change.
CLK_FROM_OTHER_XPHY		Input	-	Input for an inter-byte clock from certain nibbles. In being part of inter-byte clocking, CLK_FROM_OTHER_XPHY can only be connected to CLK_TO_LOWER or CLK_TO_UPPER of another nibble and cannot be connected to the programmable logic. CLK_FROM_OTHER_XPHY of the source nibble starting the inter-byte clocking should be set to 1'b1. The Clocking section lists the nibbles that are capable of inter-byte clocking with each other.

Table 44: XPHY Ports (cont'd)

Port Name	Width	Input/Output	Clock Domain	Description
CNTVALUEIN	[53:0]	Input	CTRL_CLK	<p>The delay value (in taps) to be loaded to the input or output delay selected by RXTX_SEL. Each NIBBLESlice is associated with 9b of CNTVALUEIN. NIBBLESlice[0] is associated with CNTVALUEIN[8:0], NIBBLESlice[1] with CNTVALUEIN[17:9], NIBBLESlice[2] with CNTVALUEIN[26:18], NIBBLESlice[3] with CNTVALUEIN[35:27], NIBBLESlice[4] with CNTVALUEIN[44:36], and NIBBLESlice[5] with CNTVALUEIN[53:45].</p> <p>Refer to Controlling Delays for how to set CE, LD, INC, and CNTVALUEIN to get the desired delay change.</p>
CTRL_CLK		Input	-	<p>Clock used for RIU access, delay line updates, and BISC.</p> <p>The ratio of different CTRL_CLK frequencies of all nibbles in a bank cannot be greater than 4:1.</p>
DATAIN	[5:0]	Input	ASYNC	<p>RX data from the IOB. DATAIN[0] enters NIBBLESlice[0], DATAIN[1] enters NIBBLESlice[1], and so on. The data is collected and stored in the FIFO of the corresponding NIBBLESlice, after which it is output on the corresponding Q port (Q0 maps to NIBBLESlice[0], etc.).</p> <p>The Controlling FIFO Modes section has more details on this mapping.</p>
D0	[7:0]	Input	PLL_CLK	<p>TX data from the programmable logic for NIBBLESlice[0]. After being serialized, D0 is output on O0[0].</p> <p>The Controlling Tristate Control section describes how D0 is serialized for different data widths.</p>
D1	[7:0]	Input	PLL_CLK	<p>TX data from the programmable logic for NIBBLESlice[1]. After being serialized, D1 is output on O0[1].</p> <p>The Controlling Tristate Control section describes how D1 is serialized for different data widths.</p>
D2	[7:0]	Input	PLL_CLK	<p>TX data from the programmable logic for NIBBLESlice[2]. After being serialized, D2 is output on O0[2].</p> <p>The Controlling Tristate Control section describes how D2 is serialized for different data widths.</p>
D3	[7:0]	Input	PLL_CLK	<p>TX data from the programmable logic for NIBBLESlice[3]. After being serialized, D3 is output on O0[3].</p> <p>The Controlling Tristate Control section describes how D3 is serialized for different data widths.</p>

Table 44: XPHY Ports (cont'd)

Port Name	Width	Input/Output	Clock Domain	Description
D4	[7:0]	Input	PLL_CLK	TX data from the programmable logic for NIBBLESlice[4]. After being serialized, D4 is output on O0[4]. The Controlling Tristate Control section describes how D4 is serialized for different data widths.
D5	[7:0]	Input	PLL_CLK	TX data from the programmable logic for NIBBLESlice[5]. After being serialized, D5 is output on O0[5]. The Controlling Tristate Control section describes how D5 is serialized for different data widths.
EN_VTC		Input	ASYNC	Assert to enable VTC for a nibble. If asserted without RX_EN_VTC/TX_EN_VTC, only QTR delays undergo VTC.
FIFO_RDEN		Input	FIFO_RD_CLK	When HIGH, increments the FIFO read pointer every FIFO_RD_CLK cycle to read data from the FIFO. Keep asserted every cycle that data is moved from the FIFO to the programmable logic.
FIFO_RD_CLK		Input	-	The FIFO read clock.
INC	[5:0]	Input	CTRL_CLK	Used with CE, LD, and CNTVALUEIN to change delay values on a per-NIBBLESlice basis. Refer to Controlling Delays for how to set CE, LD, INC, and CNTVALUEIN to get the desired delay change.
LD	[5:0]	Input	CTRL_CLK	Used with CE, INC, and CNTVALUEIN to change delay values on a per-NIBBLESlice basis. Refer to Controlling Delays for how to set CE, LD, INC, and CNTVALUEIN to get the desired delay change.
NCLK_NIBBLE_IN		Input	-	N-clk input for an inter-nibble clock from certain nibbles. In being part of inter-nibble clocking, NCLK_NIBBLE_IN can only be connected to NCLK_NIBBLE_OUT of another nibble and cannot be connected to the programmable logic. The Clocking section lists the nibbles that are capable of inter-nibble clocking.
PCLK_NIBBLE_IN		Input	-	P-clk input for an inter-nibble clock from certain nibbles. In being part of inter-nibble clocking, PCLK_NIBBLE_IN can only be connected to PCLK_NIBBLE_OUT of another nibble and cannot be connected to the programmable logic. The Clocking section lists the nibbles that are capable of inter-nibble clocking.
PHY_RDCS0	[3:0]	Input	PLL_CLK	This port is only for memory-related use.
PHY_RDCS1	[3:0]	Input	PLL_CLK	This port is only for memory-related use.

Table 44: XPHY Ports (cont'd)

Port Name	Width	Input/Output	Clock Domain	Description
PHY_RDEN	[3:0]	Input	See description	<p>PHY_RDEN controls accepting or rejecting the strobe entering on NIBBLESlice[0] or from inter-byte clocking (inter-nibble clocking does not support gating through PHY_RDEN), depending upon the settings of CONTINUOUS_DQS, RX_GATING, and RX_DATA_WIDTH. Always ensure the strobe has stabilized and BISC has completed before asserting PHY_RDEN. Refer to Controlling Built-in Self-Calibration for when BISC is considered completed. PHY_RDEN control is outlined in the next section:</p> <p>When RX_DATA_WIDTH = don't care, RX_GATING = ENABLE, and CONTINUOUS_DQS = TRUE, then the four bits of PHY_RDEN are OR'd together and that output is used to control the gate. If the result of the OR operation is 1, then the strobe is accepted. If it is 0, then the strobe is rejected. PHY_RDEN is synchronized to the strobe for this attribute combination. Setting CONTINUOUS_DQS = TRUE requires that three strobe cycles be received prior to receiving data to prevent data loss.</p> <p>When RX_DATA_WIDTH = 4 or 8, RX_GATING = ENABLE, and CONTINUOUS_DQS = FALSE, set the following bits of PHY_RDEN to 1 to accept the strobe or 0 to reject the strobe. PHY_RDEN is synchronized to PLL_CLK for this attribute combination. Each bit of PHY_RDEN controls two UI worth of data:</p> <ul style="list-style-type: none"> • If RX_DATA_WIDTH = 8: [3:0] • If RX_DATA_WIDTH = 4: [2][0] • If RX_DATA_WIDTH = 2: not supported <p>When RX_GATING = DISABLE the gate is always open, regardless of the value of RX_DATA_WIDTH, CONTINUOUS_DQS, or PHY_RDEN.</p> <p>When SERIAL_MODE = TRUE, tie all four bits High.</p> <p>See Bidirectional Datapath for more information.</p>
PHY_WRCS0	[3:0]	Input	PLL_CLK	This port is only for memory-related use.
PHY_WRCS1	[3:0]	Input	PLL_CLK	This port is only for memory-related use.

Table 44: XPHY Ports (cont'd)

Port Name	Width	Input/Output	Clock Domain	Description
PHY_WREN	[3:0]	Input	PLL_CLK	<p>When TBYTE_CTL_x is set to PHY_WREN, the PHY_WREN input is inverted and serialized by the XPHY before being used as the tristate control signal (T_OUT[x]) for NIBBLESlice[x]. The inverted and serialized PHY_WREN is output on T_OUT synchronously with the TX data. Unlike T, which has each of its bits mapped to a NIBBLESlice, PHY_WREN is applied to each NIBBLESlice in a nibble. Each bit of PHY_WREN controls the tristate enable of two UIs worth of data. PHY_WREN is not supported when TX_DATA_WIDTH = 2. See Controlling Tristate Control for more information.</p> <p>When TX_GATING = ENABLE, PHY_WREN gates the TX datapath of NIBBLESlice[0], NIBBLESlice[2], NIBBLESlice[3], NIBBLESlice[4], and NIBBLESlice[5]. NIBBLESlice[1] cannot be gated. Set the following bits of PHY_WREN to 0 to gate transmit data or 1 to not gate transmit data (PHY_WREN is serialized but not inverted when used for gating).</p> <ul style="list-style-type: none"> • If TX_DATA_WIDTH = 8: [3:0] • If TX_DATA_WIDTH = 4: [2][0] • If TX_DATA_WIDTH = 2: not supported <p>See Bidirectional Datapath for more information.</p>
PLL_CLK		Input	-	<p>Clocks the XPHY interface (clocks within the interface are generated from this). Must be connected to CLKOUTPHY of an XPLL capable of routing to the nibble.</p> <p>If PLL_CLK is less than 500 MHz, DELAY_VALUE_<0-5> and VTC are not supported.</p> <p>Set the REFCLK_FREQUENCY attribute to the same value as the PLL_CLK frequency.</p>
RIU_ADDR	[7:0]	Input	CTRL_CLK	Address bus to access RIU
RIU_NIBBLE_SEL		Input	CTRL_CLK	Assert to perform read/write operations on the RIU
RIU_WR_DATA	[15:0]	Input	CTRL_CLK	Write data for the RIU
RIU_WR_EN		Input	CTRL_CLK	Assert high to enable writes to the RIU
RST		Input	ASYNC	Resets the entire XPHY nibble, including all RX datapaths, TX datapaths, and delays. While RST is asserted, all TX IOBs and the tristate control signal are set to the values of TX_INIT_# and TX_INIT_TRI, respectively.

Table 44: XPHY Ports (cont'd)

Port Name	Width	Input/Output	Clock Domain	Description
RXTX_SEL	[5:0]	Input	CTRL_CLK	<p>When 0, RXTX_SEL applies CE, INC, LD, and CNTVALUEIN to the input delay. Similarly, the input delay value is reported by CNTVALUEOUT.</p> <p>When 1, RXTX_SEL applies CE, INC, LD, and CNTVALUEIN to the output delay. Similarly, the output delay value is reported by CNTVALUEOUT.</p>
RX_EN_VTC	[5:0]	Input	ASYNC	<p>Assert to enable the Alignment and Delay Calibration steps of BISC to be performed on the input delays. If EN_VTC is also asserted, VTC will be performed on the input delays.</p> <p>When updating an input or output delay in NIBBLESLICE[x] (regardless of whether from the PL or RIU), both RX_EN_VTC[x] and TX_EN_VTC[x] must be set to 0.</p>
RX_RST	[5:0]	Input	ASYNC	<p>Assert to reset RX data paths on a per-NIBBLESLICE basis. Does not reset the input delay.</p>
T	[5:0]	Input	ASYNC	<p>When TBYTE_CTL_x = T, T[x] is used as the tristate control signal. T is a combinatorial route and is not synchronized to the TX data.</p>
TX_EN_VTC	[5:0]	Input	ASYNC	<p>Assert to enable the Delay Calibration step of BISC to be performed on the output delays. If EN_VTC is also asserted, VTC will be performed on the output delays.</p> <p>When updating an input or output delay in NIBBLESLICE[x] (regardless of whether from the PL or RIU), both RX_EN_VTC[x] and TX_EN_VTC[x] must be set to 0.</p>
TX_RST	[5:0]	Input	ASYNC	<p>Assert to reset TX data paths on a per-NIBBLESLICE basis. Does not reset the output delay. While TX_RST[x] is asserted, the TX IOB of NIBBLESLICE[x] and the tristate control signal are set to the values of TX_INIT_x and TX_INIT_TRI, respectively.</p>
BISC_START_OUT		Output	-	<p>This is a simulation-only port required for BISC.</p> <p>If an interface is composed of multiple nibbles, connect this port to BISC_START_IN of the previous nibble in the daisy chain.</p> <p>If an interface is composed of only one nibble or this port is on the first nibble in the daisy chain, this port does not need to be connected anywhere.</p>
BISC_STOP_OUT		Output	-	<p>This is a simulation-only port required for BISC.</p> <p>If an interface is composed of multiple nibbles, connect this port to BISC_STOP_IN of the next nibble in the daisy chain.</p> <p>If an interface is composed of only one nibble or this port is on the last nibble in the daisy chain, connect this port to BISC_START_IN of the same nibble.</p>

Table 44: XPHY Ports (cont'd)

Port Name	Width	Input/Output	Clock Domain	Description
CLK_TO_LOWER		Output	-	<p>Inter-byte clock output to certain numerically lower nibbles. The exception to this naming scheme is using CLK_TO_LOWER for inter-byte clocking from XPHY nibble 6 to XPHY nibble 8.</p> <p>In being part of inter-byte clocking, CLK_TO_LOWER can only be connected to CLK_FROM_OTHER_XPHY of another nibble and cannot be connected to the programmable logic. The Clocking section lists the nibbles that are capable of inter-byte clocking with each other.</p>
CLK_TO_UPPER		Output	-	<p>Inter-byte clock output to certain numerically higher nibbles. The exception to this naming scheme is using CLK_TO_LOWER for inter-byte clocking from XPHY nibble 6 to XPHY nibble 8.</p> <p>In being part of inter-byte clocking, CLK_TO_UPPER can only be connected to CLK_FROM_OTHER_XPHY of another nibble and cannot be connected to the programmable logic. The Clocking section lists the nibbles that are capable of inter-byte clocking with each other.</p>
CNTVALUEOUT	[53:0]	Output	CTRL_CLK	<p>The delay value (in taps) of the input or output delay selected by RXTX_SEL. Each NIBBLESlice is associated with 9b of CNTVALUEOUT. NIBBLESlice[0] is associated with CNTVALUEOUT[8:0], NIBBLESlice[1] with CNTVALUEOUT[17:9], NIBBLESlice[2] with CNTVALUEOUT[26:18], NIBBLESlice[3] with CNTVALUEOUT[35:27], NIBBLESlice[4] with CNTVALUEOUT[44:36], and NIBBLESlice[5] with CNTVALUEOUT[53:45].</p>
DLY_RDY		Output	ASYNC	<p>Indicates that delay lines (input, output, quarter (QTR), and coarse (CRSE)) can now be changed. If using BISC, it also indicates that the alignment and delay calibration steps are completed. If multiple nibbles compose an interface, the assertion time for DLY_RDY can vary for each nibble. Within simulation, the assertion time of DLY_RDY will not vary for each nibble in an interface, but will vary as the XPHY configuration and connections change.</p>
DYN_DCI	[5:0]	Output	ASYNC	<p>DYN_DCI controls turning off/on receiver termination for NIBBLESlice[x]. DYN_DCI can only be used with buffers that have the DCITERMDISABLE port.</p> <p>See Controlling IBUF_DISABLE and DYN_DCI for more information.</p>

Table 44: XPHY Ports (cont'd)

Port Name	Width	Input/Output	Clock Domain	Description
FIFO_EMPTY		Output	See description	Asserts when the FIFO is empty or the read and write pointers are at the same FIFO location. FIFO_EMPTY's clock domain depends upon the value of the FIFO_MODE_x attribute: <ul style="list-style-type: none"> When FIFO_MODE_x = ASYNC, FIFO_EMPTY is in the FIFO_RD_CLK domain See Controlling FIFO Modes for information on how FIFO_EMPTY should be controlled.
FIFO_WR_CLK		Output	-	The FIFO write clock. For source-synchronous receive interfaces with DQS_SRC = LOCAL, this is generated internally from DATAIN[0] or inter-nibble clocking. If DQS_SRC = EXTERN, FIFO_WR_CLK is generated internally from inter-byte clocking. If using serial mode, FIFO_WR_CLK is generated internally from the PLL_CLK input. See Clocking for more information.
GT_STATUS		Output	ASYNC	This port is only for memory-related use.
IBUF_DISABLE	[5:0]	Output	ASYNC	IBUF_DISABLE[x] controls disabling the receiver in NIBBLESlice[x]. See Controlling IBUF_DISABLE and DYN_DCI for more information.
NCLK_NIBBLE_OUT		Output	-	N-clk output to certain nibbles for inter-nibble clocking. In being part of inter-nibble clocking, NCLK_NIBBLE_OUT can only be connected to NCLK_NIBBLE_IN of another nibble and cannot be connected to the programmable logic. The Clocking section lists the nibbles that are capable of inter-nibble clocking.
O0	[5:0]	Output	PLL_CLK	Serialized TX data to the IOB. O0[0] is the data from D0 after it has been serialized by NIBBLESlice[0], O0[1] is the data from D1 after it has been serialized by NIBBLESlice[1], and so on. The Controlling Tristate Control section describes how D<0-5> is serialized and output on O0, and how it changes based on data width.
PCLK_NIBBLE_OUT		Output	-	P-clk output to certain nibbles for inter-nibble clocking. In being part of inter-nibble clocking, PCLK_NIBBLE_OUT can only be connected to PCLK_NIBBLE_IN of another nibble and cannot be connected to the programmable logic. The Clocking section lists the nibbles that are capable of inter-nibble clocking.

Table 44: XPHY Ports (cont'd)

Port Name	Width	Input/Output	Clock Domain	Description
PHY_RDY		Output	ASYNC	Indicates that the XPHY is ready for VTC. EN_VTC must be asserted for the first PHY_RDY assertion. Afterwards, if EN_VTC is deasserted, PHY_RDY will stay asserted. If EN_VTC is then reasserted, PHY_RDY will deassert and then reassert. Refer to the Controlling Built-in Self-Calibration section for VTC control.
Q0	[7:0]	Output	FIFO_RD_CLK when FIFO_MODE_0 = ASYNC	Deserialized RX data from the FIFO or from bypassing the FIFO. Q0 is the word created from DATAIN[0]. The Controlling FIFO Modes section details how DATAIN[0] maps to Q0.
Q1	[7:0]	Output	FIFO_RD_CLK when FIFO_MODE_1 = ASYNC	Deserialized RX data from the FIFO or from bypassing the FIFO. Q1 is the word created from DATAIN[1]. The Controlling FIFO Modes section details how DATAIN[1] maps to Q1.
Q2	[7:0]	Output	FIFO_RD_CLK when FIFO_MODE_2 = ASYNC	Deserialized RX data from the FIFO or from bypassing the FIFO. Q2 is the word created from DATAIN[2]. The Controlling FIFO Modes section details how DATAIN[2] maps to Q2.
Q3	[7:0]	Output	FIFO_RD_CLK when FIFO_MODE_3 = ASYNC	Deserialized RX data from the FIFO or from bypassing the FIFO. Q3 is the word created from DATAIN[3]. The Controlling FIFO Modes section details how DATAIN[3] maps to Q3.
Q4	[7:0]	Output	FIFO_RD_CLK when FIFO_MODE_4 = ASYNC	Deserialized RX data from the FIFO or from bypassing the FIFO. Q4 is the word created from DATAIN[4]. The Controlling FIFO Modes section details how DATAIN[4] maps to Q4.
Q5	[7:0]	Output	FIFO_RD_CLK when FIFO_MODE_5 = ASYNC	Deserialized RX data from the FIFO or from bypassing the FIFO. Q5 is the word created from DATAIN[5]. The Controlling FIFO Modes section details how DATAIN[5] maps to Q5.
RIU_RD_DATA	[15:0]	Output	CTRL_CLK	Read data from the RIU
RIU_RD_VALID		Output	CTRL_CLK	Asserts when the user has control of the RIU bus. RIU writes by BISC take precedence over RIU writes from the PL. If an RIU write from the PL collides with one from BISC, RIU_RD_VALID will deassert and the PL RIU write from the cycle before RIU_RD_VALID deasserted will be stored. After BISC finishes its write(s), RIU_RD_VALID will assert and the stored PL RIU write will be executed. Any writes from the PL while RIU_RD_VALID is Low will be discarded.

Table 44: XPHY Ports (cont'd)

Port Name	Width	Input/Output	Clock Domain	Description
T_OUT	[5:0]	Output	PLL_CLK (when TBYTE_CTL_x = PHY_WREN)/ASYNC (when TBYTE_CTL_x = T]	When TBYTE_CTL_x = T: T_OUT[x] is the T[x] input. When TBYTE_CTL_x = PHY_WREN: T_OUT[x] is the inverted and serialized PHY_WREN and is synchronous with the TX data.

Attributes

For attributes ending in 0-5, each one represents the NIBBLESLICE with the matching number. For example, CASCADE_<0-5> represents CASCADE_0, CASCADE_1 ... CASCADE_5. Accordingly, CASCADE_0 is associated with NIBBLESLICE[0], CASCADE_1 is associated with NIBBLESLICE[1], and so on.

Table 45: XPHY Attributes

Attribute	Per Nibble/Per NIBBLESLICE	Values	Default Value	Description
CASCADE_<0-5>	NIBBLESLICE	TRUE, FALSE	FALSE	TRUE: Cascades the input and output delay of a NIBBLESLICE, which doubles the available delay from 512 taps/625 ps to 1024 taps/1250 ps. Cascading is only possible for the receiver and renders the TX datapath of that NIBBLESLICE inoperable outside of its output delay used for the cascading. When changing the delay, it is recommended to store half of the desired delay value in the input delay and half in the output delay. FALSE: No cascade, only the input delay is used for the receiver.

Table 45: XPHY Attributes (cont'd)

Attribute	Per Nibble/Per NIBBLESLICE	Values	Default Value	Description
CONTINUOUS_DQS	Nibble	TRUE, FALSE	FALSE	<p>Used with the RX_GATING attribute and the PHY_RDEN port to gate the strobe entering on NIBBLESLICE[0] or from inter-byte clocking.</p> <p>The capture clock must be continuous (not a strobe) to set CONTINUOUS_DQS = ENABLE. Because the capture clock is continuous and because there is no timing requirement between PHY_RDEN and the capture clock, the gate enable and disable is not cycle accurate to the capture clock (RX_GATING = ENABLE, CONTINUOUS_DQS = TRUE in this case). In contrast, when RX_GATING = ENABLE and CONTINUOUS_DQS = FALSE, gate enable and disable can be trained to be deterministic.</p> <p>Setting CONTINUOUS_DQS = TRUE requires that 3 strobe cycles be received prior to receiving data to prevent data loss.</p> <p>See the PHY_RDEN description in Ports and refer to Bidirectional Datapath for more information.</p>
CRSE_DLY_EN	Nibble	TRUE, FALSE	FALSE	<p>TRUE: In addition to the normal quarter delays applied to the strobe, a coarse delay is applied. Set CRSE_DLY_EN to TRUE if RX_CLK_PHASE_P/N = SHIFT_90, the interface receives edge-aligned clock and data, and the interface is below 1 GHz. Using coarse delays requires that SELF_CALIBRATE = ENABLE and that both RX_CLK_PHASE_P and RX_CLK_PHASE_N be set to the same value as one another (both SHIFT_90 or both SHIFT_0).</p> <p>FALSE: A coarse delay will not be applied.</p> <p>The CRSE delay can only be used with a low frequency (200 MHz – 1 GHz) PLL_CLK, edge-aligned source-synchronous receiver interface to center the strobe to data.</p>

Table 45: XPHY Attributes (cont'd)

Attribute	Per Nibble/Per NIBBLESlice	Values	Default Value	Description
DELAY_VALUE_<0-5>	NIBBLESlice	0-625 ps/1250 ps	0	<p>DELAY_VALUE_x is the delay in ps that is used for the input and output delay of NIBBLESlice[x]. If CASCADE_x = TRUE, the maximum value of DELAY_VALUE_x extends from 625 ps to 1250 ps.</p> <p>DELAY_VALUE_x must be set to 0 when TX_OUTPUT_PHASE_90_x = TRUE and on NIBBLESlices receiving a strobe (only DELAY_VALUE_0 needs to be set to 0, regardless of whether the strobe is single-ended or differential). A non-zero delay requires SELF_CALIBRATE = ENABLE.</p> <p>For interfaces with PLL_CLK below 500 MHz, DELAY_VALUE is not supported. See Controlling Delays for more information.</p>
DIS_IDLY_VT_TRACK	Nibble	TRUE, FALSE	FALSE	<p>TRUE: Disables VTC on input delays.</p> <p>FALSE: Enables VTC on input delays. Following the steps outlined in Controlling Built-in Self-Calibration is still required for VTC to operate.</p>
DIS_ODLY_VT_TRACK	Nibble	TRUE, FALSE	FALSE	<p>TRUE: Disables VTC on output delays.</p> <p>FALSE: Enables VTC on output delays. Following the steps outlined in Controlling Built-in Self-Calibration is still required for VTC to operate.</p>
DIS_QDLY_VT_TRACK	Nibble	TRUE, FALSE	FALSE	<p>TRUE: Disables VTC on QTR delays.</p> <p>FALSE: Enables VTC on QTR delays. Following the steps outlined in Controlling Built-in Self-Calibration is still required for VTC to operate.</p>
DQS_MODE	Nibble	DDR3, DDR4_1TCK, DDR4_2TCK, LPDDR4_TOGGLE, LPDDR4	DDR4_1TCK	This attribute is only for memory-related use.
DQS_SRC	Nibble	LOCAL, EXTERN	LOCAL	<p>LOCAL: Set to LOCAL if the strobe is sourced from the connecting IOB, if its source and destination nibble are the same (the latter would require TXRX_LOOPBACK_x = TRUE for NIBBLESlice[x]), or if SERIAL_MODE = TRUE.</p> <p>EXTERN: Set to EXTERN if the strobe enters through inter-byte clocking.</p> <p>DQS_SRC can be set to either value for nibbles receiving a strobe through inter-nibble clocking.</p>
EN_CLK_TO_LOWER	Nibble	ENABLE, DISABLE	DISABLE	<p>ENABLE: Enables inter-byte clocking to a numerically lower nibble.</p> <p>DISABLE: Disables inter-byte clocking to a numerically lower nibble.</p>

Table 45: XPHY Attributes (cont'd)

Attribute	Per Nibble/Per NIBBLESlice	Values	Default Value	Description
EN_CLK_TO_UPPER	Nibble	ENABLE, DISABLE	DISABLE	ENABLE: Enables inter-byte clocking to a numerically higher nibble DISABLE: Disables inter-byte clocking to a numerically higher nibble
EN_DYN_DLY_MODE	Nibble	TRUE, FALSE	FALSE	This attribute is only for memory-related use.
EN_OTHER_NCLK	Nibble	TRUE, FALSE	FALSE	TRUE: Enables sourcing the n-side of the strobe from inter-nibble clocking. FALSE: Disables sourcing the n-side of the strobe from inter-nibble clocking.
EN_OTHER_PCLK	Nibble	TRUE, FALSE	FALSE	TRUE: Enables sourcing the p-side of the strobe from inter-nibble clocking. FALSE: Disables sourcing the p-side of the strobe from inter-nibble clocking.
FAST_CK	Nibble	TRUE, FALSE	FALSE	This attribute is only for memory-related use.
FIFO_MODE_<0-5>	NIBBLESlice	ASYNC, SYNC, BYPASS	ASYNC	ASYNC: Set to ASYNC if the read and write clocks of the FIFO in NIBBLESlice[x] are the same frequency but phase independent. SYNC: Reserved BYPASS: Reserved
IBUF_DIS_SRC_<0-5>	NIBBLESlice	EXTERNAL, INTERNAL	EXTERNAL	See Controlling IBUF_DISABLE and DYN_DCI for more information.
INV_RXCLK	Nibble	TRUE, FALSE	FALSE	TRUE: Inverts the incoming strobe from the IOB to NIBBLESlice[0], which also affects all nibbles that the clock is routed to as a result of inter-nibble/byte clocking. INV_RXCLK only affects the n-clk. FALSE: Strobe is not inverted. This is the only supported INV_RXCLK value when DQS_SRC = EXTERN
LP4_DQS	Nibble	TRUE, FALSE	FALSE	This attribute is only for memory-related use.
ODELAY_BYPASS_<0-5>	NIBBLESlice	TRUE, FALSE	FALSE	This attribute is only for memory-related use.
ODT_SRC_<0-5>	NIBBLESlice	EXTERNAL, INTERNAL	EXTERNAL	See Controlling IBUF_DISABLE and DYN_DCI for more information.
PRIME_VAL		1'b0, 1'b1	1'b0	This attribute is only for memory-related use.
REFCLK_FREQUENCY	Nibble	200.0–4266.0 [MHz]	200.0 [MHz]	Set to the frequency of the PLL_CLK input.

Table 45: XPHY Attributes (cont'd)

Attribute	Per Nibble/Per NIBBLESlice	Values	Default Value	Description
RX_CLK_PHASE_N	Nibble	SHIFT_0, SHIFT_90	SHIFT_0	<p>SHIFT_0: Set to SHIFT_0 to not apply a phase shift to the n-clk.</p> <p>SHIFT_90: Set to SHIFT_90 to apply a positive 90° phase shift (relative to the data) to the n-clk. This phase shift is not retained for inter-nibble clocking. Setting RX_CLK_PHASE_N = SHIFT_90 requires that SELF_CALIBRATE = ENABLE. At least one DELAY_VALUE_x must be set to 0 in nibbles where RX_CLK_PHASE_N = SHIFT_90.</p>
RX_CLK_PHASE_P	Nibble	SHIFT_0, SHIFT_90	SHIFT_0	<p>SHIFT_0: Set to SHIFT_0 to not apply a phase shift to the p-clk.</p> <p>SHIFT_90: Set to SHIFT_90 to apply a positive 90° phase shift (relative to the data) to the p-clk. This phase shift is not retained for inter-nibble clocking. Setting RX_CLK_PHASE_P = SHIFT_90 requires that SELF_CALIBRATE = ENABLE. At least one DELAY_VALUE_x must be set to 0 in nibbles where RX_CLK_PHASE_P = SHIFT_90.</p>
RX_DATA_WIDTH	Nibble	2, 4, 8	8	Set 1:2, 1:4, or 1:8 deserialization.
RX_GATING	Nibble	DISABLE, ENABLE	DISABLE	Used with the RX_DATA_WIDTH and CONTINUOUS_DQS attributes, as well as the PHY_RDEN port, to gate the strobe entering on NIBBLESlice[0] or from inter-byte clocking. RX_GATING is necessary for bidirectional clocks which can be tristated when not active, or for strobes that are indeterministic on start-up or during normal operation. See the description for PHY_RDEN for more information.
SELF_CALIBRATE	Nibble	DISABLE, ENABLE	ENABLE	<p>ENABLE: Enables BISC (proper assertion of EN_VTC, RX_EN_VTC, and TX_EN_VTC is still required)</p> <p>DISABLE: Disables BISC.</p>
SERIAL_MODE	Nibble	TRUE, FALSE	FALSE	<p>TRUE: Set to TRUE for interfaces where data is received without an accompanying source synchronous clock (strobe). In this case, the capture clock is generated internally from the PLL_CLK input.</p> <p>FALSE: Set to FALSE for source-synchronous interfaces.</p>

Table 45: XPHY Attributes (cont'd)

Attribute	Per Nibble/Per NIBBLESlice	Values	Default Value	Description
TBYTE_CTL_<0-5>	Nibble, NIBBLESlice	PHY_WREN, T	T	<p>T: Set to T if using the T input (combinatorial path from the PL) as the tristate control signal for NIBBLESlice[x]. If TX_DATA_WIDTH = 2, only T (not PHY_WREN) can be used as the tristate control signal.</p> <p>PHY_WREN: Set to PHY_WREN if using the PHY_WREN input as the tristate control signal for NIBBLESlice[x]. The serialized and inverted PHY_WREN is synchronized with the TX data. Each bit of PHY_WREN controls the tristate for two bits of data.</p> <p>T and PHY_WREN can both be applied to the same NIBBLESlice, and the NIBBLESlice selects which one to use based on its TBYTE_CTL_x setting.</p>
TXRX_LOOPBACK_<0-5>	NIBBLESlice	TRUE, FALSE	FALSE	<p>TRUE: Set to TRUE to loop back the TX output to RX input of the same NIBBLESlice[x]. The TX output will still reach the IOB. A buffer (IBUF, OBUF, IOBUF, or a variant of one of the three) must be used for loopback to work. Connect the net from XPHY.O0[x] to the input of an OBUF, IOBUF, or variant of one of the two. Alternatively, connect the output net from an IBUF, IOBUF, or a variant of one of the two to XPHY.DATAIN[x]. Regardless of which net is used, only use one buffer (IBUF, OBUF, or variant) in this path.</p> <p>FALSE: Set to FALSE for no loopback.</p> <p>See Controlling TX to RX Loopback for more information.</p>
TX_DATA_WIDTH	Nibble	2, 4, 8	8	<p>Set 2:1, 4:1, or 8:1 serialization. If TX_DATA_WIDTH = 2, any IOBs connected to the nibble cannot use PRE_EMPHASIS.</p>
TX_GATING	Nibble	DISABLE, ENABLE	DISABLE	<p>DISABLE: The TX datapath of NIBBLESlice[0], NIBBLESlice[2], NIBBLESlice[3], NIBBLESlice[4], and NIBBLESlice[5] are not gated. NIBBLESlice[1] cannot be gated through this attribute, regardless of its setting.</p> <p>ENABLE: Use PHY_WREN to gate the TX datapath of NIBBLESlice[0], NIBBLESlice[2], NIBBLESlice[3], NIBBLESlice[4], and NIBBLESlice[5]. NIBBLESlice[1] cannot be gated.</p>

Table 45: XPHY Attributes (cont'd)

Attribute	Per Nibble/Per NIBBLESLICE	Values	Default Value	Description
TX_INIT_<0-5>	NIBBLESLICE	1'b1, 1'b0	1'b0	1'b0: Sets the TX IOB value associated with NIBBLESLICE[x] during configuration and reset (in this case, TX_RST[x] = 1 or RST = 1) to 1'b0. 1'b1: Sets the TX IOB value associated with NIBBLESLICE[x] during configuration and reset (in this case, TX_RST[x] = 1 or RST = 1) to 1'b1.
TX_INIT_TRI	Nibble	1'b1, 1'b0	1'b1	1'b0: Sets the tristate control signal during configuration and reset (in this case, TX_RST[x] = 1 or RST = 1) to 1'b0. 1'b1: Sets the tristate control signal during configuration and reset (in this case, TX_RST[x] = 1 or RST = 1) to 1'b1.
TX_OUTPUT_PHASE_90_<0-5>	NIBBLESLICE	TRUE, FALSE	FALSE	Used for TX signals to center align the transmit clock to data. Set to TRUE for clock and FALSE for data to achieve a center-aligned clock and data relationship. TRUE: Applies a positive 90° phase shift to the TX output of NIBBLESLICE[x]. DELAY_VALUE_x should be set to 0 when TX_OUTPUT_PHASE_90_x = TRUE. FALSE: Set to FALSE to not apply the 90° shift.
TX_OUTPUT_PHASE_90_TRI	Nibble	TRUE, FALSE	FALSE	Used for center-aligned DDR TX signals where tristate control is required. Only applicable when TBYTE_CTL_x = PHY_WREN. Each NIBBLESLICE being tristated must have its associated DELAY_VALUE_x set to 0. TRUE: Applies a positive 90° phase shift to T_OUT[x]. FALSE: Does not apply the 90° phase shift.
WRITE_LEVELING	Nibble	TRUE, FALSE	FALSE	This attribute is only for memory-related use.

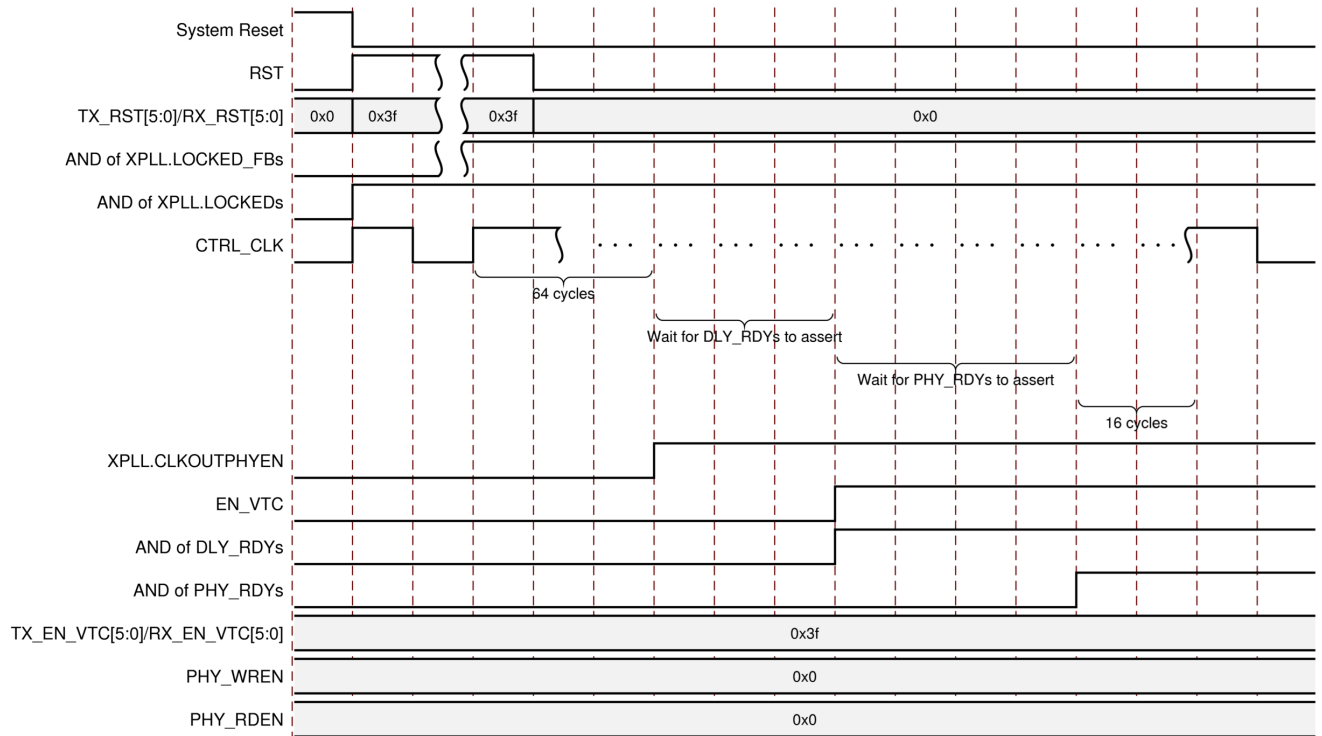
Reset Sequence

An XPHY nibble can be reset through the process below. For interfaces that span multiple nibbles, apply the same sequence to all nibbles. If a NIBBLESLICE[x] is not used, tie TX_RST[x] and RX_RST[x] to 1. Note that NIBBLESLICE[0] must be used for proper output delay calibration. If NIBBLESLICE[0] is not used (implying that TX_RST[0] and RX_RST[0] would normally be tied to 1) but output delays are used in other NIBBLESLICES in the nibble, tie both TX_RST[0] and RX_RST[0] (not just TX_RST[0]) to 0 instead of 1.

The reset sequence for XPHY nibbles with SERIAL_MODE = FALSE (this is associated with source-synchronous designs) is shown below. Unless otherwise marked, all signals refer to the XPHY:

- Always keep PHY_WREN and PHY_RDEN deasserted during the reset sequence.
 - Always keep TX_EN_VTC and RX_EN_VTC asserted during the reset sequence.
 - RST, TX_RST, and RX_RST start as asserted.
 - XPLL.CLKOUTPHYEN starts as deasserted, and should be deasserted whenever RST is asserted or upon device power-on.
 - EN_VTC starts as deasserted. If external calibration is not required and VTC is being used, EN_VTC can be tied off to 1.
1. Wait for XPLL.LOCKED_FB to assert. If multiple XPLLs are used to clock the interface, AND together each XPLL.LOCKED_FB of the XPLLs used to clock the interface.
 2. After the result of [step 1](#) asserts, deassert RST, TX_RST, and RX_RST on all nibbles that compose the interface.
 3. Wait 64 CTRL_CLK cycles after the XPLL.LOCKED_FB AND gate asserts, then assert XPLL.CLKOUTPHYEN on all XPLLs used to clock the interface.
 4. AND the DLY_RDY from each nibble in the interface, and synchronize the result in your application's clock domain.
 5. After the AND of all DLY_RDYs in the interface from step (4) is 1, assert EN_VTC on all nibbles that compose the interface. If VTC is not used or REFCLK_FREQUENCY < 500MHz (for which EN_VTC should be tied to 0, and VTC and DELAY_VALUE_x are not supported), instead of asserting EN_VTC skip to [step 7](#) after the AND of all DLY_RDYs in the interface from [step 4](#) is 1.
 6. AND the PHY_RDY from each nibble in the interface and wait for the output of this AND gate to be 1.
 7. Wait 16 CTRL_CLK cycles.
 8. After the 16th cycle, the reset sequence is considered complete if XPLL.LOCKED of all XPLLs that comprise the interface is 1. If XPLL.LOCKED_FB deasserts at any time, place all nibbles and XPLLs that compose the interface back into a reset state, defined by the bullet points prior to this sequence.

Figure 22: Source-Synchronous Reset Sequence

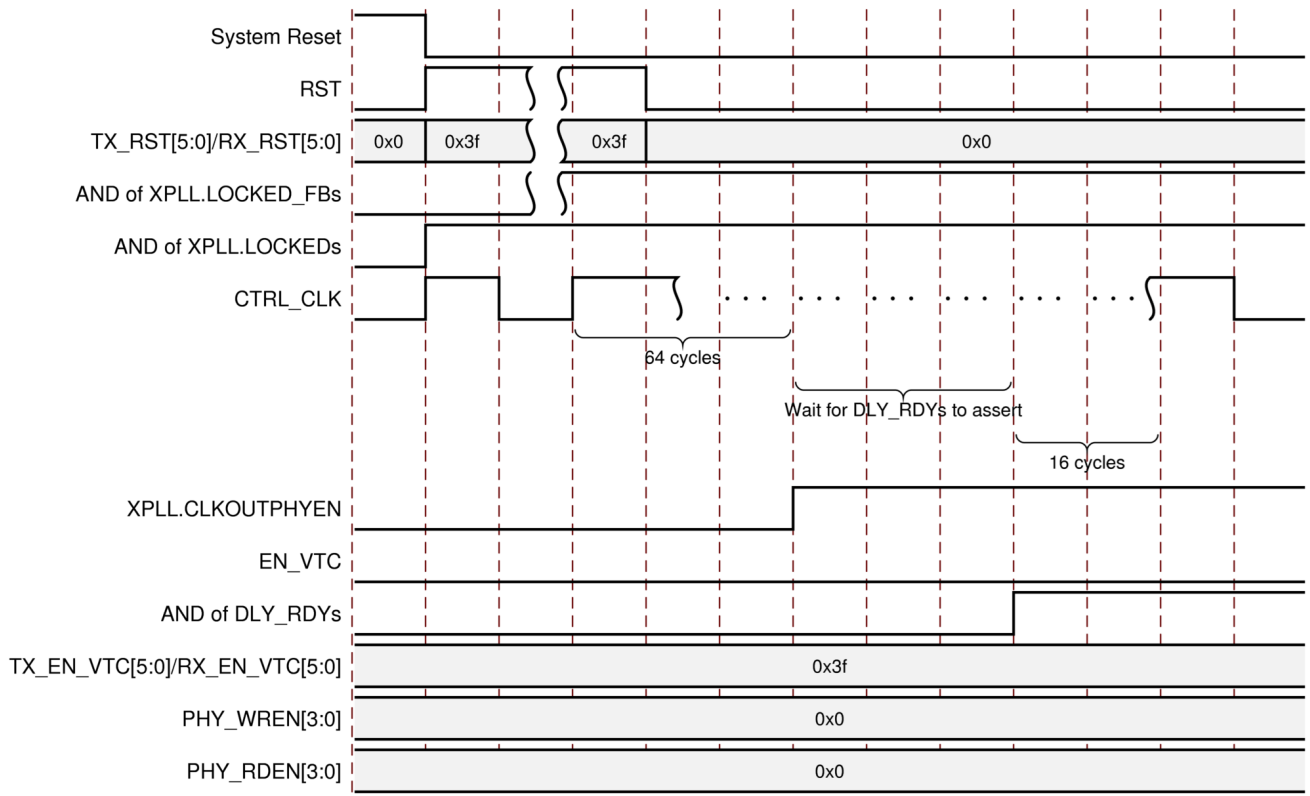


The reset sequence for XPHY nibbles with SERIAL_MODE = TRUE (any design that is not source-synchronous) is as follows. Unless otherwise marked, all signals refer to the XPHY:

- Always keep PHY_WREN and PHY_RDEN deasserted during the reset sequence.
 - Always keep TX_EN_VTC and RX_EN_VTC asserted during the reset sequence.
 - RST, TX_RST, and RX_RST start as asserted.
 - XPLL.CLKOUTPHYEN starts as deasserted, and should be deasserted whenever RST is asserted or upon device power-on.
 - This sequence assumes VTC is not used. Thus, EN_VTC should be deasserted or tied off to 1'b0.
1. Wait for XPLL.LOCKED_FB to assert. If multiple XPLLs are used to clock the interface, AND together each XPLL.LOCKED_FB of the XPLLs used to clock the interface.
 2. After the result of step (1) asserts, deassert RST, TX_RST, and RX_RST on all nibbles that comprise the interface.
 3. Wait 64 CTRL_CLK cycles after the XPLL.LOCKED_FB AND gate asserts, then assert XPLL.CLKOUTPHYEN on all XPLLs used to clock the interface.
 4. AND the DLY_RDY from each nibble in the interface, and synchronize the result in your application's clock domain.

- After the result of step (4) asserts, wait an additional 16 CTRL_CLK cycles. After the 16th cycle, the reset sequence is considered complete if XPLL.LOCKED of all XPLLs that comprise the interface is 1. If XPLL.LOCKED_FB deasserts at any time, place all nibbles and XPLLs that comprise the interface back into a reset state, defined by the bullet points prior to this sequence.

Figure 23: Serial Mode Reset Sequence




TIP: If multiple nibbles comprise an interface, the assertion time for DLY_RDY can vary for each nibble. Within simulation, the assertion time of DLY_RDY does not vary for each nibble in an interface, but can vary from interface to interface.

XPHY Usage

All high-performance interfaces must be accessed using the Advanced I/O Wizard (see *Advanced I/O Wizard LogiCORE IP Product Guide (PG320)*). It constructs all connections (including inter-nibble and inter-byte clocking), handles the reset sequence, and optimizes the XPHY and I/O hardware. The wizard has setup options for synchronous and asynchronous interfaces, and multiple instances can be built with multiple instantiations of the wizard. The XPHY

NIBBLESLICEs not used by the wizard are available to the XP IOL or for direct IOB feed-through to the programmable logic. A pin cannot be connected simultaneously to both an XPHY nibble and directly to fabric. In conjunction with the Advanced I/O Wizard, the Advanced I/O Planner simplifies XPHY pin planning with a GUI approach to place and constrain any XPHY based interface.

 **IMPORTANT!** *LVC MOS is not compatible with XPHY.*

To aid in setting input and output delays to a time-based value, the following Tcl commands can be used to set them through the top-level port connecting to DATAIN or O0 rather than setting the DELAY_VALUE_# attribute. Like the DELAY_VALUE_# attribute, setting DELAY_VALUE_XPHY on a NIBBLESLICE changes the value of both the input *and* output delay lines:

```
set_property DELAY_VALUE_XPHY <desired delay in ps> [get_ports <top-level
port(s) that ultimately connect to DATAIN or O0>]
```

The command above prepares the delays to be applied, then the command below applies them. Thus, you can issue multiples of the command above before finalizing all of them through the following `implement_xphy_cores` command:

```
implement_xphy_cores -update_delay_value_only
```

XP IOL Resources

Although each I/O pin in an XP bank has access to an XPHY for high-speed interfaces, not all interfaces require the performance and logic associated with the XPHY. For I/O interfaces at speeds below 500 Mb/s, the XP IOL provides register-based interface logic. The XP IOL supports both single and double data rate registering of output, input, and tristate control. The IOL provides coarse alignment through the IODELAY block or one of four clock managers.

XP IOL Features

Single Data Rate Flip-Flops

Each pin contains local single-data rate (SDR) flip-flop registers in the IOL for both data and tristate lines. Both output, input, and tristate SDR flip-flops use the following blocks:

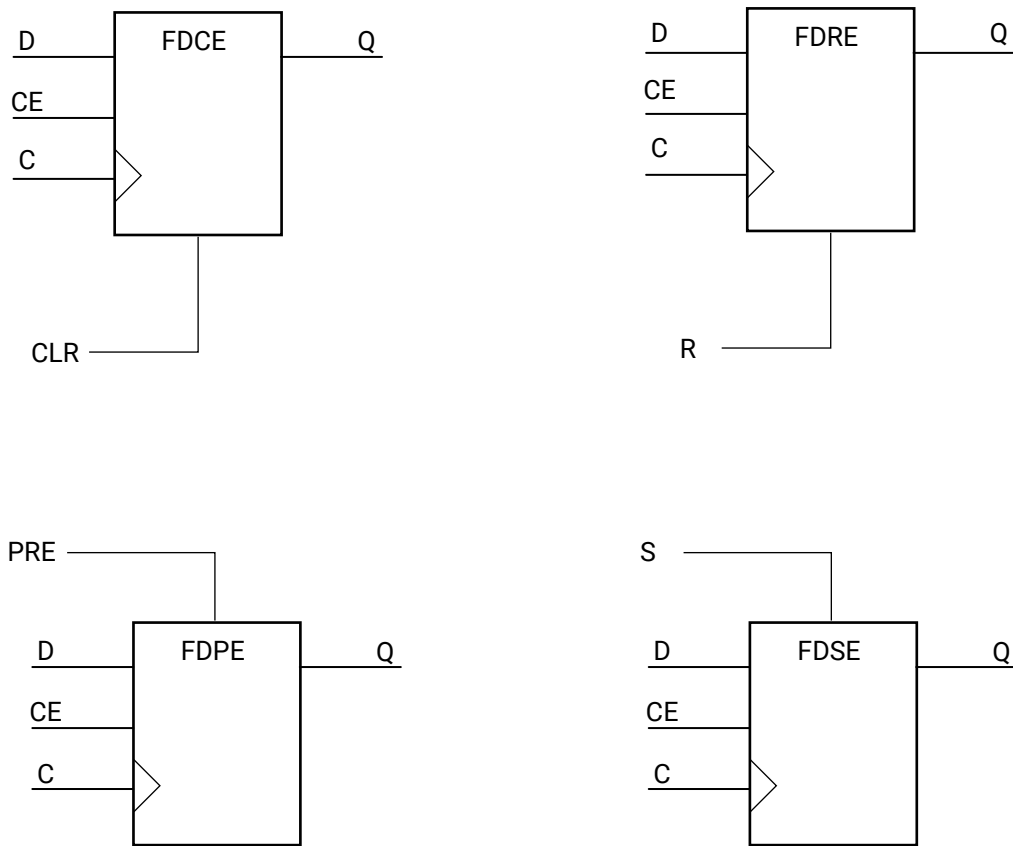
- **FDCE:** Flip-flop with clock enable and asynchronous clear
- **FDPE:** Flip-flop with clock enable and asynchronous preset
- **FDRE:** Flip-flop with clock enable and synchronous reset
- **FDSE:** Flip-flop with clock enable and synchronous set

Note: Both data and tristate registers must be used in the same manner when they are used together (that is both SDR, DDR, or not registered at all).

Single Data Rate Flip-Flop Primitives

SDR input and output registering inside the IOL use a flip-flop primitive in with the IOB = TRUE constraint applied to the flip-flop instance.

Figure 24: Single Data Rate Flip-Flop Primitives



X21636-092418

Table 46: Single Data Rate Flip-Flop (FDCE, FDPE, FDRE, FDSE) Attributes

Attribute	Values	Description
INIT	1'b0, 1'b1	Defines the initial value of the flip-flop

Table 47: Single Data Rate Flip-Flop (FDCE, FDPE, FDRE, FDSE) Ports

Port	I/O	Description
Q	Output	Data output
C	Input	Clock input pin
CE	Input	Active-High clock enable register
D	Input	Data input
CLR	Input	Asynchronous clear (FDCE only)
PRE	Input	Asynchronous preset (FDPE only)
R	Input	Synchronous reset (FDRE only)
S	Input	Synchronous set (FDSE only)

Double Data Rate Flip-Flops

Each pin contains local double-data rate (DDR) flip-flop registers in the IOL for both data and tristate lines. IDDRE1 should be used for instantiating DDR flip-flops in the input path while ODDRE1 should be used for output and tristate paths.

Note: Both data and tristate registers must be used in the same manner (that is both SDR, DDR, or not registered at all).

IDDR Modes

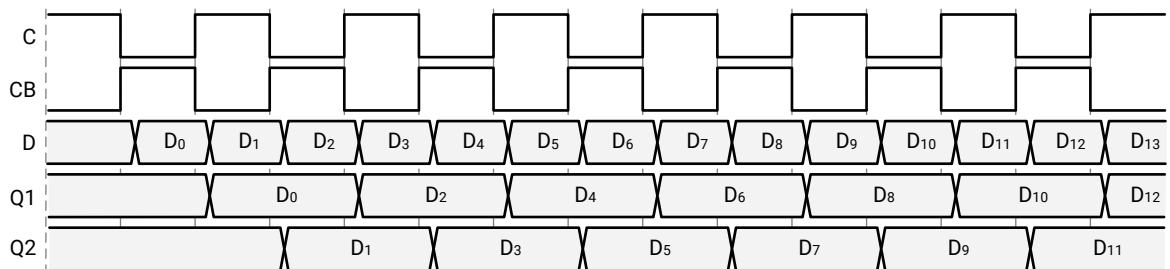
Versal™ devices have dedicated registers in the IOL to implement input DDR registers. This feature is used by instantiating the IDDRE1 primitive. The IDDRE1 primitive supports the following modes of operation:

- **OPPOSITE_EDGE:** Traditional input DDR solution. Data is presented to Q1 on the rising edge and Q2 on the falling edge.
- **SAME_EDGE:** Data is presented to the device logic on the same clock edge.
- **SAME_EDGE_PIPELINED:** Data is presented to the device logic on the same clock edge. Removes the separated effect but incurs clock latency.

The SAME_EDGE and SAME_EDGE_PIPELINED modes allow designers to transfer falling edge data to the rising edge domain within the IOL, saving configurable logic block (CLB) and clock resources and increasing performance. These modes are implemented using the DDR_CLK_EDGE attribute. The following sections describe each of the operation modes in detail.

OPPOSITE_EDGE mode, a traditional input DDR solution, is accomplished using a single input in the XIOL block. The data is presented to the device logic through the output Q1 on the rising edge of the clock and the output Q2 on the falling edge of the clock. This structure is similar to the previous FPGA implementation. The following timing diagram shows the input DDR using the OPPOSITE_EDGE mode.

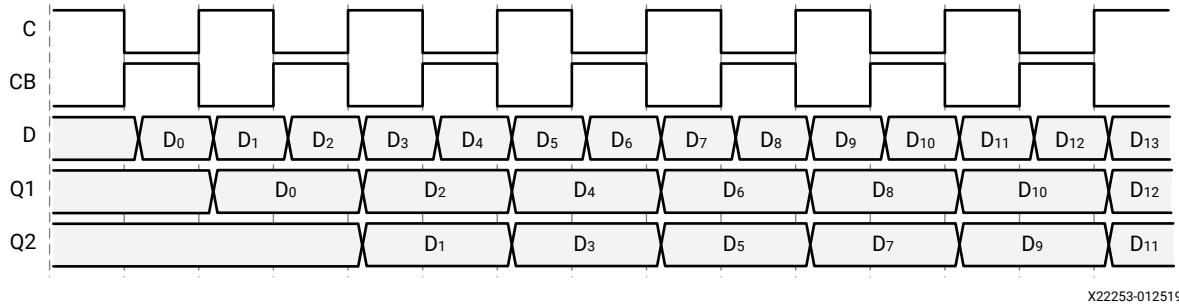
Figure 25: **OPPOSITE_EDGE** Mode



X22252-012519

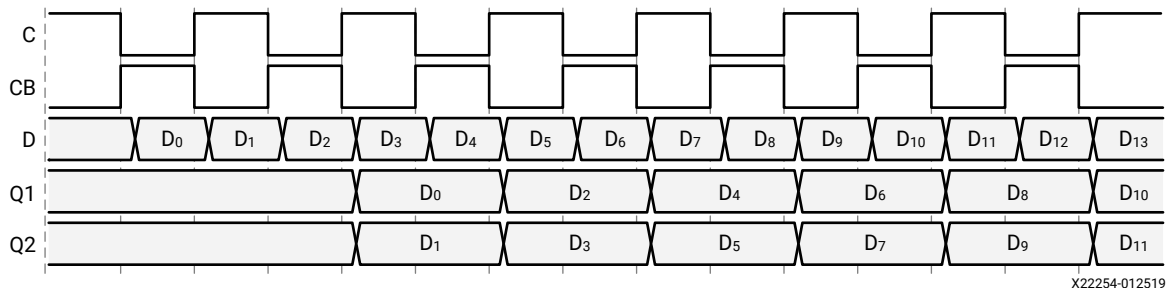
In SAME_EDGE mode, the data is presented into the device logic on the same clock edge. The following timing diagram shows the input DDR using the SAME_EDGE mode. In the timing diagram, the output pairs Q1 and Q2 are no longer (0) and (1). Instead, the first pair presented is pair Q1 (0) and Q2 (don't care), followed by pair (1) and (2) on the next clock cycle

Figure 26: SAME_EDGE Mode



In SAME_EDGE_PIPELINED mode, the data is presented into the device logic on the same clock edge. Unlike the SAME_EDGE mode, the data pair is not separated by one clock cycle. However, additional clock latency is required to remove the separated effect of the SAME_EDGE mode. The following timing diagram shows the input DDR using the SAME_EDGE_PIPELINED mode. The output pairs Q1 and Q2 are presented to the device logic at the same time.

Figure 27: SAME_EDGE_PIPELINED Mode



Double Data Rate Input Flip-Flop Primitive

Figure 28: IDDR1 Primitive

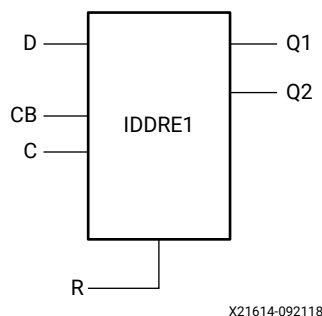


Table 48: IDDRE1 Attributes

Attribute	Values	Description
DDR_CLK_EDGE	OPPOSITE_EDGE, SAME_EDGE, SAME_EDGE_PIPELINED	Sets the IDDRE1 mode of operation with respect to the clock edge
IS_C_INVERTED	0 or 1	Sets a local clock inversion for C input when 1.
IS_CB_INVERTED	0 or 1	Sets a local clock inversion for CB input. When IS_CB_INVERTED=1, C and CB must be driven by the same global clock buffer. When IS_CB_INVERTED=0, CB must be driven by the same global clock buffer through an inverter.

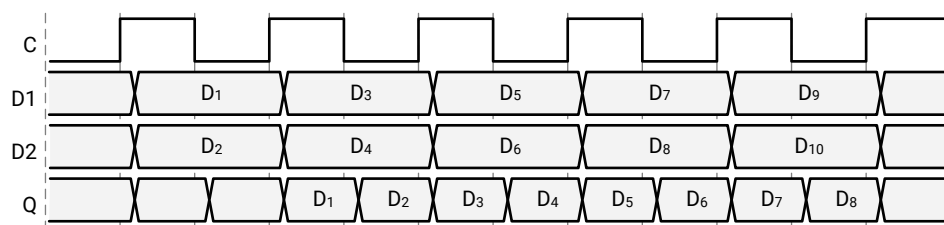
Table 49: IDDRE1 Ports

Port	I/O	Description
Q1, Q2	Output	IDDRE1 register outputs
C	Input	Clock input pin
CB	Input	Inverted clock input pin when IS_C_INVERTED = 0 and IS_CB_INVERTED = 0
D	Input	Register input from IOB
R	Input	Asynchronous High reset

ODDR Mode

Versal™ devices have registers in the IOL to implement output DDR registers as in previous generations. This feature is accessed when instantiating the ODDRE1 primitive. DDR multiplexing is automatic when using the ODDRE1. No manual control of the multiplexer select is needed. This control is generated from the clock. The ODDRE1 primitive supports only the SAME_EDGE mode of operation. The SAME_EDGE mode allows designers to present both data inputs to the ODDRE1 primitive on the rising edge of the ODDRE1 clock, which saves CLB and clock resources and increases performance. The following figure shows the timing diagram of the output DDR.

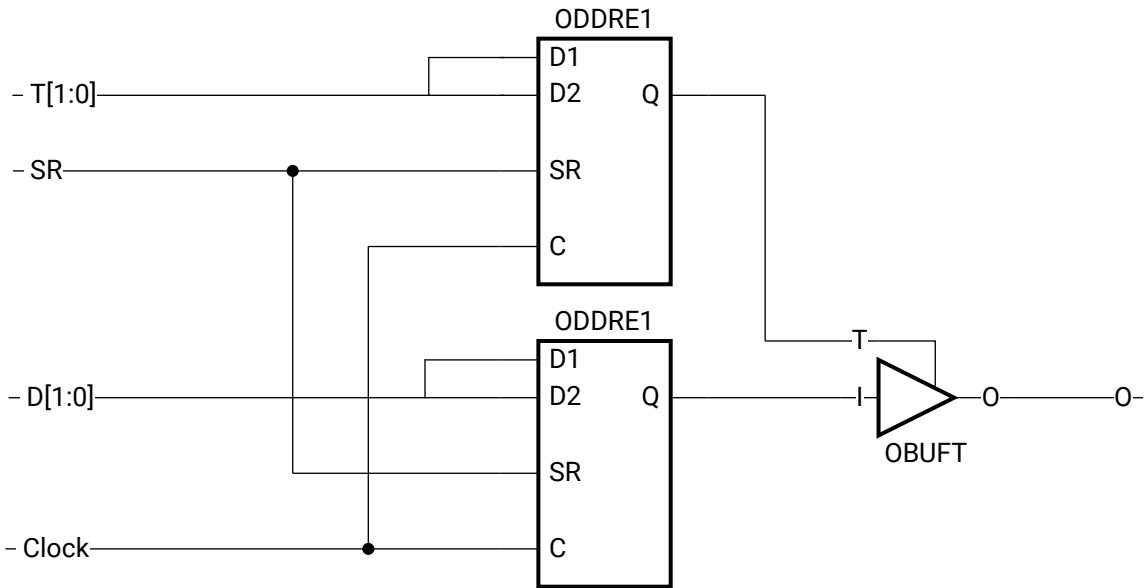
Figure 29: Output DDR Timing



X22249-012519

When using tristate control, the tristate and data paths must leverage the same registering structure. Thus, if the data path leverages the ODDRE1 block, the tristate must also leverage the ODDRE1 primitive.

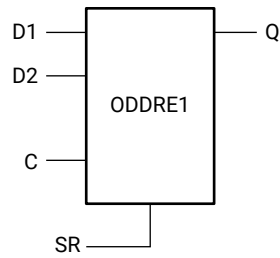
Figure 30: ODDR with ODDR Serialized Tristate



X22251-012519

Double Data Rate Output Flip-Flop Primitive

Figure 31: ODDRE1 Primitive



X21615-092118

Table 50: ODDRE1 Attributes

Attribute	Values	Description
SRVAL	1'b0, 1'b1	Initializes the value of the ODDRE1 flip-flops

Table 51: ODDRE1 Ports

Port	I/O	Description
Q	Output	ODDRE1 register output
C	Input	Clock input pin
D1, D2	Input	ODDRE1 register inputs
SR	Input	Asynchronous High reset

Uncalibrated IOB Delay

Each pin has two uncalibrated delay blocks: one for the input delay (IDELAYE5) and one for the output path (ODELAYE5). Each individual block provides an uncalibrated delay of up to approximately 1.8 ns for both data and tristate paths. When only the input portion of the IOB is used, the ODELAYE5 can be cascaded to the IDELAYE5 to provide the input path up to approximately 3.6 ns of delay. Both IDELAYE5 and ODELAYE5 have 32 taps that can be incremented or decremented using the INC and CE pins, or can be dynamically changed using the CNTVALUEIN and LOAD pins. With a 1.8 ns delay over 32 taps, each increment or decrement in the quantity of taps assigned to the delay element impacts the overall delay by around 56 ps per tap. Because IOL delay elements are not calibrated, precise delay values should not be expected. When using the ODELAY block with tristate control, the tristate path will automatically have the same delay as the data line.

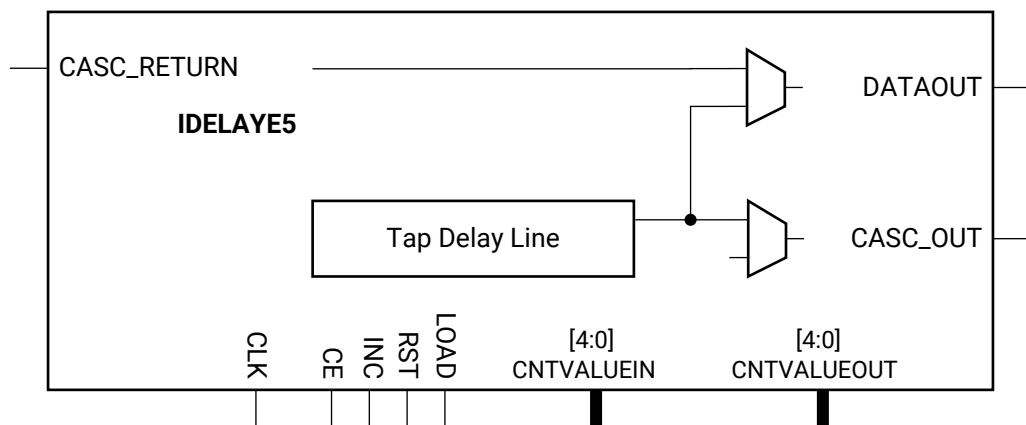
For both ODELAYE5 and IDELAYE5, the delay must be dynamically registered into the block at run time. The tap delays can not be preset with attributes for the IDELAYE5 or ODELAYE5.

Use Cases

In the IOL, IOB delays can be helpful in adjusting the clock to data timing relationship in an I/O interface. The IDELAYE5 and ODELAYE5 both must have delay levels registered at run time. Both the IDELAYE5 and ODELAYE5 use the same control ports to define the delay of the block. The CNTVALUEIN and LOAD ports are used to a specific delay value from 0 to 32 is known for the phase shift. The INC port can be used to incrementally increase or decrease the delay value from 0 to 32 taps. This is convenient when using a system that is intended to be calibrated during run time. Both methods for changing the delay value of the delay block are described in the following timing diagrams.

Note: The IDELAYE5 and ODELAYE5 will always come out of configuration with a delay value of 0 taps. The delay value must be updated by user logic via either the VARIABLE or LOAD mode.

Figure 32: IDELAYE5 Delay

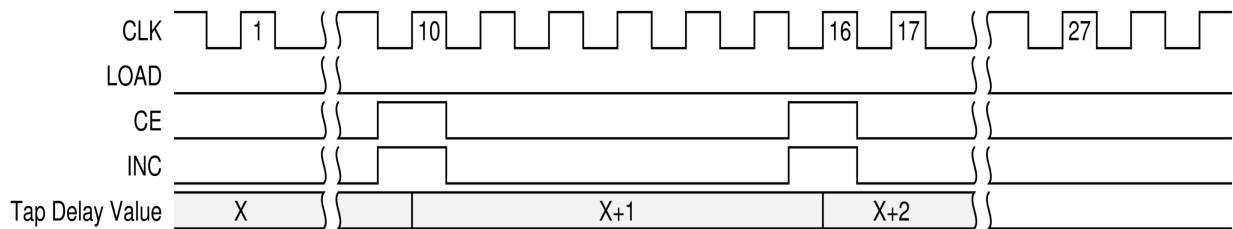


X22248-032419

Variable Mode

In variable mode, the CE and INC ports are used to manually increment and decrement the delay-line tap per tap (INC/DEC increments or decrements one tap at a time). The tap delay increments by setting CE = 1 and INC = 1, or decrements by CE = 1 and INC = 0. If the delay line incremented or decremented outside its valid values (i.e., the past the 32nd tap), the tap line will wrap around back to the smallest or largest tap value.

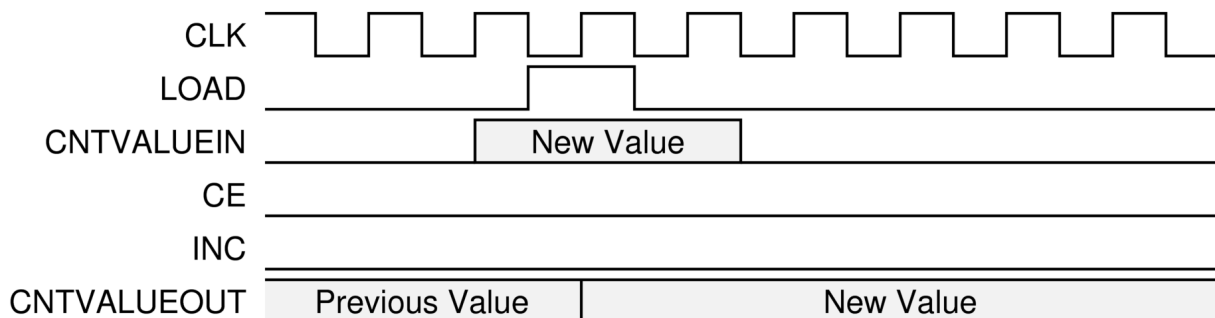
Figure 33: Variable Mode



Load Mode

In load mode, the CNTVALUEIN<4:0> and LOAD ports are used to dynamically set a specific tap delay weight to the tap delay line. In both cases the CNTVALUEOUT can be used to read the current position of the delay line.

Figure 34: Load Mode

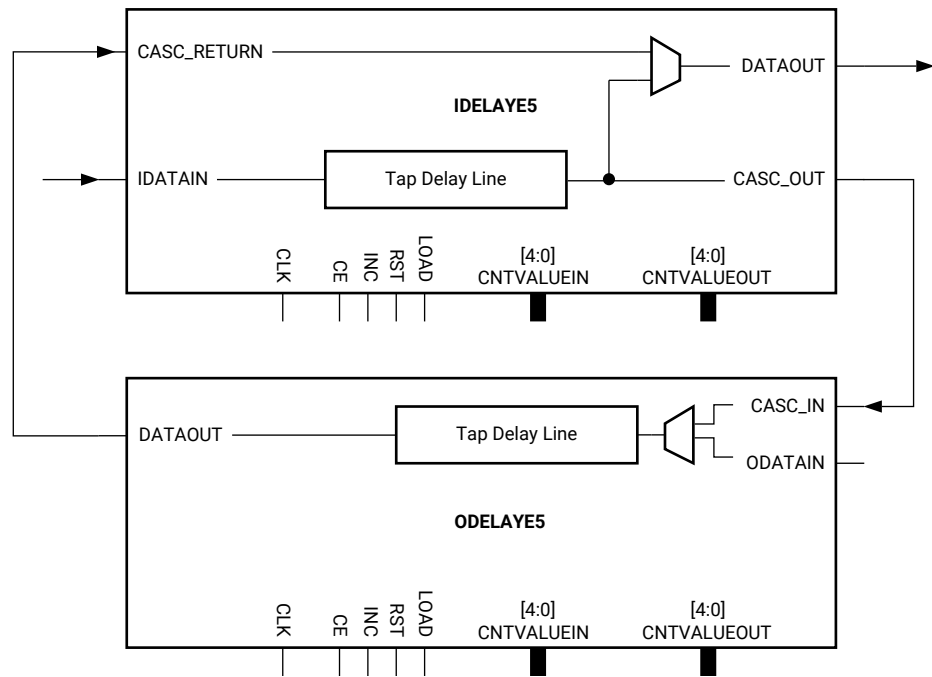


Cascade

The CASCADE attribute is set to NONE if the delay line is not cascaded. Cascading is used when a delay greater than 1.8 ns is required. The connections between delay elements are shown in the following figure. When using the IDELAYE5 and ODELAYE5 for cascading, the output is no longer available. Therefore, cascading delays is not supported for bidirectional buffers. The delay elements can only be cascaded within an IOB. Therefore, the maximum possible length of the

delay is approximately 3.6 ns. The routes used to cascade IDELAYE5 and ODELAYE5 are dedicated, high-speed routes. The total fixed intrinsic insertion delay for an IDELAYE5 or ODELAYE5 cascade is the sum of the initial insertion delay plus the cascaded insertion delay. The following figure shows the proper port connections for a cascaded IDELAYE5 and ODELAYE5

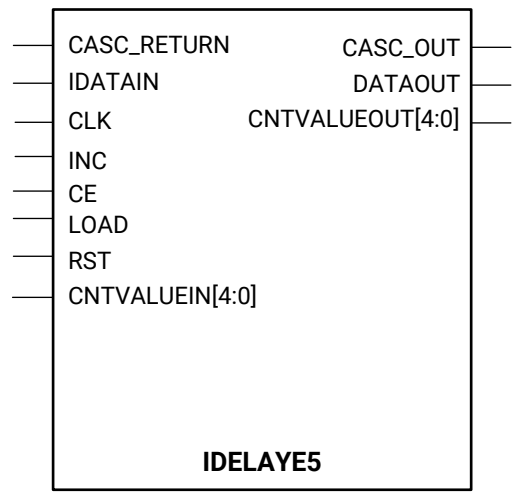
Figure 35: IDELAYE5 Cascading



X22247-013119

Uncalibrated Input Delay Primitive

Figure 36: IDELAYE5 Primitive



X22469-050519

Table 52: IDELAYE5 Attributes

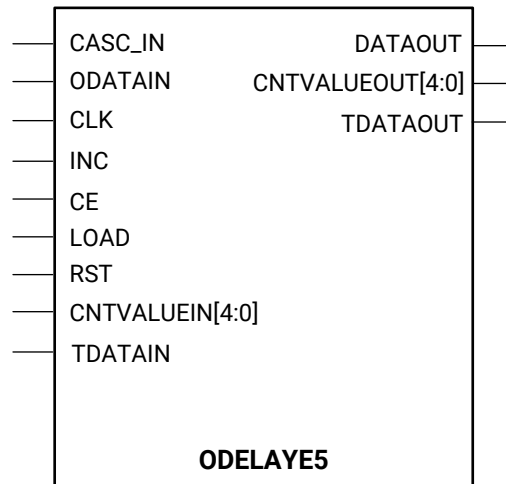
Attribute	Values	Description
CASCADE	FALSE, TRUE	The CASCADE attribute is set to TRUE when the ODELAYE5 is used to cascade the IDELAYE5. This attribute enables an input delay greater than 1.8 ns.

Table 53: IDELAYE5 Ports

Port	I/O	Description
DATAOUT	Output	Delayed data from the IOB
CNTVALUEOUT<4:0>	Output	The CNTVALUEOUT pins are used for reporting the current tap value and reads out the amount of taps in the current delay.
CASC_OUT	Output	Cascade delay to ODELAYE5 in cascade. The CASC_OUT pin is used when cascading from an IDELAYE5 to an ODELAYE5. The CASC_OUT port of the IDELAYE5 is connected to the CASC_IN of the ODELAYE5 in cascade.
IDATAIN	Input	Data from input pin
CASC_RETURN	Input	Cascade delay returning from the ODELAYE5 DATAOUT port. This port is used when cascading the ODELAYE5 and IDELAYE5.
CNTVALUEIN<4:0>	Input	The CNTVALUEIN pins are used for dynamically switching the loadable tap value. The CNTVALUEIN is the number of taps required.
CE	Input	Clock enable used in conjunction with INC port to increment (INC=1) or decrement (INC=0) the delay line. Leaving CE enabled for multiple CLK cycles will allow consecutive changes in the delay value.
CLK	Input	Clock used to sample LOAD, CE, and INC
INC	Input	Increment and decrement are controlled by the enable signal (CE). <ul style="list-style-type: none"> INC = 1 increments INC = 0 decrements
LOAD	Input	Load counter value from the CNTVALUEIN bus when High
RST	Input	The reset pin (RST) is asynchronous with the CLK

Uncalibrated Output Delay Primitive

Figure 37: ODELAYE5



X22468-071420

Table 54: ODELAYE5 Attributes

Attribute	Values	Description
CASCADE	FALSE, TRUE	The CASCADE attribute is set to TRUE when the ODELAYE5 is used to cascade the IDELAYE5. This attribute enables a delay greater than 1.8 ns.

Table 55: ODELAYE5 Ports

Port	I/O	Description
DATAOUT	Output	Delayed data to the pin. When cascading the ODELAYE5 to the IDELAYE5, DATAOUT should be connected to the CASC_RETURN pin of the IDELAYE5.
CNTVALUEOUT<4:0>	Output	The CNTVALUEOUT pins are used for reporting the current tap value and reads out the amount of taps in the current delay.
TDATAOUT	Output	Delayed tristate control connects to the IOB.
CASC_IN	Input	Cascade delay from IDELAYE5 CASC_OUT. The CASC_IN pin is used when cascading the ODELAYE5 delay to the IDELAYE5.
ODATAIN	Input	Delayed data to the IOB.
CNTVALUEIN<4:0>	Input	The CNTVALUEIN pins are used for dynamically switching the loadable tap value. The CNTVALUEIN is the number of taps required.
CE	Input	Clock enable used with INC port to increment (INC=1) or decrement (INC=0) the delay line. Leaving CE enabled for multiple CLK cycles will allow consecutive changes in the delay value.
CLK	Input	Clock used to sample LOAD, CE, and INC.
INC	Input	Increment and decrement are controlled by the enable signal (CE). <ul style="list-style-type: none"> INC = 1 increments INC = 0 decrements
LOAD	Input	Load counter value from the CNTVALUEIN bus when High.

Table 55: ODELAYE5 Ports (cont'd)

Port	I/O	Description
RST	Input	The reset pin (RST) is asynchronous with the CLK.
TDATAIN	Input	Tristate control input.

XP IOL Clock Managers

Each XPIO bank has four clock managers in or adjacent to the bank with the ability to phase shift the clock signal for optimal signal timing. In addition to the two XPLLs described in [Chapter 2: XPHY Architecture](#), an MMCM and DPLL can be used in the XP IOL to generate new clock frequencies and to eliminate skew between clock and data paths as they reach the IDDR and ODDR registers. More details on these features can be found in [Chapter 5: XP Bank Supporting Resources and Corner Banks](#).

XP IOB Resources

The XP IOB block provides resources to enable high-speed interfaces between the programmable logic (PL) and the system outside the device. The XP IOB resources are designed to accommodate the signaling needs for high-speed memory and chip-to-chip interfaces that are powered between 1.0V and 1.5V, including LVSTL06. The XP IOB provides internal termination, internal reference generation, support for a diverse set of I/O standards, driver emphasis, and receiver equalization. These features allow the XP IOB to integrate with a diverse range of systems.

XP IOB Banking Structure

The XPIO pins are grouped into banks of 54 IOBs (27 I/O pairs) with each IOB having associated XP IOL logic and shared XPHY logic. Within a bank, there are six pin groupings, defined as a nibble, which share XPHY logical resources, as shown in the following figure. Within a nibble, each pin has an associated pin pair that can be used to accommodate differential signaling standards. All IOBs in an XP bank share the same V_{CCO} power supply that is used to power driver logic, receiver logic, and termination. Regardless of whether the IOB is used as an input, output, or bidirectional pin, each I/O standard has a specific V_{CCO} voltage requirement that must be used for the I/O standard to populate a bank. Similarly, each pin in a nibble must share a compatible INTERNAL_VREF level with all the other pins in a nibble.


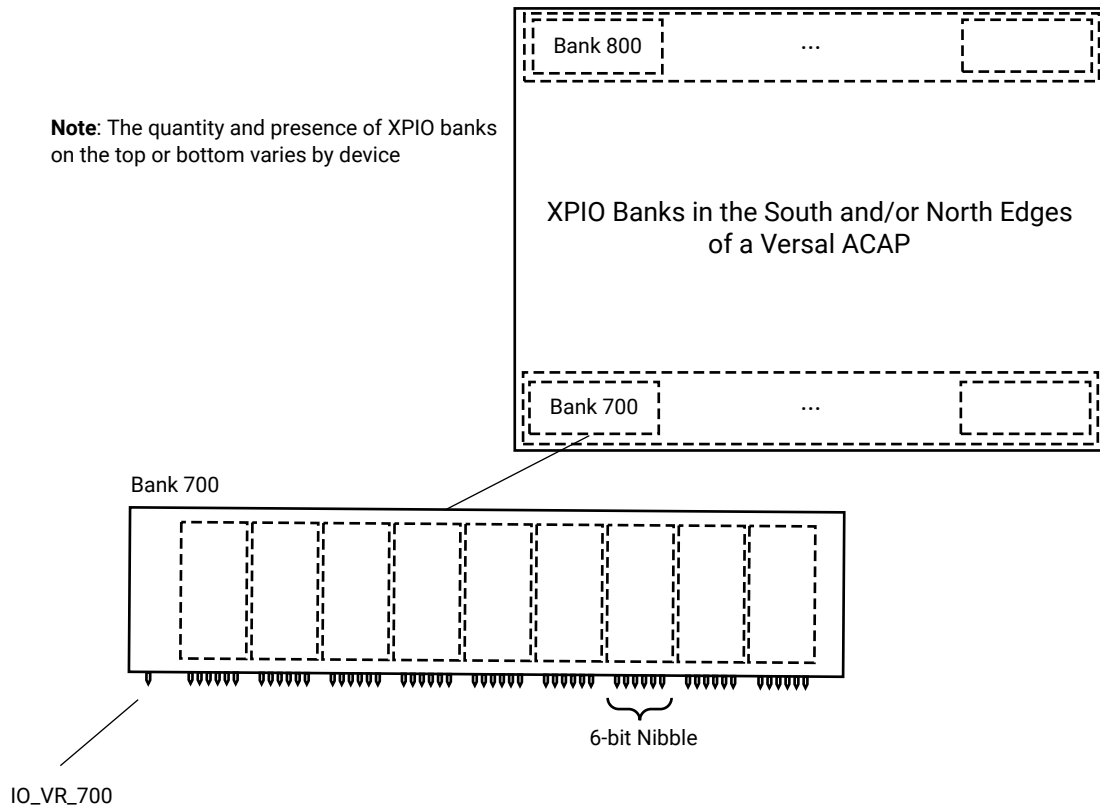
 **IMPORTANT!** Some XPIO banks (typically located on the corner of the device) have pins that have limited function and can only be used for DDR memory controller functionality. See the Versal ACAP Packaging and Pinouts Architecture Manual ([AM013](#)) for specific pin information.

Figure 38: XP Bank (54 IOB, 27 I/O Pairs, 9 Nibbles, 54 XIOL)




X21489-071320

XP IOB Supported Standards

See the [XP IOB Internal \$V_{REF}\$](#) section for more information on the INTERNAL V_{REF} , DRIVE, and TERMINATION options listed in the following table.

Table 56: Single-Ended Standards Available in the XP IOB

IOSTANDARD	Required V_{CCO} Level for Input and Output	Required INTERNAL_VREF Level for Input	Drive and Termination Options
LVCMOS12	1.2V	N/A	DRIVE: 2, 4, 6, 8
LVCMOS15	1.5V	N/A	DRIVE: 2, 4, 6, 8, 12
LVDCI_15	1.5V	N/A	OUTPUT_IMPEDANCE
LVSTL_11	1.1V	0.183V	OUTPUT_IMPEDANCE, ODT(SINGLE to GND)
LVSTL06_12	1.2V	0.125V	OUTPUT_IMPEDANCE, ODT(SINGLE to GND)
HSUL_12	1.2V	0.60V	OUTPUT_IMPEDANCE, ODT(SINGLE to VCCO)
HSLVDCI_15	1.5V	0.75V	OUTPUT_IMPEDANCE, ODT(SPLIT)
HSTL_I	1.5V	0.75V	OUTPUT_IMPEDANCE, ODT(SPLIT)
HSTL_I_12	1.2V	0.60V	OUTPUT_IMPEDANCE, ODT(SPLIT)
POD12	1.2V	0.84V	OUTPUT_IMPEDANCE, ODT(SINGLE to VCCO)
POD10	1.0V	0.70V	OUTPUT_IMPEDANCE, ODT(SINGLE to VCCO)
SSTL12	1.2V	0.60V	OUTPUT_IMPEDANCE, ODT(SPLIT)
SSTL15	1.5V	0.75V	OUTPUT_IMPEDANCE, ODT(SPLIT)
SSTL135	1.35V	0.675V	OUTPUT_IMPEDANCE, ODT(SPLIT)

 **IMPORTANT!** In the Vivado tools, the default standard assigned to an IOB is UNDEFINED (or DIFF_UNDEFINED for differential buffers). The IOSTANDARD UNDEFINED has no actual meaning in the IOB and the Vivado tools will not generate a valid programming file if it is not changed to a valid IOSTANDARD.

See the [Calibrated Termination \(Digitally Controlled Impedance\)](#) section for more information on the DRIVE and TERMINATION options listed in the following table.

Table 57: Differential Standards Available in the XP IOB

IOSTANDARD	V _{cco} Level	Drive and Termination Options
DIFF_SSTL15	1.5V	OUTPUT_IMPEDANCE, ODT(SPLIT)
DIFF_SSTL135	1.35V	OUTPUT_IMPEDANCE, ODT(SPLIT)
DIFF_SSTL12	1.2V	OUTPUT_IMPEDANCE, ODT(SPLIT)
DIFF_HSUL_12	1.2V	OUTPUT_IMPEDANCE, ODT(SINGLE to VCCO)
LVDS15	1.5V	DIFF_TERM
DIFF_HSTL_I	1.5V	OUTPUT_IMPEDANCE, ODT(SPLIT)
DIFF_HSTL_I_12	1.2V	OUTPUT_IMPEDANCE, ODT(SPLIT)
DIFF_LVSTL_11	1.1V	OUTPUT_IMPEDANCE, ODT(SINGLE to GND)
DIFF_LVSTL06_12	1.2V	OUTPUT_IMPEDANCE, ODT(SINGLE to GND)
DIFF_POD10	1.0V	OUTPUT_IMPEDANCE, ODT(SINGLE to VCCO)
DIFF_POD12	1.2V	OUTPUT_IMPEDANCE, ODT(SINGLE to VCCO)
MIPI_DPHY	1.2V	OUTPUT_IMPEDANCE, DIFF_TERM

XP IOB Primitives

The Vivado® Design Suite Library includes an extensive list of primitives supporting many I/O primitives. The generic primitives can each support most of the single-ended I/O Standards:

- **IBUFE3:** Input buffer with V_{REF} tuning, along with buffer disable control
- **IBUF:** Input buffer
- **IBUF_IBUFDISABLE:** Input buffer with buffer disable control
- **IOBUFE3:** Bidirectional buffer, V_{REF} tuning, input buffer disable, and on-die input termination enable control
- **IOBUF:** Bidirectional buffer
- **IOBUF_DCIEN:** Bidirectional buffer with input buffer disable and on-die input termination disable control
- **OBUF:** Output buffer
- **OBUFFT:** Tristate output buffer

These generic primitives can each support most of the available differential I/O standards:

- **IBUFDSE3:** Differential input buffer with offset calibration along with buffer disable control
- **IBUFDS:** Differential input buffer
- **IBUFDS_DIFF_OUT:** Differential input buffer with complementary outputs

- **IBUFDS_DIFF_OUT_IBUFDISABLE:** Differential input buffer with complementary outputs and buffer disable
- **IBUFDS_IBUFDISABLE:** Differential input buffer with buffer disable control
- **IBUFDS_DPHY:** Differential input buffer for the MIPI D-PHY
- **IOBUFDSSE3:** Differential bidirectional buffer with input buffer disable and on-die input termination enable control
- **IOBUFDS:** Differential bidirectional buffer
- **IOBUFDS_DCEN:** Differential bidirectional buffer with on-die input termination disable control and input buffer disable
- **IOBUFDS_DIFF_OUT:** Differential bidirectional buffer with complementary outputs from the input buffer
- **IOBUFDS_DIFF_OUT_DCEN:** Differential bidirectional buffer with complementary outputs from the input buffer with on-die input termination disable controls and input buffer disable controls
- **OBUFDS:** Differential output buffer
- **OBUFTDS:** Differential tristate output buffer
- **OBUFDS_DPHY:** Differential output buffer for the MIPI D-PHY
- **XPIO_VREF:** V_{REF} scan control

XP IOB Supported Single-Ended Standards

LVC MOS and LVDCI

The low-voltage CMOS standards (LVC MOS) is a widely used standard implemented with CMOS transistors. The LVDCI standard is a LVC MOS receiver with calibrated output impedance while the LVC MOS output impedance is defined by the DRIVE setting. This standard is defined in the JEDEC standard JESD 8C.01. LVC MOS12 (1.2V), LVC MOS15 (1.5V), and LVDCI_15 (1.5V) are supported in the XP IOB.

Table 58: Allowed Attributes for LVC MOS15 I/O Primitives

Attributes	IBUF/IBUFE3		OBUF/OBUFT/IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	LVC MOS15		LVC MOS15	
DRIVE	N/A		2, 4, 6, 8, 12	12
SLEW	N/A		FAST, MEDIUM, SLOW	SLOW

Table 59: Allowed Attributes for LVCMOS12 I/O Primitives

Attributes	IBUF/IBUFE3		OBUF/OBUFT/IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	LVCMOS12		LVCMOS12	
DRIVE	N/A		2, 4, 6, 8	8
SLEW	N/A		FAST, MEDIUM, SLOW	SLOW

Table 60: Allowed Attributes for LVDCI_15 I/O Primitives

Attributes	IBUF/IBUFE3		OBUF/OBUFT/IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	LVDCI_15		LVDCI_15	
OUTPUT_IMPEDANCE	N/A		RDRV_48_48	
SLEW	N/A		FAST, MEDIUM, SLOW	SLOW

LVSTL and HSUL

The low-voltage swing terminated logic (LVSTL_11 and LVSTL06_12) and high-speed unterminated logic (HSUL_12) standards are optimized for lower-power memory interfaces. LVSTL06_12 is compatible with LVSTL06 interfaces but is powered at 1.2V.

Table 61: Allowed Attributes for LVSTL_11 and LVSTL06_12 I/O Primitives

Attributes	IBUF/IBUFE3		OBUF/OBUFT		IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	LVSTL_11, LVSTL06_12		LVSTL_11, LVSTL06_12		LVSTL_11, LVSTL06_12	
SLEW	N/A		FAST, MEDIUM, SLOW	SLOW	FAST, MEDIUM, SLOW	SLOW
OUTPUT_IMPEDANCE	N/A		RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_40_40	RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_40_40
ODT	RTT_40, RTT_48, RTT_60	RTT_40	N/A		RTT_40, RTT_48, RTT_60	RTT_40
VOH (LVSTL_11 ONLY)	N/A		50	50	50	50
PRE_EMPHASIS	N/A				RDRV_240 (LVSTL_11 only), RDRV_NONE	RDRV_NONE
EQUALIZATION	EQ_LEVEL0, EQ_LEVEL1, EQ_LEVEL2, EQ_LEVEL3, EQ_LEVEL4, EQ_NONE	EQ_NONE	N/A		EQ_LEVEL0, EQ_LEVEL1, EQ_LEVEL2, EQ_LEVEL3, EQ_LEVEL4, EQ_NONE	EQ_NONE

Table 62: Allowed Attributes for HSUL_12 I/O Primitives

Attributes	IBUF/IBUFE3		OBUF/OBUFT		IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	HSUL_12		HSUL_12		HSUL_12	
SLEW	N/A		FAST, MEDIUM, SLOW	SLOW	FAST, MEDIUM, SLOW	SLOW
OUTPUT_IMPE DANCE	N//A		RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_48_48	RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_48_48
ODT	RTT_120, RTT_240, RTT_NONE	RTT_NONE	N/A		RTT_120, RTT_240, RTT_NONE	RTT_NONE

HSTL and HSLVDCI

HSTL_I (1.5V) and HSTL_I_12 (1.2V) are both high-speed transceiver logic (HSTL) standards used for the general-purpose high-speed bus JEDEC standard JESD8-6. Typically used for high-speed memory interfaces, these standards share the same receiver with the HSLVDCI_15 (1.5V) standard. The HSLVDCI_15 standard leverages a calibrated controlled impedance driver to allow operation without a split termination at the receiver, which are required by the HSTL standards.

Table 63: Allowed Attributes for HSTL_I and HSTL_I_12 I/O Primitives

Attributes	IBUF/IBUFE3		OBUF/OBUFT		IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	HSTL_I, HSTL_I_12		HSTL_I, HSTL_I_12		HSTL_I, HSTL_I_12	
ODT	RTT_40, RTT_48, RTT_60	RTT_48	N/A		RTT_40, RTT_48, RTT_60	RTT_48
OUTPUT_IMPE DANCE	N//A		RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_48_48	RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_48_48
SLEW	N/A		FAST, MEDIUM, SLOW	SLOW	FAST, MEDIUM, SLOW	SLOW

Table 64: Allowed Attributes for HSLVDCI_15 I/O Primitives

Attributes	IBUF/IBUFE3		OBUF/OBUFT		IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	HSLVDCI_15		HSLVDCI_15		HSLVDCI_15	
SLEW	N/A		FAST, MEDIUM, SLOW	SLOW	FAST, MEDIUM, SLOW	SLOW

POD

Pseudo open drain (POD) standards POD12 (1.2V) and POD10 (1.0V) are intended for DDR4 applications.

Table 65: Allowed Attributes for POD10 and POD12 I/O Primitives

Attributes	IBUF/IBUFE3		OBUF/OBUFT		IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Value	Default	Allowed Values	Default
IOSTANDARD	POD10, POD12		POD10, POD12		POD10, POD12,	
SLEW	N/A		FAST, MEDIUM, SLOW	SLOW	FAST, MEDIUM, SLOW	SLOW
OUTPUT_IMPEDANCE	N//A		RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_40_40	RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_40_40
ODT	RTT_40, RTT_48, RTT_60	RTT_40	N/A		N/A	
PRE_EMPHASIS	N/A		RDRV_240, RDRV_NONE	RDRV_NONE	RDRV_240, RDRV_NONE	RDRV_NONE
EQUALIZATION (POD12 ONLY)	EQ_LEVEL0, EQ_LEVEL1, EQ_LEVEL2, EQ_LEVEL3, EQ_LEVEL4, EQ_NONE	EQ_NONE			N/A	N/A

SSTL

The stub-series terminated logic (SSTL) I/O standards for 1.5V (SSTL15) and 1.35V (SSTL135) are used for general-purpose memory buses. SSTL15 is used for DDR3 SDRAM interfaces and is roughly defined (not by name) in the JEDEC standard JESD79-3E. SSTL135 is used for DDR3L SDRAM interfaces and is roughly defined (not by name) in the JEDEC standard JESD79-3-1.

Table 66: Allowed Attributes for SSTL15, SSTL135 and SSTL12 I/O Primitives

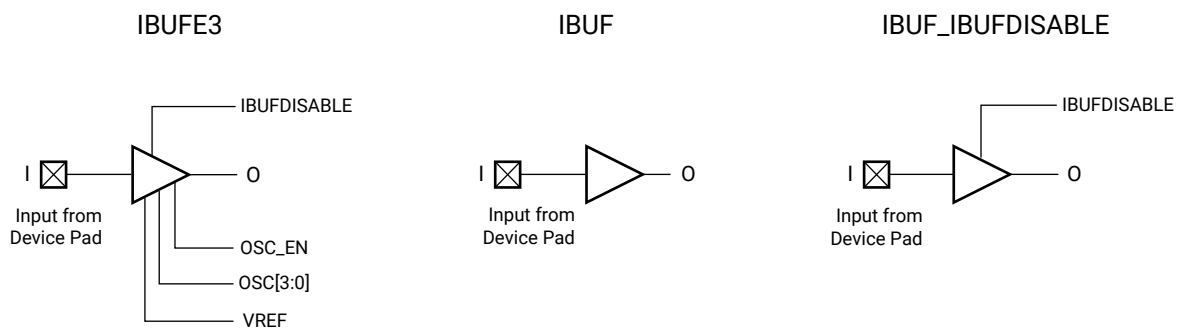
Attributes	IBUF/IBUFE3		OBUF/OBUFT		IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Value	Default	Allowed Values	Default
IOSTANDARD	SSTL15, SSTL135, SSTL12		SSTL15, SSTL135, SSTL12		SSTL15, SSTL135, SSTL12	
SLEW	N/A		FAST, MEDIUM, SLOW	SLOW	FAST, MEDIUM, SLOW	SLOW
OUTPUT_IMPEDANCE	N//A		RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_40_40	RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_40_40
ODT	RTT_40, RTT_48, RTT_60	RTT_40	N/A		RTT_40, RTT_48, RTT_60	RTT_40
VOH (SSTL15 ONLY)	N/A	N/A	75, 80	75	N/A	N/A

UNDEFINED Default IOSTANDARD

When an IOSTANDARD is not defined by the user, the default assignment for the IOSTANDARD defaults to UNDEFINED. For a Versal ACAP design to complete implementation, a non-default IOSTANDARD must be defined with one of the valid I/O standards described in this section. The UNDEFINED standard acts as a placeholder to allow a design to complete the early stages of implementation.

Single-Ended Input Buffer Primitives

Figure 39: Single-Ended Input Buffer Primitives



X21616-092118

Table 67: IBUFE3, IBUF, and IBUF_IBUFDISABLE Attributes

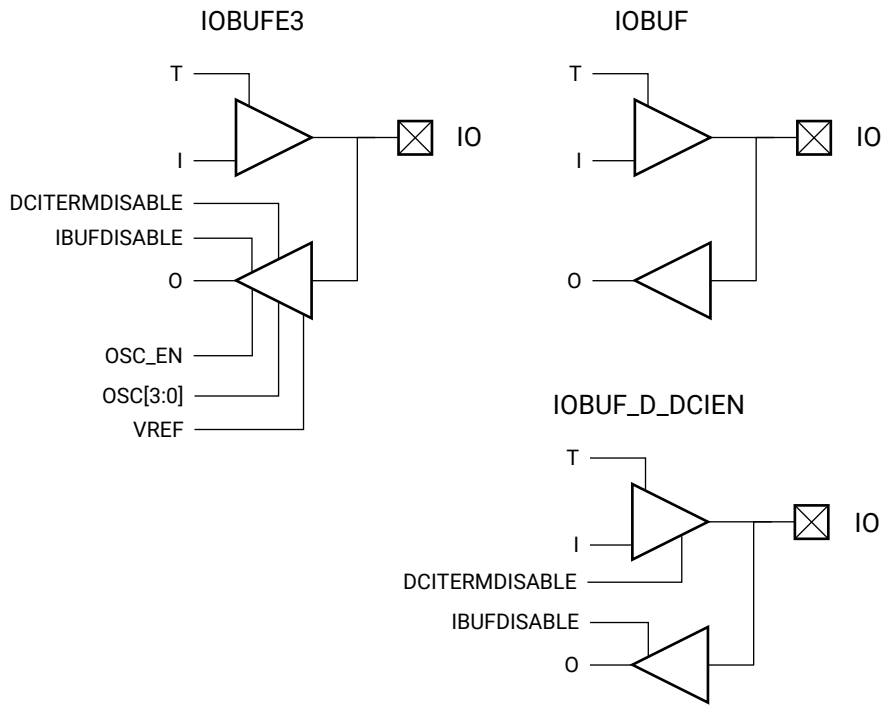
Attribute	Values	Description
IBUF_LOW_PWR	TRUE, FALSE	When set to TRUE, allows for reduced power input standards that require INTERNAL_VREF (for example: HSTL). A setting of FALSE demands more power but delivers higher performance characteristics.
IOSTANDARD	See XP IOB Supported Standards	Assigns an I/O standard to the element.
USE_IBUFDISABLE	TRUE, FALSE	Enables IBUFDISABLE port (IBUFE3, IBUF_IBUFDISABLE only)

Table 68: IBUFE3, IBUF, and IBUF_IBUFDISABLE Ports

Port	I/O	Description
O	Output	Buffer output representing the input path to the device.
I	Input	Input port connection. Connects directly to top-level port in the design.
IBUFDISABLE	Input	Disables the input buffer and forces the O output to the internal logic to a logic Low when asserted High (IBUFE3, IBUF_IBUFDISABLE only).
OSC[3:0]	Input	This is not a supported port and must be left unconnected.
OSC_EN[1:0]	Input	This is not a supported port and must be left unconnected.
VREF	Input	Connect to XPIO_VREF (IBUFE3 only).

Single-Ended Bidirectional Buffer Primitives

Figure 40: Single-Ended Bidirectional Buffer Primitives



X21624-092318

Table 69: IOBUFE3, IOBUF, and IOBUF_DDCIEN Attributes

Attribute	Values	Description
IBUF_LOW_PWR	TRUE, FALSE	When set to TRUE, allows for reduced power input standards that require INTERNAL_VREF (like HSTL). A setting of FALSE demands more power but delivers higher performance characteristics (IOBUFE3, IOBUF_DDCIEN only).
DRIVE	2, 4, 8, 12	Specifies the drive strength of the output.
SLEW	SLOW, FAST, MEDIUM	Specifies the slew rate of the output.
IOSTANDARD	See XP IOB Supported Standards	Assigns an I/O standard to the element.
USE_IBUFDISABLE	TRUE, FALSE	Enables IBUFDISABLE port. (IOBUFE3, IOBUF_DDCIEN only).

Table 70: IOBUFE3, IOBUF, and IOBUF_DCIEN Ports

Port	I/O	Description
IO	Inout	Inout port connection. Connect directly to top-level port in the design.
O	Output	Output path of the buffer.
I	Input	Input port connection. Connect directly to top-level port in the design.
T	Input	Tristate enable input signifying whether the buffer acts as an input or output.
IBUFDISABLE	Input	Disables the input buffer and forces the O output to the internal logic to a logic Low when asserted High (IOBUFE3, IOBUF_DCIEN only).
DCITERMDISABLE	Input	Control to enable/disable input termination. This is generally used to reduce power in long periods of an idle state (IOBUFE3, IOBUF_DCIEN only).
OSC[3:0]	Input	This is not a supported port and must be left unconnected.
OSC_EN[1:0]	Input	This is not a supported port and must be left unconnected.
VREF	Input	Connect to XPIO_VREF (IOBUFE3 only).

Single-Ended Output Buffer Primitives

Figure 41: Single-Ended Output Buffer Primitives

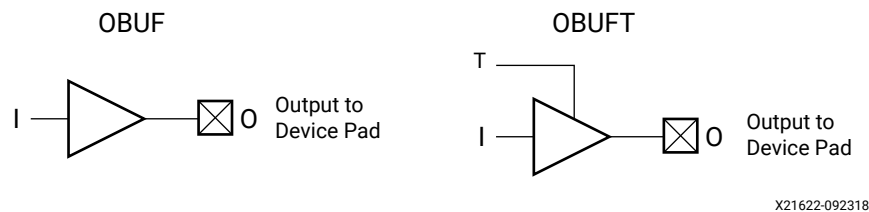


Table 71: OBUF and OBUFT Attributes

Attribute	Values	Description
SLEW	SLOW, FAST, MEDIUM	Specifies the slew rate of the output.
DRIVE	2, 4, 8, 12	Specifies the drive strength of the output.
IOSTANDARD	See XP IOB Supported Standards	Assigns an I/O standard to the element.

Table 72: OBUF and OBUFT Ports

Port	I/O	Description
O	Output	Output of OBUF to be connected directly to top-level output port.
I	Input	Input of OBUF. Connect to the logic driving the output port.
T	Input	Tristate enable input. (OBUFT only)

XP IOB Supported Differential Standards

DIFF_SSTL

The stub-series terminated logic (SSTL) for 1.5V (DIFF_SSTL15) and 1.35V (DIFF_SSTL135) are differential I/O standards used for general-purpose memory buses. DIFF_SSTL15 is used for DDR3 SDRAM interfaces and is roughly defined (not by name) in the JEDEC standard JESD79-3E. DIFF_SSTL135 is used for DDR3L SDRAM interfaces and is roughly defined (not by name) in the JEDEC standard JESD79-3-1.

Table 73: Allowed Attributes for DIFF_SSTL15, DIFF_SSTL135 and DIFF_SSTL12 I/O Primitives

Attributes	IBUF/IBUFE3		OBUF/OBUFT		IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Value	Default	Allowed Values	Default
IOSTANDARD	DIFF_SSTL15, DIFF_SSTL135, DIFF_SSTL12		DIFF_SSTL15, DIFF_SSTL135, DIFF_SSTL12		DIFF_SSTL15, DIFF_SSTL135, DIFF_SSTL12	
SLEW	N/A		FAST, MEDIUM, SLOW	SLOW	FAST, MEDIUM, SLOW	SLOW
OUTPUT_IMPEDANCE	N//A		RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_40_40	RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_40_40
ODT	RTT_40, RTT_48, RTT_60	RTT_40	N/A		RTT_40, RTT_48, RTT_60	RTT_40
DQS_BIAS (DIFF_SSTL12 ONLY)	TRUE, FALSE	FALSE			TRUE, FALSE	FALSE
VOH (DIFF_SSTL15 ONLY)	N/A	N/A	75, 80	75	N/A	N/A

DIFF_HSUL12

The high-speed unterminated logic (HSUL) DIFF_HSUL12 differential standard is optimized for lower-power memory interfaces.

Table 74: Allowed Attributes for DIFF_HSUL_12 and I/O Primitives

Attributes	IBUFDS/IBUFDSE3		OBUFDS/OBUFTDS		IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	DIFF_HSUL_12		DIFF_HSUL_12		DIFF_HSUL_12	
SLEW	N/A		FAST, MEDIUM, SLOW	SLOW	FAST, MEDIUM, SLOW	SLOW
OUTPUT_IMPEDANCE	N//A		RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_48_48	RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_48_48

Table 74: Allowed Attributes for DIFF_HSUL_12 and I/O Primitives (cont'd)

Attributes	IBUFDS/IBUFDSE3		OBUFDS/OBUFTDS		IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
ODT	RTT_120, RTT_240, RTT_NONE	RTT_NONE	N/A		RTT_120, RTT_240, RTT_NONE	RTT_NONE

DIFF_LVSTL

The low-voltage swing terminated logic (DIFF_LVSTL_11 and DIFF_LVSTL06_12) standards are optimized for lower-power memory interfaces. DIFF_LVSTL06_12 is compatible with differential LVSTL06 interfaces but is powered at 1.2V.

Table 75: Allowed Attributes for DIFF_LVSTL_11 and DIFF_LVSTL06_12 and I/O Primitives

Attributes	IBUFDS/IBUFDSE3		OBUFDS/OBUFTDS		IOBUFDSE3	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	DIFF_LVSTL_11, DIFF_LVSTL06_12		DIFF_LVSTL_11, DIFF_LVSTL06_12		DIFF_LVSTL_11, DIFF_LVSTL06_12	
SLEW	N/A		FAST, MEDIUM, SLOW	SLOW	FAST, MEDIUM, SLOW	SLOW
OUTPUT_IMPE DANCE	N/A		RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_40_40	RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_40_40
ODT	RTT_40, RTT_48, RTT_60	RTT_40	N/A		RTT_40, RTT_48, RTT_60	RTT_40
VOH (DIFF_ LVSTL_11 ONLY)	N/A		50	50	50	50
PRE_EMPHASIS	N/A				RDRV_240 (DIFF_LVSTL_11 only), RDRV_NONE	RDRV_NONE
EQUALIZATION	EQ_LEVEL0, EQ_LEVEL1, EQ_LEVEL2, EQ_LEVEL3, EQ_LEVEL4, EQ_NONE	EQ_NONE			N/A	
DC_BIAS	DC_BIAS_0, DC_BIAS1, DC_BIAS2, DC_BIAS3	DC_BIAS_0			DC_BIAS_0, DC_BIAS1, DC_BIAS2, DC_BIAS3	DC_BIAS_0

LVDS

LVDS15 is a 1.5V powered standard designed to interface with the low-voltage differential signaling (LVDS) standard. It is compatible with EIA/TIA electrical specifications but is powered at 1.5V to operate within the XPIO bank architecture.

Table 76: Allowed Attributes for LVDS15 and I/O Primitives

Attributes	IBUFDS/IBUFDSE3		OBUFDS/OBUFTDS		IOBUFDS/IOBUFDSE3 ¹	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	LVDS15		LVDS15		LVDS15	
DIFF_TERM_AD V	TERM_100, TERM_NONE	TERM_NONE	N/A		TERM_100, TERM_NONE	TERM_NONE
LVDS_PRE_EMP HASIS	N/A		FALSE, TRUE	FALSE	FALSE, TRUE	FALSE
EQUALIZATION	EQ_LEVEL0, EQ_LEVEL1, EQ_LEVEL2, EQ_LEVEL3, EQ_LEVEL4, EQ_NONE	EQ_NONE			N/A	
DC_BIAS	DC_BIAS_0, DC_BIAS1, DC_BIAS2, DC_BIAS3	DC_BIAS_0			DC_BIAS_0, DC_BIAS1, DC_BIAS2, DC_BIAS3	DC_BIAS_0

Notes:

1. The bidirectional configuration on these I/O standards is a fixed impedance structure optimized to 100Ω differential. They are only intended to be used in point-to-point transmissions that do not have turn around timing requirements.

DIFF_HSTL

DIFF_HSTL_I (1.5V) and DIFF_HSTL_I_12 (1.2V) are both differential high-speed transceiver logic (HSTL) standards used for the general-purpose high-speed bus standards defined by the JEDEC standard JESD8-6.

Table 77: Allowed Attributes for DIFF_HSTL_I and DIFF_HSTL_I_12 I/O Primitives

Attributes	IBUF/IBUFE3		OBUF/OBUFT		IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	DIFF_HSTL_I, DIFF_HSTL_I_12		DIFF_HSTL_I, DIFF_HSTL_I_12		DIFF_HSTL_I, DIFF_HSTL_I_12	
ODT	RTT_40, RTT_48, RTT_60	RTT_48	N/A		RTT_40, RTT_48, RTT_60	RTT_48
OUTPUT_IMPE DANCE	N//A		RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_48_48	RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_48_48
SLEW	N/A		FAST, MEDIUM, SLOW	SLOW	FAST, MEDIUM, SLOW	SLOW

DIFF_POD

The differential (DIFF_) versions (DIFF_POD10 and DIFF_POD12) use complementary single-ended drivers for outputs and differential receivers for inputs.

Table 78: Allowed Attributes for DIFF_POD10 and DIFF_POD12 I/O Primitives

Attributes	IBUF/IBUFE3		OBUF/OBUFT		IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Value	Default	Allowed Values	Default
IOSTANDARD	DIFF_POD10, DIFF_POD12		DIFF_POD10, DIFF_POD12		DIFF_POD10, DIFF_POD12	
SLEW	N/A		FAST, MEDIUM, SLOW	SLOW	FAST, MEDIUM, SLOW	SLOW
OUTPUT_IMPEDANCE	N//A		RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_40_40	RDRV_40_40, RDRV_48_48, RDRV_60_60	RDRV_40_40
ODT	RTT_40, RTT_48, RTT_60	RTT_40	N/A		N/A	
PRE_EMPHASIS	N/A		RDRV_240, RDRV_NONE	RDRV_NONE	RDRV_240, RDRV_NONE	RDRV_NONE
EQUALIZATION (DIFF_POD12 ONLY)	EQ_LEVEL0, EQ_LEVEL1, EQ_LEVEL2, EQ_LEVEL3, EQ_LEVEL4, EQ_NONE	EQ_NONE	N/A		N/A	
DQS_BIAS (DIFF_POD12 ONLY)	TRUE, FALSE	FALSE			TRUE, FALSE	FALSE

MIPI_DPHY

The MIPI D-PHY standard MIPI_DPHY is intended for use in mobile devices including cameras, displays, and unified protocol interfaces.

Table 79: Allowed Attributes for MIPI_DPHY I/O Primitives

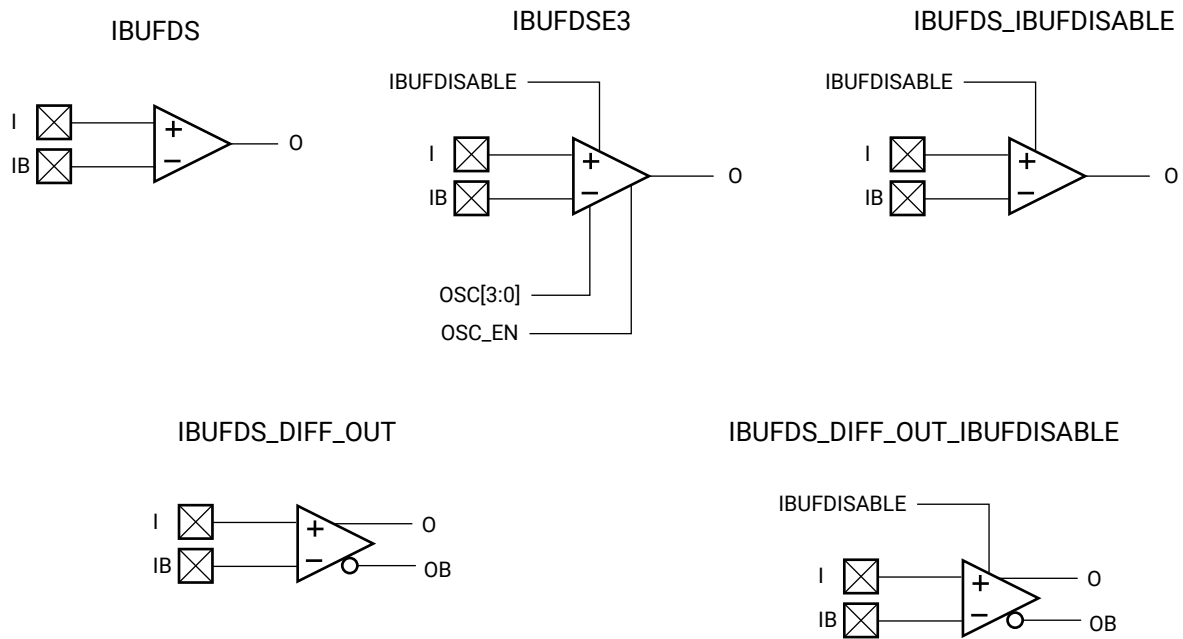
Attributes	IBUFDS_DPHY		OBUFDS_DPHY	
	Allowed Values	Default	Allowed Value	Default
IOSTANDARD	MIPI_DPHY		MIPI_DPHY	
DIFF_TERM_ADV	TERM_100, TERM_NONE	TERM_NONE	N/A	
VOH	N//A		28	28
PRE_EMPHASIS	N/A		RDRV_480, RDRV_NONE	RDRV_NONE
EQUALIZATION	EQ_LEVEL0, EQ_LEVEL1, EQ_LEVEL2, EQ_LEVEL3, EQ_LEVEL4, EQ_LEVEL5, EQ_LEVEL6, EQ_LEVEL7, EQ_LEVEL8. EQ_NONE	EQ_NONE		

DIFF_UNDEFINED Default IOSTANDARD

When an IOSTANDARD is not defined by the user, the default assignment for the IOSTANDARD defaults to DIFF_UNDEFINED. For a Versal ACAP design to complete implementation, a non-default IOSTANDARD must be defined with one of the valid I/O standards described in this section. The DIFF_UNDEFINED standard acts as a placeholder to allow a design to complete the early stages of implementation.

Differential Input Buffer Primitives

Figure 42: Differential Input Buffer Primitives



X21627-092318

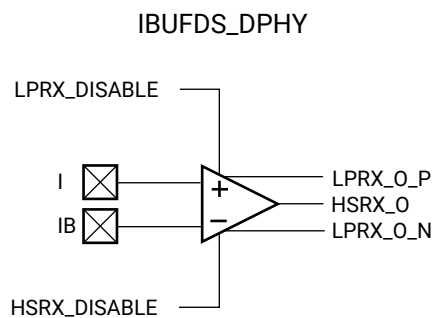
Table 80: IBUFDS3, IBUFDS, IBUFDS_DIFF_OUT, IBUFDS_DIFF_OUT_IBUFDISABLE, and IBUFDS_IBUFDISABLE Attributes

Attribute	Values	Description
IBUF_LOW_PWR	TRUE, FALSE	When set to TRUE, allows for reduced power on differential inputs for standards (for example: LVDS). A setting of FALSE demands more power but delivers higher performance characteristics.
IOSTANDARD	See XP IOB Supported Standards	Assigns an I/O standard to the element.
DIFF_TERM	TRUE, FALSE	Turns the built-in differential termination on (TRUE) or off (FALSE).
DQS_BIAS	TRUE, FALSE	Provides pull-up/pull-down feature required for some DQS memory interface pins or provides DC bias for certain LVDS applications.
USE_IBUFDISABLE	TRUE, FALSE	Enables the use of the IBUFDISABLE port. (IBUFDS3, IBUFDS_DIFF_OUT_IBUFDISABLE, IBUFDS_IBUFDISABLE only)

Table 81: IBUFDSE3, IBUFDS, IBUFDS_DIFF_OUT, IBUFDS_DIFF_OUT_IBUFDISABLE, and IBUFDS_IBUFDISABLE Ports

Port	I/O	Description
O	Output	Buffer output representing the input path to the device.
OB	Output	Complimentary buffer output representing the input path to the device. (IBUFDS_DIFF_OUT, IBUFDS_DIFF_OUT_IBUFDISABLE only)
I	Input	Input port connection. Connect directly to top-level P-side port in the design.
IB	Input	Input port connection. Connect directly to top-level N-side port in the design.
IBUFDISABLE	Input	The IBUFDISABLE pin can disable the input buffer and force the O output to the internal logic to a logic-Low when the IBUFDISABLE signal is asserted High. (IBUFE3, IBUFDISABLE,IBUFDS_DIFF_OUT_IBUFDISABLE, IBUFDS_IBUFDISABLE only)
OSC[3:0]	Input	This is not a supported port and must be left unconnected.
OSC_EN[1:0]	Input	This is not a supported port and must be left unconnected.
VREF	Input	V _{REF} input from XPIO_VREF (IBUFDSE3 only)

MIPI-DPHY Input Buffer Primitive

Figure 43: MIPI-DPHY Input Buffer Primitive


X21625-071320

Table 82: IBUFDS_DPHY Attributes

Attribute	Values	Description
IOSTANDARD	MIPI_DPHY	Assigns an I/O standard to the element.
DIFF_TERM	TRUE, FALSE	Turns the built-in differential termination on (TRUE) or off (FALSE).

Table 83: IBUFDS_DPHY Ports

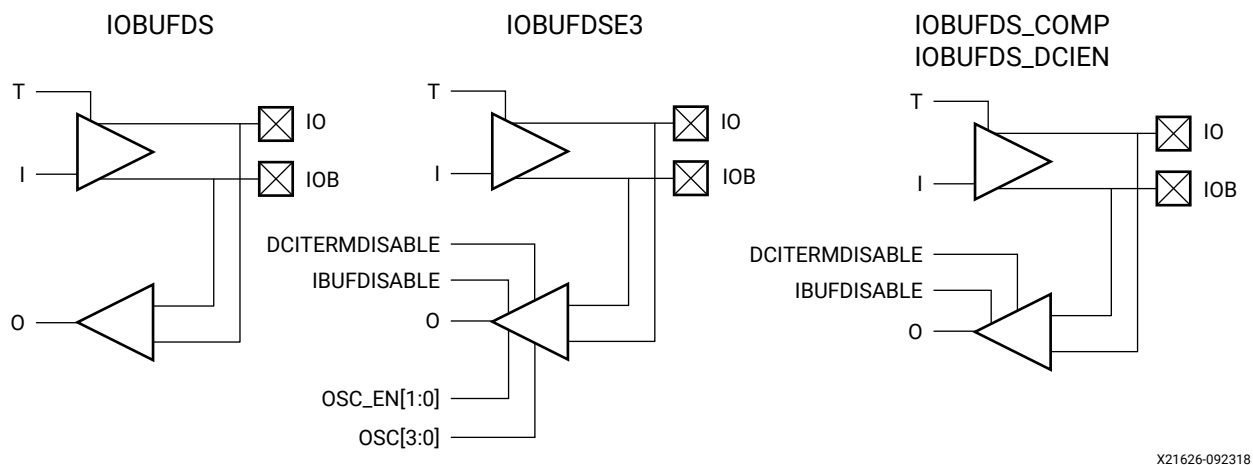
Port	I/O	Description
HSRX_O	Output	Inputs to the interconnect logic from the HS receiver.
LPRX_O_P	Output	Inputs to the interconnect logic from the low-power (LP) receiver.
LPRX_O_N	Output	Inputs to the interconnect logic from the low-power (LP) receiver.

Table 83: IBUFDS_DPHY Ports (cont'd)

Port	I/O	Description
I	Input	Input port connection. Connect directly to top-level P-side port in the design.
IB	Input	Input port connection. Connect directly to top-level N-side port in the design.
HSRX_DISABLE	Input	The HSRX_DISABLE port is used to enable or disable the MIPI D-PHY high-speed (HS) receiver.
LPRX_DISABLE	Input	The LPRX_DISABLE port is used to enable or disable the low-power (LP) receiver.

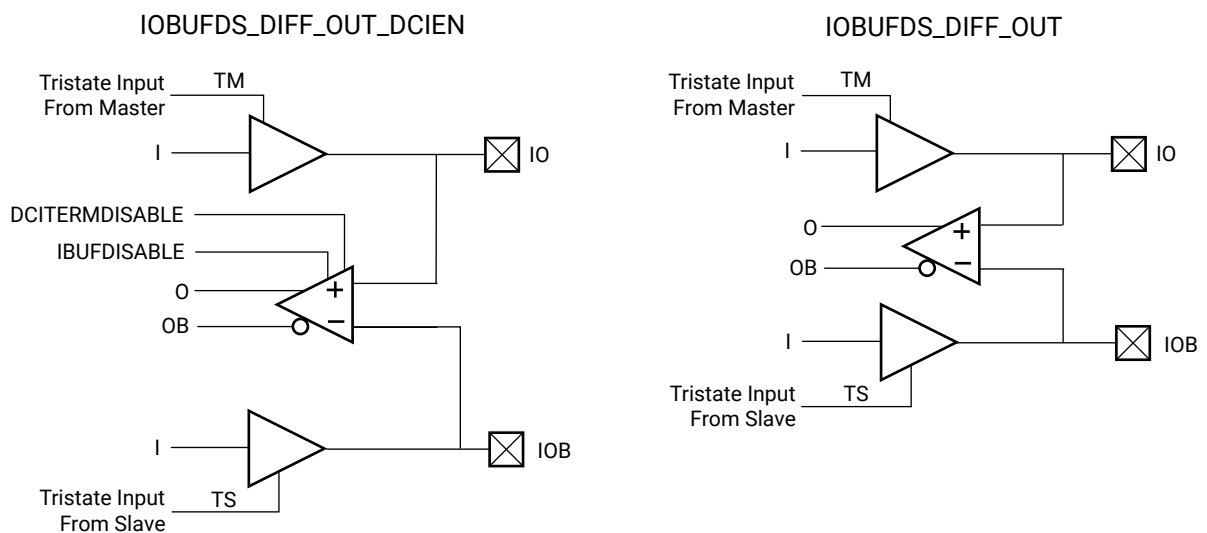
Differential Bidirectional Buffer Primitives

Figure 44: Differential Bidirectional Buffer Primitives



X21626-092318

Figure 45: Differential Bidirectional DIFF_OUT Buffer Primitives



X21620-112818

Table 84: IOBUFDSE3, IOBUFDS, IOBUFDS_COMP, IOBUFDS_DCIEN, IOBUFDS_DIFF_OUT, and IOBUF_DIFF_OUT_DCIEN Attributes

Attribute	Values	Description
IBUF_LOW_PWR	TRUE, FALSE	When set to TRUE, allows for reduced power on differential inputs for standards (for example: LVDS). A setting of FALSE demands more power but delivers higher performance characteristics.
SLEW	SLOW, FAST, MEDIUM	Specifies the slew rate of the output.
DRIVE	2, 4, 8, 12	Specifies the drive strength of the output.
IOSTANDARD	See XP IOB Supported Standards	Assigns an I/O standard to the element.
DIFF_TERM	TRUE, FALSE	Turns the built-in differential termination on (TRUE) or off (FALSE).
DQS_BIAS	TRUE, FALSE	Provides pull-up/pull-down feature required for some DQS memory interface pins or provides DC bias for certain LVDS applications.
USE_IBUFDISABLE	TRUE, FALSE	Enables use of the IBUFDISABLE port. (IOBUFDSE3, IOBUFDS_COMP, IOBUFDS_DCIEN, IOBUFDS_DIFF_OUT_DCIEN only)

Table 85: IOBUFDSE3, IOBUFDS, IOBUFDS_COMP, IOBUFDS_DCIEN, IOBUFDS_DIFF_OUT, and IOBUF_DIFF_OUT_DCIEN Ports

Port	I/O	Description
IO	Inout	Inout port connection. Connect directly to top-level P-side port in the design.
IOB	Inout	Inout port connection. Connect directly to top-level N-side port in the design. (IOBUFDS_DIFF_OUT only)
O	Output	Output path of the buffer representing the input path to the device.
OB	Output	Complimentary buffer output representing the input path to the device.
I	Input	Input port connection. Connect directly to top-level P-side port in the design.
IB	Input	Input port connection. Connect directly to top-level N-side port in the design.
T	Input	Tristate enable input signifying whether the buffer acts as an input or output.
IBUFDISABLE	Input	The IBUFDISABLE pin can disable the input buffer and force the O output to the internal logic to a logic-Low when the IBUFDISABLE signal is asserted High. (IOBUFDSE3, IOBUFDS_COMP, IOBUFDS_DCIEN, IOBUFDS_DIFF_OUT_DCIEN only)
DCITERMDISABLE	Input	Control to enable/disable DCI termination. This is generally used to reduce power in long periods of an idle state. (IOBUFDSE3, IOBUFDS_COMP, IOBUFDS_DCIEN, IOBUFDS_DIFF_OUT_DCIEN only)
OSC[3:0]	Input	This is not a supported port and must be left unconnected. Offset cancellation value. (IOBUFEDS3 only)
OSC_EN[1:0]	Input	This is not a supported port and must be left unconnected. Offset cancellation enable. (IOBUFEDS3 only)
VREF	Input	V _{REF} input from the XPIO_VREF. (IOBUFEDS3 only)
TM	Input	Tristate enable input for the P-side or master side signifying whether the buffer acts as an input or output. This pin must be connected to the same signal as the TS input. (IOBUFDS_DIFF_OUT, IOBUFDS_DIFF_OUT_DCIEN only)
TS	Input	Tristate enable input for the N-side or slave side signifying whether the buffer acts as an input or output. This pin must be connected to the same signal as the TM input. (IOBUFDS_DIFF_OUT_DCIEN, IOBUFDS_DIFF_OUT_DCIEN only)

Differential Output Buffer Primitives

Figure 46: Differential Output Buffer Primitives

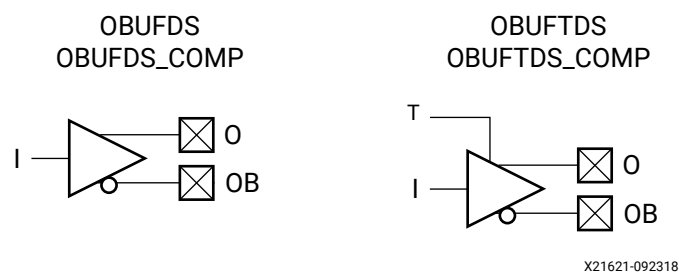


Table 86: OBUFDS, OBUFDS_COMP, OBUFTDS, and OBUFTDS_COMP Attributes

Attribute	Values	Description
SLEW	SLOW, FAST, MEDIUM	Specifies the slew rate of the output driver. (OBUFDS, OBUFTDS only)
IOSTANDARD	See XP IOB Supported Standards	Assigns an I/O standard to the element.

Table 87: OBUFDS, OBUFDS_COMP, OBUFTDS, and OBUFTDS_COMP Ports

Port	I/O	Description
O	Output	Output connected directly to P-side top-level output port.
OB	Output	Output connected directly to N-side top-level output port.
I	Input	Buffer input
T	Input	Tristate enable input. (OBUFTDS only)

MIPI-DPHY Output Buffer Primitive

Figure 47: MIPI-DPHY Output Buffer Primitive

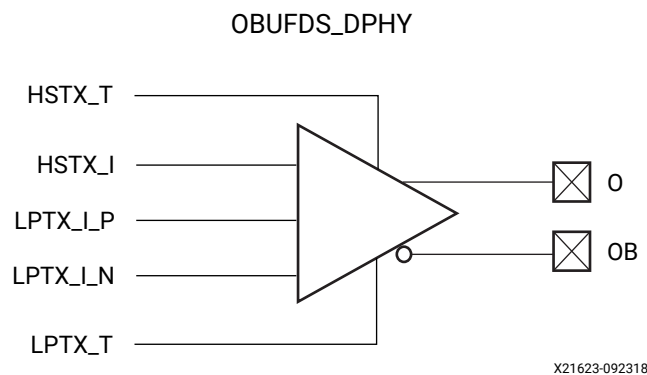


Table 88: OBUFDS_DPHY Attributes

Attribute	Values	Description
IOSTANDARD	MIPI_DPHY	Assigns an I/O standard to the element.

Table 89: OBUFDS_DPHY Ports

Port	I/O	Description
O	Output	Output connected directly to P-side top-level output port.
OB	Output	Output connected directly to N-side top-level output port.
HSTX_I	Input	Data input (HS TX)
HSTX_T	Input	Tristate control input (HS TX)
LPTX_I_N	Input	Data input (LP TX)
LPTX_I_P	Input	Data input (LP TX)

Table 89: OBUFDS_DPHY Ports (cont'd)

Port	I/O	Description
LPTX_T	Input	Tristate control input (LP TX)

XPIO_VREF

Table 90: XPIO_VREF Ports

Port	I/O	Description
FABRIC_VREF_TUNE[9:0]	Input	Tunes VREF from the interconnects logic
VREF	Output	Provides VREF to IBUFE3

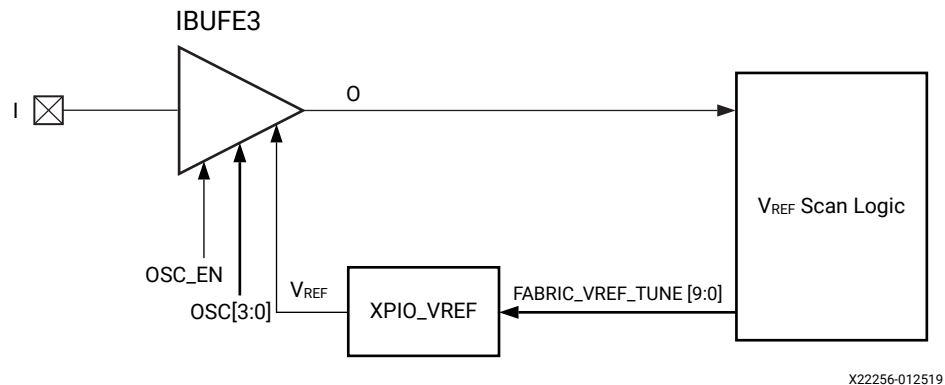
XP IOB Internal V_{REF}

The XP IOB supports several single-ended standards that typically require an external reference voltage to define the receiver switching threshold levels, known as V_{REF} . In the XP IOB, these thresholds must be internally derived from the V_{CCO} bank voltage, and are automatically assigned to the IOB at a predefined level based on the IOSTANDARD selected for the pins. Pins within the same nibble share the same V_{REF} generation circuitry and thus must share the same V_{REF} levels. See [XP IOB Supported Standards](#) for a list of standards and the associated internal V_{REF} levels.

An optional V_{REF} scan helps fine tune the internal V_{REF} of input buffers to maximize performance. It can be accessed through the IBUFE3 and IOBUFE3 primitives with the XPIO_VREF primitive. Internal V_{REF} tuning controls the V_{REF} on a per-nibble basis. Within a bank, each nibble can have its own variation of a given V_{REF} . Inputs with I/O standards of different V_{REF} specification cannot be placed within the same nibble.

Receiver V_{REF} Scan

An optional V_{REF} scan feature in XP I/O banks helps to fine tune the internal V_{REF} of input buffers to maximize the performance for a subset of I/O standards. This feature can be accessed through the IBUFE3 and IOBUFE3 primitives in conjunction with the XPIO_VREF primitive as shown in the following figure. V_{REF} scan requires building control logic into your interconnect logic design.

Figure 48: Access to the V_{REF} Scan


Internal V_{REF} tuned using the V_{REF} scan feature controls the V_{REF} of six consecutive I/Os (one nibble group) within a bank as shown in the following figure. Inputs with I/O standards of different V_{REF} specifications cannot be placed within the same bank. A tuned V_{REF} connection (V_{REF} output of the XPIO_VREF primitive) cannot traverse nibble boundaries.

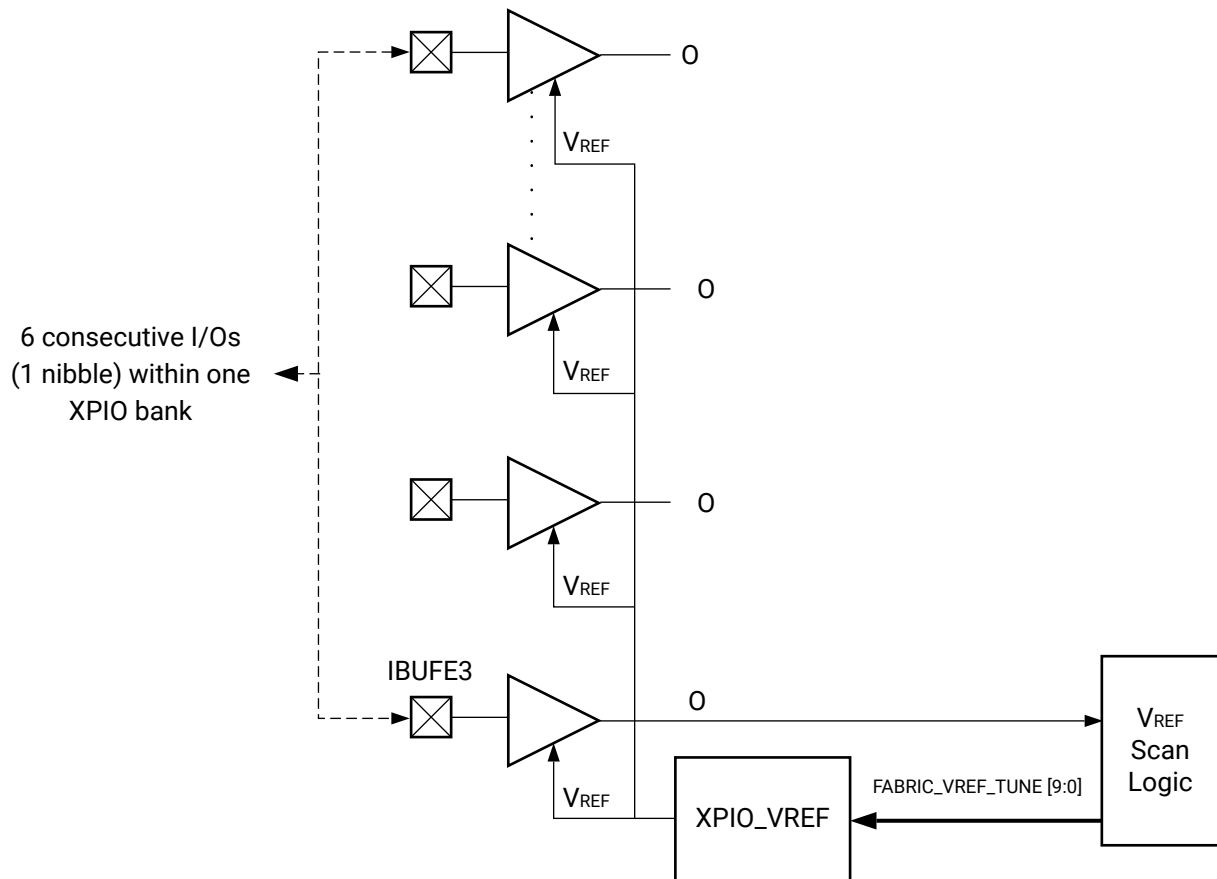
The 10-bit FABRIC_VREF_TUNE value provides a scaling factor as a percentage of V_{CC0} as follows:

Equation 1: V_{REF} Scan Scaling Factor

$$\frac{FABRIC_VREF_TUNE[9 : 0]}{1023} = \% \text{ of } V_{CC0}$$

So, for example, a FABRIC_VREF_TUNE value of 100000000_2 provides a V_{REF} of approximately 50% V_{CC0} .

★ **IMPORTANT!** *Input signals for all standards other than LVSTL_11 and LVSTL06_11 are scaled by 50% compared to V_{REF} . The V_{REF} scan value must be chosen to account for the scaled value. For example, because SSTL12's target V_{REF} is 50% V_{CC0} , the V_{REF} scan value should be 010000000_2 , or 25% V_{CC0} , because the input signal is scaled before reaching the receiver. The Vivado tools always report V_{REF} levels relative to the input pin level (unscaled).*

Figure 49: V_{REF} Scan Connection per Nibble within a Bank


X22255-012519

XP IOB Driver Control, Internal Termination, and Internal Bias

The XP IOB provides a variety of termination options to support the available I/O standards. XP IOBs provide several different types of driver control, calibrated termination, and bias control. The XP IOB drivers have several features that allow for an optimal driver to be used in a system. Single-ended and Pseudo-differential drivers have the ability to have driver strength and edge_rate optimized through the DRIVE, SLEW, or OUTPUT_IMPEDANCE attributes. Similarly, the XP IOBs receivers are equipped with internal calibrated on-die termination, differential termination, and weak bias solutions that are capable of holding an undriven line value or providing a DC bias point to an AC coupled link. The ability to control driver attributes and receive termination and biasing allows great flexibility, signal quality, and simplicity when compared to other board level solutions.

Drive Strength Control and the VOH Attribute

DRIVE Strength Control

LVC MOS drivers (LVC MOS12 and LVC MOS15) drivers support output drive strength control to size the driver output strength to the load driven. For a list of drive options and standards, see the [XP IOB Supported Standards](#) section. Assigning the DRIVE property 2, 4, 6, 8, or 12 (LVC MOS15 only) ensures that the XP IOB driver can meet valid logic thresholds for loads of 2 mA, 4 mA, 6 mA, 8 mA, or 12 mA, respectively. The DRIVE attribute uses the following syntax in the XDC file:

```
set_property DRIVE value [get_ports port_name]
```

VOH Attribute

MIPI_DPHY, LVSTL_11, and DIFF_LVSTL_11 support the VOH attribute to help define the mode of the driver. VOH helps define the desired output high level for the driver. This can be used to define the use mode for a given standard. For LVSTL_11 and DIFF_LVSTL_11, the VOH attribute must be 50 (50% of V_{CC0}). For MIPI_DPHY, the VOH attribute must be 28 (28% of V_{CC0}). For scenarios where an SSTL15 driver needs to drive a 1.8V SSTL receiver, a VOH attribute of 80 adjusts the SSTL15 driver to be compatible with a 1.8V SSTL receiver. This option is limited in its scope to a single VOH option for a given standard. If not set, this property defaults to the appropriate VOH level.

```
set_property VOH value [get_ports port_name]
```

Slew Control

Most drivers support FAST, MEDIUM, and SLOW slew-rate options. These options can be used to optimize the edge rate of the driver for specific applications. For a list of IOSTANDARDS that support slew control see the [XP IOB Supported Standards](#) section.

The SLEW attribute uses the following syntax in the XDC file:

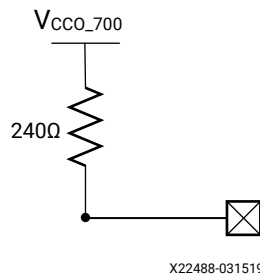
```
set_property SLEW value [get_ports port_name]
```

Calibrated Termination (Digitally Controlled Impedance)

To provide the highest precision termination, up to two external reference resistors per device must be provided to calibrate the internal device termination. Banks 700 and 800 (bank 800 does not exist in all devices) contain the IO_VR_700 and IO_VR_800 (not available on all devices) pins that must have a 240Ω reference resistor pulled up to the V_{CCO} powering the referenced bank number (that is banks 700 or 800). The reference resistor is used to calibrate all I/O banks on the same half of the device. Banks 700 and 800 (when present) must always be powered by a valid V_{CCO} level.

★ IMPORTANT! The IO_VR_700 and IO_VR_800 pins (not available on all devices) must have an external 240Ω resistor tied to V_{CCO_700} and V_{CCO_800} respectively. These pins are dedicated and can not be used as user I/O. All designs **MUST** populate these pins appropriately, regardless of the I/O standards used in a design.

Figure 50: IO_VR_700 Pin



In previous FPGA generations, specific IOSTANDARDS ending in *_DCI* were required to enable the internal calibration circuitry (DCI). In the Versal architecture, the reference calibration circuitry is always enabled and *_DCI* specific standards are not required to enable calibrated termination.

The XP IOB provides several types of calibrated termination:

- Split-termination input impedance (termination to V_{CCO}/2)
- Single-termination input impedance (termination to V_{CCO} or termination to GND)
- Source termination

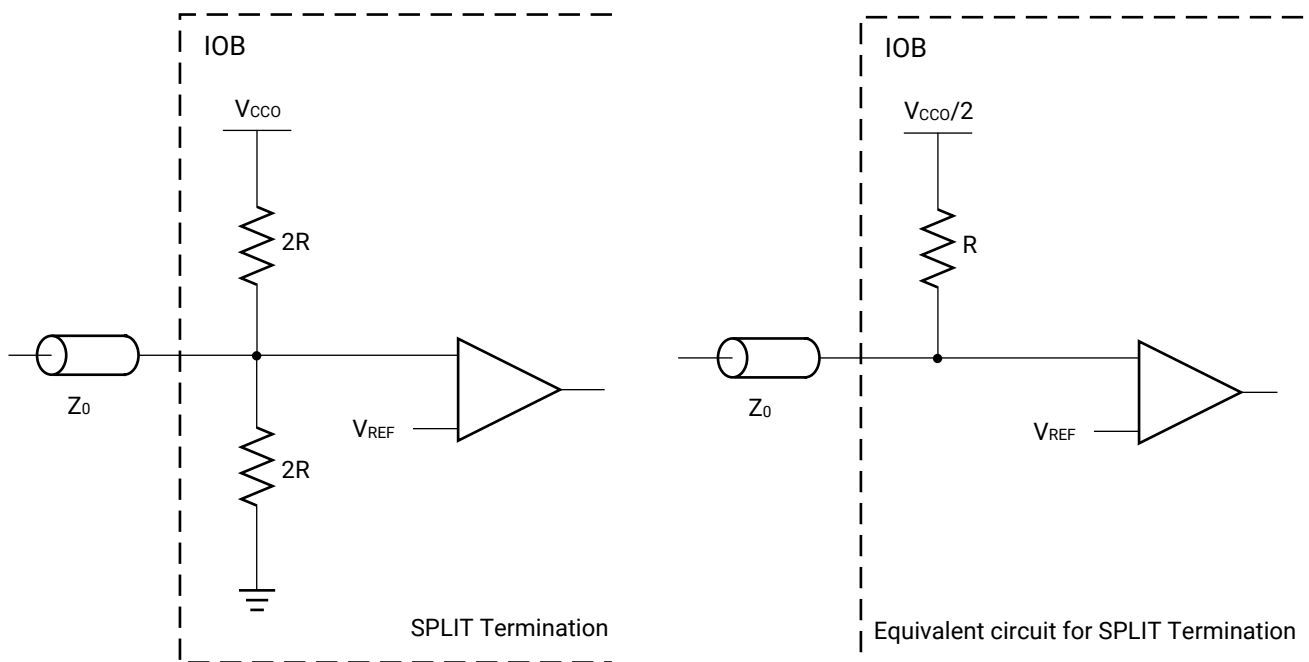
Each type of termination allows the selection of termination strengths to accommodate 40Ω, 48Ω, and 60Ω interfaces. Single-termination input impedance offsets a pull type to either V_{CCO} or GND depending on the standard (see the [XP IOB Supported Standards](#) section for further information).

On-Die Termination Attribute

The on-die termination (ODT) attribute supports split or single termination on the inputs of the HSTL, SSTL, POD, LVSTL, and HSUL standards. The advantage of using ODT over discrete resistors is that signal integrity is improved by completely removing any PCB trace stubs at the receiver. The ODT attribute is used to define the value of the on-die termination at the input. The V_{CCO} of the I/O bank must be connected to the appropriate voltage level for the ODT attribute to perform as expected. When ODT is used for differential buffers, both P and N pins have the same termination.

The ODT_SPLIT termination is for memory interfaces that use the SSTL or HSTL type I/O standards. SPLIT termination provides termination that centers the signal around the V_{REF} input threshold level. As shown in the following diagram, in Versal devices this termination is implemented internally with $2R$ pulled up to V_{CCO} and $2R$ pulled down to GND, creating a symmetric equivalent termination of R to $V_{CCO}/2$.

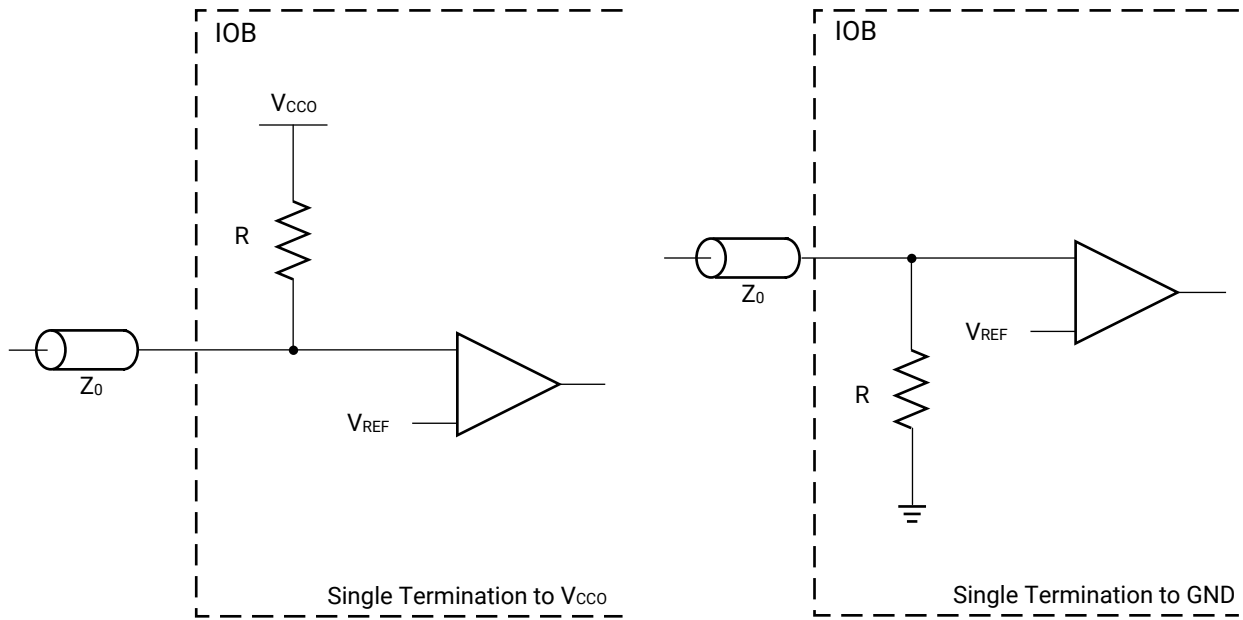
Figure 51: ODT_SPLIT



X22490-031519

ODT_SINGLE termination provides termination to V_{CCO} or GND and is used in open-drain I/O standards (POD or LVSTL). The standard used defines whether the ODT is terminated to V_{CCO} or GND (see the supported I/O standards table) and is implemented with a single R structure to V_{CCO} or GND as shown in the following figure.

Figure 52: ODT_SINGLE



X22491-041519

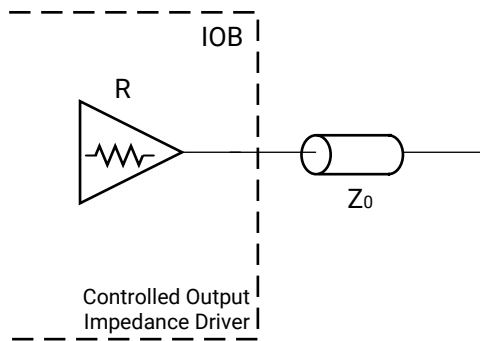
The ODT attribute uses the following syntax in the XDC file:

```
set_property ODT value [get_ports port_name]
```

Values can be RTT_40, RTT_48, and RTT_60 for termination values of 40Ω, 48Ω, or 60Ω, respectively. RTT_40 is the default ODT setting.

For source termination, Versal devices provide calibrated output impedance control.

Figure 53: Controlled Output Termination



X21641-092418

Values can be RDRV_40, RDRV_48, and RDRV_60 for termination values of 40Ω, 48Ω, or 60Ω, respectively.

```
set_property OUTPUT_IMPEDANCE value [get_ports port_name]
```

DCITERMDISABLE

Bidirectional primitives that use ODT (SPLIT or SINGLE) internal termination have a DCITERMDISABLE port that allows you to disable internal input termination. Asserting this port High can help save power during long periods of IDLE time on an interface.

Differential Termination Attribute

In addition to calibrated impedance, a fixed precision 100Ω differential input impedance block is available in XP IOB input pairs to support LVDS15 and MIPI_DPHY. For both LVDS15 and MIPI_DPHY, the bank must be powered at the VCCO level (1.5V for LVDS15 or 1.2V for MIPI_DPHY) to use the differential impedance block.

The DIFF_TERM_ADV attribute can be set to TERM_100 or TERM_NONE (default) to enable internal 100Ω termination. It uses the following syntax in the XDC file:

```
set_property DIFF_TERM_ADV [get_ports port_name]
```

XP IOB Level Hold and Bias Features

The XP IOB provides several weak hold and biasing features in Versal devices. These features are designed to provide a known level to a pin that is not actively driven.

PULLUP, PULLDOWN, KEEPER

All XP IOB pins can enable a weak internal PULLUP bias to V_{CCO} , a weak internal PULLDOWN to GND, or a weak internal KEEPER circuit that both pulls up to V_{CCO} or pulls down to GND, based on the last driven pin voltage. The PULLTYPE attribute can be set to NONE, PULLUP, PULLDOWN, or KEEPER using the following syntax in the XDC file:

```
set_property PULLTYPE value [get_ports port_name]
```

DQS_BIAS, DC_BIAS, and AC Coupling

DQS_BIAS

DQS_BIAS behaves as a logic 0 holding mechanism for undriven pins in pseudo-differential buffers (for example: DIFF_HSTL or DIFF_SSTL) by weakly pulling the P-side of the buffer to GND and the N-side of the buffer to V_{CCO} . This allows an IDLE link to maintain a fixed logic level when a driver and termination are disabled on the link. The left circuit in the following figure shows DQS_BIAS behavior on pseudo-differential links.

The allowed values for the DQS_BIAS attribute for applicable I/O standards are TRUE and FALSE (DEFAULT) and are enabled using the following syntax:

```
set_property DQS_BIAS TRUE|FALSE [get_ports port_name]
```

DC_BIAS

DC_BIAS provides an internal bias to both P and N pins used as an input in scenarios where an AC coupled differential signal needs to be re-biased such that the LVDS15 receiver specifications are met. The DC_BIAS feature creates a bias through an equivalent voltage divider network to the bank's V_{CC0} . The DC_BIAS attribute can be added to the XDC:

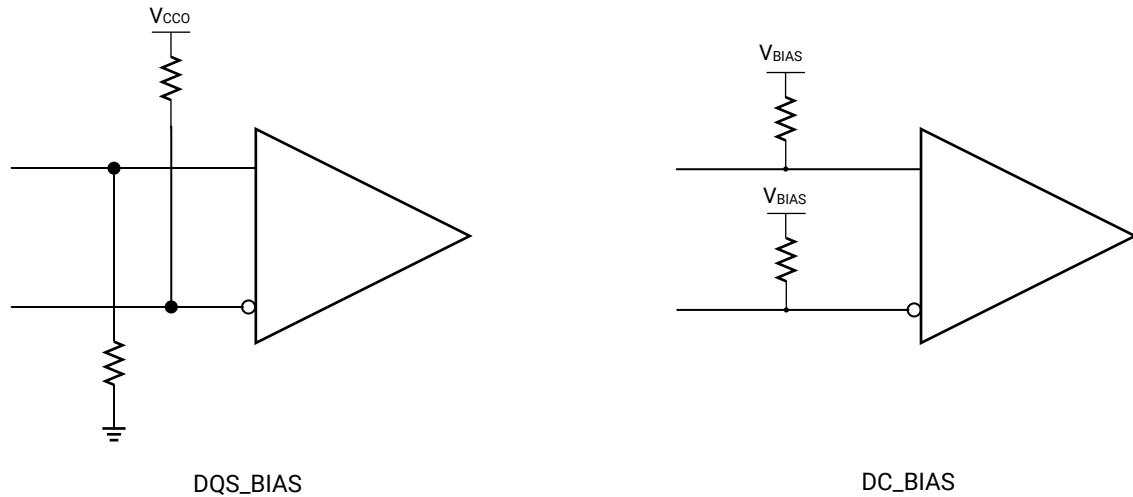
```
set_property DC_BIAS DC_BIAS_0|DC_BIAS_1|DC_BIAS_2|DC_BIAS_3 [get_ports port_name]
```

The DC_BIAS_1 setting provides the equivalent to $192\ \Omega$ at $20\% V_{CC0}$. In a 1.5V bank, the combination of DIFF_TERM_ADV and DC_BIAS_1 provide an appropriate termination and bias for an AC coupled LVDS link without the need for bias or termination components on the PCB. DC_BIAS_2 provides the equivalent of $48\ \Omega$ to $20\% V_{CC0}$, but is not recommended for use with AC coupling due to a higher current draw caused by the voltage divider used to generate the equivalent $48\ \Omega$ voltage divider. DC_BIAS_3 provides a $50\ \Omega$ to GND bias which has very limited practical use as a bias network. With a LVDS15 IOSTANDARD used in a 1.5V bank, the combination of DC_BIAS_1 and DIFF_TERM_ADV provide both a bias and termination appropriate for many differential signals that require AC coupling. Because DC_BIAS can corrupt a weaker driver, it should not be used when the IOB is configured as an output or bidirectional.

Table 91: DC_BIAS Levels Explained

DC_BIAS Attribute	Description
DC_BIAS_0	No Bias
DC_BIAS_1	$192\ \Omega$ to $20\% V_{CC0}$. Suitable for AC coupling applications needing a weaker bias. DIFF_TERM_ADV or equivalent external $100\ \Omega$ termination should be used with DC_BIAS_1. DIFF_TERM_ADV is only available in a 1.5V V_{CC0}
DC_BIAS_2	$48\ \Omega$ to $20\% V_{CC0}$. Provides a strong bias and termination. DC_BIAS_2 should not be used with DIFF_TERM_ADV.
DC_BIAS_3	$50\ \Omega$ to GND. Care must be taken to ensure that GND biased signal does not violate input lower levels outlined in the data sheet.

Figure 54: DQS_BIAS and DC_BIAS Diagram



X22475-071320

AC Coupling Recommendations

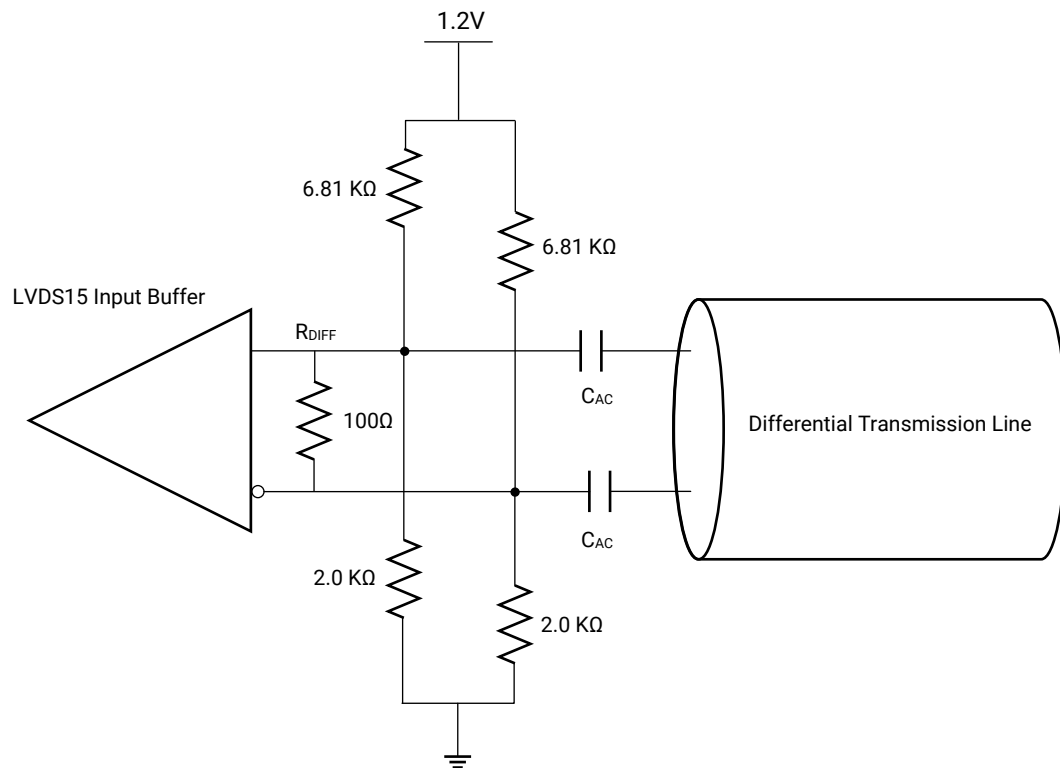
When receiving data from an AC coupled driver (like a clock source), care must be taken to ensure the appropriate bias levels are selected so that the receiver's input threshold requirements are met. If the receiver uses LVDS15 and resides in a 1.5V powered bank, a DC_BIAS value of DC_BIAS_1 along with DIFF_TERM_ADV setting of TERM_100 ensure that both a DC bias level and termination are provided inside the IOB for an AC coupled input. In scenarios where a 1.5V bank voltage is not used and AC coupling is required, it is recommended that both an external bias and external termination are used:

- Do not use the optional internal differential termination.
 - DIFF_TERM_ADV = TERM_NONE
 - DIFF_TERM = FALSE (default)
- The differential signals at the input pins must meet the V_{IN} requirements in the *Recommended Operating Conditions* table of the specific [Versal ACAP data sheets](#).
- The differential signals at the input pins must meet the V_{IDIFF} (minimum) requirements in the corresponding LVDS15 specifications tables of the specific [Versal ACAP data sheets](#).
- The differential signals at the input pins must meet the V_{IDIFF} (minimum) requirements in the corresponding LVDS15 specifications tables of the specific [Versal ACAP data sheets](#).

One way to accomplish this criteria is to use an external circuit that both AC-couples and DC-biases the input signals. The following figure shows an example circuit for providing an AC-coupled and DC-biased circuit for a differential clock input. R_{DIFF} provides the 100Ω differential receiver termination because the internal $DIFF_TERM_ADV = TERM_NONE$. V_{BIAS} should be a 1.0V to 1.5V source (typically V_{CC0}) with an asymmetric termination structure that results in a V_{BIAS} between 200 mV and 300 mV. Resistors in the 1K–10 K Ω range are recommended. The following figure is an example of an AC coupling network appropriate for a differential clock coming into a 1.2V powered bank.

IMPORTANT! On all AC coupled interfaces there is a required settling time for the DC bias to charge to appropriate levels.

Figure 55: AC-Coupled with DC-Biased Differential Clock Input Example



X22485-103020

The typical values for the AC coupling capacitors C_{AC} are in the range of 100 nF. All components should be placed physically close to the device inputs. See the specific [Versal ACAP data sheets](#) for the range of the bias voltage to be used in association with the external R_{DIFF} resistor and the C_{AC} AC coupling capacitors that will provide the appropriate bias for inputs without using the resistor network. As noted in [XP IOB Pre-emphasis and Equalization](#), an AC coupled link is required when using receiver equalization. If receiver equalization is NOT desired on an AC coupled link, the `EQUALIZATION` attribute must be set to `EQ_LEVEL0`.

XP IOB IBUFDISABLE

Several input and bidirectional primitives have an IBUFDISABLE port that can disable the input buffer and force the O output of the internal logic to a logic Low when the IBUFDISABLE signal is asserted High. The USE_IBUFDISABLE attribute must be set to TRUE, the IBUFDISABLE port must be controlled, and SIM_DEVICE set to VERSAL for this primitive to have the expected behavior that is specific to the Versal architecture. This feature can be used to reduce power at times when the I/O is idle.



IMPORTANT! The IBUFDISABLE feature can only be used on differential and V_{REF} based standards. Neither LVCMOS nor the MIPI_DPHY LP input are impacted by IBUFDISABLE.

XP IOB Pre-emphasis and Equalization

Equalization

For challenging, high-speed links, the XP IOB receiver provides several different levels of equalization designed to increase the eye opening for LVDS15, DIFF_LVSTL_11, DIFF_LVSTL06_12, DIFF_POD12, LVSTL_11, LVSTL06_12, and POD12 interfaces. Equalization should only be used on AC coupled links.

The EQUALIZATION attribute uses the following syntax in the XDC file:

```
set_property EQUALIZATION VALUE [get_ports port_name]
```

Table 92: Typical Gain for Different Values of Equalization

Value	Estimated Gain (dB)
EQ_NONE	No gain. For use on DC coupled links.
EQ_LEVEL0	No gain. For use on AC coupled links.
EQ_LEVEL1	
EQ_LEVEL2	
EQ_LEVEL3	
EQ_LEVEL4	

Transmitter Pre-emphasis

For high-speed interfaces on lossy channels the XP IOB driver provides the ability to enable several levels of pre-emphasis for LVDS_15, LVSTL_11, and POD drivers.

The LVDS_PRE_EMPHASIS attribute uses the following syntax in the XDC file when the LVDS15 standard is used:

```
set_property LVDS_PRE_EMPHASIS VALUE [get_ports port_name]
```

The PRE_EMPHASIS attribute is used for LVSTL_11, and POD drivers as well as their differential equivalents. The PRE_EMPHASIS attribute uses the following XDC command:

```
set_property PRE_EMPHASIS VALUE [get_ports port_name]
```

Table 93: Typical Pre-emphasis with an OUTPUT_IMPEDANCE of 40Ω

Attribute	Value	Estimated Gain (dB)
LVDS_PRE_EMPHASIS with an OUTPUT_IMPEDANCE of 40Ω	RDRV_240	2.5

XP Bank Supporting Resources and Corner Banks

Beyond the XPHY, IOL, and IOB, there are additional resources in or around the XPIO banks that can be used to improve the functionality of the XPIO. This chapter briefly discusses clocking resources available to XPIO banks and the boundary logic interface (BLI) can be used to aid in XPIO interfaces. In addition to discussing these supporting resources, XPIO corner banks and the limitations associated with these banks are also discussed.

Clocking Resources

Each XPIO bank has access to four clock managers as well as to several global clock buffers which can aid in XPIO interfaces by providing clock frequency synthesis, jitter filtering, and clock deskew. Because these resources are described in detail in *Versal ACAP Clocking Resources Architecture Manual* (AM003), deeper level details are not covered in this chapter.

As mentioned in [Clocking](#), there are four global clock (GC) inputs per XPIO bank which have direct access to clock routing resources through global clock buffers as well as having direct access to adjacent clock managers. Each of the four GC pins can be used as a single-ended clock or as a P-side (master) of a differential pair to drive adjacent clock managers or global clock buffers.

In each XPIO banks exist two XPLL clocking managers. The XPLL provides a user-controlled phase and deskew control and an optimized connection to the XPHY, but can also be used with the XP IOL. Refer to [Chapter 2: XPHY Architecture](#) for details on how the XPLL resources are used with the XPHY.

In addition to the two XPLL, adjacent to each XPIO bank are MMCM and DPLL clock mangers. The MMCM provides a deskew phase detector and extensive frequency synthesis capabilities which can be useful in XPIO interface clocking. The DPLL provides similar functionality as the MMCM but with reduced functionality related to fractional clock generation. Both the MMCM and DPLL can be used in XPIO interfaces to synthesize and deskew clock sources.

Besides clock managers, GC input pins have several options for driving global clock buffers directly or through the clock managers described above. Adjacent to each XPIO bank are 24 BUFGEs, 8 BUFCTRLS, and 4 BUFGE_DIVs.

Boundary Logic Interface

In addition to clocking resource, XPIO banks provide register stages for both signals going into and coming out of the programmable logic space to and from XPIO. Because this resource exists between both types of XPIO logic resources (XPHY and XPIOL) and programmable logic regions, these logical blocks are called the boundary logic interface (BLI). In certain configurations, these BLI register stages can help optimize the timing of an interface. To implement registers in the BLI region, an FDCE or FDRE primitive can be instantiated in a design leveraging the BLI attribute:

```
set_property BLI TRUE|FALSE [get_cells register_name]
```

BLI registers have a few restrictions that FDRE and FDCE typically do not have. BLI registers cannot be assigned an initial value and BLI registers do not have synchronous reset capabilities. When leveraging BLI registers, it can be helpful to leverage directives like `EXTRACT_RESET = "no"` in the Vivado Design Suite.

Figure 56: BLI Capable Flip-Flops (FDCE and FDRE)

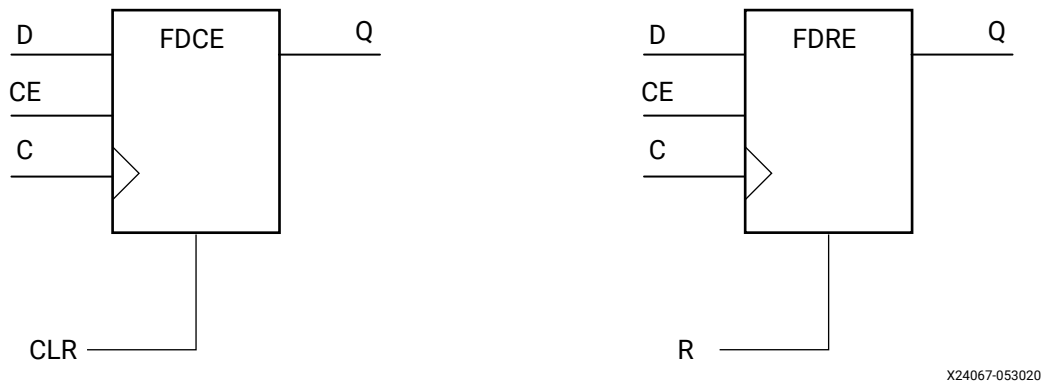


Table 94: Single Data Rate Flip-Flop (FDCE and FDRE) Ports

Port	I/O	Description
Q	Output	Data output
C	Input	Clock input pin
CE	Input	Active-High clock enable register
D	Input	Data input
CLR	Input	Asynchronous clear (FDCE only)
R	Input	R must be tied to ground for FDRE in BLI.

Both single data rate and double data rate paths to and from the XP IOL can either use or bypass BLI registers. For the XPHY, several signals can leverage the BLI, see the below table for specific XPHY signals that can leverage BLI registers. For the table below, `DIV_CLK = REFCLK_FREQUENCY / TX_DATA_WIDTH`.

Table 95: XPHY Ports That Can Connect to the BLI

XPHY Port	Clock Domain	Reset
CE	CTRL_CLK	-
PHY_RDEN	DIV_CLK	RST
Q<0-5> ¹	FIFO_RD_CLK	RX_RST[x] where x is Qx
DLY_RDY ¹	CTRL_CLK	RST
PHY_RDY ¹	CTRL_CLK	RST
GT_STATUS	-	RST
FIFO_EMPTY	FIFO_RD_CLK	RST
D<0-5>	DIV_CLK	TX_RST[x] where x is Dx
CNTVALUEOUT	CTRL_CLK	RST
CNTVALUEIN	CTRL_CLK	RST
CE	CTRL_CLK	RST
INC	CTRL_CLK	RST
LD	CTRL_CLK	RST
PHY_WREN	DIV_CLK	RST
PHY_WRCS0	DIV_CLK	RST
PHY_WRCS1	DIV_CLK	RST

Notes:

1. DLY_RDY, PHY_RDY, and Q<0-5> of the same nibble cannot all be connected to the BLI. DLY_RDY and PHY_RDY can be connected, or Q<0-5> can be connected, but not all three signals.

Corner Banks

When XPIO banking resources are located adjacent to certain resources such as the processing system (PS) or high-speed transceiver columns, the XPIO bank will have limited functionality. While clocking pins (GC) in corner banks will have full access to clocking resources, non-clocking pins will be restricted to the memory controller (DDRMC) functionality. Because these restricted XPIO bank are typically located at the corners of a device, they are referred to as corner banks. Though usually defined along bank boundaries, in some instances a partial bank on nibble boundaries might be restricted to DDRMC use. Corner bank locations vary by device and are explicitly called out in *Versal ACAP Packaging and Pinouts Architecture Manual (AM013)* with a DDRMC designation. In pin planning a Versal ACAP design, it is important to realize that corner banks should only be used for DDRMC interfaces. When using the Advanced I/O Planner for Advanced I/O Wizard designs, the corner banks are blocked from use.

HD IOL Resources

HD IOL Features

Single Data Rate Flip-Flops

Each pin contains local single-data rate (SDR) flip-flop registers in the IOB for both data and tristate lines. Both output, input, and tristate SDR flip-flops use these blocks:

- **FDCE:** Flip-flop with clock enable and asynchronous clear
- **FDPE:** Flip-flop with clock enable and asynchronous preset
- **FDRE:** Flip-flop with clock enable and synchronous reset
- **FDSE:** Flip-flop with clock enable and synchronous set

Both data and tristate registers must be used in the same manner when they are used together (that is both SDR, DDR, or not registered at all).

Double Data Rate Flip-Flops

Each pin contains local double-data rate (DDR) flip-flop registers in the IOB for both data and tristate lines. The DDR flip-flops the input path uses the `IDDRE1` and the output and tristate paths use the `ODDRE1`. When both data and tristate registers are used, it is required that Both data and tristate registers must be used in the same manner (that is both SDR, DDR, or not registered at all).

Uncalibrated IOB Delay

Each pin has two uncalibrated delay blocks: one for the input delay (`IDELAYE5`) and one for the output path (`ODELAYE5`). Each individual block provides a uncalibrated delay of up to approximately 1.8 ns for both data and tristate paths. When only the input portion of the IOB is used, the `ODELAYE5` can be cascaded to the `IDELAYE5` to provide the input path up to approximately 3.6 ns of delay. Both `IDELAYE5` and `ODELAYE5` have 32 taps that can be changed incremented or decremented using the `INC` and `CE` pins, or can be dynamically changed using the `CNTVALUEIN` and `LOAD` pins.

DPLL

Each HDIO bank has an all-digital phase locked loop (DPLL) incorporated into the bank with the ability to phase shift the clock signal for optimal signal timing. The DPLL is used to generate new clock frequencies and to eliminate skew between clock and data paths as they reach the IDDR and ODDR registers.

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

- The minimum input clock frequency (FINMIN_DPLL) is 50 MHz instead of 10 MHz.
- DPLL PD deskew function and ZHOLD mode are not supported in the listed devices. MMCMs and IDELAY have to be used instead.
- DPLLs in HDIO banks are not supported in the listed devices.

For more information, see the *Versal ACAP Clocking Resources Architecture Manual* ([AM003](#)).

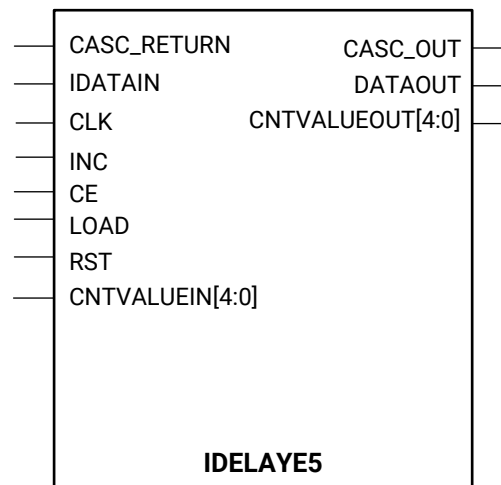
HD IOL Primitives

The IOL registers and delay blocks are instantiated in the design using HDL primitives.

IDELAY and ODELAY Primitives

Uncalibrated Input Delay

Figure 57: IDELAYE5 Primitive



X22469-050519

Table 96: IDELAYE5 Attributes

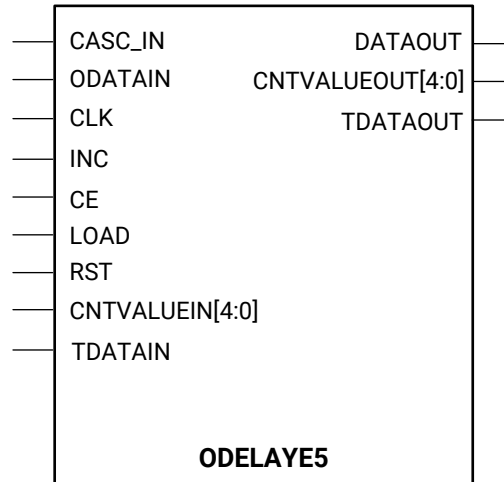
Attribute	Values	Description
CASCADE	FALSE, TRUE	The CASCADE attribute is set to TRUE when the ODELAYE5 is used to cascade the IDELAYE5. This attribute enables an input delay greater than 1.8 ns.

Table 97: IDELAYE5 Ports

Port	I/O	Description
DATAOUT	Output	Delayed data from the IOB
CNTVALUEOUT<4:0>	Output	The CNTVALUEOUT pins are used for reporting the current tap value and reads out the amount of taps in the current delay.
CASC_OUT	Output	Cascade delay to ODELAYE5 in cascade. The CASC_OUT pin is used when cascading from an IDELAYE5 to an ODELAYE5. The CASC_OUT port of the IDELAYE5 is connected to the CASC_IN of the ODELAYE5 in cascade.
IDATAIN	Input	Data from input pin
CASC_RETURN	Input	Cascade delay returning from the ODELAYE5 DATAOUT port. This port is used when cascading the ODELAYE5 and IDELAYE5.
CNTVALUEIN<4:0>	Input	The CNTVALUEIN pins are used for dynamically switching the loadable tap value. The CNTVALUEIN is the number of taps required.
CE	Input	Clock enable for the delay register clock
CLK	Input	Clock used to sample LOAD, CE, and INC
INC	Input	Increment and decrement are controlled by the enable signal (CE). <ul style="list-style-type: none"> INC = 1 increments INC = 0 decrements
LOAD	Input	Load counter value from the CNTVALUEIN bus when High.
RST	Input	The reset pin (RST) is asynchronous with the CLK

Uncalibrated Output Delay

Figure 58: ODELAYE5



X22468-071420

Table 98: ODELAYE5 Attributes

Attribute	Values	Description
CASCADE	FALSE, TRUE	The CASCADE attribute is set to TRUE when the ODELAYE5 is used to cascade the IDELAYE5. This attribute enables a delay greater than 1.8 ns.

Table 99: ODELAYE5 Ports

Port	I/O	Description
DATAOUT	Output	Delayed data to the pin. When cascading the ODELAYE5 to the IDELAYE5, DATAOUT should be connected to the CASC_RETURN pin of the IDELAYE5.
CNTVALUEOUT<4:0>	Output	The CNTVALUEOUT pins are used for reporting the current tap value and reads out the amount of taps in the current delay.
TDATAOUT	Output	Delayed tristate control connects to the IOB.
CASC_IN	Input	Cascade delay from IDELAYE5 CASC_OUT. The CASC_IN pin is used when cascading the ODELAYE5 delay to the IDELAYE5.
ODATAIN	Input	Data from IOB registers
CNTVALUEIN<4:0>	Input	The CNTVALUEIN pins are used for dynamically switching the loadable tap value. The CNTVALUEIN is the number of taps required.
CE	Input	Clock enable for the delay register clock
CLK	Input	Clock used to sample LOAD, CE, and INC
INC	Input	Increment and decrement are controlled by the enable signal (CE). <ul style="list-style-type: none"> INC = 1 increments INC = 0 decrements
LOAD	Input	Load counter value from the CNTVALUEIN bus when High.
RST	Input	The reset pin (RST) is asynchronous with the CLK

Table 99: ODELAYE5 Ports (cont'd)

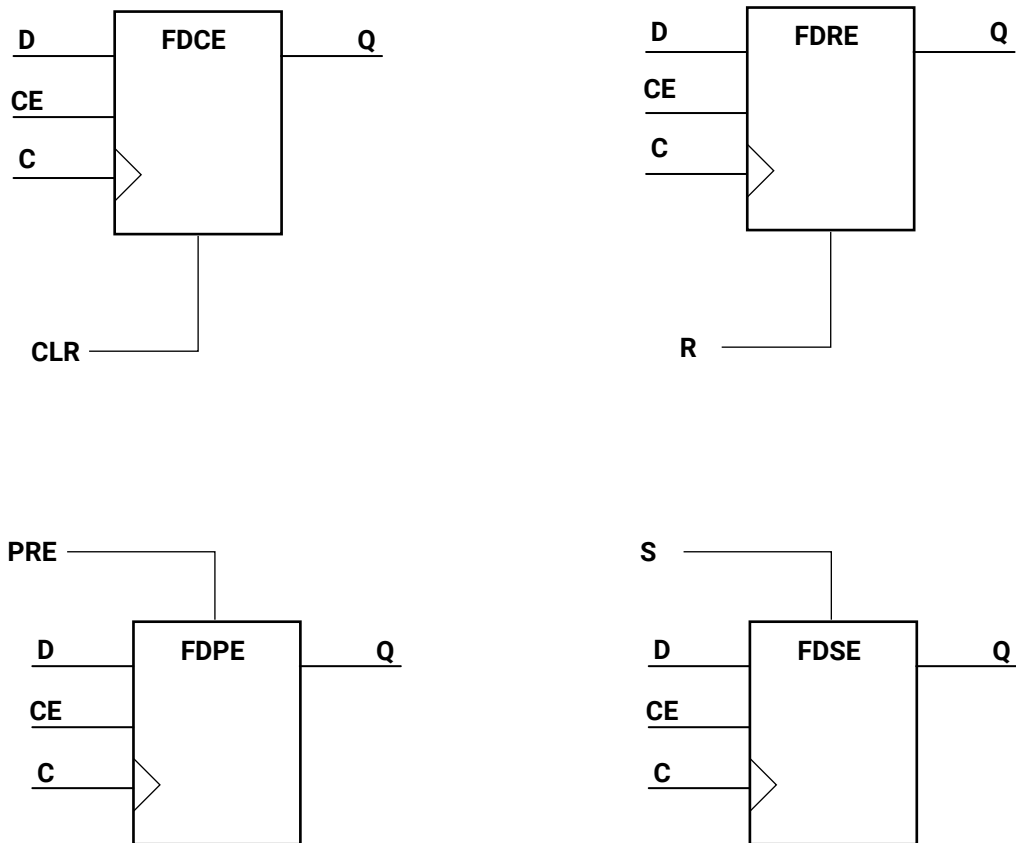
Port	I/O	Description
TDATAIN	Input	Tristate control input

Input Registers

Single Data Rate Flip-Flops

SDR input and output registering inside the NIBBLESLICE use a flip-flop primitive with the IOB = TRUE constraint applied to the flip-flop instance.

Figure 59: Single Data Rate Flip-Flop Primitives



X21636-092418

Table 100: Single Data Rate Flip-Flop (FDCE, FDPE, FDRE, FDSE) Attributes

Attribute	Values	Description
INIT	1'b0, 1'b1	Defines the initial value of the flip-flop

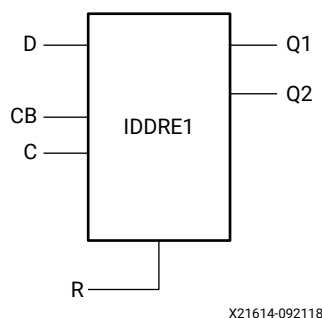
Table 101: Single Data Rate Flip-Flop (FDCE, FDPE, FDRE, FDSE) Ports

Port	I/O	Description
Q	Output	Data output
C	Input	Clock input pin
CE	Input	Active-High clock enable register
D	Input	Data input
CLR	Input	Asynchronous clear (FDCE only)
PRE	Input	Asynchronous preset (FDPE only)
R	Input	Synchronous reset (FDRE only)
S	Input	Synchronous set (FDSE only)

Double Data Rate Input Flip-Flop

Versal devices have dedicated registers in the XIOL to implement input DDR registers. This feature is used by instantiating the `IDDRE1` primitive. The `IDDRE1` primitive supports the following modes of operation:

- **OPPOSITE_EDGE**: Traditional input DDR solution. Data is presented to Q1 on the rising edge and Q2 on the falling edge.
- **SAME_EDGE**: Data is presented to the device logic on the same clock edge.
- **SAME_EDGE_PIPELINED**: Data is presented to the device logic on the same clock edge. Removes the separated effect but incurs clock latency.

 Figure 60: `IDDRE1` Primitive

 Table 102: `IDDRE1` Attributes

Attribute	Values	Description
<code>DDR_CLK_EDGE</code>	<code>OPPOSITE_EDGE</code> , <code>SAME_EDGE</code> , <code>SAME_EDGE_PIPELINED</code>	Sets the <code>IDDRE1</code> mode of operation with respect to the clock edge

Table 103: IDDRE1 Ports

Port	I/O	Description
Q1, Q2	Output	IDDRE1 register outputs
C	Input	Clock input pin
CB	Input	Inverted clock input pin when IS_C_INVERTED = 0 and IS_CB_INVERTED = 0
D	Input	Register input from IOB
R	Input	Asynchronous High reset

Double Data Rate Output Flip-Flop

Versal devices have registers in the IOL to implement output DDR registers as in previous generations. This feature is accessed when instantiating the ODDRE1 primitive. DDR multiplexing is automatic when using the ODDRE1. No manual control of the multiplexer select is needed. This control is generated from the clock. The ODDRE1 primitive supports only the SAME_EDGE mode of operation. The SAME_EDGE mode allows designers to present both data inputs to the ODDRE1 primitive on the rising edge of the ODDRE1 clock which saves CLB and clock resources and increases performance.

Figure 61: ODDRE1 Primitive

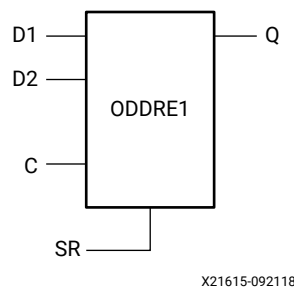


Table 104: ODDRE1 Attributes

Attribute	Values	Description
SRVAL	1'b0, 1'b1	Initializes the value of the ODDRE1 flip-flops

Table 105: ODDRE1 Ports

Port	I/O	Description
Q	Output	ODDRE1 register output
C	Input	Clock input pin
D1,D2	Input	ODDRE1 register inputs
SR	Input	Asynchronous High reset

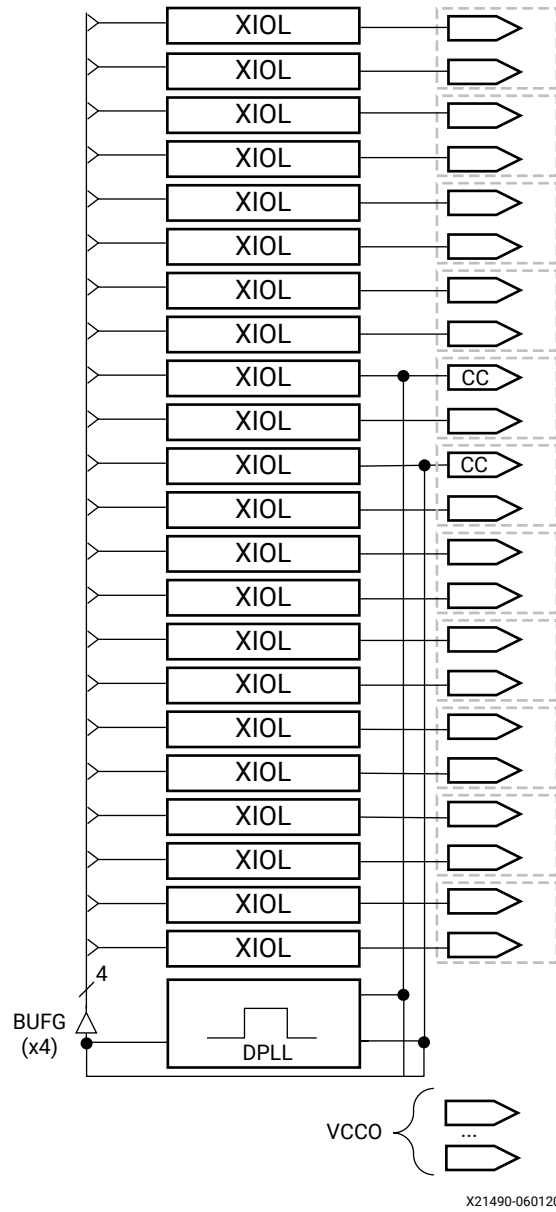
HD IOB Resources

High-density (HD) I/O banks are SelectIO resources designed to support I/O standards with voltages ranging from 1.8V to 3.3V. HDIOs support single-ended and pseudo-differential I/O bidirectional signaling operating at data rates of up to 400 Mb/s. Limited support for true differential inputs (with external termination) is also available to support LVDS and LVPECL clock inputs. HD banks contain interface logic (HD IOL) that includes registers, a DPLL, and static delay lines to support asynchronous, system synchronous, and clock-based source synchronous interfaces.

HD IOB Banking Structure

The HDIO pins are grouped into banks of 22 IOBs (11 I/O pairs) and a DPLL. Each IOB has an independent HD IOL logic block. All IOBs in an HD bank share the same V_{CCO} power supply to power driver logic, receiver logic, and termination. Regardless of whether the IOB is used as an input, output, or bi-directional pin, each I/O standard has a specific V_{CCO} voltage requirement that must be adhered to for the I/O standard to populate a bank. Each HD bank has two dedicated pins with direct access to a clock buffer, referred to as HDGC pins, which allow for distribution of the clock to the rest of the bank. The HDIO bank's DPLL provides clock deskew capabilities to a HDIO bank. For more details on the DPLL, see *Versal ACAP Clocking Resources Architecture Manual* ([AM003](#)).

Figure 62: HD Bank (22 I/O, 11 I/O pairs, 2 Clock-capable Pins, 1 DPLL)



HD IOB Features

INTERNAL V_{REF}

The receivers for HSTL (HSTL_I_18 and DIFF_HSTL_I_18) and SSTL (SSTL18_I and DIFF_SSTL18_I) standards require a reference (V_{REF}) to define input switching thresholds. In HD IOB, these levels are internally provided at $V_{CCO}/2$. In HD IOB, V_{REF} is always internally generated and there are no attributes that must be set to enable the reference. It is automatically inferred when a input that requires V_{REF} is used in a design. See the [HD IOB Supported Standards](#) section for the standards that use internal V_{REF} .

Drive Strength Control

LVC MOS drivers (LVC MOS33, LVC MOS25, and LVC MOS18) and LVTTTL drivers support output drive strength control to size the driver output strength to the load driven. Assigning the DRIVE property 4, 8, or 12 ensures that the HD IOB driver can meet valid logic thresholds for loads of 4 mA, 8 mA, or 12 mA, respectively. The DRIVE attribute uses the following syntax in the XDC file:

```
set_property DRIVE value [get_ports port_name]
```

Slew Control

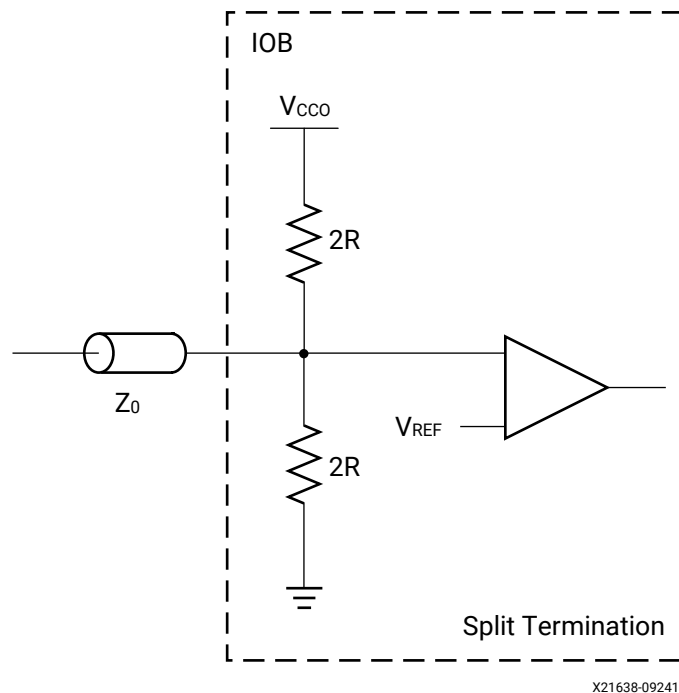
All LVC MOS (LVC MOS33, LVC MOS25, and LVC MOS18) and LVTTTL drivers support FAST and SLOW slew rate options. These options can be used to optimize the driver edge rate. The SLEW attribute uses the following syntax in the XDC file:

```
set_property SLEW value [get_ports port_name]
```

Uncalibrated On-Die Termination (ODT)

The HD IOB provides uncalibrated input split termination for HSTL (HSTL_I_18 and DIFF_HSTL_I_18) and SSTL (SSTL18_I and DIFF_SSTL18_I) I/O standards. The allowed values for the ODT attribute are RTT_48 and RTT_NONE. It defines the equivalent termination to be either 48Ω or to explicitly disable the ODT. The ODT attribute uses the following syntax in the XDC file:

```
set_property ODT value [get_ports port_name]
```


Figure 63: Input Termination to $V_{CC0}/2$ Using Split Termination


PULLUP, PULLDOWN, KEEPER

All HDIO pins can enable a weak internal PULLUP bias to V_{CC0} , a weak internal PULLDOWN to GND, or a weak internal KEEPER circuit that will both pull-up to V_{CC0} and pull-down to GND based on the last driven pin voltage. This feature can be invoked by adding the following possible constraint values for the PULLTYPE attribute to the relevant net of the buffers:

- NONE
- PULLUP
- PULLDOWN
- KEEPER

These attributes use the following syntax in the XDC file:

```
set_property PULLTYPE value [get_ports port_name]
```

HD IOB Supported Standards

Table 106: HD IOB Supported Single-Ended Standards

IOSTANDARD	Required V_{CCO} Level for Input and Output	INTERNAL_VREF Level for Input	DRIVE and Termination Options
LVTTTL	3.3V	N/A	DRIVE: 4, 8, 12
LVCNOS33	3.3V	N/A	DRIVE: 4, 8, 12
LVCNOS25	2.5V	N/A	DRIVE: 4, 8, 12
LVCNOS18	1.8V	N/A	DRIVE: 4, 8, 12
SSTL18_I	1.8V	0.9V	SPLIT
HSTL_I_18	1.8V	0.9V	SPLIT

Table 107: HD IOB Supported Differential Standards

IOSTANDARD	V_{CCO} Level ¹	Drive and Termination Options
LVDS_25 (input only)	2.5V	
LVPECL (input only)	3.3V	
SLVS_400_25 (input only)	2.5V	
SUB_LVDS (input only)	1.8V	
DIFF_HSTL_I_18	1.8V	SPLIT
DIFF_SSTL18_I	1.8V	SPLIT

Notes:

- When on-die input termination is used (ODT is set to a value other than RTT_NONE) or when PULLTYPE leverages V_{CCO} (KEEPER or PULLUP), the V_{CCO} input voltage is as specified. When ODT = RTT_NONE and PULLTYPE is unused, NONE, or PULLDOWN, the V_{CCO} input voltage is any allowed voltage higher than the highest signal level applied to the pin.

HD IOB Primitives

The Vivado Design Suite library includes an extensive list of primitives supporting many I/O primitives. The generic primitives can each support most of the single-ended I/O standards:

- **IBUF:** Input buffer
- **IOBUF:** Bidirectional buffer
- **IOBUF_INTERMDISABLE:** Bidirectional buffer with input buffer disable and on-die input termination disable control
- **OBUF:** Output buffer
- **OBUFFT:** Tristate output buffer

These generic primitives can each support most of the available differential I/O standards:

- **IBUFDS:** Differential input buffer
- **IBUFDS_INTERMDISABLE:** Differential input buffer with on-die input termination disable control and input buffer disable
- **IBUFDS_DIFF_OUT:** Differential input buffer with complementary outputs
- **IBUFDS_DIFF_OUT_INTERMDISABLE:** Differential input buffer with complementary outputs, input buffer disable, and on-die input termination disable control
- **IOBUFDS:** Differential bidirectional buffer
- **IOBUFDS_INTERMDISABLE:** Differential bidirectional buffer with on-die input termination disable control and input buffer disable
- **IOBUFDS_DIFF_OUT:** Differential bidirectional buffer with complementary outputs from the input buffer
- **IOBUF_DIFF_OUT_INTERMDISABLE:** Differential bidirectional buffer with complementary outputs from the input buffer with on-die input termination disable controls and input buffer disable controls
- **OBUFDS:** Differential output buffer
- **OBUFTDS:** Differential tristate output buffer

HD IOB Supported Single-Ended Standards

LVTTL

The LVTTL standard is a general-purpose EIA/JESD standard for 3.3V applications that uses a single-ended CMOS input buffer and a push-pull output buffer.

LVC MOS

The low-voltage CMOS (LVC MOS) standards LVC MOS33 (3.3V), LVC MOS25 (2.5V), and LVC MOS18 (1.8V) are implemented with CMOS transistors. This standard is defined in the JEDEC standard JESD 8C.01.

Table 108: Allowed Attributes for LVTTL, LVC MOS18, LVC MOS25, and LVC MOS33 I/O Primitives

Attributes	IBUF/IBUFE3		OBUF/OBUFT/IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	LVTTL, LVC MOS18, LVC MOS25, LVC MOS33		LVTTL, LVC MOS18, LVC MOS25, LVC MOS33	
DRIVE	N/A		4, 8, 12	12
SLEW	N/A		FAST, SLOW	SLOW

SSTL

SSTL18_I is defined by the JEDEC standard JESD8-15, and is used for DDR2 SDRAM interfaces. For some topologies (such as short, point-to-point interfaces), the class-I driver can result in reduced overshoot and better signal integrity.

HSTL

The high-speed transceiver logic (HSTL) HSTL_I_18 (1.8V) standard is a general-purpose high-speed bus standard as defined by the JEDEC standard JESD8-6.

Table 109: Allowed Attributes for SSTL18_I and HSTL_I_18 I/O Primitives

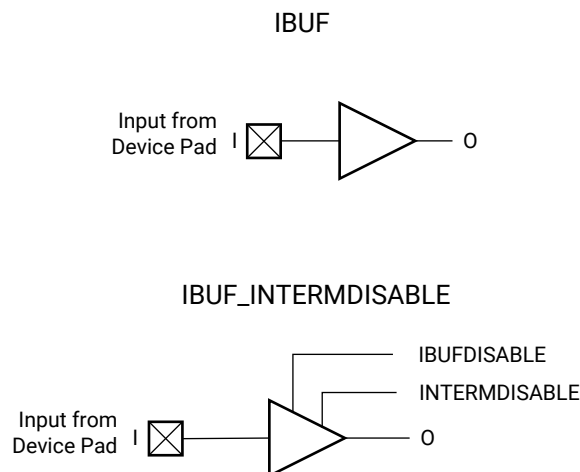
Attributes	IBUF/IBUFE3		OBUF/OBUFT		IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Value	Default	Allowed Values	Default
IOSTANDARD	SSTL18_I, HSTL_I_18		SSTL18_I, HSTL_I_18		SSTL18_I, HSTL_I_18	
SLEW	N/A		FAST, SLOW	SLOW	FAST, SLOW	SLOW
ODT	RTT_48, RTT_NONE	RTT_NONE	N/A		RTT_48, RTT_NONE	RTT_NONE

UNDEFINED Default IOSTANDARD

When an IOSTANDARD is not defined by the user, the default assignment for the IOSTANDARD defaults to UNDEFINED. For a Versal ACAP design to complete implementation, a non-default IOSTANDARD must be defined with one of the valid I/O standards described in this section. The UNDEFINED standard acts as a placeholder to allow a design to complete the early stages or implementation.

Single-Ended Input Buffer Primitives

Figure 64: Single-Ended Input Buffer Primitives



X21628-100720

Table 110: IBUF and IBUF_INTERMDISABLE Attributes

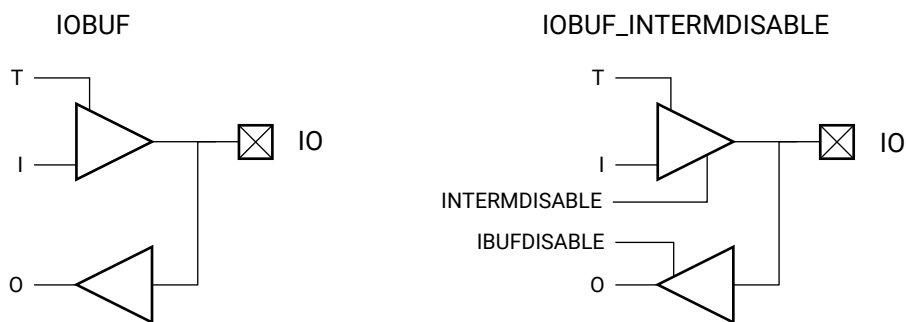
Attribute	Values	Description
IOSTANDARD	See HD IOB Supported Standards	Assigns an I/O standard to the element.
USE_IBUFDISABLE	TRUE, FALSE	Enables the IBUFDISABLE port. (IBUF_INTERMDISABLE only)

Table 111: IBUF and IBUF_INTERMDISABLE Ports

Port	I/O	Description
O	Output	Buffer output representing the input path to the device.
I	Input	Input port connection. Connect directly to top-level port in the design.
IBUFDISABLE	Input	The IBUFDISABLE pin can disable the input buffer and force the O output to the internal logic to a logic High when the IBUFDISABLE signal is asserted High. (IBUF_INTERMDISABLE only).
INTERMDISABLE	Input	Control to enable/disable of on-chip input termination. This is generally used to reduce power in long periods of an idle state. (IBUF_INTERMDISABLE only)

Single-Ended Bidirectional Buffer Primitives

Figure 65: Single-Ended Bidirectional Buffer Primitives



X21629-092318

Table 112: IOBUF, IOBUF_INTERMDISABLE Attributes

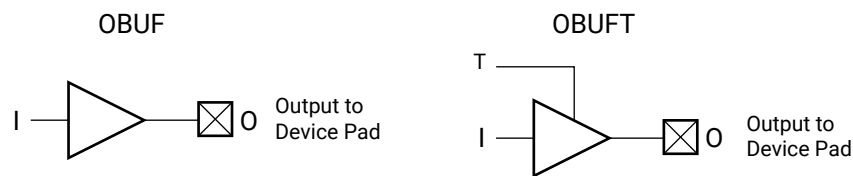
Attribute	Values	Description
DRIVE	2, 4, 8, 12	Specifies the drive strength of the output.
SLEW	SLOW, FAST	Specifies the slew rate of the output.
IOSTANDARD	See HD IOB Supported Standards	Assigns an I/O standard to the element.
USE_IBUFDISABLE	TRUE, FALSE	Enables IBUFDISABLE port. (IOBUF_INTERMDISABLE only)

Table 113: IOBUF, IOBUF_INTERMDISABLE Ports

Port	I/O	Description
IO	Inout	Inout port connection. Connect directly to top-level port in the design.
O	Output	Output path of the buffer.
I	Input	Input port connection. Connect directly to top-level port in the design.
T	Input	Tristate enable input signifying whether the buffer acts as an input or output.
IBUFDISABLE	Input	The IBUFDISABLE pin can disable the input buffer and force the O output to the internal logic to a logic High when the IBUFDISABLE signal is asserted High. (IOBUF_INTERMDISABLE only)
INTERMDISABLE (IOBUF_INTERMDISABLE)	Input	Control to enable/disable input termination. This is generally used to reduce power in long periods of an idle state. (IOBUF_INTERMDISABLE only)

Single-Ended Output Buffer Primitive

Figure 66: Single-Ended Output Buffer Primitives



X21622-092318

Table 114: OBUF and OBUFT Attributes

Attribute	Values	Description
SLEW	SLOW, FAST	Specifies the slew rate of the output.
DRIVE	2, 4, 8, 12	Specifies the drive strength of the output
IOSTANDARD	See HD IOB Supported Standards	Assigns an I/O standard to the element.

Table 115: OBUF and OBUFT Ports

Port	I/O	Description
O	Output	Output of OBUF to be connected directly to top-level output port.
I	Input	Input of OBUF. Connect to the logic driving the output port.
T	Input	Tristate enable input. (OBUFT only)

HD IOB Supported Differential Standards

LVDS

In HD IOB, the low-voltage differential signals (LVDS) standard LVDS_25 (2.5V) is a receiver only standard with no internal termination options.

LVPECL

In HD IOB, the LVPECL (3.3V) standard is a receiver only standard with no internal termination options.

SLVS_400

In HD IOB, the SLVS-400_25 can be used to receive JESD8-13 signals.

Table 116: Allowed Attributes for LVDS25, LVPECL, and SLVS_400 I/O Primitives

Attributes	IBUFDS/IBUFDS3/OBUFDS/OBUFTDS/IOBUFDS/IOBUFDSE3	
	Allowed Values	Default
IOSTANDARD	LVDS_25, LVPECL, SLVS_400_25	

DIFF_SSTL

The DIFF_SSTL18 standard is defined by the JEDEC standard JESD8-15 and is used for DDR2 SDRAM interfaces.

DIFF_HSTL

The differential high-speed transceiver logic (HSTL) standard DIFF_HSTL_I_18 (1.8V) is a general-purpose high-speed bus standard defined by the JEDEC standard JESD8-6.

Table 117: Allowed Attributes for DIFF_SSTL18_I and DIFF_HSTL_I_18 I/O Primitives

Attributes	IBUF/IBUFE3		OBUF/OBUFT		IOBUF/IOBUFE3	
	Allowed Values	Default	Allowed Value	Default	Allowed Values	Default
IOSTANDARD	DIFF_SSTL18_I, DIFF_HSTL_I_18		DIFF_SSTL18_I, DIFF_HSTL_I_18		DIFF_SSTL18_I, DIFF_HSTL_I_18	
SLEW	N/A		FAST, SLOW	SLOW	FAST, SLOW	SLOW
ODT	RTT_48, RTT_NONE	RTT_NONE	N/A		RTT_48, RTT_NONE	RTT_NONE

DIFF_UNDEFINED Default IOSTANDARD

When an IOSTANDARD is not defined by the user, the default assignment for the IOSTANDARD defaults to DIFF_UNDEFINED. For a Versal ACAP design to complete implementation, a non-default IOSTANDARD must be defined with one of the valid I/O standards described in this section. The DIFF_UNDEFINED standard acts as a placeholder to allow a design to complete the early stages or implementation.

Differential Input Buffer Primitives

Figure 67: Differential Input Buffer Primitives

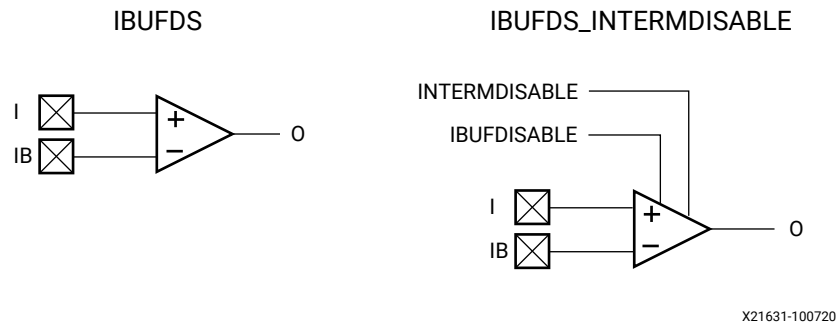


Figure 68: Differential DIFF_OUT Input Buffer Primitives

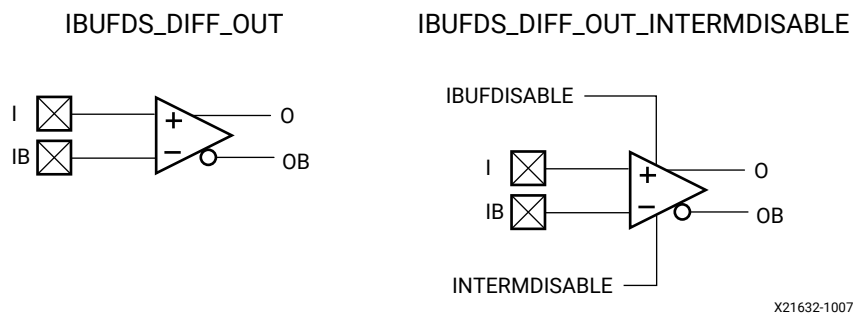


Table 118: IBUFDS, IBUFDS_DIFF_OUT, IBUFDS_IBUFDS_DIFF_OUT_INTERMDISABLE, and IBUFDS_INTERMDISABLE Attributes

Attribute	Values	Description
IOSTANDARD	See HD IOB Supported Standards	Assigns an I/O standard to the element.
DIFF_TERM	TRUE, FALSE	Turns the built-in differential termination on (TRUE) or off (FALSE).
DQS_BIAS	TRUE, FALSE	Provides pull-up/pull-down feature required for some DQS memory interface pins or provides DC bias for certain LVDS applications.
USE_IBUFDISABLE	TRUE, FALSE	Enables use of the IBUFDISABLE port. (IBUFDS_DIFF_OUT_INTERMDISABLE and IBUFDS_INTERMDISABLE only)

Table 119: IBUFDS, IBUFDS_DIFF_OUT, IBUFDS_DIFF_OUT_INTERMDISABLE, and IBUFDS_INTERMDISABLE Ports

Port	I/O	Description
O	Output	Buffer output representing the input path to the device.
OB	Output	Complimentary buffer output representing the input path to the device. (IBUFDS_DIFF_OUT, IBUFDS_DIFF_OUT_INTERMDISABLE only)
I	Input	Input port connection. Connect directly to top-level P-side port in the design.

Table 119: IBUFDS, IBUFDS_DIFF_OUT, IBUFDS_DIFF_OUT_INTERMDISABLE, and IBUFDS_INTERMDISABLE Ports (cont'd)

Port	I/O	Description
IB	Input	Input port connection. Connect directly to top-level N-side port in the design.
IBUFDISABLE	Input	The IBUFDISABLE pin can disable the input buffer and force the O output to the internal logic to a logic High when the IBUFDISABLE signal is asserted High. (IBUFDS_DIFF_OUT_INTERMDISABLE and IBUFDS_INTERMDISABLE only)

Differential Bidirectional Buffer Primitives

Figure 69: Differential Bidirectional Buffer Primitives

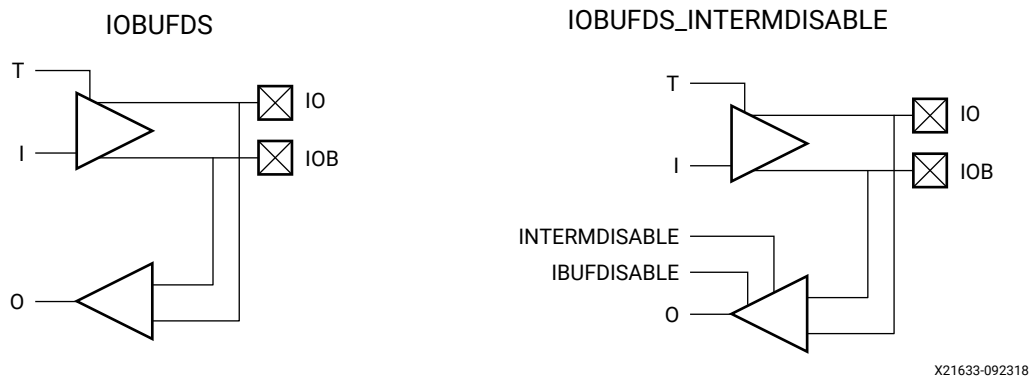


Figure 70: Differential Bidirectional DIFF_OUT Buffer Primitives

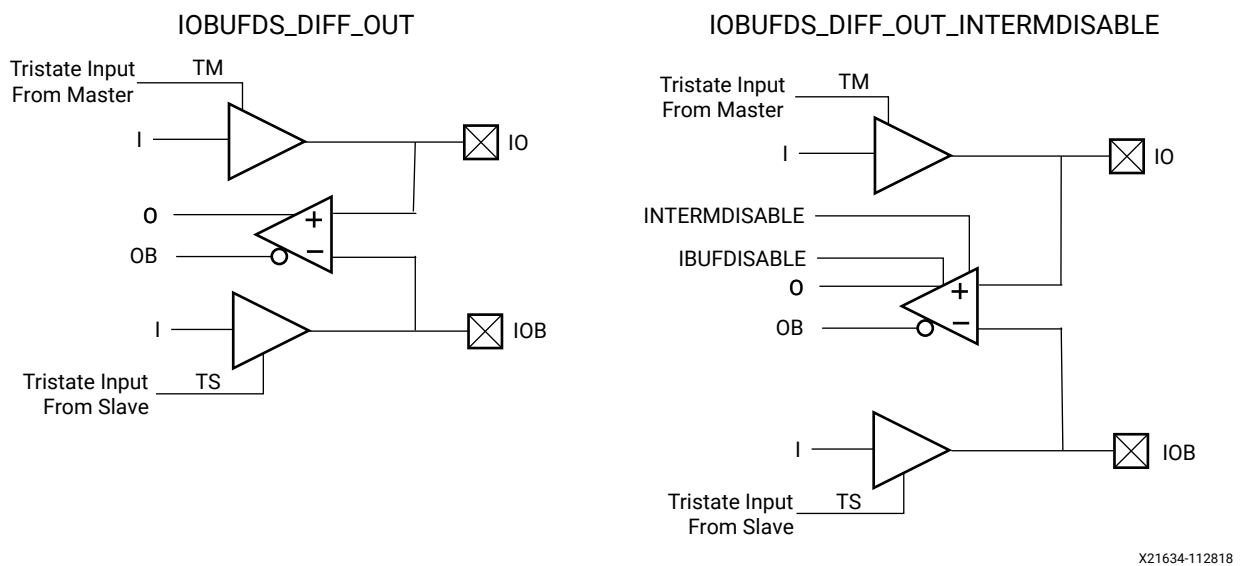


Table 120: IOBUFDS, IOBUFDS_DIFF_OUT, IOBUFDS_DIFF_OUT_INTERMDISABLE, and IOBUFDS_INTERMDISABLE Attributes

Attribute	Values	Description
SLEW	SLOW, FAST	Specifies the drive strength of the output.
IOSTANDARD	See HD IOB Supported Standards	Assigns an I/O standard to the element.
USE_IBUFDISABLE	TRUE, FALSE	Enables use of IBUFDISABLE port. (IOBUFDS_INTERMDISABLE and IOBUFDS_DIFF_OUT_INTERMDISABLE only)

Table 121: IOBUFDS, IOBUFDS_DIFF_OUT, IOBUFDS_DIFF_OUT_INTERMDISABLE, and IOBUFDS_INTERMDISABLE Ports

Port	I/O	Description
IO	Inout	Inout port connection. Connect directly to top-level P side port in the design.
IOB	Inout	Inout port connection. Connect directly to top-level N side port in the design. (IOBUFDS_DIFF_OUT, IOBUF_DIFF_OUT_INTERMDISABLE only)
O	Output	Output path of the buffer representing the input path to the device.
OB	Output	Complimentary buffer output representing the input path to the device. (IOBUFDS_DIFF_OUT, IOBUF_DIFF_OUT_INTERMDISABLE only)
I	Input	Input port connection. Connect directly to top-level P-side port in the design.
T	Input	Tristate enable input signifying whether the buffer acts as an input or output. (IOBUFDS, IOBUFDS_INTERMDISABLE only)
IBUFDISABLE	Input	The IBUFDISABLE pin can disable the input buffer and force the O output to the internal logic to a logic High when the IBUFDISABLE signal is asserted High. (IOBUFDS_INTERMDISABLE, IOBUFDS_DIFF_OUT_INTERMDISABLE only)
INTERMDISABLE	Input	Control to enable/disable DCI termination. This is generally used to reduce power in long periods of an idle state. (IOBUFDS_INTERMDISABLE, IOBUFDS_DIFF_OUT_INTERMDISABLE only)
TM	Input	Tristate enable input for the P-side or master side signifying whether the buffer acts as an input or output. This pin must be connected to the same signal as the TS input. (IOBUFDS_DIFF_OUT, IOBUFDS_DIFF_OUT_INTERMDISABLE only)
TS	Input	Tristate enable input for the N-side or slave side signifying whether the buffer acts as an input or output. This pin must be connected to the same signal as the TM input. (IOBUFDS_DIFF_OUT and IOBUFDS_DIFF_OUT_INTERMDISABLE only)

Differential Output Buffer Primitives

Figure 71: Differential Output Buffer Primitives

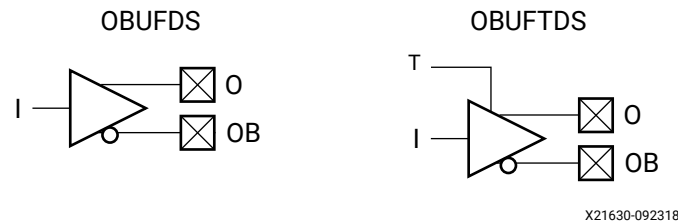


Table 122: OBUFDS, OBUFTDS Attributes

Attribute	Values	Description
SLEW	SLOW, FAST, MEDIUM	Specifies the slew rate of the output driver. (OBUFDS, OBUFTDS)
IOSTANDARD	See HD IOB Supported Standards	Assigns an I/O standard to the element.

Table 123: OBUFDS, OBUFTDS Ports

Port	I/O	Description
O	Output	Output connected directly to P-side top-level output port.
OB	Output	Output connected directly to N-side top-level output port.
I	Input	Buffer input
T	Input	Tristate enable input (OBUFTDS only)

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. 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](#))
2. *Versal ACAP Clocking Resources Architecture Manual* ([AM003](#))
3. *Versal ACAP Technical Reference Manual* ([AM011](#))
4. *Versal ACAP Packaging and Pinouts Architecture Manual* ([AM013](#))
5. *Advanced I/O Wizard LogiCORE IP Product Guide* ([PG320](#))

Please Read: Important Legal Notices

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

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

© Copyright 2020 Xilinx, Inc. Xilinx, the Xilinx logo, Alveo, Artix, Kintex, Spartan, Versal, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. AMBA, AMBA Designer, Arm, ARM1176JZ-S, CoreSight, Cortex, PrimeCell, Mali, and MPCore are trademarks of Arm Limited in the EU and other countries. PCI, PCIe, and PCI Express are trademarks of PCI-SIG and used under license. All other trademarks are the property of their respective owners.