

# UltraScale Architecture SelectIO Resources

## *User Guide*

UG571 (v1.16) January 14, 2025

AMD Adaptive Computing is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing non-inclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards. Follow this [link](#) for more information.





# Table of Contents

## Chapter 1: SelectIO Interface Resources

Introduction to the UltraScale Architecture .....	4
I/O Tile Overview .....	5
Differences from Previous Generations .....	7
SelectIO Technology Resources Introduction .....	8
SelectIO Interface General Guidelines .....	14
DCI—Only Available in the HP I/O Banks .....	18
Uncalibrated Input Termination in I/O Banks .....	31
SelectIO Interface Primitives .....	39
SelectIO Interface Attributes and Constraints .....	57
Supported I/O Standards and Terminations .....	67
Internal Differential Termination Behavior in Differential I/O Standards .....	132
Rules for Combining I/O Standards in the Same Bank .....	133
Simultaneous Switching Outputs .....	142

## Chapter 2: SelectIO Interface Logic Resources

Bank Overview .....	144
Component Primitives .....	150
Native Primitives .....	198

## Chapter 3: High Density I/O Resources

Introduction to High Density I/O Banks .....	339
HD I/O Bank Resources .....	340
HD I/O Bank Features .....	340
HD I/O Supported Standards .....	343
HD I/O Interface Logic .....	345

## Appendix A: Termination Options for Simultaneous Switching Noise Analysis

Termination Options .....	347
---------------------------	-----

## Appendix B: Additional Resources and Legal Notices

Finding Additional Documentation .....	352
Support Resources .....	353



<b>References</b> .....	<b>353</b>
<b>Revision History</b> .....	<b>354</b>
<b>Please Read: Important Legal Notices</b> .....	<b>361</b>

# SelectIO Interface Resources

---

## Introduction to the UltraScale Architecture

The AMD UltraScale™ architecture is the first ASIC-class architecture to enable multi-hundred gigabit-per-second levels of system performance with smart processing, while efficiently routing and processing data on-chip. UltraScale architecture-based devices address a vast spectrum of high-bandwidth, high-utilization system requirements by using industry-leading technical innovations, including next-generation routing, ASIC-like clocking, 3D-on-3D ICs, multiprocessor SoC (MPSoC) technologies, and new power reduction features. The devices share many building blocks, providing scalability across process nodes and product families to leverage system-level investment across platforms.

AMD Spartan™ UltraScale+™ devices provide high I/O to logic ratio, integrated memory controllers, and advanced I/O that help Industrial, Vision and Healthcare (IVH), Audio, Video and Broadcast (AVB), Automotive, and other FPGA application developers looking to build cost-optimized solutions. Available in a wide array of packaging options, this family delivers a balance of cost, power, performance, and size.

AMD Artix™ UltraScale+ devices provide high serial bandwidth and signal compute density in a cost-optimized device for critical networking applications, vision and video processing, and secured connectivity. Coupled with the innovative InFO packaging, which provides excellent thermal and power distribution, Artix UltraScale+ FPGAs are perfectly suited to applications requiring high compute density in a small footprint.

AMD Kintex™ UltraScale+ devices provide the best price/performance/watt balance in a FinFET node, delivering the most cost-effective solution for high-end capabilities, including transceiver and memory interface line rates as well as 100G connectivity cores. Our newest mid-range family is ideal for both packet processing and DSP-intensive functions and is well suited for applications including wireless MIMO technology, Nx100G networking, and data center.

Kintex UltraScale devices provide the best price/performance/watt at 20 nm and include the highest signal processing bandwidth in a mid-range device, next-generation transceivers, and low-cost packaging for an optimum blend of capability and cost-effectiveness. The family is ideal for packet processing in 100G networking and data centers applications as well as DSP-intensive processing needed in next-generation medical imaging, 8k4k video, and heterogeneous wireless infrastructure.

AMD Virtex™ UltraScale+™ devices provide the highest performance and integration capabilities in a FinFET node, including both the highest serial I/O and signal processing bandwidth, as well as the highest on-chip memory density. As the industry's most capable FPGA family, the Virtex UltraScale+ devices are ideal for applications including 1+Tb/s networking and data center and fully integrated radar/early-warning systems.

Virtex UltraScale devices provide the greatest performance and integration at 20 nm, including serial I/O bandwidth and logic capacity. As the industry's only high-end FPGA at the 20 nm process node, this family is ideal for applications including 400G networking, large scale ASIC prototyping, and emulation.

AMD Zynq™ UltraScale+ devices provide 64-bit processor scalability while combining real-time control with soft and hard engines for graphics, video, waveform, and packet processing. Integrating an Arm®-based system for advanced analytics and on-chip programmable logic for task acceleration creates unlimited possibilities for applications including 5G Wireless, next generation ADAS, and industrial Internet-of-Things.

The UltraScale architecture documentation suite is available at [docs.amd.com](https://docs.amd.com).

---

## I/O Tile Overview

UltraScale architecture-based devices provide various I/O offerings: High-performance (HP), high-density (HD), and high-range (HR) I/O banks.

- The HP I/O banks are designed to meet the performance requirements of high-speed memory and other chip-to-chip interfaces with voltages up to 1.8V.
- The HR I/O banks are designed to support a wider range of I/O standards with voltages up to 3.3V.
- The HD I/O banks are designed to support low-speed interfaces.

UltraScale devices contain different combinations of HP, HD, and HR I/O banks, and not all types of banks are supported in all devices. The *UltraScale Architecture and Product Overview* (DS890) [Ref 3] documents the available number of each type of bank for all devices.

- Kintex UltraScale and Virtex UltraScale families have high-performance I/O banks (HP I/Os) and high-range I/O banks (HR I/Os) with corresponding logic resources.
  - [Chapter 1, SelectIO Interface Resources](#) describes the electrical behavior of the output drivers and input receivers, and gives detailed examples of many standard interfaces available in these devices.
  - [Chapter 2, SelectIO Interface Logic Resources](#) describes the I/O logic resources available in these devices.

- Any references to Mobile Industry Processor Interface (MIPI) D-PHY or HD I/O in these chapters do not apply to these devices.
- The Zynq UltraScale+, Artix UltraScale+, Kintex UltraScale+, and Virtex UltraScale+ families have high-performance I/O banks (HP I/Os) with enhanced MIPI D-PHY abilities and the corresponding logic resources. They also have high-density I/Os (HD I/Os) with corresponding logic resources.
  - [Chapter 1, SelectIO Interface Resources](#) describes the electrical behavior of the output drivers and input receivers, and gives detailed examples of many standard interfaces available in these devices for HP I/Os.
  - [Chapter 2, SelectIO Interface Logic Resources](#) describes the I/O logic resources available in these devices for HP I/Os.
  - [Chapter 3, High Density I/O Resources](#) describes the electrical and logical features of HD I/Os that are available in Zynq UltraScale+ devices, Artix UltraScale+ FPGAs, Kintex UltraScale+ FPGAs, and some Virtex UltraScale+ FPGAs.
- Any references to HR I/Os in the aforementioned chapters do not apply to these devices.

Outside the aforementioned information, the content in the rest of this chapter does not pertain to HD I/Os. HD I/O information is only described in [Chapter 3, High Density I/O Resources](#).

**Note:** Spartan UltraScale+ devices have unique HD I/O features and XP5IO resources that differ from other devices in the UltraScale+ family. For more information on the SelectIO resources in Spartan UltraScale+ devices, refer to *Spartan UltraScale+ FPGAs SelectIO Resources User Guide* (UG861) [\[Ref 4\]](#).

[Table 1-1](#) highlights the features supported in the HP and HR I/O banks. See the specific UltraScale device data sheets [\[Ref 2\]](#) for details on the performance and other electrical requirements of the HP and HR I/O banks.

**Table 1-1: Supported Features in the HR and HP I/O Banks**

Feature	HP I/O Banks	HR I/O Banks
3.3V I/O standards <sup>(1)</sup>	N/A	Supported
2.5V I/O standards <sup>(1)</sup>	N/A	Supported
1.8V I/O standards <sup>(1)</sup>	Supported	Supported
1.5V I/O standards <sup>(1)</sup>	Supported	Supported
1.35V I/O standards <sup>(1)</sup>	Supported	Supported
1.2V I/O standards <sup>(1)</sup>	Supported	Supported
1.0V POD I/O standard	Supported	N/A
LVDS signaling	Supported <sup>(2)</sup>	Supported
Digitally-controlled impedance (DCI) and DCI cascading	Supported	N/A
Internal V <sub>REF</sub>	Supported	Supported

Table 1-1: Supported Features in the HR and HP I/O Banks (Cont'd)

Feature	HP I/O Banks	HR I/O Banks
Internal differential termination (DIFF_TERM)	Supported	Supported
IDELAY	Supported	Supported
ODELAY	Supported	Supported
IDELAYCTRL	Supported	Supported
ISERDES	Supported	Supported
OSERDES	Supported	Supported
Transmitter pre-emphasis	Supported	Supported <sup>(3)</sup>
Receiver equalization	Supported	Supported
Receiver offset control	Supported	Not supported
Receiver $V_{REF}$ scan	Supported	Not supported
MIPI D-PHY	Supported in Virtex UltraScale+, Kintex UltraScale+, Artix UltraScale+, and Zynq UltraScale+ devices	Not supported

**Notes:**

1. The [I/O Bank Type](#) column in [Table 1-78](#) shows the specific I/O standards that are available in the HP and HR I/O banks.
2. Although LVDS is generally considered a 2.5V I/O standard, it is supported in both the HR and HP I/O banks.
3. Only LVDS pre-emphasis is supported in HR I/O banks.

## Differences from Previous Generations

UltraScale devices support many of the same features supported in 7 series devices. However, there are some useful new features, along with changes to several existing features. These new features and changes include:

- Each I/O bank contains 52 SelectIO interface pins. In some devices, there are some HR I/O mini-banks containing 26 SelectIO pins, each with their own independent power supply and  $V_{REF}$  pin.



**TIP:** All references in this user guide discussing HR I/O banks also apply to HR I/O mini-banks.

- Support for pseudo-open-drain logic standards (POD).
- Series output termination control is available in HP I/O banks for improved signal integrity and ease of board design.
- Internal  $V_{REF}$  level scan (HP I/O banks only). One dedicated external  $V_{REF}$  pin per bank.

- Pre-emphasis is available for the DDR4 standard in HP I/O banks and the LVDS TX standard in HP/HR I/O banks. Pre-emphasis reduces inter-symbol interference and minimizes the effects of transmission line losses.
- Linear equalization on  $V_{REF}$ -based receivers (in HP I/O banks) and differential receivers (in both HP and HR I/O banks) is available to overcome high-frequency losses through the transmission channel.
- Receiver offset cancellation is available for some I/O standards to compensate for process variations (HP I/O banks only).
- Digitally controlled impedance (DCI) is only available in HP I/O banks. DCI uses only one reference resistor per bank,  $240\Omega$  to GND on the VRP pin. The values of the driver or input termination are determined by the OUTPUT\_IMPEDANCE and on-die termination (ODT) attributes, respectively.
- $V_{CCAUX\_IO}$  only supports a nominal voltage level of 1.8V.
- A SLEW value of *MEDIUM* is supported in HP I/O banks.
- The DCITERMDISABLE port can control both DCI and non-DCI on-die input termination features in HP I/O banks.
- Where applicable, asserting IBUFDISABLE causes the input to the interconnect logic to be a 0. This is different from the resulting 1 after asserting IBUFDISABLE in 7 series devices.
- The bit slice is effectively a physical layer (PHY) block that replaces and enhances the functionality of the Component mode primitives. This PHY block gives tighter control over timing and provides new features enabling higher data rate reception in UltraScale devices. See [Native Primitives in Chapter 2](#).
- MIPI D-PHY transmitter and receiver functions are supported in the HP I/Os specific to the Virtex UltraScale+, Kintex UltraScale+, Artix UltraScale+, and Zynq UltraScale+ devices.

---

## SelectIO Technology Resources Introduction

All UltraScale 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 using digitally controlled impedance (DCI), and the ability to internally generate a reference voltage (INTERNAL\_VREF).



---

**IMPORTANT:** *HR I/O banks do not have DCI. Therefore, any reference to DCI in this user guide does not apply to the HR I/O banks.*

---

With some exceptions, each I/O bank contains 52 SelectIO pins, where 48 can implement both single-ended and differential I/O standards. The other four pins, including the

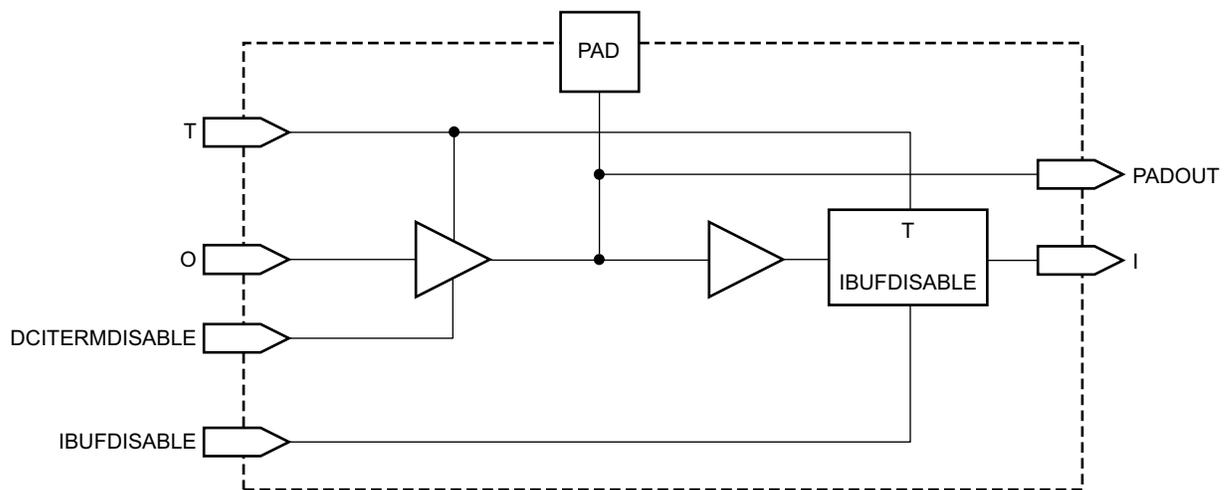
multipurpose VRP pin, are single-ended (only) IOBs. Every SelectIO resource contains input, output, and 3-state drivers.

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, and POD.
- Differential I/O standards are, for example, LVDS, Mini\_LVDS, RSDS, PPDS, BLVDS, TMDS, SLVS, LVPECL, SUB\_LVDS, and differential HSTL, POD, HSUL, and SSTL.

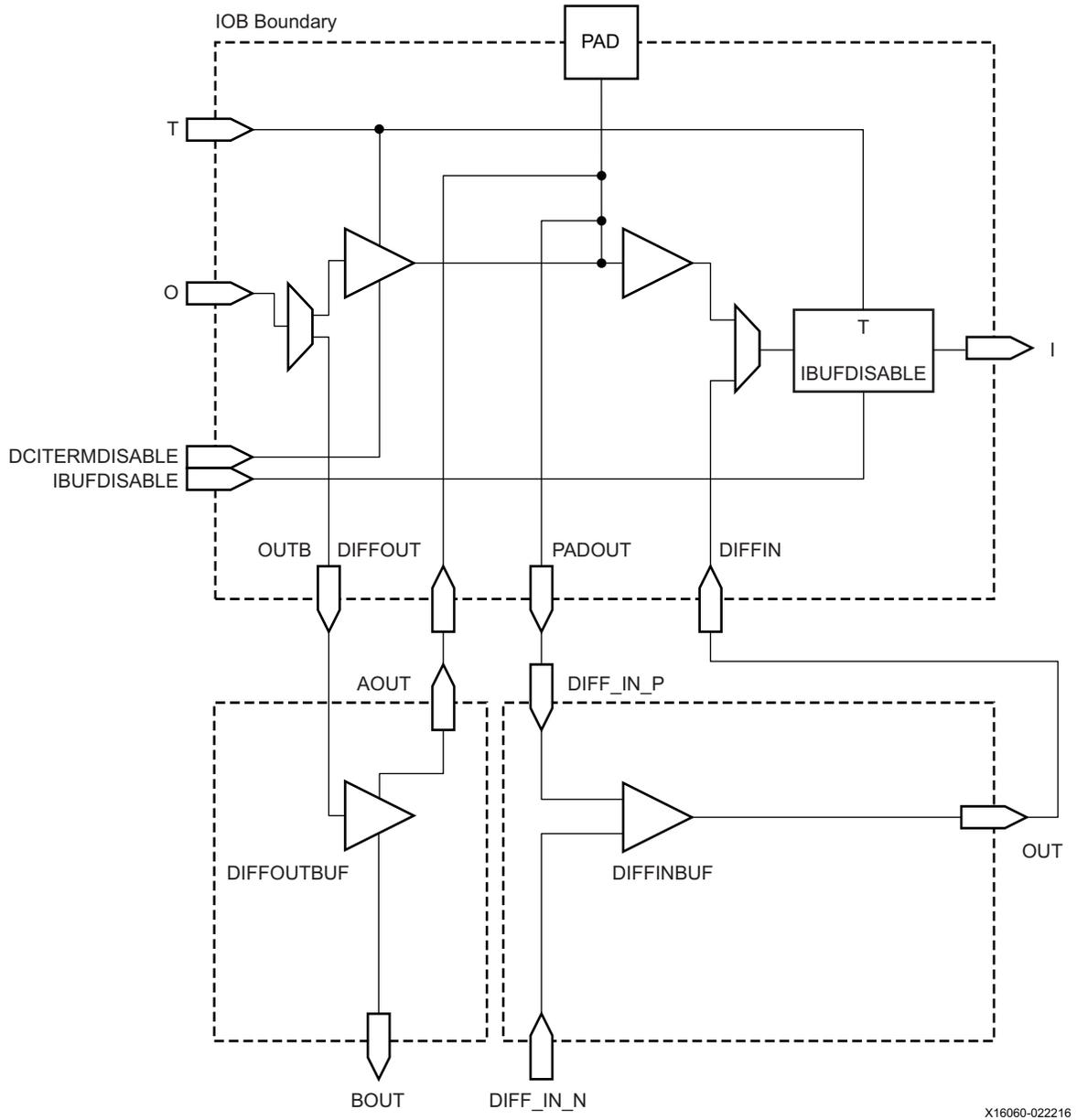
When not used as a VRP pin, the multipurpose VRP pin in each bank can only be used with single-ended I/O standards. [Figure 1-1](#) shows the single-ended (only) HP I/O block (IOB) and its connections to the internal logic and the device pad. [Figure 1-2](#) shows the standard HP IOB. [Figure 1-3](#) shows the single-ended (only) HR IOB. [Figure 1-4](#) shows the standard HR IOB. [Figure 1-5](#) shows the relative location of the single-ended IOBs within a bank. When not configured, I/O drivers are 3-stated and I/O receivers are weakly pulled-down.

Each IOB has a direct connection to bit slice components containing the input and output resources for serialization, deserialization, signal delay, clock, data, and 3-state control, and registering for the IOB. The bit slice components can be used in Component mode individually as IDELAY, ODELAY, ISERDES, OSERDES, and input and output registers. They can also be used at a lower granularity level as RX\_BITSLICE (input), TX\_BITSLICE (output), and RXTX\_BITSLICE (bidirectional) components where all of the bit slice functions are grouped together in a single interface. See [Chapter 2, SelectIO Interface Logic Resources](#) for more information.



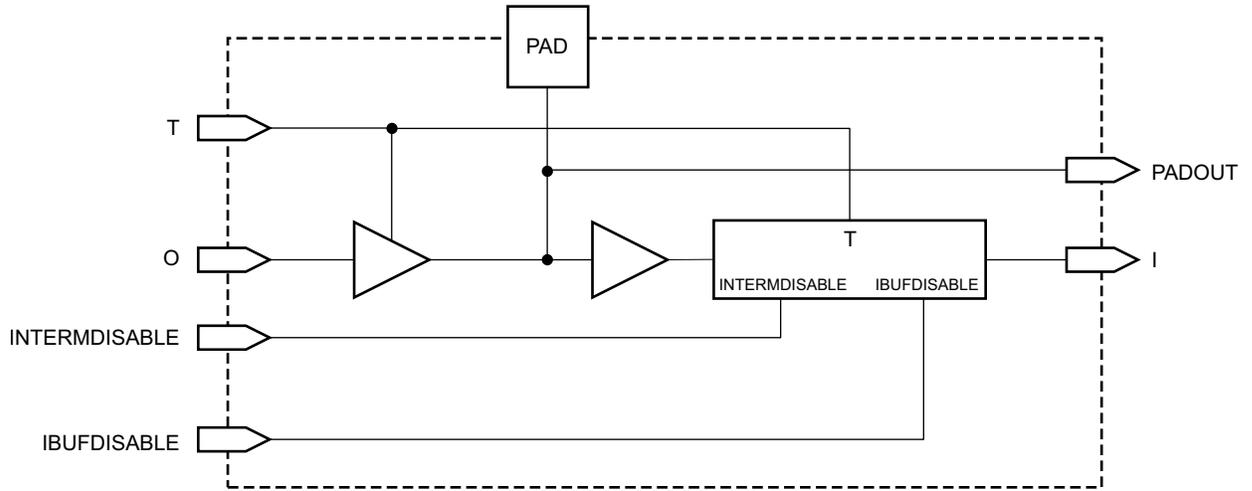
X16145-080216

Figure 1-1: Single-Ended (Only) HP IOB Diagram



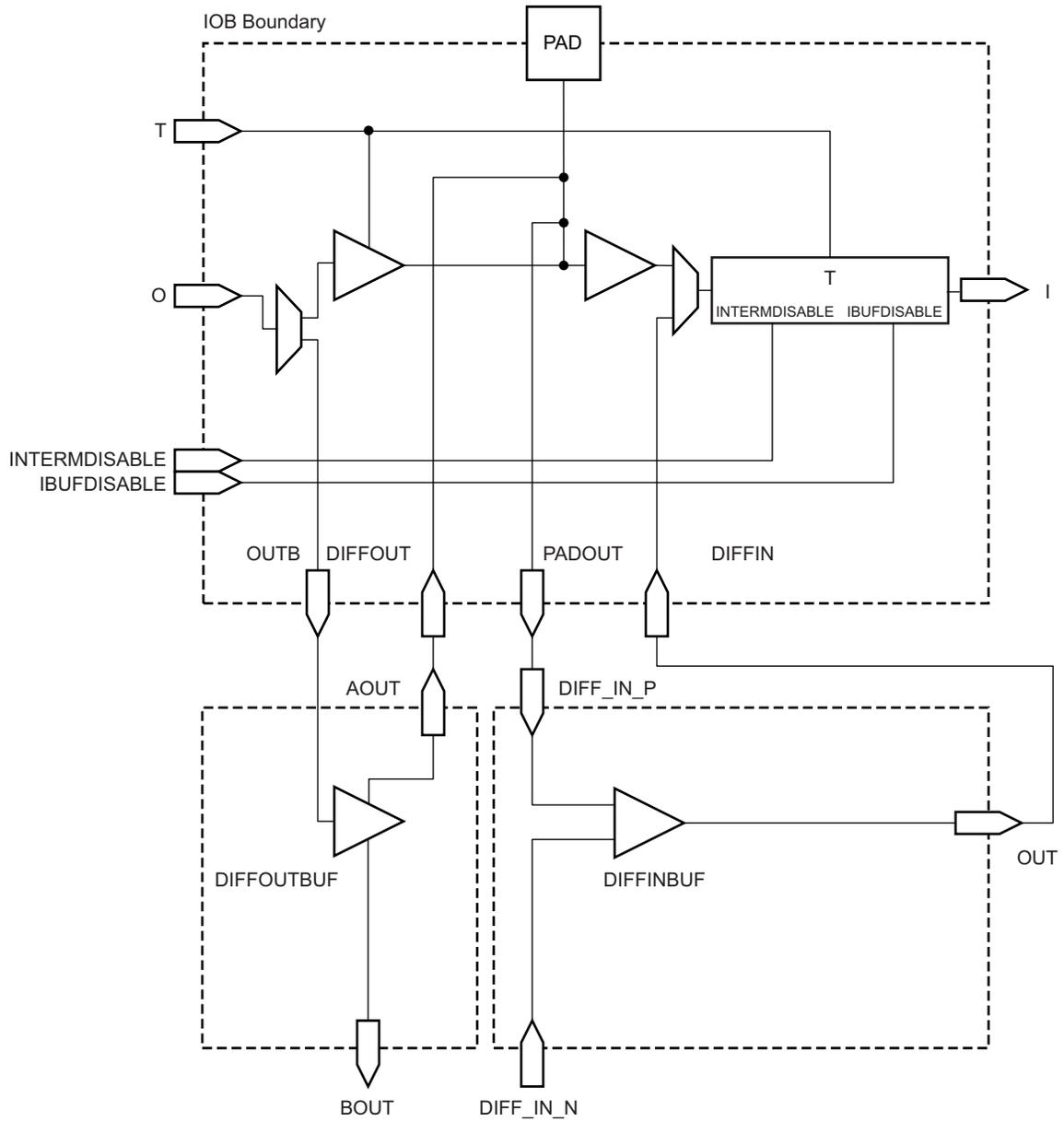
X16060-022216

Figure 1-2: Standard HP IOB Diagram



X16061-08171E

Figure 1-3: Single-Ended (Only) HR IOB Diagram



X16062-022216

Figure 1-4: Standard HR IOB Diagram

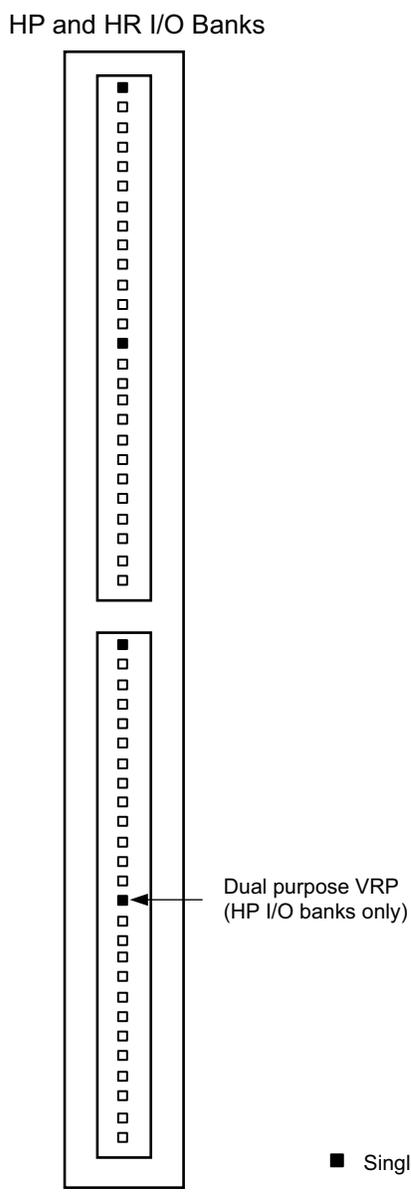


Figure 1-5: Relative Single-Ended I/O Locations within an HR or HP I/O Bank

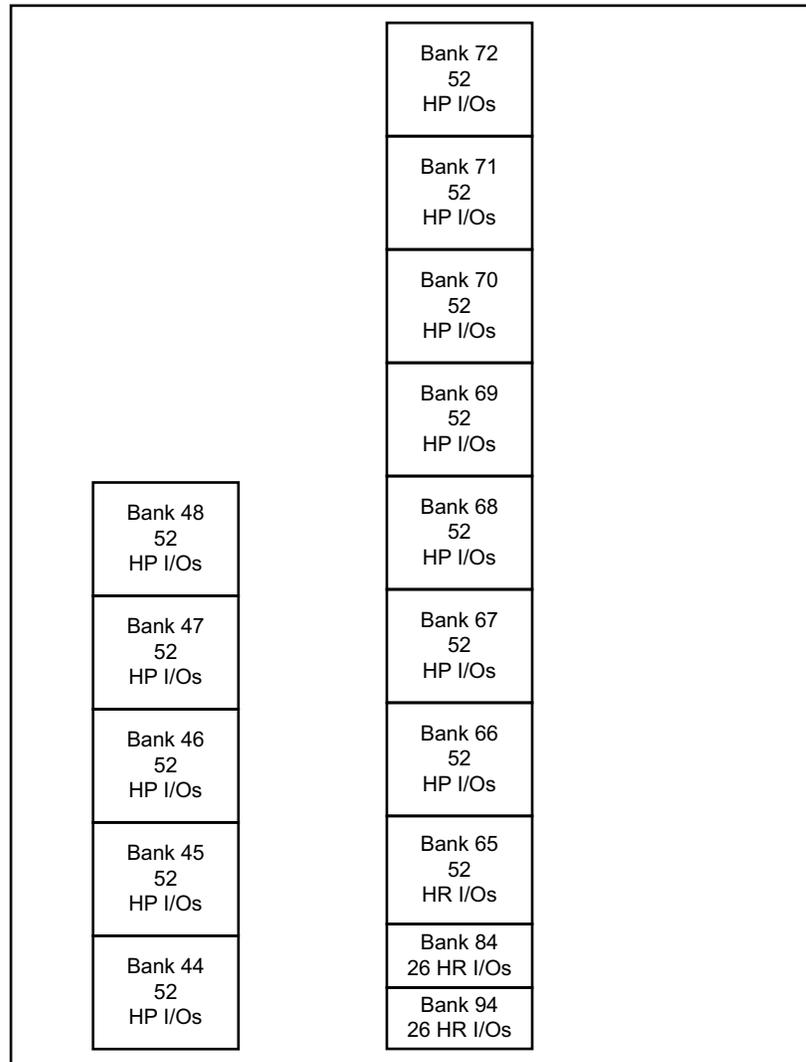
---

## SelectIO Interface General Guidelines

This section summarizes the general guidelines to be considered when designing with the SelectIO resources in UltraScale devices.

### I/O Bank Rules

Most I/O banks consist of 52 IOBs, although HR I/O mini-banks consist of 26 IOBs. The number of banks depends upon the device size and the package pinout. In the *UltraScale Architecture and Product Overview* (DS890) [Ref 3], the total number of available I/O is listed by device type. [Figure 1-6](#) is an example of a typical floorplan. The *UltraScale and UltraScale+ FPGAs Packaging and Pinouts Product Specification* (UG575) and *Zynq UltraScale+ Device Packaging and Pinouts Product Specification User Guide* (UG1075) [Ref 3] include information on the I/O banks for each device/package combination.



X16064-121018

Figure 1-6: Example I/O Banks

### Supply Voltages for the SelectIO Pins

#### $V_{CC0}$

The  $V_{CC0}$  supply is the primary power supply of the I/O circuitry. The  $V_{CC0}$  (V) columns in [Table 1-77](#) provide the  $V_{CC0}$  requirements for each of the supported I/O standards, and illustrate the  $V_{CC0}$  requirements for inputs and outputs as well as the optional internal differential termination circuit.

All  $V_{CC0}$  pins for a given HP 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 share 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  $V_{CCO}$  voltages can result in loss of functionality or damage the device.*

---

In HR I/O banks, if the I/O standard voltage requirement is  $\leq 1.8V$ , but a  $V_{CCO}$  voltage of  $\geq 2.5V$  is applied, the device automatically enters an over-voltage protection mode. Reconfiguring the device with the correct  $V_{CCO}$  voltage level restores normal operation.

### $V_{REF}$

Single-ended I/O standards with a differential input buffer require an input reference voltage ( $V_{REF}$ ). When  $V_{REF}$  is required within an I/O bank, you can use either the dedicated  $V_{REF}$  pin as a  $V_{REF}$  supply input (external) or an internally generated  $V_{REF}$  ( $INTERNAL\_VREF$  or  $V_{REF}$  scan (HP I/O banks only)). An internally generated reference voltage is enabled by using the  $INTERNAL\_VREF$  constraint. Output-only IOBs do not require a  $V_{REF}$  or  $INTERNAL\_VREF$  because only the receive portion of an IOB leverages the input reference voltage. For more information on this constraint, see [SelectIO Interface Attributes and Constraints, page 57](#).




---

**IMPORTANT:** *In banks where the input I/O standard has an input reference voltage requirement and uses an internally generated  $V_{REF}$  ( $INTERNAL\_VREF$  or  $V_{REF}$  scan), connect the dedicated  $V_{REF}$  pin to GND with a  $500\Omega$  or  $1K\Omega$  resistor.*

---

In banks where the I/O standard does not have an input reference voltage requirement, connect the dedicated  $V_{REF}$  pin to GND (with a  $500\Omega$  or  $1K\Omega$  resistor), or leave it floating.

The internal  $V_{REF}$  scan feature is available in HP I/O banks to account for process variations and system considerations.

### $V_{CCAUX}$

The global auxiliary ( $V_{CCAUX}$ ) supply rail primarily provides power to the interconnect logic of the various blocks inside the device. In the I/O banks,  $V_{CCAUX}$  is also used to power input buffer circuits for some of the I/O standards. These include some of the single-ended I/O standards at or below  $1.8V$ , and also some of the  $2.5V$  standards (HR I/O banks only). 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.

The power supply requirements, including power-on and power-off sequencing, are described in the UltraScale device data sheets [\[Ref 2\]](#).

### $V_{CCAUX\_IO}$

The auxiliary I/O ( $V_{CCAUX\_IO}$ ) voltage supply rail provides power to the I/O circuitry.  $V_{CCAUX\_IO}$  should only be powered by  $1.8V$ .

### $V_{CCINT\_IO}$

This is an internal supply for I/O banks. Connect to the  $V_{CCINT}$  voltage supply rail.

## State of I/Os During and After Configuration

UltraScale devices have pins dedicated to the configuration functions contained in I/O bank 0. There are also I/O pins in bank 65 (multi-function configuration bank) known as multi-function or multipurpose pins that can be used for configuration and converted to programmable I/O pins after configuration is complete. Additionally, during configuration of devices with multiple super logic regions (SLRs), the pins in bank 60 and bank 70 have restrictions similar to the multi-function pins. The restrictions on these banks are required even though they are not configuration banks.

During configuration, I/O drivers are 3-stated in all banks except the banks used for configuration (bank 0 and bank 65) and the aforementioned bank 60 and bank 70 in devices with multiple SLRs. During configuration (until the applications settings take over), all HP I/O banks use the default IOSTANDARD = LVCMOS18, SLEW = FAST, and DRIVE = 12 mA setting. The corresponding setting in HR I/O banks is IOSTANDARD = LVCMOS25, SLEW = FAST, and DRIVE = 12 mA. After configuration, the unconfigured I/Os have 3-stated drivers and the pads are weakly pulled-down.

In devices where bank 65 (all devices) and bank 70 (only devices with multiple SLRs) are HR I/O banks and configured with a  $V_{CCO}$  requirement  $\leq 1.8V$ , the inputs can have 0-1-0 transition to the interconnect logic during configuration if the input is tied to a 0 or floated and the configuration voltage is  $\geq 2.5V$ . For further details, see the *UltraScale Architecture Configuration User Guide* (UG570) [Ref 5].

UltraScale (not UltraScale+) devices with multiple SLRs can have the weak pull-up temporarily enabled on I/Os in the slave SLR during the configuration sequence (between power on and assertion of the INIT\_B configuration signal). In some boards, this can cause an undesired 0-1-0 transition on I/O in the slave SLR. It is recommended that any I/O pins in the slave SLR sensitive to a 0-1-0 transition during configuration be connected to I/Os in the master SLR or include external pull-downs of 1 k $\Omega$  or stronger to the pin.

---

## DCI—Only Available in the HP I/O Banks

### Introduction

As device footprints increase and system clock speeds get faster, PC board design and manufacturing becomes more difficult. With ever faster edge rates, maintaining signal integrity becomes a critical issue. PC board traces must be properly terminated to avoid reflections or ringing.

To terminate a trace, resistors are traditionally added to make the output and/or input match the impedance of the receiver or driver to the impedance of the trace. However, due to increased device I/Os, adding resistors close to the device pins increases the board area and component count, and can in some cases be physically impossible. To address these issues and to achieve better signal integrity, AMD developed the digitally controlled impedance (DCI) technology.

Depending on the I/O standard, DCI can either control the output impedance of a driver, or add a parallel termination present at the receiver, with the goal of accurately matching the characteristic impedance of the transmission line. DCI actively adjusts these impedances inside the I/O to calibrate to an external precision reference resistor placed on the VRP pin. This compensates for changes in I/O impedance due to process variation. It also continuously adjusts the impedances to compensate for variations of temperature and supply voltage fluctuations. Many designs require the use of multiple DCI reference VRP pins. In these scenarios, a unique reference resistor is required for each VRP pin.



---

**IMPORTANT:** For all DCI I/O standards, the external reference resistor ( $R_{VRP}$ ) should be  $240\Omega$ .

---

For the I/O standards with controlled parallel termination, DCI provides the parallel termination for receivers. This eliminates the need for termination resistors on the board, reduces board routing difficulties and component count, and improves signal integrity by eliminating stub reflection. Stub reflection occurs when termination resistors are located too far from the end of the transmission line. With DCI, the termination resistors are as close as possible to the output driver or the input buffer, thus, eliminating stub reflections. The exact value of the termination resistors is determined by the ODT attribute for controlled parallel termination. The exact driver termination value is determined by the OUTPUT\_IMPEDANCE attribute for the controlled impedance driver. DCI is only available in HP I/O banks. DCI is not available in HR I/O banks.

DCI uses one multipurpose reference VRP pin in each I/O bank to control the impedance of the driver or the parallel-termination value for all of the I/Os of that bank.



---

**IMPORTANT:** When using DCI standards, the VRP pin must be terminated to GND by a reference resistor. The value of the resistor should be  $240\Omega$ .

---

To implement DCI in a design:

1. Assign one of the DCI I/O standards in an HP I/O bank (see [Table 1-3](#)).
2. Connect the VRP multi-function pin to a precision resistor (240Ω) tied to GND.
3. Set the desired termination value using the ODT attribute for all applicable I/Os with controlled parallel terminations. Set the termination value using the OUTPUT\_IMPEDANCE attribute for all applicable I/Os with a controlled impedance driver.

If several I/O banks in the same I/O bank column are using DCI, the internal VRP node can be cascaded so that only one VRP pin for all of the I/O banks in the entire I/O column is required to be connected to a precision resistor. This option is called DCI cascading as is detailed in [DCI Cascading](#). This section also describes how to determine if I/O banks share the same I/O bank column. If DCI I/O standards are not used in the bank, the VRP pin is available as a standard I/O pin. The *UltraScale and UltraScale+ FPGAs Packaging and Pinouts Product Specification* (UG575) [\[Ref 3\]](#) gives detailed pin descriptions.

DCI adjusts the impedance of the I/O by selectively turning resistors in the I/Os on or off. The adjustment starts during the device start-up sequence. By default, the DONE pin does not transition High until the first part of the impedance adjustment process is completed.

The DCI calibration can be reset by instantiating the DCIRESET primitive. Toggling the RST input to the DCIRESET primitive while the device is operating resets the DCI state machine and restarts the calibration process. All I/Os using DCI are unavailable until the LOCKED output from the DCIRESET block is asserted. This functionality is useful in applications where the temperature and/or supply voltage changes significantly from device power-up to the nominal operating condition.

For controlled impedance output drivers, the exact value of the driver terminations is determined by the OUTPUT\_IMPEDANCE attribute. For the I/O standards that support parallel termination, DCI creates a Thevenin equivalent, or split-termination resistance to the  $V_{CCO}/2$  voltage level, or a single-termination resistance to the  $V_{CCO}$  voltage level. The value of split-termination resistors are determined by the ODT attribute. For POD and HSUL standards, DCI supports single termination to  $V_{CCO}$ . The value of the termination resistance is determined by the ODT attribute.

## Match\_cycle Configuration Option

Match\_cycle is a configuration option that can halt the start-up sequence at the end of the device configuration sequence until the DCI logic has performed the first match (calibration) to the external reference resistor. This option is also sometimes referred to as *DCI match*.

## DCIUpdateMode Configuration Option

DCIUpdateMode is a configuration option that can override control of how often the DCI circuit updates the impedance matching to the VRP reference resistor. This option defaults to ASREQUIRED in the AMD implementation tools. The settings for the DCIUpdateMode configuration option are:

- **ASREQUIRED:** Initial impedance calibration is made at device initialization, and dynamic impedance adjustments are made as needed throughout device operation (default).
- **QUIET:** Impedance calibration is done once at device initialization, or each time the RST pin is asserted on the DCIRESET primitive for designs that include this primitive.



---

**RECOMMENDED:** *It is strongly recommended that the DCIUpdateMode option be kept with the default value of ASREQUIRED so that the DCI circuitry is allowed to operate normally.*

---

## DCIRESET Primitive

DCIRESET is an AMD design primitive that provides the capability to perform a reset of the DCI controller state machine during normal operation of the design. This primitive is required in a design when DCIUpdateMode is set to QUIET (see [DCIUpdateMode Configuration Option](#)) or for the case outlined in [Special DCI Requirements in Some Banks](#). See the *UltraScale Architecture Libraries Guide* (UG974) [\[Ref 6\]](#) for more details on the DCIRESET primitive.

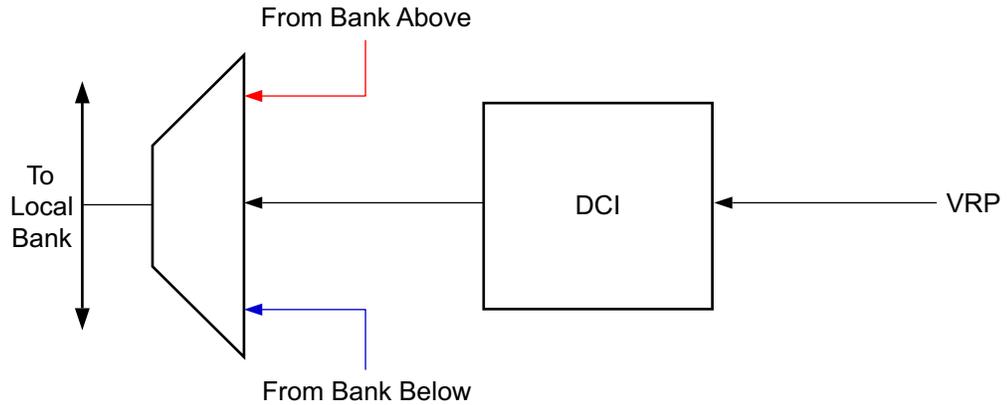
## Special DCI Requirements in Some Banks

When any of the multi-function pins in I/O bank 65 (or pins in bank 60 or bank 70 in devices with multiple SLRs) are assigned as DCI I/O standards (in HP I/O bank devices), you should also include and use the DCIRESET primitive in your design. In that case, the design should pulse the RST input of DCIRESET and then wait for the LOCKED signal to be asserted prior to using any of these pins' inputs or outputs with DCI standards. This is required because these I/O pins ignore the initial DCI calibration that happens during the normal device initialization.

As a result, if the DCIRESET primitive had not been used and DCIUpdateMode was set to ASREQUIRED, after those pins become normal I/O pins, there would be an indeterministic delay between the end of configuration and when the DCI calibration algorithm updated those pins' DCI settings. If DCIRESET was not used and DCIUpdateMode was set to QUIET, these pins would never have their DCI values set. Including and using the DCIRESET primitive in the design allows these pins to have DCI I/O standards and to perform without issue.

## DCI Cascading

The HP I/O banks using DCI I/O standards have the option of deriving the DCI impedance values from another HP I/O bank. As shown in Figure 1-7, a digital control bus is internally distributed throughout the bank to control the impedance of each I/O.

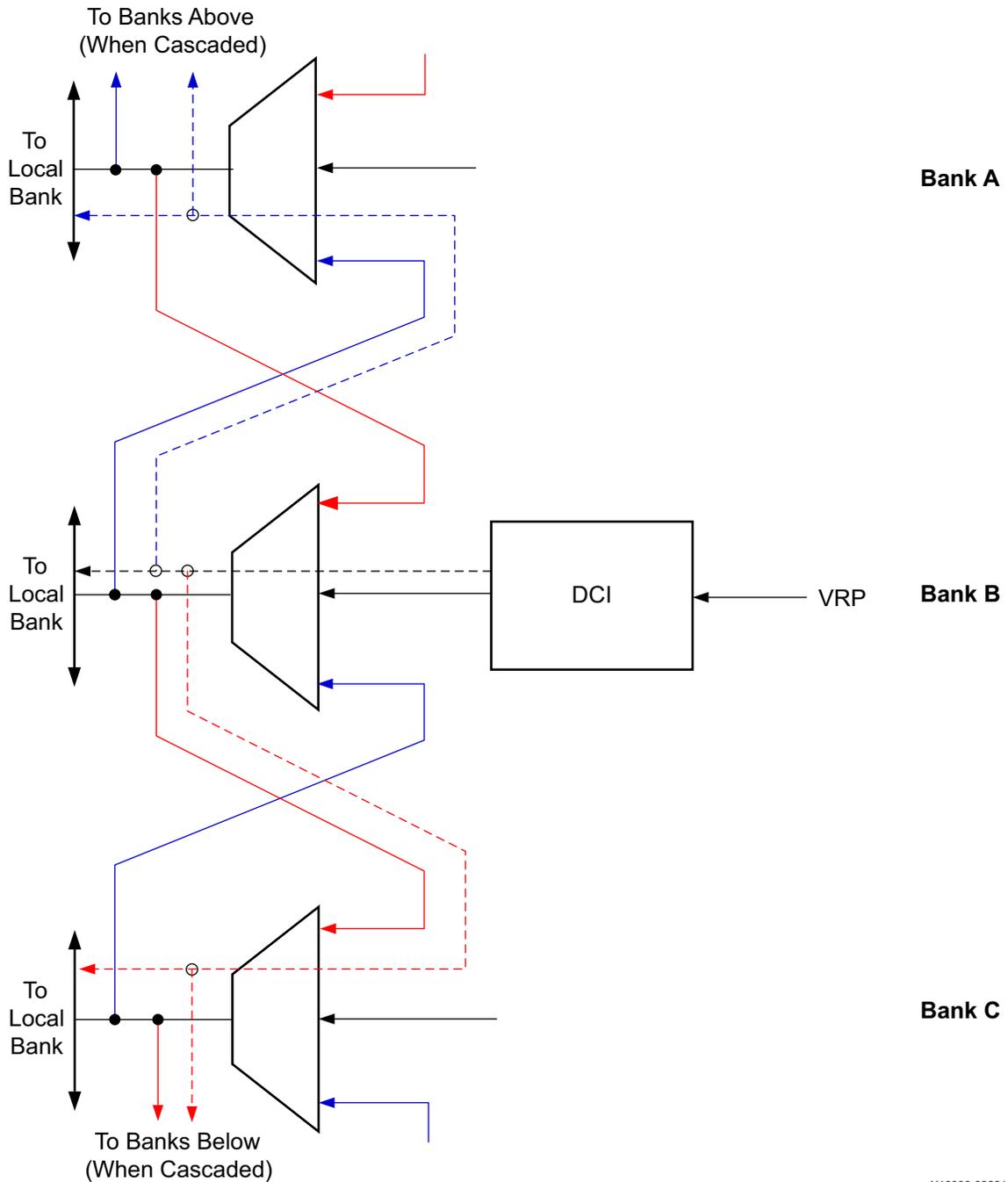


X16065-022216

Figure 1-7: DCI Use within a Bank

With DCI cascading, one I/O bank (the master bank) must have its VRP pin connected to an external reference resistor. Other I/O banks in the same HP I/O bank column (slave banks) can use DCI standards with the same impedance as the master bank, without connecting the VRP pin on these slave banks to an external resistor. DCI impedance control in cascaded banks is received from the I/O master bank.

Figure 1-8 shows DCI cascading support over multiple I/O banks. Bank B is the master I/O bank, and Banks A and C are considered slave I/O banks.



X16066-022216

Figure 1-8: DCI Cascading Supported over Multiple I/O Banks

The guidelines when using DCI cascading are as follows:

- DCI cascading is only available through a column of HP I/O banks
- The master and slave SelectIO technology banks must all reside on the same HP I/O column on the device and can span the entire column unless there is an interposer boundary.
- DCI cascading cannot pass through the interposer boundaries of the larger UltraScale devices with stacked silicon interconnect (SSI) technology.
- Master and slave I/O banks must have the same  $V_{CCO}$  and  $V_{REF}$  (if applicable) voltage.
- I/O banks in the same HP I/O column that are not using DCI (pass-through banks) do not have to comply with the  $V_{CCO}$  and  $V_{REF}$  voltage rules for combining DCI settings.
- DCI I/O banking compatibility rules must be satisfied across all master and slave banks.
- To locate I/O banks that reside in the same I/O column, see the figures in the *Die Level Bank Numbering Overview* section of the *UltraScale and UltraScale+ FPGAs Packaging and Pinouts Product Specification* (UG575) [Ref 3].
- For specific information on implementing DCI cascading in a design, see [DCI\\_CASCADE Constraint, page 57](#).



---

**RECOMMENDED:** *Unused banks must be powered up because leaving the  $V_{CCO}$  pins of unused I/O banks floating reduces the level of ESD protection on these pins and the I/O pins in the bank. If the bank is unpowered, DCI can still be cascaded through the unpowered bank.*

---

When DCI cascading is used, source and on-die input terminations have a bigger variation as compared to DCI used on a per bank basis without cascade. See the product's data sheet for quantity [Ref 2].

## Controlled Impedance Driver (Source Termination)

To optimize signal integrity for high-speed or high-performance applications, extra measures are required to match the output impedance of drivers to the impedance of the transmission lines and receivers. Optimally, drivers must have an output impedance matching the characteristic impedance of the driven line, otherwise reflections can occur due to discontinuities. To solve this issue, designers sometimes use external-source series-termination resistors placed close to the pins of high-strength, low-impedance drivers. The resistance values are chosen such that the sum of the output impedance of the driver plus the resistance of the source series-termination resistor roughly equals the impedance of the transmission line.

DCI can provide controlled impedance output drivers to eliminate reflections without requiring the use of an external source-termination resistor. The impedance is derived from the external reference resistor.

Figure 1-9 illustrates a controlled impedance driver inside a device.

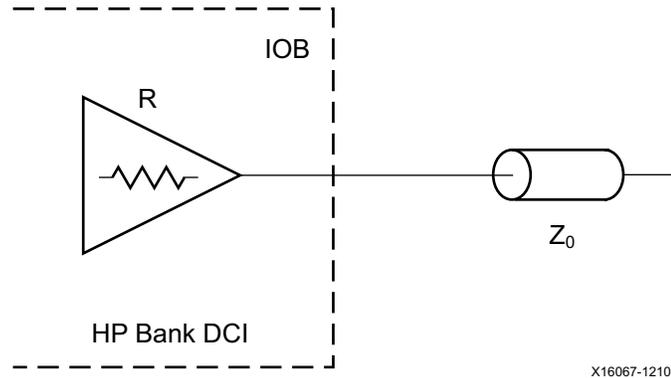


Figure 1-9: Controlled Impedance Driver

The DCI input standards supporting the controlled impedance driver are shown in Table 1-2.

Table 1-2: All DCI I/O Standards Supporting Controlled Impedance Driver

HSTL_I_DCI	DIFF_HSTL_I_DCI	LVDCI_18	HSUL_12_DCI	DIFF_HSUL_12_DCI	SSTL18_I_DCI	DIFF_SSTL18_I_DCI
HSTL_I_DCI_18	DIFF_HSTL_I_DCI_18	LVDCI_15	POD12_DCI	DIFF_POD12_DCI	SSTL15_DCI	DIFF_SSTL15_DCI
HSTL_I_DCI_12	DIFF_HSTL_I_DCI_12	HSLVDCI_18	POD10_DCI	DIFF_POD10_DCI	SSTL135_DCI	DIFF_SSTL135_DCI
		HSLVDCI_15			SSTL12_DCI	DIFF_SSTL12_DCI

## Split-Termination DCI (Thevenin Equivalent Termination to $V_{CC0}/2$ )

Some I/O standards (HSTL and SSTL) require an input termination resistance ( $R$ ) to a  $V_{TT}$  voltage of  $V_{CC0}/2$  (see [Figure 1-10](#)).

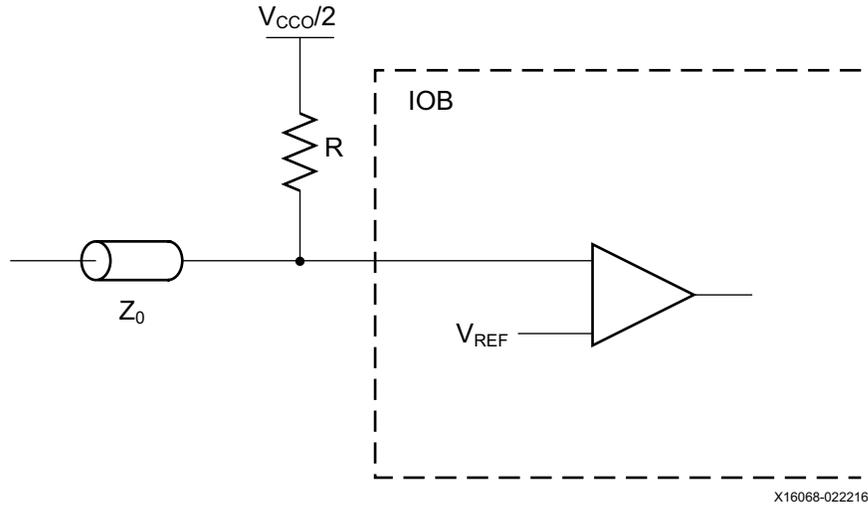


Figure 1-10: Input Termination to  $V_{CC0}/2$  without DCI (where  $R = Z_0$ )

Split-termination DCI creates a Thevenin equivalent circuit using two resistors of twice the resistance value ( $2R$ ). One terminates to  $V_{CC0}$ , the other to GND. Split-termination DCI provides an equivalent termination to  $V_{CC0}/2$  using this method. The  $2R$  termination resistance is set by programming the ODT attribute. The resistors to  $V_{CC0}$  and GND are equal to twice the value set by ODT. For example, to achieve the Thevenin equivalent parallel-termination circuit of approximately  $50\Omega$  to  $V_{CC0}/2$ , a  $240\Omega$  external precision resistor is required at the VRP pin and ODT is set to  $RTT_{48}$ . Possible values for ODT for split-termination DCI are  $RTT_{40}$ ,  $RTT_{48}$ , or  $RTT_{60}$ .

The DCI input standards supporting split termination are shown in [Table 1-3](#).

Table 1-3: All DCI I/O Standards Supporting Split-Termination DCI

HSTL_I_DCI	DIFF_HSTL_I_DCI	SSTL18_I_DCI	DIFF_SSTL18_I_DCI
HSTL_I_DCI_18	DIFF_HSTL_I_DCI_18	SSTL15_DCI	DIFF_SSTL15_DCI
HSTL_I_DCI_12 (UltraScale devices only)	DIFF_HSTL_I_DCI_12 (UltraScale devices only)	SSTL135_DCI	DIFF_SSTL135_DCI
		SSTL12_DCI	DIFF_SSTL12_DCI

Figure 1-11 illustrates split-termination DCI.

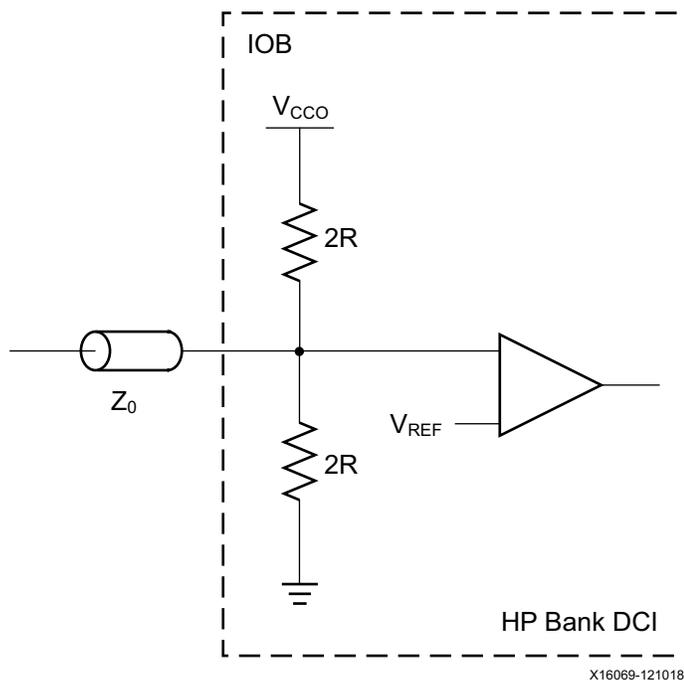


Figure 1-11: Input Termination to  $V_{CC0}/2$  Using Split-Termination DCI (where  $R = Z_0$ )

## Single-Termination DCI

Some I/O standards (POD10, POD12, HSUL\_12, and DIFF\_HSUL\_12) require an input termination resistance (R) to a  $V_{TT}$  voltage of  $V_{CC0}$  (see [Figure 1-12](#)).

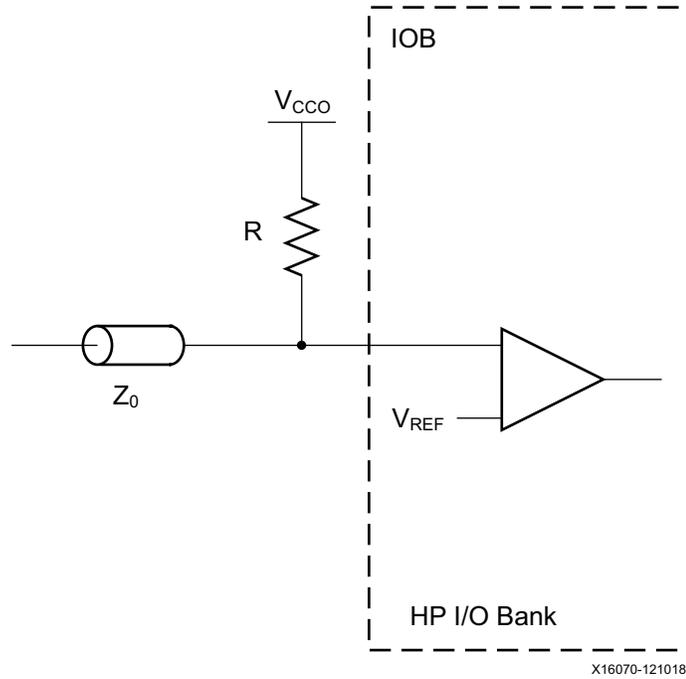


Figure 1-12: Input Termination to  $V_{CC0}$  without DCI (where  $R = Z_0$ )

The DCI input standards supporting single termination are shown in [Table 1-4](#).

Table 1-4: All DCI I/O Standards Supporting Single-Termination DCI

POD12_DCI	DIFF_POD12_DCI	HSUL_12_DCI
POD10_DCI	DIFF_POD10_DCI	DIFF_HSUL_12_DCI

Single-termination DCI creates a termination to  $V_{CC0}$  internally as shown in Figure 1-13. Value of the termination resistor is determined by the ODT attribute. Possible values for ODT:

- POD standards only: RTT\_40, RTT\_48, and RTT\_60
- HSUL\_12\_DCI and DIFF\_HSUL\_12\_DCI only: RTT\_120 and RTT\_240
- RTT\_NONE

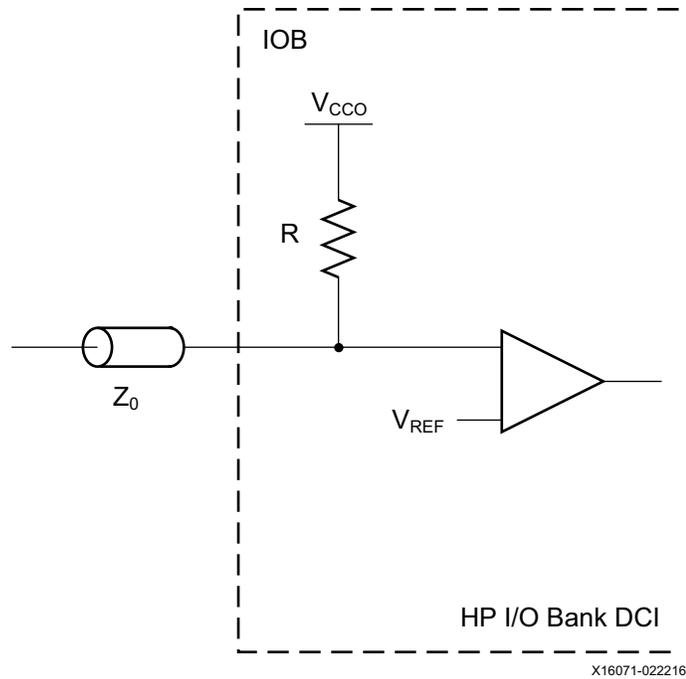


Figure 1-13: Input Termination to  $V_{CC0}$  using Single-Termination DCI (where  $R = Z_0$ )

For example, to achieve single termination of approximately  $50\Omega$  to  $V_{CC0}$  for the POD12\_DCI standard, a  $240\Omega$  external precision resistor is required at the VRP pin, ODT must be set to the value RTT\_48.

## VRP External Resistance Design Migration Guidelines

Previous AMD Adaptive Computing FPGA families featuring DCI used a slightly different circuit for calibrating the controlled impedance driver and split-termination impedance from the external reference resistors placed on the VRN and VRP pins. In AMD 7 series FPGAs, DCI calibrated each leg of the split-termination circuit to be directly equal to the external resistor values. For example, a 7 series device with a target parallel termination of  $50\Omega$  to  $V_{CC0}/2$  requires  $100\Omega$  external resistors on the VRN and VRP pins.

In UltraScale devices, irrespective of the DCI termination value requirement, the external resistor on the VRP pin is required to be  $240\Omega$ . Instead of two resistors, only one resistor is

required at an UltraScale device VRP pin. The exact value of the split-termination or single-termination resistors are determined by the user-controllable ODT attribute.

Possible ODT values for split-termination DCI standards (HSTL and SSTL) are RTT\_40, RTT\_48, or RTT\_60.




---

**IMPORTANT:** *The ODT value represents the desired Thevenin resistance to  $V_{CC0}/2$  for split-termination DCI standards.*

---

Possible ODT values for single-termination POD standards are RTT\_40, RTT\_48, or RTT\_60. Possible ODT values for single-termination HSUL standards are RTT\_120, RTT\_240, or RTT\_NONE.




---

**IMPORTANT:** *The ODT value represents the desired resistance to  $V_{CC0}$  for single-termination DCI standards.*

---

The termination value for the controlled impedance driver is determined by the DCI state machine when a DCI standard with a controlled impedance driver is chosen, using OUTPUT\_IMPEDANCE attribute values. Possible values for OUTPUT\_IMPEDANCE attributes are RDRV\_40\_40, RDRV\_48\_48, RDRV\_60\_60, and RDRV\_NONE\_NONE.

## T\_DCI Design Migration Guidelines

The AMD 7 series architecture supported T\_DCI standards for bidirectional I/O configurations with 3-state support for internal input split-termination. Those T\_DCI standards are not supported in UltraScale devices. However, many of the UltraScale architecture DCI standards are capable of supporting similar bidirectional configurations. [Table 1-5](#) lists the T\_DCI standards that are transparently ported or migrated to an equivalent UltraScale architecture standard when designing with the Vivado™ Design Suite.

*Table 1-5: T\_DCI I/O Standards for Migration between AMD Device Architectures*

7 Series Architecture I/O Standard	UltraScale Architecture Equivalent I/O Standard
DIFF_SSTL15_T_DCI	DIFF_SSTL15_DCI
DIFF_SSTL135_T_DCI	DIFF_SSTL135_DCI
DIFF_SSTL12_T_DCI	DIFF_SSTL12_DCI
SSTL15_T_DCI	SSTL15_DCI
SSTL135_T_DCI	SSTL135_DCI
SSTL12_T_DCI	SSTL12_DCI

## DCI I/O Standard Support

DCI supports the standards shown in [Table 1-6](#).

**Table 1-6: All Supported DCI I/O Standards**

LVDCI_18	HSTL_I_DCI	DIFF_HSTL_I_DCI	SSTL18_I_DCI	DIFF_SSTL18_I_DCI
LVDCI_15	HSTL_I_DCI_18	DIFF_HSTL_I_DCI_18	SSTL15_DCI	DIFF_SSTL15_DCI
HSLVDCI_18	HSTL_I_DCI_12 (UltraScale devices only)	DIFF_HSTL_I_DCI_12 (UltraScale devices only)	SSTL135_DCI	DIFF_SSTL135_DCI
HSLVDCI_15			SSTL12_DCI	DIFF_SSTL12_DCI
			HSUL_12_DCI	DIFF_HSUL_12_DCI
			POD12_DCI	DIFF_POD12_DCI
			POD10_DCI	DIFF_POD10_DCI

To correctly use DCI:

1.  $V_{CCO}$  pins must be connected to the appropriate  $V_{CCO}$  voltage based on the I/O standards in that I/O bank.
2. Correct DCI I/O buffers must be used in the Vivado Design Suite either by using I/O standard attributes or instantiations in the hardware description language (HDL) code. The Vivado Design Suite automatically restricts the VRP pin from being used as an I/O when a VRP needs to be used as a reference.
3. DCI standards require connecting an external reference resistor to the multipurpose VRP pin. When this is required, that multipurpose pin cannot be used as a general-purpose I/O in the I/O bank using DCI or in the master I/O bank when cascading DCI. See the pinout tables for the specific pin locations. The VRP pin must be pulled to GND by a reference resistor. An exception to this requirement comes when cascading DCI in slave I/O banks because the VRP pin can be used as general-purpose I/O.
4. The value of the external reference resistor is fixed at  $240\Omega$ , terminated to GND.
5. Follow the DCI I/O banking rules:
  - a.  $V_{REF}$  must be compatible for all of the inputs in the same I/O bank or in a group of I/O banks when using DCI cascade.
  - b.  $V_{CCO}$  must be compatible for all of the inputs and outputs in the same I/O bank.
  - c. Impedances are no longer constrained by  $R_{VRP}$  ( $240\Omega$ ). The DCI state machine calculates the appropriate scaling for controlled impedance drivers, and split and single-termination configurations using OUTPUT\_IMPEDANCE and ODT attribute values.

## Uncalibrated Input Termination in I/O Banks

The HR I/O and HP I/O banks have an optional uncalibrated input on-chip split-termination feature for HSTL and SSTL standards and a single-termination feature for POD and HSUL standards that are similar to the DCI feature. This option creates a Thevenin equivalent circuit using two internal resistors of twice the target resistance value ( $2R$  where  $R = Z_0$ ) for HSTL and SSTL standards. One resistor terminates to  $V_{CC0}$  and the other to GND, providing a Thevenin equivalent termination circuit of half the resistor value to the mid-point  $V_{CC0}/2$  for HSTL and SSTL standards. A single resistor terminates to  $V_{CC0}$  for POD and HSUL standards.

The termination is present constantly on inputs, and on bidirectional pins whenever the output buffer is 3-stated except when DCITERMDISABLE (in HP I/O banks) or INTERMDISABLE (in HR I/O banks) are asserted. However, an important difference between this uncalibrated option and DCI is that instead of calibrating to an external reference resistor on the VRP pin when using DCI, the uncalibrated input termination feature invokes internal resistors determined by the ODT attribute that have no calibration routine to compensate for temperature, process, or voltage variations.

- Possible ODT values for split-termination standards (HSTL and SSTL) are RTT\_40, RTT\_48, RTT\_60, or RTT\_NONE.
- Possible values for ODT for single-termination POD standards are RTT\_40, RTT\_48, RTT\_60, or RTT\_NONE.
- Possible values for ODT for single-termination HSUL standards are RTT\_120, RTT\_240, or RTT\_NONE.

The main difference in how DCI or uncalibrated termination is invoked in a design is whether or not a DCI I/O standard is chosen. In both DCI and uncalibrated I/O standards, the values of the termination resistors are determined by the ODT attribute.

Table 1-7 shows a list of I/O standards that support the uncalibrated termination in both the HR and HP I/O banks.

**Table 1-7: I/O Standards that Support Uncalibrated Termination**

HSTL_I	DIFF_HSTL_I	SSTL18_I	DIFF_SSTL18_I	POD12	DIFF_POD12
HSTL_II	DIFF_HSTL_II	SSTL18_II	DIFF_SSTL18_II	POD10	DIFF_POD10
HSTL_I_18	DIFF_HSTL_I_18	SSTL15_R	DIFF_SSTL15_R	HSUL_12	DIFF_HSUL_12
HSTL_II_18	DIFF_HSTL_II_18	SSTL15	DIFF_SSTL15		
		SSTL135_R	DIFF_SSTL135_R		
		SSTL135	DIFF_SSTL135		
		SSTL12	DIFF_SSTL12		

## Uncalibrated Source Termination in HP I/O banks

HP I/O banks have an optional uncalibrated source termination feature for SSTL, HSTL, POD, and HSUL standards that is similar to DCI. This feature provides an option of 40Ω, 48Ω, or a 60Ω driver for the supported standards to match the characteristic impedance of the driven line.

An important difference between this uncalibrated option and DCI is that instead of calibrating to an external reference resistor on the VRP pin when using DCI, the uncalibrated source termination feature invokes internal resistors determined by the OUTPUT\_IMPEDANCE attribute where no calibration routine is available to compensate for temperature, process, or voltage variations.




---

**IMPORTANT:** *This feature is only available in HP I/O banks.*

---

The allowed values for OUTPUT\_IMPEDANCE are RDRV\_40\_40, RDRV\_48\_48, or RDRV\_60\_60.

The main difference in how DCI or uncalibrated termination is invoked in a design is determined when choosing to use either a DCI I/O standard or an uncalibrated one. In both DCI and uncalibrated I/O standards, source termination value is determined by the attribute OUTPUT\_IMPEDANCE.

Table 1-8 shows a list of I/O standards that support uncalibrated source termination in HP I/O banks.

**Table 1-8: I/O Standards that Support Uncalibrated Source Termination in HP I/O Banks**

HSTL_I	DIFF_HSTL_I	SSTL18_I	DIFF_SSTL18_I	POD12	DIFF_POD12
HSTL_I_18	DIFF_HSTL_I_18	SSTL15	DIFF_SSTL15	POD10	DIFF_POD10
HSTL_I_12	DIFF_HSTL_I_12	SSTL135	DIFF_SSTL135	HSUL_12	DIFF_HSUL_12
		SSTL12	DIFF_SSTL12		

## Receiver Offset Control in HP I/O Banks

In HP I/O banks, for a subset of I/O standards, the UltraScale architecture provides the option of canceling the inherent offset of the input buffers that occurs due to process variations (up to  $\pm 35$  mV). This feature can be accessed through IBUFE3, IBUFDSE3, IOBUFE3, and IOBUFDSE3 primitives as shown in Figure 1-14 and Figure 1-15. Offset calibration requires building control logic into your interconnect logic design.

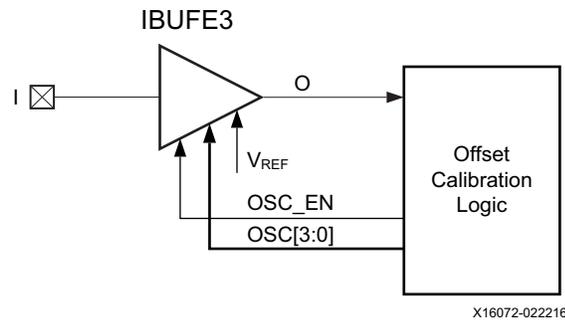


Figure 1-14: Offset Calibration Connections for Single-Ended I/O Standards

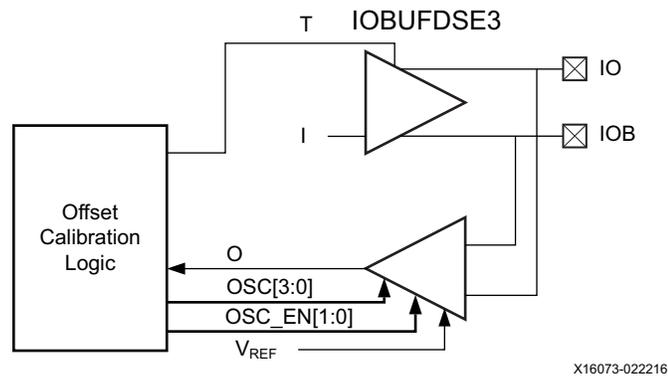


Figure 1-15: Offset Calibration Connections for Differential I/O Standards

1. The offset cancellation feature is activated for the supported I/O standards when:
  - a. Offset control attribute OFFSET\_CNTRL is set to FABRIC.
  - b. OSC\_EN port is set to 1'b1 (single-ended I/O standards) or 2'b11 (for differential I/O standards).



**IMPORTANT:** The value of 2'b10 or 2'b01 is illegal when using OSC\_EN for differential I/O standards.

2. After the offset cancellation feature is activated, the input to the buffer is pulled-up to  $V_{REF}$  (in differential I/Os, both legs are pulled-up to  $V_{REF}$ ). Based on the inherent offset of the buffer, the output (O) is either a logic 1 or 0. A logic 1 suggests a positive offset. A logic 0 suggests a negative offset. In simulation, this hardware behavior can be mimicked by setting the simulation-only attribute SIM\_INPUT\_BUFFER\_OFFSET to a

negative or positive value from  $-50$  mV to  $+50$  mV. This simulation-only attribute is supported with IBUFE3, IBUFDSE3, IOBUFE3, and IOBUFDSE3 primitives.

- Based on the value of O, the FABRIC calibration logic should sweep OSC[3:0] in the positive or negative direction until O is seen to flip. The value where O flips is the required offset to cancel the inherent offset of the buffer. Table 1-9 shows the approximate amount of offset cancellation provided by each setting of OSC.

Table 1-9: Approximate Amount of Offset Cancellation for Each Setting of OSC

OSC[3:0]	Estimated Offset Cancellation (mV)	OSC[3:0]	Estimated Offset Cancellation (mV)
0000	0	1000	0
0001	-5	1001	5
0010	-10	1010	10
0011	-15	1011	15
0100	-20	1100	20
0101	-25	1101	25
0110	-30	1110	30
0111	-35	1111	35

For example, if the buffer input offset is 15 mV, setting OSC[3:0] = 1011 cancels the offset. If the buffer input offset is  $-10$  mV, setting OSC[3:0] = 0010 cancels the offset.

- If O does not flip, even at the maximum possible offset ( $-35$ mV or  $35$ mV), OSC should be set to the maximum  $-35$ mV (0111) if O stays at a logic 1 throughout, or  $+35$ mV (1111) if O stays at a logic 0 throughout and continue to step 5.
- After the required offset is determined, OSC\_EN should be turned off by setting it to  $1'b0$  (single ended I/O standards) or  $2'b00$  (differential I/O standards) and normal operation can resume.



**RECOMMENDED:** Offset calibration should not be attempted on inputs with external bias or termination.



**IMPORTANT:** OSC[3:0] is a shared bus among all the I/Os within a half bank (26 consecutive I/Os in the top half or bottom half of a bank).

The I/O standards that support receiver offset control are shown in Table 1-10.

Table 1-10: I/O Standards Supporting Receiver Offset Control

POD12	DIFF_POD12
POD12_DCI	DIFF_POD12_DCI

## Receiver $V_{REF}$ Scan in HP I/O Banks

An optional  $V_{REF}$  scan feature in HP 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 HPIO\_VREF primitive as shown in Figure 1-16.  $V_{REF}$  scan requires building control logic into your interconnect logic design.

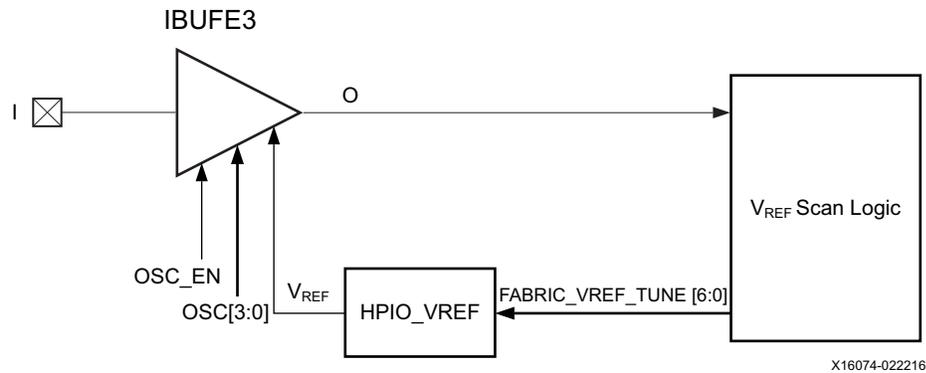


Figure 1-16: Connection from the Interconnect Logic to Access the  $V_{REF}$  Scan Feature

Internal  $V_{REF}$  tuned using the  $V_{REF}$  scan feature controls the  $V_{REF}$  of 13 consecutive I/Os (1 byte group) within a bank as shown in Figure 1-17. There are four byte groups within a bank. Four different variations of a given  $V_{REF}$  are possible within a bank (for four byte groups in each bank). However, to use this feature, the central  $V_{REF}$  of the bank needs to be set using the INTERNAL\_VREF attribute (See Internal  $V_{REF}$ ). Inputs with I/O standards of different  $V_{REF}$  specifications cannot be placed within the same bank. Tuned  $V_{REF}$  connection ( $V_{REF}$  output of HPIO\_VREF primitive) cannot traverse byte group boundaries.

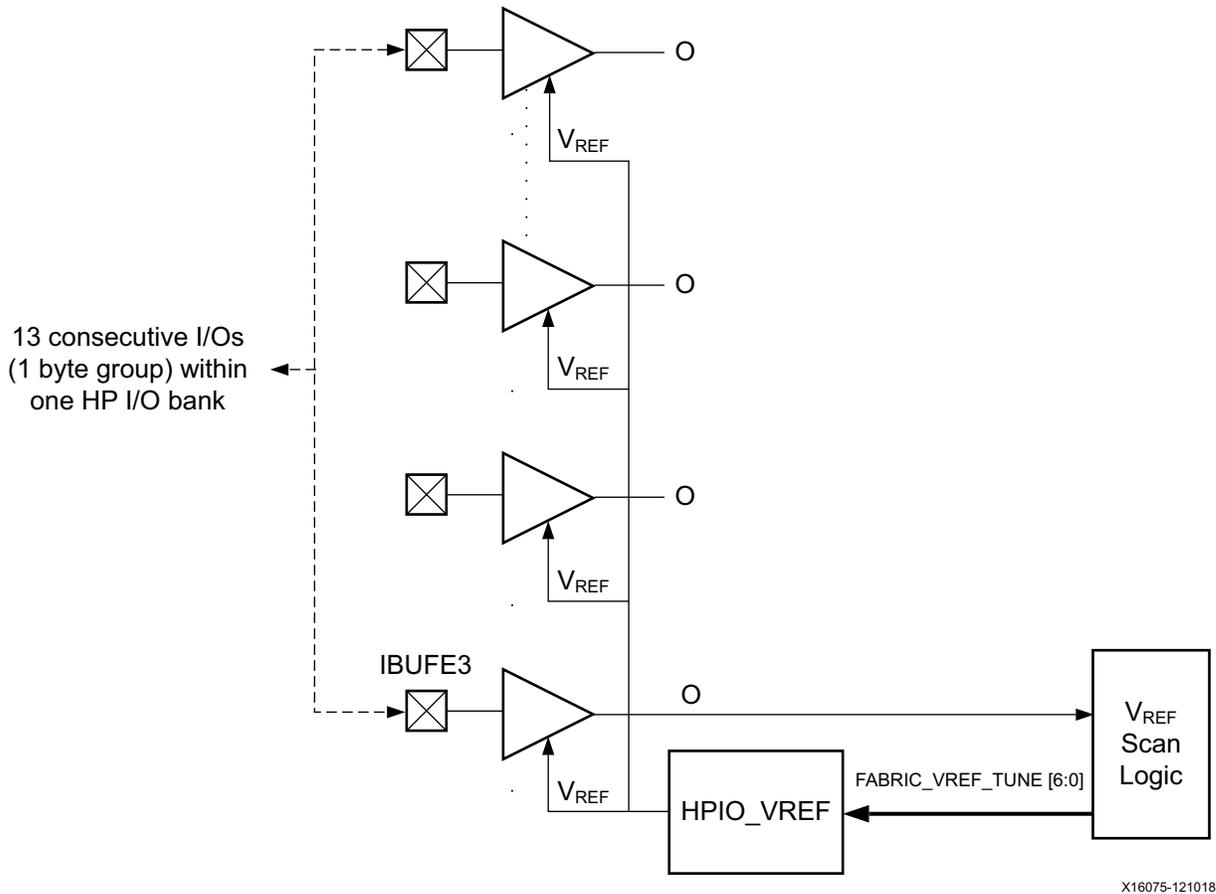


Figure 1-17:  $V_{REF}$  Scan Connection per Byte Group within a Bank

Internal  $V_{REF}$  (INTERNAL\_VREF and  $V_{REF}$  scan) cannot be combined with external  $V_{REF}$  usage within a bank.

VREF\_CNTR is used with the HPIO\_VREF UNISIM primitive to set the  $V_{REF}$  scan-range based on the I/O standard.

The valid values for the VREF\_CNTR attribute are described in this section.

- FABRIC\_RANGE1 (POD standards)
- FABRIC\_RANGE2 (other applicable standards)

FABRIC\_RANGE1 is used with the POD standards and the FABRIC\_RANGE2 is used with the other applicable standards when the receiver  $V_{REF}$  scan feature is invoked. The FABRIC\_VREF\_TUNE[6:0] port is used to tune the  $V_{REF}$  from the interconnect logic. The approximate value of  $V_{REF}$  for various values in reference to FABRIC\_VREF\_TUNE and VREF\_CNTR is shown in [Table 1-11](#).

Table 1-11: Approximate  $V_{REF}$  Value as a Result of Using the  $V_{REF}$  Scan Function

FABRIC_TUNE_VREF[6:0]	$V_{REF}$ (% of $V_{CC0}$ )	
	VREF_CNTR = FABRIC_RANGE1	VREF_CNTR = FABRIC_RANGE2
000 0001	58.00%	43.00%
000 0010	58.50%	43.50%
000 0011	59.00%	44.00%
000 0100	59.50%	44.50%
000 0101	60.00%	45.00%
000 0110	60.50%	45.50%
000 0111	61.00%	46.00%
000 1000	61.50%	46.50%
000 1001	62.00%	47.00%
000 1010	62.50%	47.50%
000 1011	63.00%	48.00%
000 1100	63.50%	48.50%
000 1101	64.00%	49.00%
000 1110	64.50%	49.50%
000 0000	65.00%	50.00%
000 1111	65.50%	50.50%
001 0000	66.00%	51.00%
001 0001	66.50%	51.50%
001 0010	67.00%	52.00%
001 0011	67.50%	52.50%
001 0100	68.00%	53.00%
001 0101	68.50%	53.50%
001 0110	69.00%	54.00%
001 0111	69.50%	54.50%
001 1000	70.00%	55.00%
001 1001	70.50%	55.50%
001 1010	71.00%	56.00%
001 1011	71.50%	56.50%
001 1100	72.00%	57.00%
001 1101	72.50%	57.50%
001 1110	73.00%	58.00%
001 1111	73.50%	58.50%
010 0000	74.00%	59.00%

Table 1-11: Approximate  $V_{REF}$  Value as a Result of Using the  $V_{REF}$  Scan Function (Cont'd)

FABRIC_TUNE_VREF[6:0]	$V_{REF}$ (% of $V_{CC0}$ )	
	VREF_CNTR = FABRIC_RANGE1	VREF_CNTR = FABRIC_RANGE2
010 0001	74.50%	59.50%
010 0010	75.00%	60.00%
010 0011	75.50%	60.50%
010 0100	76.00%	61.00%
010 0101	76.50%	61.50%
010 0110	77.00%	62.00%
010 0111	77.50%	62.50%
010 1000	78.00%	63.00%
010 1001	78.50%	63.50%
010 1010	79.00%	64.00%
010 1011	79.50%	64.50%
010 1100	80.00%	65.00%
010 1101	80.50%	65.50%
010 1110	81.00%	66.00%
010 1111	81.50%	66.50%
011 0000	82.00%	67.00%
011 0001	82.50%	67.50%
011 0010	83.00%	68.00%
011 0011	83.50%	68.50%
011 0100	84.00%	69.00%
011 0101	84.50%	69.50%
011 0110	85.00%	70.00%
011 0111	85.50%	70.50%
011 1000	86.00%	71.00%
011 1001	86.50%	71.50%
011 1010	87.00%	72.00%
011 1011	87.50%	72.50%
011 1100	88.00%	73.00%
011 1101	88.50%	73.50%
011 1110	89.00%	74.00%
011 1111	89.50%	74.50%
100 0000	90.00%	75.00%
100 0001	90.50%	75.50%
100 0010	91.00%	76.00%

Table 1-11: Approximate  $V_{REF}$  Value as a Result of Using the  $V_{REF}$  Scan Function (Cont'd)

FABRIC_TUNE_VREF[6:0]	$V_{REF}$ (% of $V_{CC0}$ )	
	VREF_CNTR = FABRIC_RANGE1	VREF_CNTR = FABRIC_RANGE2
100 0011	91.50%	76.50%
100 0100	92.00%	77.00%
100 0101	92.50%	77.50%
100 0110	93.00%	78.00%
100 0111	93.50%	78.50%
100 1000	94.00%	79.00%

## SelectIO Interface Primitives

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

- IBUF (input buffer)
- IBUF\_ANALOG (input buffer specific to system monitor inputs). The IBUF\_ANALOG is used by the Vivado Design Suite tools to route analog signals to the SYSMONE1 or SYSMONE4 primitive. It is not a physical buffer and is purely a software construct that should be viewed as a physical pass through.
- IBUF\_IBUFDISABLE (input buffer with buffer disable control)
- IBUF\_INTERMDISABLE (input buffer with buffer disable and on-die input termination disable controls (HR I/O banks only))
- IBUFE3 (input buffer with offset calibration and  $V_{REF}$  tuning, along with buffer disable control (HP I/O banks only))
- IOBUF (bidirectional buffer)
- OBUF (output buffer)
- OBUFT (3-state output buffer)
- IOBUF\_DCIEN (bidirectional buffer with input buffer disable and on-die input termination disable control (HP I/O banks only))
- IOBUF\_INTERMDISABLE (bidirectional buffer with input buffer disable and on-die input termination disable control (HR I/O banks only))
- IOBUFE3 (bidirectional buffer with offset calibration and  $V_{REF}$  tuning, along with input buffer disable and on-die input termination enable control (HP I/O banks only))

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

- IBUFDS (differential input buffer)
- IBUFDS\_DIFF\_OUT (differential input buffer with complementary outputs). The IBUFDS\_DIFF\_OUT primitive can be beneficial to accommodate the need to invert differential pins that might have been inadvertently swapped
- IBUFDS\_DIFF\_OUT\_IBUFDISABLE (differential input buffer with complementary outputs and buffer disable)
- IBUFDS\_DIFF\_OUT\_INTERMDISABLE (differential input buffer with complementary outputs, input buffer disable and on-die input termination disable control (HR I/O banks only))
- IBUFDS\_IBUFDISABLE (differential input buffer with buffer disable control)
- IBUFDS\_INTERMDISABLE (differential input buffer with input buffer disable and on-die input termination disable control (HR I/O banks only))
- IBUFDSE3 (differential input buffer with offset calibration along with buffer disable control (HP I/O banks only))
- IBUFDS\_DPHY (differential input buffer for the MIPI D-PHY. Only supported by the HP I/O banks in the Virtex UltraScale+, Kintex UltraScale+, Artix UltraScale+, and Zynq UltraScale+ devices)
- IOBUFDS (differential bidirectional buffer)
- IOBUFDS\_DCIEN (differential bidirectional buffer with on-die input termination disable control and input buffer disable (HP I/O banks only))
- IOBUFDS\_DIFF\_OUT (differential bidirectional buffer with complementary outputs from the input buffer)
- IOBUFDS\_DIFF\_OUT\_DCIEN (differential bidirectional buffer with complementary outputs from the input buffer with on-die input termination disable controls and input buffer disable controls (HP I/O banks only))
- IOBUFDS\_INTERMDISABLE (bidirectional buffer with on-die input termination disable control and input buffer disable (HR I/O banks only))
- IOBUFDS\_DIFF\_OUT\_INTERMDISABLE (bidirectional buffer with complementary outputs from the input buffer with on-die input termination disable control and input buffer disable (HR I/O banks only))
- IOBUFDSE3 (differential bidirectional buffer with offset calibration along with input buffer disable and on-die input termination enable control (HP I/O banks only))
- OBUFDS (differential output buffer)
- OBUFTDS (differential 3-state output buffer)

- OBUFDS\_DPHY (differential output buffer for the MIPI D-PHY. Only supported by the HP I/O banks in the Virtex UltraScale+, Kintex UltraScale+, Artix UltraScale+, and Zynq UltraScale+ devices)
- HPIO\_VREF ( $V_{REF}$  scan feature (HP I/O banks only))

More information including instantiation techniques and available attributes for these and all other design primitives is available in the *UltraScale Architecture Libraries Guide* (UG974) [Ref 6].

## IBUF

Signals used as inputs must use an input buffer (IBUF). The generic IBUF primitive is shown in Figure 1-18.

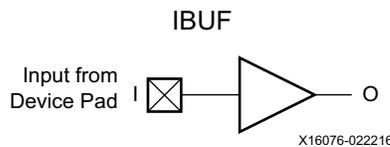


Figure 1-18: Input Buffer Primitive (IBUF)

## IBUF\_IBUFDISABLE

The IBUF\_IBUFDISABLE primitive shown in Figure 1-19 is an input buffer with a disable port that can be used as an additional power saving feature for periods when the input is not used.

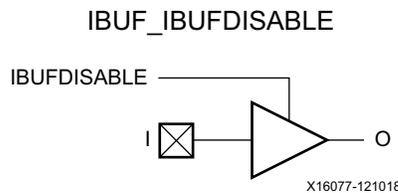
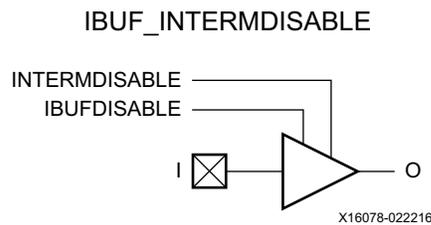


Figure 1-19: Input Buffer with Input Buffer Disable (IBUF\_IBUFDISABLE)

The IBUF\_IBUFDISABLE primitive 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. The USE\_IBUFDISABLE attribute must be set to TRUE, the IBUFDISABLE port must be controlled, and SIM\_DEVICE set to ULTRASCALE for this primitive to have the expected behavior that is specific to the UltraScale architecture. This feature can be used to reduce power at times when the I/O is idle. Input buffers that use the  $V_{REF}$  power rail (such as SSTL and HSTL) benefit the most from the IBUFDISABLE signal being set to a logic-High because they tend to have higher static power consumption than the non-VREF standards such as LVCMOS and LVTTTL.

## IBUF\_INTERMDISABLE

The IBUF\_INTERMDISABLE primitive shown in [Figure 1-20](#) is available in the HR I/O banks and is similar to the IBUF\_IBUFDISABLE primitive in that it has a IBUFDISABLE port that can be used to disable the input buffer during periods that the buffer is not being used. The USE\_IBUFDISABLE attribute must be set to TRUE, the IBUFDISABLE port must be controlled, and SIM\_DEVICE set to ULTRASCALE for this primitive to have the expected behavior that is specific to the UltraScale architecture. The IBUF\_INTERMDISABLE primitive also has an INTERMDISABLE port that can be used to disable the optional on-die receiver termination feature. See [Uncalibrated Input Termination in I/O Banks](#) for more details about this feature.



**Figure 1-20: Input Buffer with Input Buffer Disable and On-Die Input Termination Disable (IBUF\_INTERMDISABLE)**

The IBUF\_INTERMDISABLE primitive 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. The IBUF\_INTERMDISABLE primitive further allows the termination legs to be disabled whenever the INTERMDISABLE signal is asserted High. These features can be combined to reduce power whenever the input is idle. Input buffers that use the  $V_{REF}$  power rail (such as SSTL and HSTL) benefit the most from the IBUFDISABLE signal being set to a logic-High because they tend to have higher static power consumption than the non- $V_{REF}$  standards such as LVCMOS and LVTTTL.

## IBUFE3

The input buffer (IBUFE3) primitive (shown in [Figure 1-21](#)) is only supported in HP I/O banks. This UltraScale architecture specific primitive has functions similar to the [IBUF\\_IBUFDISABLE](#) with added controls for offset calibration and  $V_{REF}$  tuning, along with input buffer disable (IBUFDISABLE). The offset calibration feature is accessed using the OSC\_EN and OSC[3:0] ports. The  $V_{REF}$  scan feature is accessed using the HPIO\_VREF primitive in conjunction with IBUFE3.

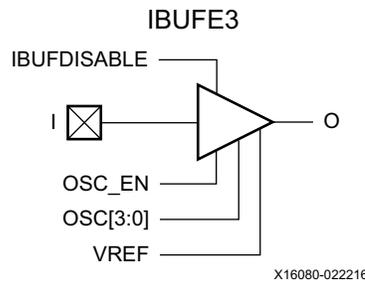


Figure 1-21: **IBUFE3 Primitive—Input Buffer with Offset Calibration and  $V_{REF}$  Tuning (HP I/O Banks Only)**

## IBUFDS

The usage and rules corresponding to the differential primitives are similar to the single-ended SelectIO primitives. Differential SelectIO primitives have two pins to and from the device pads to show the P and N channel pins in a differential pair. N channel pins have a B suffix.

Figure 1-22 shows the differential input buffer primitive.

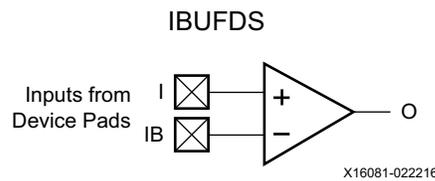


Figure 1-22: **Differential Input Buffer Primitive (IBUFDS)**

## IBUFDS\_DIFF\_OUT

Figure 1-23 shows the differential input buffer primitive with complementary outputs (O and OB).

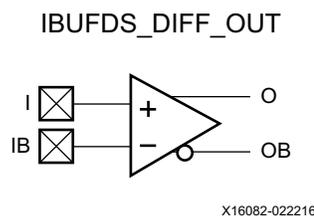


Figure 1-23: **Differential Input Buffer Primitive with Complementary Outputs (IBUFDS\_DIFF\_OUT)**



**IMPORTANT:** When this primitive is used for a clock signal, the OB output to the interconnect logic has no direct connection to a clock buffer.

## IBUFDS\_DIFF\_OUT\_IBUFDISABLE

The IBUFDS\_DIFF\_OUT\_IBUFDISABLE primitive shown in Figure 1-24 is a differential input buffer with complementary differential outputs.



**TIP:** The IBUFDISABLE feature is not supported with this primitive in the UltraScale architecture.

IBUFDS\_DIFF\_OUT\_IBUFDISABLE

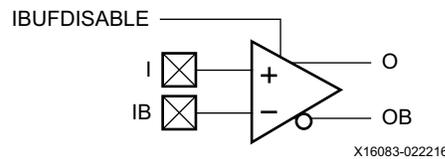


Figure 1-24: Differential Input Buffer with Complementary Outputs and Input Buffer Disable (IBUFDS\_DIFF\_OUT\_IBUFDISABLE)

## IBUFDS\_DIFF\_OUT\_INTERMDISABLE

The IBUFDS\_DIFF\_OUT\_INTERMDISABLE primitive shown in Figure 1-25 is available in the HR I/O banks. It has complementary differential outputs and an INTERMDISABLE port that can be used to manually disable the optional on-die receiver termination features (uncalibrated). See [Uncalibrated Input Termination in I/O Banks](#) for more details.



**TIP:** The IBUFDISABLE feature is not supported with this primitive in the UltraScale architecture.

IBUFDS\_DIFF\_OUT\_INTERMDISABLE

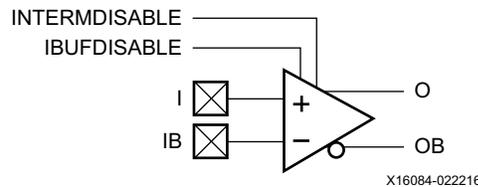
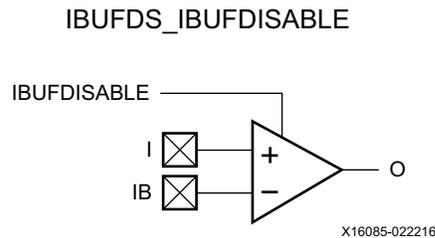


Figure 1-25: Differential Input Buffer with Complementary Outputs, Input Path Disable, and On-Die Input Termination Disable (IBUFDS\_DIFF\_OUT\_INTERMDISABLE)

If the I/O is using any on-die receiver termination features (uncalibrated), this primitive disables the termination legs whenever the INTERMDISABLE signal is asserted High.

## IBUFDS\_IBUFDISABLE

The IBUFDS\_IBUFDISABLE primitive shown in [Figure 1-26](#) is a differential input buffer with a disable port that can be used as an additional power saving feature for periods when the input is not used.

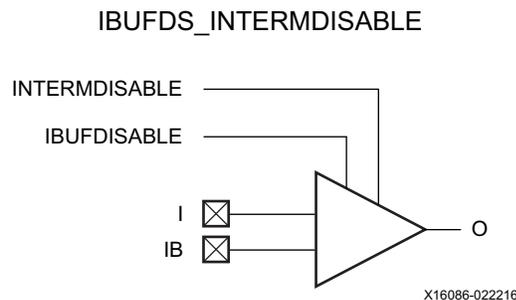


*Figure 1-26: Differential Input Buffer with Input Buffer Disable (IBUFDS\_IBUFDISABLE)*

The IBUFDS\_IBUFDISABLE primitive 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. The USE\_IBUFDISABLE attribute must be set to TRUE, the IBUFDISABLE port must be controlled, and SIM\_DEVICE set to ULTRASCALE for this primitive to have the expected behavior that is specific to the UltraScale architecture. This feature can be used to reduce power whenever the I/O is idle.

## IBUFDS\_INTERMDISABLE

The IBUFDS\_INTERMDISABLE primitive (shown in [Figure 1-27](#)), available in the HR I/O banks, is similar to the IBUFDS\_IBUFDISABLE primitive because it has a IBUFDISABLE port to disable the input buffer when not in use. The IBUFDS\_INTERMDISABLE primitive also has an INTERMDISABLE port to use to disable the optional on-die receiver termination feature. See [Uncalibrated Input Termination in I/O Banks](#) for more details.



*Figure 1-27: Differential Input Buffer with Input Buffer Disable and On-Die Input Termination Disable (IBUFDS\_INTERMDISABLE)*

The IBUFDS\_INTERMDISABLE primitive can disable the input buffer and force the O output 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 ULTRASCALE for this primitive to have the expected behavior that is specific to the UltraScale architecture. If the I/O is using the optional on-die receiver termination feature, this primitive disables the termination legs whenever the INTERMDISABLE signal is asserted High. Both these features can be combined to reduce power whenever the input is idle.

### IBUFDSE3

The differential input buffer (IBUFDSE3) primitive is only supported in HP I/O banks (Figure 1-28). This UltraScale architecture specific primitive has functions similar to the IBUFDS\_IBUFDISABLE along with controls for offset calibration and input buffer disable (IBUFDISABLE). The offset calibration feature is accessed using the OSC\_EN[1:0] and OSC[3:0] ports. The V<sub>REF</sub> scan feature is not supported with this primitive.

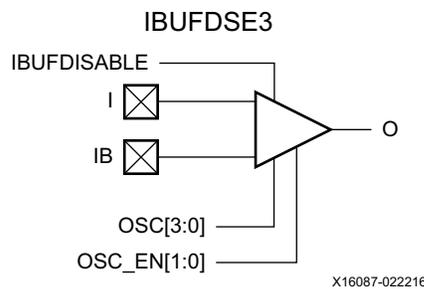


Figure 1-28: IBUFDSE3 Primitive—Differential Input Buffer with Offset Calibration (HP I/O Banks Only)

### IBUFDS\_DPHY

The differential input buffer primitive (IBUFDS\_DPHY) is only supported in the HP I/O banks in Virtex UltraScale+ devices (Figure 1-29), Kintex UltraScale+ devices, Artix UltraScale+ devices, and Zynq UltraScale+ devices. This UltraScale architecture specific primitive is specifically meant for MIPI D-PHY receiver implementation. The HSRX\_DISABLE port is used to enable or disable the MIPI D-PHY high-speed (HS) receiver. The LPRX\_DISABLE port is used to enable or disable the low-power (LP) receiver. HSRX\_O and LPRX\_O(P/\_N) are inputs to the interconnect logic from the HS and LP receivers, respectively. The primitive only supports MIPI\_DPHY\_DCI as the value for the IOSTANDARD attribute.

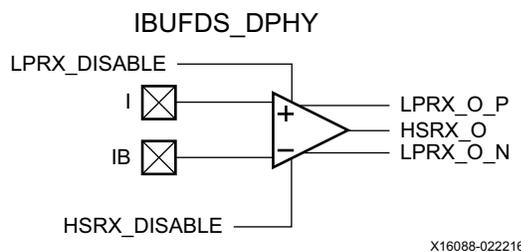


Figure 1-29: Differential Input Buffer Primitive (IBUFDS\_DPHY)

## IOBUF

The IOBUF primitive is needed when bidirectional signals require both an input buffer and a 3-state output buffer with an active-High 3-state T pin. Figure 1-30 shows a generic IOBUF. A logic-High on the T pin disables the output buffer. When the output buffer is 3-stated (T = High), the input buffer and any on-die receiver termination (uncalibrated or DCI) are ON. When the output buffer is not 3-stated (T = Low), any on-die receiver termination (uncalibrated or DCI) is disabled.

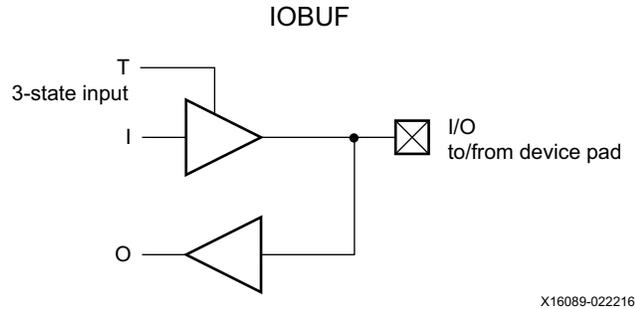


Figure 1-30: Input/Output Buffer Primitive (IOBUF)

## IOBUF\_DCIE

The IOBUF\_DCIE primitive shown in Figure 1-31 is available in the HP I/O banks. It has an IBUFDISABLE port that can be used to disable the input buffer during periods that the buffer is not being used. The IOBUF\_DCIE primitive also has a DCITERMDISABLE port that can be used to manually disable the optional on-die receiver termination features (uncalibrated and DCI). See the DCI—Only Available in the HP I/O Banks and Uncalibrated Input Termination in I/O Banks sections for more details.

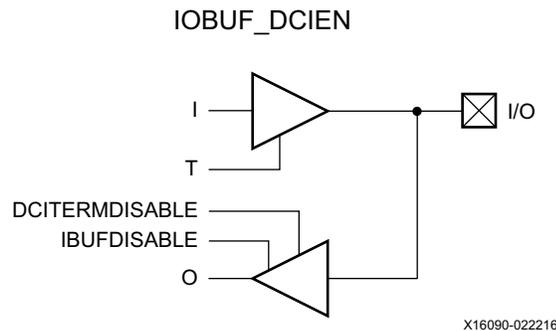


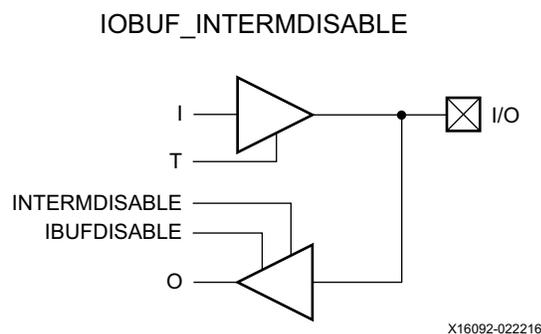
Figure 1-31: Input/Output Buffer DCI Enable Primitive (IOBUF\_DCIE)

The IOBUF\_DCIE primitive 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 and output buffer is 3-stated (T = High). If the I/O is using any on-die receiver termination features (uncalibrated and DCI), this primitive disables the termination legs whenever the

DCITERMDISABLE signal is asserted High and the output buffer is 3-stated ( $T = \text{High}$ ). When the output buffer is 3-stated ( $T = \text{High}$ ), the input buffer and any on-die receiver termination (uncalibrated or DCI) are controlled by IBUFDISABLE and DCITERMDISABLE, respectively. The USE\_IBUFDISABLE attribute must be set to TRUE, the IBUFDISABLE port must be controlled, and SIM\_DEVICE set to ULTRASCALE for this primitive to have the expected behavior that is specific to the UltraScale architecture. When the output buffer is not 3-stated ( $T = \text{Low}$ ), the input buffer and any on-die receiver termination (uncalibrated or DCI) are disabled and the O output (to the internal logic) is forced to a logic-Low. These features can be combined to reduce power whenever the input is idle for a period of time. Although uncommon, if a design requires an input be constantly enabled while maintaining the ability to dynamically control DCI, this primitive can be used by floating the IBUFDISABLE pin and setting the USE\_IBUFDISABLE attribute to FALSE.

## IOBUF\_INTERMDISABLE

The IOBUF\_INTERMDISABLE primitive shown in [Figure 1-32](#) is available in the HR I/O banks. It has an IBUFDISABLE port that can be used to disable the input buffer during periods that the buffer is not being used. The IOBUF\_INTERMDISABLE primitive also has an INTERMDISABLE port that can be used to manually disable the optional on-die receiver termination feature. See [Uncalibrated Input Termination in I/O Banks](#) for more details.



*Figure 1-32:* **Bidirectional Buffer with Input Path Disable and On-Die Input Termination Disable (IOBUF\_INTERMDISABLE)**

The IOBUF\_INTERMDISABLE primitive 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 and the output buffer is 3-stated ( $T = \text{High}$ ). If the I/O is using the on-die receiver termination feature (uncalibrated), this primitive disables the termination legs whenever the INTERMDISABLE signal is asserted High and the output buffer is 3-stated ( $T = \text{High}$ ). When the output buffer is 3-stated ( $T = \text{High}$ ), the input buffer and any on-die receiver termination are controlled by IBUFDISABLE and INTERMDISABLE, respectively. The USE\_IBUFDISABLE attribute must be set to TRUE, the IBUFDISABLE port must be controlled, and SIM\_DEVICE set to ULTRASCALE for this primitive to have the expected behavior that is specific to the UltraScale architecture. When the output buffer is not 3-stated ( $T = \text{Low}$ ), the input buffer and any on-die receiver termination are disabled and the O output (to the

internal logic) is forced to a logic-Low. These features can be combined to reduce power whenever the input is idle for a period of time.

## IOBUFE3

The bidirectional input/output buffer primitive (IOBUFE3) is only supported in HP I/O banks (Figure 1-33). This UltraScale architecture specific primitive has functions similar to the IOBUF\_DCIEN along with controls for offset calibration and  $V_{REF}$  tuning with input buffer disable (IBUFDISABLE) and on-die input termination control (DCITERMDISABLE) for the input buffer. The offset calibration feature is accessed using the OSC\_EN and OSC[3:0] ports. The  $V_{REF}$  scan feature is accessed using the HPIO\_VREF primitive in conjunction with IOBUFE3.

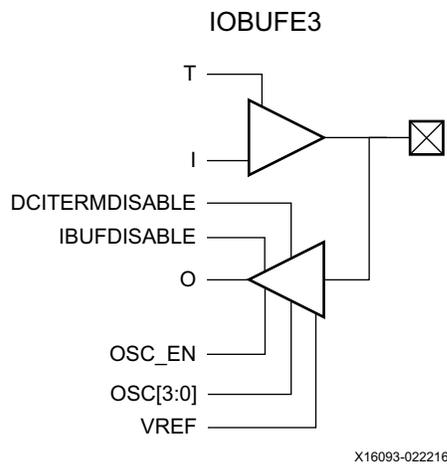


Figure 1-33: IOBUFE3 Primitive—Bidirectional I/O Buffer with Offset Calibration and  $V_{REF}$  Tuning (HP I/O Banks Only)

## IOBUFDS

Figure 1-34 shows the differential input/output buffer primitive. A logic-High on the T pin disables the output buffer. When the output buffer is 3-stated (T = High), the input buffer and any on-die receiver termination (uncalibrated or DCI) are ON. When the output buffer is not 3-stated (T = Low), any on-die receiver termination (uncalibrated or DCI) is disabled.

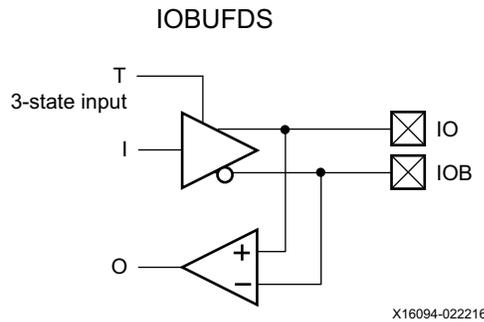


Figure 1-34: Differential Input/Output Buffer Primitive (IOBUFDS)

## IOBUFDS\_DCIEEN

The IOBUFDS\_DCIEEN primitive shown in Figure 1-35 is available in the HP I/O banks. It has an IBUFDISABLE port that can be used to disable the input buffer during periods that the buffer is not being used. The USE\_IBUFDISABLE attribute must be set to TRUE, the IBUFDISABLE port must be controlled, and SIM\_DEVICE set to ULTRASCALE for this primitive to have the expected behavior that is specific to the UltraScale architecture. The IOBUFDS\_DCIEEN primitive also has a DCITERMDISABLE port that can be used to manually disable the optional on-die receiver termination features (uncalibrated or DCI). See DCI—Only Available in the HP I/O Banks and Uncalibrated Input Termination in I/O Banks for more details.

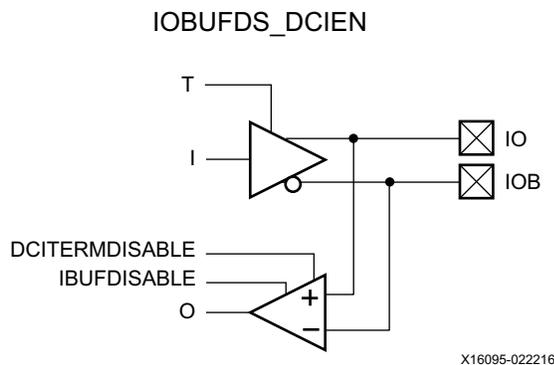


Figure 1-35: Differential Bidirectional Buffer with Input Buffer Disable and On-Die Input Termination Disable (IOBUFDS\_DCIEEN)

The IOBUFDS\_DCIEN primitive 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 and the output buffer is 3-stated (T = High). If the I/O is using an on-die receiver termination feature (uncalibrated or DCI), this primitive disables the termination legs whenever the DCITERMDISABLE signal is asserted High and the output buffer is 3-stated (T = High).

When the output buffer is 3-stated (T = High), the input buffer and any on-die receiver termination (uncalibrated or DCI) are controlled by IBUFDISABLE and DCITERMDISABLE, respectively. When the output buffer is not 3-stated (T = Low), the input buffer and any on-die receiver termination (uncalibrated or DCI) are disabled and force the O output (to the internal logic) to a logic-Low. These features can be combined to reduce power whenever the input is idle for a period of time.

Although uncommon, if a design requires an input be constantly enabled while maintaining the ability to dynamically control DCI, this primitive can be used by floating the IBUFDISABLE pin and setting the USE\_IBUFDISABLE attribute to FALSE.

### IOBUFDS\_DIFF\_OUT

Figure 1-36 shows the differential input/output buffer primitive with complementary outputs (O and OB). A logic-High on the T pin disables the output buffers. When the output buffers are 3-stated (T = High), the input buffer and any on-die receiver termination (uncalibrated or DCI) are ON. When the output buffer is not 3-stated (T = Low), any on-die receiver termination (uncalibrated or DCI) is disabled. TM and TS must be connected to the same input from the interconnect logic for this primitive to have the expected behavior that is specific to the UltraScale architecture.

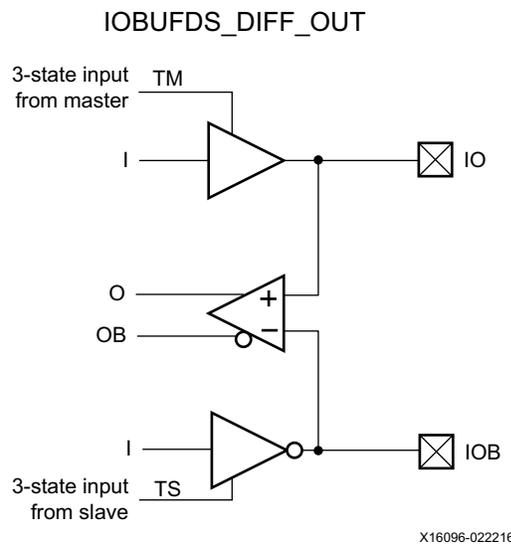
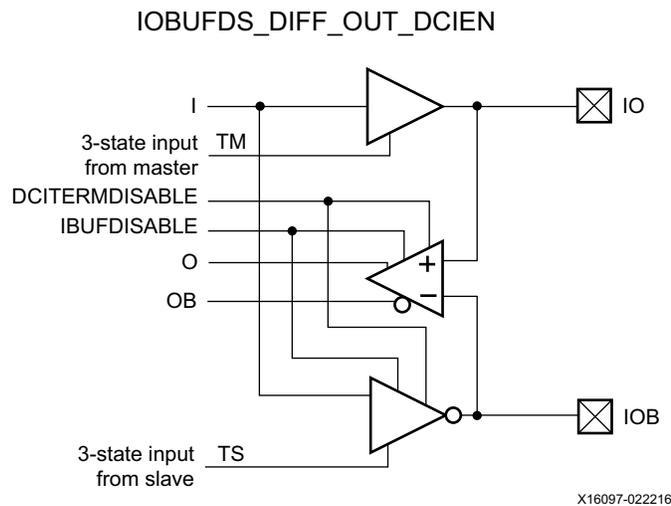


Figure 1-36: Differential Input/Output Buffer Primitive with Complementary Outputs for the Input Buffer (IOBUFDS\_DIFF\_OUT)

## IOBUFDS\_DIFF\_OUT\_DCIEEN

The IOBUFDS\_DIFF\_OUT\_DCIEEN primitive shown in [Figure 1-37](#) is available in the HP I/O banks. It has complementary differential outputs, an IbufDisable port, and a DCITerMDisable port that can be used to manually disable the optional DCI on-die receiver termination features (uncalibrated or DCI). See [DCI—Only Available in the HP I/O Banks](#) and [Uncalibrated Input Termination in I/O Banks](#) for more details. TM and TS must be connected to the same input from the interconnect logic for this primitive to have the expected behavior that is specific to the UltraScale architecture.



**Figure 1-37: Differential Bidirectional Buffer with Complementary Outputs, Input Path Disable, and On-Die Input Termination Disable (IOBUFDS\_DIFF\_OUT\_DCIEEN)**

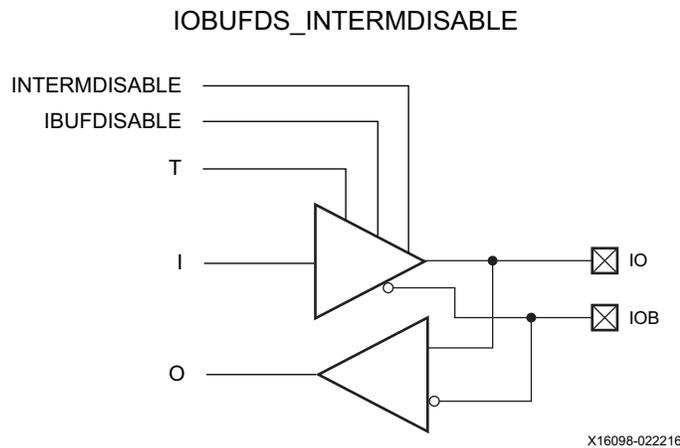
If the I/O is using any on-die receiver termination features (uncalibrated or DCI), this primitive disables the termination legs whenever the DCITerMDisable signal is asserted High and the output buffer is 3-stated. When the output buffer is 3-stated (T = High), any on-die receiver termination (uncalibrated or DCI) is controlled by DCITerMDisable. When the output buffer is not 3-stated (T = Low), the input buffer and on-die receiver termination (uncalibrated or DCI) are disabled and the O output (to the internal logic) is forced to a logic-Low.



**TIP:** The IbufDisable feature is not supported with this primitive in the UltraScale architecture.

## IOBUFDS\_INTERMDISABLE

The IOBUFDS\_INTERMDISABLE primitive (shown in [Figure 1-38](#)) is available in the HR I/O banks. It has an IBUFDISABLE port that can be used to disable the input buffer during periods when the buffer is not being used. The IOBUFDS\_INTERMDISABLE primitive also has an INTERMDISABLE port that can be used to disable the optional on-die receiver termination feature. See [Uncalibrated Input Termination in I/O Banks](#) for more details on this feature.



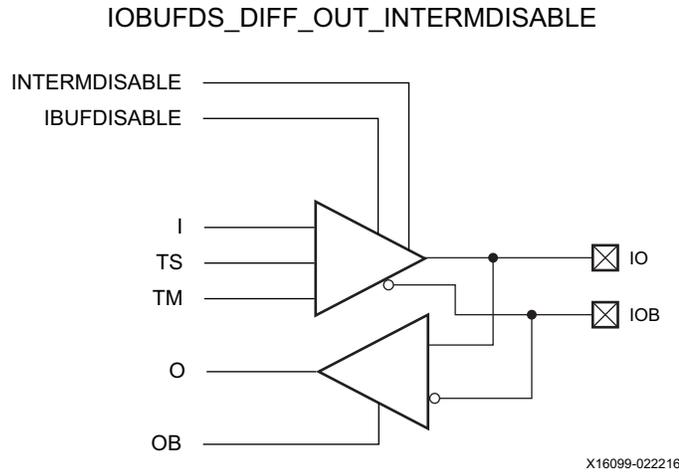
**Figure 1-38: Differential Bidirectional Buffer with Input Buffer Disable and On-Die Input Termination Disable (IOBUFDS\_INTERMDISABLE)**

The IOBUFDS\_INTERMDISABLE primitive 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 and the output buffer is 3-stated (T = High). The USE\_IBUFDISABLE attribute must be set to TRUE, the IBUFDISABLE port must be controlled, and SIM\_DEVICE set to ULTRASCALE for this primitive to have the expected behavior that is specific to the UltraScale architecture. If the I/O is using the on-die receiver termination feature, this primitive disables the termination legs whenever the INTERMDISABLE signal is asserted High and the output buffer is 3-stated. When the output buffer is 3-stated (T = High), the input buffer and any on-die receiver termination are controlled by IBUFDISABLE and INTERMFIDISABLE, respectively. When the output buffer is not 3-stated (T = Low), the input buffer and on-die receiver termination are disabled and the O output (to the internal logic) is forced to a logic-Low. These features can be combined to reduce power whenever the input is idle for a period of time.

Although uncommon, if a design requires an input be constantly enabled while maintaining the ability to dynamically control DCI, this primitive can be used by floating the IBUFDISABLE pin and setting the USE\_IBUFDISABLE attribute to FALSE.

## IOBUFDS\_DIFF\_OUT\_INTERMDISABLE

The IOBUFDS\_DIFF\_OUT\_INTERMDISABLE primitive (shown in Figure 1-39) is available in the HR I/O banks. The IOBUFDS\_DIFF\_OUT\_INTERMDISABLE primitive has an INTERMDISABLE port that can be used to disable the optional on-die receiver termination feature. See [Uncalibrated Input Termination in I/O Banks](#) for more details on this feature. TM and TS must be connected to the same input (T) from the interconnect logic for this primitive to have the expected behavior that is specific to the UltraScale architecture.



**Figure 1-39: Differential Bidirectional Buffer with Complementary Outputs, Input Buffer Disable, and On-Die Input Termination Disable (IOBUFDS\_DIFF\_OUT\_INTERMDISABLE)**

If the I/O is using the on-die receiver termination features, this primitive disables the termination legs whenever the INTERMDISABLE signal is asserted High and the output buffer is 3-stated. When the output buffer is 3-stated (T = High), any on-die receiver termination is controlled by INTERMDISABLE. When the output buffer is not 3-stated (T = Low), the input buffer and on-die receiver termination are disabled and the O output (to the internal logic) is forced to a logic-Low.



**TIP:** The IBUFDISABLE feature is not supported with this primitive in the UltraScale architecture.

## IOBUFDSE3

The differential bidirectional input/output buffer primitive (IOBUFDSE3) is only supported in HP I/O banks. This UltraScale architecture specific primitive has functions similar to the [IOBUFDS\\_DCIEN](#) along with controls for offset calibration with input buffer disable control (IBUFDISABLE) and on-die input termination disable control (DCITERMDISABLE) for the input buffer. The offset calibration feature is accessed using the OSC\_EN[1:0] and OSC[3:0] ports. The  $V_{REF}$  scan feature is not supported with this primitive.

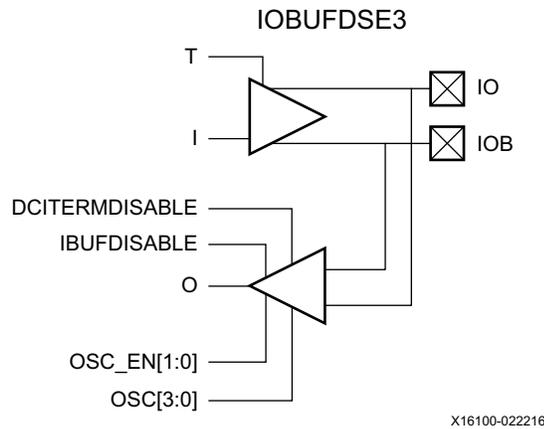


Figure 1-40: IOBUFDSE3 Primitive—Differential Bidirectional I/O Buffer with Offset Calibration (HP I/O Banks Only)

## OBUF

An output buffer (OBUF) must be used to drive signals from the device to external output pads. A generic OBUF primitive is shown in [Figure 1-41](#).

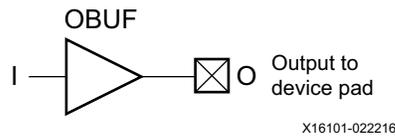


Figure 1-41: Output Buffer Primitive (OBUF)

## OBUFDS

Figure 1-42 shows the differential output buffer primitive.

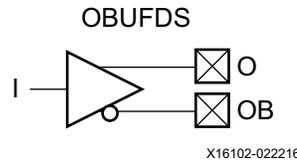


Figure 1-42: Differential Output Buffer Primitive (OBUFDS)

## OBUFT

The generic 3-state output buffer OBUFT, shown in Figure 1-43, typically implements 3-state outputs or bidirectional I/O.

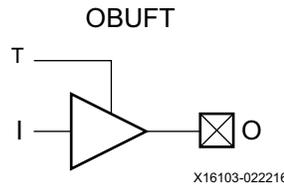


Figure 1-43: 3-State Output Buffer Primitive (OBUFT)

## OBUFTDS

Figure 1-44 shows the differential 3-state output buffer primitive.

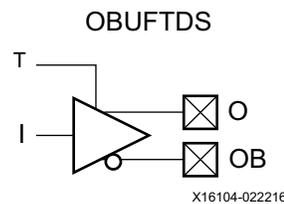


Figure 1-44: Differential 3-State Output Buffer Primitive (OBUFTDS)

## OBUFDS\_DPHY

The differential output buffer primitive (OBUFDS\_DPHY) is only supported in the HP I/O banks in the Virtex UltraScale+, Kintex UltraScale+, Artix UltraScale+, and Zynq UltraScale+ devices (Figure 1-45). This UltraScale architecture specific primitive is for MIPI D-PHY transmitter implementation. The HSTX\_T port is used to 3-state the MIPI D-PHY high-speed (HS) transmitter. The LPTX\_T port is used to 3-state the low-power (LP) transmitter. The HSTX\_I and LPTX\_I(P/\_N) inputs are from the interconnect logic to the HS

and LP transmitters, respectively. The primitive only has support for MIPI\_DPHY\_DCI as the value for the IOSTANDARD attribute.

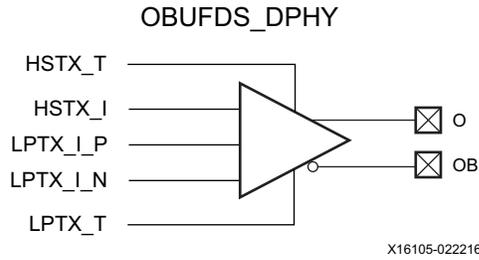


Figure 1-45: Differential Output Buffer Primitive (OBUFDS\_DPHY)

## HPIO\_VREF

The HPIO\_VREF primitive is only supported in HP I/O banks (Figure 1-46). This UltraScale architecture specific primitive provides access to the  $V_{REF}$  scan feature that is available in HP I/O banks. The  $V_{REF}$  scan feature is accessed using the HPIO\_VREF primitive in conjunction with either the IBUFE3 or IOBUFE3 primitives.

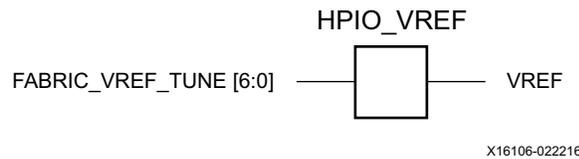


Figure 1-46: HPIO\_VREF Primitive— $V_{REF}$  Scan Feature (HP I/O Banks Only)

## SelectIO Interface Attributes and Constraints

Access to some I/O resource features (for example, location constraints, input delay, output drive strength, and slew rate) is available through the attributes/constraints associated with these features. For more information about implementing these constraints and attributes as well as others, see the *Vivado Design Suite Properties Reference Guide* (UG912) [Ref 7].

### DCI\_CASCADE Constraint

The DCI\_CASCADE constraint identifies a DCI master bank and its corresponding slave banks. See [DCI Cascading](#), page 21 for more information.

The DCI\_CASCADE attribute uses the following syntax in the Xilinx Design Constraints (XDC) file:

```
set_property DCI_CASCADE {slave_banks} [get_iobanks master_bank]
```

## PACKAGE\_PIN Constraint

The PACKAGE\_PIN constraint must be used to specify the I/O location of an external port identifier (A8, M5, or AM6). These values are device and package size dependent.

The PACKAGE\_PIN attribute uses the following syntax in the XDC file:

```
set_property PACKAGE_PIN pin_name [get_ports port_name]
```

## IOSTANDARD Attribute

The IOSTANDARD attribute is available to choose the values for an I/O standard for all I/O buffers. The supported I/O standards are listed in the specific UltraScale device data sheet [Ref 2]; however, Table 1-77 lists the IOSTANDARD support by I/O bank type (HR, HP, or both). The IOSTANDARD attribute uses the following syntax in the XDC file:

```
set_property IOSTANDARD value [get_ports port_name]
```

## IBUF\_LOW\_PWR Attribute

The IBUF\_LOW\_PWR attribute allows an optional trade-off between performance and power. This attribute is set to TRUE by default, which implements the input buffer in the lower-power rather than the higher-performance mode.



---

**RECOMMENDED:** For receivers that are expected to operate at data rates  $\geq 1600$  Mb/s, this attribute should be set to FALSE.

---

The change in power between high-performance and low-power mode can be estimated using the Xilinx Power Estimator (XPE) spreadsheet tool (download at [www.xilinx.com/power](http://www.xilinx.com/power)).

The IBUF\_LOW\_PWR attribute is applied to the I/O buffer instance and uses the following syntax in the XDC file:

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

See the *UltraScale Architecture Libraries Guide* (UG974) [Ref 6] for information on accessing this attribute in UNISIM instantiation.

## Output Slew Rate Attributes

Many attribute values provide the option of choosing the desired slew rate for I/O output buffers. For LVCMOS, LVTTTL, SSTL, HSTL, and HSUL output buffers, including the differential versions, the desired slew rate can be specified with the SLEW attribute.

Although the default SLEW attribute is SLOW, it might be important to specify FAST slew rate for high-performance applications such as high-frequency memory interfaces.

However, faster slew rates can also lead to reflections or increased noise issues if not properly designed (such as with terminations, transmission line impedance continuity, and cross-coupling).

The allowed values for the SLEW attribute are SLOW, MEDIUM (HP I/O banks only), or FAST.

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

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

By default, the slew rate for each output buffer is set to SLOW. This is the default used to minimize the power bus transients when switching non-critical signals.

## Output Drive Strength Attributes

For LVCMOS and LVTTTL output buffers, the DRIVE attribute can be used to specify the load that the driver can safely drive to valid logic levels (in mA). The allowed values for the DRIVE attribute are shown in [Table 1-12](#).

**Table 1-12: Allowed Values for the DRIVE Attribute**

Standard	HR I/O Bank Current Drive (mA)		HP I/O Bank Current Drive (mA)	
	Allowed Values	Default	Allowed Values	Default
LVCMOS12	4, 8, or 12	12	2, 4, 6, or 8	12 <sup>(1)</sup>
LVCMOS15	4, 8, 12, or 16	12	2, 4, 6, 8, or 12	12
LVCMOS18	4, 8, 12, or 16	12	2, 4, 6, 8, or 12	12
LVCMOS25	4, 8, 12, or 16	12	N/A	N/A
LVCMOS33	4, 8, 12, or 16	12	N/A	N/A
LVTTTL	4, 8, 12, or 16	12	N/A	N/A

**Notes:**

1. Change the drive setting from the default setting in the RTL or XDC file to one of the allowed values before running through the Vivado Design Suite.

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

```
set_property DRIVE drive_value [get_ports port_name]
```

## PULLTYPE Attribute

Input buffers, 3-state outputs, and bidirectional buffers can have a weak pull-up resistor, a weak pull-down resistor, or a weak keeper circuit. PULLTYPE attribute has the following possible values.

- NONE
- PULLUP
- PULLDOWN

- KEEPER

This feature can be invoked by adding the following possible constraint values to the relevant net of the buffers. These attributes use the following syntaxes in the XDC file:

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

For more information on implementing these attributes on either individual I/Os or globally for all I/Os, see the pull-up, pull-down, and keeper descriptions in the *Vivado Design Suite Properties Reference Guide* (UG912) [Ref 7].

## On-Die Termination (ODT) Attribute

The ODT attribute supports split or single termination on the inputs of the HSTL, SSTL, POD, and HSUL standards. The advantage of using ODT over discrete resistors is that signal integrity is improved by completely removing the stub at the receiver.

The ODT attribute is used to define the value of the on-die termination at the input for both DCI and non-DCI versions of the standards supported.

The  $V_{CCO}$  of the I/O bank must be connected to the appropriate voltage level for the ODT attribute to perform as expected.

**Note:** For additional information, see [Table 1-77](#) in [Rules for Combining I/O Standards in the Same Bank](#) for the  $V_{CCO}$  levels required for I/O standards.

Allowed values for the ODT attribute:

- RTT\_40
- RTT\_48
- RTT\_60
- RTT\_120
- RTT\_240
- RTT\_NONE

**Note:** Not all values are allowed for all applicable I/O standards and configurations.

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

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

## Source Termination Attribute (OUTPUT\_IMPEDANCE)

The OUTPUT\_IMPEDANCE attribute provides the option of choosing the driver impedance for HSTL, SSTL, HSUL, LVDCI, HSLVDCI, and POD drivers to match the characteristic impedance of the driven line. The OUTPUT\_IMPEDANCE attribute is used to define the value of source termination at the driver for both DCI and non-DCI versions of the standards supported.

Allowed values for the OUTPUT\_IMPEDANCE attribute:

- RDRV\_40\_40
- RDRV\_48\_48
- RDRV\_60\_60
- RDRV\_NONE\_NONE

**Note:** Not all values are allowed for all applicable I/O standards and configurations.

The XDC syntax for this attribute is:

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

## Differential Termination Attribute

The differential termination (DIFF\_TERM or DIFF\_TERM\_ADV) attributes support the differential I/O standards when used as inputs. These attributes are used to turn the built-in, 100Ω, differential termination on or off. The on-chip input differential termination provides advantages over using a discrete resistor by completely removing the stub at the receiver, which improves signal integrity. Additionally it:

- Consumes less power than DCI termination
- Does not use VRP pins (DCI)

The  $V_{CC0}$  of the I/O bank must be connected to 1.8V for HP I/O banks and 2.5V for HR I/O banks to provide 100Ω of effective differential termination. DIFF\_TERM and DIFF\_TERM\_ADV are only available for inputs and can *only* be used with the appropriate  $V_{CC0}$  voltage.

The DIFF\_TERM\_ADV attribute can be specified in the XDC constraints file. The DIFF\_TERM attribute can be specified by setting the appropriate value in the generic map (VHDL) or in-line parameter (Verilog) of the instantiated primitives. Refer to the Vivado Design Suite HDL templates [\[Ref 12\]](#) [\[Ref 9\]](#) or the *UltraScale Architecture Libraries Guide* (UG974) [\[Ref 6\]](#) for the proper syntax for instantiating these primitives and setting the DIFF\_TERM attribute.

Differential termination can either be invoked using the DIFF\_TERM or DIFF\_TERM\_ADV attributes. DIFF\_TERM is used if specified in the instantiated primitive. DIFF\_TERM\_ADV is

used if specified in the XDC constraints file. DIFF\_TERM values specified in the instantiated primitive gets translated to the corresponding DIFF\_TERM\_ADV setting in the XDC file.

The allowed values for the DIFF\_TERM attribute are:

- DIFF\_TERM = TRUE automatically maps to DIFF\_TERM\_ADV = TERM\_100
- DIFF\_TERM = FALSE automatically maps to DIFF\_TERM\_ADV = TERM\_NONE (default)

The allowed values for DIFF\_TERM\_ADV attribute are:

- DIFF\_TERM\_ADV = TERM\_NONE (default)
- DIFF\_TERM\_ADV = TERM\_100

The DIFF\_TERM\_ADV attribute uses the following syntax in the XDC file:

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

## Internal $V_{REF}$

The  $V_{REF}$  for I/O banks can be (optionally) generated inside UltraScale devices. Internal generation removes the need to provide for a particular  $V_{REF}$  supply rail on the printed circuit board (PCB). The internally generated  $V_{REF}$  (INTERNAL\_VREF) is sourced from the  $V_{CC0}$ .

In I/O banks, there is a one  $V_{REF}$  plane per bank and each bank can have the optional INTERNAL\_VREF set to a single voltage level for the entire bank.

The constraint INTERNAL\_VREF is assigned to one bank at a time.

Example 1: INTERNAL\_VREF for Bank 84 using HSTL\_II (1.5V), which requires a 0.75V reference voltage, uses the following constraint:

```
set_property INTERNAL_VREF 0.75 [get_iobanks 84]
```

Example 2: INTERNAL\_VREF for Bank 65 using HSTL\_II\_18 (1.8V), which requires a 0.9V reference voltage, uses the following constraint.

```
set_property INTERNAL_VREF 0.90 [get_iobanks 65]
```

The rules for using INTERNAL\_VREF are:

- One value of  $V_{REF}$  can be set for the bank.
- INTERNAL\_VREF can only be set to the nominal reference voltage value of a given I/O standard.
- Valid settings of INTERNAL\_VREF are listed. Not all values are supported in all types of banks:
  - 0.60

- 0.675
- 0.70
- 0.75
- 0.84
- 0.90
- $V_{REF}$  is a dedicated pin and cannot be used as a normal I/O pin even when INTERNAL\_VREF is used.

The rules for combining I/O standards in the same bank also apply for INTERNAL\_VREF. In HP I/O banks only, an internal  $V_{REF}$  scan feature is available for internal  $V_{REF}$  control. To use the  $V_{REF}$  scan feature in a bank, the INTERNAL\_VREF must be set to the appropriate  $V_{REF}$  value for the I/O standards used in that bank. The internal  $V_{REF}$  scan is invoked using either the IbufE3 or IObufE3 primitive in conjunction with the HPIO\_VREF primitive.

Within a bank, you cannot combine or use both the internal  $V_{REF}$  (INTERNAL\_VREF or  $V_{REF}$  scan) with the external  $V_{REF}$  pins. When either INTERNAL\_VREF or  $V_{REF}$  scan is used in a bank, connect the dedicated external  $V_{REF}$  pins to GND through a 500 $\Omega$  or 1K $\Omega$  resistor.

## DQS\_BIAS

DQS\_BIAS behaves as a logic holding mechanism for undriven pins in pseudo-differential (DIFF\_SSTL, DIFF\_HSUL, and DIFF\_POD) buffers by weakly pulling the N side of the buffer to  $V_{CC0}$  and the P side of the buffer to ground. For LVDS inputs, DQS\_BIAS provides a DC bias of  $V_{CC0}/2$  to both the P and N sides of the buffer.

The allowed values for the DQS\_BIAS attribute for applicable I/O standards are:

- TRUE where DQS\_BIAS = TRUE cannot be used in conjunction with the PULLTYPE attribute set to PULLUP, PULLDOWN, or KEEPER in the same port.
- FALSE (default)




---

**IMPORTANT:** Starting with Vivado Design Suite 2018.1, the DQS\_BIAS attribute should be set on the port, not on the cell.

---

The DQS\_BIAS attribute should be set on the port using the syntax:

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

## Transmitter Pre-Emphasis

The transmitter pre-emphasis (PRE\_EMPHASIS) feature allows pre-emphasis on the drivers for certain I/O standards. This attribute must be used in conjunction with ENABLE\_PRE\_EMPHASIS.

The allowed values for the PRE\_EMPHASIS attribute are:

- PRE\_EMPHASIS = RDRV\_NONE (default)
- PRE\_EMPHASIS = RDRV\_240 (where ENABLE\_PRE\_EMPHASIS must be set to TRUE)

The PRE\_EMPHASIS attribute uses the following syntax in the XDC file:

```
set_property PRE_EMPHASIS value [get_ports port_name]
```

A typical pre-emphasis gain when using the PRE\_EMPHASIS attribute in a DDR4 application with an OUTPUT\_IMPEDANCE of 40Ω is listed in [Table 1-13](#).

**Table 1-13: Typical Pre-emphasis Gain when Using the PRE\_EMPHASIS Attribute in a DDR4 Application**

Attribute	Value	Estimated Gain (dB)
PRE_EMPHASIS (HP I/O banks) with an OUTPUT_IMPEDANCE of 40Ω.	RDRV_240 <sup>(1)</sup>	2.5

**Notes:**

1. ENABLE\_PRE\_EMPHASIS must be set to TRUE.

## LVDS Transmitter Pre-Emphasis

The LVDS transmitter pre-emphasis (LVDS\_PRE\_EMPHASIS) feature allows pre-emphasis on the drivers for certain I/O standards. This attribute must be used in conjunction with ENABLE\_PRE\_EMPHASIS.

The allowed values for the LVDS\_PRE\_EMPHASIS attribute are:

- LVDS\_PRE\_EMPHASIS = FALSE (Default)
- LVDS\_PRE\_EMPHASIS = TRUE (where ENABLE\_PRE\_EMPHASIS must be set to TRUE)

The LVDS\_PRE\_EMPHASIS attribute uses the following syntax in the XDC file:

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

A typical pre-emphasis gain when using the LVDS\_PRE\_EMPHASIS attribute is listed in [Table 1-14](#).

**Table 1-14: Typical Pre-Emphasis Gain when Using the LVDS\_PRE\_EMPHASIS Attribute**

Attribute	Value	Estimated Gain (dB)
LVDS_PRE_EMPHASIS (HP I/O banks)	TRUE <sup>(1)</sup>	4
LVDS_PRE_EMPHASIS (HR I/O banks)	TRUE <sup>(1)</sup>	4

**Notes:**

1. ENABLE\_PRE\_EMPHASIS must be set to TRUE.

## Receiver EQUALIZATION

The receiver equalization (EQUALIZATION) feature allows equalization at the receiver for certain I/O standards.

The allowed values for the EQUALIZATION attribute are:

### HP I/O Banks

- EQ\_LEVEL0
- EQ\_LEVEL1
- EQ\_LEVEL2
- EQ\_LEVEL3
- EQ\_LEVEL4
- EQ\_NONE (Default)

### HR I/O Banks

- EQ\_LEVEL0
- EQ\_LEVEL0\_DC\_BIAS
- EQ\_LEVEL1
- EQ\_LEVEL1\_DC\_BIAS
- EQ\_LEVEL2
- EQ\_LEVEL2\_DC\_BIAS
- EQ\_LEVEL3
- EQ\_LEVEL3\_DC\_BIAS
- EQ\_LEVEL4
- EQ\_LEVEL4\_DC\_BIAS
- EQ\_NONE (Default)



---

**IMPORTANT:** *The HR I/O banks with `_BIAS` equalization values cannot be combined in the same port with the `PULLTYPE` attribute set to `PULLUP`, `PULLDOWN`, or `KEEPER`.*

---

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

```
set_property EQUALIZATION value [get_ports port_name]
```

Typical AC gain for different values of EQUALIZATION for DDR4 and SGMII interfaces are listed in [Table 1-15](#).

**Table 1-15: Typical AC Gain for Different Values of Equalization in DDR4 and SGMII Interfaces**

Attribute	Value	Estimated Gain (dB)
Equalization in DDR4 interfaces at 2.66 Gb/s (HP I/O banks)	EQ_LEVEL0	0
	EQ_LEVEL1	0.75
	EQ_LEVEL2	1.50
	EQ_LEVEL3	2.25
	EQ_LEVEL4	3.00
Equalization in SGMII interfaces at 1.25 Gb/s (HR and HP I/O banks)	EQ_LEVEL0/EQ_LEVEL0_DC_BIAS	0
	EQ_LEVEL1/EQ_LEVEL1_DC_BIAS	1.50
	EQ_LEVEL2/EQ_LEVEL2_DC_BIAS	3.00
	EQ_LEVEL3/EQ_LEVEL3_DC_BIAS	4.50
	EQ_LEVEL4/EQ_LEVEL4_DC_BIAS	6.00

## Receiver OFFSET Control

The receiver offset control (OFFSET\_CNTRL) feature allows offset cancellation in receivers for certain I/O standards to overcome offset variations due to process.

The valid values for the OFFSET\_CNTRL attribute are:

- CNTRL\_NONE (Default)
- FABRIC

**Note:** OFFSET\_CNTRL = MEM\_CTRL is not a valid option.

The OFFSET\_CNTRL attribute uses the following syntax in the XDC file:

```
set_property OFFSET_CNTRL value [get_ports port_name]
```

To invoke the offset cancellation feature in an I/O bank, OFFSET\_CNTRL must be set to FABRIC. The controls for offset cancellation are available using the IBUFE3, IBUFDSE3, IOBUFE3, or IOBUFDSE3 primitives.

## VREF\_CNTR

VREF\_CNTR is an attribute specific to the receiver  $V_{REF}$  scan feature in HP I/O banks. It is used with the HPIO\_VREF UNISIM primitive.

The valid values for VREF\_CNTR attribute are:

- FABRIC\_RANGE1 (POD standards)

- FABRIC\_RANGE2 (other applicable standards)

FABRIC\_RANGE1 is used with the POD standards and the FABRIC\_RANGE2 is used with the other applicable standards when the receiver  $V_{REF}$  scan feature is invoked.

The VREF\_CNTR attribute should be set in the UNISIM instantiation. Refer to the *UltraScale Architecture Libraries Guide* (UG974) [Ref 6] for more information.

## DATA\_RATE

DATA\_RATE is an info-only attribute that is used by power analysis, timing analysis, and SSN tools in the Vivado Design Suite. This attribute provides information regarding the toggle rate of the I/O to these tools.

Valid values of this attribute are:

- Single data rate (SDR)
- Double data rate (DDR)

In non-native PHY applications, the default value of this attribute is SDR. In native mode applications (if the I/O is connected to one of the native PHY primitives such as IDDRE1, ODDRE1, RX\_BITSLICE, TX\_BITSLICE etc.), the default value is DDR.

The DATA\_RATE attribute uses the following syntax in the XDC file:

```
set_property DATA_RATE SDR|DDR [get_ports port_name]
```

## I/O Resource VHDL/Verilog Examples

The VHDL and Verilog example syntaxes for instantiating the I/O resources are found in the Vivado Design Suite HDL templates [Ref 12] [Ref 9].

## Supported I/O Standards and Terminations

The following sections provide an overview of the supported I/O standards and options. While most I/O supported standards specify a range of allowed voltages, this chapter records typical voltage values only. These standards are outlined in the Electronic Industry Alliance JEDEC® specification [Ref 8].

### LVTTL

Table 1-16: Available I/O Bank Type

HR	HP
Available	N/A

Low voltage TTL (LVTTTL) 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. This standard requires a 3.3V output source voltage ( $V_{CC0}$ ), but does not require the use of a reference voltage ( $V_{REF}$ ) or a termination voltage ( $V_{TT}$ ). This standard is defined by JEDEC (JESD 8C.01) [Ref 8].

Sample circuits illustrating both unidirectional and bidirectional LVTTTL termination techniques are shown in Figure 1-47 and Figure 1-48. These two diagrams show examples of source-series and parallel terminated topologies.

Figure 1-47 shows unidirectional terminated topologies.

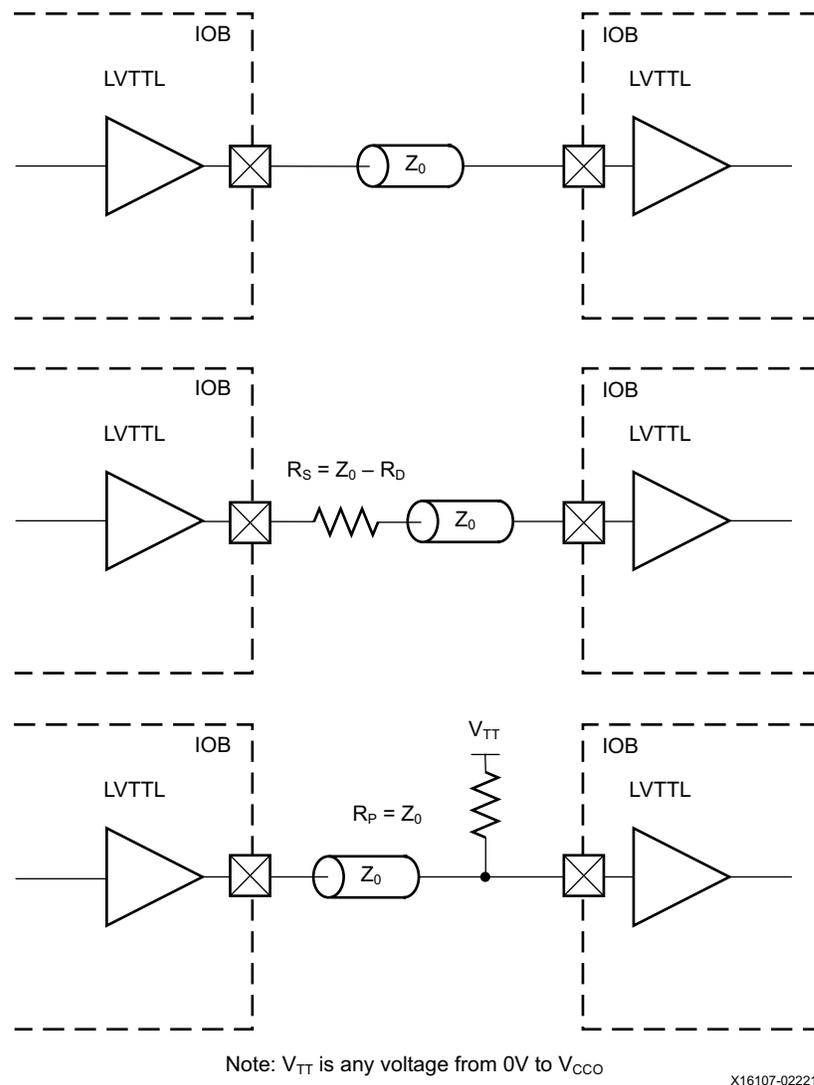
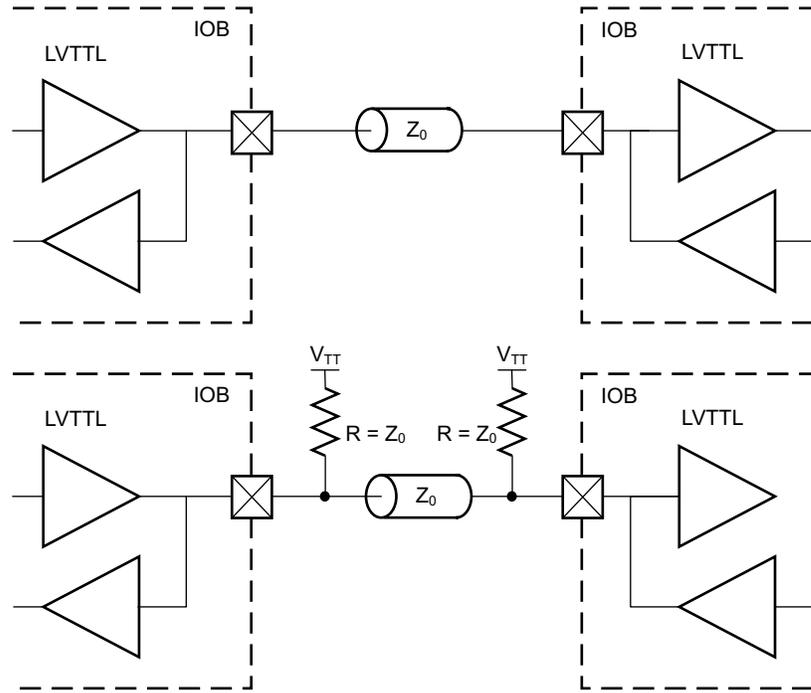


Figure 1-47: LVTTTL Unidirectional Termination

Figure 1-48 shows a bidirectional, parallel-terminated topology.



Note:  $V_{TT}$  is any voltage from 0V to  $V_{CC0}$

X16108-022216

Figure 1-48: LVTTTL Bidirectional Termination

Table 1-17 details the allowed attributes that can be applied to the LVTTTL I/O standard. This standard is only available in the HR I/O banks. Support is implied for primitives that are derivatives of the primitives listed in Table 1-17 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the SelectIO Interface Primitives section for all supported derivatives.

Table 1-17: Allowed Attributes for the LVTTTL I/O Standards

Attributes	Primitives		
	IBUF	OBUF/OBUFT/IOBUF	
		Allowed Values	Default
IOSTANDARD	LVTTTL	LVTTTL	
DRIVE	N/A	4, 8, 12, or 16	12
SLEW	N/A	FAST or SLOW	SLOW

## LVC MOS

Table 1-18: Available I/O Bank Type

HR	HP
Available	Available

Low voltage CMOS (LVC MOS) is a widely used switching standard implemented in CMOS transistors. This standard is defined by JESD8-12A.01, JESD8-11A.01, JESD8-7A, JESD8-5A.01, and JESD8C.01 [Ref 8]. The LVC MOS standards supported in UltraScale devices are: LVC MOS12, LVC MOS15, LVC MOS18, LVC MOS25, and LVC MOS33.

Sample circuits illustrating both unidirectional and bidirectional LVC MOS termination techniques are shown in Figure 1-49 and Figure 1-50. These two diagrams show examples of source-series and parallel terminated topologies.

Figure 1-49 shows unidirectional terminated topologies.

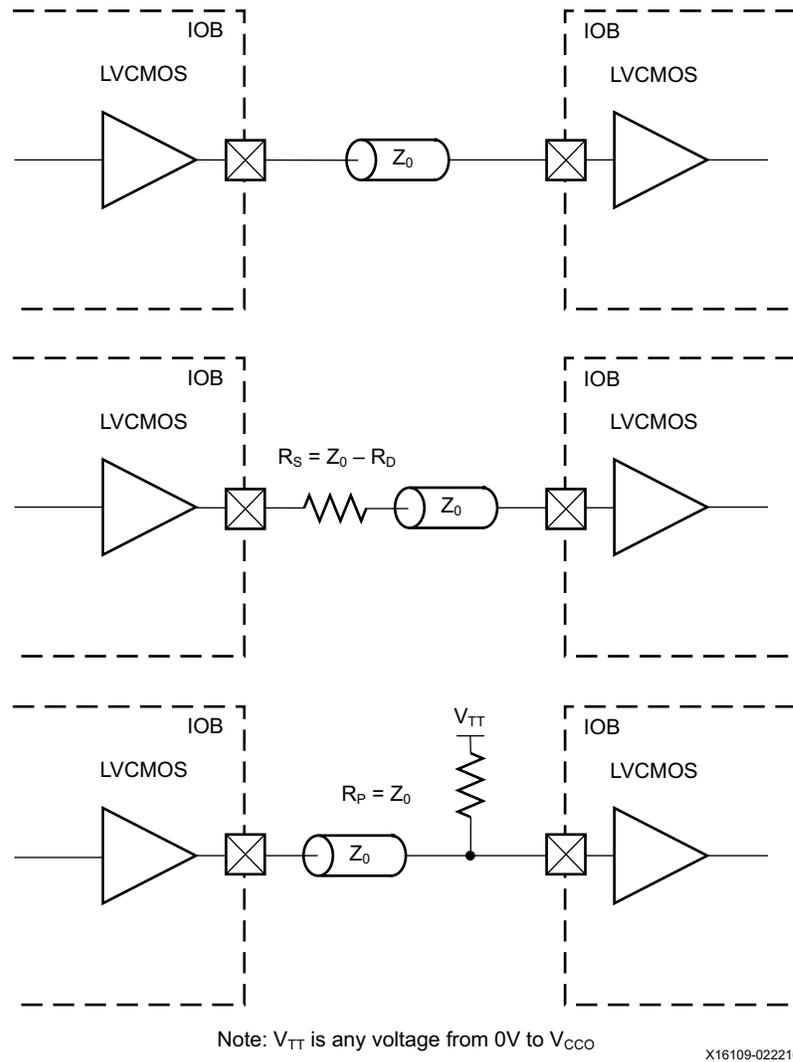


Figure 1-49: LVC MOS Unidirectional Termination

Figure 1-50 shows a bidirectional, parallel-terminated topology.

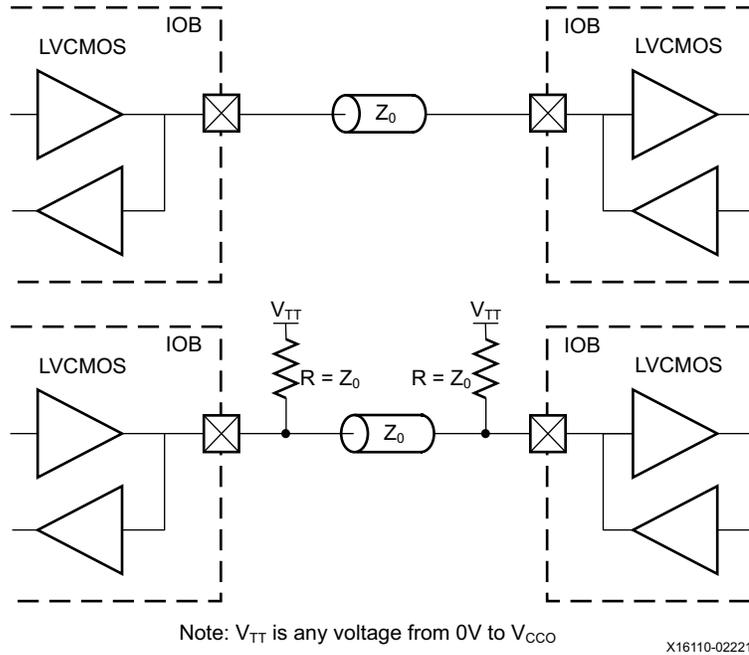


Figure 1-50: LVC MOS Bidirectional Termination

Table 1-19 details the allowed attributes that can be applied to the LVC MOS33 and LVC MOS25 I/O standards. These standards are only available in the HR I/O banks. Support is implied for primitives that are derivatives of the primitives listed in Table 1-19 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the SelectIO Interface Primitives section for all supported derivatives.

Table 1-19: Allowed Attributes for the LVC MOS33 and LVC MOS25 I/O Standards

Attributes	Primitives		
	IBUF	OBUF/OBUFT/IOBUF	
		Allowed Values	Default
IOSTANDARD	LVC MOS33, LVC MOS25	LVC MOS33, LVC MOS25	
DRIVE	N/A	4, 8, 12, or 16	12
SLEW	N/A	FAST or SLOW	SLOW

Table 1-20 details the allowed attributes that can be applied to the LVCMOS18 I/O standard. This standard is available in both the HR and HP I/O banks. For MOBILE DDR applications, the LVCMOS18 I/O standard is used with an 8 mA unterminated drive. Support is implied for primitives that are derivatives of the primitives listed in Table 1-20 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-20: Allowed Attributes for the LVCMOS18 I/O Standard

Attributes	Primitives				
	IBUF	OBUF/OBUFT/IOBUF			
		HP I/O Banks		HR I/O Banks	
		Allowed Values	Default	Allowed Values	Default
IOSTANDARD	LVCMOS18	LVCMOS18		LVCMOS18	
DRIVE	N/A	2, 4, 6, 8, 12	12	4, 8, 12, 16	12
SLEW	N/A	FAST, MEDIUM, SLOW	SLOW	FAST, SLOW	SLOW

Table 1-21 details the allowed attributes that can be applied to the LVCMOS15 I/O standard. This standard is available in both the HR and HP I/O banks. Support is implied for primitives that are derivatives of the primitives listed in Table 1-21 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-21: Allowed Attributes for the LVCMOS15 I/O Standard

Attributes	Primitives				
	IBUF	OBUF/OBUFT/IOBUF			
		HP I/O Banks		HR I/O Banks	
		Allowed Values	Default	Allowed Values	Default
IOSTANDARD	LVCMOS15	LVCMOS15		LVCMOS15	
DRIVE	N/A	2, 4, 6, 8, 12	12	4, 8, 12, 16	12
SLEW	N/A	FAST, MEDIUM, SLOW	SLOW	FAST, SLOW	SLOW

Table 1-22 details the allowed attributes that can be applied to the LVCMOS12 I/O standard. This standard is available in both the HR and HP I/O banks. Support is implied for primitives that are derivatives of the primitives listed in Table 1-22 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-22: Allowed Attributes for the LVCMOS12 I/O Standard

Attributes	Primitives				
	IBUF	OBUF/OBUFT/IOBUF			
		HP I/O Banks		HR I/O Banks	
		Allowed Values	Default	Allowed Values	Default
IOSTANDARD	LVCMOS12	LVCMOS12		LVCMOS12	
DRIVE	N/A	2, 4, 6, 8	12 <sup>(1)</sup>	4, 8, 12	12
SLEW	N/A	FAST, MEDIUM, SLOW	SLOW	FAST, SLOW	SLOW

**Notes:**

1. Change the drive setting from the default setting in the RTL or XDC file to one of the allowed values before running through the Vivado Design Suite.

## LVDCI

Table 1-23: Available I/O Bank Type

HR	HP
N/A	Available

Using these I/O buffers configures the outputs as controlled impedance drivers. The receiver of low-voltage digitally controlled impedance (LVDCI) is identical to an LVCMOS receiver. Some I/O standards, such as LVCMOS, must have a drive impedance that matches the characteristic impedance of the driven line. The HP I/O banks provide a controlled impedance output driver to provide series termination without external-source termination resistors.

Source termination is controlled using the OUTPUT\_IMPEDANCE attribute. The exact value of the impedance is determined by the OUTPUT\_IMPEDANCE attribute and an external 240Ω resistor on the VRP pin. The only valid value of this attribute for LVDCI standards is RDRV\_48\_48, which corresponds to a 48Ω setting.

Sample circuits illustrating both unidirectional and bidirectional topologies for a controlled impedance driver are shown in [Figure 1-51](#) and [Figure 1-52](#). The DCI I/O standards supporting a controlled impedance driver are: LVDCI\_15 and LVDCI\_18.

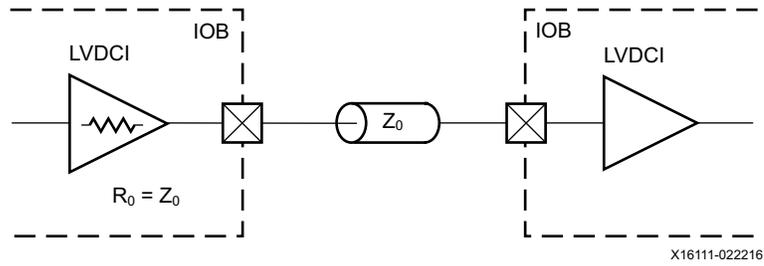


Figure 1-51: Unidirectional Controlled Impedance Driver Topology

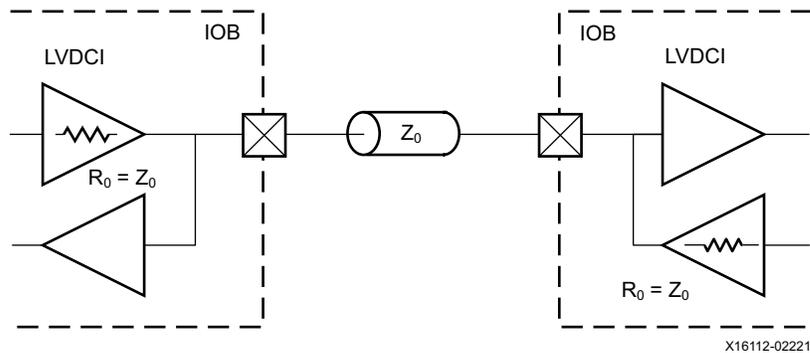


Figure 1-52: Bidirectional Controlled Impedance Driver Topology

Table 1-24 details the allowed attributes that can be applied to the LVDCI I/O standard. This standard is available in the HP I/O banks. Support is implied for primitives that are derivatives of the primitives listed in Table 1-24 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-24: Allowed Attributes for the LVDCI I/O Standards

Attributes	Primitives		
	IBUF	OBUF/OBUFT/IOBUF	
		Allowed Values	Default
IOSTANDARD	LVDCI_15, LVDCI_18	LVDCI_15, LVDCI_18	
SLEW	N/A	FAST, MEDIUM, SLOW	SLOW
OUTPUT_IMPEDANCE	N/A	RDRV_48_48	

## HSLVDCI

Table 1-25: Available I/O Bank Type

HR	HP
N/A	Available

The driver is identical to LVDCI, while the input is identical to HSTL and SSTL. By using a  $V_{REF}$ -referenced input, high-speed LVDCI (HSLVDCI) allows greater input sensitivity at the receiver than when using a single-ended LVCMOS-type receiver.

The HP I/O banks have a controlled impedance output driver to provide series termination without external-source termination resistors. The exact value of the impedance is set by the OUTPUT\_IMPEDANCE attribute and an external  $240\Omega$  resistor on the VRP pin. The only valid value of the OUTPUT\_IMPEDANCE attribute for HSLVDCI standards is RDRV\_48\_48, which corresponds to a  $48\Omega$  setting.

A sample circuit illustrating bidirectional termination techniques for an HSLVDCI controlled impedance driver is shown in Figure 1-53. The DCI I/O standards supporting a controlled impedance driver with a  $V_{REF}$  referenced input are: HSLVDCI\_15 and HSLVDCI\_18.

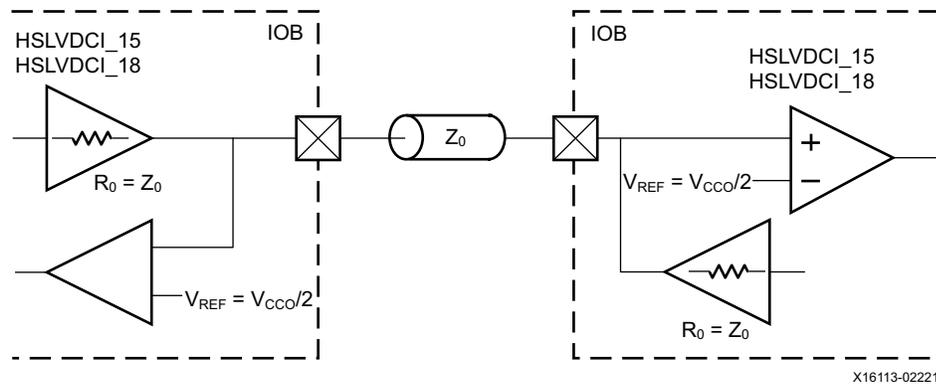


Figure 1-53: HSLVDCI Controlled Impedance Driver with Bidirectional Termination

For electrical specifications, see the LVDCI  $V_{OH}$  and  $V_{OL}$  entries in the UltraScale device data sheets [Ref 2].

Table 1-26 details the allowed attributes that can be applied to the HSLVDCI I/O standard. This standard is available in the HP I/O banks. Support is implied for primitives that are derivatives of the primitives listed in Table 1-26 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the SelectIO Interface Primitives section for all supported derivatives.

Table 1-26: Allowed Attributes for the HSLVDCI I/O Standards

Attributes	Primitives		
	IBUF	OBUF/OBUFT/IOBUF	
		Allowed Values	Default
IOSTANDARD	HSLVDCI_15, HSLVDCI_18	HSLVDCI_15, HSLVDCI_18	
SLEW	N/A	FAST, MEDIUM, SLOW	SLOW
OUTPUT_IMPEDANCE	N/A	RDRV_48_48	

## HSTL

The high-speed transceiver logic (HSTL) standard is a general purpose high-speed bus standard is defined by JEDEC (JESD8-6) [Ref 8]. To support clocking high-speed memory interfaces, differential versions are also available. The UltraScale architecture I/O supports class-I for the 1.2V version (in HP I/O banks) along with the 1.5V version and 1.8V versions (both HP and HR I/O banks), including the differential versions. Class-II supports the 1.5V version and 1.8V version (in HR I/O banks), including the differential versions. The differential versions of the standard require a differential amplifier input buffer and a push-pull output buffer. The HP I/O banks also support DCI versions.

### ***HSTL\_I and HSTL\_I\_18***

Table 1-27: Available I/O Bank Type

HR	HP
Available	Available

HSTL\_I and HSTL\_I\_18 use  $V_{CCO}/2$  as a parallel-termination voltage ( $V_{TT}$ ).

Optional untuned split input ODT provides Thevenin equivalent resistance of R (where  $R = Z_0$ ) to the  $V_{CCO}/2$ . The untuned on-die source termination feature (OUTPUT\_IMPEDANCE) with optional 40Ω, 48Ω, or 60Ω of driver impedance is available in HP I/O banks. The default value of the driver output impedance is 48Ω.

### ***HSTL\_I\_12***

Table 1-28: Available I/O Bank Type

HR	HP
N/A	Available

HSTL\_I\_12 uses  $V_{CCO}/2$  as a parallel-termination voltage ( $V_{TT}$ ).

Optional untuned split input ODT provides Thevenin equivalent resistance of R (where  $R = Z_0$ ) to the  $V_{CCO}/2$ . The untuned on-die source termination feature (OUTPUT\_IMPEDANCE) with optional 40Ω, 48Ω, or 60Ω of driver impedance is available in HP I/O banks. The default value of the driver output impedance is 48Ω.

### ***HSTL\_I\_DCI, HSTL\_I\_DCI\_12, and HSTL\_I\_DCI\_18***

Table 1-29: Available I/O Bank Type

HR	HP
N/A	Available (HSTL_I_DCI_12 in UltraScale devices only)

In UltraScale and UltraScale+ devices, HSTL\_I\_DCI and HSTL\_I\_DCI\_18 provide on-chip split-Thevenin termination powered from  $V_{CCO}$ , using the ODT attribute, creating an equivalent parallel-termination voltage ( $V_{TT}$ ) of  $V_{CCO}/2$ . HSTL\_I\_DCI\_12 provides on-chip split-Thevenin termination in UltraScale devices. HSTL\_I\_DCI\_12 does not provide input split termination in UltraScale+ devices. STL12\_DCI provides equivalent functionality.

The source termination feature (OUTPUT\_IMPEDANCE) provides the option of  $40\Omega$ ,  $48\Omega$ , or  $60\Omega$  of tuned driver impedance in HP I/O banks. The default value of the driver output impedance is  $48\Omega$ .

### ***HSTL\_II and HSTL\_II\_18***

**Table 1-30: Available I/O Bank Type**

HR	HP
Available	Not Available

HSTL\_II and HSTL\_II\_18 use  $V_{CCO}/2$  as a parallel-termination voltage ( $V_{TT}$ ).

Optional untuned split input ODT provides Thevenin equivalent resistance of  $R$  (where  $R = Z_0$ ) to the  $V_{CCO}/2$ .

### ***DIFF\_HSTL\_I and DIFF\_HSTL\_I\_18***

**Table 1-31: Available I/O Bank Type**

HR	HP
Available	Available

Differential HSTL class-I pairs complementary single-ended HSTL\_I type drivers with a differential receiver.

Optional untuned split input ODT provides Thevenin equivalent resistance of  $R$  (where  $R = Z_0$ ) to the  $V_{CCO}/2$ . The untuned on-die source termination feature (OUTPUT\_IMPEDANCE) with optional  $40\Omega$ ,  $48\Omega$ , or  $60\Omega$  of driver impedance is available in HP I/O banks. The default value of the driver output impedance is  $48\Omega$ .

### ***DIFF\_HSTL\_I\_DCI and DIFF\_HSTL\_I\_DCI\_18***

**Table 1-32: Available I/O Bank Type**

HR	HP
N/A	Available

Differential HSTL class-I pairs complementary single-ended HSTL\_I type drivers with a differential receiver, including on-chip split-Thevenin termination using the ODT attribute.

The source termination feature (OUTPUT\_IMPEDANCE) provides the option of 40Ω, 48Ω, or 60Ω of tuned driver impedance in HP I/O banks. The default value of the driver output impedance is 48Ω.

### ***DIFF\_HSTL\_II and DIFF\_HSTL\_II\_18***

**Table 1-33: Available I/O Bank Type**

HR	HP
Available	Not Available

Differential HSTL class-II pairs complementary single-ended HSTL\_II type drivers with a differential receiver. Differential HSTL can also be used for differential clock and DQS signals in memory interface designs.

Optional untuned split input ODT provides Thevenin equivalent resistance of R (where  $R = Z_0$ ) to the  $V_{CCO}/2$ .

### ***DIFF\_HSTL\_I\_12***

**Table 1-34: Available I/O Bank Type**

HR	HP
Not Available	Available

Differential HSTL class-I pairs complementary single-ended HSTL\_I\_12 type drivers with a differential receiver.

Optional untuned split input ODT provides Thevenin equivalent resistance of R (where  $R = Z_0$ ) to the  $V_{CCO}/2$ . The untuned on-die source termination feature (OUTPUT\_IMPEDANCE) with optional 40Ω, 48Ω, or 60Ω of driver impedance is available in HP I/O banks. The default value of the driver output impedance is 48Ω.

### ***DIFF\_HSTL\_I\_12\_DCI***

**Table 1-35: Available I/O Bank Type**

HR	HP
Not Available	Available

Differential HSTL class-I pairs complementary single-ended HSTL\_I\_12 type drivers with a differential receiver, including on-chip split-Thevenin termination using the ODT attribute. The source termination feature (OUTPUT\_IMPEDANCE) provides the option of 40Ω, 48Ω, or 60Ω of tuned driver impedance in HP I/O banks. The default value of the driver output impedance is 48Ω.

## HSTL Class I (1.2V, 1.5V, or 1.8V)

Figure 1-54 shows a sample circuit illustrating a termination technique for HSTL class-I for the 1.2V, 1.5V, or 1.8V versions. In a specific circuit, all drivers and receivers must be at the same voltage level (1.2V, 1.5V, or 1.8V); they are not interchangeable (i.e., HSTL\_I\_12 should only interface with HSTL\_I\_12). Only HP I/O banks support the DCI standards.

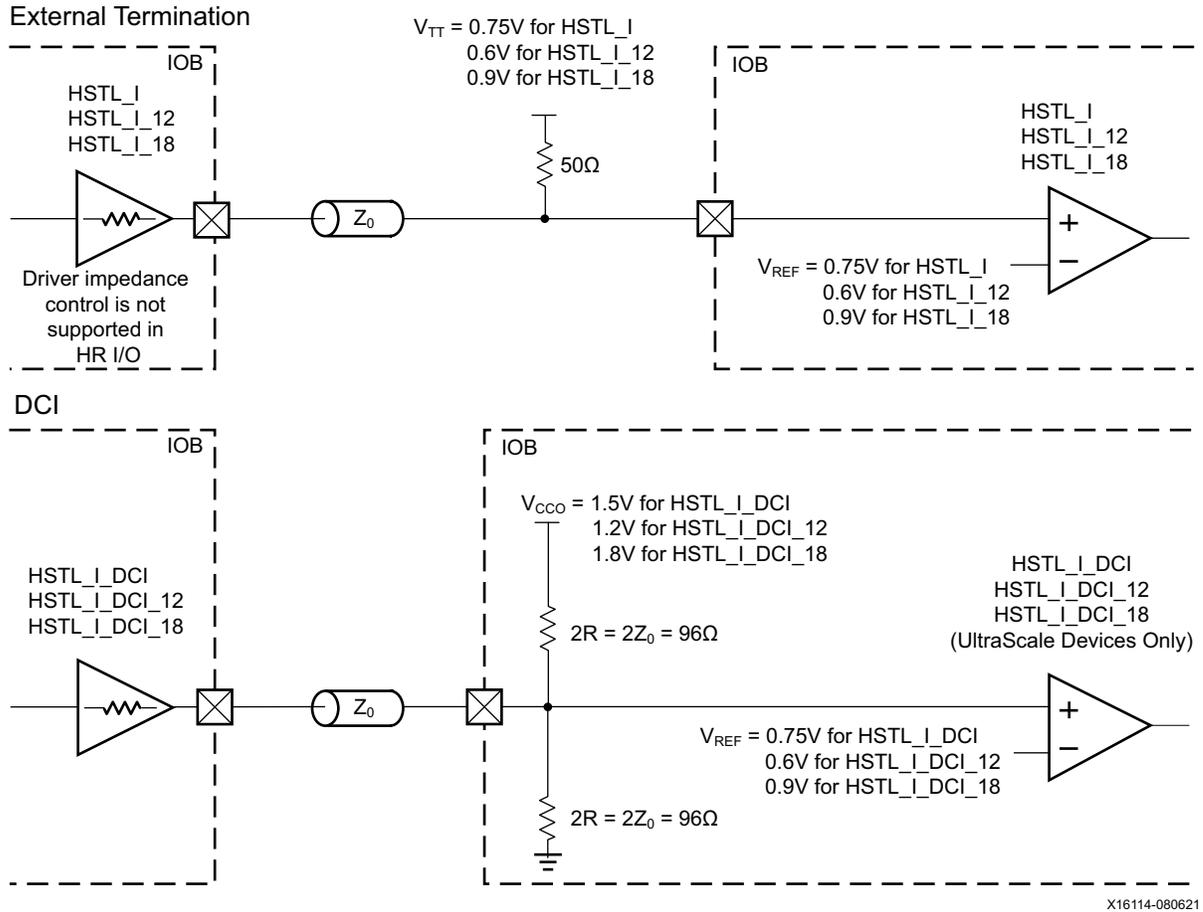


Figure 1-54: HSTL Class I (1.2V, 1.5V, or 1.8V) Unidirectional Termination

X16114-080621

Figure 1-55 shows a sample circuit illustrating a termination technique for HSTL class-I for the 1.2V, 1.5V, or 1.8V versions in a bidirectional configuration. In a specific circuit, all drivers and receivers must be at the same voltage level (1.2V, 1.5V, or 1.8V); they are not interchangeable (i.e., HSTL\_I\_18 should only interface with HSTL\_I\_18). Only HP I/O banks support the DCI standards.

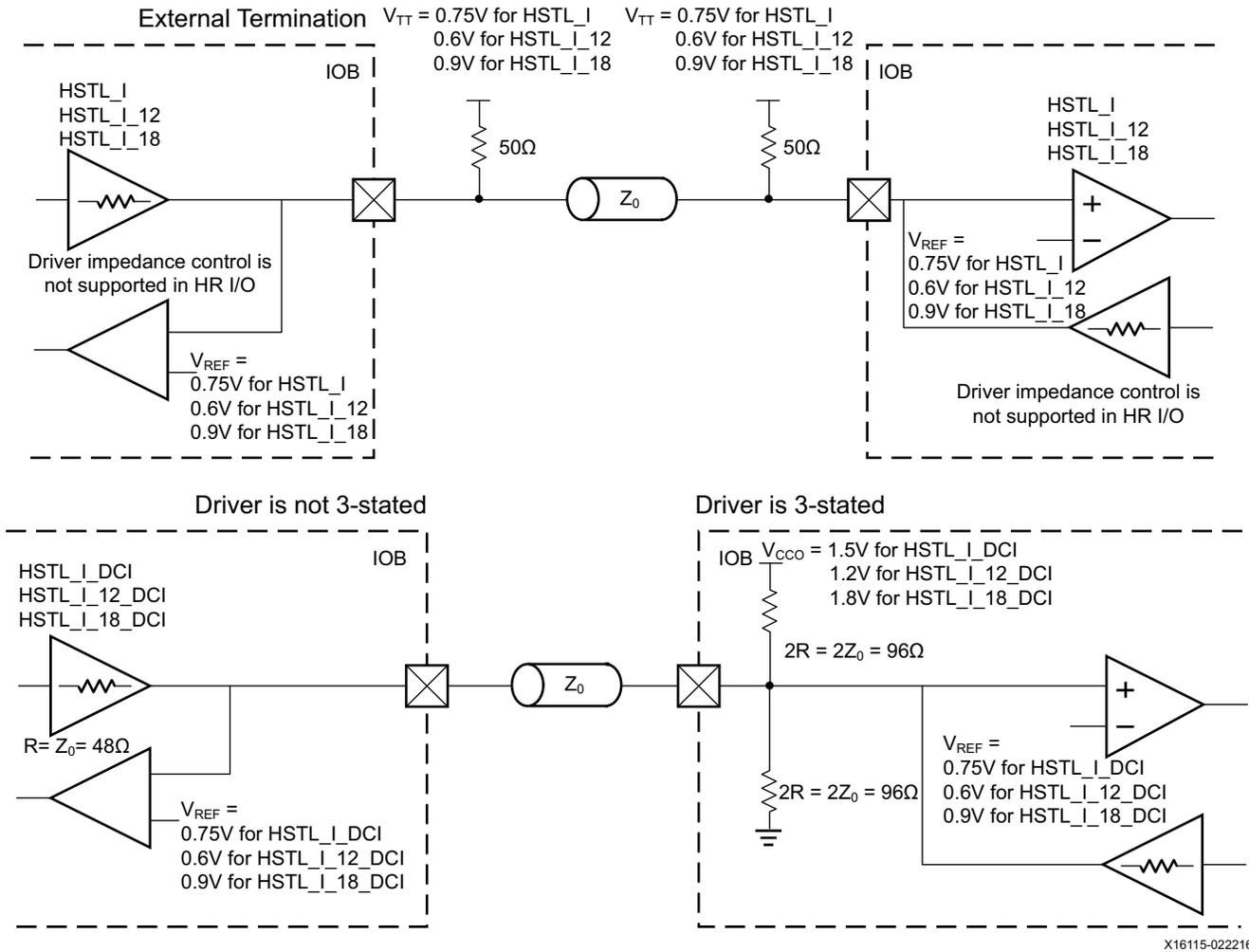


Figure 1-55: HSTL Class I (1.2V, 1.5V, or 1.8V) Bidirectional Termination

X16115-022216

## Differential HSTL Class I

Figure 1-56 shows a sample circuit illustrating a termination technique for differential HSTL class-I (1.2V, 1.5V, or 1.8V) with unidirectional termination. In a specific circuit, all drivers and receivers must be at the same voltage level (1.2V, 1.5V, or 1.8V); they are not interchangeable (i.e., HDIFF\_HSTL\_I\_18 should only interface with HSTL\_I\_18).

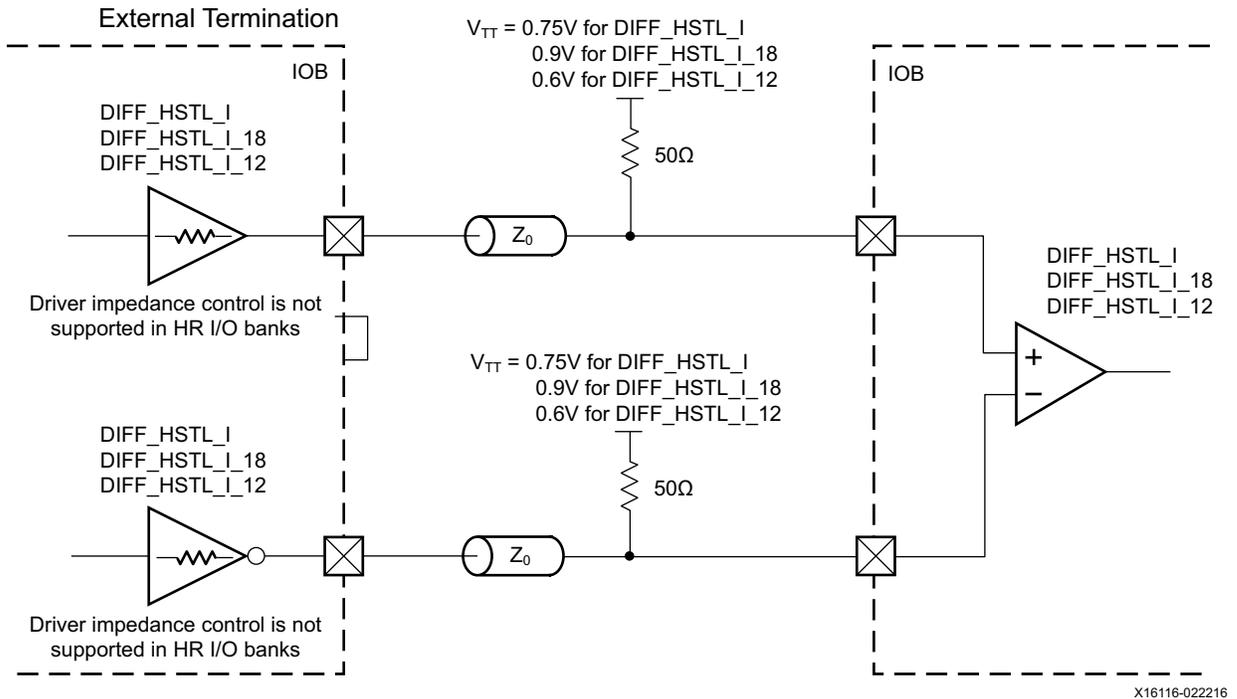


Figure 1-56: Differential HSTL Class I (1.2V, 1.5V, or 1.8V) Unidirectional Termination



Figure 1-58 shows a sample circuit illustrating a termination technique for differential HSTL class-I (1.2V, 1.5V, or 1.8V) with unidirectional DCI termination. In a specific circuit, all drivers and receivers must be at the same voltage level (1.2V, 1.5V, or 1.8V); they are not interchangeable (i.e., DIFF\_HSTL\_I\_DCI should only interface with DIFF\_HSTL\_I\_DCI). Only HP I/O banks support these DCI standards.

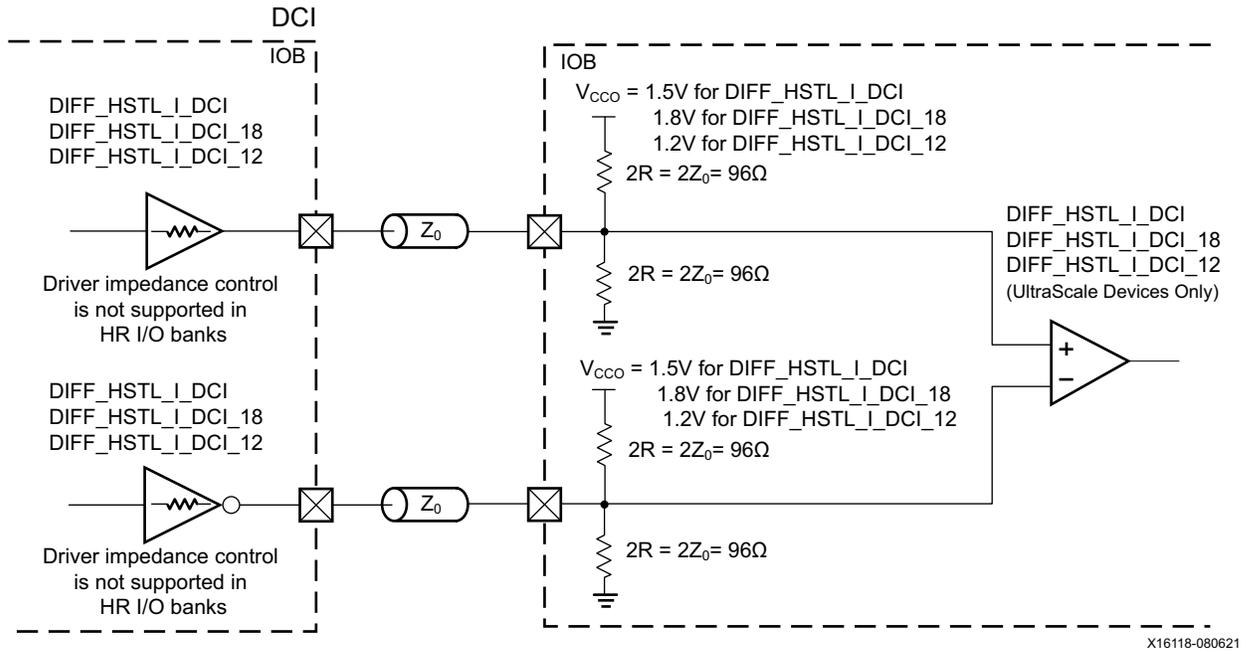
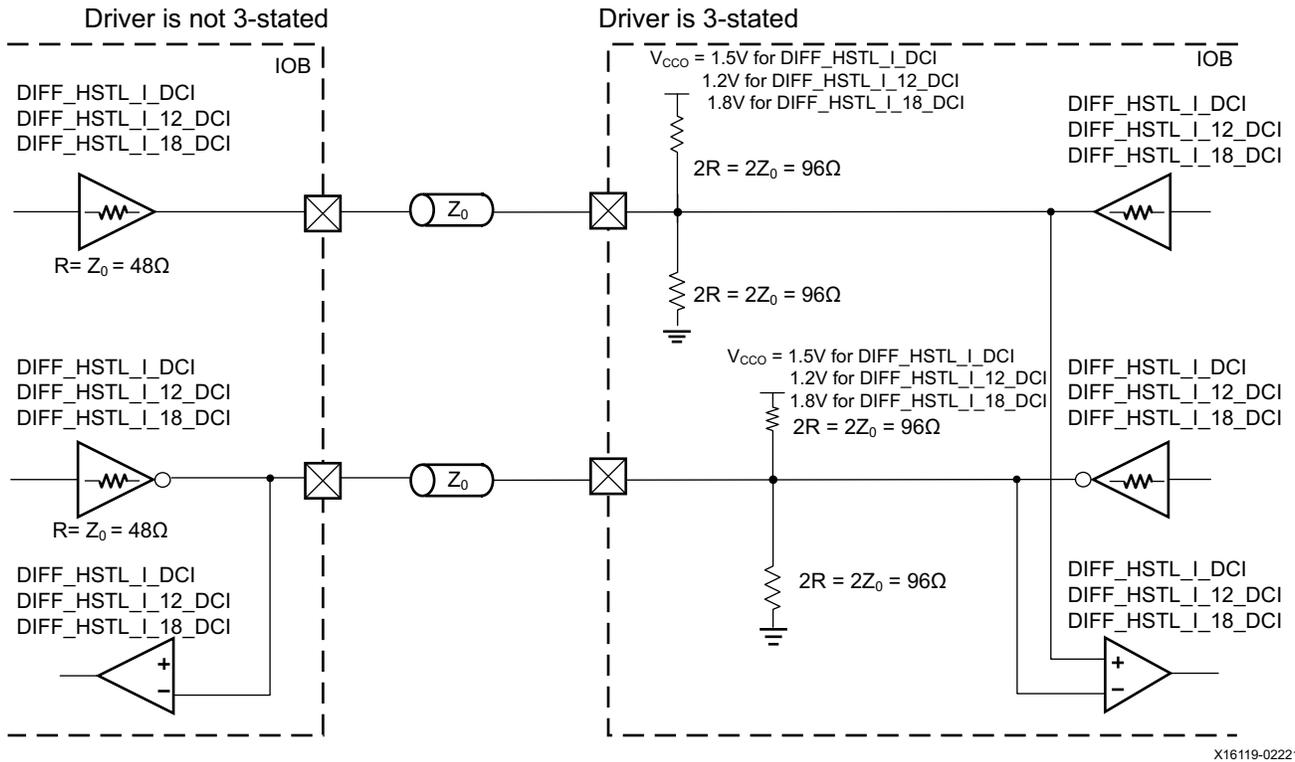


Figure 1-58: Differential HSTL Class I (1.2V, 1.5V, or 1.8V) DCI Unidirectional Termination

Figure 1-59 shows a sample circuit illustrating a termination technique for differential HSTL class-I (1.2V, 1.5V, or 1.8V) with bidirectional DCI termination. In a specific circuit, all drivers and receivers must be at the same voltage level (1.2V, 1.5V, or 1.8V); they are not interchangeable (i.e., DIFF\_HSTL\_I\_DCI should only interface with DIFF\_HSTL\_I\_DCI). Only HP I/O banks support these DCI standards.



X16119-022216

Figure 1-59: Differential HSTL Class I (1.2V, 1.5V, or 1.8V) DCI Bidirectional Termination

## HSTL Class II

Figure 1-60 shows a sample circuit illustrating a termination technique for HSTL class-II (1.5V or 1.8V) with unidirectional termination. In a specific circuit, all drivers and receivers must be at the same voltage level (1.5V or 1.8V); they are not interchangeable (i.e., HSTL\_II\_18 should only interface with HSTL\_II\_18). Only HR I/O banks support the class-II standards.

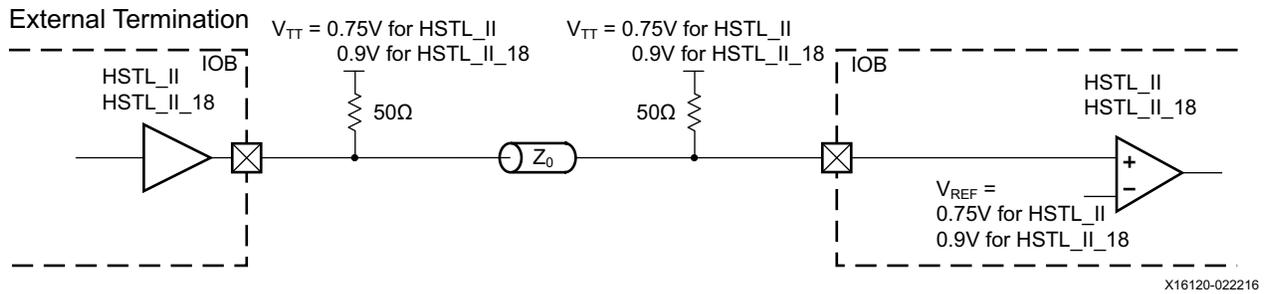


Figure 1-60: HSTL Class II (1.5V or 1.8V) Unidirectional Termination

Figure 1-61 shows a sample circuit illustrating a termination technique for HSTL class-II (1.5V or 1.8V) with bidirectional termination. In a specific circuit, all drivers and receivers must be at the same voltage level (1.5V or 1.8V); they are not interchangeable (i.e., HSTL\_II\_18 should only interface with HSTL\_II\_18).

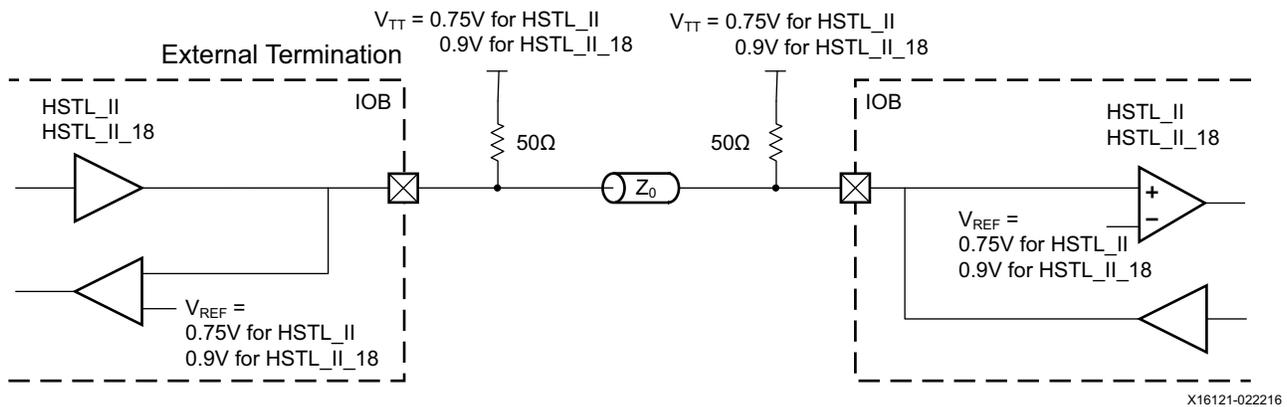


Figure 1-61: HSTL Class II (1.5V or 1.8V) Bidirectional Termination

## Differential HSTL Class II

Figure 1-62 shows a sample circuit illustrating a termination technique for differential HSTL (1.5V or 1.8V) with unidirectional termination. In a specific circuit, all drivers and receivers must be at the same voltage level (1.5V or 1.8V); they are not interchangeable (i.e., DIFF\_HSTL\_II\_18 should only interface with DIFF\_HSTL\_II\_18). Only HR I/O banks support the class-II standards (i.e., DIFF\_HSTL\_II\_18 should only interface with DIFF\_HSTL\_II\_18).

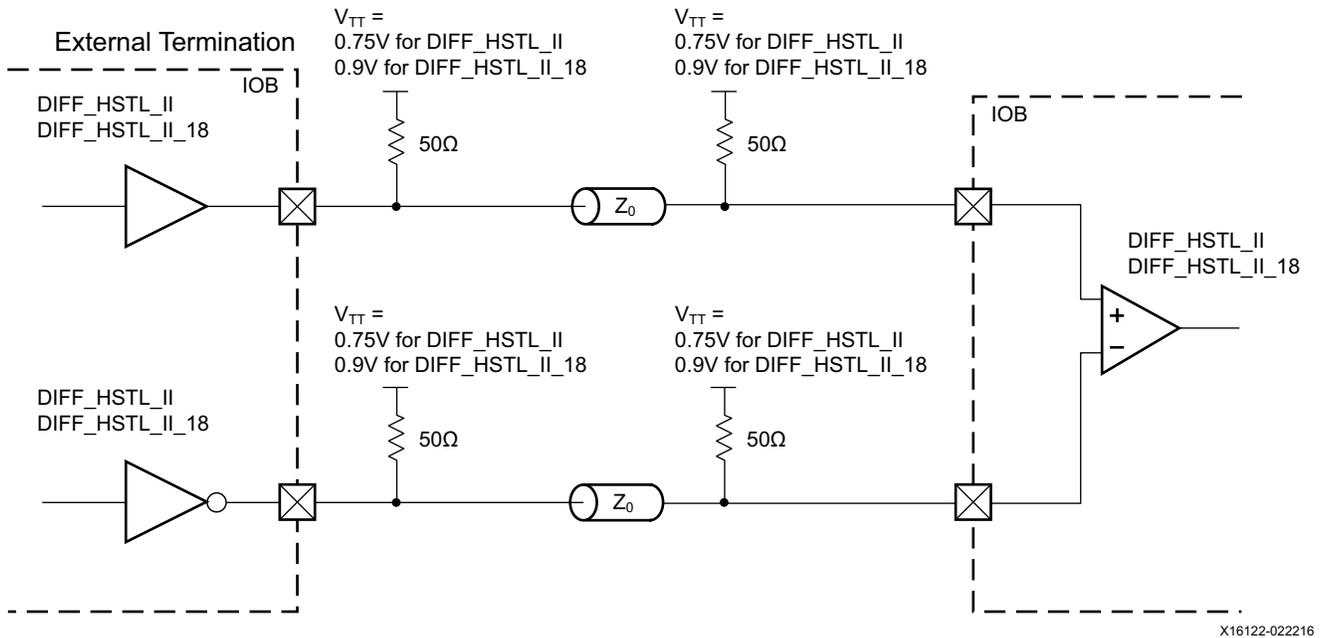
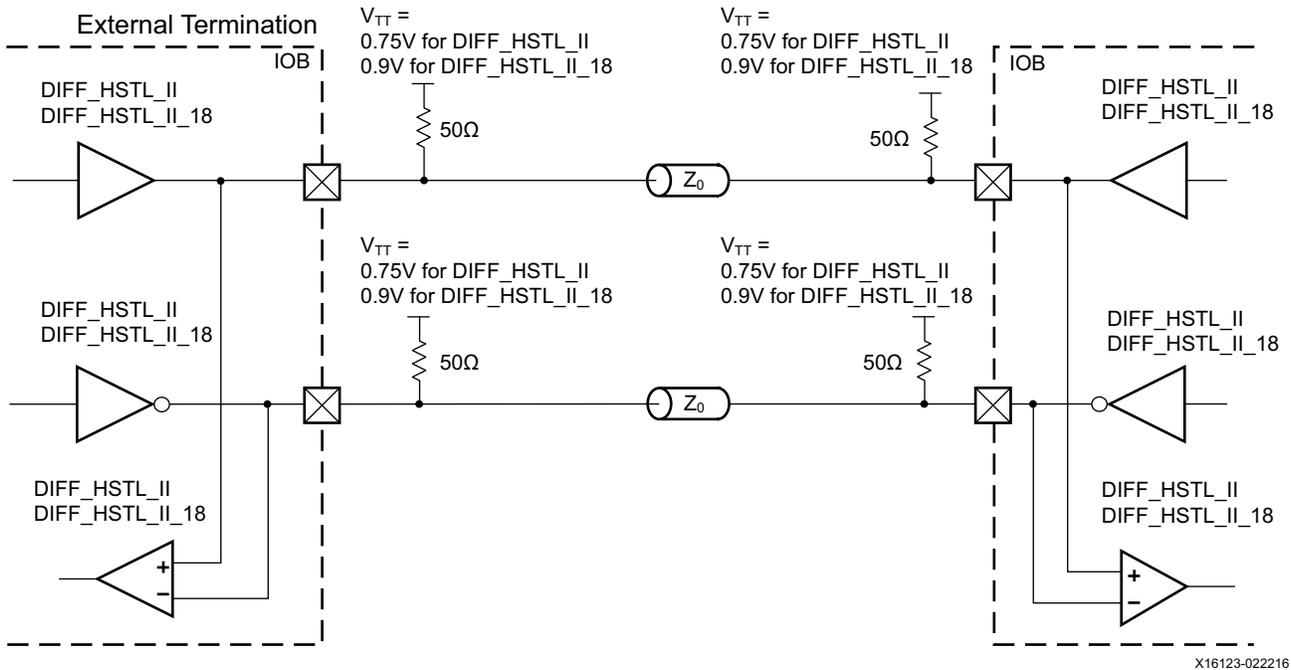


Figure 1-62: Differential HSTL (1.5V or 1.8V) Unidirectional Termination

Figure 1-63 shows a sample circuit illustrating a termination technique for differential HSTL class-II (1.5V or 1.8V) with bidirectional termination. In a specific circuit, all drivers and receivers must be at the same voltage level (1.5V or 1.8V); they are not interchangeable (i.e., DIFF\_HSTL\_II\_18 should only interface with DIFF\_HSTL\_II\_18).



X16123-022216

Figure 1-63: Differential HSTL Class II (1.5V or 1.8V) Bidirectional Termination

Table 1-36 and Table 1-37 list the supported attributes for the HSTL I/O standards. Support is implied for primitives that are derivatives of the primitives listed in these tables (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-36: HSTL Class I Allowed Attributes

Attributes	IBUF/IBUFE3/IBUFDS/IBUFDSE3				OBUF/OBUFT				IOBUF/IOBUFE3/IOBUFDS/IOBUFDSE3			
	HP I/O		HR I/O		HP I/O		HR I/O		HP I/O		HR I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	HSTL_I HSTL_I_12 HSTL_I_18		HSTL_I HSTL_I_18		HSTL_I HSTL_I_12 HSTL_I_18		HSTL_I HSTL_I_18		HSTL_I HSTL_I_12 HSTL_I_18		HSTL_I HSTL_I_18	
SLEW	N/A		N/A		FAST MEDIUM SLOW	SLOW	FAST SLOW	SLOW	FAST MEDIUM SLOW	SLOW	FAST SLOW	SLOW
ODT	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE	N/A		N/A		RTT_40 RTT_48 RTT_60 RTT_NONE <sup>(1)</sup>	RTT_NONE	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE
OUTPUT_IMPEDANCE	N/A		N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60	RDRV_48_48	N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60 <sup>(1)</sup>	RDRV_48_48	N/A	
IOSTANDARD	HSTL_I_DCI HSTL_I_DCI_12 HSTL_I_DCI_18		N/A		HSTL_I_DCI HSTL_I_DCI_12 HSTL_I_DCI_18		N/A		HSTL_I_DCI HSTL_I_DCI_12 HSTL_I_DCI_18		N/A	
SLEW	N/A		N/A		FAST MEDIUM SLOW	SLOW	N/A		FAST MEDIUM SLOW	SLOW	N/A	
ODT	RTT_40 RTT_48 RTT_60 <sup>(2)</sup>	RTT_48	N/A		N/A		N/A		RTT_40 RTT_48 RTT_60 <sup>(1)(2)</sup>	RTT_48	N/A	
OUTPUT_IMPEDANCE	N/A		N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60	RDRV_48_48	N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60 <sup>(1)</sup>	RDRV_48_48	N/A	

Table 1-36: HSTL Class I Allowed Attributes (Cont'd)

Attributes	IBUF/IBUFE3/IBUFDS/IBUFDSE3				OBUF/OBUFT				IOBUF/IOBUFE3/IOBUFDS/IOBUFDSE3			
	HP I/O		HR I/O		HP I/O		HR I/O		HP I/O		HR I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	DIFF_HSTL_I DIFF_HSTL_I_12 DIFF_HSTL_I_18		DIFF_HSTL_I DIFF_HSTL_I_18		DIFF_HSTL_I DIFF_HSTL_I_12 DIFF_HSTL_I_18		DIFF_HSTL_I DIFF_HSTL_I_18		DIFF_HSTL_I DIFF_HSTL_I_12 DIFF_HSTL_I_18		DIFF_HSTL_I DIFF_HSTL_I_18	
SLEW	N/A		N/A		FAST MEDIUM SLOW	SLOW	FAST SLOW	SLOW	FAST MEDIUM SLOW	SLOW	FAST SLOW	SLOW
ODT	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE	N/A		N/A		RTT_40 RTT_48 RTT_60 RTT_NONE <sup>(1)</sup>	RTT_NONE	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE
OUTPUT_IMPEDANCE	N/A		N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60	RDRV_48_48	N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60 <sup>(1)</sup>	RDRV_48_48	N/A	
IOSTANDARD	DIFF_HSTL_I_DCI DIFF_HSTL_I_DCI_12 DIFF_HSTL_I_DCI_18		N/A		DIFF_HSTL_I_DCI DIFF_HSTL_I_DCI_12 DIFF_HSTL_I_DCI_18		N/A		DIFF_HSTL_I_DCI DIFF_HSTL_I_DCI_12 DIFF_HSTL_I_DCI_18		N/A	
SLEW	N/A		N/A		FAST MEDIUM SLOW	SLOW	N/A		FAST MEDIUM SLOW	SLOW	N/A	
ODT	RTT_40 RTT_48 RTT_60 <sup>(2)</sup>	RTT_48	N/A		N/A		N/A		RTT_40 RTT_48 RTT_60 <sup>(1)(2)</sup>	RTT_48	N/A	
OUTPUT_IMPEDANCE	N/A		N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60	RDRV_48_48	N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60 <sup>(1)</sup>	RDRV_48_48	N/A	

**Notes:**

1. The allowed bidirectional configuration combinations for driver output impedance (OUTPUT\_IMPEDANCE) and ODT are listed in [Table 1-37](#).
2. ODT = RTT\_NONE is not a valid setting for DCI I/O standards.

Table 1-37: Only Allowed Combinations for Bidirectional Configurations

OUTPUT_IMPEDANCE	ODT
RDRV_40_40 (40Ω)	RTT_40
RDRV_40_40 (40Ω)	RTT_60
RDRV_40_40 (40Ω)	RTT_NONE
RDRV_48_48 (48Ω)	RTT_48
RDRV_48_48 (48Ω)	RTT_NONE
RDRV_60_60 (60Ω)	RTT_40
RDRV_60_60 (60Ω)	RTT_60
RDRV_60_60 (60Ω)	RTT_NONE

Table 1-38 lists the supported attributes for the HSTL Class-II I/O standards. Support is implied for primitives that are derivatives of the primitives listed in Table 1-38 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-38: HSTL Class II Allowed Attributes

Attributes	IBUF/IBUFDS			OBUF/OBUFT			IOBUF/IOBUFD		
	HP I/O	HR I/O		HP I/O	HR I/O		HP I/O	HR I/O	
		Allowed Values	Default		Allowed Values	Default		Allowed Values	Default
IOSTANDARD	N/A	HSTL_II HSTL_II_18		N/A	HSTL_II HSTL_II_18		N/A	HSTL_II HSTL_II_18	
SLEW	N/A	N/A		N/A	FAST SLOW	SLOW	N/A	FAST SLOW	SLOW
ODT	N/A	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE	N/A	N/A		N/A	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE
IOSTANDARD	N/A	DIFF_HSTL_II DIFF_HSTL_II_18		N/A	DIFF_HSTL_II DIFF_HSTL_II_18		N/A	DIFF_HSTL_II DIFF_HSTL_II_18	
SLEW	N/A	N/A		N/A	FAST SLOW	SLOW	N/A	FAST SLOW	SLOW
ODT	N/A	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE	N/A	N/A		N/A	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE

## SSTL

The stub-series terminated logic (SSTL) for 1.8V (SSTL18), 1.5V (SSTL15), and 1.35V (SSTL135) are I/O standards used for general-purpose memory buses.

While example termination techniques are discussed in this section, the optimal termination schemes for a given memory interface are determined using signal-integrity analysis of the actual PCB topology including the memory devices used, the board layout, and transmission line impedances. AMD provides both IBIS model files and encrypted HSPICE model files for all of the I/O standards. These SSTL standards are supported for both single-ended signaling and differential signaling. The differential versions use a true differential amplifier input buffer and complementary push-pull output buffers. The DCI versions of these standards are the preferred I/O standards to use for memory interfaces implemented in the HP I/O banks. The uncalibrated split termination (using the ODT attributes) is recommended for interfaces implemented without the DCI standards.

SSTL18 is defined by the JEDEC standard JESD8-15 [Ref 8], 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.

SSTL18 class-I is available in both the HP and HR I/O banks. Both HP and HR I/O banks provide ODT attributes for untuned internal parallel split-termination resistors for the non-DCI versions of these standards. In addition, the source termination feature (OUTPUT\_IMPEDANCE) provides the option of 40 $\Omega$ , 48 $\Omega$ , or 60 $\Omega$  tuned driver impedance in HP I/O banks in both DCI and non-DCI versions. The driver output impedance is set to a default of 40 $\Omega$ . The optimal drive and termination scheme for any new design is determined through careful signal-integrity analysis. SSTL18 class-II is available in HR I/O banks. HR I/O banks provide the option of ODT attributes for untuned internal parallel split-termination resistors for the standard.

SSTL15 is used for DDR3 SDRAM interfaces and is roughly defined (not by name) in the JEDEC standard JESD79-3E [Ref 8]. For this standard, the full-strength driver (SSTL15) is available in both the HP and HR I/O banks. A weaker, reduced-strength driver, designated by an R in the standard name (SSTL15\_R), is available in the HR I/O banks. For some topologies (such as short point-to-point interfaces), the reduced-strength driver can result in reduced overshoot and better signal integrity. The HP I/O banks provide DCI options for tuned internal parallel split-termination resistors. HP and HR I/O banks provide options for untuned internal parallel split-termination resistors (using the ODT attributes). In addition, the source termination feature (OUTPUT\_IMPEDANCE) provides the option of 40 $\Omega$ , 48 $\Omega$ , or 60 $\Omega$  tuned driver impedance in HP I/O banks in both DCI and non-DCI versions. The driver output impedance is set to a default of 40 $\Omega$ . The optimal drive and termination scheme for any new design is determined through careful signal-integrity analysis.

SSTL135 is used for DDR3L SDRAM interfaces and is roughly defined (not by name) in the JEDEC standard JESD79-3-1 [Ref 8]. For this standard, the full-strength driver (SSTL135) is available in both the HP and HR I/O banks. A weaker, reduced-strength driver, designated by an R in the standard name (SSTL135\_R), is available in the HR I/O banks. For some

topologies (such as short point-to-point interfaces), the reduced-strength driver can result in reduced overshoot and better signal integrity.

The HP I/O banks also provide DCI options for tuned internal parallel split-termination resistors. HP and HR I/O banks also provide options for untuned internal parallel split-termination resistors (using the ODT attributes). In addition, the source termination feature (OUTPUT\_IMPEDANCE) provides the option of 40Ω, 48Ω, or 60Ω tuned driver impedance in HP I/O banks in both DCI and non-DCI versions. The driver output impedance is set to a default of 40Ω. The optimal drive and termination scheme for any new design is determined through careful signal-integrity analysis.

SSTL12 supports Micron's next-generation RLD RAM3 memory. The DCI option is available to improve the signal integrity through the use of tuned internal split-termination resistors in HP I/O banks. HR and HP I/O banks also provide the ODT attribute options for untuned internal parallel split-termination resistors. In addition, the source termination feature (OUTPUT\_IMPEDANCE) provides the option of 40Ω, 48Ω, or 60Ω tuned driver impedance in HP I/O banks in both DCI and non-DCI versions. The driver output impedance is set to a default of 40Ω. The optimal drive and termination scheme for any new design is determined through careful signal-integrity analysis.

### **SSTL18\_I, DIFF\_SSTL18\_I**

Table 1-39: Available I/O Bank Type

HR	HP
Available	Available

Class-I drivers can be preferred for short, point-to-point board topologies. Parallel end-termination resistors (commonly 50Ω) to  $V_{TT} = (V_{CC0}/2)$  are typically placed on the board close to any receiver. Optional untuned split input ODT provides Thevenin equivalent resistance of R (where  $R = Z_0$ ) to the  $V_{CC0}/2$ . The untuned on-die source termination feature (OUTPUT\_IMPEDANCE) provides the option of 40Ω, 48Ω, or 60Ω of driver impedance in HP I/O banks. The driver output impedance is set to a default of 40Ω. The differential (DIFF\_) version uses complementary single-ended drivers for outputs, and differential receivers for inputs.

### **SSTL18\_I\_DCI, DIFF\_SSTL18\_I\_DCI**

Table 1-40: Available I/O Bank Type

HR	HP
N/A	Available

Class-I drivers can be preferred for short, point-to-point board topologies. DCI provides tuned internal parallel split-termination resistors that are always present. The value of the ODT attributes represents the Thevenin equivalent resistance of R (where  $R = Z_0$ ) to the  $V_{CC0}/2$  mid-point level. The source termination feature (OUTPUT\_IMPEDANCE) provides the

option of 40Ω, 48Ω, or 60Ω tuned driver impedance in HP I/O banks. The driver output impedance is set to a default of 40Ω. The differential (DIFF\_) version uses complementary single-ended drivers for outputs, and differential receivers for inputs.

***SSTL18\_II, SSTL15\_R, SSTL135\_R, DIFF\_SSTL18\_II, DIFF\_SSTL15\_R, DIFF\_SSTL135\_R***

Table 1-41: Available I/O Bank Type

HR	HP
Available	Not Available

Parallel end-termination resistors (commonly 50Ω) to  $V_{TT} = (V_{CCO}/2)$  are typically placed on the board close to any receiver. Depending on the board topology, source-termination series resistors help match the output driver impedance to the transmission line and end-termination impedances, to reduce reflections and improve signal integrity. Optional untuned split input ODT provides Thevenin equivalent resistance of R (where  $R = Z_0$ ) to the  $V_{CCO}/2$ . The differential (DIFF\_) versions use complementary single-ended drivers for outputs, and differential receivers for inputs.

***SSTL15, SSTL135, SSTL12, DIFF\_SSTL15, DIFF\_SSTL135, DIFF\_SSTL12***

Table 1-42: Available I/O Bank Type

HR	HP
Available	Available

Parallel end-termination resistors (commonly 50Ω) to  $V_{TT} = (V_{CCO}/2)$  are typically placed on the board close to any receiver. Depending on the board topology, source-termination series resistors help match the output driver impedance to the transmission line and end-termination impedances, to reduce reflections and improve signal integrity. Optional untuned split input ODT provides Thevenin equivalent resistance of R (where  $R = Z_0$ ) to the  $V_{CCO}/2$ . Untuned on-die source termination feature (OUTPUT\_IMPEDANCE) provides the option of 40Ω, 48Ω, or 60Ω of driver impedance in HP I/O banks. The driver output impedance is set to a default of 40Ω. The differential (DIFF\_) versions use complementary single-ended drivers for outputs, and differential receivers for inputs.

***SSTL15\_DCI, SSTL135\_DCI, SSTL12\_DCI, DIFF\_SSTL15\_DCI, DIFF\_SSTL135\_DCI, DIFF\_SSTL12\_DCI***

Table 1-43: Available I/O Bank Type

HR	HP
N/A	Available

The DCI standards provide tuned internal parallel split-termination resistors that are always present at the receivers. The value of both the resistance set by the ODT attributes, creates

the Thevenin equivalent of  $R$  (where  $R = Z_0$ ) to the  $V_{CC0}/2$  mid-point level. The source termination feature (OUTPUT\_IMPEDANCE) provides the option of  $40\Omega$ ,  $48\Omega$ , or  $60\Omega$  tuned driver impedance in HP I/O banks. The driver output impedance is set to a default of  $40\Omega$ . The differential (DIFF\_) versions use complementary single-ended drivers for outputs, and differential receivers for inputs.

## SSTL18, SSTL15, SSTL135, SSTL12

Figure 1-64 shows a sample circuit illustrating a unidirectional termination technique for SSTL18, SSTL15, SSTL135, or SSTL12. In a specific circuit, all drivers and receivers must be at the same voltage level (1.8V, 1.5V, 1.35V, or 1.2V); they are not interchangeable (i.e., SSTL12 should only interface with SSTL12).

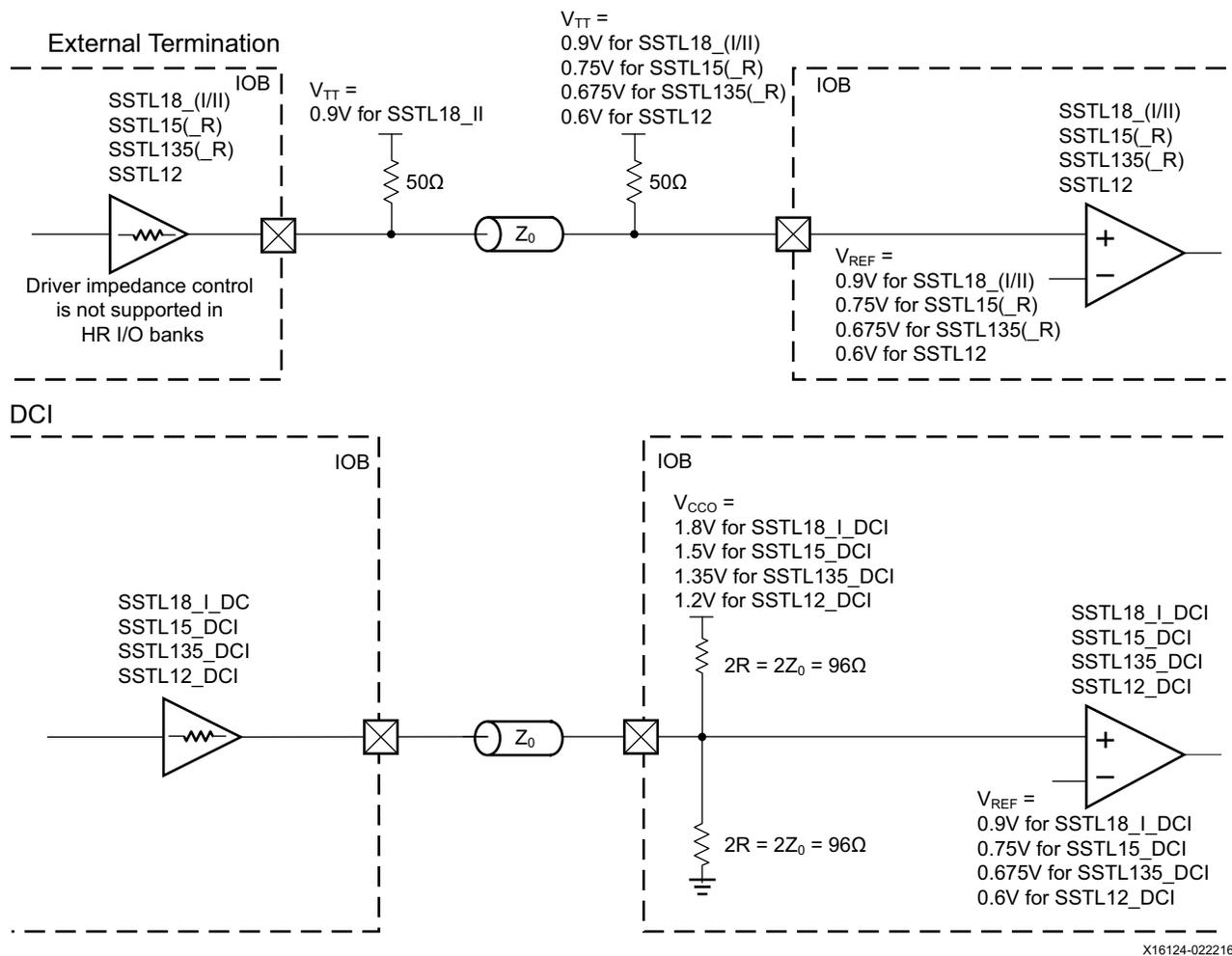


Figure 1-64: SSTL18, SSTL15, SSTL135, or SSTL12 Unidirectional Termination

Figure 1-65 shows a sample circuit illustrating a bidirectional termination technique for SSTL18, SSTL15, SSTL135, or SSTL12. In a specific circuit, all drivers and receivers must be at the same voltage level (1.8V, 1.5V, 1.35V, or 1.2V); they are not interchangeable (i.e., SSTL12 should only interface with SSTL12).

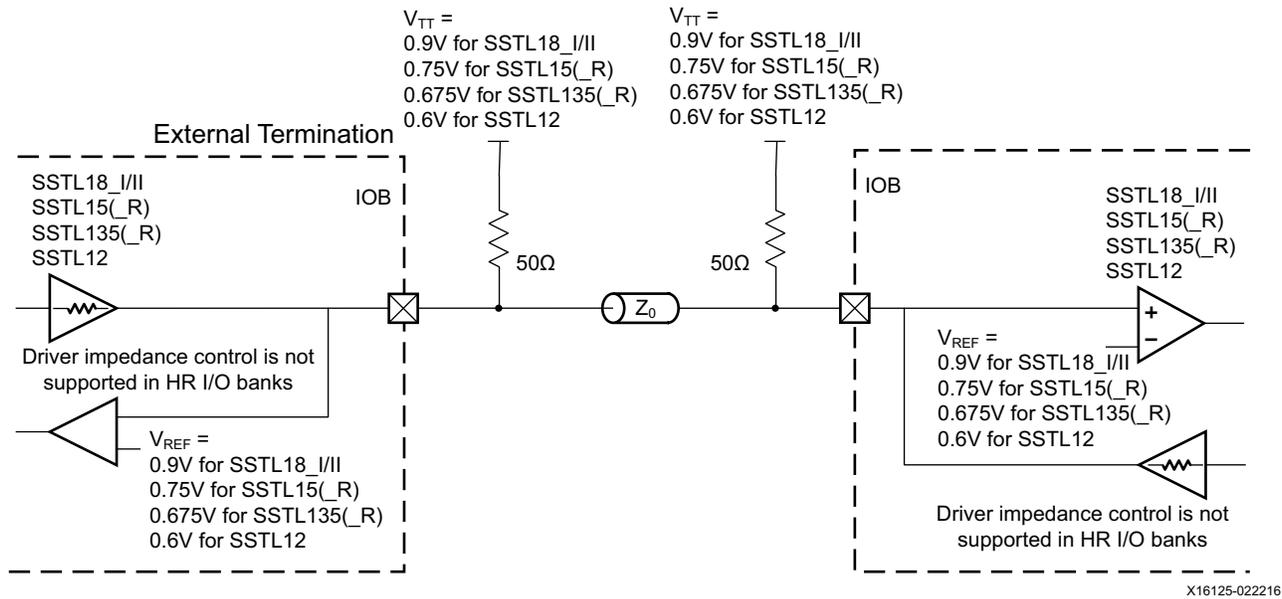


Figure 1-65: SSTL18, SSTL15, SSTL135, or SSTL12 Bidirectional Termination

Figure 1-66 shows a sample circuit illustrating a bidirectional termination technique for SSTL18, SSTL15, SSTL135, or SSTL12 with DCI. In a specific circuit, all drivers and receivers must be at the same voltage level (1.8V, 1.5V, 1.35V, or 1.2V); they are not interchangeable (i.e., SSTL12\_DCI should only interface with SSTL2\_DCI). DCI standards are only supported in HP I/O banks.

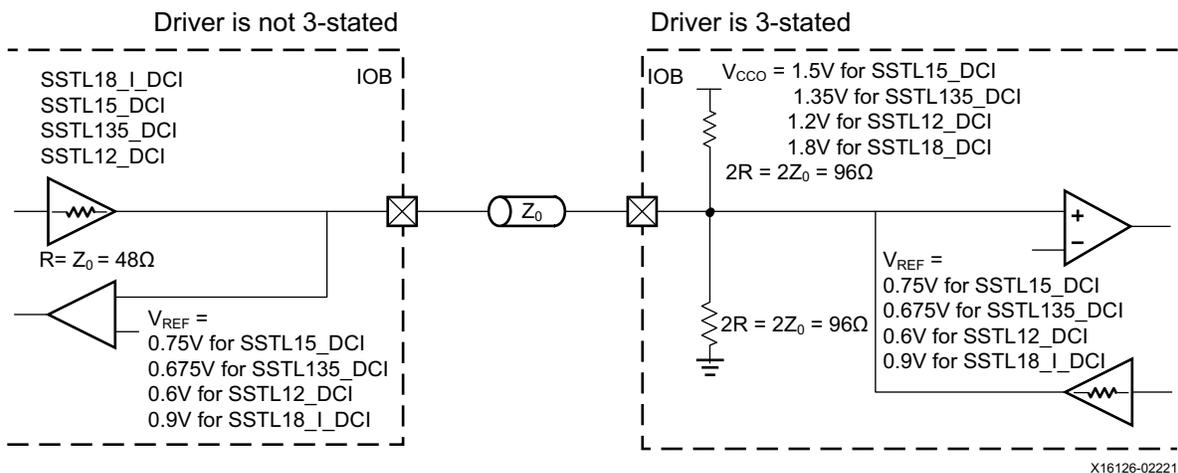


Figure 1-66: SSTL18\_DCI, SSTL15\_DCI, SSTL135\_DCI, or SSTL12\_DCI Bidirectional Termination

## Differential SSTL18, SSTL15, SSTL135, SSTL12

Figure 1-67 shows a sample circuit illustrating a termination technique for differential SSTL18, SSTL15, SSTL135, or SSTL12 with unidirectional termination. In a specific circuit, all drivers and receivers must be at the same voltage level (1.8V, 1.5V, 1.35V, or 1.2V); they are not interchangeable (i.e., DIFF\_SSTL12 should only interface with DIFF\_SSTL12).

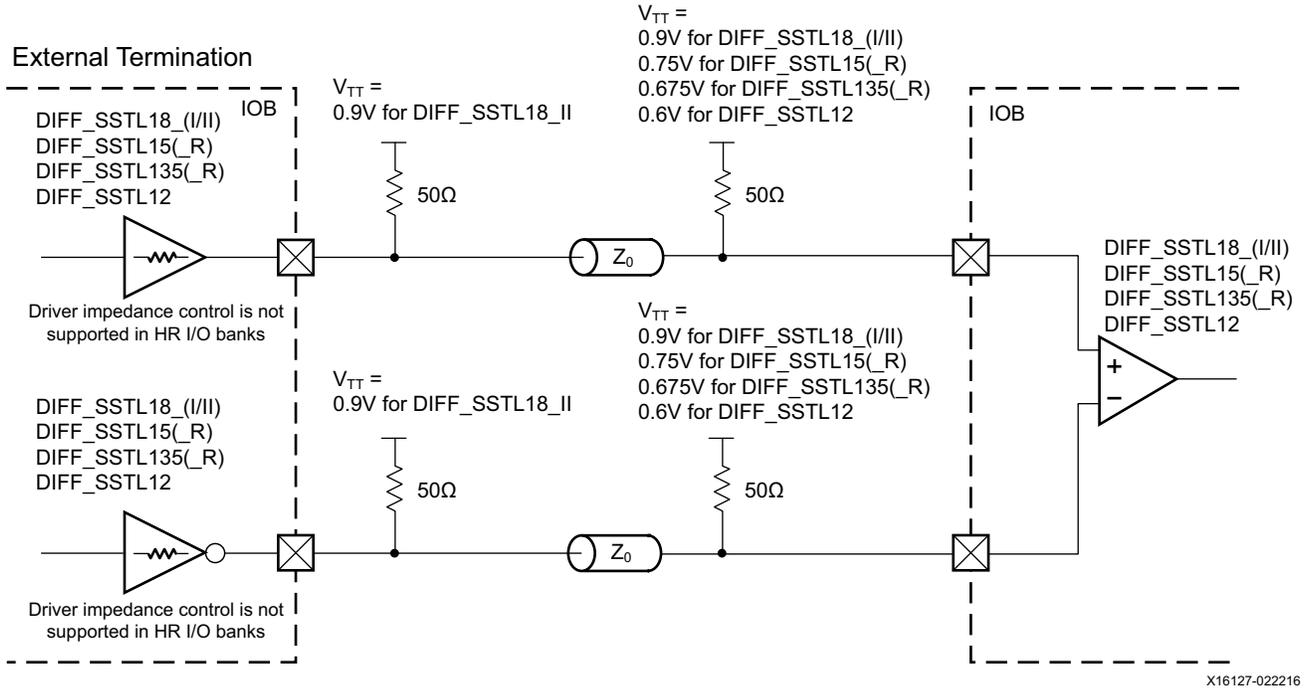


Figure 1-67: Differential SSTL18, SSTL15, SSTL135, or SSTL12 Unidirectional Termination

Figure 1-68 shows a sample circuit illustrating a termination technique for differential SSTL18, SSTL15, SSTL135, or SSTL12 with unidirectional DCI termination. In a specific circuit, all drivers and receivers must be at the same voltage level (1.8V, 1.5V, 1.35V, or 1.2V); they are not interchangeable (i.e., DIFF\_SSTL12\_DCI should only interface with DIFF\_SSTL12\_DCI).

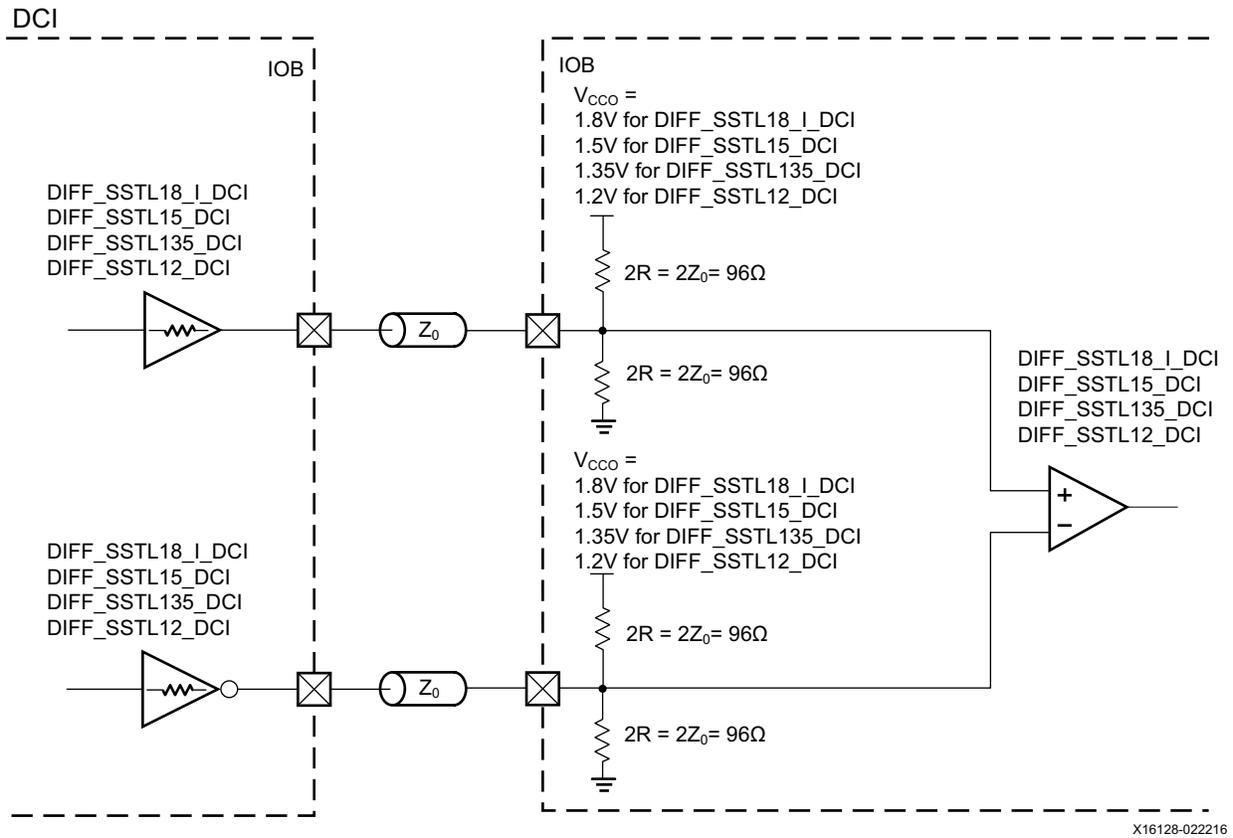
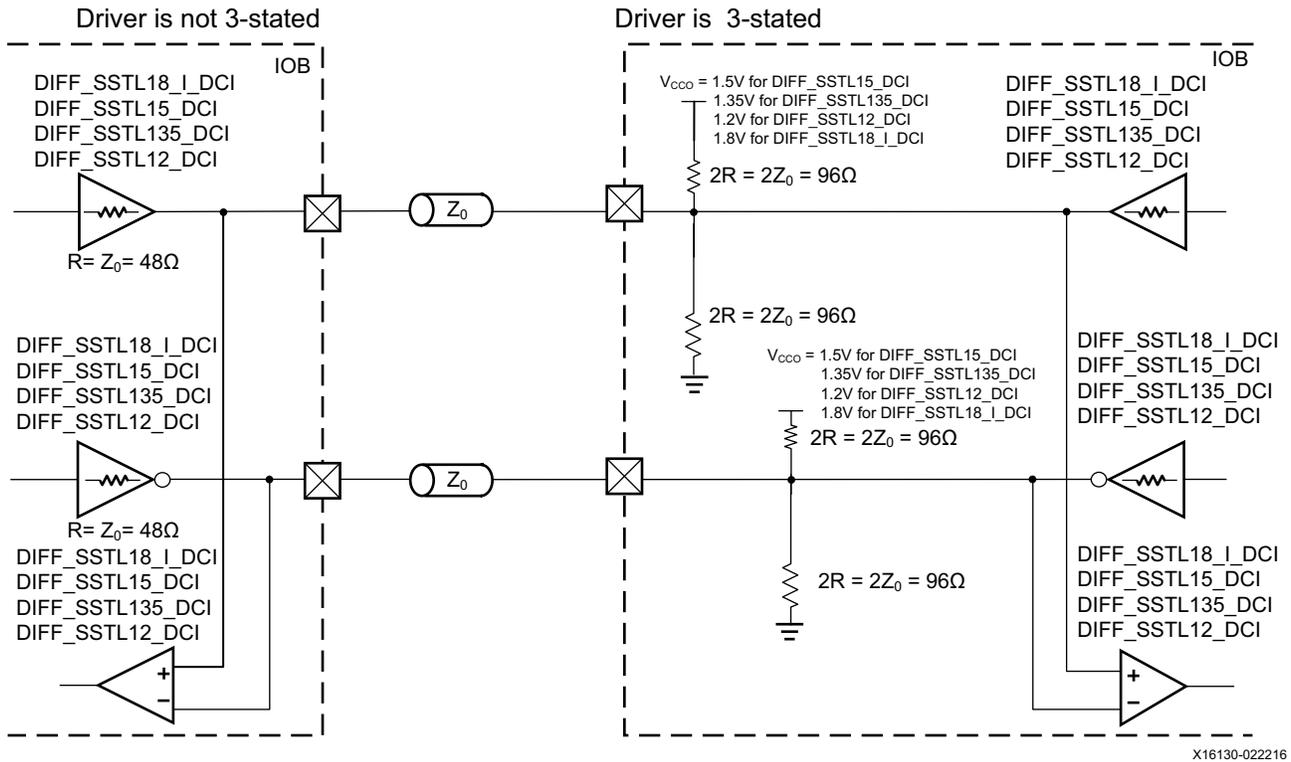


Figure 1-68: Differential SSTL18, SSTL15, SSTL135, or SSTL12 Unidirectional DCI Termination



Figure 1-70 shows a sample circuit illustrating a termination technique for differential SSTL18, SSTL15, SSTL135, or SSTL12 with bidirectional DCI termination. In a specific circuit, all drivers and receivers must be at the same voltage level (1.8V, 1.5V, 1.35V, or 1.2V); they are not interchangeable (i.e., DIFF\_SSTL12\_DCI should only interface with DIFF\_SSTL12\_DCI). DCI standards are supported only in HP I/Os.



X16130-022216

Figure 1-70: Differential SSTL18, SSTL15, SSTL135, or SSTL12 with Bidirectional DCI Termination

Table 1-44 lists the allowed attributes for SSTL I/O standards. Support is implied for primitives that are derivatives of the primitives listed in Table 1-44 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-44: SSTL Allowed Attributes

Attributes	IBUF/IBUFE3/IBUFDS/IBUFDSE3				OBUF/OBUFT				IOBUF/IOBUFE3/IOBUFDS/IOBUFDSE3			
	HP I/O		HR I/O		HP I/O		HR I/O		HP I/O		HR I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	SSTL12 SSTL135 SSTL15 SSTL18_I		SSTL12 SSTL135 SSTL135_R SSTL15 SSTL15_R SSTL18_I		SSTL12 SSTL135 SSTL15 SSTL18_I		SSTL12 SSTL135 SSTL135_R SSTL15 SSTL15_R SSTL18_I		SSTL12 SSTL135 SSTL15 SSTL18_I		SSTL12 SSTL135 SSTL135_R SSTL15 SSTL15_R SSTL18_I	
SLEW	N/A		N/A		FAST, MEDIUM, SLOW	SLOW	FAST SLOW	SLOW	FAST, MEDIUM, SLOW	SLOW	FAST SLOW	SLOW
ODT	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE	N/A		N/A		RTT_40 RTT_48 RTT_60 RTT_NONE <sup>(1)</sup>	RTT_NONE	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE
OUTPUT_IMPEDANCE	N/A		N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60	RDRV_40_40	N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60 <sup>(1)</sup>	RDRV_40_40	N/A	

Table 1-44: SSTL Allowed Attributes (Cont'd)

Attributes	IBUF/IBUFE3/IBUFDS/IBUFDSE3				OBUF/OBUFT				IOBUF/IOBUFE3/IOBUFDS/IOBUFDSE3			
	HP I/O		HR I/O		HP I/O		HR I/O		HP I/O		HR I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	SSTL12_DCI SSTL135_DCI SSTL15_DCI SSTL18_I_DCI		N/A		SSTL12_DCI SSTL135_DCI SSTL15_DCI SSTL18_I_DCI		N/A		SSTL12_DCI SSTL135_DCI SSTL15_DCI SSTL18_I_DCI		N/A	
SLEW	N/A		N/A		FAST, MEDIUM, SLOW	SLOW	N/A		FAST, MEDIUM, SLOW	SLOW	N/A	
ODT	RTT_40 RTT_48 RTT_60	RTT_40	N/A		N/A		N/A		RTT_40 RTT_48 RTT_60 <sup>(1)(3)</sup>	RTT_40	N/A	
OUTPUT_IMPEDANCE	N/A		N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60	RDRV_40_40	N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60 <sup>(1)</sup>	RDRV_40_40	N/A	
IOSTANDARD	DIFF_SSTL12 DIFF_SSTL135 DIFF_SSTL15 DIFF_SSTL18_I		DIFF_SSTL12 DIFF_SSTL135 DIFF_SSTL15 DIFF_SSTL18_I		DIFF_SSTL12 DIFF_SSTL135 DIFF_SSTL15 DIFF_SSTL18_I		DIFF_SSTL12 DIFF_SSTL135 DIFF_SSTL15 DIFF_SSTL18_I		DIFF_SSTL12 DIFF_SSTL135 DIFF_SSTL15 DIFF_SSTL18_I		DIFF_SSTL12 DIFF_SSTL135 DIFF_SSTL15 DIFF_SSTL18_I	
SLEW	N/A		N/A		FAST MEDIUM SLOW	SLOW	FAST SLOW	SLOW	FAST MEDIUM SLOW	SLOW	FAST SLOW	SLOW
DQS_BIAS <sup>(2)</sup>	TRUE FALSE	FALSE	TRUE FALSE	FALSE	N/A		N/A		TRUE FALSE	FALSE	TRUE FALSE	FALSE
ODT	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE	N/A		N/A		RTT_40 RTT_48 RTT_60 RTT_NONE <sup>(1)</sup>	RTT_NONE	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE
OUTPUT_IMPEDANCE	N/A		N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60	RDRV_40_40	N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60 <sup>(1)</sup>	RDRV_40_40	N/A	

Table 1-44: SSTL Allowed Attributes (Cont'd)

Attributes	IBUF/IBUFE3/IBUFDS/IBUFDSE3				OBUF/OBUFT				IOBUF/IOBUFE3/IOBUFDS/IOBUFDSE3			
	HP I/O		HR I/O		HP I/O		HR I/O		HP I/O		HR I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	DIFF_SSTL12_DCI DIFF_SSTL135_DCI DIFF_SSTL15_DCI DIFF_SSTL18_I_DCI		N/A		DIFF_SSTL12_DCI DIFF_SSTL135_DCI DIFF_SSTL15_DCI DIFF_SSTL18_I_DCI		N/A		DIFF_SSTL12_DCI DIFF_SSTL135_DCI DIFF_SSTL15_DCI DIFF_SSTL18_I_DCI		N/A	
SLEW	N/A		N/A		FAST MEDIUM SLOW	SLOW	N/A		FAST MEDIUM SLOW	SLOW	N/A	
DQS_BIAS <sup>(2)(4)</sup>	TRUE FALSE	FALSE	N/A		N/A		N/A		TRUE FALSE	FALSE	N/A	
ODT	RTT_40 RTT_48 RTT_60 <sup>(3)</sup>	RTT_40	N/A		N/A		N/A		RTT_40 RTT_48 RTT_60 <sup>(1)(3)</sup>	RTT_40	N/A	
OUTPUT_IMPEDANCE	N/A		N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60	RDRV_40_40	N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60 <sup>(1)</sup>	RDRV_40_40	N/A	
IOSTANDARD	N/A		DIFF_SSTL135_R DIFF_SSTL15_R		N/A		DIFF_SSTL135_R DIFF_SSTL15_R		N/A		DIFF_SSTL135_R DIFF_SSTL15_R	
SLEW	N/A		N/A		N/A		FAST SLOW	SLOW	N/A		FAST SLOW	SLOW
ODT	N/A		RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE	N/A		N/A		N/A		RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE

**Notes:**

1. The allowed bidirectional configuration combinations for driver output impedance (OUTPUT\_IMPEDANCE) and ODT are listed in [Table 1-37](#).
2. The DQS\_BIAS attribute is set on the I/O port rather than the primitive.
3. ODT = RTT\_NONE is not a valid setting for DCI I/O standards.
4. This is read-only on the primitive.

Table 1-45 lists the allowed attributes for SSTL Class II I/O standards. Support is implied for primitives that are derivatives of the primitives listed in Table 1-45 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-45: SSTL Class II Allowed Attributes

Attributes	IBUF/IBUFDS			OBUF/OBUFT			IOBUF/IOBUFDS		
	HP I/O	HR I/O		HP I/O	HR I/O		HP I/O	HR I/O	
		Allowed Values	Default		Allowed Values	Default		Allowed Values	Default
IOSTANDARD	N/A	SSTL18_II DIFF_SSTL18_II		N/A	SSTL18_II DIFF_SSTL18_II		N/A	SSTL18_II DIFF_SSTL18_II	
SLEW	N/A	N/A		N/A	FAST SLOW	SLOW	N/A	FAST SLOW	SLOW
ODT	N/A	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE	N/A	N/A		N/A	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE

## HSUL\_12

The high speed unterminated logic (HSUL\_12) standard is for LPDDR2 and LPDDR3 memory buses. HSUL\_12 is defined by the JEDEC standard JESD8-22 [Ref 8]. UltraScale devices support this standard for single-ended signaling and differential signaling. Similar to SSTL, this standard also requires a differential amplifier input buffer and a push-pull output buffer.

### HSUL\_12 and DIFF\_HSUL\_12

Table 1-46: Available I/O Bank Type

HR	HP
Available	Available

The differential (DIFF\_) version uses complementary single-ended drivers for outputs, and differential receivers for inputs. In HP I/O banks, an optional untuned split input ODT provides a weak pull-up to  $V_{CC0}$ . The untuned on-die source termination feature (OUTPUT\_IMPEDANCE) provides the option of 40Ω, 48Ω, or 60Ω of driver impedance in HP I/O banks. The driver output impedance is set to a default of 48Ω.

**HSUL\_DCI\_12 and DIFF\_HSUL\_12\_DCI**

Table 1-47: Available I/O Bank Type

HR	HP
N/A	Available

DCI provides a tuned on-die input single termination to  $V_{CC0}$  on the receivers and a tuned on-die source termination option (OUTPUT\_IMPEDANCE) of 40Ω, 48Ω, or 60Ω of driver impedance in HP I/O banks. The impedances are scaled from the reference resistor on the VRP pin. The driver output impedance is set to a default of 48Ω. The differential (DIFF\_) versions use complementary single-ended drivers for outputs, and differential receivers for inputs.

**HSUL\_12**

Figure 1-71 shows a sample circuit illustrating a unidirectional board topology for HSUL\_12. Only HP I/O banks support the DCI version.

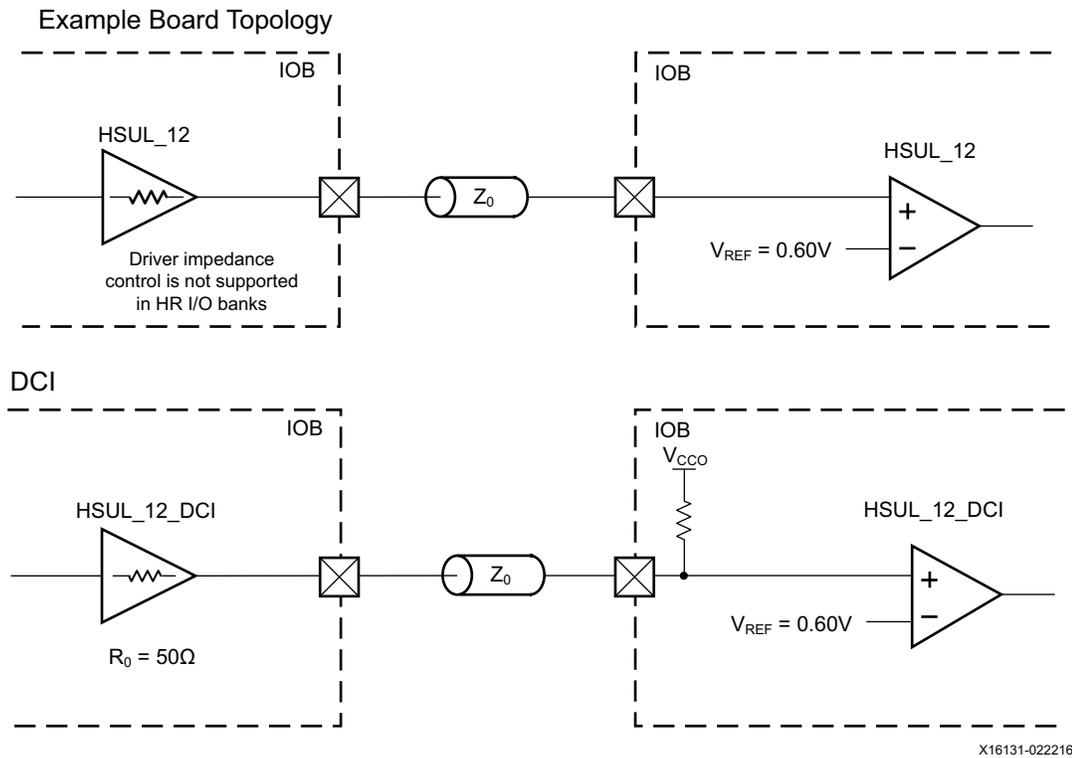


Figure 1-71: HSUL\_12 with Unidirectional Signaling

Figure 1-72 shows a sample circuit illustrating a bidirectional board topology (with no termination) for HSUL\_12. Only HP I/O banks support the DCI version.

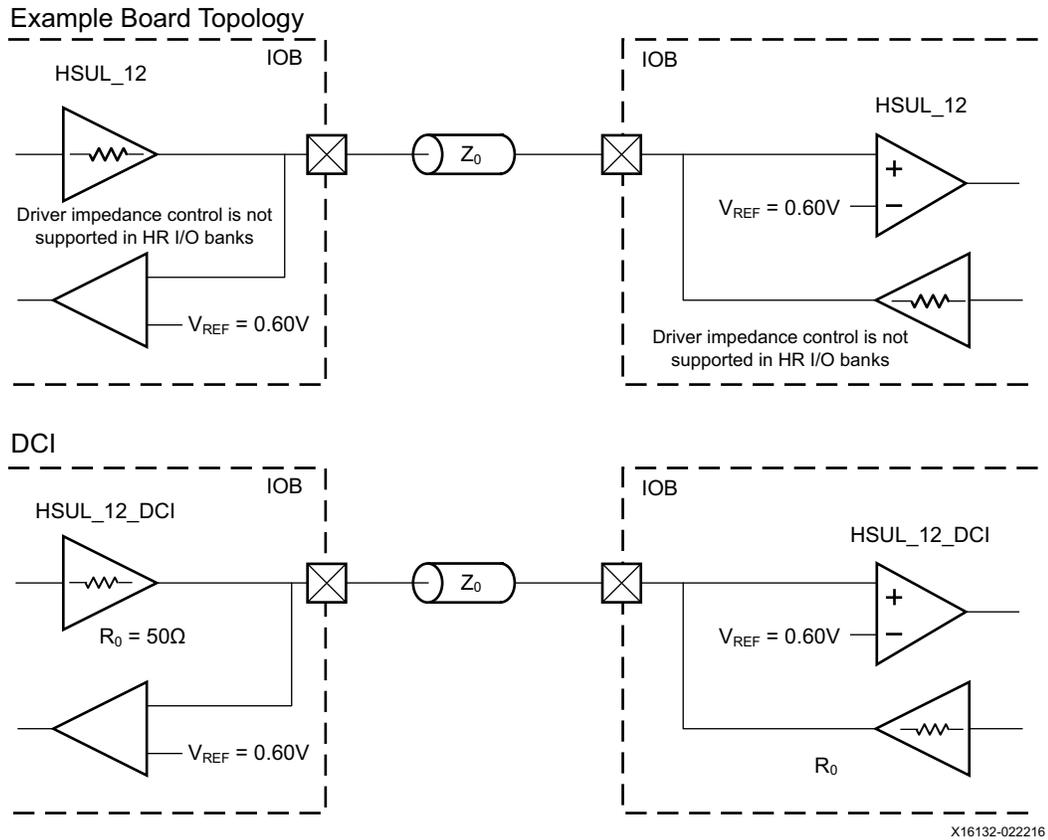


Figure 1-72: HSUL\_12 with Bidirectional Signaling

## Differential HSUL\_12

Figure 1-73 shows a sample circuit illustrating a board topology for differential HSUL\_12 with unidirectional signaling.

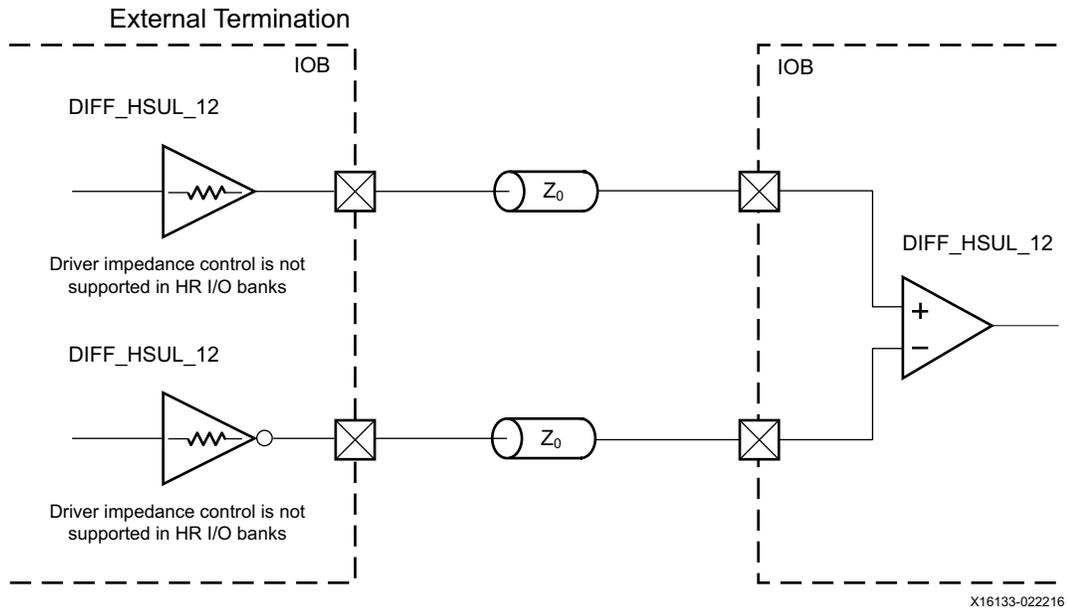


Figure 1-73: Differential HSUL\_12 with Unidirectional Signaling

Figure 1-74 shows a sample circuit illustrating a board topology for differential HSUL\_12 with unidirectional DCI signaling.

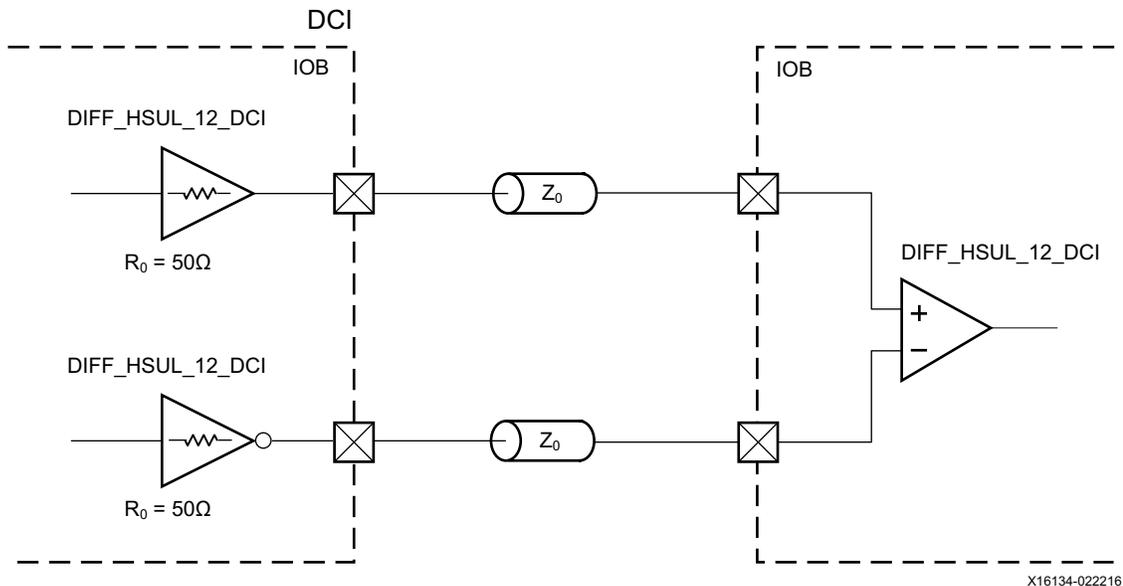


Figure 1-74: Differential HSUL\_12 with Unidirectional DCI Signaling

Figure 1-75 shows a sample circuit illustrating a board topology for differential HSUL<sub>12</sub> with bidirectional signaling.

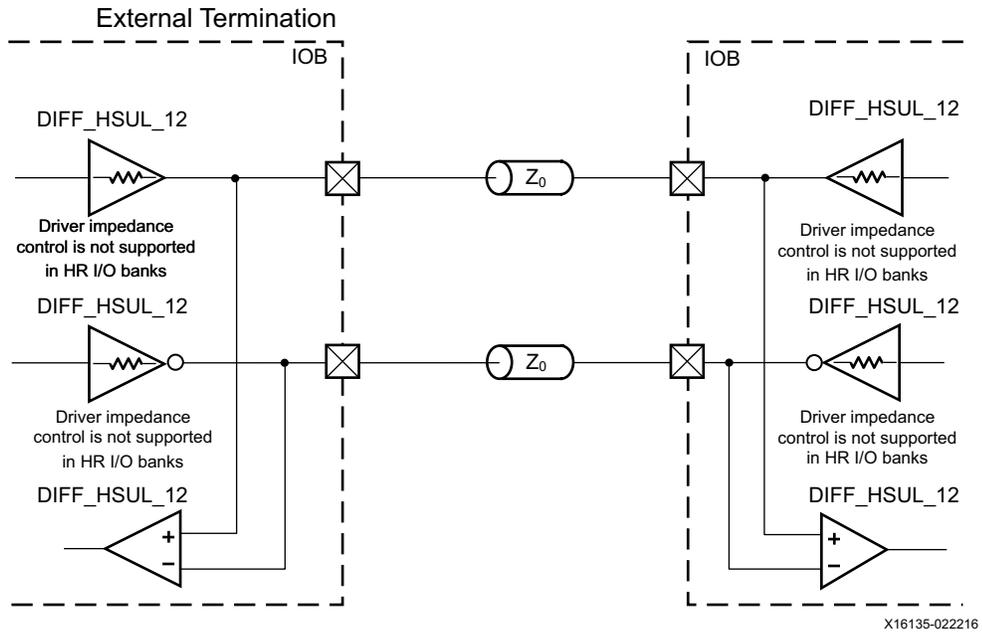


Figure 1-75: Differential HSUL<sub>12</sub> with Bidirectional Signaling

Figure 1-76 shows a sample circuit illustrating a board topology for differential HSUL<sub>12</sub> with bidirectional DCI signaling.

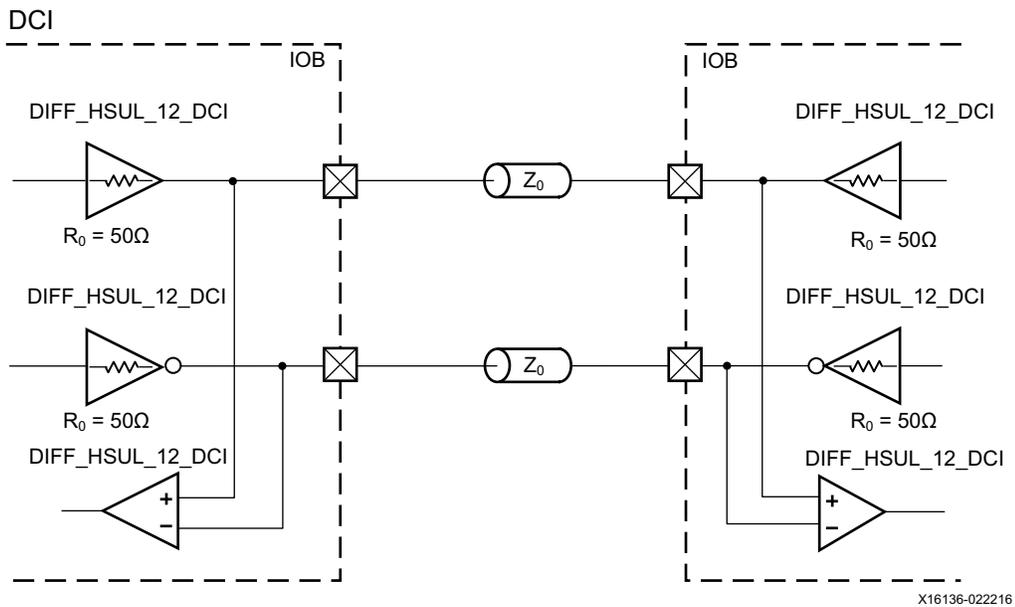


Figure 1-76: Differential HSUL<sub>12</sub> with DCI Bidirectional Signaling

Table 1-48 lists the allowed attributes for HSUL I/O standards. Support is implied for primitives that are derivatives of the primitives listed in Table 1-48 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-48: HSUL Allowed Attributes

Attributes	IBUF/IBUFE3/IBUFDS/IBUFDSE3				OBUF/OBUFT				IOBUF/IOBUFE3/IOBUFDSE3			
	HP I/O		HR I/O		HP I/O		HR I/O		HP I/O		HR I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	HSUL_12 DIFF_HSUL_12		HSUL_12 DIFF_HSUL_12		HSUL_12 DIFF_HSUL_12		HSUL_12 DIFF_HSUL_12		HSUL_12 DIFF_HSUL_12		HSUL_12 DIFF_HSUL_12	
SLEW	N/A		N/A		FAST MEDIUM SLOW	SLOW	FAST SLOW	SLOW	FAST MEDIUM SLOW	SLOW	FAST SLOW	SLOW
ODT	RTT_120 RTT_240 RTT_NONE	RTT_NONE	N/A		N/A		N/A		RTT_120 RTT_240 RTT_NONE	RTT_NONE	N/A	
OUTPUT_IMPEDANCE	N/A		N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60	RDRV_48_48	N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60	RDRV_48_48	N/A	
IOSTANDARD	HSUL_12_DCI DIFF_HSUL_12_DCI		N/A		HSUL_12_DCI DIFF_HSUL_12_DCI		N/A		HSUL_12_DCI DIFF_HSUL_12_DCI		N/A	
SLEW	N/A		N/A		FAST MEDIUM SLOW	SLOW	N/A		FAST MEDIUM SLOW	SLOW	N/A	
DQS_BIAS <sup>(1)</sup>	TRUE FALSE	FALSE	N/A		N/A		N/A		TRUE FALSE	FALSE	N/A	
ODT	RTT_120 RTT_240 RTT_NONE	RTT_NONE	N/A		N/A		N/A		RTT_120 RTT_240 RTT_NONE	RTT_NONE	N/A	

Table 1-48: HSUL Allowed Attributes

Attributes	IBUF/IBUFE3/IBUFDS/IBUFDSE3				OBUF/OBUFT				IOBUF/IOBUFE3/IOBUFDS/IOBUFDSE3			
	HP I/O		HR I/O		HP I/O		HR I/O		HP I/O		HR I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
OUTPUT_IMPEDANCE	N/A		N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60	RDRV_48_48	N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60	RDRV_48_48	N/A	

**Notes:**

1. Applies to DIFF\_HSUL12 I/O standards.

## POD12 and POD10

Pseudo Open Drain (POD) standards POD12 and POD10 are intended for DDR4, DDR4L, and LLDRAM3 applications. POD12 and POD10 are only available in HP I/O banks and use  $V_{REF}$ .

### *POD10, POD12, DIFF\_POD10, and DIFF\_POD12*

Table 1-49: Available I/O Bank Type

HR	HP
N/A	Available

The differential (DIFF\_) versions (DIFF\_POD10 and DIFF\_POD12) use complementary single-ended drivers for outputs, and differential receivers for inputs. Optional untuned split input ODT provides pull-up to  $V_{CC0}$ . The untuned on-die source termination feature (OUTPUT\_IMPEDANCE) provides the option of 40Ω, 48Ω, or 60Ω of driver impedance in HP I/O banks. The driver output impedance is set to a default of 40Ω. POD12 standards also have optional EQUALIZATION and OFFSET\_CNTRL features in the receivers and PRE\_EMPHASIS in the drivers.

### *POD10\_DCI, POD12\_DCI, DIFF\_POD10\_DCI, and DIFF\_POD12\_DCI*

Table 1-50: Available I/O Bank Type

HR	HP
N/A	Available

DCI provides a tuned single termination to  $V_{CC0}$  at the receiver that matches the ODT attribute setting. The differential (DIFF\_) versions use complementary single-ended drivers for outputs, and differential receivers for inputs.

DCI provides a tuned single ODT pull-up to  $V_{CC0}$  in the receivers and a source termination option (OUTPUT\_IMPEDANCE) of 40Ω, 48Ω, or 60Ω in the driver. The driver output impedance is set to a default of 40Ω. POD12 standards also have optional EQUALIZATION and OFFSET\_CNTRL features in the receivers and PRE\_EMPHASIS in the drivers.

POD

Figure 1-77 shows a sample circuit illustrating a unidirectional board topology for POD (1.0V or 1.2V) with matched driver and receiver termination values. Only HP I/O banks support these standards.

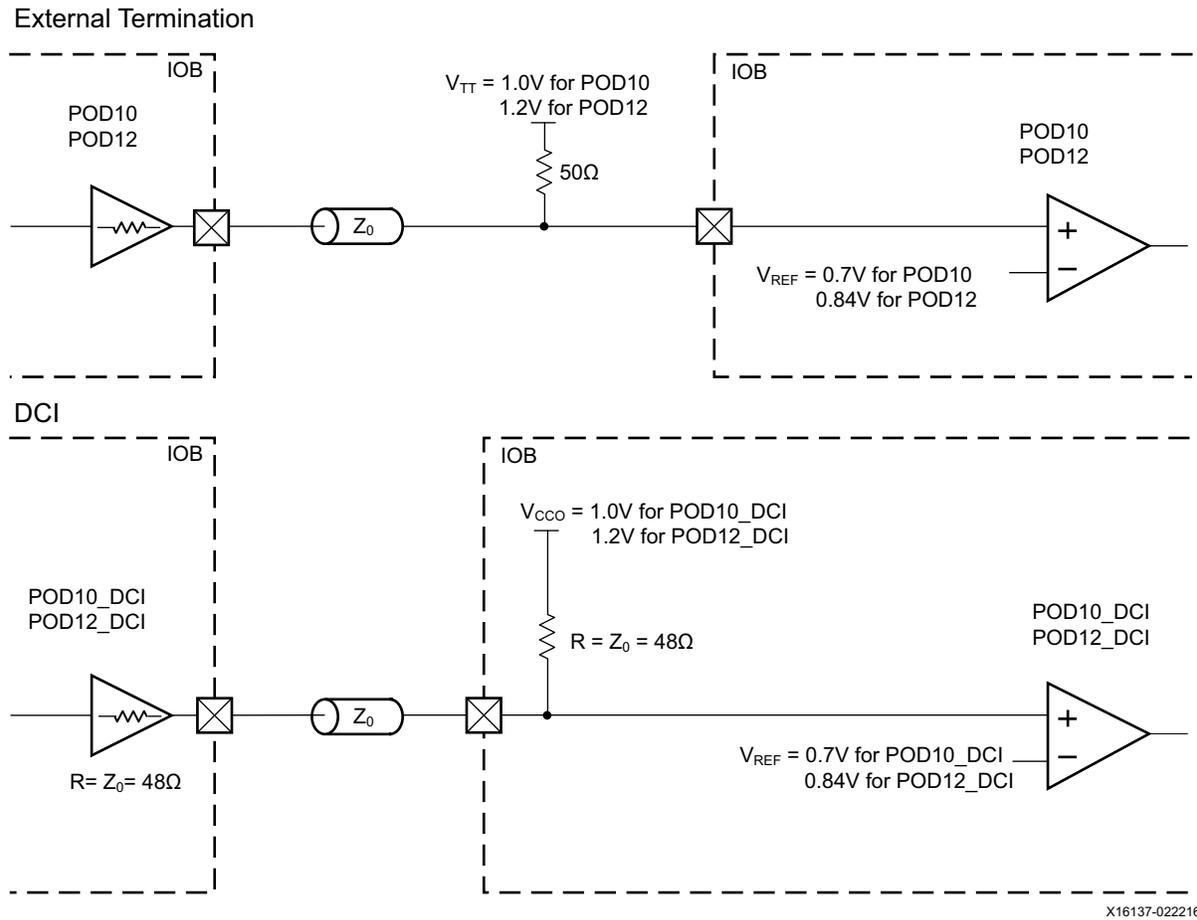


Figure 1-77: POD with Unidirectional Signaling

Figure 1-78 shows a sample circuit illustrating a termination technique for POD (1.0V or 1.2V) with bidirectional termination and with matched driver and receiver termination values. In a specific circuit, all drivers and receivers must be at the same voltage level (1.0V or 1.2V); they are not interchangeable (i.e., POD12 should only interface with POD12).

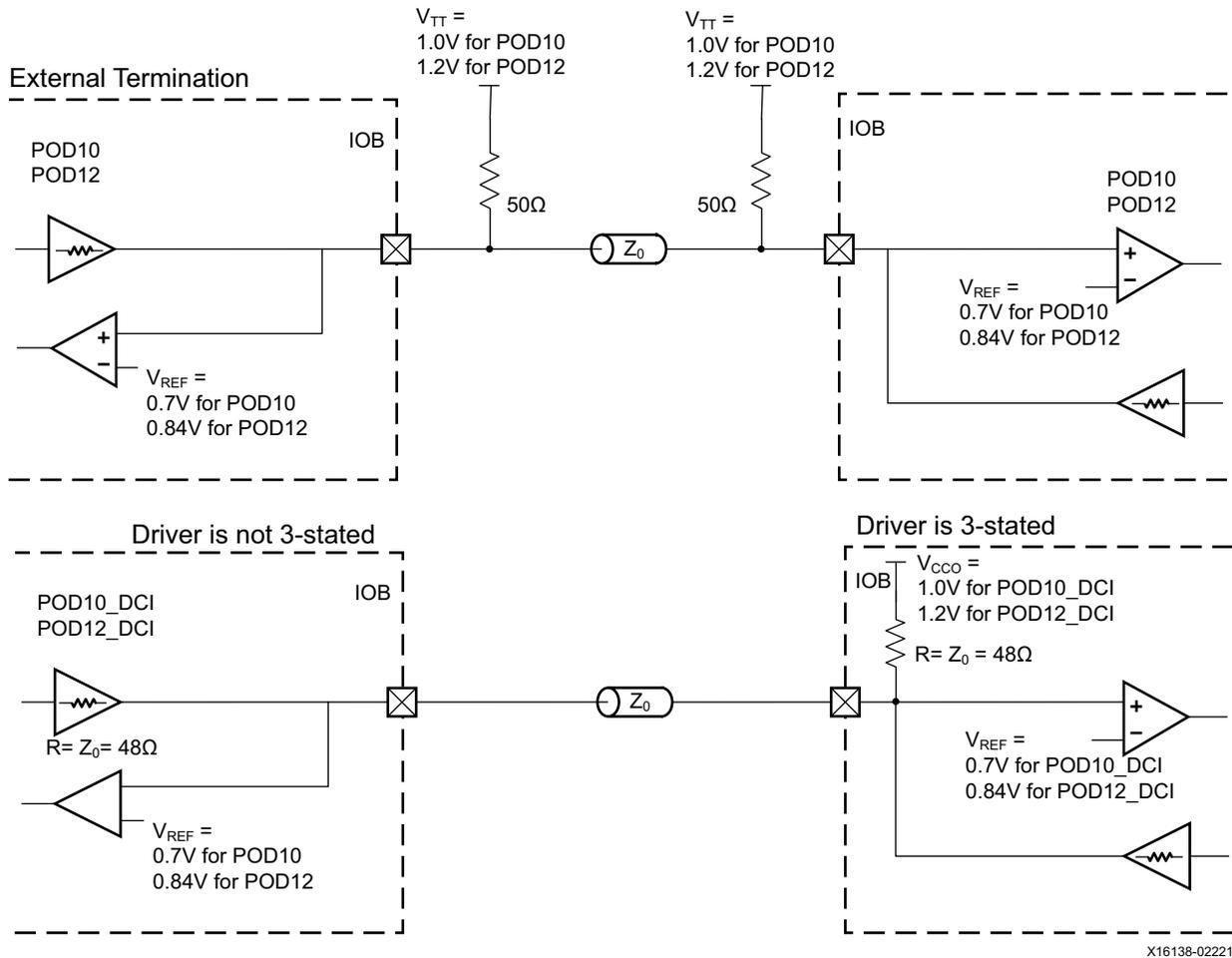


Figure 1-78: POD with Bidirectional Signaling

## Differential POD

Figure 1-79 shows a sample circuit illustrating a termination technique for differential POD (1.0V, 1.2V) with unidirectional termination and with matched driver and receiver termination values. In a specific circuit, all drivers and receivers must be at the same voltage level (1.2V or 1.0V); they are not interchangeable (i.e., DIFF\_POD12 should only interface with DIFF\_POD12).

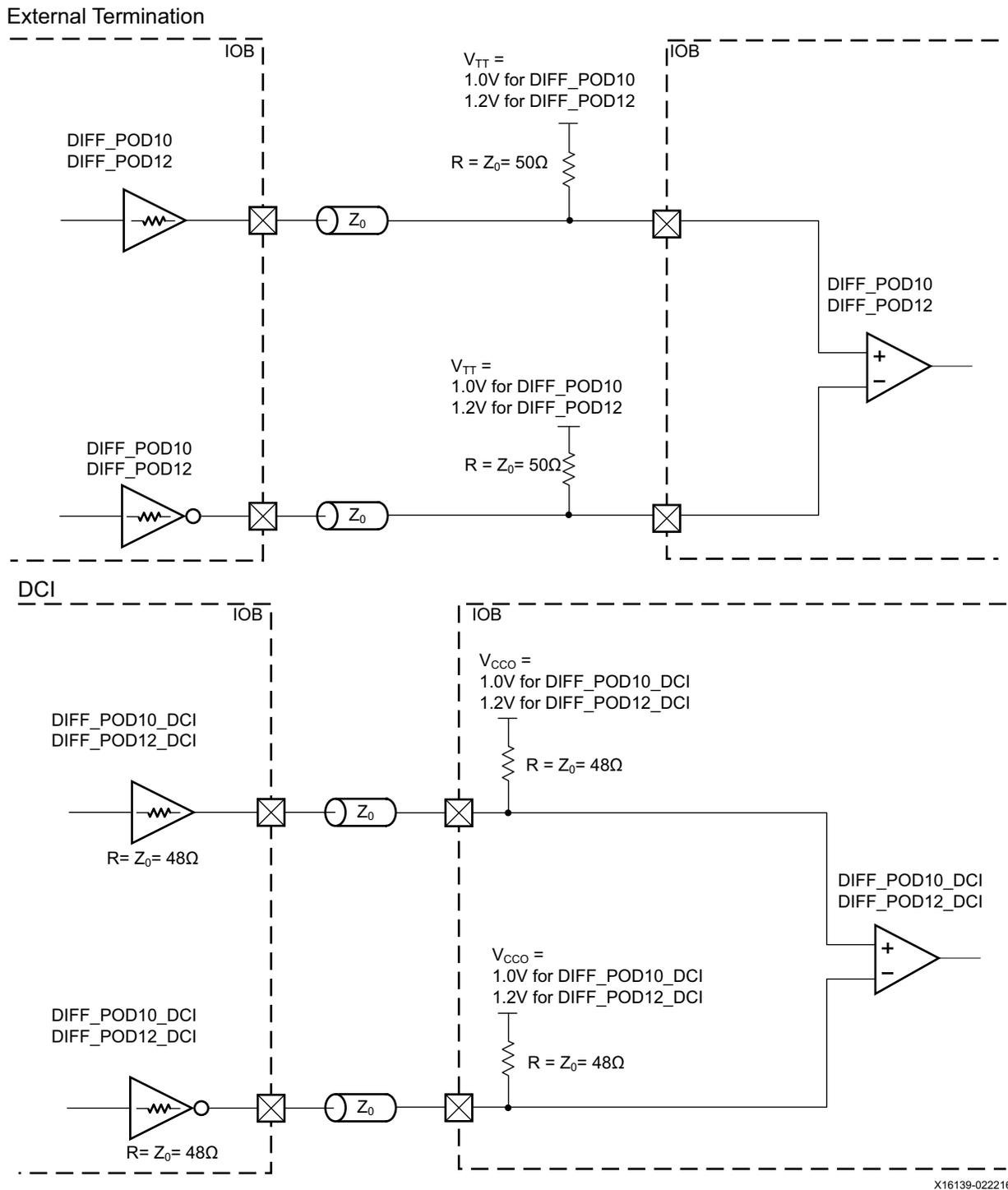
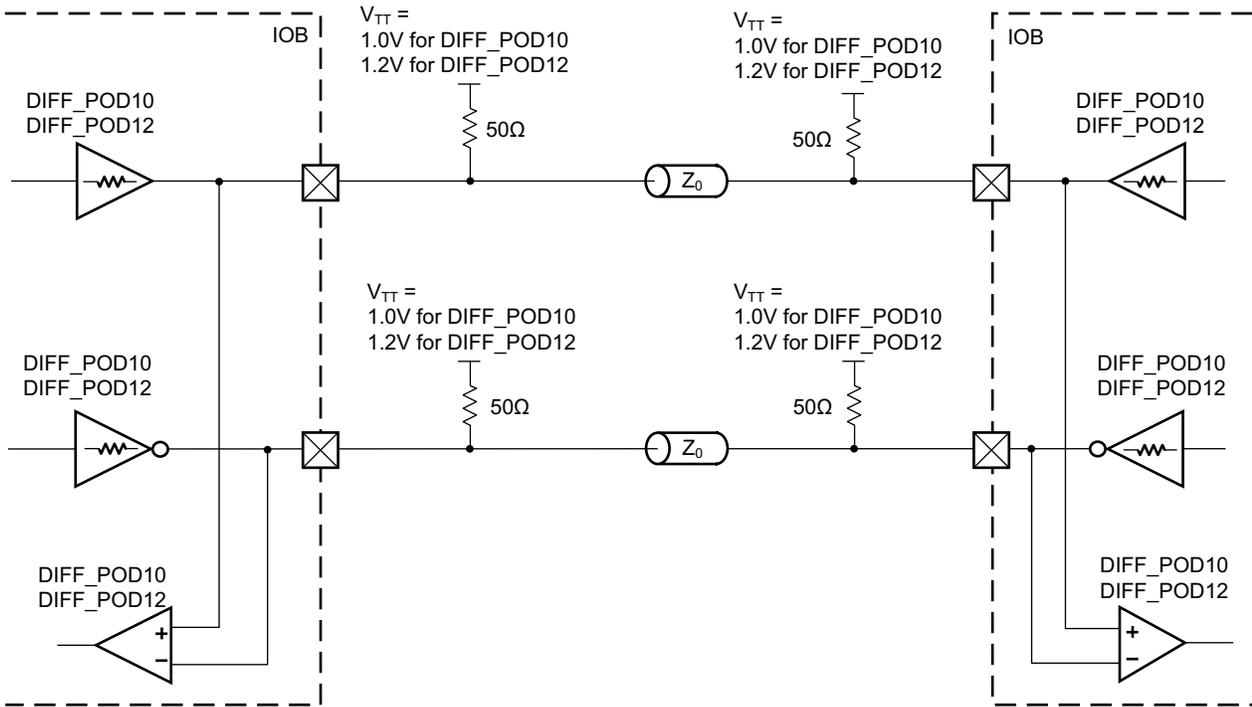


Figure 1-79: Differential POD with Unidirectional Signaling

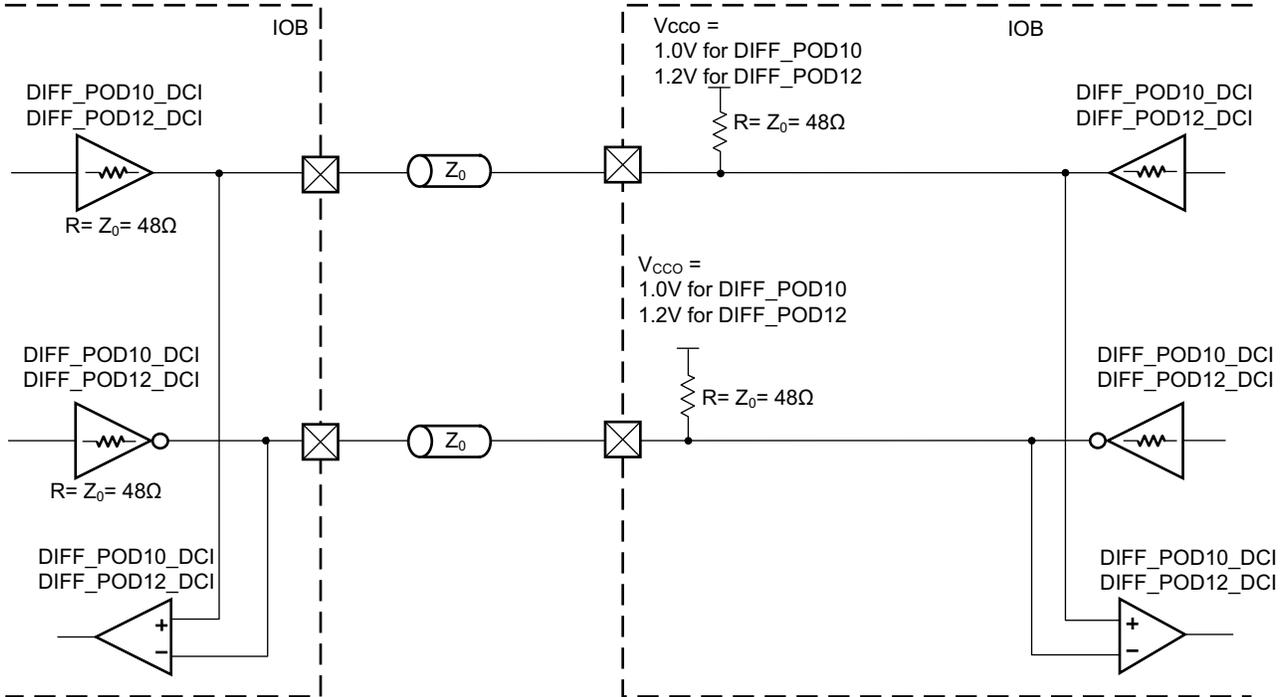
Figure 1-80 shows a sample circuit illustrating a termination technique for differential POD (1.0V or 1.2V) with bidirectional termination and with matched driver and receiver termination values. In a specific circuit, all drivers and receivers must be at the same voltage level (1.0V or 1.2V); they are not interchangeable (i.e., DIFF\_POD10\_DCI should only interface with DIFF\_POD10\_DCI).

External Termination



Driver is not 3-stated

Driver is 3-stated



X16140-022216

Figure 1-80: Differential Pod with Bidirectional Signaling

Table 1-51 lists the allowed attributes for POD I/O standards. Support is implied for primitives that are derivatives of the primitives listed in Table 1-51 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-51: POD Allowed Attributes

Attributes	IBUF/IBUFE3/IBUFDS/ IBUFDSE3		OBUF/OBUFT		IOBUF/IOBUFE3/IOBUFD3/ IOBUFDSE3	
	HP I/O		HP I/O		HP I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	POD10 DIFF_POD10		POD10 DIFF_POD10		POD10 DIFF_POD10	
SLEW	N/A		FAST MEDIUM SLOW	SLOW	FAST MEDIUM SLOW	SLOW
DQS_BIAS <sup>(5)</sup>	TRUE FALSE	FALSE	N/A		TRUE FALSE	FALSE
ODT	RTT_40, RTT_48 RTT_60, RTT_NONE	RTT_NONE	N/A		RTT_40, RTT_48 RTT_60, RTT_NONE <sup>(1)</sup>	RTT_NONE
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 <sup>(1)</sup>	RDRV_40_40
IOSTANDARD	POD10_DCI DIFF_POD10_DCI		POD10_DCI DIFF_POD10_DCI		POD10_DCI DIFF_POD10_DCI	
SLEW	N/A		FAST MEDIUM SLOW	SLOW	FAST MEDIUM SLOW	SLOW
DQS_BIAS <sup>(5)</sup>	TRUE FALSE	FALSE	N/A		TRUE FALSE	FALSE
ODT	RTT_40 RTT_48 RTT_60 <sup>(2)</sup>	RTT_40	N/A		RTT_40 RTT_48 RTT_60 <sup>(1)(2)</sup>	RTT_40

Table 1-51: POD Allowed Attributes (Cont'd)

Attributes	IBUF/IBUFE3/IBUFDS/ IBUFDSE3		OBUF/OBUFT		IOBUF/IOBUFE3/IOBUFDS/ IOBUFDSE3	
	HP I/O		HP I/O		HP I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
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 <sup>(1)</sup>	RDRV_40_40
IOSTANDARD	POD12 DIFF_POD12		POD12 DIFF_POD12		POD12 DIFF_POD12	
SLEW	N/A		FAST MEDIUM SLOW <sup>(4)</sup>	SLOW	FAST MEDIUM SLOW <sup>(3)</sup>	SLOW
PRE_EMPHASIS	N/A		RDRV_240 RDRV_NONE <sup>(4)</sup>	RDRV_NONE	RDRV_240 RDRV_NONE <sup>(3)</sup>	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
OFFSET_CNTRL	CNTRL_NONE FABRIC	CNTRL_NONE	N/A		CNTRL_NONE FABRIC	CNTRL_NONE
DQS_BIAS <sup>(5)</sup>	TRUE FALSE	FALSE	N/A		TRUE FALSE	FALSE
ODT	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE	N/A		RTT_40 RTT_48 RTT_60 (RTT_NONE) <sup>(3)</sup>	RTT_NONE
OUTPUT_ IMPEDANCE	N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60 <sup>(4)</sup>	RDRV_40_40	RDRV_40_40 RDRV_48_48 RDRV_60_60 <sup>(3)</sup>	RDRV_40_40

Table 1-51: POD Allowed Attributes (Cont'd)

Attributes	IBUF/IBUFE3/IBUFDS/ IBUFDSE3		OBUF/OBUFT		IOBUF/IOBUFE3/IOBUFDSD/ IOBUFDSE3	
	HP I/O		HP I/O		HP I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	POD12_DCI DIFF_POD12_DCI		POD12_DCI DIFF_POD12_DCI		POD12_DCI DIFF_POD12_DCI	
SLEW	N/A		FAST MEDIUM SLOW <sup>(4)</sup>	SLOW	FAST MEDIUM SLOW <sup>(3)</sup>	SLOW
PRE_EMPHASIS <sup>(6)</sup>	N/A		RDRV_240 RDRV_NONE <sup>(4)</sup>	RDRV_NONE	RDRV_240 RDRV_NONE <sup>(3)</sup>	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
OFFSET_CNTRL	CNTRL_NONE FABRIC	CNTRL_NONE	N/A		CNTRL_NONE FABRIC	CNTRL_NONE
DQS_BIAS <sup>(5)(7)</sup>	TRUE FALSE	FALSE	N/A		TRUE FALSE	FALSE
ODT	RTT_40 RTT_48 RTT_60 <sup>(2)</sup>	RTT_40	N/A		RTT_40 RTT_48 RTT_60 <sup>(2)(3)</sup>	RTT_40
OUTPUT_ IMPEDANCE	N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60 <sup>(4)</sup>	RDRV_40_40	RDRV_40_40 RDRV_48_48 RDRV_60_60 <sup>(3)</sup>	RDRV_40_40

**Notes:**

1. The allowed bidirectional configuration combinations for driver output impedance (OUTPUT\_IMPEDANCE) and ODT are listed in [Table 1-37](#).
2. ODT = RTT\_NONE is not a valid setting for DCI I/O standards.
3. The allowed bidirectional configuration combinations for driver output impedance (OUTPUT\_IMPEDANCE), ODT, and PRE\_EMPHASIS are listed in [Table 1-52](#).
4. The combinations allowed of driver output impedance (OUTPUT\_IMPEDANCE) and PRE\_EMPHASIS are listed in [Table 1-53](#).
5. Only applicable to DIFF\_POD I/O standards.
6. This attribute has to be used in conjunction with ENABLE\_PRE\_EMPHASIS to enable the pre-emphasis function.
7. This is read-only on the primitive. The DQS\_BIAS attribute is set on the I/O port rather than the primitive.

Table 1-52 and Table 1-53 lists the allowed combinations of attributes for POD I/O standards.

**Table 1-52: Allowed Combinations of OUTPUT\_IMPEDANCE, ODT, and PRE\_EMPHASIS**

OUTPUT_IMPEDANCE	SLEW	ODT	PRE_EMPHASIS
RDRV_40_40 (40Ω)	SLOW, MEDIUM, FAST	RTT_40	RDRV_NONE
RDRV_40_40 (40Ω)	SLOW, MEDIUM, FAST	RTT_60	RDRV_NONE
RDRV_40_40 (40Ω)	SLOW, MEDIUM, FAST	RTT_NONE	RDRV_NONE
RDRV_48_48 (48Ω)	SLOW, MEDIUM, FAST	RTT_48	RDRV_NONE
RDRV_48_48 (48Ω)	SLOW, MEDIUM, FAST	RTT_NONE	RDRV_NONE
RDRV_60_60 (60Ω)	SLOW, MEDIUM, FAST	RTT_40	RDRV_NONE
RDRV_60_60 (60Ω)	SLOW, MEDIUM, FAST	RTT_60	RDRV_NONE
RDRV_60_60 (60Ω)	SLOW, MEDIUM, FAST	RTT_NONE	RDRV_NONE
RDRV_40_40 (40Ω)	FAST	RTT_40	RDRV_240
RDRV_40_40 (40Ω)	FAST	RTT_60	RDRV_240
RDRV_40_40 (40Ω)	FAST	RTT_NONE	RDRV_240

**Table 1-53: Allowed Combinations of OUTPUT\_IMPEDANCE and PRE\_EMPHASIS**

OUTPUT_IMPEDANCE	SLEW	PRE_EMPHASIS
RDRV_40_40 (40Ω)	SLOW, MEDIUM, FAST	RDRV_NONE
RDRV_48_48 (48Ω)	SLOW, MEDIUM, FAST	RDRV_NONE
RDRV_60_60 (60Ω)	SLOW, MEDIUM, FAST	RDRV_NONE
RDRV_40_40 (40Ω)	FAST	RDRV_240

## LVDS and LVDS\_25

Low-voltage differential signaling (LVDS) is a powerful high-speed interface in many system applications. The I/Os are designed to be compatible with the EIA/TIA electrical specifications for LVDS system and board design. With the use of an LVDS current-mode driver in the IOBs and the optional internal differential termination feature, the need for external source termination in point-to-point applications is eliminated. UltraScale devices provide a flexible solution for creating an LVDS design.

The LVDS I/O standard is only available in the HP I/O banks. It requires a  $V_{CC0}$  to be powered at 1.8V for outputs and for inputs when the optional internal differential termination is implemented.

- `DIFF_TERM_ADV = TERM_100`
- `DIFF_TERM = TRUE`

The LVDS\_25 I/O standard is available in the HR I/O banks. It requires a  $V_{CC0}$  to be powered at 2.5V for outputs and for inputs when the optional internal differential termination is implemented.

- `DIFF_TERM_ADV = TERM_100`
- `DIFF_TERM = TRUE`

*Table 1-54: Available I/O Bank Type*

HR	HP
Available for LVDS_25 only	Available for LVDS only

### ***Transmitter Termination***

The LVDS transmitter does not require any external termination. [Table 1-55](#) lists the allowed attributes corresponding to the LVDS current-mode drivers. LVDS current-mode drivers are a true current source and produce the proper (EIA/TIA compliant) LVDS signal.

**Receiver Termination**

Figure 1-81 is an example of differential termination for an LVDS or LVDS\_25 receiver on a board with 50Ω transmission lines.

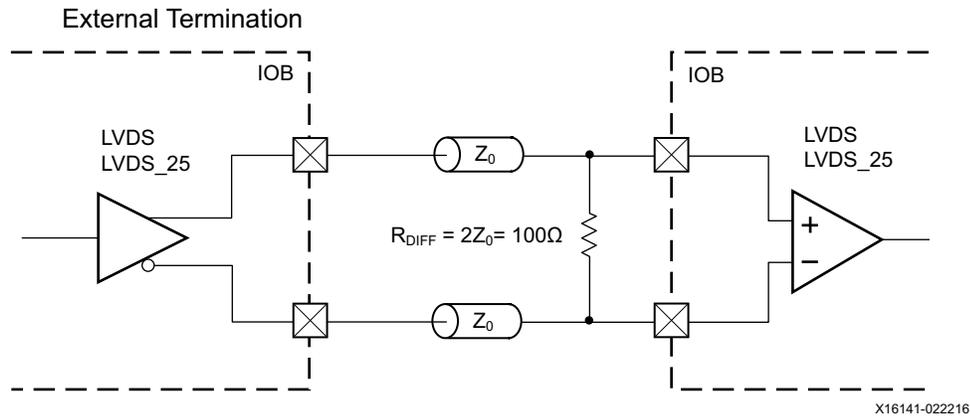


Figure 1-81: LVDS or LVDS\_25 Receiver Termination

Figure 1-82 is an example of internal differential termination for an LVDS or LVDS\_25 receiver on a board with 50Ω transmission lines.

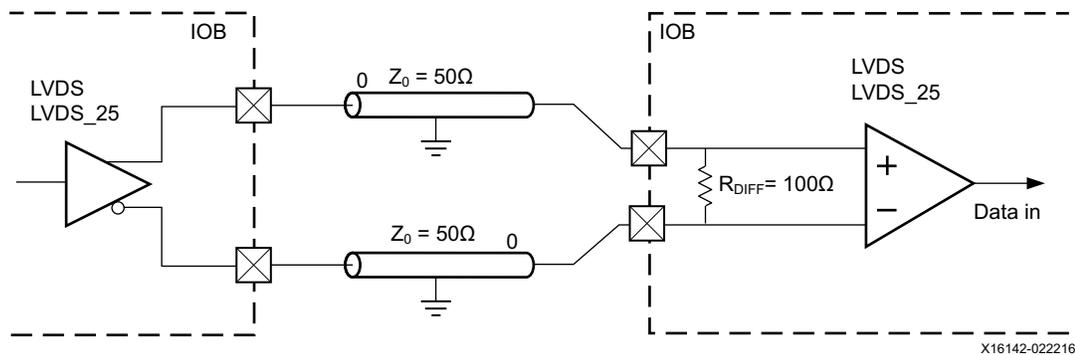


Figure 1-82: LVDS or LVDS\_25 with DIFF\_TERM Receiver Termination

Table 1-55 lists the allowed attributes for the LVDS I/O standards. Support is implied for primitives that are derivatives of the primitives listed in Table 1-55 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-55: Allowed Attributes for the LVDS I/O Standards

Attributes	IBUFDS				OBUFDS			
	HP I/O		HR I/O		HP I/O		HR I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	LVDS		LVDS_25		LVDS		LVDS_25	
DQS_BIAS <sup>(1)</sup>	TRUE FALSE <sup>(2)(3)</sup>	FALSE	N/A		N/A		N/A	
EQUALIZATION	EQ_LEVEL0 EQ_LEVEL1 EQ_LEVEL2 EQ_LEVEL3 EQ_LEVEL4 EQ_NONE <sup>(2)</sup>	EQ_NONE	EQ_LEVEL0 EQ_LEVEL1 EQ_LEVEL2 EQ_LEVEL3 EQ_LEVEL4 EQ_LEVEL0_DC_BIAS EQ_LEVEL1_DC_BIAS EQ_LEVEL2_DC_BIAS EQ_LEVEL3_DC_BIAS EQ_LEVEL4_DC_BIAS EQ_NONE <sup>(5)</sup>	EQ_NONE	N/A		N/A	
LVDS_PRE_EMPHASIS <sup>(6)</sup>	N/A		N/A		TRUE <sup>(4)</sup> FALSE	FALSE	TRUE <sup>(4)</sup> FALSE	FALSE
DIFF_TERM	TRUE FALSE	FALSE	TRUE FALSE	FALSE	N/A		N/A	
DIFF_TERM_ADV	TERM_100 TERM_NONE	TERM_NONE	TERM_100 TERM_NONE	TERM_NONE	N/A		N/A	

**Notes:**

1. The DQS\_BIAS attribute is set on the I/O port rather than the primitive.
2. The allowed combinations of DQS\_BIAS and EQUALIZATION are listed in [Table 1-56](#).
3. DQS\_BIAS = TRUE is only allowed in AC coupled applications to provide a bias level to  $V_{CCO}/2$  on both P and N sides of the input pin.
4. LVDS\_PRE\_EMPHASIS = TRUE is only supported in AC coupled applications.
5. Allowed values of equalization for AC coupled and DC coupled interface are listed in [Table 1-57](#).
6. This attribute must be used in conjunction with ENABLE\_PRE\_EMPHASIS to enable the pre-emphasis function.

Table 1-56: Allowed Combinations of DQS\_BIAS and EQUALIZATION (HP I/O Banks)

Coupling	DQS_BIAS	Equalization
AC coupling	FALSE or TRUE	EQ_LEVEL0, EQ_LEVEL1, EQ_LEVEL2, EQ_LEVEL3, EQ_LEVEL4
DC coupling	FALSE	EQ_NONE

Table 1-57: HR I/O Bank Equalization

Interface	Equalization
AC coupling (external bias)	EQ_LEVEL0, EQ_LEVEL1, EQ_LEVEL2, EQ_LEVEL3, EQ_LEVEL4
AC coupling (internal bias)	EQ_LEVEL0_DC_BIAS, EQ_LEVEL1_DC_BIAS, EQ_LEVEL2_DC_BIAS, EQ_LEVEL3_DC_BIAS, EQ_LEVEL4_DC_BIAS
DC coupling	EQ_NONE

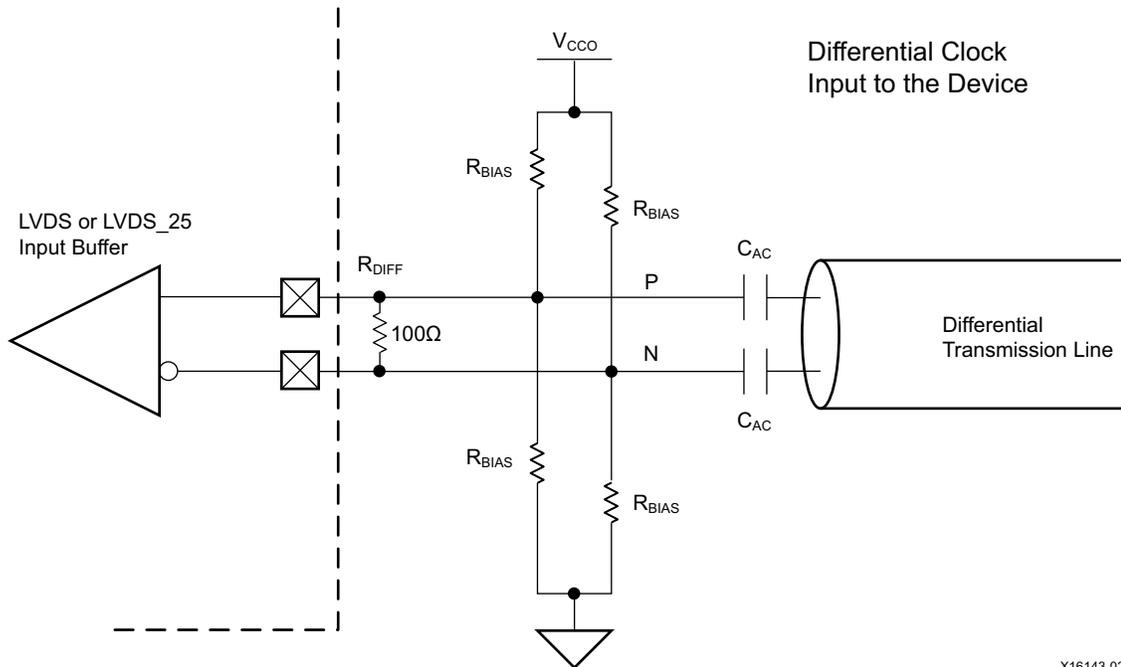
It is acceptable to have differential inputs such as LVDS and LVDS\_25 in I/O banks that are powered at voltage levels other than the nominal voltages required for the outputs of those standards (1.8V for LVDS outputs, and 2.5V for LVDS\_25 outputs). However, these criteria must be met:

- The optional internal differential termination is not used.
  - DIFF\_TERM\_ADV = TERM\_NONE
  - DIFF\_TERM = FALSE (default).
- The differential signals at the input pins meet the  $V_{IN}$  requirements in the *Recommended Operating Conditions* table of the specific UltraScale device data sheet [Ref 2].
- The differential signals at the input pins meet the  $V_{DIFF}$  (min) requirements in the corresponding LVDS or LVDS\_25 DC specifications tables of the specific UltraScale device data sheet [Ref 2].

One way to accomplish this criteria is to use an external circuit that both AC-couples and DC-biases the input signals. Figure 1-83 shows an example circuit for providing an AC-coupled and DC-biased circuit for a differential clock input.  $R_{DIFF}$  provides the  $100\Omega$  differential receiver termination because the internal DIFF\_TERM\_ADV = TERM\_NONE or DIFF\_TERM = FALSE. To maximize the input noise margin, all  $R_{BIAS}$  resistors should be the same value, essentially creating a  $V_{ICM}$  level of  $V_{BIAS}/2$ .  $V_{BIAS}$  should be a 1.8V source (typically  $V_{CC0}$  or  $V_{CCAUX}$ ) to ensure the input common-mode voltage for AC-coupled signals is maintained. Resistors in the 1K–100K $\Omega$  range are recommended. 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 UltraScale device data sheets [Ref 2] for the range of the bias voltage to be used with the receiver with and without equalization. Although the AC coupling mentioned in this section is intended for LVDS signaling, there are many alternate ways to provide bias and termination. For example, IOSTANDARDS like DIFF\_SSTL and DCI split termination can often be used to provide both internal termination and bias to an AC-coupled signal at the cost of higher current draw through the termination network.

In UltraScale device HP I/O banks, there is an option to use internal bias voltage (DQS\_BIAS) in AC-coupled LVDS applications. In such a configuration, EQUALIZATION must be set to EQ\_LEVEL0 (1, 2, 3, or 4) for the correct operation, even though EQ\_LEVEL0 does not provide equalization. When designing with Vivado Design Suite, the simulation behavior of the DQS\_BIAS feature is not modeled when DQS\_BIAS is used for DC biasing with an AC-coupled LVDS standard. When an input is 3-stated and the DQS\_BIAS is set to TRUE for an LVDS input, the input to the general interconnect is an X in the hardware. Simulation models this condition as an input of 0 to the general interconnect.

In HR I/O banks there is an option to use internal bias voltage in AC-coupled LVDS applications by setting the attribute value, EQUALIZATION = EQ\_LEVEL0\_DC\_BIAS (when EQUALIZATION is not required) or EQ\_LEVEL1/2/3/4\_DC\_BIAS. In DC-coupled applications, EQUALIZATION must be set to EQ\_NONE.



X16143-022216

Figure 1-83: Example Circuit for AC-Coupled and External DC-Biased Differential Clock Input

## RSDS

Table 1-58: Available I/O Bank Type

HR	HP
Available	N/A

Reduced-swing differential signaling (RSDS) is similar to an LVDS high-speed interface using differential signaling. RSDS has a similar implementation to LVDS\_25 and is only intended for point-to-point applications. RSDS is only available in HR I/O banks and requires a  $V_{CCO}$  voltage level of 2.5V. The IOSTANDARD is called RSDS\_25.

Table 1-59 summarizes the allowed attributes for the RSDS I/O standard. Support is implied for primitives that are derivatives of the primitives listed in Table 1-59 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-59: Allowed Attributes for the RSDS I/O Standard

Attributes	Primitives		
	IBUFDS		OBUFDS or OBUFTDS
	Allowed Values	Default	
IOSTANDARD	RSDS_25		
DIFF_TERM	TRUE FALSE	FALSE	N/A
DIFF_TERM_ADV	TERM_NONE TERM_100	TERM_NONE	N/A

## Mini-LVDS

Table 1-60: Available I/O Bank Type

HR	HP
Available	N/A

Mini low-voltage differential signaling (Mini-LVDS) is a serial, intra-flat panel differential I/O standard that serves as an interface between the timing control function and an LCD source driver. Mini-LVDS inputs require a parallel-termination resistor, either by using a discrete resistor on the PCB, or by using the DIFF\_TERM\_ADV or DIFF\_TERM attributes to enable internal termination. Mini-LVDS is only available in HR I/O banks and requires a  $V_{CCO}$  voltage level of 2.5V. The IOSTANDARD is called MINI\_LVDS\_25.

Table 1-61 summarizes the allowed attributes for the Mini-LVDS I/O standard. Support is implied for primitives that are derivatives of the primitives listed in Table 1-61 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-61: Allowed Attributes of the Mini-LVDS I/O Standard

Attributes	Primitives		
	IBUFDS		OBUFDS or OBUFTDS
	Allowed Values	Default	
IOSTANDARD	MINI_LVDS_25		
DIFF_TERM	TRUE FALSE	FALSE	N/A
DIFF_TERM_ADV	TERM_NONE TERM_100	TERM_NONE	N/A

## PPDS

Table 1-62: Available I/O Bank Type

HR	HP
Available	N/A

Point-to-point differential signaling (PPDS) is a differential I/O standard for next-generation LCD interface row and column drivers. PPDS inputs require a parallel-termination resistor, either through the use of a discrete resistor on the PCB, or by using the DIFF\_TERM\_ADV or DIFF\_TERM attributes to enable internal termination. PPDS is only available in HR I/O banks and requires a  $V_{CC0}$  voltage level of 2.5V. The IOSTANDARD is called PPDS\_25.

Table 1-63 summarizes the allowed attributes for the PPDS I/O standard. Support is implied for primitives that are derivatives of the primitives listed in Table 1-63 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-63: Allowed Attributes for the PPDS I/O Standard

Attributes	Primitives		
	IBUFDS		OBUFDS or OBUFTDS
	Allowed Values	Default	
IOSTANDARD	PPDS_25		
DIFF_TERM	TRUE FALSE	FALSE	N/A
DIFF_TERM_ADV	TERM_NONE TERM_100	TERM_NONE	N/A

## TMDS

Table 1-64: Available I/O Bank Type

HR	HP
Available	N/A

Transition minimized differential signaling (TMDS) is a differential I/O standard for transmitting high-speed serial data used by the DVI and HDMI™ video interfaces. The TMDS standard requires external 50Ω pull-up resistors to 3.3V on the inputs. TMDS inputs do not require parallel input termination resistors. TMDS is only available in HR I/O banks and requires a  $V_{CC0}$  voltage level of 3.3V. The IOSTANDARD is called TMDS\_33.

Table 1-65 summarizes the allowed attributes for the TMDS I/O standard. Support is implied for primitives that are derivatives of the primitives listed in Table 1-65 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-65: Allowed Attributes for the TMDS I/O Standard

Attributes	Primitives	
	IBUFDS	OBUFDS or OBUFTDS
IOSTANDARD	TMDS_33	

## BLVDS

Table 1-66: Available I/O Bank Type

HR	HP
Available	N/A

Because LVDS is intended for point-to-point applications, bus LVDS (BLVDS) is not an EIA/TIA standard implementation and requires careful adaptation of I/O and PCB layout design rules. The primitive supplied in the Vivado Design Suite library for bidirectional LVDS does not use the LVDS current-mode driver, instead, it uses complementary single-ended differential drivers. Therefore, source termination is required. BLVDS is only available in HR I/O banks and requires a  $V_{CC0}$  voltage level of 2.5V. The IOSTANDARD is called BLVDS\_25.

Table 1-67 summarizes the allowed attributes for the BLVDS I/O standard. Support is implied for primitives that are derivatives of the primitives listed in Table 1-67 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-67: Allowed Attributes for the BLVDS I/O Standard

Attributes	Primitives	
	IBUFDS	OBUFDS, OBUFTDS, IOBUFDS, or IOBUFDS_DIFF_OUT
IOSTANDARD	BLVDS_25	

Figure 1-84 shows the BLVDS transmitter termination.

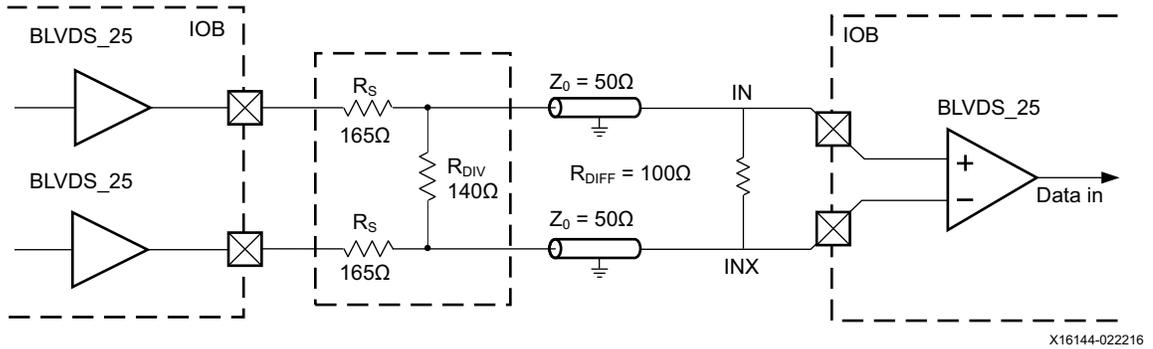


Figure 1-84: BLVDS Transmitter Termination

## SUB\_LVDS

Table 1-68: Available I/O Bank Type

HR	HP
Available for SUB_LVDS	Available for SUB_LVDS

Table 1-69 lists the allowed attributes for the SUB\_LVDS I/O standards. Support is implied for primitives that are derivatives of the primitives listed in Table 1-69 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-69: Allowed Attributes for the SUB\_LVDS I/O Standards

Attributes	Primitives		
	IBUFDS		OBUFDS or OBUFTDS
	Allowed Values	Default	
IOSTANDARD	SUB_LVDS		
DIFF_TERM	TRUE FALSE	FALSE	N/A
DIFF_TERM_ADV	TERM_NONE TERM_100	TERM_NONE	N/A

## SLVS\_400

SLVS\_400 is supported in HR I/O banks as SLVS\_400\_25 and in HP I/O banks as SLVS\_400\_18. SLVS\_400 is only supported in receivers.

Table 1-70: Available I/O Bank Type

HR	HP
Available for SLVS_400_25 only	Available for SLVS_400_18 only

Table 1-71 lists the allowed attributes for the SLVS\_400 I/O standards. Support is implied for primitives that are derivatives of the primitives listed in Table 1-71 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-71: Allowed Attributes for the SLVS\_400 I/O Standards

Attributes	Primitives		OBUFDS or OBUFTDS
	IBUFDS		
	Allowed Values	Default	
IOSTANDARD	SLVS_400_25 for HR I/O banks SLVS_400_18 for HP I/O banks		N/A
DIFF_TERM	TRUE FALSE	FALSE	N/A
DIFF_TERM_ADV	TERM_NONE TERM_100	TERM_NONE	N/A

## LVPECL

LVPECL is supported only in HR I/O banks, and is only for receivers.

Table 1-72: Available I/O Bank Type

HR	HP
Available	N/A

Table 1-73 lists the allowed attributes for the LVPECL I/O standard. Support is implied for primitives that are derivatives of the primitives listed in Table 1-73 (for example: \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

Table 1-73: Allowed Attributes for the LVPECL I/O Standard

Attributes	Primitives	
	IBUFDS	OBUFDS or OBUFTDS
IOSTANDARD	LVPECL (HR I/O banks only)	N/A

## MIPI D-PHY

The MIPI D-PHY standard MIPI\_DPHY\_DCI is intended for use in mobile devices including cameras, displays, and unified protocol interfaces. This standard is only supported in the HP I/O banks of the Virtex UltraScale+, Kintex UltraScale+, Artix UltraScale+, and Zynq UltraScale+ devices. Support for this standard in UltraScale devices is in adherence to the MIPI Alliance interface specifications.



**IMPORTANT:** *MIPI\_DPHY\_DCI, as with other DCI standards, requires a 240Ω external resistor at the VRP Pin. This standard does leverage calibration on the LP driver even though the values for the OUTPUT\_IMPEDANCE attribute are not configurable.*

### MIPI\_DPHY\_DCI I/O Standard

Table 1-74: Available I/O Bank Type

HR	HP
N/A	Virtex UltraScale+, Kintex UltraScale+, Artix UltraScale+, and Zynq UltraScale+ devices (only)

Table 1-75 lists the allowed attributes for the MIPI\_DPHY\_DCI I/O standard.

Table 1-75: Allowed Attributes for the MIPI\_DPHY\_DCI Standard

Attributes	IBUFDS_DPHY			OBUFDS_DPHY		
	HP I/O		HR I/O	HP I/O		HR I/O
	Allowed Values	Default		Allowed Values	Default	
IOSTANDARD	MIPI_DPHY_DCI		N/A	MIPI_DPHY_DCI		N/A
SLEW	N/A	N/A	N/A	N/A	N/A	N/A
DIFF_TERM_ADV	TERM_100 TERM_NONE	TERM_NONE	N/A	N/A	N/A	N/A
DIFF_TERM	TRUE FALSE	FALSE	N/A	N/A	N/A	N/A
LVDS_PRE_EMPHASIS	N/A	N/A	N/A	TRUE FALSE	FALSE	N/A
EQUALIZATION	EQ_LEVEL 1 EQ_LEVEL 2 EQ_LEVEL 3 EQ_LEVEL 4	EQ_NONE	N/A	N/A	N/A	N/A

Starting with Vivado Design Suite version 2019.1.1, the LVDS\_PRE\_EMPHASIS attribute was made available to MIPI\_DPHY\_DCI. Setting the LVDS\_PRE\_EMPHASIS attribute to TRUE increases the maximum operating speed for the MIPI\_DPHY\_DCI I/O. For lower-speed behavior in equivalent earlier versions of Vivado tools, LVDS\_PRE\_EMPHASIS should be set to FALSE.

Also in Vivado Design Suite 2019.1.1, EQUALIZATION is supported when operating at enhanced speeds. In MIPI\_DPHY\_DCI, EQUALIZATION is supported at levels of EQ\_LEVEL1, EQ\_LEVEL2, EQ\_LEVEL3, and EQ\_LEVEL4. For details on enhanced speed levels, refer to the appropriate UltraScale+ device data sheet [Ref 2].

## Internal Differential Termination Behavior in Differential I/O Standards

The internal differential termination (100Ω) is turned on in the driver and bidirectional modes of operation (default) for these I/O standards: LVDS, LVDS\_25, SLVS\_400\_18, SLVS\_400\_25, SUB\_LVDS, PPDS\_25, RSDS\_25, and MINI\_LVDS\_25. The behavior of internal differential termination in bidirectional mode and the input or output modes of operation are listed in [Table 1-76](#).

**Table 1-76: Internal Differential Termination Behavior In Differential I/O Standards**

Primitive <sup>(1)</sup>	Driving	3-state/Receiving
OBUFDS	Internal differential termination is ON	N/A
OBUFTDS	Internal differential termination is ON	Internal differential termination is ON.
IBUFDS	N/A	When DIFF_TERM = TRUE or DIFF_TERM_ADV = TERM_100, then internal differential termination is ON. When DIFF_TERM = FALSE or DIFF_TERM_ADV = TERM_NONE, then internal differential termination is OFF.
IOBUFDS	Internal differential termination is ON	Internal differential termination is ON, irrespective of the DIFF_TERM or DIFF_TERM_ADV attributes. DIFF_TERM = TRUE or FALSE DIFF_TERM_ADV = TERM_100 or TERM_NONE

**Notes:**

- Support is implied for primitives that are derivatives of the primitives listed (for example, \*\_DIFF\_OUT, \*\_DCIEN, \*\_IBUFDISABLE, or \*\_INTERMDISABLE). Refer to the [SelectIO Interface Primitives](#) section for all supported derivatives.

## Rules for Combining I/O Standards in the Same Bank

The following rules must be obeyed to combine different input, output, and bidirectional standards in the same bank:

1. **Combining output standards only.** Output standards with the same output  $V_{CCO}$  requirement can be combined in the same bank.

*Compatible example:*

SSTL15\_I and LVDCI\_15 outputs

*Incompatible example:*

SSTL15 (output  $V_{CCO} = 1.5V$ ) and LVCMOS18 (output  $V_{CCO} = 1.8V$ ) outputs

Only two true differential output I/O standards can be combined in HR I/O banks. LVDS\_25 with LVDS\_PRE\_EMPHASIS = FALSE (default) and LVDS\_25 with LVDS\_PRE\_EMPHASIS = TRUE are considered as two different true-differential output standards in HR I/O banks.

Only one true differential output I/O standard can be used in HP I/O banks. LVDS with LVDS\_PRE\_EMPHASIS = FALSE (default) and LVDS with LVDS\_PRE\_EMPHASIS = TRUE are considered as two different true-differential output standards and cannot be combined within the same HP I/O bank.

2. **Combining input standards only.** Input standards with the same  $V_{CCO}$  and  $V_{REF}$  requirements can be combined in the same bank.

*Compatible example:*

LVCMOS15 and HSTL\_II inputs

*Incompatible example:*

LVCMOS15 (input  $V_{CCO} = 1.5V$ ) and LVCMOS18 (input  $V_{CCO} = 1.8V$ ) inputs

*Incompatible example:*

HSTL\_I\_DCI\_18 ( $V_{REF} = 0.9V$ ) and HSTL\_I\_DCI ( $V_{REF} = 0.75V$ ) inputs

3. **Combining input standards and output standards.** Input standards and output standards with the same  $V_{CCO}$  requirement can be combined in the same bank.

*Compatible example:*

LVDS\_25 output and LVCMOS25 input

*Incompatible example:*

LVDS\_25 output (output  $V_{CCO} = 2.5V$ ) and HSTL\_I input (input  $V_{CCO} = 1.5V$ )

4. **Combining bidirectional standards with input or output standards.** When combining bidirectional I/O with other standards, make sure the bidirectional standard can meet the first three rules.

The implementation tools enforce these design rules.

Table 1-77, summarizes the  $V_{CCO}$  and  $V_{REF}$  requirements for each supported I/O standard. For more detailed DC specifications, including the recommended operating ranges of the supplies for each supported I/O standard, see the specific UltraScale device data sheet [Ref 2].

Table 1-77:  $V_{CCO}$  and  $V_{REF}$  Requirements for Each Supported I/O Standard

I/O Standard	I/O Bank Availability	$V_{CCO}$ (V)			$V_{REF}$ (V)
		Output	Input	Input with DIFF_TERM_ADV and DIFF_TERM Support	Input
LVTTTL	HR	3.3	3.3	N/A	N/A
LVC MOS33	HR	3.3	3.3	N/A	N/A
LVC MOS25	HR	2.5	2.5	N/A	N/A
LVC MOS18	Both	1.8	1.8	N/A	N/A
LVC MOS15	Both	1.5	1.5	N/A	N/A
LVC MOS12	Both	1.2	1.2	N/A	N/A
HSUL_12	Both	1.2	1.2	N/A	0.60
LVDCI_18	HP	1.8	1.8	N/A	N/A
LVDCI_15	HP	1.5	1.5	N/A	N/A
HSUL_12_DCI	HP	1.2	1.2	N/A	0.60
HSLVDCI_18	HP	1.8	1.8	N/A	0.90
HSLVDCI_15	HP	1.5	1.5	N/A	0.75
HSTL_I	Both	1.5	1.5	N/A	0.75
HSTL_II	HR	1.5	1.5	N/A	0.75
HSTL_I_DCI	HP	1.5	1.5	N/A	0.75
HSTL_I_18	Both	1.8	1.8	N/A	0.90
HSTL_II_18	HR	1.8	1.8	N/A	0.90
HSTL_I_DCI_18	HP	1.8	1.8	N/A	0.90
HSTL_I_12	HP	1.2	1.2	N/A	0.60
HSTL_I_DCI_12	HP	1.2	1.2	N/A	0.60
SSTL18_I	Both	1.8	1.8	N/A	0.90
SSTL18_II	HR	1.8	1.8	N/A	0.90

Table 1-77:  $V_{CCO}$  and  $V_{REF}$  Requirements for Each Supported I/O Standard (Cont'd)

I/O Standard	I/O Bank Availability	$V_{CCO}$ (V)			$V_{REF}$ (V)
		Output	Input	Input with DIFF_TERM_ADV and DIFF_TERM Support	Input
SSTL15	Both	1.5	1.5	N/A	0.75
SSTL15_R	HR	1.5	1.5	N/A	0.75
SSTL135	Both	1.35	1.35	N/A	0.675
SSTL135_R	HR	1.35	1.35	N/A	0.675
SSTL12	Both	1.2	1.2	N/A	0.60
SSTL18_I_DCI	HP	1.8	1.8	N/A	0.90
SSTL15_DCI	HP	1.5	1.5	N/A	0.75
SSTL135_DCI	HP	1.35	1.35	N/A	0.675
SSTL12_DCI	HP	1.2	1.2	N/A	0.60
DIFF_HSTL_I	Both	1.5	1.5 <sup>(2)</sup>	N/A	N/A
DIFF_HSTL_II	HR	1.5	1.5 <sup>(2)</sup>	N/A	N/A
DIFF_HSTL_I_18	Both	1.8	1.8 <sup>(2)</sup>	N/A	N/A
DIFF_HSTL_II_18	HR	1.8	1.8 <sup>(2)</sup>	N/A	N/A
DIFF_SSTL18_I	Both	1.8	1.8 <sup>(2)</sup>	N/A	N/A
DIFF_SSTL18_II	HR	1.8	1.8 <sup>(2)</sup>	N/A	N/A
DIFF_SSTL15	Both	1.5	1.5 <sup>(2)</sup>	N/A	N/A
DIFF_SSTL15_R	HR	1.5	1.5 <sup>(2)</sup>	N/A	N/A
DIFF_SSTL135	Both	1.35	1.35 <sup>(2)</sup>	N/A	N/A
DIFF_SSTL135_R	HR	1.35	1.35 <sup>(2)</sup>	N/A	N/A
DIFF_SSTL12	Both	1.2	1.2 <sup>(2)</sup>	N/A	N/A
DIFF_HSUL_12	Both	1.2	1.2 <sup>(3)</sup>	N/A	N/A
DIFF_HSTL_I_DCI	HP	1.5	1.5	N/A	N/A
DIFF_HSTL_I_DCI_18	HP	1.8	1.8	N/A	N/A
DIFF_SSTL18_I_DCI	HP	1.8	1.8	N/A	N/A
DIFF_SSTL15_DCI	HP	1.5	1.5	N/A	N/A
DIFF_SSTL135_DCI	HP	1.35	1.35	N/A	N/A
DIFF_SSTL12_DCI	HP	1.2	1.2	N/A	N/A
DIFF_HSUL_12_DCI	HP	1.2	1.2	N/A	N/A
BLVDS_25	HR	2.5	Any	N/A	N/A
LVDS_25	HR	2.5 <sup>(5)</sup>	2.5 <sup>(1)</sup>	2.5	N/A
RSDS_25	HR	2.5 <sup>(5)</sup>	2.5 <sup>(1)</sup>	2.5	N/A
TMDS_33	HR	3.3	Any	N/A	N/A
MINI_LVDS_25	HR	2.5 <sup>(5)</sup>	2.5 <sup>(1)</sup>	2.5	N/A

Table 1-77:  $V_{CCO}$  and  $V_{REF}$  Requirements for Each Supported I/O Standard (Cont'd)

I/O Standard	I/O Bank Availability	$V_{CCO}$ (V)			$V_{REF}$ (V)
		Output	Input	Input with DIFF_TERM_ADV and DIFF_TERM Support	Input
PPDS_25	HR	2.5 <sup>(5)</sup>	2.5 <sup>(1)</sup>	2.5	N/A
LVDS	HP	1.8	1.8 <sup>(1)</sup>	1.8	N/A
LVPECL	HR	N/A	Any	N/A	N/A
SLVS_400_18	HP	N/A	1.8 <sup>(1)</sup>	1.8	N/A
SLVS_400_25	HR	N/A	2.5 <sup>(1)</sup>	2.5	N/A
SUB_LVDS	Both	1.8	1.8 <sup>(1)</sup>	1.8	N/A
DIFF_HSTL_I_12	HP	1.2	1.2 <sup>(2)</sup>	N/A	N/A
DIFF_POD10	HP	1.0	1.0 <sup>(2)</sup>	N/A	N/A
DIFF_POD12	HP	1.2	1.2 <sup>(2)</sup>	N/A	N/A
DIFF_HSTL_I_DCI_12	HP	1.2	1.2	N/A	N/A
DIFF_POD10_DCI	HP	1.0	1.0	N/A	N/A
DIFF_POD12_DCI	HP	1.2	1.2	N/A	N/A
POD10	HP	1.0	1.0	N/A	0.70
POD12	HP	1.2	1.2	N/A	0.84
POD10_DCI	HP	1.0	1.0	N/A	0.70
POD12_DCI	HP	1.2	1.2	N/A	0.84
MIPI_DPHY_DCI	HP <sup>(4)</sup>	1.2	1.2	1.2	N/A

**Notes:**

- Differential inputs for these standards can be placed in banks with  $V_{CCO}$  levels that are different from the required level for outputs. Some important criteria to consider:
  - The optional internal differential termination is not used, DIFF\_TERM\_ADV = TERM\_NONE or DIFF\_TERM = FALSE (default value), unless the  $V_{CCO}$  voltage is at the level required for outputs.
  - The differential signals at the input pins meet the  $V_{IN}$  requirements in the *Recommended Operating Conditions* table of the specific UltraScale device data sheet [Ref 2].
  - The differential signals at the input pins meet the  $V_{IDIFF}$  and  $V_{ICM}$  requirements in the DC Specifications tables in the specific UltraScale device data sheet [Ref 2]. In some cases, to accomplish this it might be necessary to provide an external circuit to both AC-couple and DC-bias the pins.
- When on-die input termination is used (ODT is set to a value other than RTT\_NONE) or when DQS\_BIAS = TRUE, the  $V_{CCO}$  input voltage is as specified. When ODT = RTT\_NONE and DQS\_BIAS = FALSE, the  $V_{CCO}$  input voltage is any allowed voltage.
- When on-die input termination is used (ODT value set to something other than RTT\_NONE) or when DQS\_BIAS is set to TRUE, the  $V_{CCO}$  input voltage is 1.2V for HP I/O banks. In HR I/O banks, when DQS\_BIAS = FALSE, or in HP I/O banks when ODT = RTT\_NONE, the  $V_{CCO}$  input voltage is any allowed voltage.
- The DPHY\_DCI I/O standard is only supported in Virtex UltraScale+, Kintex UltraScale+, Artix UltraScale+, and Zynq UltraScale+ devices.
- If the  $V_{CCO}$  voltage exceeds 2.85V, the outputs are 3-stated. The device should always be operated within the recommended operating range as specified in the UltraScale device data sheets [Ref 2].

Table 1-78, summarizes the DRIVE and SLEW attribute options, bidirectional buffer availability, and DCI termination type for each supported I/O standard.

Table 1-78: Attribute Options, Bidirectional Buffer Availability, and DCI Termination Type

I/O Standard	I/O Bank Type	Output Slew				Output Drive				Bidirectional Buffers <sup>(1)</sup>	Termination Type <sup>(2)</sup>	
		HR I/O Banks		HP I/O Banks		HR I/O Banks		HP I/O Banks			Input	Output <sup>(3)</sup>
		Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default			
LVTTTL	HR	SLOW FAST	SLOW	N/A		4, 8, 12, 16	12	N/A		Yes	None	None
LVC MOS33	HR	SLOW FAST	SLOW	N/A		4, 8, 12, 16	12	N/A		Yes	None	None
LVC MOS25	HR	SLOW FAST	SLOW	N/A		4, 8, 12, 16	12	N/A		Yes	None	None
LVC MOS18	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	4, 8, 12, 16	12	2, 4, 6, 8, 12	12	Yes	None	None
LVC MOS15	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	4, 8, 12, 16	12	2, 4, 6, 8, 12	12	Yes	None	None
LVC MOS12	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	4, 8, 12	12	2, 4, 6, 8	12	Yes	None	None
HSUL_12	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Single <sup>(4)</sup>	Driver <sup>(4)</sup>
LVDCI_18	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	None	Driver
LVDCI_15	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	None	Driver
HSUL_12_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Single	Driver
HSLVDCI_18	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	None	Driver
HSLVDCI_15	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	None	Driver
HSTL_I	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
HSTL_II	HR	SLOW FAST	SLOW	N/A		N/A		N/A		Yes	Split	None

Table 1-78: Attribute Options, Bidirectional Buffer Availability, and DCI Termination Type (Cont'd)

I/O Standard	I/O Bank Type	Output Slew				Output Drive				Bidirectional Buffers <sup>(1)</sup>	Termination Type <sup>(2)</sup>	
		HR I/O Banks		HP I/O Banks		HR I/O Banks		HP I/O Banks			Input	Output <sup>(3)</sup>
		Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default			
HSTL_I_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
HSTL_I_18	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
HSTL_II_18	HR	SLOW FAST	SLOW	N/A		N/A		N/A		Yes	Split	None
HSTL_I_DCI_18	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
HSTL_I_12	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
HSTL_I_DCI_12	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split (UltraScale devices only)	Driver
SSTL18_I	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
SSTL18_II	HR	SLOW FAST	SLOW	N/A		N/A		N/A		Yes	Split	None
SSTL15	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
SSTL15_R	HR	SLOW FAST	SLOW	N/A		N/A		N/A		Yes	Split	None
SSTL135	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
SSTL135_R	HR	SLOW FAST	SLOW	N/A		N/A		N/A		Yes	Split	None
SSTL12	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
SSTL18_I_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
SSTL15_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver

Table 1-78: Attribute Options, Bidirectional Buffer Availability, and DCI Termination Type (Cont'd)

I/O Standard	I/O Bank Type	Output Slew				Output Drive				Bidirectional Buffers <sup>(1)</sup>	Termination Type <sup>(2)</sup>	
		HR I/O Banks		HP I/O Banks		HR I/O Banks		HP I/O Banks			Input	Output <sup>(3)</sup>
		Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default			
SSTL135_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
SSTL12_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
DIFF_HSTL_I	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
DIFF_HSTL_II	HR	SLOW FAST	SLOW	N/A		N/A		N/A		Yes	Split	None
DIFF_HSTL_I_18	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
DIFF_HSTL_II_18	HR	SLOW FAST	SLOW	N/A		N/A		N/A		Yes	Split	None
DIFF_SSTL18_I	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
DIFF_SSTL18_II	HR	SLOW FAST	SLOW	N/A		N/A		N/A		Yes	Split	None
DIFF_SSTL15	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
DIFF_SSTL15_R	HR	SLOW FAST	SLOW	N/A		N/A		N/A		Yes	Split	None
DIFF_SSTL135	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
DIFF_SSTL135_R	HR	SLOW FAST	SLOW	N/A		N/A		N/A		Yes	Split	None
DIFF_SSTL12	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
DIFF_HSUL_12	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Single <sup>(4)</sup>	Driver <sup>(4)</sup>
DIFF_HSTL_I_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver

Table 1-78: Attribute Options, Bidirectional Buffer Availability, and DCI Termination Type (Cont'd)

I/O Standard	I/O Bank Type	Output Slew				Output Drive				Bidirectional Buffers <sup>(1)</sup>	Termination Type <sup>(2)</sup>	
		HR I/O Banks		HP I/O Banks		HR I/O Banks		HP I/O Banks			Input	Output <sup>(3)</sup>
		Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default			
DIFF_HSTL_I_DCI_18	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A	N/A	Yes	Split	Driver
DIFF_SSTL18_I_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A	N/A	Yes	Split	Driver
DIFF_SSTL15_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A	N/A	Yes	Split	Driver
DIFF_SSTL135_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A	N/A	Yes	Split	Driver
DIFF_SSTL12_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A	N/A	Yes	Split	Driver
DIFF_HSUL_12_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A	N/A	Yes	Single	Driver
BLVDS_25	HR	N/A		N/A		N/A		N/A	N/A	Yes	None	None
LVDS_25	HR	N/A		N/A		N/A		N/A	N/A	Yes <sup>(5)</sup>	None	None
RSDS_25	HR	N/A		N/A		N/A		N/A	N/A	Yes <sup>(5)</sup>	None	None
TMDS_33	HR	N/A		N/A		N/A		N/A	N/A	Yes <sup>(5)</sup>	None	None
MINI_LVDS_25	HR	N/A		N/A		N/A		N/A	N/A	Yes <sup>(5)</sup>	None	None
PPDS_25	HR	N/A		N/A		N/A		N/A	N/A	Yes <sup>(5)</sup>	None	None
LVDS	HP	N/A		N/A		N/A		N/A	N/A	Yes <sup>(5)</sup>	None	None
LVPECL	HR	N/A		N/A		N/A		N/A	N/A	No	None	None
SLVS_400_18	HP	N/A		N/A		N/A		N/A	N/A	No	None	None
SLVS_400_25	HR	N/A		N/A		N/A		N/A	N/A	No	None	None
SUB_LVDS	Both	N/A		N/A		N/A		N/A	N/A	Yes <sup>(5)</sup>	None	None
DIFF_HSTL_I_12	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A	N/A	Yes	Split	Driver
DIFF_POD10	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A	N/A	Yes	Single	Driver
DIFF_POD12	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A	N/A	Yes	Single	Driver

Table 1-78: Attribute Options, Bidirectional Buffer Availability, and DCI Termination Type (Cont'd)

I/O Standard	I/O Bank Type	Output Slew				Output Drive				Bidirectional Buffers <sup>(1)</sup>	Termination Type <sup>(2)</sup>	
		HR I/O Banks		HP I/O Banks		HR I/O Banks		HP I/O Banks			Input	Output <sup>(3)</sup>
		Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default			
DIFF_HSTL_I_DCI_12	HP	N/A	SLOW MEDIUM FAST	SLOW	N/A	N/A	Yes	Split (UltraScale devices only)	Driver			
DIFF_POD10_DCI	HP	N/A	SLOW MEDIUM FAST	SLOW	N/A	N/A	Yes	Single	Driver			
DIFF_POD12_DCI	HP	N/A	SLOW MEDIUM FAST	SLOW	N/A	N/A	Yes	Single	Driver			
POD10	HP	N/A	SLOW MEDIUM FAST	SLOW	N/A	N/A	Yes	Single	Driver			
POD12	HP	N/A	SLOW MEDIUM FAST	SLOW	N/A	N/A	Yes	Single	Driver			
POD10_DCI	HP	N/A	SLOW MEDIUM FAST	SLOW	N/A	N/A	Yes	Single	Driver			
POD12_DCI	HP	N/A	SLOW MEDIUM FAST	SLOW	N/A	N/A	Yes	Single	Driver			
MIPI_DPHY_DCI	HP <sup>(6)</sup>	N/A	N/A	N/A	N/A	N/A	No	N/A	Driver			

**Notes:**

1. The bidirectional buffers column describes the I/O standards use of a bidirectional signal.
2. The DCI termination type column describes the type of termination available for the DCI I/O standards. Split refers to the split-termination resistors. Single refers to single resistor termination to  $V_{CC0}$ .
3. A value of DRIVER in this column only applies to HP I/O banks
4. INTERM = Single and OUTTERM = DRIVER for HP I/O banks, INTERM = NONE and OUTTERM = NONE for HR I/O banks.
5. The bidirectional configuration on these I/O standards is a fixed impedance structure optimized to 100Ω differential. They are intended to only be used in point-to-point transmissions that do not have turn around timing requirements. Use BLVDS\_25 for bus structures.
6. The MIPI\_DPHY\_DCI standard is only supported in Virtex UltraScale+, Kintex UltraScale+, Artix UltraScale+, and Zynq UltraScale+ devices.

## Simultaneous Switching Outputs

Due to package inductance, each part/package supports a limited number of simultaneous switching outputs (SSOs), particularly when using fast, high-drive outputs. Fast, high-drive outputs should only be used when required by the application.

The SSN predictor tool provides a way of analyzing the amount of noise margin on each I/O pin in a design based on information for the pin (the victim), as well as all other pins (aggressors) in the design. The tool takes into account I/O pin locations, I/O standards, slew rates, and terminations used, and provides a value for the noise margin for each pin based on these characteristics. The noise margin does not include any system-level characteristics such as board trace cross-talk or reflections due to board impedance discontinuities.

Ground or power bounce occurs when a large number of outputs simultaneously switch in the same direction. The output drive transistors all conduct current to a common rail. Low-to-High transitions connect to the  $V_{CCO}$  rail, while High-to-Low transitions connect to the GND rail. The resulting cumulative current transient induces a voltage difference across the inductance that exists between the internal and external ground levels, or internal and external  $V_{CCO}$  levels. The inductance is associated with bumps, die routing, package routing, and ball inductance. Any SSO-induced voltage consequently affects internal switching noise margins and ultimately signal quality.

The SSN predictor results assume that the device is soldered on the PCB and that the board uses sound design practices. The noise margin values do not apply for devices mounted in sockets due to the additional BGA ball inductance introduced by the socket.

### Pin Planning to Mitigate SSO Sensitivity



**IMPORTANT:** *When performing pin planning of a design, it is important to choose I/O pin placements that separate strong outputs and/or SSOs from sensitive inputs and outputs (particularly asynchronous inputs).*

Strong outputs tend to be the class-II versions of HSTL and SSTL drivers, PCI™ variants, and any LVCMOS or LVTTTL with drive strengths over 8 mA. Sensitive inputs and outputs can have a low noise margin and tend to be high-speed signals or signals where the swing is reduced by parallel receiver termination. Because localized SSO noise is based on the proximity of signals to one another, it is important to try to separate signals based on the position of the package solder balls. To further reduce potential noise induced from SSOs, outputs should be distributed evenly rather than clustered in one area. SSOs within a bank should be spread across the bank as much as possible. Whenever possible, SSOs should be distributed into multiple banks.

The floorplanning capability in the Vivado Design Suite can help accomplish pin planning to avoid SSO sensitivity issues. By clicking on a package pin in the Package window, a corresponding IOB is highlighted in the Device window. These IOB site types represent the

die pads and show the relative physical location around the die edge. Through the use of the floorplanning tool, intelligent pin placement can be used to separate the die pads of pins. This is implemented by separating the die pads of pins with strong outputs and SSOs from the die pads of pins with sensitive inputs and outputs. SSO effects can also be minimized by adding virtual GND pins and virtual  $V_{CCO}$  pins. A virtual GND is created by defining an output pin driven by a logic 0 at the highest drive strength available and connected to GND on the board. Similarly, a virtual  $V_{CCO}$  pin is created by defining an output pin driven by a logic 1 at the highest drive strength and connected to  $V_{CCO}$  on the board.

# SelectIO Interface Logic Resources

## Bank Overview

Each I/O bank contains 52 pins that can be used for input, output, or bidirectional operations using single-ended standards appropriate for the bank. The I/O banks can be either high-range (HR) or high-performance (HP) I/O banks. Up to 48 of these pins can be configured as up to 24 differential signaling pin pairs with signaling standards appropriate for either an HR I/O or an HP I/O bank. The logic associated with each single-ended pin is known as a *bit slice*, and the differential pin pairs are referenced as a *master* bit slice for the *\_P* pin and *slave* bit slice for the *\_N* pin throughout this user guide.

An overview of each bank is shown in [Figure 2-1](#). The input/output control block bit slices can be programmed using either component primitives as in previous generations of AMD Adaptive Computing devices or, where maximum performance is required, configured using native PHY primitives. Both mechanisms are discussed in this chapter.

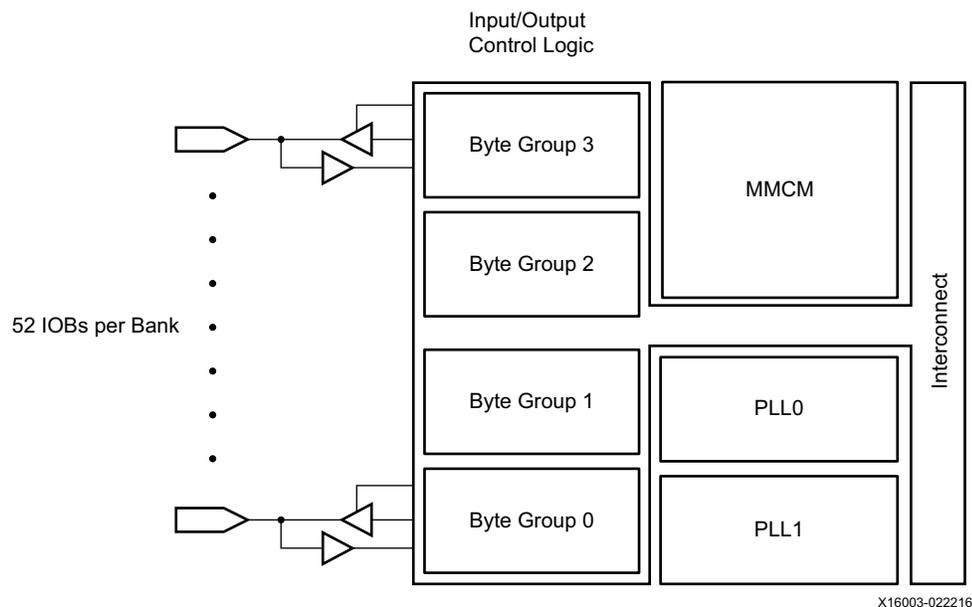


Figure 2-1: Bank Overview



---

**TIP:** *Native mode designs have additional restrictions. The High-Speed SelectIO wizard (HSSIO-Wiz) automatically sets up all required settings and checks the design rules to ensure a working design. AMD recommends using the HSSIO-Wiz for Native mode designs.*

---

The two available PLLs are associated with bit slices in the same I/O bank. Each PLL has a dedicated high-speed clock connection to the controller of the bit slices and two extra outputs that can be used for application clocks for logic placed in the clock area that the I/O bank covers. The mixed-mode clock manager (MMCM) can be used as a clock source for the controller of the bit slices in the I/O bank and logic placed in the clock area the I/O bank covers, but the MMCM can also be used as a clock source for I/O banks and logic in the entire FPGA.



---

**TIP:** *Use the PLLs placed in the clock area behind the I/O bank for applications requiring high performance and low jitter. The MMCM could be used for slower applications requiring clocking in multiple I/O banks and clock areas.*

---

Use the following XDC constraint if the clock input is not part of the I/O bank used for the interface that needs to be designed.

```
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets <clock_net_name>]
```

This constraint causes Vivado™ tools to issue a warning instead of stopping with an error.



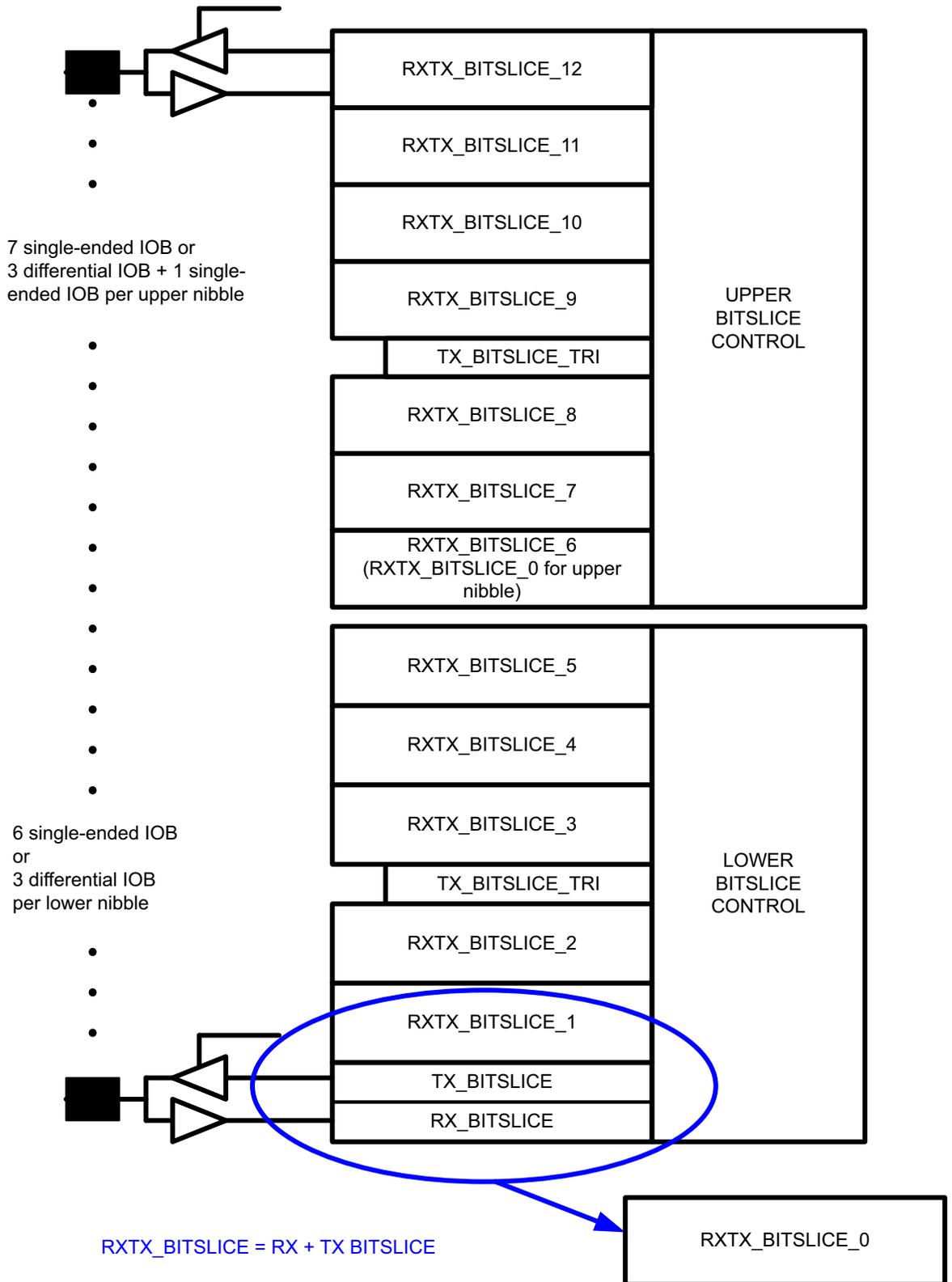
---

**TIP:** *In Native mode, the high-speed clock output of the PLL connects to the BITSLICE\_CONTROL.PLL\_CLK inputs over dedicated routing (without a clock buffer). So always place the PLL in the clock area joining the I/O bank of the interface. The input clock buffer can be placed in a different I/O bank while using the XDC constraint.*

*Clock buffers must be used when an MMCM is used to clock a Native or Component mode interface. Although the best solution is to place the MMCM near the constructed I/O interface, the MMCM can be placed in a different clock area than the one adjacent to the I/O bank used. The clocks for the I/O interface are then distributed by clock buffers and clock routing.*

---

Each bank is subdivided into four byte groups, each group containing 13 I/O pins as shown in [Figure 2-1](#). Each byte group is further sub-divided into two nibble groups, as shown in [Figure 2-2](#). The 3-state control bit slice blocks and upper and lower nibble control blocks are only relevant when using the Native mode, and are further described in subsequent sections. All bit slices can be used for either single-ended or differential signaling, with the exception of BITSLICE\_12 (BITSLICE\_6 in the upper nibble), which is only intended for single-ended signaling. Any single-ended clocks for use with the bit slices should use BITSLICE\_0 of a nibble, and any differential clocks should use BITSLICE\_0 (P-side) and BITSLICE\_1 (N-side) of a nibble. Other pins can be used for clocks that require access to global clocking resources, as described in the *UltraScale Architecture Clocking Resource User Guide* (UG572) [[Ref 10](#)].



X16004-011518

Figure 2-2: Byte Group Overview

The two central byte groups (1 and 2) each contain clocks quad byte clock (QBC) and global clock (GC)-capable input pins or pin pairs. The QBC pins can be used as capture clock inputs for the nibble or byte group they are placed in, but they can also deliver a capture clock through a dedicated clock backbone to all other nibbles and byte groups in the I/O bank. The GC pins are clock inputs that can drive MMCM and/or PLL primitives. Some of these clock-capable inputs have dual function capabilities—QBC and GC. The upper and lower byte groups each contain dedicated byte clock (DBC) clock-capable input pins (pin pairs) that can be used for clocking inside the byte group but do not have the capability to drive a capture clock to other byte groups in the I/O bank or to drive MMCM or PLLs in the I/O bank.

Additional restrictions might apply to BITSlice\_0 for the upper nibble and lower nibble.

**Note:** BITSlice\_0 for the upper nibble is the equivalent of BITSlice\_6 for a byte group. See the example I/O bank (bank 44 of XCKU040FFVA1156) in [Figure 2-3](#) for an explanation of bitslice numbering within a byte and nibble.

When using RX\_BITSlice or RXTX\_BITSlice, inter-byte clocking might affect BITSlice\_0 availability.

- If using inter-byte clocking (QBC) from a nibble in one byte (source) to a nibble in another byte (sink), the nibble in the sink byte must always include BITSlice 0 and its DATA\_TYPE set to DATA.
- For Receive serial mode applications, every nibble must include BITSlice 0 and its DATA\_TYPE set to SERIAL.

See [Clocking in Native Mode, page 302](#) for additional details.

IDELAY/ODELAY and RX\_BITSlice/TX\_BITSlice/RXTX\_BITSlice support TIME mode, which provides more precise delays by continuously adjusting the alignment. When TIME mode is used for IDELAY/ODELAY and native primitives, BITSlice\_0 is used during an initial calibration process. In the case of IDELAY/ODELAY, this initial calibration process is completed when RDY (IDELAYCTRL) is asserted high. Component logic connected to BITSlice\_0 might not be available during the initial calibration in these conditions:

- IDELAY/ODELAY in TIME mode
- RX\_BITSlice/TX\_BITSlice/RXTX\_BITSlice in TIME mode

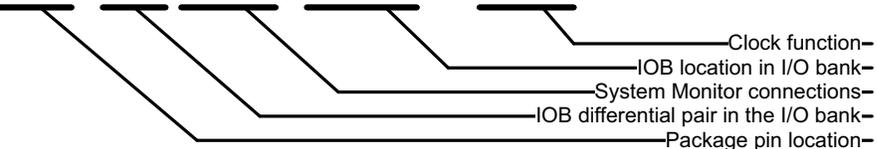
Vivado will issue an error message to indicate input routing and logic associated with BITSlice\_0 within a nibble will be unavailable during the BISC operation. If these restrictions do not affect a design, the DRC can be disabled with the following constraint:

```
set_property UNAVAILABLE_DURING_CALIBRATION TRUE [get_ports <name>]
```

As BITSlice\_0 is used for calibration for the TIME mode, all other bit slices within the nibble will not be available until after calibration has completed when IDELAY/ODELAY are in TIME mode.

Each of the UltraScale™ FPGA I/O banks, bytes, and nibbles have the same setup. [Figure 2-3](#) shows an example of a pin setting of I/O bank 44 of a XCKU040FFVA1156 FPGA. The setup shown can be applied to all I/O banks over the entire FPGA family. Using [Figure 2-3](#) simplifies making pin nibble, byte, and I/O bank pin assignments.

IO_BANK	BYTE_3	NIBBLE_1	AM25			IOB_X0Y51	DBC DBC DBC DBC	T3U	BITSlice no. for the upper nibble group (pin name label)	N12	N6	
			AP23	L24N		IOB_X0Y50		T3U		N11	N5	
			AN23	L24P		IOB_X0Y49		T3U		N10	N4	
		AP25	L23N		IOB_X0Y48	T3U		N9		N3		
		AP24	L23P		IOB_X0Y47	T3U		N8		N2		
		AP21	L22N	AD0N	IOB_X0Y46	T3U		N7		N1		
	NIBBLE_0	AP20	L22P	AD0P	IOB_X0Y45	T3U	N6	N0	BITSlice no. for the lower nibble	N5		
		AN24	L21N	AD8N	IOB_X0Y44	T3L	N5	N4				
		AM24	L21P	AD8P	IOB_X0Y43	T3L	N4	N3				
		AN22	L20N	AD1N	IOB_X0Y42	T3L	N3	N2				
		AM22	L20P	AD1P	IOB_X0Y41	T3L	N2	N1				
		AN21	L19N	AD9N	IOB_X0Y40	T3L	N1	N0				
	BYTE_2	NIBBLE_1	AK25			IOB_X0Y38	QBC QBC GC GC GC/QBC GC/QBC	T2U	BITSlice no. for the upper nibble group (pin name label)	N12	N6	
			AL25	L18N	AD2N	IOB_X0Y37		T2U		N11	N5	
			AL24	L18P	AD2P	IOB_X0Y36		T2U		N10	N4	
			AL23	L17N	AD10N	IOB_X0Y35		T2U		N9	N3	
			AL22	L17P	AD10P	IOB_X0Y34		T2U		N8	N2	
		NIBBLE_0	AK20	L16N	AD3N	IOB_X0Y33		T2U	N7	N1	BITSlice no. for the lower nibble	N0
			AJ20	L16P	AD3P	IOB_X0Y32		T2L	N6	N5		
			AM20	L15N	AD11N	IOB_X0Y31		T2L	N5	N4		
AL20			L15P	AD11P	IOB_X0Y30	T2L		N4	N3			
AK23			L14N		IOB_X0Y29	T2L		N3	N2			
NIBBLE_0	AK22	L14P		IOB_X0Y28	T2L	N2	N1	BITSlice no. for the lower nibble	N0			
	AK21	L13N		IOB_X0Y27	T2L	N1	N0					
	AJ21	L13P		IOB_X0Y26	T2L	N0	N0					
	NIBBLE_1	AF25			IOB_X0Y25	GC GC GC GC QBC QBC	T1U		BITSlice no. for the upper nibble group (pin name label)	N12	N6	
		AH23	L12N		IOB_X0Y24		T1U			N11	N5	
AH22		L12P		IOB_X0Y23	T1U		N10	N4				
AJ24		L11N		IOB_X0Y22	T1U		N9	N3				
AJ23		L11P		IOB_X0Y21	T1U		N8	N2				
AJ25		L10N	AD4N	IOB_X0Y20	T1U		N7	N1				
NIBBLE_0	AH24	L10P	AD4P	IOB_X0Y19	T1U	N6	N0	BITSlice no. for the lower nibble	N5			
	AG25	L9N	AD12N	IOB_X0Y18	T1L	N5	N4					
	AG24	L9P	AD12P	IOB_X0Y17	T1L	N4	N3					
	AF24	L8N	AD5N	IOB_X0Y16	T1L	N3	N2					
	AF23	L8P	AD5P	IOB_X0Y15	T1L	N2	N1					
	AE26	L7N	AD13N	IOB_X0Y14	T1L	N1	N0					
BYTE_1	NIBBLE_1	AD24			IOB_X0Y12	VRP DBC DBC	T0U	BITSlice no. for the upper nibble group (pin name label)	N12	N6		
		AG22	L6N	AD6N	IOB_X0Y11		T0U		N11	N5		
		AF22	L6P	AD6P	IOB_X0Y10		T0U		N10	N4		
		AE23	L5N	AD14N	IOB_X0Y9		T0U		N9	N3		
		AE22	L5P	AD14P	IOB_X0Y8		T0U		N8	N2		
	NIBBLE_0	AH21	L4N	AD7N	IOB_X0Y7		T0U	N7	N1	BITSlice no. for the lower nibble	N0	
		AG21	L4P	AD7P	IOB_X0Y6		T0U	N6	N5			
		AE20	L3N	AD15N	IOB_X0Y5		T0L	N5	N4			
		AD20	L3P	AD15P	IOB_X0Y4		T0L	N4	N3			
		AG20	L2N		IOB_X0Y3		T0L	N3	N2			
NIBBLE_0	AF20	L2P		IOB_X0Y2	T0L	N2	N1	BITSlice no. for the lower nibble	N0			
	AE21	L1N		IOB_X0Y1	T0L	N1	N0					
	AD21	L1P		IOB_X0Y0	T0L	N0	N0					



X16263-071817

Figure 2-3: Example I/O Bank (Bank 44 of XCKU040FFVA1156)

## Component Primitives

### Simple Registered Inputs and Outputs

The SDR input and output registers use different resources than IDDR/ODDR registers and thus have different performance characteristics. SDR input and output registering inside the bit slice is performed using a flip-flop primitive in conjunction with the `IOB = TRUE` constraint applied to the flip-flop instance. Note that `IS_D_INVERTED` is not supported for UltraScale and UltraScale+ and must be set to 0. Simulation results do not match hardware if `IS_D_INVERTED` is set to 1. This can either be instantiated directly or is inferred by synthesis. Applicable elements are;

- 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

### IDDRE1

UltraScale devices have dedicated registers in the bit slice to implement input DDR registers. This feature is used by instantiating the IDDRE1 primitive. The IDDRE1 primitive supports these modes of operation:

- OPPOSITE\_EDGE
- SAME\_EDGE
- SAME\_EDGE\_PIPELINED

The SAME\_EDGE and SAME\_EDGE\_PIPELINED modes allow designers to transfer falling edge data to the rising edge domain within the bit slice, 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***

OPPOSITE\_EDGE mode, a traditional input DDR solution, is accomplished using a single input in the ILOGIC 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 7 series FPGA implementation. Figure 2-4 shows the timing diagram of the input DDR using the OPPOSITE\_EDGE mode.

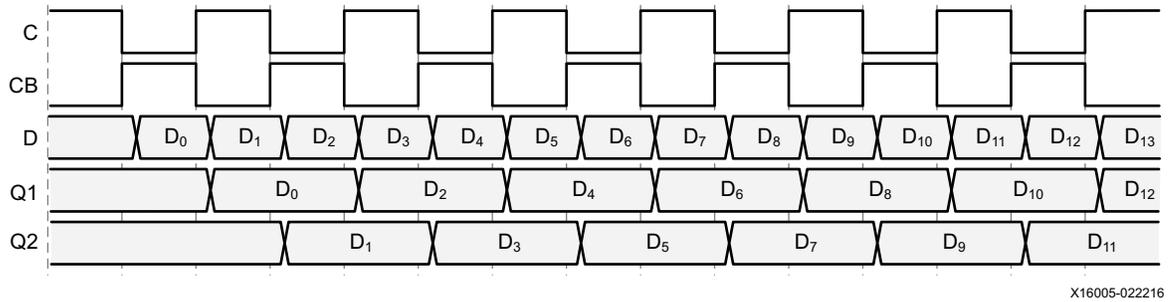


Figure 2-4: Input DDR Timing in OPPOSITE\_EDGE Mode

**SAME\_EDGE Mode**

In SAME\_EDGE mode, the data is presented into the device logic on the same clock edge. Figure 2-5 shows the timing diagram of 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.

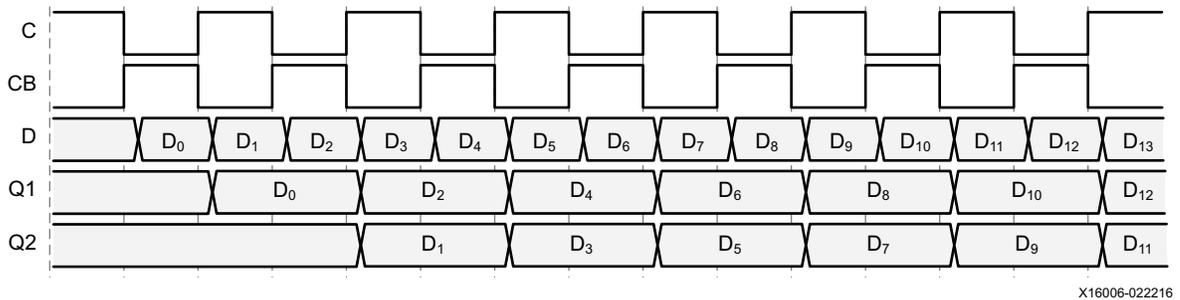


Figure 2-5: Input DDR Timing in SAME\_EDGE Mode

**SAME\_EDGE\_PIPELINED 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. Figure 2-6 shows the timing diagram of 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.

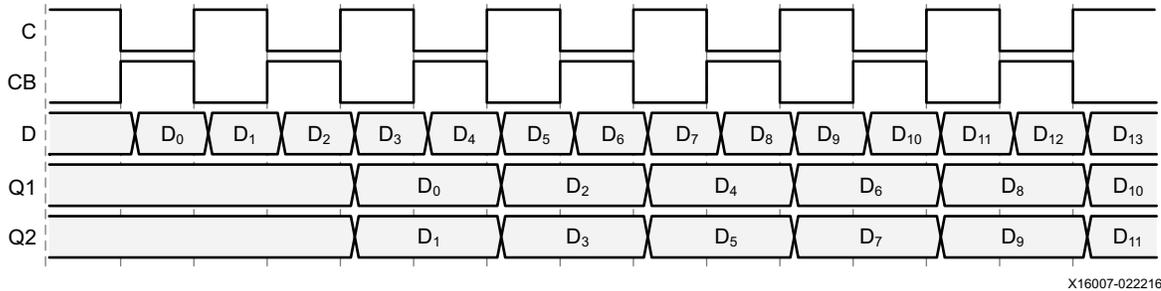


Figure 2-6: Input DDR Timing in SAME\_EDGE\_PIPELINED Mode

Figure 2-7 shows a block diagram of the IDDRE1 primitive.

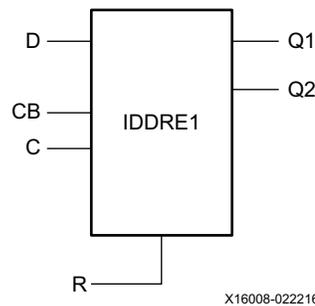


Figure 2-7: IDDRE1 Primitive Block Diagram

**IDDRE1 Ports**

**Note:** IDDRE1 components used in a design are translated and implemented by the Vivado design tools as ISERDESE3 components.

Table 2-1 lists the IDDRE1 ports.

Table 2-1: 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

Table 2-1: **IDDRE1 Ports (Cont'd)**

Port	I/O	Description
D	Input	Register input from IOB
R	Input	Asynchronous reset, release synchronous to C/CB

### IDDRE1 Attributes

Table 2-2 lists the IDDRE1 attributes.

Table 2-2: **IDDRE1 Attributes**

Attribute	Values	Default	Type	Description
DDR_CLK_EDGE	OPPOSITE_EDGE SAME_EDGE SAME_EDGE_PIPELINED	OPPOSITE_EDGE	String	Sets the IDDRE1 mode of operation with respect to clock edge.
IS_C_INVERTED	0 or 1	0	Bit	Sets a local clock inversion for C input when 1.
IS_CB_INVERTED	0 or 1	0	Bit	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.

## ODDRE1

UltraScale devices have registers in the bit slice to implement output DDR registers as in previous FPGA 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, saving CLB and clock resources and increasing performance.

This mode is also supported for 3-state control. The timing diagram of the output DDR is shown in [Figure 2-8](#).

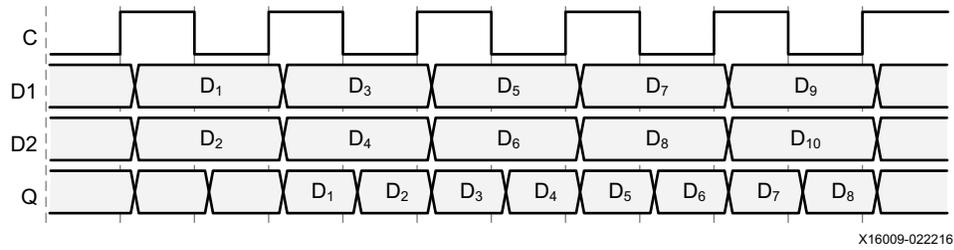


Figure 2-8: Output DDR Timing

[Figure 2-9](#) shows a block diagram of the ODDRE1 primitive.

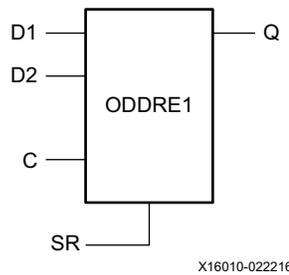


Figure 2-9: ODDRE1 Primitive Block Diagram

**Note:** ODDRE1 components used in a design are translated and implemented by the Vivado design tools as OSERDESE3 components.

### ODDRE1 Ports

[Table 2-3](#) lists the ODDRE1 ports.

Table 2-3: 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 set/reset. Release synchronously. When SR is asserted, the Q output is asynchronously set to the SRVAL. The SRVAL is held for 4 clock cycles before resuming normal operation.

### ODDRE1 Attributes

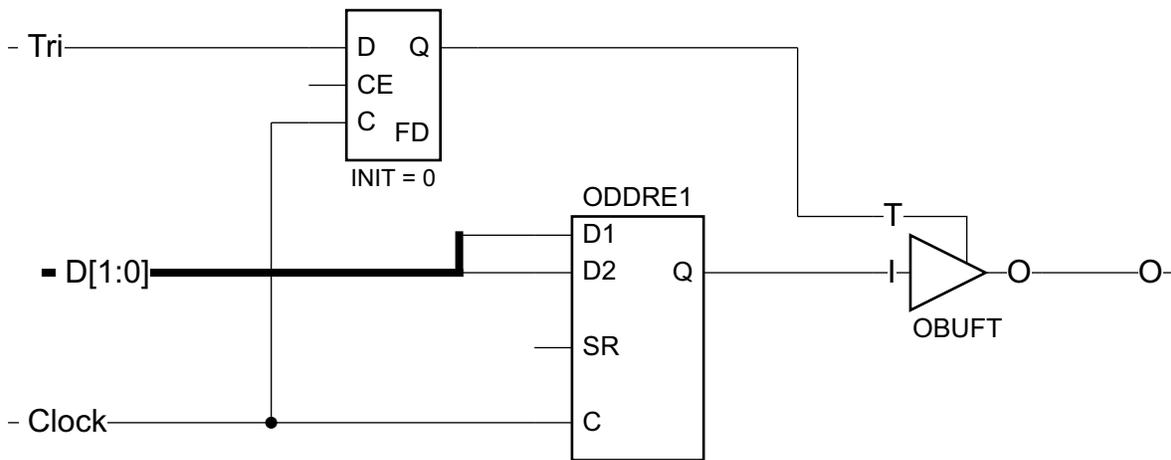
Table 2-4 lists the ODDRE1 attributes.

Table 2-4: ODDRE1 Attributes

Attribute	Values	Default	Type	Description
SRVAL	0 or 1	0	Bit	Value of Q output configuration after a reset
IS_C_INVERTED	0 or 1	0	Bit	Local signal inversion
IS_D1_INVERTED	0 or 1	0	Bit	Not supported <b>Note:</b> Simulation results will not match hardware if IS_D1_INVERTED is set to 1.
IS_D2_INVERTED	0 or 1	0	Bit	Not supported <b>Note:</b> Simulation results will not match hardware if IS_D2_INVERTED is set to 1.
SIM_DEVICE	ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2, ULTRASCALE	ULTRASCALE	String	Device family for behavioral simulation.

### ODDR with Serialized 3-State

The UltraScale device ODDRE1 solution supports both a single (Figure 2-10) and a serialized (Figure 2-11) 3-state source.



X16011-022216

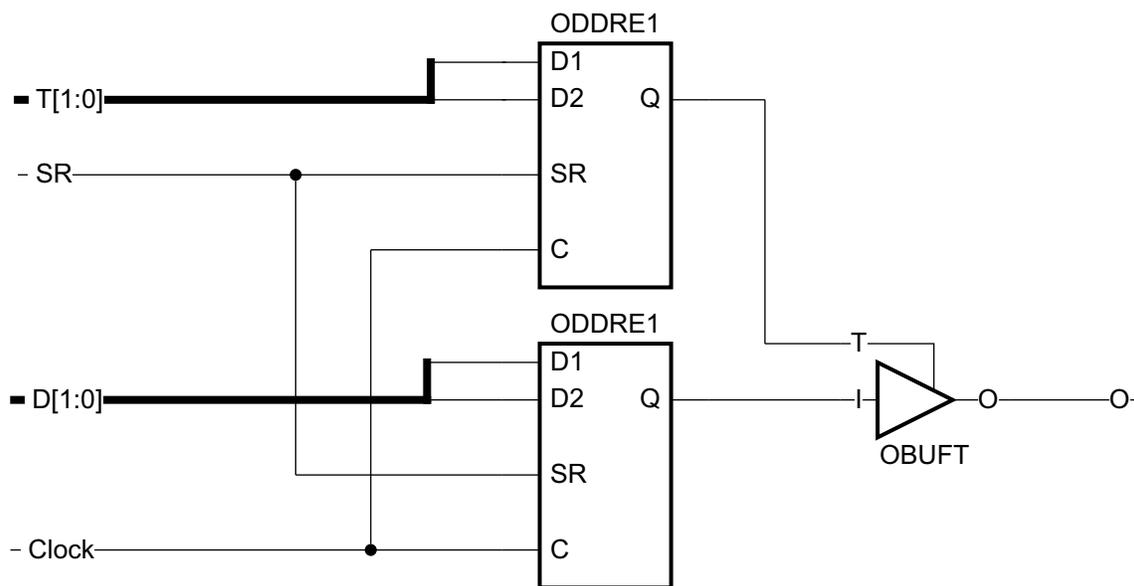
Figure 2-10: ODDR with Internal Logic Flip-Flop 3-State



**TIP:** To achieve required timing constraints for a design set up as in Figure 2-10, it might be necessary to LOC the flip-flop in FPGA logic close to the ODDRE1/OSERDESE3 used.

In the single 3-state solution, the flip-flop driving the 3-state is placed in the internal logic with the ODDRE1 placed in a bit slice site. To have the 3-state flip-flop also placed in the same bit slice site as the ODDRE1, the arrangement shown in Figure 2-11 can be altered to tie the 3-state D1 and D2 inputs together to a common 3-state.

Figure 2-11 shows the serialized ODDRE1 circuit. The SR and C pins of both ODDRE1s must have a common source to allow the implementation software to transform this circuit into a single OSERDESE3 instance that supports the desired function. Although the previously discussed circuit (using ODDRE1 primitives) is preferred, a different way to achieve the discussed circuit is provided in OSERDESE3.



X16012-022216

Figure 2-11: ODDR with ODDR Serialized 3-State

### ISERDESE3

The ISERDESE3 element is available to perform input deserialization for designs migrating from previous FPGA families or for designs not requiring native mode primitives. The ISERDESE3 in UltraScale devices is a serial-to-parallel converter with specific clocking and logic features designed to facilitate the implementation of high-speed source-synchronous applications. The ISERDESE3 avoids the additional timing complexities encountered when designing deserializers in the device logic.

There are some differences between the ISERDESE3 and its predecessors. ISERDESE3 does not have:

- BITSLIP input performing a bitflip operation synchronous to CLKDIV.
- Selectable CE inputs able to function as a 2:1 serial-to-parallel converter clocked by CLKDIV.

- OFB input, being a direct connection between the OSERDES serial output and this input.
- SHIFTIN and SHIFTOUT pins allow extending the deserialization capability up to 14 bits by cascading two ISERDES using direct connections.

The ISERDESE3 can deserialize an incoming signal by 2 or 4 in SDR data capture, and by 4 or 8 in DDR data capture mode. When used for SDR data capture, the valid outputs are every other data output pin. For example, when used as a 1:4 deserializer using an SDR clock, the data width should be set to 8, and the data received is taken from Q0, Q2, Q4, and Q6. Details of which SerDes output pins to use and which value to apply to the DATA\_WIDTH attribute are shown in [Table 2-5](#).



**TIP:** The first serial bit received in a word is Q0.

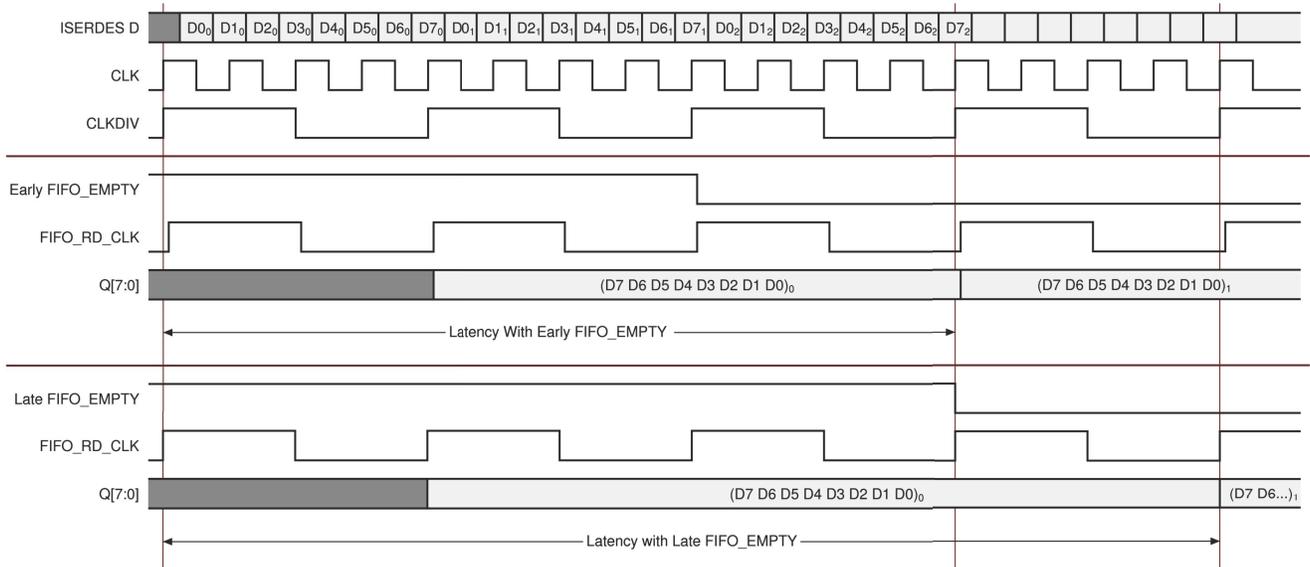
**Table 2-5: ISERDESE3 Output Connections in SDR and DDR Modes**

SDR or DDR	Required Ratio	DATA_WIDTH Attribute to Apply to ISERDESE3	SerDes Output Data Bits to Use
DDR	1:8	8	Q7, Q6, Q5, Q4, Q3, Q2, Q1, Q0
DDR	1:4	4	Q3, Q2, Q1, Q0
SDR	1:8	N/A	N/A
SDR	1:4	8	Q6, Q4, Q2, Q0
SDR	1:2	4	Q2, Q0

Other deserialization ratios and word alignment schemes are possible through the use of additional logic resources in the FPGA logic (see *Bitslip in Logic Application Note* (XAPP1208) [Ref 11]). The ISERDESE3 also contains a shallow eight entry FIFO that can optionally be used for clock domain transfers. When not used, the FIFO control signals should be connected to GND. When using the FIFO, the FIFO\_RD\_EN should be driven by the inverted FIFO\_EMPTY signal to ensure the FIFO\_write and read pointers do not overlap every eight clock cycles. As shown in [Figure 2-12](#) and [Figure 2-13](#), latency through the FIFO depends on FIFO\_RD\_CLK. When the write pointer is updated early with respect to FIFO\_RD\_CLK the latency through the FIFO is shorter.

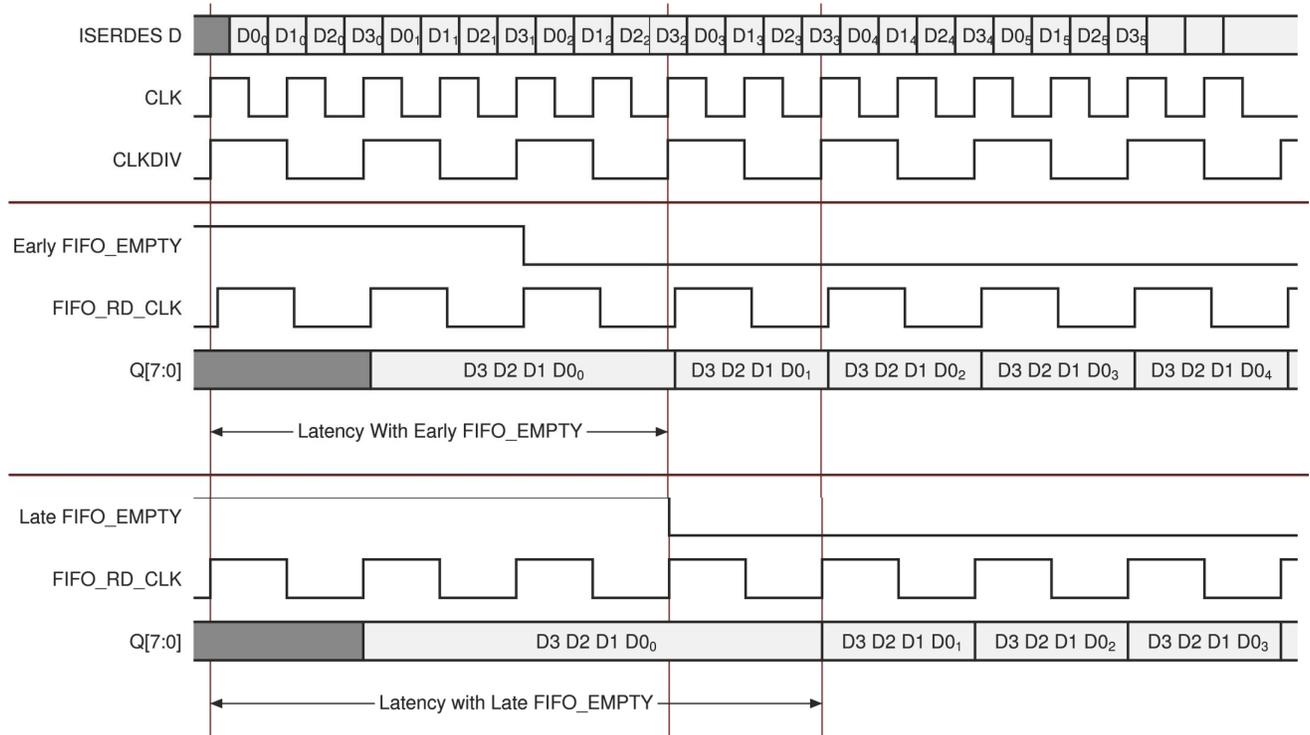
Because clock routing can vary, the MMCM with the ZHOLD compensation compensates for the clock routing. To ensure all of the clock outputs from the MMCM are properly compensated for, the CLOCK\_DELAY\_GROUP must be used (see [Figure 2-28](#)). To ensure the ISERDES is properly aligned after a reset, see [Component Mode Reset Sequence](#).

When the clocks are not compensated for, such as when clock-capable inputs are directly connected to clock buffers (BUFG, BUFGCE, BUFGCE\_DIV), additional bitslip logic is required. See *Bitslip in Logic Application Note* (XAPP1208) [Ref 11].



X19091-121718

Figure 2-12: ISERDES FIFO Latency with DATA\_WIDTH = 8



X19090-121718

Figure 2-13: ISERDES FIFO Latency with DATA\_WIDTH = 4

## ISERDESE3 Ports

Table 2-6 lists the ISERDESE3 ports.

Table 2-6: ISERDESE3 Ports

Port	I/O	Type	Description
CLK	Input	Clock	High-speed clock input. Clock serial input data stream.
CLK_B	Input	Clock	Inverted version of CLK when IS_CLK_INVERTED=0 and IS_CLK_B_INVERTED=0.
CLKDIV	Input	Clock	Low-speed divided clock input.
D	Input	Data	Serial input data Synchronous to CLK/CLK_B.
Q[7:0]	Output	Data	Registered outputs. Synchronous to FIFO_RD_CLK when FIFO_ENABLE is TRUE.
RST	Input	Reset	Asynchronous reset.
FIFO_RD_CLK	Input	Clock	FIFO read clock.
FIFO_RD_EN	Input	Enable	Enables reading the FIFO when asserted.
FIFO_EMPTY	Output		Indicates the FIFO is empty when asserted.
INTERNAL_DIVCLK	Output	Clock	Reserved

## ISERDESE3 Attributes

Table 2-7 lists the ISERDESE3 attributes.

Table 2-7: ISERDESE3 Attributes

Attribute	Values	Default	Type	Description
DATA_WIDTH	4 or 8	8	Decimal	Defines the serial-to-parallel converter width.
FIFO_ENABLE	TRUE/FALSE	FALSE	String	The FIFO is used when the attribute is set TRUE and bypassed when the attribute is set FALSE.
FIFO_SYNC_MODE	TRUE/FALSE	FALSE	String	Set to FALSE when the ISERDES internal FIFO write clock and the FIFO read clock accessed from FPGA logic are from separate or common clock domains. This is the preferred selection because it supports all clocking options. TRUE: Reserved for later use.
IS_CLK_INVERTED	1 or 0	0	Bit	Sets a local clock inversion for CLK input.

Table 2-7: ISERDESE3 Attributes (Cont'd)

Attribute	Values	Default	Type	Description
IS_CLK_B_INVERTED	1 or 0	0	Bit	Sets a local clock inversion for CLK_B input. When IS_CLK_B_INVERTED=1, CLK and CLK_B must be driven by the same global clock buffer. When IS_CLK_B_INVERTED=0, CLK_B must be driven by the same global clock buffer as CLK through an inverter.
IS_RST_INVERTED	1 or 0	0	Bit	Sets a local inversion for RST input when 1.
SIM_DEVICE	ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2	ULTRASCALE	String	Device family for behavioral simulation

## OSERDESE3

The OSERDESE3 primitive is available to perform output serialization for designs migrating from previous FPGA families or for designs not requiring native mode primitives. The OSERDESE3 in UltraScale devices is a 4- or 8-bit parallel-to-serial converter with specific clocking features to facilitate the implementation of source-synchronous and other applications. If other serial-to-parallel conversion factors are required, use the ODDRE1 primitive or implement a gearbox in internal logic.

There are some differences between the OSERDESE3 and its predecessors. The following functionality is not available in the OSERDESE3:

- OCE input enable pin for the serial output of the OSERDES.
- SHIFTIN and SHIFTOUT pins can use local dedicated connections to extend the serialization capabilities of the OSERDES.
- OFB output providing a straight and direct connection between the OSERDES output and ISERDES input without using an input and/or output buffer (IOB) and pin.
- Parallel 3-state and serial TBYTE functionality.

The latency through the OSERDES depends on the DATA\_WIDTH setting as shown in [Figure 2-14](#).

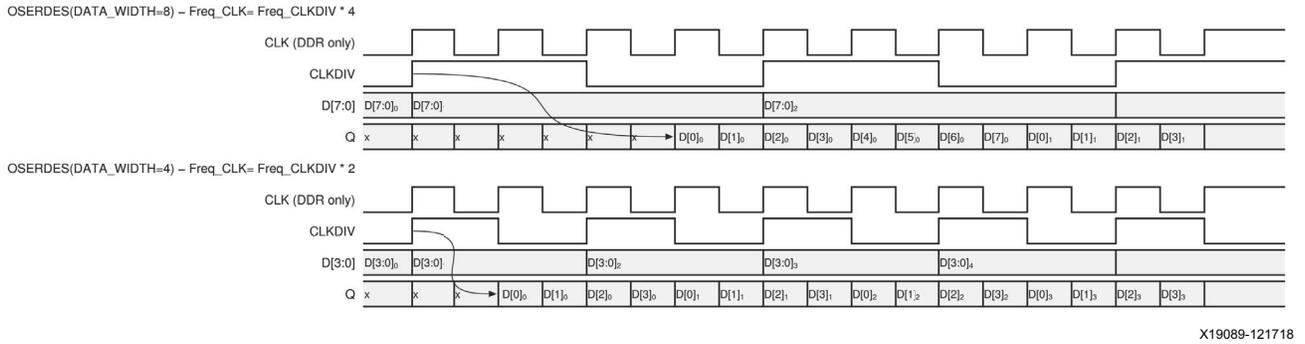


Figure 2-14: OSERDES Latency

The OSERDESE3 can serialize an outgoing signal by a 2 or 4 in SDR mode, or by a 4 or 8 in DDR mode. When used with SDR clocking, the DATA\_WIDTH attribute is to be set to twice the desired width and data to be transmitted should be applied to two pins at a time. See Figure 2-15.

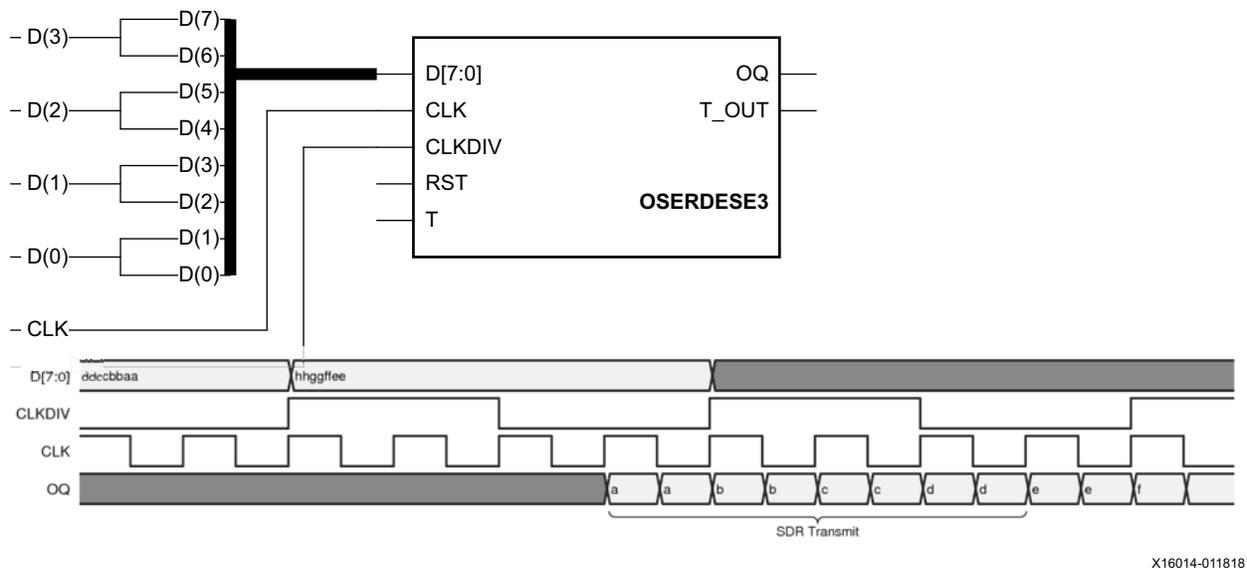


Figure 2-15: OSERDES Used in x4 SDR Mode (DATA\_WIDTH = 8)

The full range of possibilities together with the associated attribute settings and required connections are shown in Table 2-8.



**TIP:** The data applied to SerDes input D0 is the first bit to be transmitted in all cases.

**Table 2-8: OSERDESE3 Output Connections in SDR and DDR Modes**

SDR or DDR	Required Ratio	DATA_WIDTH Attribute to Apply to OSERDESE3	Data Bits to Connect to the SerDes
DDR	8:1	8	D7, D6, D5, D4, D3, D2, D1, D0
DDR	4:1	4	0, 0, 0, 0, D3, D2, D1, D0
SDR	8:1	N/A	N/A
SDR	4:1	8	D3, D3, D2, D2, D1, D1, D0, D0
SDR	2:1	4	0, 0, 0, 0, D1, D1, D0, D0

### OSERDESE3 Ports

Table 2-9 lists the OSERDESE3 ports.

**Table 2-9: OSERDESE3 Ports**

Port	I/O	Description
CLK	Input	High-speed clock input
CLKDIV	Input	Low-speed divided clock input
D[7:0]	Input	Parallel data inputs for serialization synchronous to CLKDIV
OQ	Output	Datapath output
RST	Input	Asynchronous reset.
T_OUT	Output	3-state control output to IOB
T	Input	3-state input from internal logic, combinational 3-state T to T_OUT path. A logic High means the data is 3-stated and a logic Low means the data is not 3-stated.

### OSERDESE3 Attributes

Table 2-10 lists the OSERDESE3 attributes.

**Table 2-10: OSERDESE3 Attributes**

Attribute	Values	Default	Type	Description
DATA_WIDTH	4 or 8	8	Decimal	Defines the parallel-to-serial data converter width.
INIT	1 or 0	0	Binary	Initializes the OSERDESE3 flip-flops to the value specified.

Table 2-10: OSERDESE3 Attributes (Cont'd)

Attribute	Values	Default	Type	Description
ODDR_MODE	TRUE/FALSE	FALSE	String	Internal property set by the Vivado tools when using ODDRE1. Do not modify. Forces the OSERDESE3 into an ODDRE1 mode with a 3-state ODDRE1 flip-flop as shown in <a href="#">Figure 2-11</a> . In the ODDRE1 mode, data is connected to D[4,0] and the clock must be connected to CLKDIV (UltraScale devices) or CLK (UltraScale+ devices). The ODDRE1 primitive is recommended.
OSERDES_D_BYPASS	TRUE/FALSE	FALSE	String	When TRUE, D[0] is passed onto OQ. When FALSE, serialized D[0] and D[4] is output on OQ.
OSERDES_T_BYPASS	TRUE/FALSE	FALSE	String	When TRUE, D[1] is passed onto T_OUT. When FALSE, serialized D[1] and D[5] is output on T_OUT.
IS_CLK_INVERTED	1 or 0	0	Bit	Sets a local clock inversion for CLK input when 1.
IS_CLKDIV_INVERTED	1 or 0	0	Bit	Sets a local clock inversion for CLKDIV input when 1.
IS_RST_INVERTED	1 or 0	0	Bit	Sets a local inversion for RST input when 1.
SIM_DEVICE	ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2	ULTRASCALE	String	Device family for behavioral simulation.

## IDELAYE3

Any input signal except clocks can be delayed using the IDELAYE3 primitive and then either forwarded to the device logic directly, or registered in a simple flip-flop, IDDR, or ISERDESE3, using a single data rate (SDR) clock or a double data rate (DDR) clock inside the input/output interconnect (IOI). Clocks should not be delayed using an IDELAYE3, because the IDELAY cannot directly route to the global clock buffers. When clocks must be delayed, use an MMCM or PLL for clock generation, and delay the clocks using the fine-phase shift capabilities.

The IDELAYE3 primitive contains a 512 tap delay line. See the tap resolution in the UltraScale data sheet [\[Ref 2\]](#). Each individual tap is uncalibrated. However, the logic to calibrate the delay line is available in the IDELAYCTRL component. The IDELAYE3 can be used in two modes, COUNT and TIME.

- COUNT mode:
  - There is no need to use an IDELAYCTRL component because the delay line is used in the uncalibrated state without voltage and temperature compensation.
  - The delay line must be used counting only taps and not counting delay/tap.
    - The DELAY\_VALUE is expressed as taps (0 to 511).
  - Example: Scanning a serial data stream for transitions must be expressed in a number of taps and not translated to time in picoseconds (ps).
- TIME mode:
  - An IDELAYCTRL component **must** be used.
  - The delay line is calibrated for the requested time value and voltage/temperature compensation ensures this value is kept over time.
  - For all delays within a nibble, the REFCLK\_FREQUENCY must match the clock frequency for the IDELAYCTRL to ensure the delays are properly aligned. When mixed with native mode, the REFCLK frequency for the BITSlice\_CONTROL should match the REFCLK\_FREQUENCY. The DELAY\_VALUE is expressed in ps.

The IDELAYE3 primitive is shown in [Figure 2-16](#).

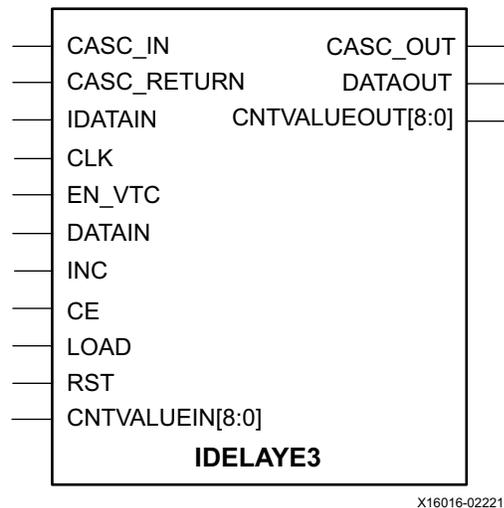


Figure 2-16: IDELAYE3 Primitive

## IDELAYE3 Ports

Table 2-11 lists the IDELAYE3 ports.

Table 2-11: IDELAYE3 Ports

Port	I/O	Description
CASC_RETURN	Input	Cascade delay returning from slave ODELAYE3 DATAOUT. The CASC_RETURN pin is the input cascade delay returning from slave ODELAYE3. The CASC_RETURN of the IDELAYE3 is connected to the slave ODELAYE3 DATAOUT port.
CASC_IN	Input	Cascade delay from slave ODELAYE3 CASC_OUT. The CASC_IN pin is used when the IDELAYE3 is used in a cascade chain as a slave input cascade delay from the master ODELAYE3 CASC_OUT.
CASC_OUT	Output	Cascade delay to ODELAYE3 in cascade. The CASC_OUT pin is used when cascading from an IDELAYE3 to an ODELAYE3. The CASC_OUT port of the IDELAYE3 is connected to the CASC_IN of the ODELAYE3 in cascade.
CE	Input	Clock enable for the delay register clock. <b>Note:</b> Delays might take up to three clock cycles (CLK) to be applied. During this time, input data should not change to ensure output data does not glitch.
CLK	Input	Clock used to sample LOAD, CE, and INC. All control inputs to IDELAYE3 primitive (LOAD, CE, and INC) are synchronous to the clock input (CLK). A clock must be connected to this port when IDELAYE3 is configured in VARIABLE or VAR_LOAD. CLK can be locally inverted, and must be supplied by a global or regional clock buffer. The CLK of the IDELAYE3 must be the same CLK as the ISERDESE3 CLKDIV. Unused when configured in FIXED mode.
INC	Input	The increment/decrement is controlled by the enable signal (CE). This interface is only available when the IDELAYE3 is in VARIABLE or VAR_LOAD mode. As long as CE remains High, IDELAYE3 increments or decrements by one tap every CLK cycle. The state of INC determines whether IDELAYE3 increments or decrements: INC = 1 increments, INC = 0 decrements, synchronously to the CLK. If CE is Low, the delay through IDELAYE3 does not change, regardless of the state of INC. When CE transitions High, the increment/decrement operation begins on the next positive clock edge. When CE transitions Low, the increment/decrement operation ceases on the next positive clock edge. The programmable delay taps in the IDELAYE3 primitive wraps around to the start or end of the taps. When the last tap delay is reached (tap 512), a subsequent increment function returns to tap 0. The same applies to the decrement function—a decrement from zero moves to tap 512.
LOAD	Input	Load counter value from the attribute DELAY_VALUE or the CNTVALUEIN bus when High. When in VAR_LOAD mode and UPDATE_MODE=ASYNC, the IDELAYE3 load port, LOAD, loads the value set by the CNTVALUEIN into registers connected to the delay line tap selection logic. The value present at CNTVALUEIN[8:0] is the new tap value. The LOAD signal is an active-High signal and is synchronous to the input CLK signal. Wait at least one clock cycle after applying a new value on the CNTVALUEIN bus before applying the LOAD signal. CE must be held Low during LOAD operation. When in UPDATE_MODE = SYNC mode, connect LOAD to ground (GND). <b>Note:</b> Delays may take up to three clock cycles (CLK) to be applied. During this time, input data should not change to ensure output data does not glitch.

Table 2-11: IDELAYE3 Ports (Cont'd)

Port	I/O	Description
CNTVALUEIN[8:0]	Input	The CNTVALUEIN pins are used for dynamically switching the loadable tap value. The CNTVALUEIN is the number of taps required. The new value is best applied one clock cycle before applying the LOAD signal. The delay line can be changed from 1 to 8 taps at a time.
CNTVALUEOUT[8:0]	Output	The CNTVALUEOUT pins are used for reporting the current tap value and reads out the amount of taps in the current delay. When EN_VTC is High, CNTVALUEOUT is updated by the IDELAYCTRL.
DATAIN	Input	The DATAIN input is directly driven by the interconnect logic providing a logic accessible delay line. The data is driven back into the interconnect logic through the DATAOUT port with a delay set by the DELAY_VALUE. DATAIN can be locally inverted. The data cannot be driven to an IOB.
IDATAIN	Input	The IDATAIN input is driven by its associated IOB.
DATAOUT	Output	Delayed data from the two data input ports. DATAOUT drives ILOGIC (IFD/IDDR), ISERDESE3, and logic in the FPGA.
RST	Input	The RST pin (reset) is an asynchronous input. When the IDELAYE3 is reset, the delay is set to the value defined by the DELAY_VALUE attribute. RST must follow the Component Mode Reset Sequence when used with the IDELAYCTRL. After IDELAYCTRL.RDY goes High, IDELAY can be used for normal operation.
EN_VTC	Input	EN_VTC: Enable voltage temperature compensation. <ul style="list-style-type: none"> <li>High: Enables IDELAYCTRL to keep delay constant over VT.</li> <li>Low: VT compensation is disabled.</li> </ul> To make delay line updates, EN_VTC must be kept Low. EN_VTC is an asynchronous input but must follow the Component Mode Reset Sequence when used with the IDELAYCTRL.

### IDELAYE3 Attributes

Table 2-12 lists the IDELAYE3 attributes.

Table 2-12: IDELAYE3 Attributes

Attribute	Possible Values	Default	Type	Description
DELAY_SRC	DATAIN IDATAIN	IDATAIN	String	For more information, see <a href="#">DELAY_SRC Attribute</a> .
CASCADE	NONE MASTER SLAVE_MIDDLE SLAVE_END	NONE	String	For more information, see <a href="#">CASCADE Attribute</a> .
DELAY_TYPE	FIXED VAR_LOAD VARIABLE	FIXED	String	The DELAY_TYPE attribute sets the type of delay used. It can be FIXED, VARIABLE or VAR_LOAD. See <a href="#">DELAY_TYPE Attribute</a> for further information.

Table 2-12: IDELAYE3 Attributes (Cont'd)

Attribute	Possible Values	Default	Type	Description
DELAY_VALUE	0–1250 (TIME UltraScale) 0–1100 (TIME UltraScale+) 0–511 (COUNT)	0	Decimal	TIME mode: Desired value in ps. UltraScale devices support delays up to 1.25 ns. UltraScale+ devices support up to 1.1 ns. COUNT mode: Desired value in taps. For more information, see <a href="#">DELAY_VALUE Attribute</a> .
REFCLK_FREQUENCY	200.00–800.00 (UltraScale IDELAYCTRL) 300.00–800.00 (UltraScale+ IDELAYCTRL) 200.00–2400.00 (UltraScale BITSlice_CONTROL) 300.00–2666.67 (UltraScale+ BITSlice_CONTROL)	300.0	1 significant digit float	The REFCLK_FREQUENCY attribute specifies the reference clock of the IDELAYCTRL frequency in MHz. This attribute <b>must</b> mimic the clock frequency applied at the IDELAYCTRL component, except when DELAY_FORMAT is set to COUNT (when the attribute can be left at the default value).
DELAY_FORMAT	TIME <sup>(1)</sup> COUNT	TIME	String	When set to TIME, the delay equals the value given in DELAY_VALUE (specified in ps) plus an Align_Delay. Calibrated using the REFCLK_FREQUENCY set here applied at the IDELAYCTRL component. When set to COUNT, the initial tap setting goes to whatever number of taps is specified in DELAY_VALUE. This does not give a constant delay because the tap delays vary with PVT. For more information, see <a href="#">DELAY_FORMAT Attribute</a> .
UPDATE_MODE	ASYNC SYNC MANUAL	ASYNC	String	For more information, see <a href="#">UPDATE_MODE Attribute</a> .
SIM_DEVICE	ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2	ULTRASCALE	String	Sets the device version (ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2)

**Notes:**

- When in TIME mode, calibration affects the availability of bit slices within the nibble. See [Bank Overview](#) for more information.

**DELAY\_SRC Attribute**

DELAY\_SRC (Figure 2-17) is set based on where the input to be delayed originates. When the input comes from an IOB, it should be set to IDATAIN. When the input comes from the interconnect logic, it should be set to DATAIN. When enabling the IDELAYE3, there is an additional insertion delay added because the signal must pass through a multiplexer. The delay associated with this multiplexer is the insertion delay. If an IDELAYE3 is used with a DELAY\_VALUE = 0, the data still incurs an insertion delay to propagate through the delay element. This delay is accounted for in the Vivado Design Suite timing analysis.

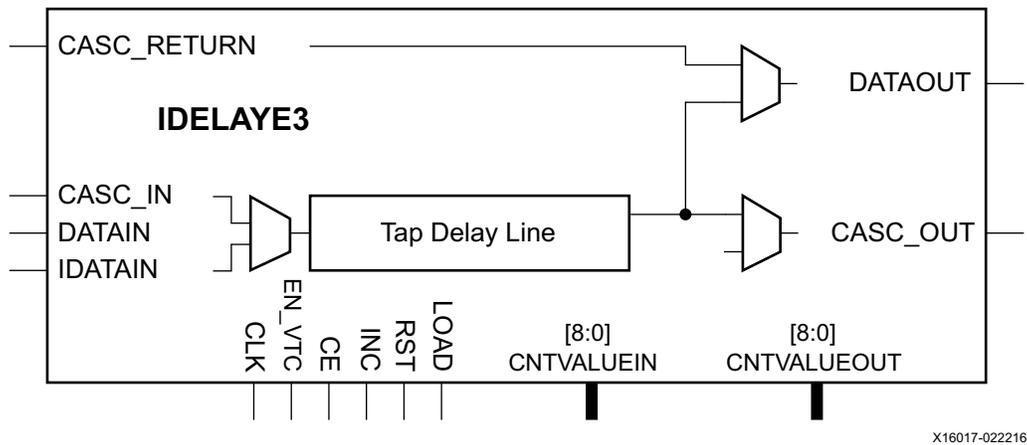
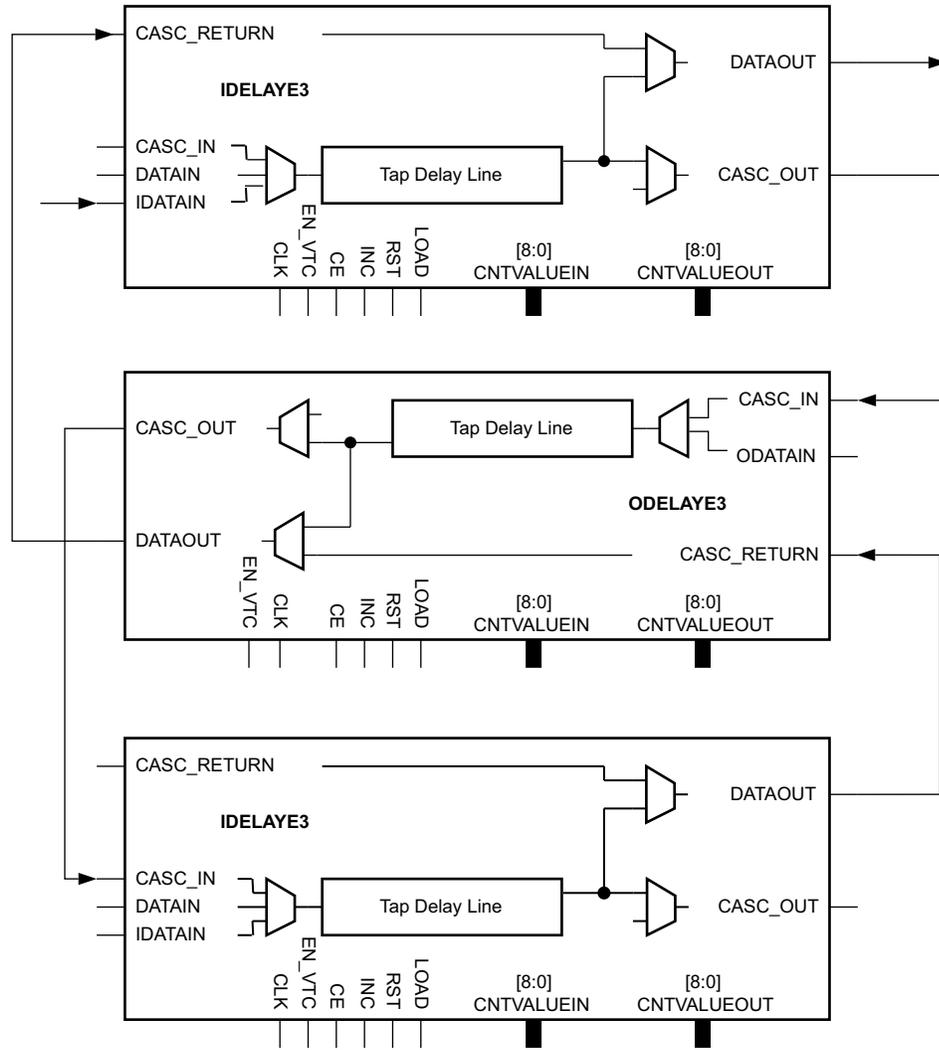


Figure 2-17: IDELAYE3 DELAY\_SRC Diagram

**CASCADE Attribute**

The CASCADE attribute is set to NONE if the delay line is not cascaded. Cascading is used when a delay greater than 1.25 ns (1.10 ns for UltraScale+ devices) is required. The connections between delay elements are shown in Figure 2-18. When using the IDELAYE3 (or ODELAYE3) for cascading, the delay (and the IOB) are no longer available to the design. The delay elements can be cascaded up to the byte boundary in a downward direction. Therefore, the maximum possible length of the delay depends on where the I/O is placed in the byte.

The routes used to cascade IDELAYE3 and ODELAYE3 are dedicated, high-speed routes. The total fixed intrinsic insertion delay for an IDELAYE3 or ODELAYE3 cascade is the sum of the initial insertion delay plus the cascaded insertion delay. This delay grows as multiple of the times that IDELAYE3 and ODELAYE3 are cascaded. However, the delay is always a fixed value.



X16018-050516

Figure 2-18: IDELAYE3 Cascading to Three Slave Delays

If cascading to achieve a delay > 1.25 ns (UltraScale devices) or 1.10 ns (UltraScale+ devices) with a DELAY\_FORMAT = TIME, delays in the same site must have an equal delay. For example, a delay of 1.5 ns is split between a 0.75 ns IDELAYE3 and a 0.75 ns ODELAYE3. If cascading with IDELAYE3 and ODELAYE3 used in VAR\_LOAD mode, the values are entered for both components separately. VAR\_LOAD is discussed in [DELAY\\_TYPE Attribute](#), page 171.

**DELAY\_FORMAT Attribute**

The tap size of the IDELAYE3 primitive is defined in the UltraScale device data sheets as  $T_{IDELAY\_RESOLUTION}$  [Ref 2].

If the DELAY\_FORMAT is set to TIME, the delay line is calibrated, controlled, and maintained for voltage and temperature by the IDELAYCTRL component.

- An IDELAYCTRL component **must** be used.
- The REFCLK\_FREQUENCY attribute must reflect the clock frequency applied to the IDELAYCTRL component.
- The EN\_VTC pin must be actively manipulated when the delay line is used in VARIABLE or VAR\_LOAD mode. When FIXED mode is used, tie the EN\_VTC pin High.



---

**CAUTION!** During the built-in self-calibration (BISC) process, the input delay line (IDELAY) is used to eliminate the clock-to-data skew at the input of the first flip-flops in the serial-to-parallel conversion process.

This process consumes a number of taps of the input delay line, and is called **Align\_Delay**. The Align\_Delay is reported as a value between 45 and 65 taps, when DELAY\_VALUE is set as 0 ps and read out through the CNTVALUEOUT or RIU\_RD\_DATA when using the register interface unit (RIU) interface with the BITSlice\_CONTROL.

Writing all zeros or an amount of taps smaller than the reported Align\_Delay to an input delay line can impact the Align\_Delay inserted by the BISC and might cause issues when capturing data.

---

When the DELAY\_FORMAT is set to COUNT, the delay line is not calibrated and is not maintained over voltage and temperature.

- Therefore, do not use an IDELAYCTRL component.
- Leave the REFCLK\_FREQUENCY attribute at the default value (300 MHz).
- Tie the EN\_VTC input pin Low.
  - This pin ensures that calibration and VT maintenance logic in the IDELAYE3 is disabled.
- The delay line must be used to represent an amount of taps.
  - It does not matter how long the tap delay is; it is the amount of taps that is important.
  - 512 taps are available.
- The CNTVALUEIN/OUT[8:0] values represent the amount of taps the delay line is set or tuned to.

### **DELAY\_VALUE Attribute**

When the DELAY\_FORMAT attribute is set to TIME mode, the DELAY\_VALUE attribute represents an amount in ps. The IDELAYE3 has a clock/data align delay that is in addition to the DELAY\_VALUE attribute. The total delay through the IDELAYE3 is the align delay plus the DELAY\_VALUE.

Despite that in TIME mode the DELAY\_VALUE represents a time value in ps, the value read or written from or to the delay line by the CNTVALUEIN[8:0] and/or CNTVALUEOUT[8:0] is expressed in an amount of taps. So changing the time of a delay line requires some calculation, which is provided in the DELAY\_MODE/VAR\_LOAD paragraph.

When the DELAY\_FORMAT attribute is set to COUNT mode, the DELAY\_VALUE attribute represents an amount of taps. Because there is no calibration or compensation in COUNT mode, there is no Align\_Delay for clock/data. Therefore the total through the IDELAYE3 is equal to the number of taps.



---

**TIP:** When using delay lines in COUNT mode, the EN\_VTC pin must be deasserted (Low). When using delay lines in TIME mode, the EN\_VTC pin must be asserted (High) while IDELAYCTRL.RDY is Low. It can optionally be deasserted after RDY goes High.

---

### **UPDATE\_MODE Attribute**

If UPDATE\_MODE Attribute is set to ASYNC, increments or decrements to the delay value executed on the IDELAY3.CLK clock and is independent of the data being received.

If UPDATE\_MODE Attribute is set to SYNC, increments or decrements to the delay value required by the CLK clock and DATAIN (or IDATAIN) transitions to synchronously update the delay with the DATAIN edges. This mode is suitable for clocks or data that always switch on a periodic basis.

If set to MANUAL, it takes two assertions of LOAD for the new value to take effect. The first LOAD loads the value defined by CNTVALUEIN into the delay line selection register, the second LOAD must be asserted together with an assertion of the CE to make the new value take effect. This is beneficial for designs that need to update an amount of data channels using delay lines because it is possible to update all delay lines when the data becomes IDLE.

**Note:** The preferred method is ASYNC mode because the delay line is updated on the CLK clock of the delay line only, without the need to account for other signals or events.

### **DELAY\_TYPE Attribute**

#### **FIXED Mode**

The DELAY\_TYPE attribute set to FIXED selects the delay through the IDELAYE3 primitive and is determined by the DELAY\_VALUE and DELAY\_FORMAT attribute. When the DELAY\_FORMAT is set to TIME, the value loaded in the delay line is in ps. When the DELAY\_FORMAT is set to COUNT, the delay value loaded in the delay line is the number of taps.

- When DELAY\_FORMAT is TIME, then EN\_VTC must be pulled High so that the delay automatically changes the number of taps over voltage and temperature to ensure the delay stays at the requested time in ps.
- With DELAY\_FORMAT set to COUNT, EN\_VTC must be Low. In COUNT mode, the delay is not compensated for voltage and temperature.

### VARIABLE Mode

The DELAY\_TYPE attribute set to VARIABLE selects the variable tap delay line (Table 2-13). In VARIABLE mode, the CE and INC pins 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. The increment/decrement operation depends on the UPDATE\_MODE attribute (see Figure 2-19). The EN\_VTC pin should be held Low during the delay change command to ensure that any automatic adjustments are stopped.

To increment/decrement delay lines when using TIME mode, use the following steps (see Figure 2-19.):

1. Deassert (Low) the EN\_VTC pin.
2. Wait a minimum of 10 clock cycles.
3. Use the CE and INC ports to increment or decrement the delay line.
4. Wait a minimum of 5 clock cycles.
5. (Option for multiple updates) Increment or decrement of the delay line needs to be performed. Go to step 3, or else proceed to step 6.
6. Wait a minimum of 10 clock cycles.
7. Assert the EN\_VTC pin.

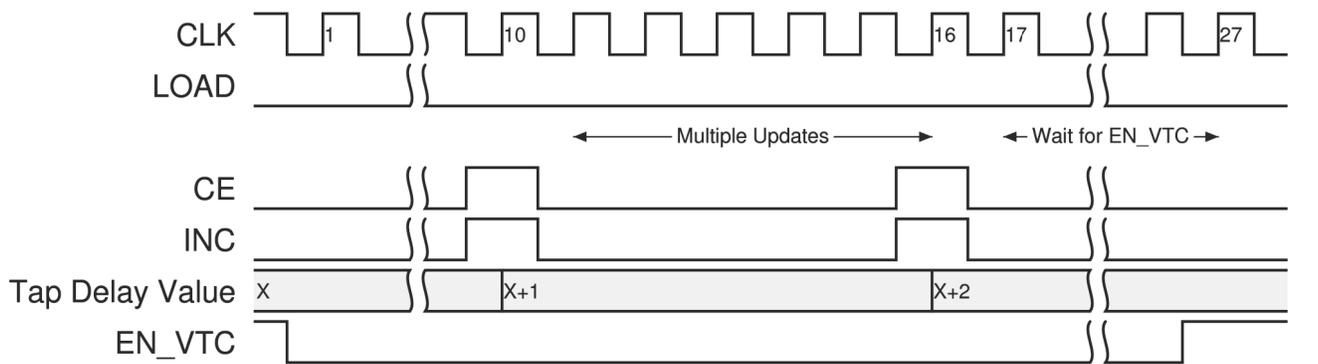
In COUNT mode, the EN\_VTC port is always Low. Use the preceding TIME mode procedure, step 2 through step 4.

Table 2-13: Control Pin when DELAY\_TYPE = VARIABLE<sup>(1)</sup>

EN_VTC	CLK	LOAD	CE	INC	Tap Setting
1	1/0	X	X	X	Not supported, EN_VTC must be Low when LOAD, CE, and INC are active.
0	0	X	X	X	No change
0	1	0	0	X	No change
0	1	0	1	1	Current value +1 Tap <sup>(2)</sup>
0	1	0	1	0	Current value -1 Tap <sup>(2)</sup>
0	1	0	0	0	No change

**Notes:**

1. Only valid port combinations are provided in the table.
2. Value depends upon the UPDATE\_MODE attribute. See Figure 2-19.



X19088-121018

Figure 2-19: Variable Mode, UPDATE\_MODE = ASYNC

### VAR\_LOAD Mode

When the DELAY\_TYPE attribute is set to VAR\_LOAD, the delay line can be changed using the CE and INC inputs or using the CNTVALUEIN and LOAD inputs. The CNTVALUEOUT can in both cases be used to read the current position of the delay line. The CE and INC inputs change the delay line on a per tap basis while the COUNTVALUEIN/OUT buses allow the delay line to be changed dynamically.

The VAR\_LOAD method is suitable for both COUNT and TIME mode usage of the delay line.

In both modes, the tap amount can be read from the CNTVALUEOUT bus and changed through the CNTVALUEIN bus or INC port if necessary.

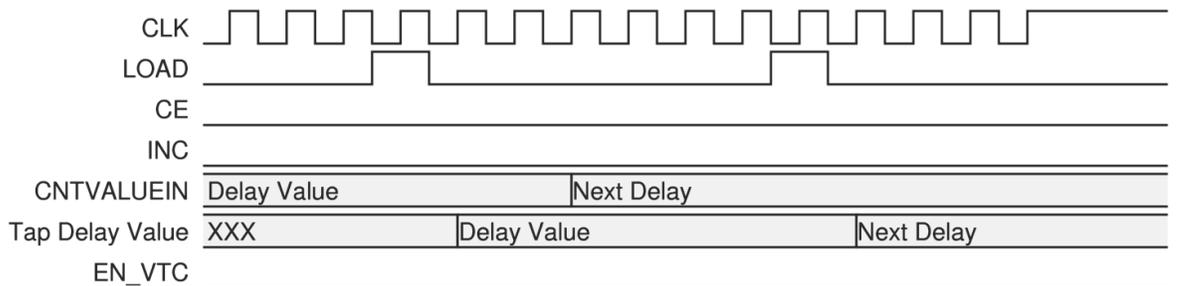
**Note:** Use the explanation of VARIABLE mode when incrementing or decrementing the delay line using the INC/CE input pins. The VAR\_LOAD procedure to calculate the value to update the delay line is different for IDELAY and ODELAY. The VAR\_LOAD procedure to update the delay line is different for TIME and COUNT modes.

If DELAY\_TYPE is VAR\_LOAD and DELAY\_FORMAT is TIME, the procedure to update the delay line follows (see [Figure 2-22](#)).

1. Wait for IDELAYCTRL.RDY to go High.
2. Make EN\_VTC Low to modify the delay line.
3. Wait for at least 10 clock cycles.
4. Read CNTVALUEOUT[8:0] and load the value into a register.
5. Check if updating the delay line is necessary.
6. Calculate the new delay value to be written in the delay line.
7. Put the new delay line value on the CNTVALUEIN[8:0] bus.
8. Wait for at least one clock cycle and pulse LOAD High for a clock cycle.

9. Option for multiple updates: Wait at least 5 clock cycles.
10. Option for multiple updates: Assign a new value to CNTVALUEIN.
11. Option for multiple updates: Wait for one clock cycle and pulse LOAD High for a clock cycle.
12. Option for multiple updates: Go back to [step 9](#) for multiple updates.
13. Wait for at least 10 clock cycles.
14. Pull EN\_VTC back High.
15. Go back to [step 2](#) for a new delay line update.

If DELAY\_TYPE is VAR\_LOAD and DELAY\_FORMAT is COUNT, the procedure to update the delay line follows (see [Figure 2-20](#)):



X26905-072122

Figure 2-20: VAR\_LOAD Mode Using COUNT as DELAY\_FORMAT

1. EN\_VTC is kept Low for COUNT mode.
2. Read CNTVALUEOUT[8:0] and load the value into a register.
3. Check if updating the delay line is necessary.
4. Calculate the new delay value to be written in the delay line.
5. Put the new delay line value on the CNTVALUEIN[8:0] bus.
6. Wait for at least one clock cycle and pulse LOAD High for a clock cycle.
7. (Option for multiple updates) Wait at least four clock cycles.
8. (Option for multiple updates) Assign a new value to CNTVALUEIN.
9. (Option for multiple updates) Wait for one clock cycle and pulse LOAD High for a clock cycle.
10. (Option for multiple updates) Go back to [step 7](#) for multiple updates.

To calculate new values to be written in delay lines, the following details must be known:

- A delay line has 512 taps and is at least 1250 ps for UltraScale devices or 1100 ps for UltraScale+ devices.

- The delay range of a single tap is specified in the UltraScale device data sheet [\[Ref 2\]](#).

Delay lines are not calibrated before the FPGA is downloaded and the BISC engine has run.

As such the real delay of a single tap in an FPGA is unknown.

In TIME mode:

- The initial DELAY\_VALUE, in the design attribute, must be provided in ps.
- Afterward, the initial delay setting can be modified by writing a value represented as a number of taps into the delay line.
- The BISC process uses a number of taps of an input delay line to eliminate the insertion delay difference between the data and the clock at the first data capture flip-flops of the receiver. This delay is called Align\_Delay. Total delay provided by IDELAYE3 is the sum of the Align\_Delay and DELAY\_VALUE.
- The Align\_Delay can be between 45 and 65 taps. It averages 50 to 54 taps.
- When writing all zeros or an amount of taps smaller than the reported Align\_Delay to an input delay line, the tuned Align\_Delay is impacted.
- An output delay line does not have this feature so the total output delay provided by ODELAY is equal to the DELAY\_VALUE, because the output flip-flops act before the output delay line and BISC does not need to run tuning for the middle of the data eye.
- The BISC process is always running in the background to compensate for voltage and temperature variations.

In COUNT mode:

- The initial DELAY\_VALUE, in the design attribute, **must** be provided in taps.
- The BISC procedure is not used and the real delay value of a tap cannot be known.
- There is no voltage and temperature compensation for the delay lines because BISC does not run.
- In COUNT mode, the delay line **must** be used as a delay of a maximum of 512 taps.
- Measurements and adjustments must be calculated in taps. For example:
  - A measurement of a data eye is expressed as 450 taps.
  - Jitter between two data eyes is expressed as 31 taps.

When DELAY\_TYPE is VAR\_LOAD and DELAY\_FORMAT is COUNT, a delay line is used in *bare-metal* mode, because only the depth or amount of taps of the delay line are important (512 in case of UltraScale devices).

Thus, this is the only parameter a design using COUNT mode must take care of. The value of a measured data, clock, or strobe eye is expressed as an amount of taps without providing

the delay this represents. It is thus not necessary to calculate the delay of a single tap, and all 512 provided taps in a delay line are available to the user.

When DELAY\_TYPE is VAR\_LOAD and DELAY\_FORMAT is TIME, the Align\_Delay must be measured and the single tap delay must be calculated if a new delay time must be set into a delay line. Two input delay lines must be used to calculate the delay value of a single tap.

When using single-ended inputs, two inputs are necessary to calculate the single tap delay, because behind each input pad with input buffer (IBUF) there is an IDELAYE and a ISERDESE (Figure 2-21).

When using differential inputs, a single data channel input can be used to calculate the single tap delay. A differential input occupies two pads and thus it also covers two IDELAY/ISERDES. When a normal differential input buffer (IBUFDS) is used, only the even ISERDES of the two is used. When using a differential input buffer with differential output (IBUFDS\_DIFF\_OUT), you can use both ISERDESs covered by the two input pads. This is the solution for measuring a single tap value for a single differential data channel (Figure 2-21).

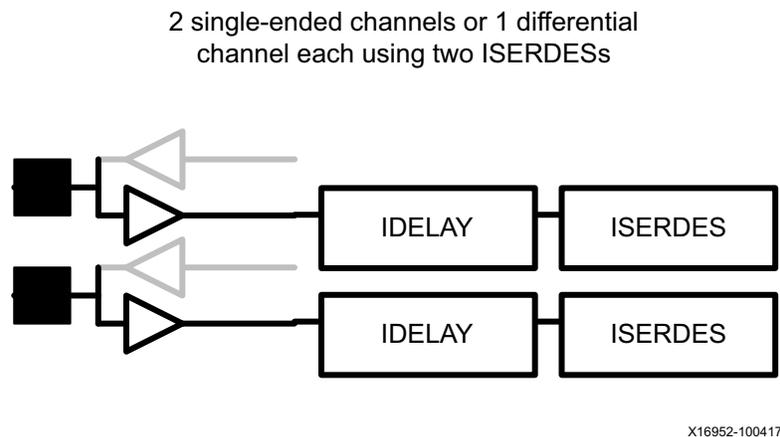


Figure 2-21: Two Single-ended or One Differential RX Channel

To measure Align\_Delay and calculate a single tap delay:

1. In the HDL design, for the even bit slice, set the DELAY\_VALUE to zero.
2. In the HDL design, for the odd bit slice, set the DELAY\_VALUE to a larger non-zero value, for example 700 ps.
3. When the design is downloaded and running in an FPGA, read CNTVALUEOUT of both delay lines and store the amount of taps obtained in a set of registers.

The tap value from the even bit slice is the Align\_Delay and that from the odd bit slice is the total delay value (Align\_Delay + Requested value = Total\_Value).

4. The requested delay value, 700 ps in this case, is represented by Equation 2-1:

$$\text{Total\_Value} - \text{Align\_Delay} = n \text{ taps} \tag{Equation 2-1}$$

5. The delay of a single tap is then equal to Equation 2-2:

$$\text{odd channel DELAY\_VALUE} / n \text{ taps} = \text{single tap} \tag{Equation 2-2}$$

6. The new CNTVALUEIN value to write to the delay line or lines used in taps is shown in Equation 2-3 for IDELAYs or Equation 2-4 for ODELAYs:

$$\text{CNTVALUEIN} \langle \text{IDELAY} \rangle = (\text{wanted delay} / \text{single tap}) + \text{Align\_Delay} \tag{Equation 2-3}$$

$$\text{CNTVALUEIN} \langle \text{ODELAY} \rangle = (\text{wanted delay} / \text{single tap}) \tag{Equation 2-4}$$

7. Write this new value in the delay line using the delay line update procedure.



**TIP:** When using an IBUFDS\_DIFF\_OUT, both IDELAY and ISERDES can be used to capture data. The even one captures the p-side of the differential data channel while the odd one captures the n-side. To use the n-side data in FPGA logic, inverse the data output of the ISERDES.

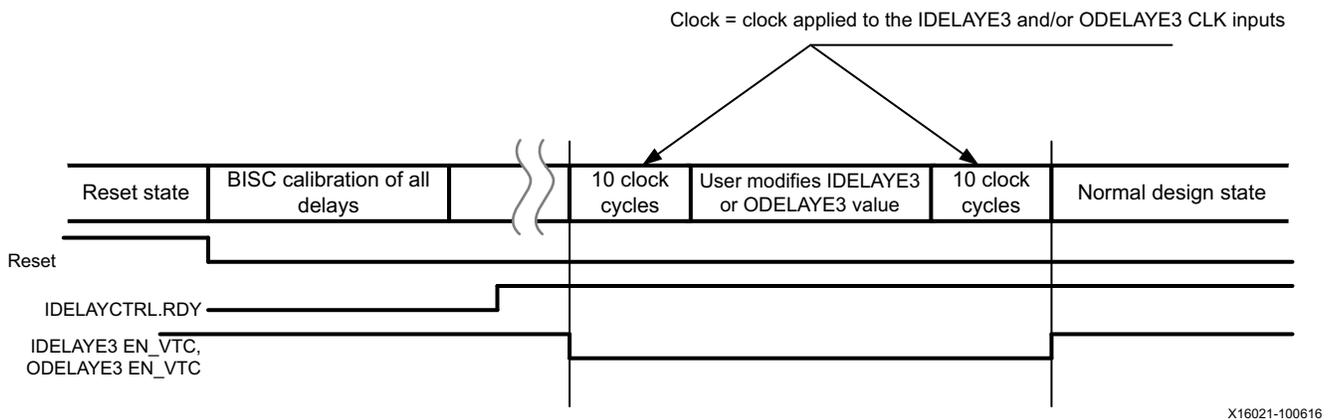


Figure 2-22: Changing Delay when DELAY\_TYPE = VAR\_LOAD

Table 2-14: Control Pin when DELAY\_TYPE = VAR\_LOAD

CLK	LOAD	CE	INC	CNTVALUEIN	CNTVALUEOUT	Tap setting
0	X	X	X	X	X	No change
1	1	0	X	CNTVALUEIN	CNTVALUEIN	CNTVALUEIN
1	1	1	X	X	X	Not a valid combination, CE must be Low during LOAD
1	0	1	1	X	Current value + 1	Current value + 1 <sup>(1)</sup>
1	0	1	0	X	Current value - 1	Current value - 1 <sup>(1)</sup>
1	0	0	0	X	No change	No change

**Notes:**

1. Value depends upon the UPDATE\_MODE attribute.

## ODELAYE3

Any output signal can be delayed using the ODELAYE3 primitive, having been either forwarded from the device logic directly or registered in a simple flip-flop or OSERDES using an SDR or DDR clock.

The ODELAYE3 primitive (Figure 2-23) contains a 512-tap delay line. (See the tap resolution in the UltraScale device data sheet [Ref 2]). Each individual tap is uncalibrated. However, the logic to calibrate the delay line is available in the IDELAYCTRL component.

The ODELAYE3 can be used in two modes:

- COUNT mode:
  - No need to use an IDELAYCTRL component because the delay line is used in the uncalibrated state without voltage and temperature compensation.
  - The delay line must count only taps and not count delay/tap.
  - The DELAY\_VALUE is expressed as taps (0 to 511).
  - Example: Scanning a serial data stream for transitions must be expressed in a number of taps and not translated to time in ps.
- TIME mode:
  - An IDELAYCTRL component **must** be used.
  - The delay line is calibrated for the requested time value and voltage/temperature compensation makes sure that this value is kept over time.
  - The DELAY\_VALUE is expressed in ps.

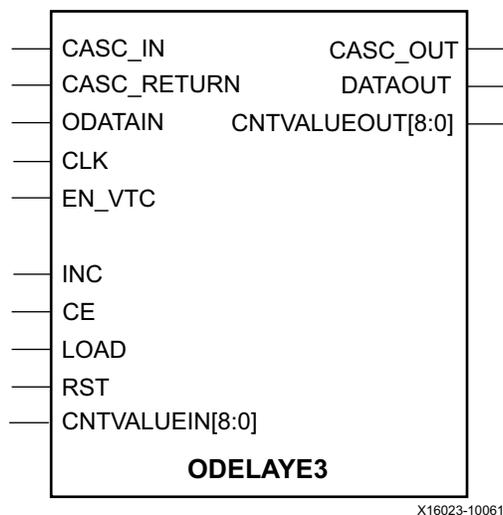


Figure 2-23: ODELAYE3 Primitive

## ODELAYE3 Ports

Table 2-15 lists the ODELAYE3 ports.

Table 2-15: ODELAYE3 Ports

Port	I/O	Description
CASC_RETURN	Input	The CASC_RETURN pin is the output cascade delay returning from slave IDELAYE3/ODELAYE3. The CASC_RETURN of the ODELAYE3 is connected to the slave IDELAYE3 DATAOUT port.
CASC_IN	Input	The CASC_IN pin is used when the ODELAYE3 is used in a cascade chain as a slave input cascade delay from IDELAYE3 CASC_OUT.
CASC_OUT	Output	The CASC_OUT pin is used when cascading from an ODELAYE3 to an IDELAYE3. The CASC_OUT port of the ODELAYE3 is connected to the CASC_IN of the IDELAYE3 in cascade.
CE	Input	Clock enable for the delay register clock. <b>Note:</b> Delays might take up to three clock cycles (CLK) to be applied. During this time, input data should not change to ensure output data does not glitch.
CLK	Input	All control inputs to the ODELAYE3 primitive (LOAD, CE, and INC) are synchronous to the clock input (CLK). A clock must be connected to this port when ODELAYE3 is configured in VARIABLE or VAR_LOAD modes. The CLK can be locally inverted. The CLK of the ODELAYE3 must be the same CLK as the OSERDESE3 CLKDIV or the ODDRE1 C port (UltraScale devices only). For UltraScale+ devices, the C port is remapped to CLK in XPIO banks. Unused when configured in FIXED mode.
INC	Input	The increment/decrement is controlled by the enable signal (CE). This interface is only available when the ODELAYE3 is in VARIABLE or VAR_LOAD modes. As long as CE remains High, the ODELAYE3 increments or decrements by one tap every CLK cycle. The state of INC determines whether ODELAYE3 increments or decrements; INC = 1 increments, INC = 0 decrements, synchronously to the CLK. If CE is Low, the delay through ODELAYE3 does not change regardless of the state of INC. When CE transitions High, the increment/decrement operation begins on the next positive clock edge. When CE transitions Low, the increment/decrement operation ceases on the next positive clock edge.  The programmable delay taps in the ODELAYE3 primitive wraps around to the start or end of the taps. When the last tap delay is reached (tap 512), a subsequent increment function returns to tap 0. The same applies to the decrement function—a decrement from zero moves to tap 512.
LOAD	Input	Loads counter value from attribute DELAY_VALUE or bus CNTVALUEIN. When in VAR_LOAD mode, the ODELAYE3 LOAD port loads the value set by the CNTVALUEIN into registers connected to the delay line tap selection logic. The value present at CNTVALUEIN[8:0] is the new tap value. The LOAD signal is an active-High signal and is synchronous to the input CLK signal. Wait at least one clock cycle after applying a new value on the CNTVALUEIN bus before applying the LOAD signal. CE must be held Low during LOAD operation. When in VARIABLE mode, connect LOAD to ground (GND). <b>Note:</b> Delays might take up to three clock cycles (CLK) to be applied. During this time, input data should not change to ensure output data does not glitch.

Table 2-15: ODELAYE3 Ports (Cont'd)

Port	I/O	Description
CNTVALUEIN[8:0]	Input	The CNTVALUEIN pins are used for dynamically switching the loadable tap value. The CNTVALUEIN is the number of taps required. The new value is best applied one clock cycle before applying the LOAD signal. The delay line can be changed from 1 to 8 taps at a time.
CNTVALUEOUT[8:0]	Output	The CNTVALUEOUT pins are used for reporting the current tap value and reads out the amount of taps in the current delay. When EN_VTC is High, CNTVALUEOUT is updated by the IDELAYCTRL.
ODATAIN	Input	The ODATAIN input is driven by the ODDRE1Q port or the OSERDESE3 (OQ).
DATAOUT	Output	The DATAOUT port is the output port of the ODELAYE3 and connects to the output IOB.
RST	Input	The RST pin (reset) is an asynchronous input. When the ODELAYE3 is reset, the delay is set to the value defined by the DELAY_VALUE attribute. RST must follow the Component Mode Reset Sequence when used with the IDELAYCTRL. After IDELAYCTRL.RDY goes High, ODELAY can be used for normal operation.
EN_VTC	Input	EN_VTC: Enable voltage temperature compensation. <ul style="list-style-type: none"> <li>High: Enables IDELAYCTRL to keep delay constant over VT.</li> <li>Low: VT compensation is disabled.</li> </ul> To make delay line updates, EN_VTC is an asynchronous input but must be kept Low. EN_VTC must follow the Component Mode Reset Sequence when used with the IDELAYCTRL.

### ODELAYE3 Attributes

Table 2-16 lists the ODELAYE3 attributes.

Table 2-16: ODELAYE3 Attributes

Attribute	Values	Default	Type	Description
CASCADE	NONE MASTER SLAVE_MIDDLE SLAVE_END	NONE	String	For more information, see <a href="#">CASCADE Attribute</a> .
DELAY_TYPE	FIXED VAR_LOAD VARIABLE	FIXED	String	The DELAY_TYPE attribute sets the type of delay used, it can be FIXED, VARIABLE, or VAR_LOAD. See the <a href="#">DELAY_TYPE Attribute</a> for further information.
DELAY_VALUE	0–1250 (TIME UltraScale) 0–1100 (TIME UltraScale+)	0	Decimal	TIME mode: Desired value in ps. UltraScale devices support delays up to 1.25 ns. UltraScale+ devices support up to 1.1 ns. COUNT mode: Desired value in taps. For more information, see <a href="#">DELAY_VALUE Attribute</a> .

Table 2-16: ODELAYE3 Attributes (Cont'd)

Attribute	Values	Default	Type	Description
REFCLK_FREQUENCY	200.00–800.00 (UltraScale IDELAYCTRL) 300.00–800.00 (UltraScale+ IDELAYCTRL) 200.00–2400.00 (UltraScale BITSlice_CONTROL) 300.00–2666.67 (UltraScale+ BITSlice_CONTROL)	300.0	1 significant digit float	The REFCLK_FREQUENCY attribute specifies the reference clock of the IDELAYCTRL frequency in MHz. This attribute <b>must</b> mimic the clock frequency applied at the IDELAYCTRL component except when DELAY_FORMAT is set to COUNT, in which case the attribute can be left at the default value.
DELAY_FORMAT	TIME <sup>(1)</sup> COUNT	TIME	String	When set to TIME, the delay equals the value given in DELAY_VALUE, specified in ps, and is calibrated by the IDELAYCTRL primitive using the REFCLK port input. The IDELAYCTRL.REFCLK must be reflected in the ODELAY attribute REFCLK_FREQUENCY. When set to COUNT, the initial tap setting goes to whatever number of taps is specified in DELAY_VALUE. This does not give a constant delay because the tap delays vary with PVT. The number of taps in the delay line is important in COUNT mode. For more information, see <a href="#">DELAY_FORMAT Attribute</a> .
UPDATE_MODE	ASYNC SYNC MANUAL	ASYNC	String	For more information, see <a href="#">UPDATE_MODE Attribute</a> .
SIM_DEVICE	ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2	ULTRASCALE	String	Sets the device version (ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2)

**Notes:**

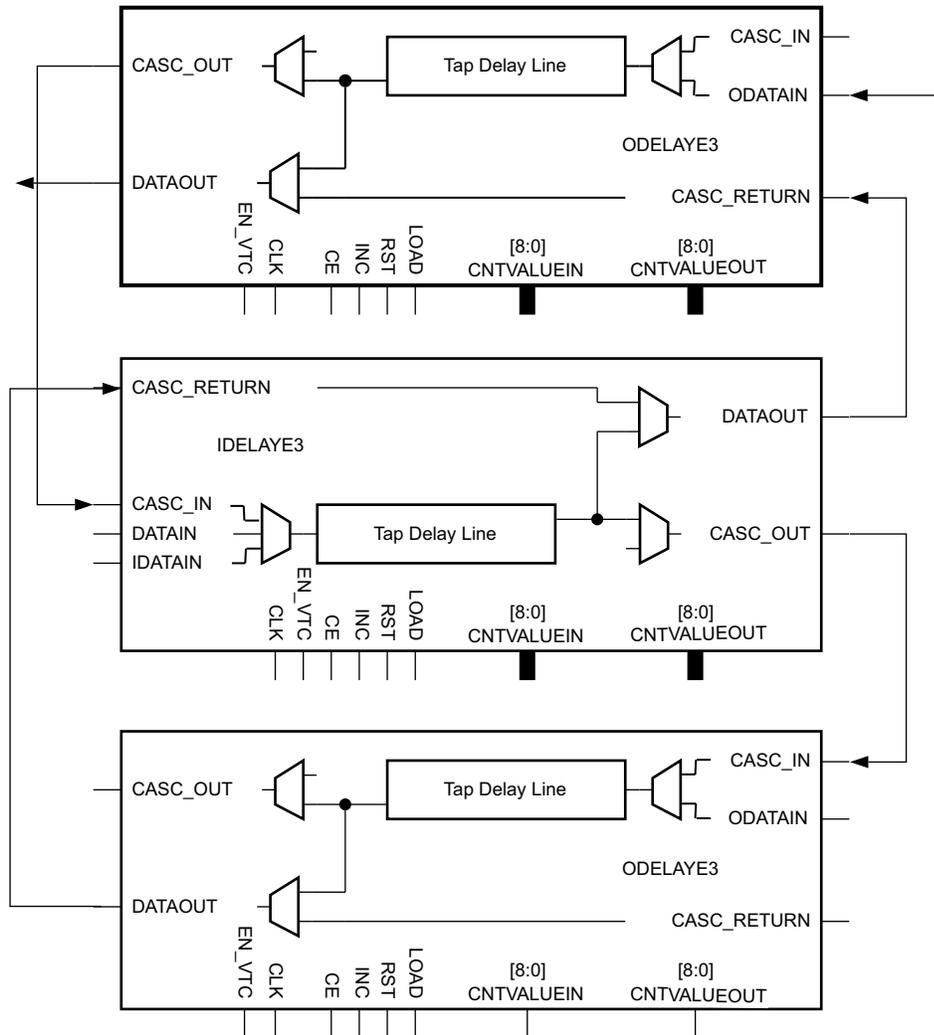
- When in TIME mode, calibration affects the availability of bit slices within the nibble. See [Bank Overview](#) for more information.

**CASCADE Attribute**

The CASCADE attribute is set to NONE if the delay line is not cascaded. Cascading is used when a delay greater than 1.25 ns is required. The connections between delay elements are shown in [Figure 2-24](#). When using the ODELAYE3 (or IDELAYE3) for cascading, the delay (and the IOB) are no longer available to the design. The delay elements can be cascaded up

to the byte boundary in a downward direction. Therefore, the maximum length of the delay depends on where the I/O is placed in the byte.

The routes used to cascade IDELAYE3 and ODELAYE3 are dedicated, high-speed routes. The total fixed intrinsic insertion delay for an IDELAYE3 or ODELAYE3 cascade is the sum of the initial insertion delay plus the cascaded insertion delay. This delay grows as multiple of the times that IDELAYE3 and ODELAYE3 are cascaded. However, the delay is always a fixed value.



X16024-050516

Figure 2-24: ODELAYE3 Cascading to Three Delays

**DELAY\_FORMAT Attribute**

The tap size of the ODELAYE3 primitive is defined in the UltraScale device data sheets as  $T_{ODELAY\_RESOLUTION}$  ( $T_{IDELAY\_RESOLUTION}$  for IDELAYE3) [Ref 2]. If the DELAY\_FORMAT is set to

TIME, the delay line is calibrated, controlled, and maintained for voltage and temperature by the IDELAYCTRL component.

- An IDELAYCTRL component **must** be used.
- The REFCLK\_FREQUENCY attribute must reflect the clock frequency applied to the IDELAYCTRL component.
- The EN\_VTC pin must be actively manipulated when the delay line is used in VARIABLE or VAR\_LOAD mode. When FIXED mode is used, tie the EN\_VTC pin High.

When the DELAY\_FORMAT is set to COUNT, the delay line is not calibrated and is not maintained over voltage and temperature. Therefore:

- Do not use an IDELAYCTRL component.
- Leave the REFCLK\_FREQUENCY attribute at the default value (300 MHz).
- Tie the EN\_VTC input pin Low.
  - This pin ensures that calibration and VT maintenance logic in the ODELAYE3 is disabled.
- The delay line must be used to represent an amount of taps.
  - It does not matter how long the tap delay is; it is the amount of taps that is important.
  - 512 taps are available.
- The CNTVALUEIN/OUT[8:0] values represent the amount of taps the delay line is set or tuned to.

Examples of how the DELAY\_FORMAT attribute is used are provided in the mode paragraphs in the [DELAY\\_TYPE Attribute, page 184](#).

### ***DELAY\_VALUE Attribute***

When the DELAY\_FORMAT attribute is set to TIME mode, the DELAY\_VALUE attribute represents time in ps. Unlike IDELAYE3, the ODELAYE3 has no clock/data align delay. The total delay through the ODELAYE3 is thus the value of the DELAY\_VALUE.

In TIME mode, the DELAY\_VALUE represents time in ps, but the value read or written from or to the delay line by the CNTVALUEIN[8:0] and/or CNTVALUEOU[8:0] is expressed in taps. So changing the time of a delay line requires some calculation, which is provided in the DELAY\_MODE/VAR\_LOAD paragraph. When the DELAY\_FORMAT attribute is set to COUNT mode, the DELAY\_VALUE attribute represents an amount of taps.



---

**TIP:** When using delay lines in COUNT mode, the EN\_VTC pin must be deasserted (Low). When using delay lines in TIME mode, the EN\_VTC pin must be asserted (High) while IDELAYCTRL.RDY is Low. It can optionally be deasserted after RDY goes High.

---

### **UPDATE\_MODE Attribute**

Leave this attribute set to ASYNC to increment or decrement to the delay value independent of the data being received. When set otherwise (SYNC), the delay line is synchronously updated upon receiving changing input data.



---

**CAUTION!** *When no precautions are taken in the application design, the attribute set to SYNC might cause unwanted effects such as sudden data glitches in the design.*

---

### **DELAY\_TYPE Attribute**

#### **FIXED Mode**

The DELAY\_TYPE attribute set to FIXED selects the delay through the ODELAYE3 primitive and is determined by the DELAY\_VALUE and DELAY\_FORMAT attributes. When DELAY\_FORMAT is set to TIME, the value loaded in the delay line is in ps. When the DELAY\_FORMAT is set to COUNT, the delay value loaded in the delay line is the number of taps.

- When DELAY\_FORMAT is TIME, EN\_VTC must be pulled High so that the delay automatically changes the number of taps over voltage and temperature to ensure the delay stays at the requested time in ps.
- With DELAY\_FORMAT set to COUNT, EN\_VTC must be Low in Count mode. Then the delay is not compensated for voltage and temperature.

#### **VARIABLE Mode**

The DELAY\_TYPE attribute set to VARIABLE selects the variable tap delay line (Table 2-13). In VARIABLE mode, the CE and INC pins are used to manually increment and decrement the delay line tap per tap (INC/DEC increments/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. The increment/decrement operation depends on the UPDATE\_MODE attribute (see Figure 2-19). The EN\_VTC pin should be held Low during the delay change command to ensure that any automatic adjustments are stopped.

To increment/decrement delay lines when using TIME mode, use the following procedure (see Figure 2-14):

1. Deassert (Low) the EN\_VTC pin.
2. Wait a minimum of 10 clock cycles.
3. Use the CE and INC ports to increment or decrement the delay line.
4. Wait a minimum of 5 clock cycles.
5. (Option for multiple updates) Increment or decrement of the delay line needs to be performed. Go to [step 3](#), or else proceed to [step 6](#).

6. Wait a minimum of 10 clock cycles.
7. Assert the EN\_VTC pin.

When using the delay line in COUNT mode, the EN\_VTC port is always Low. Use the TIME mode procedure in [step 2](#) through [step 4](#) in this [VARIABLE Mode](#) section.

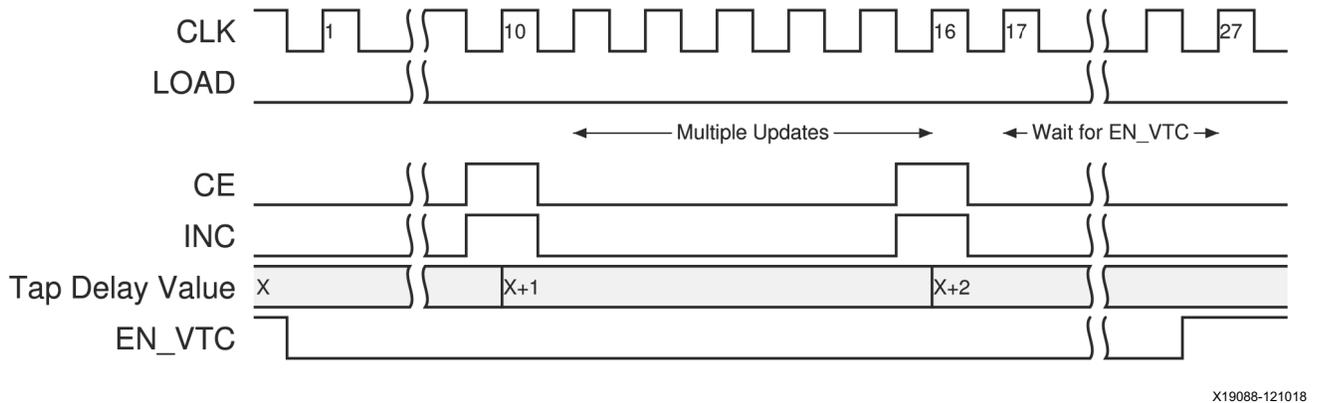


Figure 2-25: Variable Mode, UPDATE\_MODE = ASYNC

### VAR\_LOAD Mode

The VAR\_LOAD method is suitable for both COUNT and TIME mode usage of the delay line.

In both modes, the tap amount can be read from the CNTVALUEOUT bus and changed through the CNTVALUEIN bus or INC port if necessary.

**Note:** The procedure to calculate the value to update the delay line is different for IDELAY and ODELAY, and different for TIME and COUNT mode.

If DELAY\_TYPE is VAR\_LOAD and DELAY\_FORMAT is TIME, the procedure to update the delay line follows (see [Figure 2-22](#)):

1. Wait for DELAYCTRL.RDY to go High.
2. Make EN\_VTC Low to modify the delay line.
3. Wait for at least 10 clock cycles.
4. Read CNTVALUEOUT[8:0] and load the value into a register.
5. Check if updating the delay line is necessary.
6. Calculate the new delay value to be written in the delay line.
  - a. Increment or decrement the current tap position (Org\_Val) by 8 taps for glitchless transition. Jumps higher than 8 taps might result in the delay line jump causing data to glitch. For outputs the glitch can corrupt the serialization.

**Note:** This step might require fewer than 8 taps.

- b. Put the new delay line value on the CNTVALUEIN[8:0] bus.
- c. Wait for one clock cycle and pulse LOAD High for a clock cycle.
- d. Check if the new delay line value (New\_Val) is reached.
  - If not, wait 5 clock cycles and continue from [step a](#).
  - If so, continue to [step 7](#).

or

- a. Calculate the difference (Dif\_Val) between New\_Val and Org\_Val and the direction to step.
  - b. Make the INC input High or Low to increment or decrement the delay line.
  - c. Toggle the CE pin to execute the increment or decrement.
  - d. Decrement the Dif\_Val and check if it is zero.
    - If not continue from [step a](#).
    - If so, continue to [step 7](#).
7. Wait for at least 10 clock cycles.
  8. Set EN\_VTC High for VT compensation.
  9. Go back to [step 2](#) for a new delay line update.

If DELAY\_TYPE is VAR\_LOAD and DELAY\_FORMAT is COUNT, the procedure to update the delay line follows:

1. After IDELAYCTRL.RDY goes High, EN\_VTC is kept Low.
2. Read CNTVALUEOUT[8:0] and load the value into a register (Org\_Val).
3. Check if updating the delay line is necessary.
4. Calculate the new delay value, in taps, to be written in the delay line (New\_Val).
  - a. Increment or decrement the current tap position (Org\_Val) by 8 taps for glitchless transition. Jumps higher than 8 taps might result in the delay line jump causing data to glitch. For outputs the glitch can corrupt the serialization.

**Note:** This step might require fewer than 8 taps.
- b. Put the new delay line value on the CNTVALUEIN[8:0] bus.
- c. Wait for one clock cycle and pulse LOAD High for a clock cycle.
- d. Check if the new delay line value (New\_Val) is reached.
  - If not, continue from [step 4](#).
  - If so, continue to [step 5](#).

or

- a. Calculate the difference (Dif\_Val) between New\_Val and Org\_Val and the direction to step.
- b. Make the INC input High or Low to increment or decrement the delay line.
- c. Toggle the CE pin to execute the increment or decrement.
- d. Decrement the Dif\_Val and check if it is zero.
  - If not, continue from [step 4](#).
  - If so, continue to [step 5](#).
5. Wait for at least 10 clock cycles.
6. Go back to [step 2](#) for a new delay line update.

## IDELAYCTRL

If the IDELAYE3 (or ODELAYE3) primitives are instantiated, the IDELAYCTRL module must be instantiated, except when the DELAY\_FORMAT is set to COUNT or when mixing component and native mode in native mode designs (see [Mixing Native and Non-Native Mode I/O in a Nibble](#)). There is one IDELAYCTRL module per nibble (eight per bank). The IDELAYCTRL module continuously calibrates the individual delay lines configured in TIME mode in its region to their programmed value to reduce the effects of process, voltage, and temperature (PVT) variations. The IDELAYCTRL module calibrates IDELAYE3 (and ODELAYE3) using the system-supplied REFCLK. The frequency value of this REFCLK is applied to individual IDELAYE3 (and ODELAYE3) primitives with an attribute (REFCLK\_FREQUENCY). Each delay element in a nibble therefore requires having this attribute set to the same value. [Figure 2-26](#) shows a block diagram of the IDELAYCTRL module.



---

**TIP:**

1. Resetting the IDELAYCTRL component when the delay lines are used in TIME mode re-invokes the BISC of the nibble that used delay lines.
  2. When the EN\_VTC pins of the used IDELAYE3 / ODELAYE3 are not set correctly, the IDELAYCTRL.RDY pin never is asserted High by the BISC controller.
  3. Within each bank, all of the used IDELAYCTRLs and BITSLICE\_CONTROLS have a cascaded RDY signal requiring the reset to be asserted at the same time.
- 



**CAUTION!** IDELAYE3, ISERDESE3, and IDDRE1 lines used and positioned at I/O positions labeled with DBC and/or QBC are not functional during the BISC stage. These components are available after the IDELAYCTRL.RDY pin is asserted High.

---

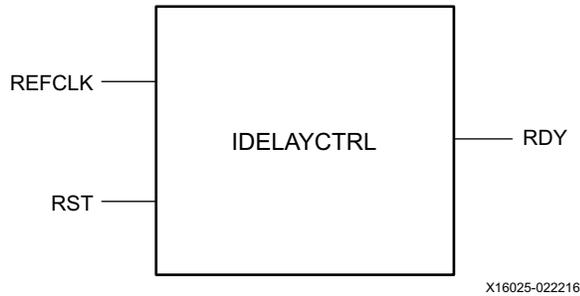


Figure 2-26: IDELAYCTRL Module

### IDELAYCTRL Ports

Table 2-17 lists the IDELAYCTRL ports.

Table 2-17: IDELAYCTRL Ports

Port	I/O	Type	Description
REFCLK	Input	Clock	Reference clock for delay calibration.
RST	Input	Reset	Active-High asynchronous reset for IDELAYCTRL. <b>Note:</b> The reset for all used IDELAYCTRLs/BITSLICE_CONTROLS within a bank must be released at the same time due to the cascaded DLY_RDY connections between the BITSLICE_CONTROLS. Failure to do so might result in DLY_RDY for one of the IDELAYCTRLs/BITSLICE_CONTROLS not asserting.
RDY	Output	Data	The ready signal goes High to signal that controlled IDELAYE3 and ODELAYE3 primitives are calibrated.

Table 2-18: IDELAYCTRL Attribute

Attribute	Values	Default	Type	Description
SIM_DEVICE	7SERIES, ULTRASCALE	ULTRASCALE	String	Set to ULTRASCALE for UltraScale and UltraScale+ devices.

## Component Mode Reset Sequence

To operate the Component mode primitives correctly (including IDDR and ODDR primitives), follow this reset sequence:

### Apply Reset

1. The EN\_VTCs are High for all of the USED IDELAYs and ODELAYs.
2. Assert Reset to the PLL/MMCM, which generates the clock for IDELAY and IDELAYCTRL.
3. Apply Reset to IDELAYCTRL, IDELAY TIME mode, ISERDES, OSERDES, and ODELAY TIME mode.

4. Wait the minimum PLL/MMCM reset assertion time before releasing the reset. For this timing specification, consult the PLL/MMCM section of the UltraScale device data sheets [Ref 2].

### **Release Reset**

1. Hold all the EN\_VTCs High for all of the used IDELAYs and ODELAYs.
2. Use the following sequence to bring the I/O out of reset:
  - a. Release the reset of the PLL/MMCM generating the clocks for the interface.
  - b. Wait for the PLL/MMCM to reach the LOCKED state.
  - c. Release the reset of following primitives: IDELAY, ODELAY, ISERDES, and OSERDES.
  - d. After [step c](#) has completed, release the reset for IDELAYCTRL.
  - e. Wait until the RDY of all the used IDELAYCTRL primitives are asserted High.

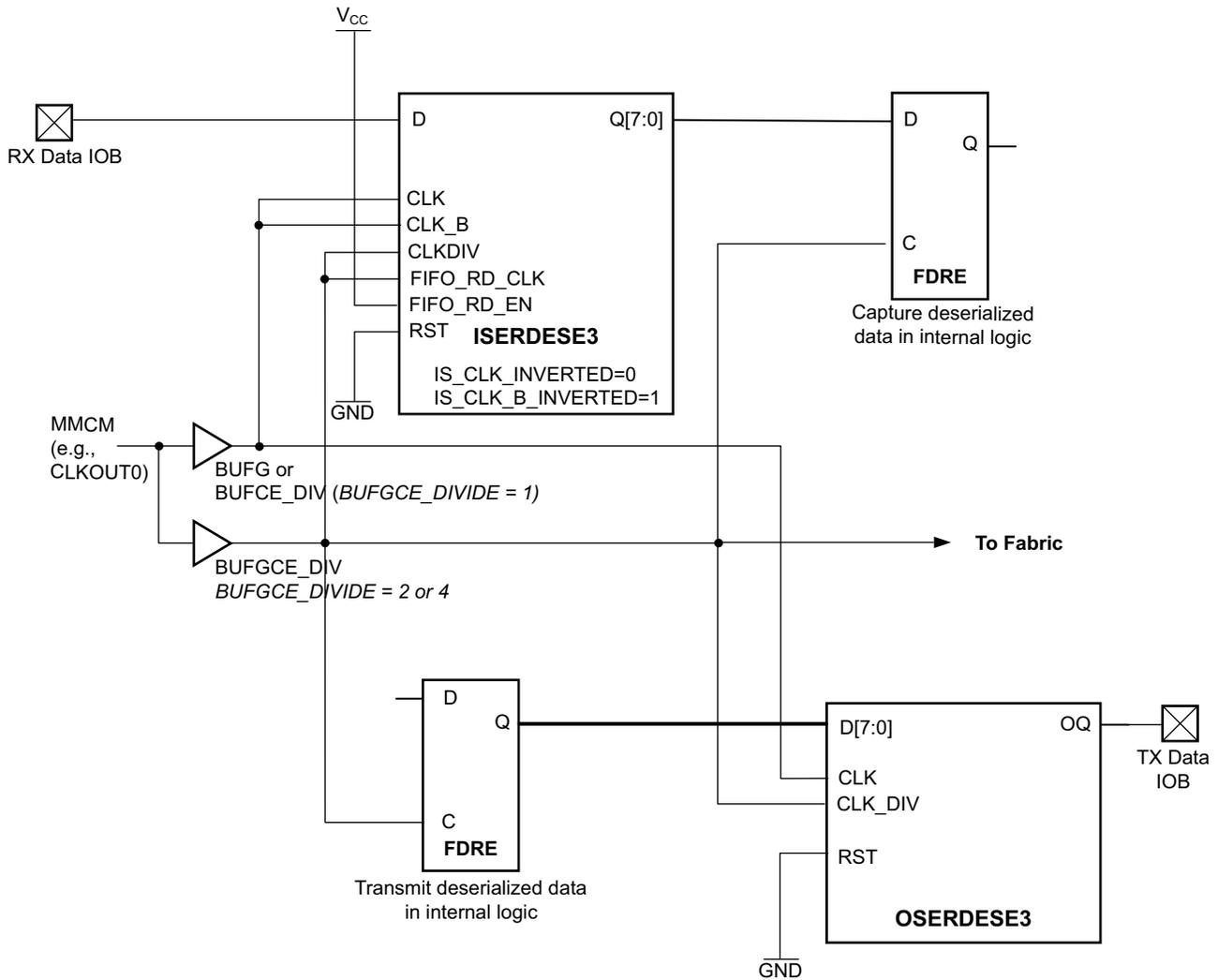
Now the application in the FPGA logic can be released after a delay of at least 64 clock cycles.

## **Clocking Considerations Using Component Primitives**

In Component mode, the ISERDES/OSERDES component clocks must be driven from global clocking. The clocks can be sourced from any of the following global clock resources:

- Clock-capable I/O driving a BUFGCE or BUFGCE\_DIV
- MMCM driving a BUFGCE or BUFGCE\_DIV
- PLL driving a BUFGCE/BUFGCE\_DIV

A typical Component mode receive and transmit clocking topology using SerDes is shown in Figure 2-27.



X16048-042117

Figure 2-27: Component Mode Clocking Circuit

In the receive circuit, the ISERDESE3 is configured with the FIFO enabled (attribute FIFO\_ENABLE = TRUE). The incoming serialized data is captured in the ISERDESE3 using a high-speed clock sourced from a clock-capable I/O driving a BUFGCE, connected to the ISERDESE3 CLK/CLK\_B pins. The deserialized data is read out by a divided version of the high-speed clock, with the division factor relating to the deserialized width (SerDes attribute DATA\_WIDTH). For example, with DATA\_WIDTH = 8, the clock is divided by 4, assuming it is a DDR transmission. The circuit in Figure 2-27 uses a BUFGCE\_DIV to perform the division. The divided clock is connected to both the CLKDIV and FIFO\_RD\_CLK of the ISERDESE3. An alternative uses a capture circuit with a disabled ISERDESE3 FIFO (attribute FIFO\_ENABLE = FALSE). In this arrangement (not shown), the FIFO\_RD\_CLK signal should not be connected, although the CLKDIV signal must still be in place. The deserialized data

is output from the ISERDESE3 using an automatic, internally generated, divided clock. In this mode, static timing analysis using the Vivado design tools shows the ISERDESE3 read timing relative to this internally generated divided clock.

The Component mode transmit circuit is also shown in [Figure 2-27](#). The parallel/deserialized transmit data is sampled at the OSERDESE3 data inputs using a divided clock that must be supplied to the OSERDESE3 CLKDIV input. Like the ISERDESE3, the divided clock can be generated using a BUFGCE\_DIV (as shown), or alternatively using an MMCM or PLL. The serialized data is output from the OSERDESE3 using the supplied high-speed clock connected to the OSERDESE3 CLK input. When the IDDRE1 and ODDRE1 are used instead of the ISERDESE3 and OSERDESE3 (for a deserialized width of 2), connect the CLK input to the high-speed global clock. No divided clock is necessary. Although not shown in [Figure 2-27](#), it is also possible to insert an IDELAYE3 in the receive circuit and an ODELAYE3 in the transmit circuit between the IOB and the SerDes.

In [Figure 2-27](#), a single clock source from an MMCM output drives the BUFG and BUFGCE\_DIV to minimize clock skews. In this situation, clock skews are analyzed by Vivado.

**Note:** When using the BUFGCE\_DIV, the divided clock is not guaranteed to be aligned, which is why fabric logic should be driven by the BUFGCE\_DIV.

In some situations, multiple clock outputs from the MMCM are required see [Figure 2-28](#). Skews are introduced by the MMCM outputs, making clock skew hard to meet. To ensure clock skews are correctly calculated, define a CLOCK\_DELAY\_GROUP for the clock buffers.

```
set_property CLOCK_DELAY_GROUP <Clock Delay Group Name> [get_nets-of_objects
[get_pins <BUFG CLKOUT1 Instance>/O ]

set_property CLOCK_DELAY_GROUP <Clock Delay Group Name> [get_nets-of_objects
[get_pins <BUFG CLKOUT2 Instance>/O ]
```

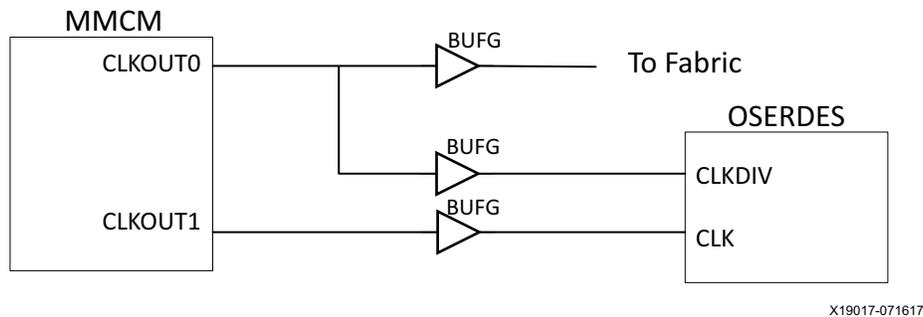
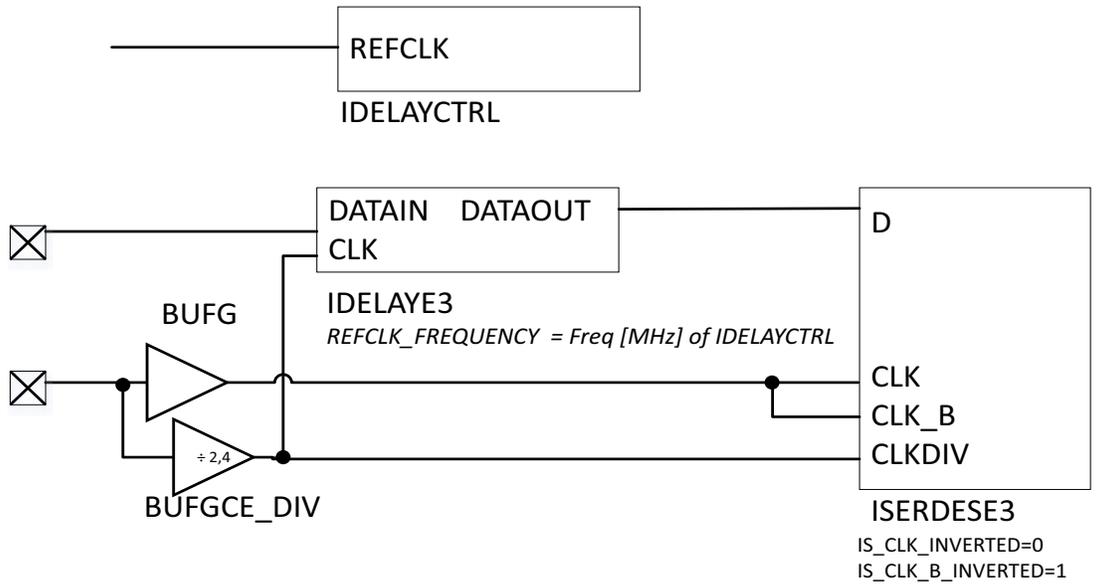


Figure 2-28: MMCM with Separate Clock Outputs



X19513-071823

Figure 2-29: Clocking Connections for ISERDES with IDELAY (TIME Mode)

When using IDELAY in TIME mode, the CLK input (IDELAY) should be connected to the low-speed divided clock (CLKDIV) for the ISERDES as shown in Figure 2-29. The same is true for ODELAY, such that the CLK (ODELAY) should be connected to CLKDIV (OSERDES).

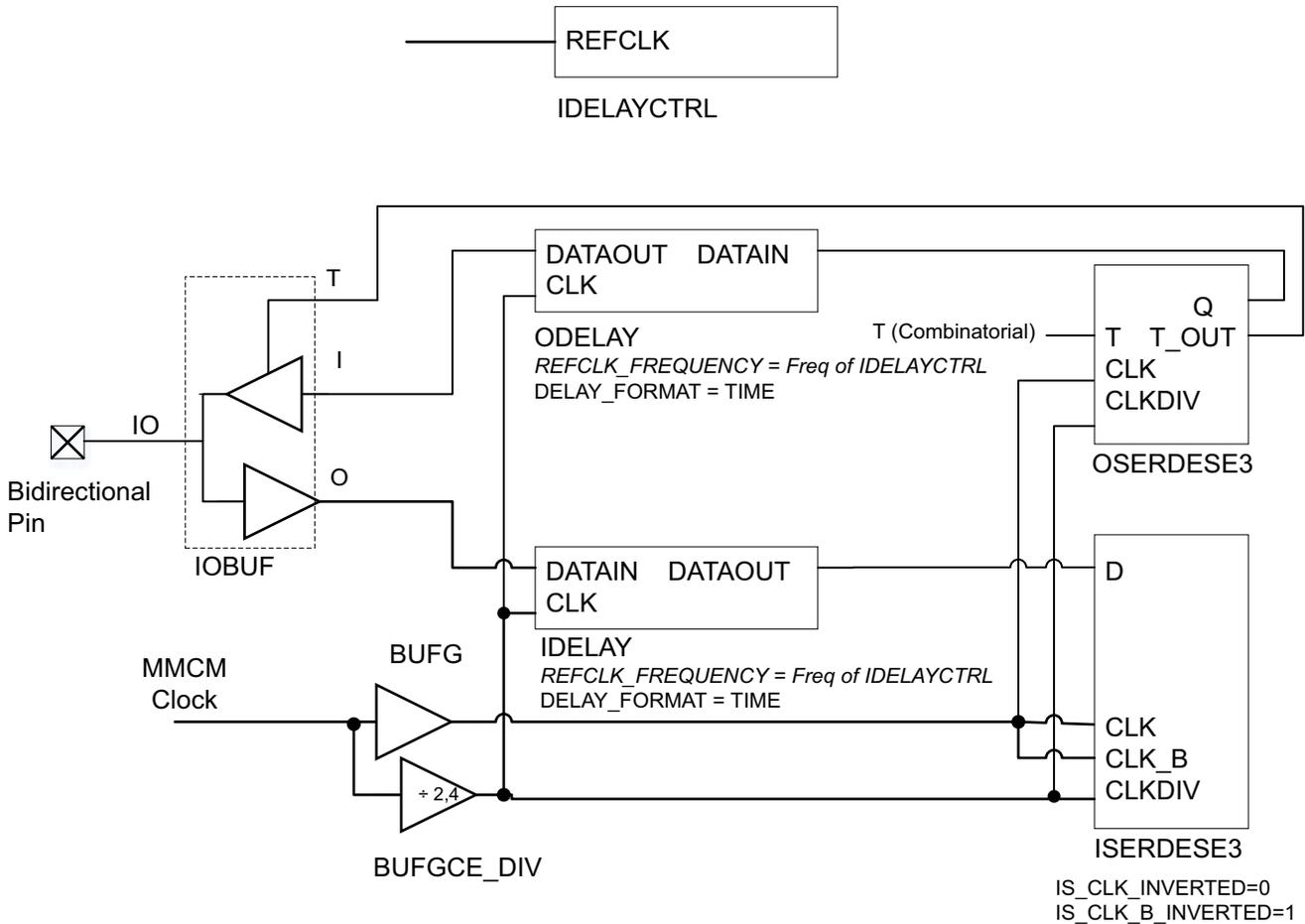
The reference clock for IDELAYCTRL is the reference clock for all IDELAYs and ODELAYs that are used in TIME mode and is typically a different clock source. Because each nibble is controlled by a single IDELAYCTRL, all of the IDELAYs and ODELAYs within that nibble must set the REFCLK\_FREQUENCY to the frequency of the clock connected to REFCLK to ensure the delays.

## Bidirectional Signaling Using Component Mode

All 52 pins in a bank are capable of bidirectional operation using the same component primitives.

For bidirectional signaling with 3-state support, for the output and 3-state path use the solutions as discussed in ODDRE1, page 153. Note the 3-state path driving the T input for the IOBUF does not support simple registered outputs such as FDCE/FDPE/FDRE/FDSE. Registers for FDCE/FDPE/FDRE/FDSE for the 3-state path are implemented in internal logic.

For designs using an OSERDES, Figure 2-30 shows an example bidirectional pin with IDELAY and ODELAY. The OSERDES only supports combinatorial 3-state controls for the T input of the IOBUF.



X19512-071823

Figure 2-30: Bidirectional Signaling Using IDELAY/ODELAY with ISERDES and OSERDES

**Note:** When using bidirectional interfaces, the delay cascades are not available. For all other solutions, Native Primitives must be used.

Bidirectional support varies by I/O bank. Table 2-19 lists the recommended bidirectional logic. When unsupported, the Vivado tool can borrow resources from additional I/Os.

Table 2-19: Bidirectional Support by I/O Bank

Input Datapath	Output Datapath	Tristate Control	I/O Banks
Fabric Logic	Fabric Logic	Fabric Logic	HRIO/HDIO/HPIO
IFD	OFD	Fabric Logic	HRIO/HDIO/HPIO
IFD	OFD	OFD	HDIO
IDDRE1	ODDRE1	Fabric Logic	HRIO/HPIO

Table 2-19: Bidirectional Support by I/O Bank (Cont'd)

Input Datapath	Output Datapath	Tristate Control	I/O Banks
IDDE1	ODDRE1	ODDRE1	HRIO/HPIO
ISERDESE3	OSERDESE3	Fabric Logic	HRIO/HPIO
ISERDESE3	OSERDESE3	OSERDESE3	Not supported

## Mixing Native and Non-Native Mode I/O in a Nibble

As described in [RXTX\\_BITSLICE, page 198](#), a BITSlice\_CONTROL is connected to one or more bit slices (RX\_BITSLICE/TX\_BITSLICE or RXTX\_BITSLICE) in a nibble, with the position of the bit slice I/O determined by the dedicated control bus connections.

If there are unused I/O bit slices in a native mode nibble, other I/O can be positioned (mixed) in the free locations. No special connectivity is necessary in the design because the I/O buffers are connected in the usual manner. All SelectIO component primitives (IFD/OFD, IDDR/ODDR, IDELAY/ODELAY, ISERDESE3/OSERDESE3) can also be used when mixing in a native mode nibble.

For designs with multiple nibbles being used within a bank, care must be taken when resetting the IDELAYCTRLs and BITSlice\_CONTROLS. The reset for all used BITSlice\_CONTROLS and IDELAYCTRLs within a bank must be released at the same time due to the DLY\_RDY connections between the BITSlice\_CONTROLS within a bank. For example, if a bank has two different interfaces, both interfaces should be controlled by a single reset to ensure that calibration completes. Failure to do so might result in DLY\_RDY for one of the interfaces not asserting. To place IDELAYs/ODELAYs (component) into a nibble that already contains TX\_BITSLICE (native), use the IODELAY\_GROUP constraint and placement constraints for the delay elements. An IDELAYCTRL element must not be associated with the mixed Component mode IDELAYE3/ODELAYE3 instances because the native mode BITSlice\_CONTROL is already configured to carry out the delay calibration in the nibble. To implement mixed delays using the Vivado Design Suite, place the IODELAY\_GROUP constraint on both the BITSlice\_CONTROL instance as well as on each primitive IDELAYE3/ODELAYE3 instance to be mixed in that nibble. The syntax is as follows:

```
set_property IODELAY_GROUP MIXED_DELAY_GROUP_NAME [get_cells BITSlice_CONTROL_INST]
set_property IODELAY_GROUP MIXED_DELAY_GROUP_NAME [get_cells COMPONENT_MODE_DELAY_INST]
```

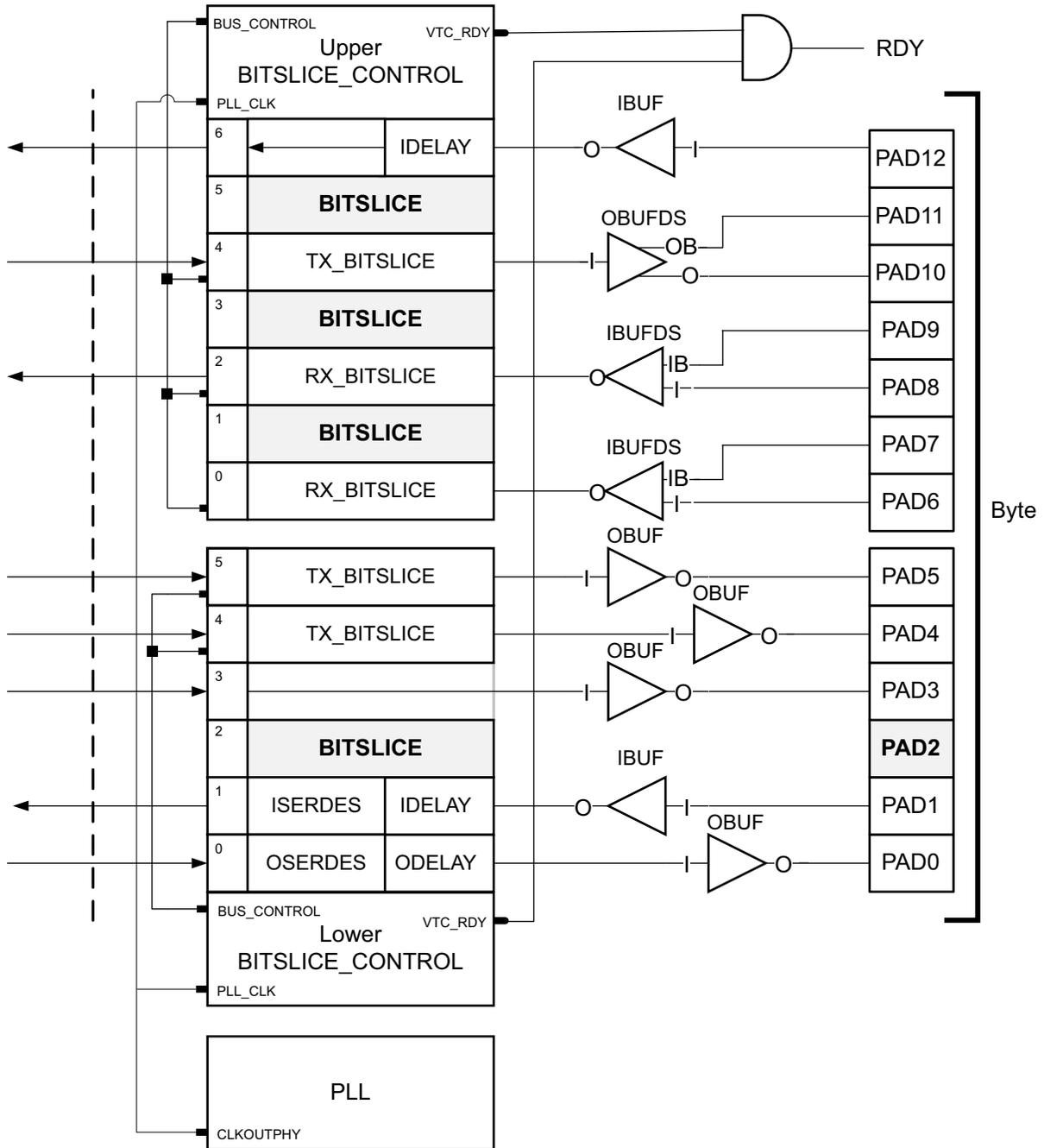
Each nibble requires an IODELAY\_GROUP, which corresponds to a BITSlice\_CONTROL or IDELAYCTRL within that nibble. Because each nibble only contains a single IDELAYCTRL or BITSlice\_CONTROL, each nibble can only contain IDELAYs/ODELAYs from a single IODELAY\_GROUP.

The frequency of REFCLK connected to BITSlice\_CONTROL should be specified as the REFCLK\_FREQUENCY attribute of the IDELAYE3/ODELAYE3 primitive instances. All of the IDELAYE3/ODELAYE3 primitives within the nibble must match the frequency for REFCLK to ensure the delays set by the DELAY\_VALUE are calibrated correctly. The VTC\_RDY signal

from the BITSlice\_CONTROL indicates that calibration is completed for all native and non-native delays in the mixed nibble.

When not mixing native and Component mode delays, it is not necessary to specify the IODELAY\_GROUP constraint for BITSlice\_CONTROLS.

An example mixed-mode byte is illustrated in Figure 2-31.



X16055-022216

Figure 2-31: Example Mixed-Mode Byte

Example XDC constraints for mixed nibbles depicted in [Figure 2-31](#) are listed:

```

set_property IODELAY_GROUP UPPER_GROUP [get_cells UPPER_BITSLICE_CONTROL_INST]
set_property IODELAY_GROUP UPPER_GROUP [get_cells UPPER_RXBIT_0/IDELAYE3]
set_property IODELAY_GROUP LOWER_GROUP [get_cells LOWER_BITSLICE_CONTROL_INST]
set_property IODELAY_GROUP LOWER_GROUP [get_cells LOWER_RXBIT_0/IDELAYE3]
set_property IODELAY_GROUP LOWER_GROUP [get_cells LOWER_TXBIT_0/ODELAYE3]
set_property PACKAGE_PIN PAD12 [get_ports UPPER_RXBIT_0]
set_property PACKAGE_PIN PAD11 [get_ports UPPER_TXOUT_0_N]
set_property PACKAGE_PIN PAD10 [get_ports UPPER_TXOUT_0_P]
...
set_property PACKAGE_PIN PAD3 [get_ports LED_OUT]
set_property PACKAGE_PIN PAD2 [get_ports LOWER_RX_BIT_0]
set_property PACKAGE_PIN PAD1 [get_ports LOWER_TX_BIT_0]

```

Notes on the example in [Figure 2-31](#):

- Upper nibble
  - There is one incoming differential strobe/clock located at the lower bit slice position 0, using pads 6 and 7. This is captured using a native primitive RX\_BITSLICE (DATA\_AND\_CLOCK).
  - There are two further native mode primitive DATA bit slices, at bit slice positions 2 (RX\_BITSLICE) and 4 (TX\_BITSLICE), each using differential I/O.
  - There is one mixed Component primitive IDELAYE3 driving straight to the internal logic in the upper bit slice position 6, at pad 12 using a single-ended IBUF.
  - The XDC constraint defines IODELAY\_GROUP called UPPER\_GROUP to group the Component primitive IDELAYE3 with the upper BITSLICE\_CONTROL instance.
  - All 7 of the I/O pads are used in the upper nibble.
- Lower nibble
  - Two native primitive TX\_BITSLICES are located in the upper two bit slice positions 4 and 5, driving single-ended OBUFs at pads 4 and 5.
  - Two mixed Component primitive delays are located in the lower nibble, one IDELAYE3 driving an ISERDESE3 at position 1 and one OSERDESE3 driving an ODELAYE3 at position 0.
  - There is one other non-native I/O located in the lower nibble. Signal LED\_OUT directly drives an I/O without any I/O logic elements used. This is achieved by LOCing the I/O into the correct package pin.
  - The XDC constraint defines the IODELAY\_GROUP called the LOWER\_GROUP to group the Component primitive ODELAYE3 and IDELAYE3 with the lower BITSLICE\_CONTROL instance.
  - Five of the six I/O pads in the lower nibble are used. Another I/O could potentially be placed in the unused pad 2 by applying the appropriate PACKAGE\_PIN property as was done in the XDC example for LED\_OUT, assuming the proposed I/O meets the SelectIO bank combination rules.

- A PLL is used to supply the master clock for both upper and lower BITSlice\_CONTROLS using the PLL\_CLK dedicated path. Because this clock is used as the BISC reference clock for the mixed Component mode, IDELAYE3/ODELAYE3 primitives (as well as any native mode delays), the frequency of this clock should be set for each Component primitive delay instance, on the REFCLK\_FREQUENCY attribute. No Component primitive IDELAYCTRL elements should be associated with the Component primitive delay instances in this byte.
- The VTC\_RDY signal from the two BITSlice\_CONTROL signals signifies that BISC is complete for the two nibbles, the same function that the Component primitive IDELAYCTRL RDY signal does for non-mixed Component primitive nibbles.

---

## Native Primitives

The Native primitives are the fundamental structures from which Component primitives are created. The Component primitives use specific settings of the Native primitives to provide the same functionality from previous FPGA families. With native primitives you can construct component interfaces that run at high speeds and are much more complex than those constructed with Component primitives. To ease the interface logic generation process using native primitives, AMD developed a High-Speed SelectIO wizard (HSSIO-Wiz).

### RXTX\_BITSLICE

This basic primitive can be used as receiver, transmitter, or bidirectional circuit. This primitive is the base from which the RX\_BITSLICE and TX\_BITSLICE are generated.

The RXTX\_BITSLICE contains both an input and output path. Included in the input and output paths are input and output delays that can be continuously corrected for VT variation by BITSlice\_CONTROL, serialization logic for either 4:1 or 8:1 on the output path, and deserialization logic for 1:4 or 1:8 on the input path. The input path also includes a shallow FIFO to allow connection of received data to another clock domain in the general interconnect logic. A block diagram of RXTX\_BITSLICE is shown in [Figure 2-32](#).

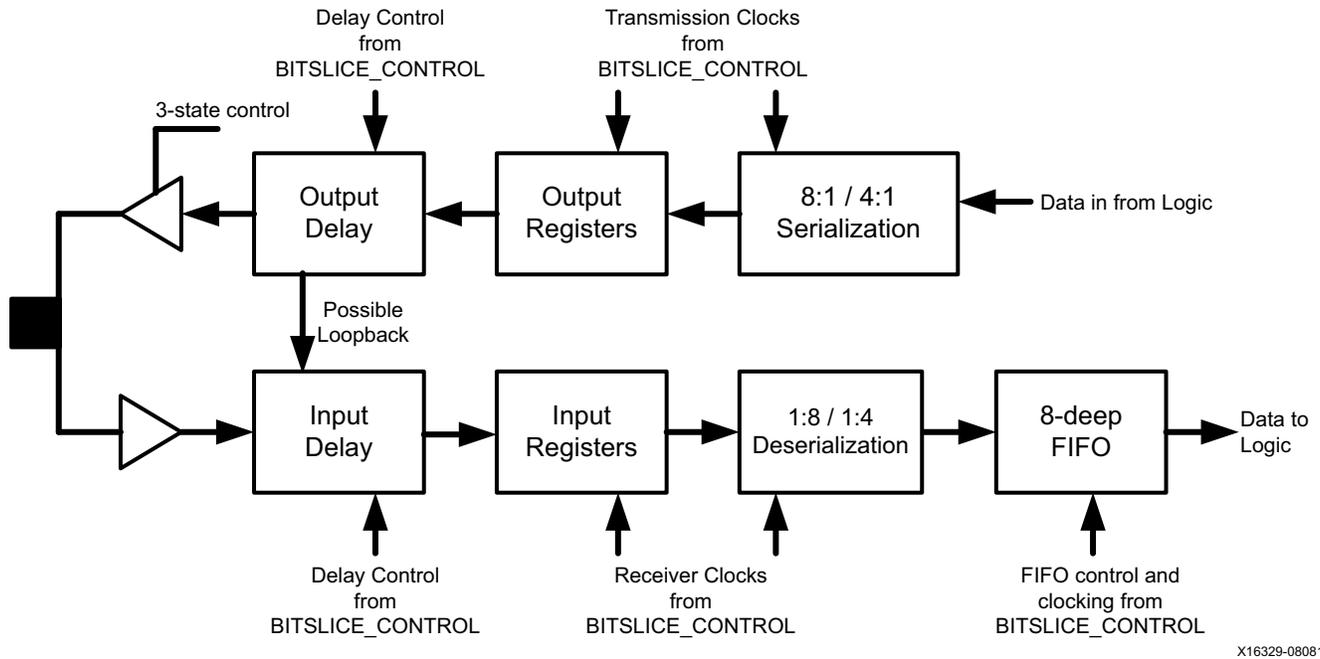


Figure 2-32: RXTX\_BITSLICE Block Diagram

### Input and Output Delay Lines

The input and output delays are each 512 taps deep (the delay of one tap is provided in the UltraScale device data sheets as  $T_{\text{ODELAY\_RESOLUTION}}$  [Ref 2]. The delay element can be controlled either from the BITSlice\_CONTROL via the RIU interface or directly from interconnect logic using the delay control signals on the RXTX\_BITSLICE (CLK, CE, INC, LOAD, CNTVALUEIN[8:0], CNTVALUEOU[8:0], RST\_DLY, and EN\_VTC).

The delay lines can be used in two distinct modes, TIME and COUNT. In TIME mode, the initial delay (DELAY\_VALUE) is defined in ps; in COUNT mode the initial delay is provided as a number of taps. When TIME mode is used, the built-in self-calibration (BISC) controller calibrates and maintains the delay line.

### Delay Line Cascading

A feature not available in the RXTX\_BITSLICE but available in the RX\_BITSLICE is called *cascade*. This feature allows unused output delay lines in TX\_BITSLICE cascaded to input delay lines in RX\_BITSLICES. The result is a delay line of double length passing data to the RX\_BITSLICE deserializer registers. A single delay line is 512 taps. Cascading both input and output delay lines can double the length of the available delay. This feature is discussed in detail in the RX\_BITSLICE [Extended Delay Control Signals](#), page 261.

### 3-State Control

The transmitter side of a RXTX\_BITSLICE and thereby a TX\_BITSLICE provides two possibilities to 3-state an output buffer in an IOB. The two ways to 3-state can be seen as per channel block 3-state and as a nibble based per bit 3-state in a serial stream 3-state.

Each RXTX\_BITSLICE and thus each TX\_BITSLICE has a T input. The input is a combinatorial feed through of the 3-state signal generated in the FPGA logic to the T input of an output buffer in the IOB. This is called *block 3-state* because the serial output at the output buffer is 3-stated for an amount of bit periods. When 3-state of serial outputs must occur on a specified bit or bits in the serial stream, a combination of the BITSlice\_CONTROL.TBYTE\_IN[3:0] inputs with a TX\_BITSLICE\_TRI must be used. The output of the TX\_BITSLICE\_TRI is routed to and through the TX\_BITSLICES to the 3-state input of an output buffer.

The output of the TX\_BITSLICE\_TRI is a serial stream that can be connected to all TX\_BITSLICES in a nibble and in this way, to all 3-state output buffer inputs. Four bits written at the TBYTE\_IN inputs of the BITSlice\_CONTROL determine the 3-state occurrence in a serial stream. See the 3-state explanation in [TX\\_BITSLICE\\_TRI, page 273](#).

### FIFO

The receiver of each RXTX\_BITSLICE and thus RX\_BITSLICE has an 8-deep shallow FIFO.

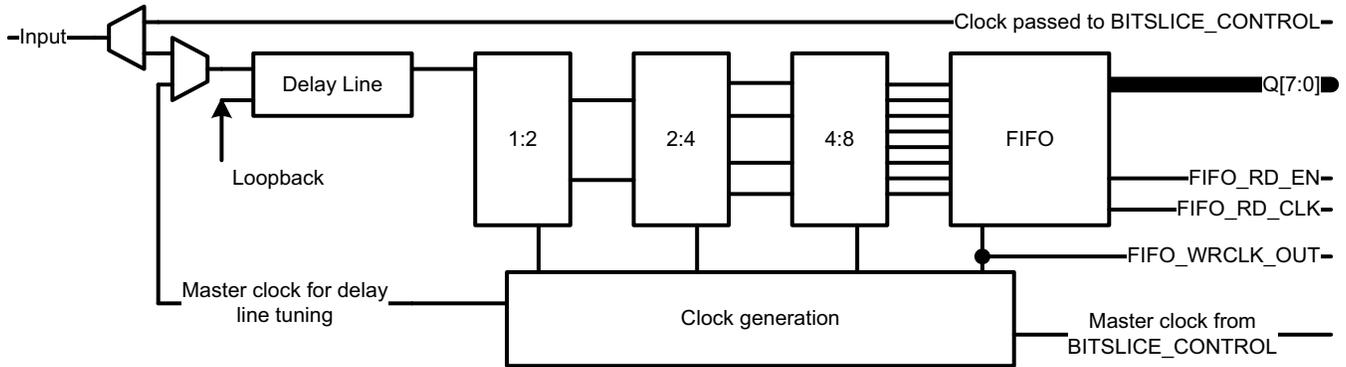
The deserialized 4-bit or 8-bit data is written using the FIFO\_WR\_CLK domain in the bit slice-generated clock (FIFO\_WR\_CLK) in the FIFO.

The FIFO writes the 4-bit or 8-bit deserialized data on the rising edge of FIFO\_WR\_CLK. The FIFO can be read after interpretation of some FIFO status signals from the FPGA logic side. In this way, the FIFO performs the role of clock domain crossing element. See more about the FIFO in [FIFO Function, page 210](#).

### RXTX\_BITSLICE Receiver Function

#### Receiver Setup

As shown in [Figure 2-33](#), the setup of a receiver can be divided into the following blocks; an input delay element feeding a set of deserializer registers writing parallelized data in a FIFO and clock generation logic.



X16330-072216

Figure 2-33: Receiver Block Diagram

The input delay line is always in the signal input path. When an input delay is not wanted, the delay value of the delay should be set to zero. The deserializer registers are divided into three stages, 1:2, 2:4, and 4:8. From here the FIFO input is written. All necessary clocks for the register stages and FIFO write side are generated in the clock generation logic and fed by BITSlice\_CONTROL outputs. The BISC controller in the BITSlice\_CONTROL uses clocks for tuning and aligning the clock to the data. This is discussed in detail in [Clocking in Native Mode](#) in the [BITSlice\\_CONTROL](#) section.

Assume capturing data with the data forwarded clock:

- The forwarded clock must be connected to a BITSlice\_0 of a nibble. These are the QBC or DBC balled inputs.
- The forwarded clock is passed through the BITSlice\_0 into the BITSlice\_CONTROL.
- A clock generator in the BITSlice\_CONTROL creates the necessary clocks to capture and write the data bits into the FIFO.
- The BITSlice\_CONTROL also needs a master or reference clock applied to its PLL\_CLK input.

Typically, for low jitter and high performance, this clock is generated by one of the two PLLs in the area behind the I/O bank. The CLKOUTPHY output of the PLL must connect to the BITSlice\_CONTROL.PLL\_CLK input without a clock buffer. For source synchronous systems, the frequency of that clock is equal to the bit rate of the captured data.

- Data is captured by a deserializer using the received forwarded clock and writes the deserialized data into the FIFO.
- The clock used to write data into the FIFO is made available to the FPGA logic as FIFO\_WRCLK\_OUT. Although each RXTX\_BITSlice has a FIFO\_WRCLK\_OUT pin, the signal is only available at BITSlice\_0 of a nibble.

- The FIFO\_WRCLK\_OUT can be used instead of a PLL or MMCM as a clock for logic designed in the FPGA and as a read clock for the FIFO. To do this, the FIFO\_WRCLK\_OUT is passed through a BUFG clock buffer.

Data is written into the FIFO at each rising edge of the internal FIFO write clock, reflected as FIFO\_WRCLK\_OUT to the FPGA logic. The FIFO write pointer runs from 0 to 7 and then loops around, filling the FIFO with new data.

Here are conditions to read the captured and deserialized data from the FIFO:

- To read data from the FIFO, it needs a read clock (FIFO\_RD\_CLK). This clock must have the same frequency as the FIFO\_WRCLK\_OUT clock, and phase unknown (mesochronous). If needed or wanted, the FIFO\_WRCLK\_OUT can thus be used as FIFO\_RD\_CLK.
- A second condition to read data from the FIFO is that the FIFO\_RD\_EN input is High.
  - Note:** When FIFO\_RD\_EN is kept Low, the read pointer is stopped. Assuming the write clock continues, the write pointer continues to increment, resulting in the FIFO\_EMPTY—typically matching every eight FIFO\_WRCLK\_OUT clock cycles. The eight-cycle behavior is typically observed but is not guaranteed. FIFO\_RD\_EN circuitry should use the first deassertion of FIFO\_EMPTY.
- When the FIFO write and read pointer are equal, meaning that write and read access the same position in the FIFO, a FIFO\_EMPTY pulse is generated. This pulse is synchronized with the FIFO\_RD\_CLK, and it takes two FIFO\_RD\_CLK cycles before the status is presented at the FIFO\_EMPTY pin of the RXTX\_BITSLICE.

An example of legal operation of the FIFO in the RXTX\_BITSLICE is as follows:

1. Apply a FIFO\_RD\_CLK to the FIFO.
2. Use the registered version of inverted FIFO\_EMPTY signal (operated by the FIFO\_RD\_CLK).
3. Use this output to enable the FIFO by way of the FIFO\_RD\_EN input.

The following approach ensures that after the first data is written into the FIFO, it is read and FIFO read pointers never cause the FIFO to generate an empty status signal:

1. At the start, the write and read pointers are zero.
2. The FIFO\_EMPTY status signal is High, showing that the FIFO is empty.
3. The read pointer is stuck because the FIFO is disabled.
4. After the first data write, a non-empty situation is generated.
5. That status is only available to the application after two read clock cycles. This means that write runs two clock cycles ahead of read.

6. FIFO\_EMPTY is used through a register, operated on the FIFO\_RD\_CLK, to enable the FIFO requiring an extra FIFO\_RD\_CLK cycle. Thus write runs three clock cycles ahead of read.
7. FIFO\_EMPTY does not show empty as long as data is written and read from the FIFO.

Two use cases apply:

- Do not touch the FIFO\_RD\_EN. It is only controlled by the FIFO\_EMPTY signal.
- When data from the FIFO needs to be ignored by the application, disable a capture register in the application.

To re-enable the FIFO, wait for FIFO\_EMPTY to pulse (Low-High-Low) before applying FIFO\_RD\_EN. This follows the sequence of the numbered list starting at [step 2](#).

### Calculation of Required Frequencies

#### Example 1

- Source synchronous DDR interface (data+clock) at 1250 Mb/s.
  - RX\_DATA\_TYPE = DATA\_AND\_CLOCK for the bit slice receiving the forwarded clock
- With 1250 Mb/s comes a forwarded clock of 625 MHz.
  - This clock is used to capture the data bits. Continuous clocks can optionally be used as PLL clock inputs.
- BITSlice\_CONTROL needs a PLL\_CLK clock equal to the data rate of the received signals.
  - The PLL must deliver a 1250 MHz clock to the BITSlice\_CONTROL.
- The receiver used in 8-bits requires a FIFO\_RD\_CLK shown in [Equation 2-5](#):

$$\text{forwarded clock} / 4 = 156.25 \text{ MHz} \qquad \text{Equation 2-5}$$

- The receiver used in 4-bits requires a FIFO\_RD\_CLK shown in [Equation 2-6](#):

$$\text{forwarded clock} / 2 = 312.5 \text{ MHz} \qquad \text{Equation 2-6}$$

#### Example 2

- Asynchronous interface (data only) at 1250 Mb/s.
  - RX\_DATA\_TYPE = SERIAL
- Clock needs to be delivered by PLL or MMCM.
- To sample 1250 Mb/s data, a DDR clock of 625 MHz is required.
  - The PLL/MMCM must deliver a 625 MHz clock to the BITSlice\_CONTROL.
- The receiver used in 8-bits requires a FIFO\_RD\_CLK shown in [Equation 2-7](#):

$$\text{DDR (PLL/MMCM) clock} / 4 = 156.25 \text{ MHz} \qquad \text{Equation 2-7}$$

- The receiver used in 4-bits requires a FIFO\_RD\_CLK shown in [Equation 2-8](#):

$$\text{DDR (PLL/MMCM) clock} / 2 = 312.5 \text{ MHz} \quad \text{Equation 2-8}$$

### **Native Input Delay Type Usage**

#### **FIXED Mode**

The DELAY\_TYPE attribute set to FIXED selects the fixed delay through the input delay line and is determined by the DELAY\_VALUE and DELAY\_FORMAT attribute. When the DELAY\_FORMAT is set to TIME, the value loaded in the delay line is in ps. When the DELAY\_FORMAT is set to COUNT, the delay value loaded in the delay line is the number of taps. During the reset sequence, RXTX\_EN\_VTC is kept High by the High-Speed SelectIO wizard. After the reset sequence has finished, the RXTX\_EN\_VTC should be controlled based on the DELAY\_FORMAT setting.

- When DELAY\_FORMAT is TIME, then RXTX\_BITSLICE.EN\_VTC must be pulled High so that the delay automatically changes the number of taps over voltage and temperature to ensure the delay stays at the requested time in ps.
- With DELAY\_FORMAT set to COUNT, RXTX\_BITSLICE.EN\_VTC must be Low. In COUNT mode, the delay is not compensated for voltage and temperature.

#### **VARIABLE Mode**

The DELAY\_TYPE attribute set to VARIABLE selects the variable tap delay line ([Table 2-13](#)). In VARIABLE mode, the CE and INC pins 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. The increment/decrement operation depends on the UPDATE\_MODE attribute (see [Figure 2-34](#)). The RXTX\_BITSLICE.EN\_VTC pin should be held Low during the delay change command to ensure that any automatic adjustments are stopped.

To increment/decrement delay lines when using TIME mode:

- Deassert (Low) the RXTX\_BITSLICE.EN\_VTC pin.
- Wait a minimum of 10 clock cycles.
- Use the CE and INC ports to increment or decrement the delay line.
- Wait a minimum of 10 clock cycles.
- Assert the RXTX\_BITSLICE.EN\_VTC pin.

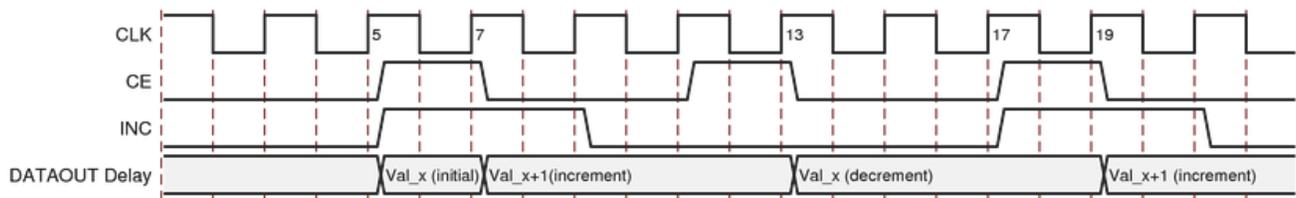
In COUNT mode, the RXTX\_BITSLICE.EN\_VTC port is always Low after the reset sequence has completed. Use the preceding TIME mode procedure, [step 2](#) through [step 4](#).

Table 2-20: RXTX\_BITSLICE Control Pin when DELAY\_TYPE = VARIABLE

EN_VTC	CLK	LOAD	CE	INC	Tap Setting
1	1/0	X	X	X	Not supported, EN_VTC must be Low when LOAD, CE, and INC are active.
0	0	X	X	X	No change
0	1	0	0	X	No change
0	1	0	1	1	Current value +1 tap <sup>(1)</sup>
0	1	0	1	0	Current value -1 tap <sup>(1)</sup>
0	1	0	0	0	No change

**Notes:**

- Value depends upon the UPDATE\_MODE attribute. See [Figure 2-34](#).



X17477-072516

Figure 2-34: Variable Mode, UPDATE\_MODE = ASYNC

 Notes on [Figure 2-34](#):

- At the rising edge of clock event 7, CE and INC are both High. The current tap is incremented by one because the INC signal is High for two clock cycles while CE only is High for a single cycle. UPDATE\_MODE = ASYNC. The new setting is represented by Val<sub>x</sub>+1.
- At the rising edge of clock event 13, CE is High and INC is Low. The delay line is decremented by one tap. The DATAOUT again has taken the position of the first loaded value.
- At the rising edge of clock event 19, CE and INC are both High. The current tap is incremented by one because the INC signal is High for two clock cycles while CE is High only for a single cycle. UPDATE\_MODE = ASYNC. The new setting is represented by Val<sub>x</sub>+1.

**VAR\_LOAD Mode**

When the DELAY\_TYPE attribute is set to VAR\_LOAD, the delay line can be changed using the CE and INC inputs. Or the CNTVALUEIN, CNTVALUEOUT, and LOAD pins can be used to parallel load the delay line tap selection. The CE and INC inputs change the delay line on a per tap basis while the COUNTVALUEIN/OUT buses allow a dynamic change of the delay line, meaning that it is possible to pass from one delay line tap setting to a completely

different value upon the load of the value presented at the CNTVALUEIN inputs to any tap in the delay line.

The VAR\_LOAD method is suitable for both COUNT and TIME mode usage of the delay line.

In both modes, the tap amount can be read from the CNTVALUEOUT bus, and if necessary, changed through the CNTVALUEIN bus or INC port.

Notes:

- Use the explanation of VARIABLE mode when incrementing or decrementing the delay line using the INC/CE input pins.
- The VAR\_LOAD procedure to calculate the value to update the delay line is different for IDELAY and ODELAY.
- The VAR\_LOAD procedure to update the delay line is different for TIME and COUNT mode. During the reset sequence, RXTX\_EN\_VTC is kept High by the High-Speed SelectIO wizard.

The DELAY\_TYPE is VAR\_LOAD and DELAY\_FORMAT is TIME procedure to update the delay line follows (see [Figure 2-22](#)):

1. After BITSlice\_CONTROL.DLY\_RDY goes High, BITSlice\_CONTROL.EN\_VTC must be pulled High.
2. After BITSlice\_CONTROL.VTC\_RDY goes High, make RXTX\_BITSlice.EN\_VTC Low to modify the delay line.
3. Wait for at least 10 clock cycles.
4. Read CNTVALUEOUT[8:0] and load the value into a register.
5. Check if updating the delay line is necessary.
6. Calculate the new delay value to be written in the delay line.
7. Put the new delay line value on the CNTVALUEIN[8:0] bus.
8. Wait for one clock cycle and pulse LOAD High for a clock cycle.
9. For continuous loads, wait 5 clock cycles before returning to [step 7](#).
10. Wait for at least 10 clock cycles.
11. Pull RXTX\_BITSlice.EN\_VTC back High.
12. Go back to [step 2](#) for a new delay line update.

The DELAY\_TYPE is VAR\_LOAD and DELAY\_FORMAT is COUNT procedure to update the delay line follows:

1. After BITSlice\_CONTROL.DLY\_RDY goes High, RXTX\_BITSlice.EN\_VTC is kept Low.
2. Read CNTVALUEOUT[8:0] and load the value into a register.

3. Check if updating the delay line is necessary.
4. Calculate the new delay value to be written in the delay line.
5. Put the new delay line value on the CNTVALUEIN[8:0] bus.
6. Wait for one clock cycle and pulse LOAD High for a clock cycle.
7. For continuous loads, wait 5 clock cycles before returning to [step 5](#).

To calculate new values to be written to the delay lines, the following must be known:

- A delay line has 512 taps and is at least 1250 ps.
- The delay range of a single tap is specified in the UltraScale device data sheet [\[Ref 2\]](#).
- Delay lines are not calibrated until the `FPGA.bit` file is configured and the per nibble BISC engine has run. As such the real delay of a single tap in an FPGA is unknown.
- In TIME mode:
  - The initial `DELAY_VALUE`, in the design attribute, **must** be provided in ps.
  - Afterward, the initial delay setting can be modified by writing a value represented as a number of taps into the delay line.
- The BISC process uses a number of taps of an input delay line to eliminate the delay difference between the data and the clock paths before arriving at the first data capture flip-flops of the receiver. This delay is called *Align\_Delay*. This process is done to let BISC align the clock to the data.
- Within a nibble, the *Align\_Delay* can be between 45 and 65 taps. It averages 50 to 54 taps. When using inter-nibble or inter-byte clocking, the *Align\_delay* will be greater.
- When writing all zeros or an amount of taps smaller than the reported *Align\_Delay* to an input delay line, the tuned *Align\_Delay* is impacted.
- An output delay line does not have this align delay.
- The BISC process is always running in the background to compensate for voltage and temperature variations.
- In COUNT mode:
  - The initial `DELAY_VALUE`, in the design attribute, **must** be provided in taps.
  - The BISC procedure is not used and the real delay value of a tap cannot be known.
  - There is no voltage and temperature compensation for the delay lines because BISC does not run.
  - The delay line **must** be used as a delay of at least 512 taps.
  - Measurements and adjustments must be calculated in taps. For example:
    - A measurement of a data eye is expressed as 450 taps.

- Jitter between two data eyes is expressed as 31 taps.

When DELAY\_TYPE is VAR\_LOAD and DELAY\_FORMAT is COUNT, a delay line is used in *bare-metal* mode because only the depth or amount of taps of the delay line are important (512 in the case of UltraScale devices).

Thus this is the only parameter a design using COUNT mode needs to consider. The value of a measured data, clock, or strobe eye is expressed as an amount of taps without providing the delay this represents. Thus it is not necessary to calculate the delay of a single tap and all 512 provided taps that are available to the user in a delay line.

When DELAY\_TYPE is VAR\_LOAD and DELAY\_FORMAT is TIME, the Align\_Delay must be measured and the single tap delay must be calculated if a new delay time must be set into a delay line. Two input delay lines must be used to calculate the delay value of a single tap.

- When using single-ended inputs, two inputs are necessary to calculate the single tap delay, because behind each input pad with input buffer (IBUF) there is a RXTX\_BITSLICE or RX\_BITSLICE having an input delay line and serial-to-parallel conversion engine (Figure 2-35).
- When using differential inputs, a single data channel input can be used to calculate the single tap delay. A differential input occupies two pads and thus covers two RXTX\_BITSLICES. When a normal differential input buffer (IBUFDS) is used, only the even RXTX\_BITSLICE of the two is used. When using a differential input buffer with differential output (IBUFDS\_DIFF\_OUT) one can use both RXTX\_BITSLICES covered by the two input pads. This is the solution for measuring a single tap value for a single differential data channel (Figure 2-35).

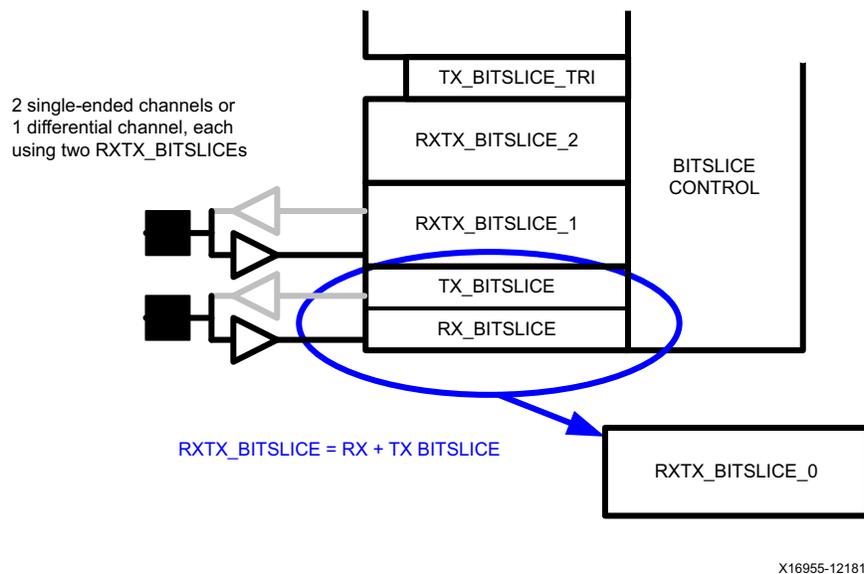


Figure 2-35: Two Single-Ended or One Differential RX Channel

The measure Align\_Delay and calculate a single tap delay using two IDELAYS procedure follows:

1. In the HDL design, for the even bit slice, set the DELAY\_VALUE to zero.
2. In the HDL design, for the odd bit slice, set the DELAY\_VALUE to a larger non-zero value, for example, 700 ps.
3. When the design is downloaded and running in an FPGA, read CNTVALUEOUT of both delay lines and store the amount of taps obtained in a set of registers.
4. The tap value from the even bit slice is the Align\_Delay and that from the odd bit slice is the total delay value (Align\_Delay + Requested value), called *Total\_Value*.
5. The requested delay value, 700 ps in this case, is represented by [Equation 2-9](#):

$$\text{Total\_Value} - \text{Align\_Delay} = n \text{ taps} \quad \text{Equation 2-9}$$

6. The delay of a single tap is then equal to [Equation 2-10](#):

$$\text{odd channel DELAY\_VALUE} / n \text{ taps} = \text{single tap} \quad \text{Equation 2-10}$$

7. The new CNTVALUEIN value to write to the delay line or lines used in taps is shown in [Equation 2-11](#) for RX\_BITSLICES or [Equation 2-11](#) for TX\_BITSLICES:

$$\text{CNTVALUEIN} \langle \text{RX\_BITSLICE} \rangle = (\text{wanted delay} / \text{single tap}) + \text{Align\_Delay} \quad \text{Equation 2-11}$$

$$\text{CNTVALUEIN} \langle \text{TX\_BITSLICE} \rangle = (\text{wanted delay} / \text{single tap}) \quad \text{Equation 2-12}$$

8. Write this new value in the delay line using the delay line update procedure.

Alternatively measure single tap delay and Align\_Delay using an input delay and output delay:

1. In the HDL, set the DELAY\_VALUE for an input delay to some value and record the delay (INPUT\_DELAY\_CNT).
2. In the HDL, set the DELAY\_VALUE for an output delay to the same value and record the delay (OUTPUT\_DELAY\_CNT).
3. Because the ODELAY does not require an Align\_Delay, the delay for each tap can immediately be calculated ([Equation 2-13](#)):

$$\text{DELAY\_VALUE} / \text{OUTPUT\_DELAY\_CNT} = \text{single tap [ps/tap]} \quad \text{Equation 2-13}$$

4. The number of taps required for the Align\_Delay is shown in [Equation 2-14](#):

$$\text{INPUT\_DELAY\_CNT} - \text{OUTPUT\_DELAY\_CNT} = \text{Align\_Delay [# taps]} \quad \text{Equation 2-14}$$

5. CNTVALUEIN can now be set as needed ([Equation 2-15](#) for RX\_BITSLICES or [Equation 2-16](#) for TX\_BITSLICES):

$$\text{CNTVALUEIN} \langle \text{RX\_BITSLICE} \rangle = (\text{wanted delay/single tap}) + \text{Align\_Delay} \quad \text{Equation 2-15}$$

$$\text{CNTVALUEIN} \langle \text{TX\_BITSLICE} \rangle = (\text{wanted delay/single tap}) \quad \text{Equation 2-16}$$



**TIP:** When using an `IBUFDS_DIFF_OUT`, both `RXTX_BITSLICES` or `RX_BITSLICES` can be used to capture data. The even one captures the *p*-side of the differential data channel; the odd one captures the *n*-side. To use the *n*-side data in FPGA logic, invert the data output of the bit slice.

## FIFO Function

The FIFO is controlled by the following signals; `FIFO_RD_CLK`, `FIFO_RD_EN`, `FIFO_EMPTY`, and `FIFO_WRCLK_OUT`. Data from the FIFO is available at the `Q[7:0]` pins. The attributes controlling the FIFO behavior are `RX_DATA_WIDTH` and `FIFO_SYNC_MODE`. All of these are discussed in [Table 2-22](#) and [Table 2-23](#).

`RX_DATA_WIDTH` is an attribute acting for the entire receiver. When set to 4, the FIFO passes data out at pins `Q[3:0]` and a signal mimicking the serial input of the receiver is available at the `Q5` pin. If `RX_DATA_WIDTH` is 8, the FIFO passes 8-bit data and the mimicked serial data output is not available. This `Q5` serial data is only supported for `RX_BITSLICE` and `RXTX_BITSLICE`. `ISERDES` is not supported.

Set `FIFO_SYNC_MODE` to `FALSE` to use the FIFO as a clock domain crossing element. This mode allows data captured from the interface clock domain to successfully cross over to the interconnect logic clock domain(s). The typical latency through the FIFO is equal to two clock cycles (three when counting the extra FIFO enable cycles), but depending on the design, the latency can also be eight read clock cycles. [Table 2-21](#) describes the behavior of the clock and control inputs of the FIFO.

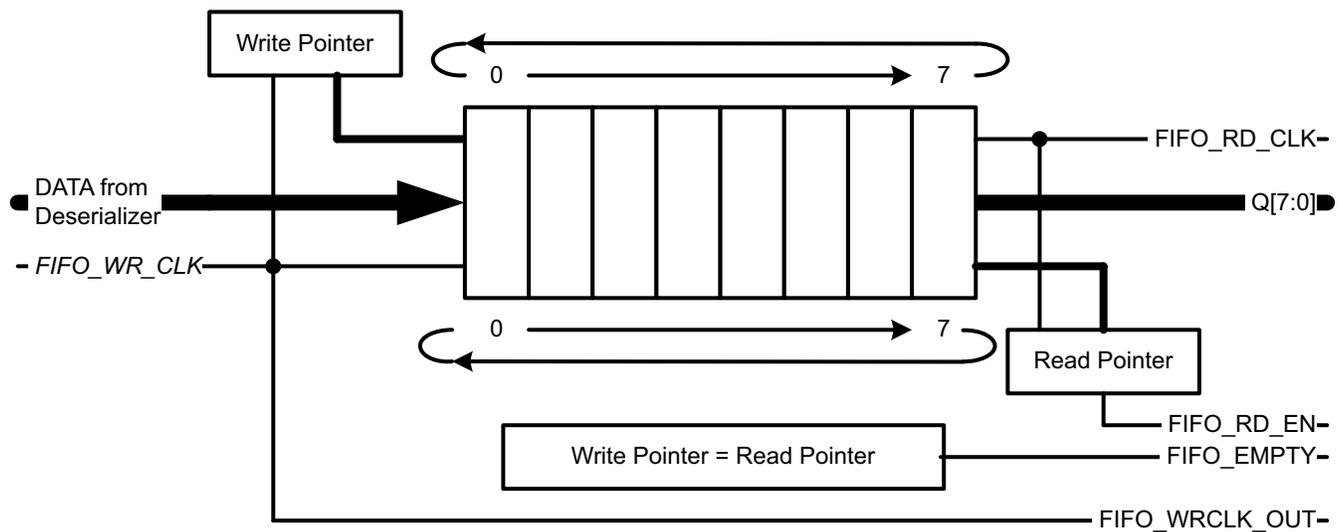
**Table 2-21: Behavior of the Clock and Control Inputs of the FIFO**

Clock or Control Input	Behavior
<code>FIFO_WRCLK_OUT</code>	This clock can be used as a read clock for the FIFO. Clocks to capture the data are generated inside the <code>BITSLICE_CONTROL</code> and <code>RXTX_BITSLICE</code> primitives. One of these internally generated clocks, the divided sample clock, is the clock used to write data into the FIFO (call it <code>FIFO_WR_CLK</code> ). A copy of this clock is provided as <code>FIFO_WRCLK_OUT</code> output of the bit slice. Each bit slice has a <code>FIFO_WRCLK_OUT</code> output pin but only bit slices in nibble position zero can route and use this clock.
<code>FIFO_RD_CLK</code>	This is the clock used to pull data out of the FIFO. It needs to have the same frequency as the <code>FIFO_WR_CLK</code> , but there does not need to be a phase relationship between <code>FIFO_RD_CLK</code> and <code>FIFO_WRCLK_OUT</code> clocks. The FIFO read clock can be supplied by a <code>FIFO_WRCLK_OUT</code> of a <code>BITSLICE_0</code> in the same nibble, byte, or I/O bank, or it can be supplied by a PLL or MMCM generated clock. This clock routes over normal clock nets in the FPGA and requires a clock buffer ( <code>BUFG</code> , <code>BUFGCE</code> , or other).
<code>FIFO_RD_EN</code>	This pin must be High to read the FIFO. When this input pin is left Low, the FIFO output shows new data for every eight <code>FIFO_RD_CLK</code> cycles. This is because the <code>FIFO_RD_EN</code> locks the read pointer of the FIFO while the write pointer advances with each write operation inside the receiver. When the write pointer reaches the eighth pointer position, it loops back to position zero and continues. Because the read pointer is locked, new data appears at the output pins of the bit slice. An empty situation is detected and a <code>FIFO_EMPTY</code> status is generated.

Table 2-21: Behavior of the Clock and Control Inputs of the FIFO (Cont'd)

Clock or Control Input	Behavior
FIFO_EMPTY	This output is High when the FIFO is empty. When data is written into the FIFO and the write and read point access the same position in the FIFO, an empty situation is detected and signaled by a FIFO_EMPTY pin. The FIFO empty situation is synchronized on the FIFO_RD_CLK and therefore it takes two FIFO_RD_CLK cycles before the FIFO_EMPTY pin changes state. This mechanism makes sure that in normal operation the write pointer always runs ahead of the read pointer.

The behavior described in Table 2-21 is shown in Figure 2-36.



X16332-030916

Figure 2-36: Schematic View of a Receiver FIFO

The FIFO can be used to align data at the Q pins of different receivers. When the FIFO\_WRCLK\_OUT is used as FIFO\_RD\_CLK in an I/O bank, the approach is:

- Use the inverted FIFO\_EMPTY signal of the used bit slice *farthest* away from the bit slice receiving the clock and thus generating the FIFO\_WRCLK\_OUT through an optional flip-flop to all FIFO\_RD\_EN inputs of used bit slices. The optional flip-flop can be used to help meet timing.

**Note:** Farthest means the bit slice at the end of the clock backbone. As shown in Figure 2-37, the clock arrives in the lower nibble of byte\_2 and is passed through the inter-byte and inter-nibble backbones to the upper nibble of byte\_0.

- For high clock rates, timing can be challenging. Solve this by adding pipelining.

When multiple clocks/strobes present, the approach is:

- Use a NOR gate combining the FIFO\_EMPTY signals of all used bit slices through an optional flip-flop as input for the FIFO\_RD\_EN of all used bit slices. The optional flip-flop can be used to help meet timing.
- The NOR gate waits for the last FIFO\_EMPTY to transition Low before triggering the FIFO\_RD\_EN through the optional flip-flop (see [Figure 2-37](#)).

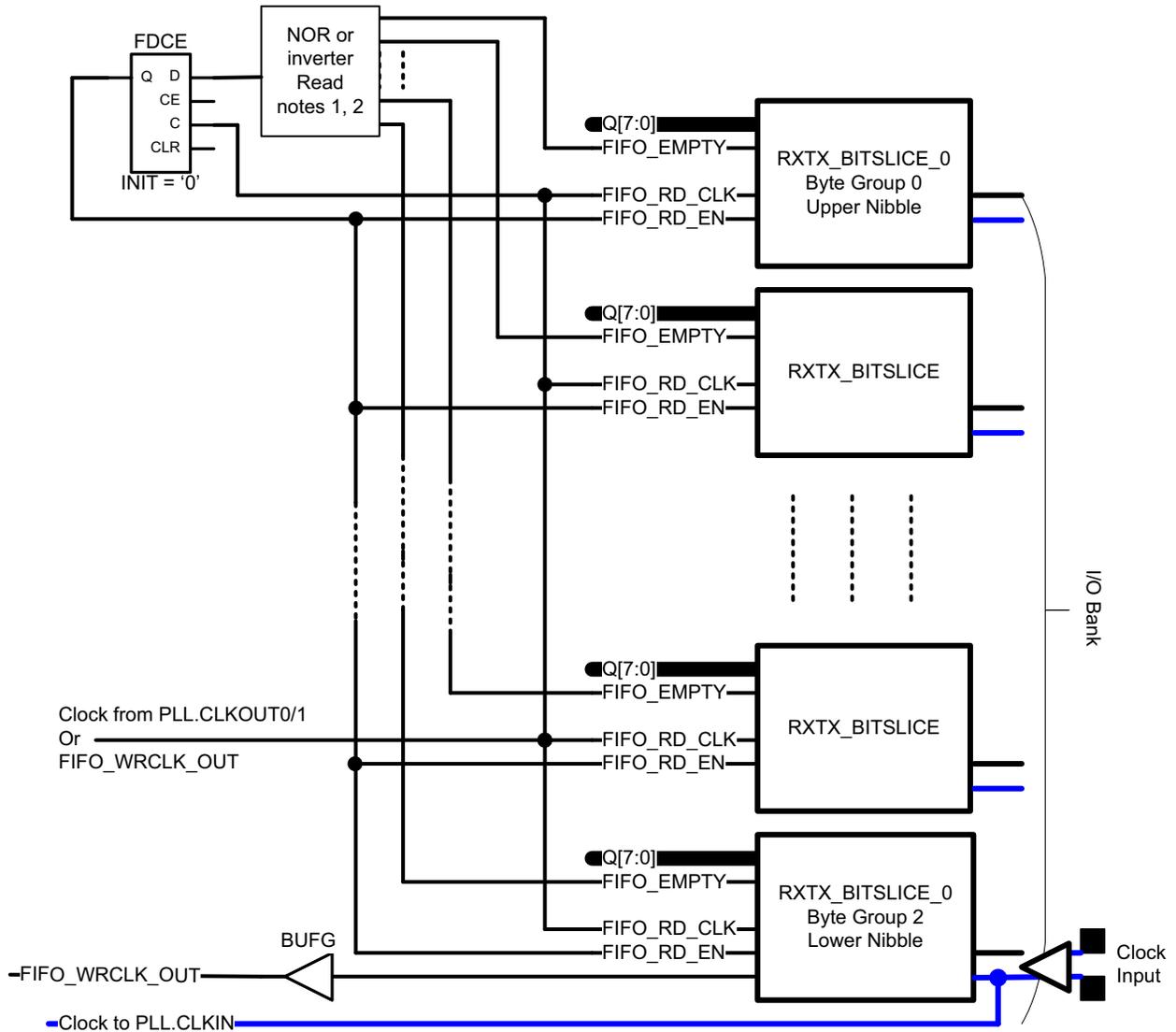
Notes:

- For static timing purposes, a generated clock should be specified as part of the timing constraint when using FIFO\_WRCLK\_OUT. As an example, assume a DATA\_WIDTH = 4 (RX\_BITSLICE) and a sample clock of 500 MHz on a port named rx\_clk\_in. Additionally, assume the instance rx\_clock\_bitslice\_inst (at nibble position zero) with attribute DATA\_TYPE = DATA\_AND\_CLOCK (RX\_BITSLICE), and SERIAL\_MODE = FALSE (BITSLICE\_CONTROL).

The following example XDC generates the required generated clock for the FIFO\_WRCLK\_OUT pin:

```
create_clock -name rx_clk -period 2.000 -waveform {0.000 1.000} [get_ports rx_clk_in]
create_generated_clock -divide_by 2 -source [get_ports rx_clk_in] -name fifo_wrclk
rx_clock_bitslice_inst/FIFO_WRCLK_OUT
```

- It is good practice to enable application logic in the FPGA after BITSlice\_CONTROL.VTC\_RDY goes High. The VTC\_RDY signal provides the status that the I/O interface is initialized and up and running.
- The FIFO\_WRCLK\_OUT clock requires the use of a BUFG clock buffer. Although it is possible to connect the FIFO\_WRCLK\_OUT clock directly to the FIFO\_RD\_CLK in HDL, because Vivado tools automatically insert a BUFG clock buffer.
- To ensure that all RX bit slices start aligned, the rx\_clk\_in (FIFO\_RD\_CLK) [(rx\_clk\_in in the above example)] should be stopped until the RX VTC\_RDY signal is asserted.
- If you have control of the TX while the RX is coming out of reset, the CLK to the RX side should be stopped until the RX VTC\_RDY signal is asserted.
- If you do not have control of the TX side, then a bitslip module needs to be implemented in the RX side to ensure all channels are aligned if alignment is needed.



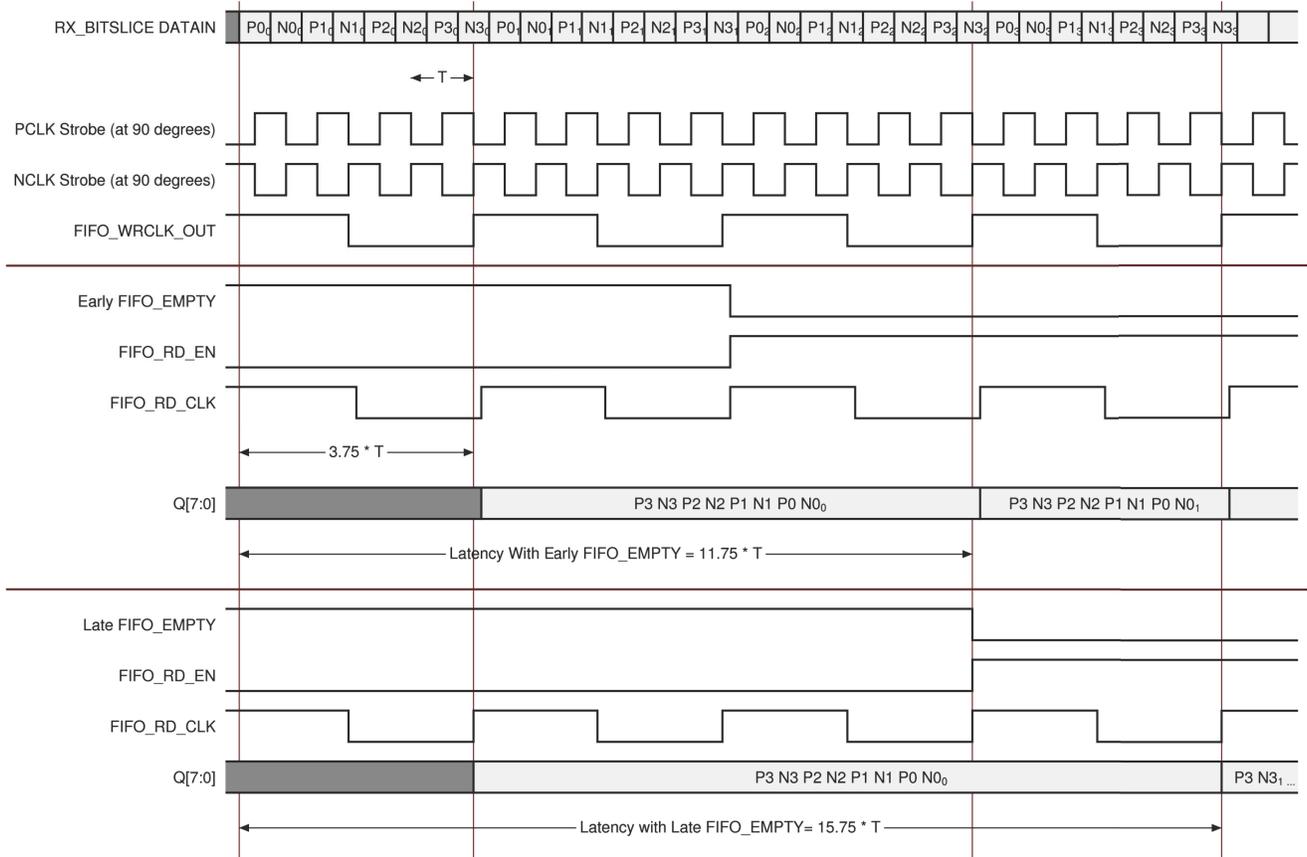
X16333-121718

Figure 2-37: FIFO as Bit Slice Output Synchronizer (Multiple Clocks or Strokes)

**Note 1:** When a single clock or strobe is used, use an inverter in the FIFO\_EMPTY path of the farthest bit slice to flip-flop instead of a NOR gate. The optional flip-flop helps close timing.

**Note 2:** When multiple clocks or strobes are used, use a NOR gate assembling the FIFO\_EMPTY signals of all used BITSLICED in the path to the flip-flop. The optional flip-flop helps close timing.

As shown in Figure 2-38 and Figure 2-39, the latency through the FIFO depends on FIFO\_RD\_CLK. When the write pointer is updated early with respect to FIFO\_RD\_CLK, the latency through the FIFO is shorter.



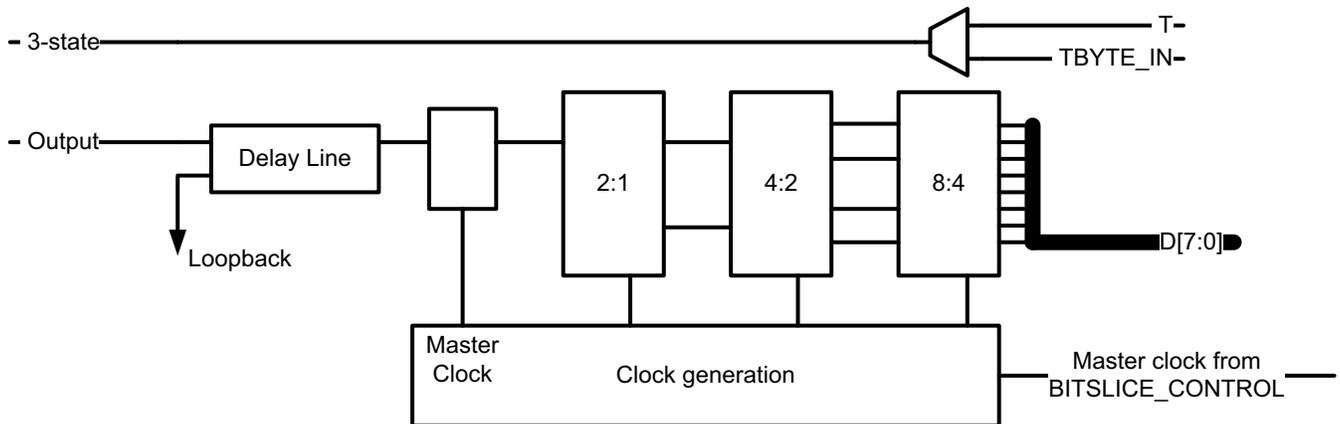
X19087-121718

Figure 2-38: RX\_BITSLICE FIFO, DATA\_WIDTH = 8



The RXTX\_BITSLICE has a loopback attribute allowing the output of the transmitter at the output of the delay line to be looped back to the receiver at the input of the delay line.

**Note:** This is a very useful option for debug and control applications.



X16348-030916

Figure 2-40: RXTX\_BITSLICE Transmitter Block Diagram

Both 3-state possibilities are passing through the transmitter (Figure 2-40). The chosen 3-state option is set in the transmitter by the TBYTE\_CTL attribute.

- The transmitter operates on the high-speed clock supplied to the BITSlice\_CONTROL.PLL\_CLK input. In the BITSlice\_CONTROL primitive, a clock generator ensures all clocks for the transmitter are generated.
- The PLL\_CLK is best generated by one of the two PLLs behind the I/O bank in the same clock area. When following the [Native Mode Bring-up and Reset](#) section of the [BITSlice\\_CONTROL](#) section, the FPGA interconnect and internal RXTX\_BITSLICE clocks are aligned.
- Data presented to the RXTX\_BITSLICE.D inputs is captured in the bit slice and serialized to the bit slice output by clocks generated in BITSlice\_CONTROL.
- This data, 8-bit or 4-bit wide, is serialized and transmitted at the rate of the applied BITSlice\_CONTROL.PLL clock.
- The transmitter part of the RXTX\_BITSLICE or TX\_BITSLICE is normally used to serially transmit data bits, but when the bit slice D-inputs are pulled to a static level, it is possible to generate and transmit any predicted signal format. A 50/50 clock pattern is generated when the D[7:0] or D[3:0] inputs are pulled to 10101010 or 1010.
- The OUTPUT\_PHASE\_90 attributes in each transmitter provide help generating phase-aligned data and clocks or 90-degree shifted data or clock setups.

Latency through the transmitter:

- With `OUTPUT_PHASE_90 = FALSE`, the latency from loading eight parallel bits to a first serial output bit is shown in [Equation 2-17](#) for 8-bit ([Figure 2-41](#)):

$$T + (13/16)T = \text{latency} \quad \text{Equation 2-17}$$

where T is the period of the parallel load or interconnect logic clock, and shown in [Equation 2-18](#) for 4-bit (see [Figure 2-42](#)):

$$T + (5/8)T = \text{latency} \quad \text{Equation 2-18}$$

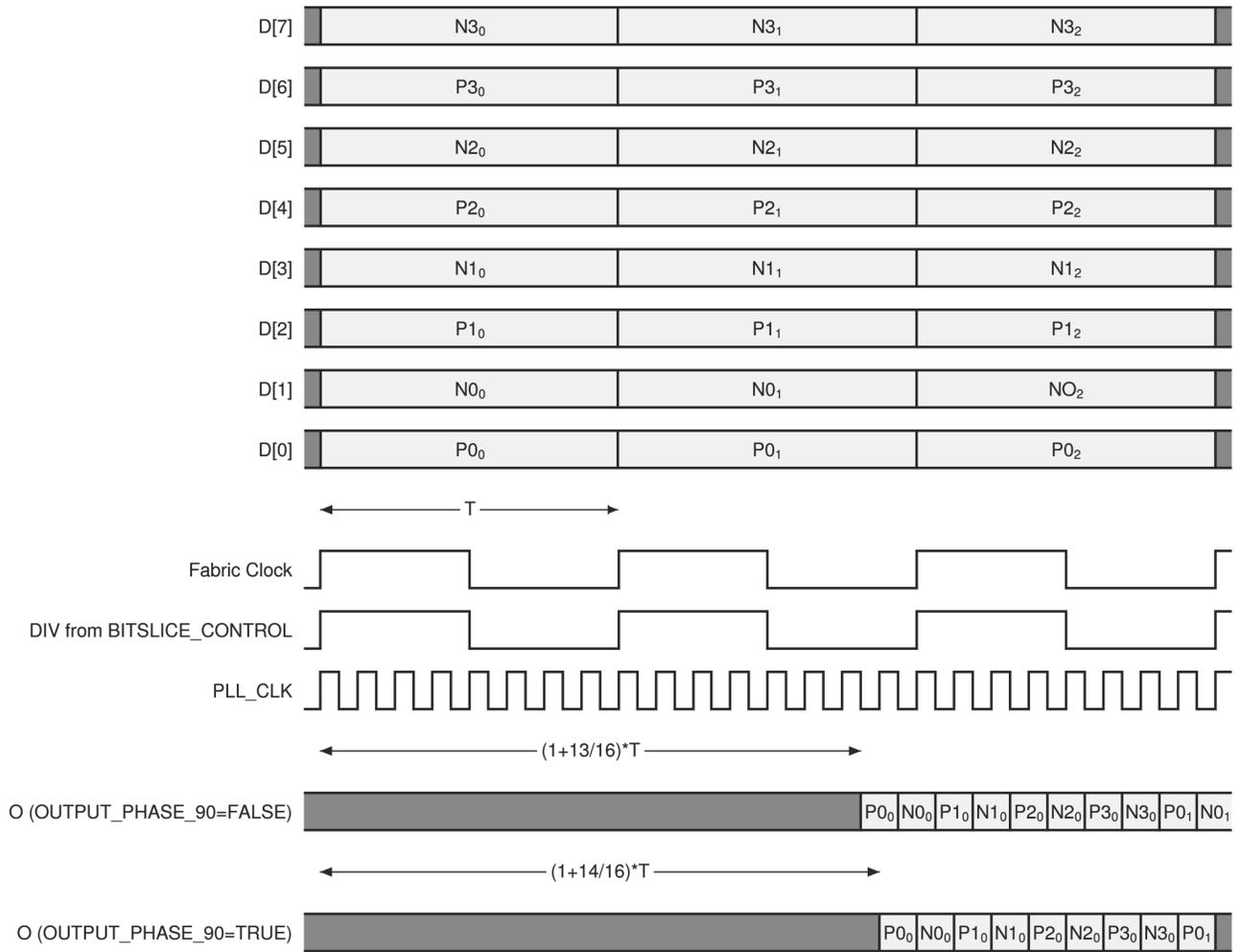
- With `OUTPUT_PHASE_90 = TRUE`, the latency from loading eight parallel bits to a first serial output bit is shown in [Equation 2-19](#) for 8-bit ([Figure 2-41](#)):

$$T + (14/16)T = \text{latency} \quad \text{Equation 2-19}$$

where T is the period of the parallel load or interconnect logic clock, and shown in [Equation 2-20](#) for 4-bit ([Figure 2-42](#)):

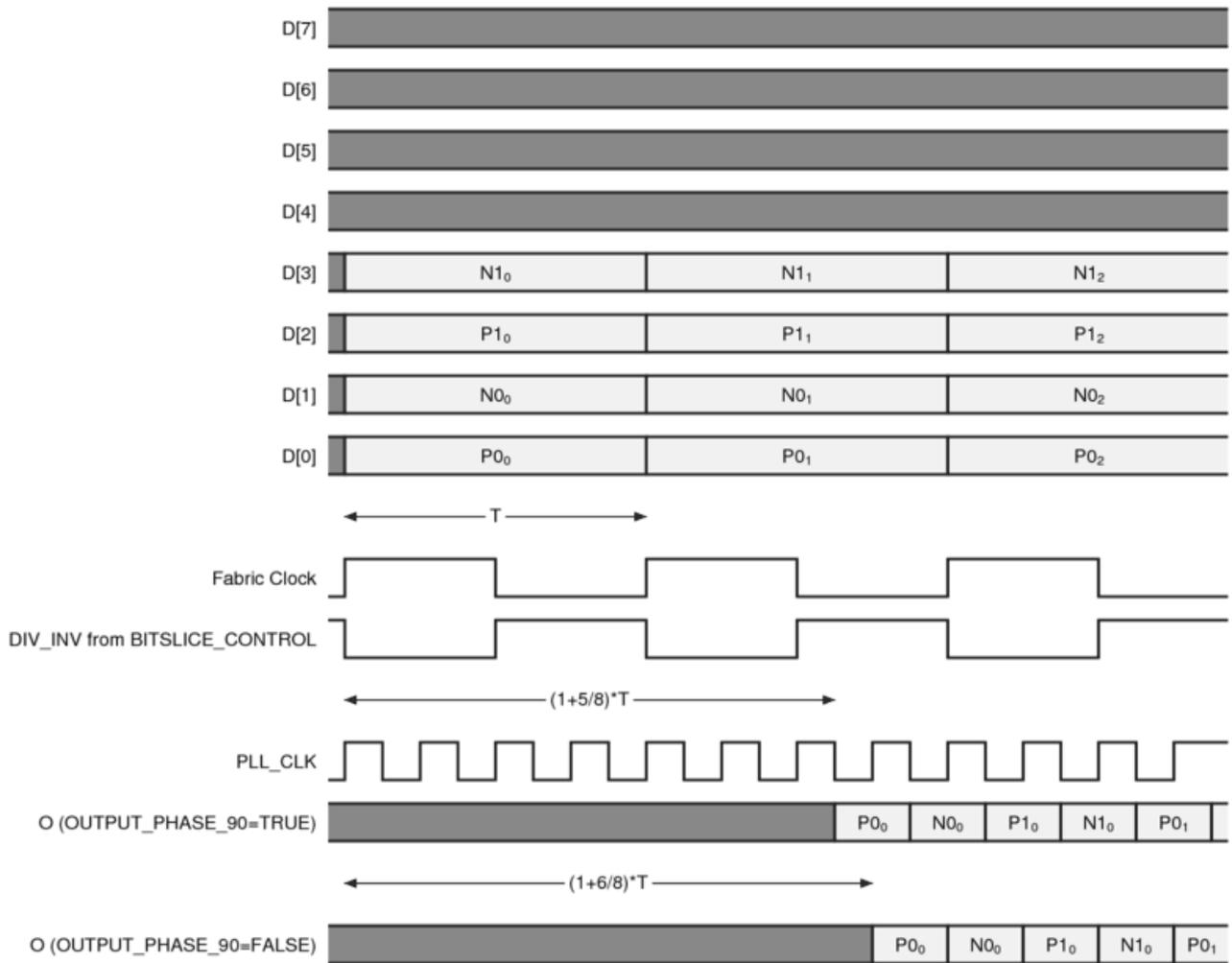
$$1T + (6/8)T = \text{latency} \quad \text{Equation 2-20}$$

where T is the period of the parallel load or FPGA logic clock.



X19085-121718

Figure 2-41: TX\_BITSLICE Latency, DATA\_WIDTH = 8



X20090-112117

Figure 2-42: TX\_BITSLICE Latency, DATA\_WIDTH = 4

For the TX\_BITSLICE, the Fabric Clock does not directly connect to the TX\_BITSLICE. The PLL and BITSlice\_CONTROL create a divided clock. When DATA\_WIDTH=4, the divided clock is inverted. Transmitting clock and data are similar operations, so the same clocking rules must be followed:

- Transmitting a clock can be done from any of the RXTX\_BITSLICES in a nibble.
- The generated clock depends on the pattern applied at the D[7:0] pins of the transmitter.
- For example, when 01010101 is applied, a 50/50 clock with frequency equal to half the RXTX\_BITSLICE.PLL\_CLK is generated.

- Assume an output data rate of 1250 Mb/s is required and a clock must be generated, too.
  - 1250 Mb/s requires a PLL generated high-speed clock of 1250 MHz connected at the BITSlice\_CONTROL.PLL\_CLK.

### ***Native Output Delay Type Usage***

#### **FIXED Mode**

The DELAY\_TYPE attribute set to FIXED selects the fixed delay through the output delay line and is determined by the DELAY\_VALUE and DELAY\_FORMAT attribute. When the DELAY\_FORMAT is set to TIME, the value loaded in the delay line is in ps. When the DELAY\_FORMAT is set to COUNT, the delay value loaded in the delay line is the number of taps. During the reset sequence, RXTX\_EN\_VTC is kept High by the High-Speed SelectIO wizard. After the reset sequence has finished, the RXTX\_EN\_VTC should be controlled based on the DELAY\_FORMAT setting.

- When DELAY\_FORMAT is TIME, then RXTX\_BITSLICE.EN\_VTC must be pulled High so that the delay automatically changes the number of taps over voltage and temperature to ensure the delay stays at the requested time in ps.
- With DELAY\_FORMAT set to COUNT, RXTX\_BITSLICE.EN\_VTC must be Low in Count mode. Then the delay is not compensated for voltage and temperature.

#### **VARIABLE Mode**

The DELAY\_TYPE attribute set to VARIABLE selects the variable tap delay line ([Table 2-13, page 172](#)). In VARIABLE mode, the CE and INC pins 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. The increment/decrement operation depends on the UPDATE\_MODE attribute (see [Figure 2-43](#)).

The RXTX\_BITSLICE.EN\_VTC pin should be held Low during the delay change command to ensure that any automatic adjustments are stopped.

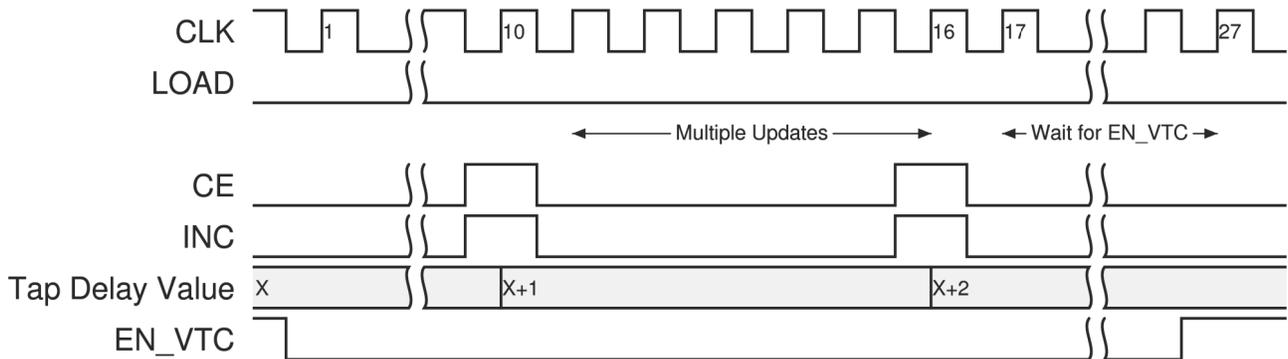
**Note:** The input and output delays for each bitslice share a delay ratio register (see [Table 2-52](#)). The delay ratio register ensures the input delays are correctly calibrated based on the reference clock frequency. Output delays must match input delays for optimal calibration accuracy for the output delays.

To increment/decrement delay lines when using TIME mode:

1. Deassert (Low) the RXTX\_BITSLICE.EN\_VTC pin.
2. Wait a minimum of 10 clock cycles.
3. Use the CE and INC ports to increment or decrement the delay line.
4. Wait a minimum of 10 clock cycles.

5. Assert the RXTX\_BITSLICE.EN\_VTC pin.

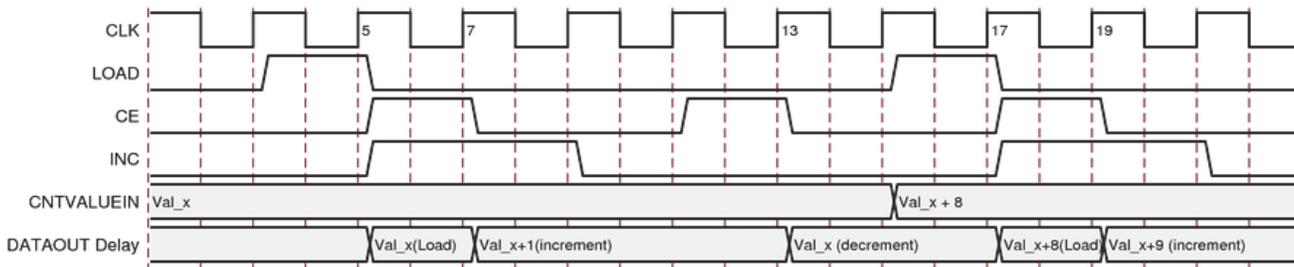
When using the delay line in COUNT mode, the RXTX\_BITSLICE.EN\_VTC port is always Low after the reset sequence has completed. Use the TIME mode procedure in [step 2](#) through [step 4](#) in this VARIABLE Mode section.



X19088-121018

Figure 2-43: Variable Mode, UPDATE\_MODE = ASYNC

### VAR\_LOAD Mode



X17479-072516

Figure 2-44: VAR\_LOAD Mode, UPDATE\_MODE = ASYNC

When the DELAY\_TYPE attribute is set to VAR\_LOAD, the delay line can be changed using the CE and INC inputs or the CNTVALUEIN, CNTVALUEOUT, and LOAD pins can be used to parallel load the delay line tap selection. Using these inputs, the delay line can be changed from 1 to 8 taps at a time.

The VAR\_LOAD method is suitable for both COUNT and TIME mode usage of the delay line.

In both modes, the tap amount can be read from the CNTVALUEOUT bus and changed through the CNTVALUEIN bus or CE and INC ports if necessary.

**Note:** The procedure to calculate the value to update the delay line is different for input and output delay lines. The procedure to calculate the value to update the delay line is different for TIME and COUNT mode.

If DELAY\_TYPE is VAR\_LOAD and DELAY\_FORMAT is TIME/COUNT, the procedure to update the delay line follows (Figure 2-22):

1. After BITSlice\_CONTROL.DLY\_RDY goes High, BITSlice\_CONTROL.EN\_VTC must be pulled High during the reset sequence.
2. When BITSlice\_CONTROL.VTC\_RDY goes High, BISC has completed. Make RXTX\_BITSLICE.EN\_VTC Low to modify the delay line.
3. Wait at least 10 clock cycles.
4. Read CNTVALUEOUT[8:0] and load the value into a register.
5. Check if updating the delay line is necessary.
6. Calculate the new delay value to be written in the delay line.
  - a. Increment or decrement the current tap position (Org\_Val) by 8 taps for glitchless transition. Jumps higher than 8 taps might result in the delay line jump causing data to glitch. For outputs, the glitch can corrupt the serialization.

**Note:** For the last pass, fewer than 8 taps might be needed.
  - b. Put the new delay line value on the CNTVALUEIN[8:0] bus.
  - c. Wait for one clock cycle and pulse LOAD High for a clock cycle.
  - d. Check if the new delay line value (New\_Val) is reached.
    - If not, continue from [step a](#).
    - If so, continue with [step 7](#).or
  - a. Calculate the difference (Dif\_Val) between New\_Val and Org\_Val and the direction to step.
  - b. Make the INC input High or Low to increment or decrement the delay line.
  - c. Toggle the CE pin to execute the increment or decrement.
  - d. Decrement the Dif\_Val and check if it is zero.
    - If not, continue from [step a](#).
    - If so, continue to [step 7](#).
7. Wait for at least 10 clock cycles.
8. Pull RXTX\_BITSLICE.EN\_VTC back High.
9. Go back to [step 2](#) for a new delay line update.

**RXTX\_BITSLICE Ports**

The RXTX\_BITSLICE primitive is shown in Figure 2-45. In this figure, black represents inputs and gray represents outputs. Table 2-22 lists the RXTX\_BITSLICE ports.

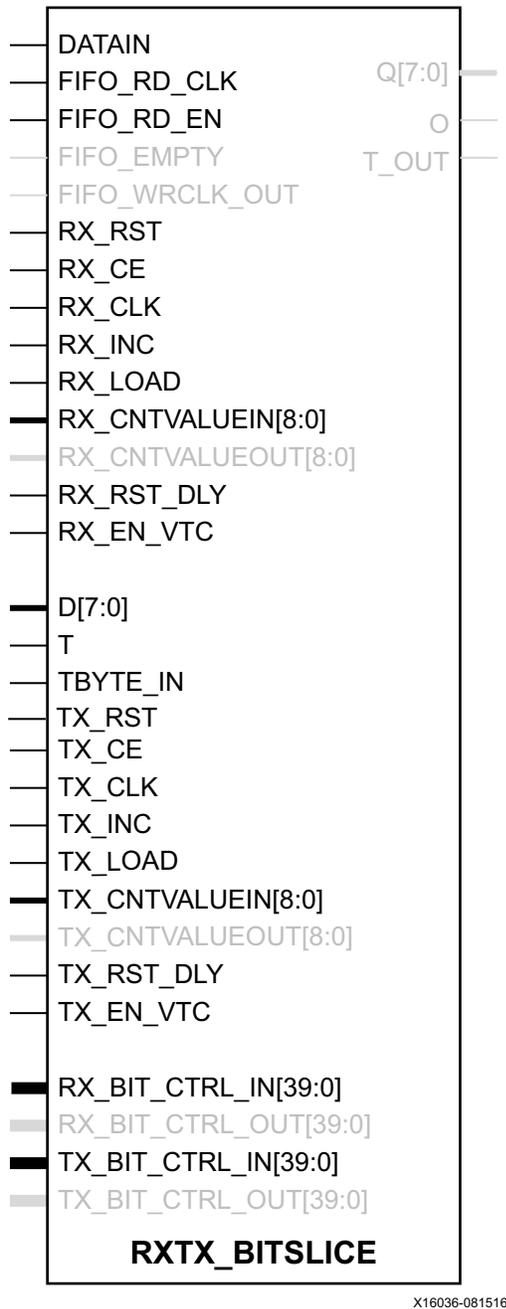


Figure 2-45: RXTX\_BITSLICE Primitive

Table 2-22 lists the RXTX\_BITSLICE ports.

Table 2-22: RXTX\_BITSLICE Ports

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
DATAIN	I/O RX	Input	Asynchronous	<p>This is the input signal from the IOB. When a differential input buffer is used with a single output (example IBUFDS), the RX_BITSLICE adjacent to the P-side of the differential pair is used. If a differential input buffer with complementary outputs (example IBUFDS_DIFF_OUT) is used, adjacent RX_BITSLICES for both the P and N inputs are used.</p> <p>The incoming signal from the IOB can be data, clock, or strobe, selected by the DATA_TYPE attribute on the RX_BITSLICE.</p> <p>When configured as either a clock or both clock and data, DATAIN is the incoming strobe/clock being forwarded through the RX_BITSLICE and to the BITSlice_CONTROL to create the clock to the other RX_BITSLICES to capture data. This strobe/clock bit slice must be positioned on a QBC or DBC IOB site, which is always located at bit slice position zero in a nibble. See the <a href="#">Clocking in Native Mode, page 302</a> in the <a href="#">BITSlice_CONTROL</a> section for more information.</p> <p>When the incoming signal from the IOB is data only, it can be located at any bit slice position in the nibble.</p>
Q[7:0]	RX FPGA	Output	FIFO_RD_CLK	<p>Deserialized (parallel) output data from the RX FIFO goes to the interconnect logic.</p> <p>If the DATA_WIDTH = 4, Q[3:0] outputs the captured data. Q[7:4] can be left unconnected and Q5 represents the serial data stream arriving at DATAIN.</p> <p><b>Note:</b> For BITSlice 0 and 6 (upper nibble BITSlice 0), the route through from DATAIN to Q5 can only be used after DLY_RDY is asserted.</p> <p>If DATA_WIDTH = 8, Q[7:0] represent 8 bits of captured serial data, and route through from DATAIN to Q5 is not available.</p>

Table 2-22: RXTX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
RX_RST	RX FPGA	Input	Asynchronous	Resets the receive side (RX_BITSLICE) logic, asynchronous assertion and synchronous deassertion and is active-High. Q resets to zero while RST is asserted.  See <a href="#">Native Mode Bring-up and Reset</a> for more information.
RX_CLK	RX FPGA	Input	Asynchronous	Delay line clock used to control RX_LOAD, RX_CE and RX_INC. All control inputs to delay line element within the receiver logic are synchronous to the clock input (RX_CLK). A clock must be connected to this port when the delay is configured in VARIABLE or VAR_LOAD. RX_CLK can be locally inverted and must be supplied by a global clock buffer.
RX_CE	RX FPGA	Input	RX_CLK	Clock enable for the delay line register lock.
RX_RST_DLY	RX FPGA	Input	Asynchronous (synchronously deassert to RX_CLK)	Reset port for the delay line component within the receiver logic. Resets the internal delay line to the value defined in RX_DELAY_VALUE.
RX_INC	RX FPGA	Input	RX_CLK	The increment/decrement is controlled by the enable signal (RX_CE). This interface is only available when the delay line is in VARIABLE or VAR_LOAD mode. As long as CE remains High, the delay line is incremented or decremented by one tap every clock (RX_CLK) cycle. The state of RX_INC determines whether delay line is incremented or decremented: RX_INC = 1 increments; RX_INC = 0 decrements, synchronously to the clock (RX_CLK). If RX_CE is Low, the delay through the delay line does not change (regardless of the state of RX_INC).  The programmable delay taps in the delay line wraps around. When the last tap delay is reached (RX_CNTVALUEOUT = 511), a subsequent increment function returns to tap 0. The same applies to the decrement function: decrementing from zero moves to tap 511.

Table 2-22: RXTX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
RX_LOAD	RX FPGA	Input	RX_CLK	When in VAR_LOAD mode and RX_UPDATE_MODE=ASYNCR, the delay line load port, RX_LOAD, loads the value set by the RX_CNTVALUEIN into the delay line. The value present at RX_CNTVALUEIN[8:0] is the new tap value. The RX_LOAD signal is an active-High signal and is synchronous to the input clock signal (RX_CLK). Wait at least one RX_CLK clock cycle after applying a new value on the RX_CNTVALUEIN bus before applying the LOAD signal. RX_CE must be held Low during RX_LOAD operation.
RX_EN_VTC	RX FPGA	Input	Asynchronous	Enable Voltage, temperature, and process calibration/compensation. High: Allows BITSlice_CONTROL to keep delay constant over VT. BITSlice_CONTROL.EN_VTC must be HIGH for VT compensation to be enabled. Low: VT compensation is disabled. When TIME mode is used, the RX_EN_VTC signal must be pulled High during initial built-in self-calibration (BISC). When COUNT mode is used, the RX_EN_VTC signal must be pulled Low. When bit slices are used in both TIME and COUNT mode in a nibble, RX_EN_VTC must be pulled High for the bit slices used in TIME mode, and must be pulled Low for those used in COUNT mode.
RX_CNTVALUEIN[8:0]	RX FPGA	Input	RX_CLK	The RX_CNTVALUEIN bus is used to dynamically change the loadable tap value. The 9-bit value at the RX_CNTVALUEIN is the number of taps required. New RX_CNTVALUEIN values should only be applied when RX_EN_VTC is Low. Apply new RX_CNTVALUEIN one clock cycle before RX_LOAD pulse. The new value is best applied one clock cycle before applying the LOAD signal. The delay line can be changed from 1 to 8 taps at a time.

Table 2-22: RXTX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
RX_CNTVALUEOUT[8:0]	RX FPGA	Output	RX_CLK	The RX_CNTVALUEOUT pins are used for reporting the current tap value, and reads out the amount of taps in the current delay. When RX_EN_VTC is High, RX_CNTVALUEOUT is updated by the BITSlice_CONTROL.
FIFO_RD_CLK	RX FPGA	Input	Asynchronous	The deserialized received data is read from the FIFO using the FIFO_RD_CLK signal. The FIFO read clock can be generated by PLL/MMCM or come from the FIFO_WRCLK_OUT output. Read <a href="#">FIFO Function, page 210</a> for more information.
FIFO_RD_EN	RX FPGA	Input	FIFO_RD_CLK	Enables a read operation from the RX FIFO, active-High. When this signal is Low, the FIFO read pointer is held at the same position. The effect of this is that the Q-output shows new data every eight clock cycles. See <a href="#">FIFO Function, page 210</a> .
FIFO_EMPTY	RX FPGA	Output	FIFO_RD_CLK	FIFO empty flag for this bit. When the FIFO write pointer and read pointer are the same, this signal is High. When inverted and registered, connect FIFO_EMPTY to FIFO_RD_EN to obtain a continuous data stream from the FIFO.
FIFO_WRCLK_OUT	RX FPGA	Output	PLL_CLK (for SERIAL_MODE) or DQS_IN (for source synchronous interfaces) (BITSlice_CONTROL)	This signal is only valid for a bit slice positioned at BITSlice 0 of a nibble. These pins for bit slices in other positions have no routing in the FPGA. The FIFO_WRCLK_OUT is a copy of the bit slice internal FIFO_WR_CLK. It is a divided version of the data sample clock/strobe. This clock writes the deserialized parallel data in the bit slice into the FIFO. <b>Note:</b> The use of this port is only recommended for experienced designers. Additional timing constraints are described in <a href="#">FIFO Function, page 210</a> .
D[7:0]	TX FPGA	Input	PLL_CLK (BITSlice_CONTROL)	Parallel incoming data from interconnect logic for transmit. Width is determined by the TX_DATA_WIDTH attribute and can be either 8 or 4. If the TX_DATA_WIDTH is 4, D[3:0] is used and D[7:4] should be tied to 0.

Table 2-22: RXTX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
T	TX FPGA	Input	Asynchronous	<p>T assigns a combinatorial path through the TX_BITSLICE to the 3-state pin of an output buffer.</p> <p>When the 3-state control is sourced from the interconnect logic, the T port must be used. Use of the T input of a bit slice can be seen as a block 3-state of the serial bitstream.</p> <p>Each TX_BITSLICE in a nibble has a T input, meaning that there are 13 T inputs for a byte (byte = two nibbles)</p>
TBYTE_IN	TX FPGA	Input	PLL_CLK (BITSlice_CONTROL)	<p>The TBYTE_IN is 1-bit wide input of the TX_BITSLICE side of the RXTX_BITSLICE. When using this 3-state, the TX_BITSLICE_TRI component must be used to serialize the TBYTE_IN[3:0] 3-state bus input of the BITSlice_CONTROL, giving the ability to 3-state individual bits in the serial output data stream. The TBYTE_IN[3:0] port of the BITSlice_CONTROL is handled and passes through the BITSlice_CONTROL to connect to the TX_BITSLICE_TRI. The TRI_OUT (TX_BITSLICE_TRI) then connects to each TBYTE_IN (TX_BITSLICE) input. A logic High means the data is not 3-stated and a logic Low means the data is 3-stated.</p>
O	I/O TX	Output	PLL_CLK (BITSlice_CONTROL)	<p>Serialized output data from the TX_BITSLICE that should be connected to the output buffer (or bidirectional buffer).</p>
T_OUT	I/O TX	Output	PLL_CLK (when TBYTE_CTL set to TBYTE_IN) otherwise Asynchronous (BITSlice_CONTROL)	<p>3-state output from the TX_BITSLICE that should be connected to the output buffer (or bidirectional buffer). The output can be either the combinatorial output when TBYTE_CTL is set to T or the serialized output when TBYTE_CTL is set to TBYTE_IN.</p>

Table 2-22: RXTX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
TX_RST	TX FPGA	Input	Asynchronous	Resets the transmit side (TX_BITSLICE), asynchronous assertion and synchronous deassertion and is active-High. 0 resets to the INIT attribute value while RST is asserted. For deterministic bring-up, follow the steps in <a href="#">Native Mode Bring-up and Reset, page 296</a> .
TX_CLK	TX FPGA	Input	Asynchronous	Delay line clock used to sample TX_LOAD, TX_CE, and TX_INC. All control inputs to output delay line element within the TX part of the RXTX_BITSLICE are synchronous to the clock input (TX_CLK). A clock must be connected to this port when the delay is configured in VARIABLE or VAR_LOAD. The TX_CLK can be locally inverted, and must be supplied by a global clock buffer.
TX_CE	TX FPGA	Input	TX_CLK	Clock enable for the output delay line register clock.
TX_RST_DLY	TX FPGA	Input	Asynchronous (synchronously deassert to TX_CLK)	Reset port for the delay line component within the transmit logic. Resets the internal delay line to the value defined in the TX_DELAY_VALUE attribute.

Table 2-22: RXTX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
TX_INC	TX FPGA	Input	TX_CLK	<p>The increment/decrement is controlled by the enable signal (TX_CE). This interface is only available when the delay line is in VARIABLE or VAR_LOAD mode. As long as TX_CE remains High, the delay line is incremented or decremented by one tap every clock (TX_CLK) cycle. The state of TX_INC determines whether the delay line is incremented or decremented: TX_INC = 1 increments; TX_INC = 0 decrements, synchronously to the clock (TX_CLK). If TX_CE is Low, the delay through the delay line does not change (regardless of the state of TX_INC). When TX_CE goes High, the increment/decrement operation begins on the next positive clock edge. When TX_CE goes Low, the increment/decrement operation ceases on the next positive clock edge.</p> <p>The programmable delay taps in the delay line primitive wrap around. When the last tap delay is reached (TX_CNTVALUEOUT = 511), a subsequent increment function returns to tap 0. The same applies to the decrement function: decrementing from zero moves to tap 511.</p>
TX_LOAD	TX FPGA	Input	TX_CLK	<p>When in VAR_LOAD mode, this input loads the value set by the TX_CNTVALUEIN into the delay line and TX_UPDATE_MODE = ASYNC. The value present at TX_CNTVALUEIN[8:0] is the new tap value. The TX_LOAD signal is an active-High signal and is synchronous to the input clock signal (TX_CLK). The TX_CE must be held Low during TX_LOAD operation.</p>

Table 2-22: RXTX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
TX_EN_VTC	TX FPGA	Input	Asynchronous	<p>Enable voltage, temperature, and process compensation.</p> <p>High: Allows BITSlice_CONTROL to keep delay constant over VT. BITSlice_CONTROL.EN_VTC must be High for VT compensation to be enabled.</p> <p>Low: VT compensation is disabled.</p> <p>When TIME mode is used, the TX_EN_VTC signal must be pulled High during initial BISC.</p> <p>When used in COUNT mode, the TX_EN_VTC signal must be pulled Low.</p> <p>When bit slices are used in both COUNT and TIME mode in a nibble, TX_EN_VTC must be pulled High for the bit slices used in TIME mode, and pulled Low for those used in COUNT mode.</p>
TX_CNTVALUEIN[8:0]	TX FPGA	Input	TX_CLK	<p>The TX_CNTVALUEIN bus is used for dynamically changing the loadable tap value. The 9-bit value at the TX_CNTVALUEIN bus is the new tap value the delay line is set to after TX_LOAD. Provide the value on this bus at least one clock cycle before TX_LOAD. The delay line can be changed from 1 to 8 taps at a time.</p> <p><b>Note:</b> For VT compensation, the RXTX_BITSLICE only compensates for the input delay value when using TX_EN_VTC. Applications requiring the output delay to be compensated for require the input delay to match the output delay.</p>
TX_CNTVALUEOUT[8:0]	TX FPGA	Output	TX_CLK	<p>The TX_CNTVALUEOUT pins are used for reporting the current tap value, and read out the amount of taps in the current delay. When TX_EN_VTC is High, TX_CNTVALUEOUT is updated by the BITSlice_CONTROL.</p>

Table 2-22: RXTX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
The following RX/TX_BIT_CTRL_OUT and RX/TX_BIT_CTRL_IN pins are 40-bit bus connections between the RXTX_BITSLICE (RX_BITSLICE and/or TX_BITSLICE) and the BITSlice_CONTROL. Each of these 40-bit buses carry data, clocks, RIU, and status signals between the RXTX_BITSLICE (RX_BITSLICE, TX_BITSLICE), TX_BITSLICE_TRI, and BITSlice_CONTROL and vice versa. When a bit slice is used, these buses <b>must</b> be connected to the appropriate BITSlice_CONTROL input and output bus. Example: When RXTX_BITSLICE_2 is used, RX/TX_BIT_CTRL_OUT of that RXTX_BITSLICE must connect to the BITSlice_CONTROL RX/TX_BIT_CTRL_IN2, and the RX/TX_BIT_CTRL_IN of the RXTX_BITSLICE buses must connect to the BITSlice_CONTROL RX/TX_BIT_CTRL_OUT2 buses. These buses are made of dedicated routing between the BITSlice_CONTROL and bit slices and cannot be accessed or used by logic in the FPGA. It is also not possible to connect an ILA or VIO to these buses and viewing the buses in simulation is meaningless because the content and bit names of the buses is not disclosed.				
RX_BIT_CTRL_IN[39:0]		Input	N/A	Input bus from BITSlice_CONTROL
RX_BIT_CTRL_OUT[39:0]		Output	N/A	Output bus to BITSlice_CONTROL
TX_BIT_CTRL_IN[39:0]		Input	N/A	Input bus from BITSlice_CONTROL
TX_BIT_CTRL_OUT[39:0]		Output	N/A	Output bus to BITSlice_CONTROL

**Notes:**

- I/O RX: Connections between the RX\_BITSLICE side of the RXTX\_BITSLICE and the I/O buffers.  
 I/O TX: Connections between the TX\_BITSLICE side of the RXTX\_BITSLICE and the I/O buffers.  
 RX FPGA: Connections from/to the RX\_BITSLICE side of the RXTX\_BITSLICE and the FPGA logic.  
 TX FPGA: Connections from/to the TX\_BITSLICE side of the RXTX\_BITSLICE and the FPGA logic.

### RXTX\_BITSLICE Attributes

Table 2-23 lists the RXTX\_BITSLICE attributes.

Table 2-23: RXTX\_BITSLICE Attributes

Attributes	Values	Default	Type	Description
RX_DATA_TYPE	DATA DATA_AND_CLOCK SERIAL	DATA	String	<p>Attribute defining the type of signal BITSlice is receiving (DATA, DATA_AND_CLOCK, or SERIAL) and the capture clock to be used.</p> <p>SERIAL = When received data must be captured by an unrelated clock (for example SGMII).</p> <p>DATA_AND_CLOCK = When the received signal is either clock/strobe or data. When the received clock/strobe must be sampled as if it is data.</p> <p>DATA = When the received signal contains purely data information.</p> <p>DATA_AND_CLOCK is only used for bit slices positioned at DBC, QBC or GC pins (bitslice_0).</p> <p>DATA can be used for all bit slices in a nibble when the received signal contains pure data information.</p>

Table 2-23: RXTX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description	
RX_DATA_WIDTH	4 or 8	8	Decimal	<p><b>Note:</b> Since BITSlice_CONTROL.DIV_MODE must match the data width, TX_DATA_WIDTH and RX_DATA_WIDTH must be same.</p> <p>Attribute defining the output width of the serial-to-parallel converter.</p> <p>This specifies the width that the incoming data is expanded to in the serial-to-parallel converter (deserialization), and it should match the DIV_MODE clock division setting of the corresponding BITSlice_CONTROL as shown in this table:</p>	
				RXTX_BITSLICE_DATA_WIDTH	BITSlice_CONTROL_DIV_MODE
				4	2
				8	4
RX_DELAY_FORMAT	TIME <sup>(1)</sup> or COUNT	TIME	String	<p><b>Note:</b> For BISC to properly align RXTX_BITSLICES, set TX_DELAY_FORMAT= RX_DELAY_FORMAT.</p> <p>DELAY_FORMAT can be either TIME or COUNT.</p> <p>When set to TIME, the input delay equals DELAY_VALUE (specified in ps) plus an additional alignment delay (Align_Delay) after BISC completes (DLY_RDY goes HIGH).</p> <p>BISC uses the RX_REFCLK_FREQUENCY attribute in conjunction with the incoming master clock to determine how many taps are required to achieve the requested TIME value (RX_DELAY_VALUE). This calibration accounts for the process variation in the device.</p> <p>When set to COUNT, the value given in RX_DELAY_VALUE is the number of taps required.</p>	

Table 2-23: RXTX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
RX_DELAY_TYPE	FIXED VAR_LOAD VARIABLE	FIXED	String	Delay mode of the input delay line. For more information, see <a href="#">Native Input Delay Type Usage, page 204</a> .
RX_DELAY_VALUE	0–1250 (TIME UltraScale) 0–1100 (TIME UltraScale+) 0–511 (COUNT)	0	Decimal	<p><b>Note:</b> For BISC to properly align, set RX_CLK_PHASE_P = RX_CLK_PHASE_N = SHIFT_0.</p> <p>TIME mode: Desired value in ps. UltraScale devices support delays up to 1.25 ns. UltraScale+ devices support up to 1.1 ns.</p> <p>COUNT mode: Desired value in taps. To ensure TX_BITSLICE data alignment, limit COUNT delays to 1.5 UI.</p> <p>For more information, see <a href="#">Native Input Delay Type Usage, page 204</a>.</p>
TX_DATA_WIDTH	4 or 8	8	Decimal	<p>Since BITSlice_CONTROL.DIV_MODE must match the data width, TX_DATA_WIDTH and RX_DATA_WIDTH must be same.</p> <p>Attribute defining the input width of the parallel-to-serial converter.</p> <p><math>TX\_DATA\_WIDTH = 2 \times BITSlice\_CONTROL.DIV\_MODE</math></p>

Table 2-23: RXTX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
TX_DELAY_FORMAT	TIME <sup>(1)</sup> or COUNT	TIME	String	<p><b>Note:</b> For BISC to properly calibrate RXTX_BITSLICES, set TX_DELAY_FORMAT= RX_DELAY_FORMAT.</p> <p>TX_DELAY_FORMAT can be either TIME or COUNT.</p> <p>When set to TIME, the delay after BISC completes (DLY_RDY goes High) equals the delay given in TX_DELAY_VALUE (specified in ps).</p> <p>BISC uses the TX_REFCLK_FREQUENCY attribute in conjunction with the incoming master clock to determine how many taps are required to achieve the requested TIME value (TX_DELAY_VALUE). This calibration accounts for the process variation in the device.</p> <p>When set to COUNT, the value given in TX_DELAY_VALUE is the number of taps required.</p>
TX_DELAY_TYPE	FIXED VAR_LOAD VARIABLE	FIXED	String	<p>Delay mode of the output delay line.</p> <p>For further information, see <a href="#">Native Output Delay Type Usage, page 220</a>.</p>
TX_DELAY_VALUE	0–1250 (TIME UltraScale) 0–1100 (TIME UltraScale+) 0–511 (COUNT)	0	Decimal	<p><b>Note:</b> For BISC to properly calibrate RXTX_BITSLICES, set TX_DELAY_VALUE= RX_DELAY_VALUE and TX_OUTPUT_PHASE_90 = FALSE.</p> <p>TIME mode: Desired value of the delay line in ps.</p> <p>UltraScale devices support delays up to 1.25 ns.</p> <p>UltraScale+ devices support up to 1.1 ns.</p> <p>COUNT mode: Desired value of the delay line in taps. To ensure TX_BITSLICE data alignment, limit COUNT delays to 1.5 UI.</p>

Table 2-23: RXTX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
RX_REFCLK_FREQUENCY	200.00–2400.00 (UltraScale) 300.00–2666.67 (UltraScale+)	300.0	1 significant digit float	<p><b>Note:</b> Since there is only a single reference clock for BITSlice_CONTROL, set TX_REFCLK_FREQUENCY = RX_REFCLK_FREQUENCY.</p> <p>Specification of reference clock frequency in MHz.</p> <p>This is the frequency of the master clock, PLL_CLK, the BITSlice_CONTROL uses. This master clock is used by BISC to calibrate any TIME mode delays. The tap size is not determined by the RX_REFCLK_FREQUENCY. The tap size is defined in the UltraScale device data sheets as TIDELAY_RESOLUTION [Ref 2].</p> <p>The RX_REFCLK_FREQUENCY attribute along with the requested delay, RX_DELAY_VALUE, is used by BISC to calibrate the amount of taps to provide the requested delay of RX_DELAY_VALUE when RX_DELAY_FORMAT is set to TIME mode.</p>

Table 2-23: RXTX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
TX_REFCLK_FREQUENCY	200.00–2400.00 (UltraScale) 300.00–2666.67 (UltraScale+)	300.0	1 significant digit float	<p><b>Note:</b> Since there is only a single reference clock for BITSlice_CONTROL, set TX_REFCLK_FREQUENCY = RX_REFCLK_FREQUENCY.</p> <p>Specification of reference clock frequency in MHz.</p> <p>This is the frequency of the master clock, PLL_CLK, the BITSlice_CONTROL uses. This master clock is used by BISC to calibrate any TIME mode delays (Refer to native mode clocking/BISC sections). The tap size is not determined by the TX_REFCLK_FREQUENCY. The tap size is defined in the UltraScale device data sheets as TIDELAY_RESOLUTION [Ref 2].</p> <p>The TX_REFCLK_FREQUENCY attribute along with the requested delay, TX_DELAY_VALUE, is used by BISC to calibrate the amount of taps to provide the requested delay when TX_DELAY_FORMAT is set to TIME mode.</p>

Table 2-23: RXTX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
RX_UPDATE_MODE	ASYNC, SYNC, or MANUAL	ASYNC	String	<p>ASYNC: This is the default and preferred use method. Updates to the delay value are independent of the data being received. This mode is the preferred operation mode because it covers the function of both other modes, too.</p> <p>SYNC: updates require DATAIN transitions to synchronously update the delay with the DATAIN edges. This mode is suitable for clocks or data that are always available and switches on a periodic basis.</p> <p>MANUAL: It takes two assertions of LOAD for the new value to take effect. The first LOAD loads the value defined by CNTVALUEIN and the second LOAD must be asserted with an assertion of the CE for the new value to take effect. This is beneficial because you can update the delay when the data becomes idle.</p>

Table 2-23: RXTX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
TX_UPDATE_MODE	ASYNC, SYNC, or MANUAL	ASYNC	String	<p>ASYNC: This is the default and preferred use method. Updates to the delay value are independent of the data being received.</p> <p>This mode is the preferred operation mode because it covers the function of both other modes, too.</p> <p>SYNC: Updates require DATAIN transitions to synchronously update the delay with the DATAIN edges. This mode is suitable for clocks or data that are always available and switches on a periodic basis.</p> <p>MANUAL: It takes two assertions of LOAD for the new value to take effect. The first LOAD loads the value defined by CNTVALUEIN and the second LOAD must be asserted with an assertion of the CE for the new value to take effect. This is beneficial because you can update the delay when the data becomes idle.</p>
FIFO_SYNC_MODE	TRUE or FALSE	FALSE	BOOLSTRING	<p>Attribute defining the relationship between FIFO_WRCLK_OUT and FIFO_RD_CLK. Always set this attribute to FALSE.</p> <p>FIFO_SYNC_MODE = TRUE. Reserved for later use.</p> <p>See the <a href="#">Clocking in Native Mode, page 302</a> in the <a href="#">BITSlice_CONTROL</a> section for more information on these clocks.</p>
INIT	1'b0 or 1'b1	1'b1	Binary	<p>Defines the initial value of the serialized data output of the RXTX_BITSLICE/TX_BITSLICE.</p>

Table 2-23: RXTX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
LOOPBACK	TRUE or FALSE	FALSE	BOOLEAN	<p>FALSE: RXTX_BITSLICE has distinct input (DATAIN) and/or output (O) to the input or output of bidirectional buffers in the IOB.</p> <p>TRUE: The output O is looped back to the DATAIN. This loopback is achieved inside the RXTX_BITSLICE by connecting the output delay output to the input delay input. The delay lines are thus part of the loopback cycle.</p>
TBYTE_CTL	TBYTE_IN or T	TBYTE_IN	Decimal	<p>TBYTE_IN: The BITSlice_CONTROL.TBYTE_IN[3:0] input is used to pass the 3-state information to the T_OUT output. This requires that the RXTX_BITSLICE/TX_BITSLICE is used together with a TX_BITSLICE_TRI.</p> <p>T: The T input is used to pass the 3-state information from logic to the T_OUT output. T requires that the 3-state information is generated in the logic.</p>
TX_OUTPUT_PHASE_90	TRUE or FALSE	FALSE	String	<p>FALSE: Output of RXTX_BITSLICE/TX_BITSLICE is not phase-shifted.</p> <p>TRUE: Output of RXTX_BITSLICE/TX_BITSLICE is phase-shifted 90 degrees.</p> <p>RX_DELAY_VALUE/ TX_DELAY_VALUE must be set to 0 when TX_OUTPUT_PHASE_90 = TRUE.</p> <p>The phase shift can easily be observed when different transmitters are used. This attribute is most used to shift the generated clock 90 degrees to the generated data.</p>
ENABLE_PRE_EMPHASIS	TRUE or FALSE	FALSE	String	<p>Used in conjunction with attributes on the bidirectional IOB to enable and disable pre-emphasis.</p> <p>Pre-emphasis is documented in <a href="#">Transmitter Pre-Emphasis in Chapter 1</a>.</p>

Table 2-23: RXTX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
IS_RX_CLK_INVERTED	1'b0 or 1'b1	1'b0	Binary	<p>When 1 reverses polarity (inverts) the RX_CLK signal.</p> <p>Similar to the IS_RX_RST_INVERTED attribute but on the RX_CLK path.</p> <p>When IS_RX_CLK_INVERTED = 1, the inverter is used.</p> <p>When IS_RX_CLK_INVERTED = 0, the inverter is not used.</p>
IS_RX_RST_DLY_INVERTED	1'b0 or 1'b1	1'b0	Binary	<p>When 1 reverses polarity (inverts) the RX_RST_DLY signal.</p> <p>Similar to the IS_RX_RST_INVERTED attribute but on the RX_RST_DLY path.</p> <p>When IS_RX_RST_DLY_INVERTED = 1, the inverter is used.</p> <p>When IS_RX_RST_DLY_INVERTED = 0, the inverter is not used.</p>
IS_RX_RST_INVERTED	1'b0 or 1'b1	1'b0	Binary	<p>When 1 reverses polarity (inverts) the RX_RST signal.</p> <p>A selectable local inverter on the reset path that can change the polarity of the reset input.</p> <p>When IS_RX_RST_INVERTED = 1, the inverter is used.</p> <p>When IS_RX_RST_INVERTED = 0, the inverter is not used.</p>
IS_TX_CLK_INVERTED	1'b0 or 1'b1	1'b0	Binary	<p>When 1 reverses polarity (inverts) the TX_CLK signal.</p> <p>This attribute is similar to the IS_RX_RST_INVERTED attribute but on the TX_CLK path.</p> <p>When IS_TX_CLK_INVERTED = 1, the inverter is used.</p> <p>When IS_TX_CLK_INVERTED = 0, the inverter is not used.</p>

Table 2-23: RXTX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
IS_TX_RST_DLY_INVERTED	1'b0 or 1'b1	1'b0	Binary	When 1 reverses polarity (inverts) the TX_RST_DLY signal. Similar to the IS_RX_RST_INVERTED attribute but on the TX_RST_DLY path. When IS_TX_RST_DLY_INVERTED = 1, the inverter is used. When IS_TX_RST_DLY_INVERTED = 0, the inverter is not used.
IS_TX_RST_INVERTED	1'b0 or 1'b1	1'b0	Binary	When 1 reverses polarity (inverts) the TX_RST signal. A selectable local inverter on the reset path that can change the polarity of the reset input. When IS_TX_RST_INVERTED = 1, the inverter is used. When IS_TX_RST_INVERTED = 0, the inverter is not used.
NATIVE_ODELAY_BYPASS	TRUE or FALSE	FALSE	String	When TRUE, bypass the ODELAY. UltraScale+ FPGAs only: Reserved for memory interface generator (MIG). When TRUE, bypass the ODELAY.
SIM_DEVICE	Possible Values: ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2	ULTRASCALE	String	Sets the device version (ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2)

**Notes:**

1. When in TIME mode, calibration will affect the availability of bit slices within the nibble. See [Bank Overview](#) for more information.

### IS\_...\_INVERTED Attributes

Attributes of this format allow that signals with corresponding names are locally inverted. This means that the inversion of the signal happens inside the native component boundary without consuming any logic resources. An example of such a local inversion is shown in Figure 2-46.

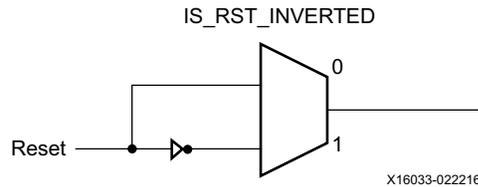


Figure 2-46: IS\_RST\_INVERTED Attribute

### RX\_BITSLICE

The RX\_BITSLICE is the receiver of the RXTX\_BITSLICE. For all receive interfaces, RXTX\_BITSLICE can be used except where CASCADE delay is needed. RX\_BITSLICE allows the two delay lines in the bit slice to be cascaded for a large delay.

Like the RXTX\_BITSLICE, the RX\_BITSLICE contains an input delay that can continuously be corrected for VT variation by the BITSlice\_CONTROL. High-speed capture registers, deserialization logic for either 1:4 or 1:8, and a shallow FIFO allow easy connection to another clock domain. A block diagram of RX\_BITSLICE is shown in Figure 2-47.

**Note:** The input buffer is not part of the RX\_BITSLICE.

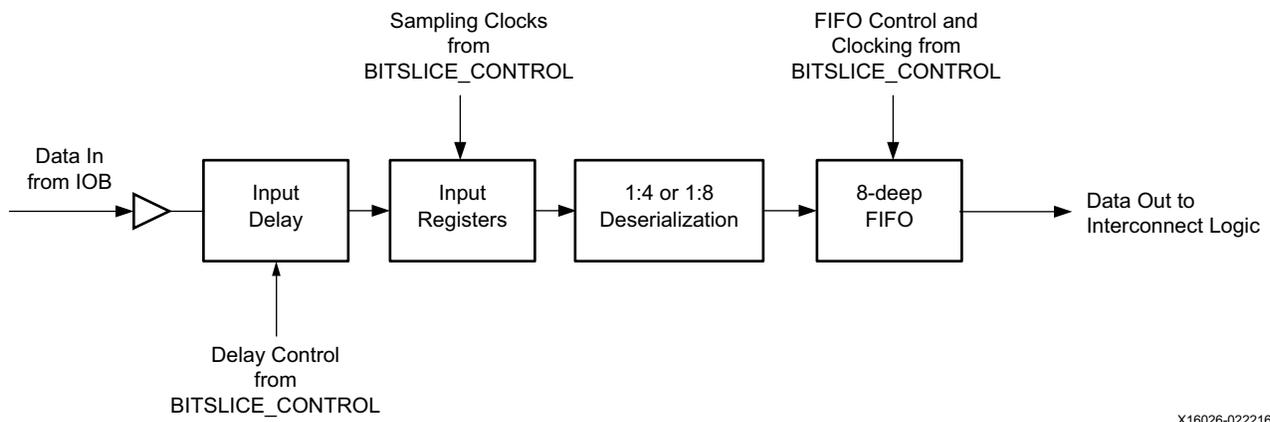


Figure 2-47: RX\_BITSLICE Block Diagram

The RX\_BITSLICE primitive is shown in Figure 2-48. In this figure, black represents inputs and gray represents outputs. Table 2-24 lists the RXTX\_BITSLICE ports.

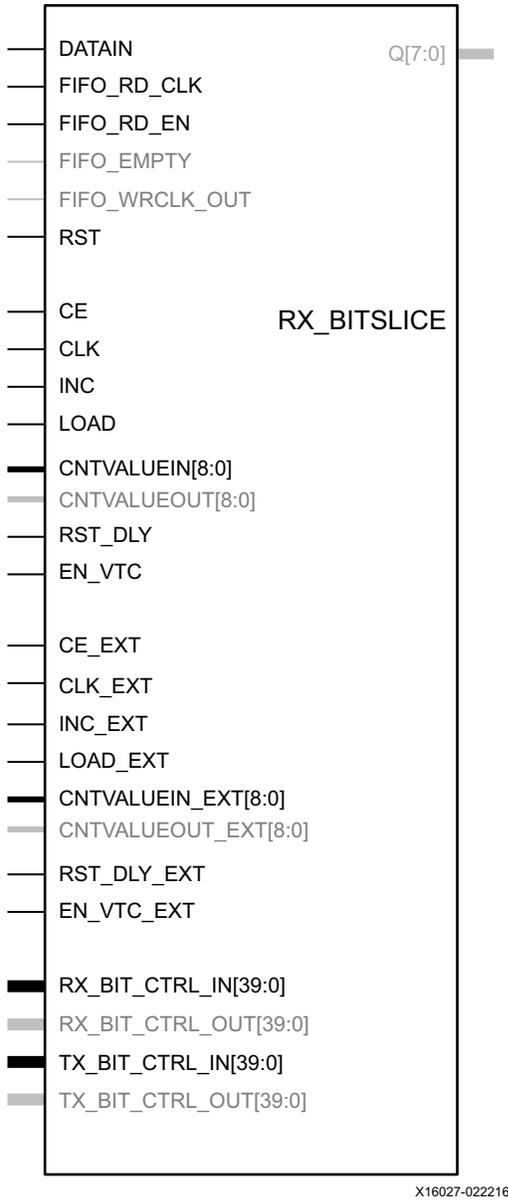


Figure 2-48: RX\_BITSLICE Primitive

### RX\_BITSLICE Function

The function of the RX\_BITSLICE is discussed in depth earlier in [RXTX\\_BITSLICE Receiver Function](#).

### RX\_BITSLICE Ports

[Table 2-24](#) lists the RX\_BITSLICE ports.

Table 2-24: RX\_BITSLICE Ports

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
DATAIN	I/O RX	Input	Asynchronous	<p>This is the input signal from the IOB. When a differential input buffer is used with a single output (example IBUFDS), the RX_BITSLICE adjacent to the P-side of the differential pair is used. If a differential input buffer with complementary outputs (example IBUFDS_DIFF_OUT) is used, adjacent RX_BITSLICES for both the P and N inputs are used.</p> <p>The incoming signal from the IOB can be data, clock, or strobe, selected by the DATA_TYPE attribute on the RX_BITSLICE.</p> <p>When configured as either a clock or both clock and data, DATAIN is the incoming strobe/clock being forwarded through the RX_BITSLICE and to the BITSlice_CONTROL to create the clock to the other RX_BITSLICES to capture data. This strobe/clock bit slice must be positioned on a QBC or DBC IOB site, which is always located at bit slice position zero in a nibble. See <a href="#">Clocking in Native Mode, page 302</a> in the <a href="#">BITSlice_CONTROL</a> section for more information.</p> <p>When the incoming signal from the IOB is data only, it can be located at any bit slice position in the nibble.</p>

Table 2-24: RX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
Q[7:0]	RX FPGA	Output	FIFO_RD_CLK	<p>Deserialized (parallel) output data from the RX FIFO going to the interconnect logic.</p> <p>If the DATA_WIDTH = 4, Q[3:0] outputs the captured data. Q[7:4] can be left unconnected and Q5 represents the serial data stream arriving at DATAIN.</p> <p><b>Note:</b> For BITSlice 0 and 6 (upper nibble BITSlice 0), the route through from DATAIN to Q5 can only be used after DLY_RDY is asserted.</p> <p>If DATA_WIDTH = 8, Q[7:0] represent 8 bits of captured serial data.</p>
RST	RX FPGA	Input	Asynchronous	<p>Resets the RX_BITSLICE 0 logic, asynchronous assertion, synchronous deassertion, and is active-High. Q resets to zero while RST is asserted.</p> <p>See <a href="#">Native Mode Bring-up and Reset, page 296</a> section for more information.</p>
CLK	RX FPGA	Input	Asynchronous	<p>Delay line clock used to control LOAD, CE, and INC. All control inputs to delay line element (LOAD, CE, and INC) are synchronous to the clock input (CLK). A clock must be connected to this port when the delay is configured in VARIABLE or VAR_LOAD. CLK can be locally inverted and must be supplied by a global clock buffer.</p>
CE	RX FPGA	Input	CLK	<p>Clock enable for the delay line register clock.</p> <p><b>Note:</b> Delays might take up to three clock cycles (CLK) to be applied. During this time, input data should not change to ensure output data does not glitch.</p>
RST_DLY	RX FPGA	Input	Asynchronous (synchronous deassertion to CLK)	<p>Reset port for the delay line within the receiver logic. Resets the internal delay line to the value defined in DELAY_VALUE.</p>

Table 2-24: RX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
INC	RX FPGA	Input	CLK	<p>The increment/decrement is controlled by the enable signal (CE). This interface is only available when the delay line is in VARIABLE or VAR_LOAD mode. As long as CE remains High, the delay line is incremented or decremented by one tap every clock (CLK) cycle. The state of INC determines whether the delay line is incremented or decremented: INC = 1 increments; INC = 0 decrements, synchronously to the clock (CLK). If CE is Low, the delay through the delay line does not change (regardless of the state of INC).</p> <p>When CE goes Low, the increment/decrement operation ceases on the next positive clock edge.</p> <p>The programmable delay taps in the delay line primitive wrap around. When the last tap delay is reached (CNTVALUEOUT = 511), a subsequent increment function returns to tap 0. The same applies to the decrement function: decrementing from zero moves to tap 511.</p>
LOAD	RX FPGA	Input	CLK	<p>When in VAR_LOAD mode and UPDATE_MODE = ASYNC, the delay line load port, LOAD, loads the value set by the CNTVALUEIN into the delay line. The value present at CNTVALUEIN[8:0] is the new tap value. The LOAD signal is an active-High signal and is synchronous to the input clock signal (CLK). Wait at least one clock cycle after applying a new value on the CNTVALUEIN bus before applying the LOAD signal. The CE must be held Low during LOAD operation.</p> <p><b>Note:</b> Delays might take up to three clock cycles (CLK) to be applied. During this time, input data should not change to ensure output data does not glitch.</p>

Table 2-24: RX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
EN_VTC	RX FPGA	Input	Asynchronous	<p>Enable Voltage, temperature, and process calibration/compensation.</p> <p>High: Allows BITSlice_CONTROL to keep delay constant over VT. BITSlice_CONTROL.EN_VTC must be High for VT compensation to be enabled.</p> <p>Low: VT compensation is disabled.</p> <p>When TIME mode is used, the EN_VTC signal must be pulled High during initial BISC.</p> <p>When COUNT mode is used, the EN_VTC signal must be pulled Low.</p> <p>When bit slices are used in both TIME and COUNT mode in a nibble, EN_VTC must be pulled High for the bit slices used in TIME mode, and pulled Low for those used in COUNT mode.</p>
CNTVALUEIN[8:0]	RX FPGA	Input	CLK	<p>The CNTVALUEIN bus is used to dynamically change the loadable tap value. The 9-bit value at the CNTVALUEIN is the number of taps required. New CNTVALUEIN values should only be applied when EN_VTC is Low.</p> <p>The new value is best applied one clock cycle before applying the LOAD signal. The delay line can be changed from 1 to 8 taps at a time.</p> <p>For RX_BITSLICES used as clock/strobe, CNTVALUEIN is not supported.</p> <p><a href="#">Clocking in Native Mode, page 302</a> in the <a href="#">BITSlice_CONTROL</a> section describes how the strobe/clock is tuned using BISC. Provide CNTVALUEIN one clock cycle before LOAD pulse High.</p>
CNTVALUEOUT[8:0]	RX FPGA	Output	CLK	<p>The CNTVALUEOUT pins are used for reporting the current tap value, and read out the amount of taps in the current delay. When EN_VTC is High, CNTVALUEOUT is updated by the BITSlice_CONTROL.</p>

Table 2-24: RX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
FIFO_RD_CLK	RX FPGA	Input	Asynchronous	The deserialized received data is read from the FIFO using the FIFO_RD_CLK signal. The FIFO_RD_CLK signal must be a divided version of the sampling frequency of the incoming data. See <a href="#">FIFO Function in RXTX_BITSLICE, page 198</a> .
FIFO_RD_EN	RX FPGA	Input	FIFO_RD_CLK	Enables a read operation from the FIFO when High. When Low, the FIFO read pointer is held at the same position. The effect of this is that the Q-output shows new data every eight clock cycles assuming write is happening continuously on every clock.
FIFO_EMPTY	RX FPGA	Output	FIFO_RD_CLK	FIFO empty flag for this bit. This is asserted High when FIFO write and read pointers are the same. When inverted and registered, connect FIFO_EMPTY to the FIFO_RD_EN to obtain a continuous data stream from the FIFO.
FIFO_WRCLK_OUT	RX FPGA	Output	PLL_CLK (for SERIAL_MODE) or DQS_IN (for source synchronous interfaces) (BITSlice_CONTROL)	This signal is only valid for a bit slice positioned at BITSlice 0 of a nibble. These pins for bit slices in other positions have no routing in the FPGA.  The FIFO_WRCLK_OUT is a copy of the bit slice internal FIFO_WR_CLK. It is a divided version of the data sample clock/strobe. This clock writes the deserialized parallel data in the bit slice into the FIFO.  The use of this port is only recommended for experienced designers.  Additional timing constraints are described in <a href="#">FIFO Function, page 210</a> .

Table 2-24: RX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
CLK_EXT	TX FPGA	Input	Asynchronous	When CASCADE=TRUE, CLK_EXT and CLK must be connected to the same clock source. Delay line clock used to sample LOAD_EXT, CE_EXT, and INC_EXT. All control inputs to the output delay line element are synchronous to the clock input (CLK_EXT). A clock must be connected to this port when the delay is configured in VARIABLE or VAR_LOAD. The CLK_EXT can be locally inverted, and must be supplied by a global clock buffer.
CE_EXT	TX FPGA	Input	CLK_EXT	Clock enable for the cascaded output delay line register clock.
RST_DLY_EXT	TX FPGA	Input	Asynchronous (synchronous deassertion to CLK)	Reset port for the cascaded output delay line. Resets the internal delay line to the value defined in the DELAY_VALUE attribute.

Table 2-24: RX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
INC_EXT	TX FPGA	Input	CLK_EXT	<p>The increment/decrement is controlled by the enable signal (CE_EXT). This interface is only available when the delay line is in VARIABLE or VAR_LOAD mode. As long as CE_EXT remains High, the delay line is incremented or decremented by one tap every clock (CLK_EXT) cycle. The state of INC_EXT determines whether the delay line is incremented or decremented: INC_EXT = 1 increments; INC_EXT = 0 decrements, synchronously to the clock (CLK_EXT). If CE_EXT is Low, the delay through the delay line does not change (regardless of the state of INC_EXT). When CE_EXT goes High, the increment/decrement operation begins on the next positive clock edge. When CE_EXT goes Low, the increment/decrement operation ceases on the next positive clock edge.</p> <p>The programmable delay taps in the delay line primitive wrap around. When the last tap delay is reached (CNTVALUEOUT_EXT = 511), a subsequent increment function returns to tap 0. The same applies to the decrement function: decrementing from zero moves to tap 511.</p>
LOAD_EXT	TX FPGA	Input	CLK_EXT	<p>When in VAR_LOAD mode, this input loads the value set by the CNTVALUEIN_EXT into the delay line. The value present at CNTVALUEIN_EXT [8:0] is the new tap value. The LOAD_EXT signal is an active-High signal and is synchronous to the input clock signal (CLK_EXT). Wait at least one CLK_EXT clock cycle after applying a new value on the CNTVALUEIN_EXT bus before applying the LOAD_EXT signal. The CE_EXT must be held Low during LOAD_EXT operation.</p>

Table 2-24: RX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
EN_VTC_EXT	TX FPGA	Input	Asynchronous	<p>Enable voltage, temperature, and process compensation.</p> <p>High: Allows BITSlice_CONTROL to keep delay constant over VT. BITSlice_CONTROL.EN_VTC must be High for VT compensation to be enabled.</p> <p>Low: VT compensation is disabled.</p> <p>When TIME mode is used, the EN_VTC_EXT signal must be pulled High during initial BISC.</p> <p>When COUNT mode is used, the EN_VTC_EXT signal must be pulled Low.</p> <p>When bit slices are used in both COUNT and TIME mode in a nibble, EN_VTC_EXT must be pulled High for the bit slices used in TIME mode, and pulled Low for those used in COUNT mode.</p>
CNTVALUEIN_EXT[8:0]	TX FPGA	Input	CLK_EXT	<p>The CNTVALUEIN_EXT bus is used for dynamically changing the loadable tap value. The 9-bit value at the CNTVALUEIN_EXT bus is the new tap value the output delay line is set to after LOAD_EXT. Provide the value on this bus at least one clock cycle before LOAD_EXT. The delay line can be changed from 1 to 8 taps at a time.</p>
CNTVALUEOUT_EXT[8:0]	TX FPGA	Output	CLK_EXT	<p>The CNTVALUEOUT_EXT pins are used for reporting the current output delay tap value, and reads out the amount of taps in the current delay. When EN_VTC_EXT is High, CNTVALUEOUT_EXT is updated by the BITSlice_CONTROL.</p>

Table 2-24: RX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
<p>The following RX/TX_BIT_CTRL_OUT and RX/TX_BIT_CTRL_IN pins are 40-bit bus connections between the RXTX_BITSLICE (RX_BITSLICE and/or TX_BITSLICE) and the BITSlice_CONTROL. Each of these 40-bit buses carry data, clocks, RIU, and status signals between the RXTX_BITSLICE (RX_BITSLICE, TX_BITSLICE), TX_BITSLICE_TRI, and BITSlice_CONTROL, and vice versa.</p> <p>When a bit slice is used, these buses <b>must</b> be connected to the appropriate BITSlice_CONTROL input and output bus.</p> <p>Example:</p> <p>When RXTX_BITSLICE_2 is used, RX/TX_BIT_CTRL_OUT of that RXTX_BITSLICE must connect to the BITSlice_CONTROL RX/TX_BIT_CTRL_IN2 and the RX/TX_BIT_CTRL_IN of the RXTX_BITSLICE buses must connect to the BITSlice_CONTROL RX/TX_BIT_CTRL_OUT2 buses.</p> <p>These buses are made of dedicated routing between the BITSlice_CONTROL and bit slices and cannot be accessed or used by logic. It's also not possible to connect an ILA or VIO to these buses, and viewing the buses in simulation is meaningless because the content and bit names of the buses is not disclosed.</p>				
RX_BIT_CTRL_IN[39:0]		Input	N/A	Input bus from BITSlice_CONTROL
RX_BIT_CTRL_OUT[39:0]		Output	N/A	Output bus to BITSlice_CONTROL
TX_BIT_CTRL_IN[39:0]		Input	N/A	Input bus from BITSlice_CONTROL
TX_BIT_CTRL_OUT[39:0]		Output	N/A	Output bus to BITSlice_CONTROL

**Notes:**

1. I/O RX: Connections between the RX\_BITSLICE side of the RXTX\_BITSLICE and the I/O buffers.  
 I/O TX: Connections between the TX\_BITSLICE side of the RXTX\_BITSLICE and the I/O buffers.  
 RX FPGA: Connections from/to the RX\_BITSLICE side of the RXTX\_BITSLICE and the logic.  
 TX FPGA: Connections from/to the TX\_BITSLICE side of the RXTX\_BITSLICE and the logic.

### RX\_BITSLICE Attributes

Table 2-25 lists the RX\_BITSLICE attributes.

Table 2-25: RX\_BITSLICE Attributes

Attributes	Values	Default	Type	Description						
DATA_TYPE	DATA DATA_AND_CLOCK SERIAL	DATA	String	<p>SERIAL = When received data must be captured by a unrelated clock (for example, SGMI).</p> <p>DATA_AND_CLOCK = When the received signal is either clock/strobe or data. When the received clock/strobe must be sampled as if it is data.</p> <p>DATA = When the received signal contains purely data information.</p> <p>DATA_AND_CLOCK is only used for bit slices positioned at DBC, QBC or GC pins (BITSlice_0).</p> <p>DATA can be used for all bit slices in a nibble when the received signal contains pure data information.</p>						
DATA_WIDTH	4 or 8	8	Decimal	<p>Attribute defining the output width of the serial-to-parallel converter.</p> <p>This specifies the width that the incoming data is expanded to in the serial-to-parallel converter (deserialization), and it should match the DIV_MODE clock division setting of the corresponding BITSlice_CONTROL as shown in this table:</p> <table border="1" data-bbox="1045 1304 1482 1467"> <thead> <tr> <th>RX_BITSLICE DATA_WIDTH</th> <th>BITSlice_CONTROL DIV_MODE</th> </tr> </thead> <tbody> <tr> <td>4</td> <td>2</td> </tr> <tr> <td>8</td> <td>4</td> </tr> </tbody> </table>	RX_BITSLICE DATA_WIDTH	BITSlice_CONTROL DIV_MODE	4	2	8	4
RX_BITSLICE DATA_WIDTH	BITSlice_CONTROL DIV_MODE									
4	2									
8	4									

Table 2-25: RX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
DELAY_FORMAT	TIME <sup>(1)</sup> or COUNT	TIME	String	<p>DELAY_FORMAT can be either TIME or COUNT.</p> <p>When set to TIME, the delay will be equal to DELAY_VALUE (specified in ps) plus an additional alignment delay (Align_Delay) after BISC completes (DLY_RDY goes HIGH). BISC uses the REFCLK_FREQUENCY attribute in conjunction with the incoming master clock to determine the current tap size and therefore how many taps are required to achieve the requested TIME value (DELAY_VALUE). This calibration accounts for the process variation in the device. When EN_VTC is High, the delay is calibrated to provide the requested TIME across voltage and temperature.</p> <p>When set to COUNT, the value given in DELAY_VALUE is the number of taps required. EN_VTC must be tied Low when using COUNT.</p>
DELAY_TYPE	FIXED VAR_LOAD VARIABLE	FIXED	String	<p>Delay mode of the input delay line. For further information, see <a href="#">Native Input Delay Type Usage, page 204</a>.</p>
DELAY_VALUE	0–1250 (TIME UltraScale) 0–1100 (TIME UltraScale+) 0–511 (COUNT)	0	Decimal	<p><b>Note:</b> For BISC to properly align, set OUTPUT_PHASE_90 = FALSE.</p> <p>TIME mode: Desired value in ps. UltraScale devices support delays up to 1.25 ns. UltraScale+ devices support delays up to 1.1 ns.</p> <p>COUNT mode: Desired value in taps.</p>

Table 2-25: RX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
DELAY_FORMAT_EXT	TIME <sup>(1)</sup> or COUNT	TIME	String	<p>DELAY_FORMAT_EXT can be either TIME or COUNT. Must match DELAY_FORMAT.</p> <p>The attribute value should match DELAY_FORMAT when CASCADE is set to TRUE.</p> <p>When set to COUNT, the value given in DELAY_VALUE is the number of taps required. EN_VTC_EXT must be tied Low when using COUNT.</p>
DELAY_TYPE_EXT	FIXED VAR_LOAD VARIABLE	FIXED	String	<p>Delay mode of the extended delay line. For further information, see <a href="#">Extended Delay Control Signals</a>.</p>
DELAY_VALUE_EXT	0–1250 (TIME UltraScale) 0–1100 (TIME UltraScale+) 0–511 (COUNT)	0	Decimal	<p>Delay value for extended delay.</p> <p>TIME mode: Desired value in ps. COUNT mode: Desired value in taps.</p> <p>For further information, see <a href="#">Extended Delay Control Signals, page 261</a>.</p>

Table 2-25: RX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
REFCLK_FREQUENCY	200.00–2400.00 (UltraScale) 300.00–2666.67 (UltraScale+)	300.0	1 significant digit float	<p>Specification of reference clock frequency in MHz.</p> <p>This is the frequency of the master clock, PLL_CLK or REFCLK, the BITSlice_CONTROL uses. This master clock is used by BISC to calibrate any TIME mode delays. The master clock is also used to generate necessary internal clocks for data capturing or data generation. The tap size is not determined by the REFCLK_FREQUENCY. The tap size is defined in the UltraScale device data sheets as TIDELAY_RESOLUTION [Ref 2].</p> <p>The REFCLK_FREQUENCY attribute is used by the BISC algorithm to calculate the tap size but not affect the tap size.</p> <p>When the DELAY_FORMAT attribute is set to TIME, the delay equals the value given in the DELAY_VALUE attribute. The delay is specified in ps and is calibrated using the REFCLK_FREQUENCY attribute. The REFCLK_FREQUENCY attribute is used in conjunction with the incoming reference clock to determine the current tap size and therefore how many taps are required to achieve the requested TIME. This calibration, using the reference clock, accounts for the process variation in the device. When the EN_VTC pin is High, the delay is calibrated to provide the TIME across voltage and temperature.</p>

Table 2-25: RX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
UPDATE_MODE	ASYNC, SYNC, or MANUAL	ASYNC	String	<p>ASYNC: This is the default and preferred use method. Updates to the delay value are independent of the data being received. This mode is the preferred operation mode because it covers the function of both other modes, too.</p> <p>SYNC: Updates require DATAIN transitions to synchronously update the delay with the DATAIN edges. This mode is suitable for clocks or data that are always available and switches on a periodic basis.</p> <p>MANUAL: It takes two assertions of LOAD for the new value to take effect. The first LOAD loads the value defined by CNTVALUEIN and the second LOAD must be asserted with an assertion of the CE for the new value to take effect. This is beneficial because you can update the delay when the data becomes idle.</p>
UPDATE_MODE_EXT	ASYNC, SYNC, or MANUAL	ASYNC	String	<p>ASYNC: This is the default and preferred use method. Updates to the delay value are independent of the data being received. This mode is the preferred operation mode because it covers the function of both other modes, too.</p> <p>SYNC: Updates require DATAIN transitions to synchronously update the delay with the DATAIN edges. This mode is suitable for clocks or data that are always available and switches on a periodic basis.</p> <p>MANUAL: It takes two assertions of LOAD for the new value to take effect. The first LOAD loads the value defined by CNTVALUEIN and the second LOAD must be asserted with an assertion of the CE for the new value to take effect. This is beneficial because you can update the delay when the data becomes idle. For further information, see <a href="#">Extended Delay Control Signals, page 261</a>. Value should match UPDATE_MODE value.</p>

Table 2-25: RX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
FIFO_SYNC_MODE	TRUE (Reserved) or FALSE	FALSE	BOOLSTRING	<p>FALSE: This attribute defines the relationship between FIFO_WRCLK_OUT and FIFO_RD_CLK. Always set this attribute to FALSE.</p> <p><b>Note:</b> FIFO_SYNC_MODE = TRUE. Reserved for later use.</p> <p>See <a href="#">Clocking in Native Mode, page 302</a> in the <a href="#">BITSlice_CONTROL</a> section for more information on these clocks.</p>
CASCADE	TRUE or FALSE	FALSE	String	<p>TRUE: Enables cascading of input and output delay lines of neighboring RX and TX bit slices. When both delay lines are cascaded, a delay of 2.5 ns can be realized. The extended delay is controlled by the _EXT pins.</p> <p>Consider the use of the attributes for the cascaded output delay line in addition to the master input delay attributes.</p> <p>FALSE: Disables cascading and the extended (_EXT) attributes can be ignored (pull input Low and leave outputs open).</p> <p>See the <a href="#">Extended Delay Control Signals, page 261</a> description for more information on delay cascading with the RX_BITSLICE.</p> <p><b>Note:</b> CASCADE = TRUE has reduced performance and should not be used when performance is critical.</p>
IS_CLK_INVERTED	1'b0 or 1'b1	1'b0	Binary	<p>Similar to the IS_RST_INVERTED attribute but on the RX_CLK path. When IS_CLK_INVERTED = 1 the inverter is used.</p> <p>When 1, reverses polarity (inverts) the CLK signal.</p> <p>When 0, the inverter is not used.</p>
IS_RST_DLY_INVERTED	1'b0 or 1'b1	1'b0	Binary	<p>Similar to the IS_RST_INVERTED attribute but on the RST_DLY path. When IS_RST_DLY_INVERTED = 1 the inverter is used.</p> <p>When 1, reverses polarity (inverts) the RST_DLY signal.</p> <p>When 0, the inverter is not used.</p>

Table 2-25: RX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
IS_RST_INVERTED	1'b0 or 1'b1	1'b0	Binary	A selectable local inverter on the reset path that can change the polarity of the reset input. When IS_RST_INVERTED = 1, the inverter is used. When 1, reverses polarity (inverts) the RST signal. When 0, the inverter is not used.
SIM_DEVICE	Possible Values: ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2	ULTRASC ALE	String	Sets the device version (ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2)

**Notes:**

1. When in TIME mode, calibration affects the availability of bit slices within the nibble. See [Bank Overview](#) for more information.

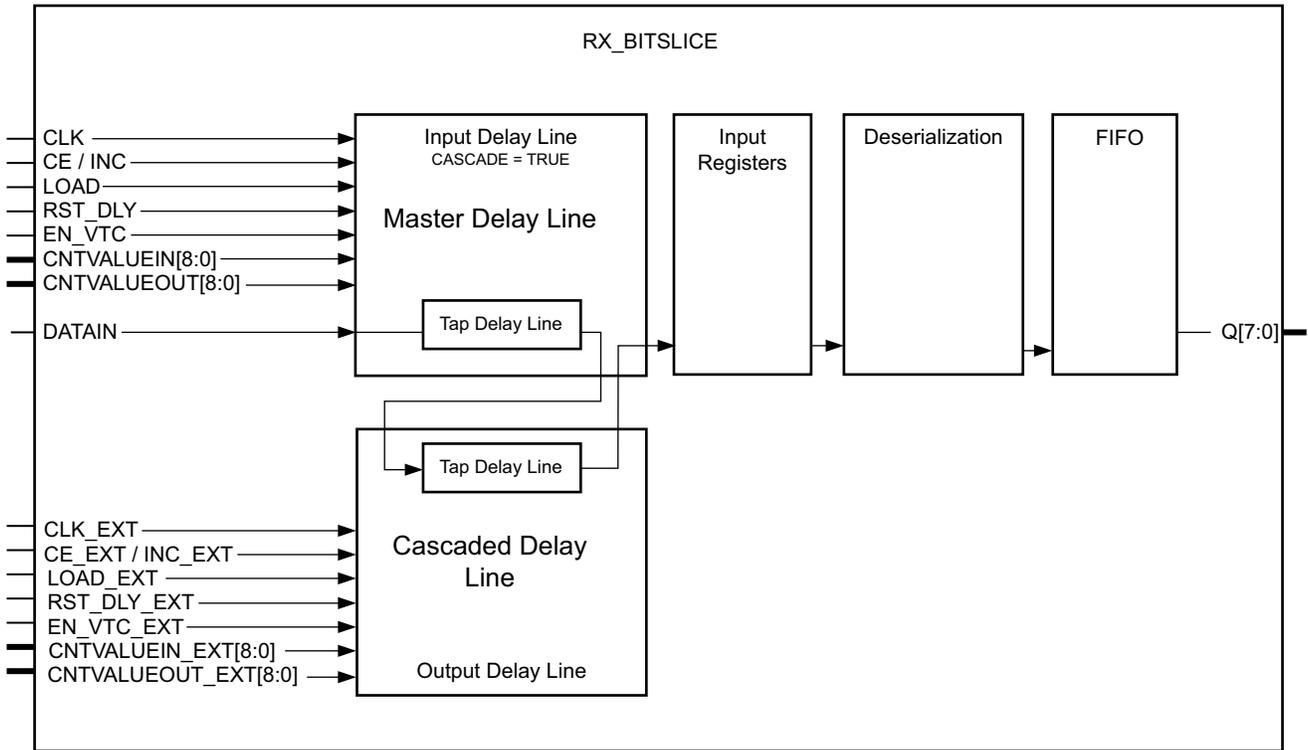
### Extended Delay Control Signals

Further information on the CE\_EXT, CLK\_EXT, EN\_VTC\_EXT, INC\_EXT, RST\_DLY\_EXT, LOAD\_EXT, CNTVALUEIN\_EXT[8:0], and CNTVALUEOUT\_EXT[8:0] signals is presented in this section.

In an RX\_BITSLICE, the input delay line can be extended with the output delay line. To do this, the CASCADE attribute must be set to TRUE. When this is the case, all ports with extension \_EXT must be used.

If CASCADE = FALSE, all ports with extension \_EXT must be tied to ground (GND).

When the RX\_BITSLICE is using the cascaded delay (attribute CASCADE = TRUE), then the output of the RX\_BITSLICE input delay is connected to the input of the TX\_BITSLICE (not used) output delay line. The output of the output delay line in the unused TX\_BITSLICE is connected to the input of the deserializer logic in the RX\_BITSLICE shown in [Figure 2-49](#). This effectively doubles the length of the delay line. Control of both delay lines is done through the control ports of each of the delays. The control ports of the output delay line are named with the \_EXT extension. The function of the control ports of both delay lines is the same.



X16319-030916

Figure 2-49: Extended Delay Control Signals

When DELAY\_TYPE is FIXED, the input delay line and output delay line (extended or \_EXT) control signals can be tied off to ground (GND) with the exception of EN\_VTC and EN\_VTC\_EXT. When the delay lines are used in TIME mode, both pins must be tied to V<sub>CC</sub> and/or actively manipulated. When the delay lines are used in COUNT mode, both pins should be tied to ground (GND).

When the DELAY\_TYPE is VARIABLE or VAR\_LOAD:

- If DELAY\_FORMAT = TIME, the attribute DELAY\_VALUE\_EXT for the cascaded delay must have a delay that is equal to the master DELAY\_VALUE attribute. For example, a total required delay of 1.5 ns is split between a 0.75 ns DELAY\_VALUE\_EXT and a 0.75 ns DELAY\_VALUE. After BISC completes, the master and cascaded delay lines can have different values.
- When configured in VAR\_LOAD mode, the input delay line and the extended output delay line tap delay values should be loaded for both components separately using LOAD and LOAD\_EXT with values set by CNTVALUEIN and CNTVALUEIN\_EXT, respectively. They can have different values.
- When configured in VARIABLE or VAR\_LOAD modes, the CE/INC and CE\_EXT/INC\_EXT must each be incremented/decremented. The functional requirements for controlling these signals are the same as described for non-cascaded mode.

**Note:** CASCADE = TRUE supports legacy design approaches that required the use of IDELAY to find the center of a UI for clock placement, and the amount of IDELAY can be up to 2.5 ns.



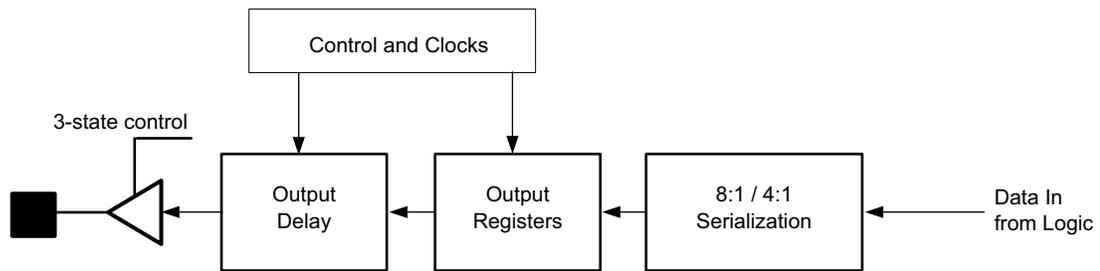
**RECOMMENDED:** For better signal integrity and more robust eye sampling at higher data rates, AMD strongly recommends the use of BISC and its QTR Delays for finding and maintaining the center of a UI and avoid using IDELAYs for sweeping.

**Note:** CASCADE = TRUE supports legacy designs where the use of IDELAY up to 2.5 ns is needed. Due to the nature of the cascaded delays and the additional delay taps, performance will degrade. Use the alignment associated with BISC and adjust the strobe clocks using PQTR/NQTR delay adjustments (RIU) for optimal performance.

## TX\_BITSLICE

The TX\_BITSLICE is the transmitter function of the RXTX\_BITSLICE.

Because the TX\_BITSLICE is the transmitter part of the RXTX\_BITSLICE, it contains an output delay that can continuously be corrected for VT variation by the BITSLICE\_CONTROL high-speed output serializing register and serialization logic for either 4:1 or 8:1. A block diagram of TX\_BITSLICE is shown in [Figure 2-50](#).



X16031-060916

Figure 2-50: TX\_BITSLICE Block Diagram

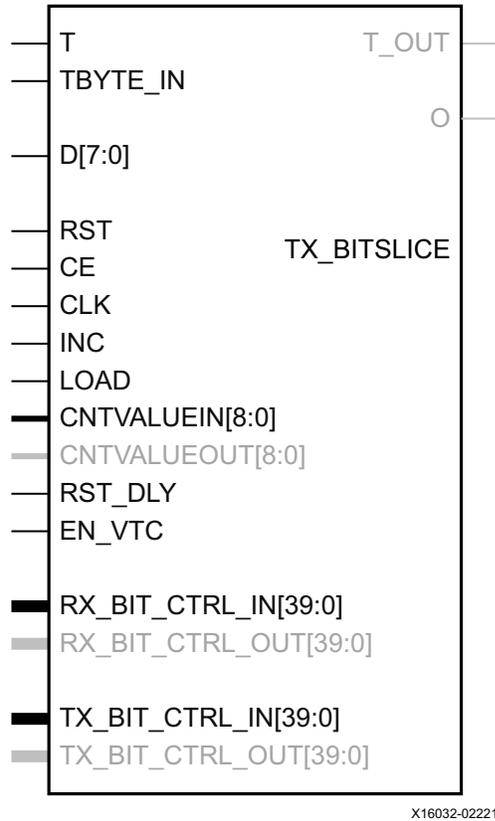


Figure 2-51: TX\_BITSLICE Primitive

**TX\_BITSLICE Function**

The function of the TX\_BITSLICE is discussed in depth in the earlier section [RXTX\\_BITSLICE Transmitter Function](#), page 215.

**TX\_BITSLICE Ports**

Table 2-26 lists the TX\_BITSLICE ports.

Table 2-26: TX\_BITSLICE Ports

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
D[7:0]	TX FPGA	Input	PLL_CLK (BITSlice_CONTROL)	Parallel incoming data from interconnect logic for transmit. Width is determined by the DATA_WIDTH attribute and can be either 8 or 4. If the DATA_WIDTH is 4, D[3:0] is used and D[7:4] should be tied to 0.

Table 2-26: TX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
T	TX FPGA	Input	Asynchronous	<p>T assigns a combinatorial path through the TX_BITSLICE to the 3-state pin of an output buffer.</p> <p>When the 3-state control is sourced from the interconnect logic, the T port must be used. Use of the T input of a bit slice can be seen as a block 3-state of the serial bitstream.</p> <p>Each TX_BITSLICE in a nibble has a T input, meaning that there are 13 T inputs for a byte (byte = two nibbles).</p> <p>A logic High means the output buffer is 3-stated and a logic Low means the output buffer is not 3-stated. Active-High.</p>
TBYTE_IN	TX FPGA	Input	PLL_CLK (BITSlice_CONTROL)	<p>The TBYTE_IN is 1-bit width wide input of the TX_BITSLICE side of the RXTX_BITSLICE. When using this 3-state, the TX_BITSLICE_TRI component must be used to serialize the TBYTE_IN[3:0] 3-state bus input of the BITSlice_CONTROL, giving the ability to 3-state individual bits in the serial output data stream. The TBYTE_IN[3:0] port of the BITSlice_CONTROL is handled and passes through the BITSlice_CONTROL to connect to the TX_BITSLICE_TRI. The TRI_OUT then connects to each TX_BITSLICE.TBYTE_IN input port in the nibble. When the BITSlice_CONTROL TBYTE_IN is High it means the output buffer is not 3-stated and a logic Low means the output buffer is 3-stated.</p>
RST	TX FPGA	Input	Asynchronous	<p>Resets the transmit side (TX_BITSLICE), asynchronous assertion and synchronous deassertion and is active-High. O resets to the INIT attribute value while RST is asserted.</p> <p>For deterministic bring-up, follow the steps in <a href="#">Native Mode Bring-up and Reset, page 296</a>.</p>

Table 2-26: TX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
CLK	TX FPGA	Input	Asynchronous	Delay line clock used to sample LOAD, CE, and INC. All control inputs to output delay line element within the TX part of the RXTX_BITSLICE are synchronous to the clock input (CLK). A clock must be connected to this port when the delay is configured in VARIABLE or VAR_LOAD. The CLK can be locally inverted, and must be supplied by a global clock buffer.
CE	TX FPGA	Input	CLK	Clock enable for the output delay line register clock. <b>Note:</b> Delays might take up to three clock cycles (CLK) to be applied. During this time, input data should not change to ensure output data does not glitch.
RST_DLY	TX FPGA	Input	Asynchronous (synchronous deassertion to CLK)	Reset port for the delay line within the transmitter logic. Resets the internal delay line to the value defined in the DELAY_VALUE attribute.
INC	TX FPGA	Input	CLK	The increment/decrement is controlled by the enable signal (CE). This interface is only available when the delay line is in VARIABLE or VAR_LOAD mode. As long as CE remains High, the delay line is incremented or decremented by one tap every clock (CLK) cycle. The state of INC determines whether the delay line is incremented or decremented: INC = 1 increments; INC = 0 decrements, synchronously to the clock (CLK). If CE is Low, the delay does not change (regardless of the state of INC). When CE goes High, the increment/decrement operation begins on the next positive clock edge. When CE goes Low, the increment/decrement operation ceases on the next positive clock edge. The programmable delay taps in the delay line primitive wrap around. When the last tap delay is reached (CNTVALUEOUT = 511), a subsequent increment function returns to tap 0. The same applies to the decrement function: decrementing from zero moves to tap 511.

Table 2-26: TX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
LOAD	TX FPGA	Input	CLK	<p>When in VAR_LOAD mode and UPDATE_MODE = ASYNC, this input loads the value set by the CNTVALUEIN into the delay line. The value present at CNTVALUEIN[8:0] is the new tap value. The LOAD signal is an active-High signal and is synchronous to the input clock signal (CLK). Wait at least one clock cycle after applying a new value on the CNTVALUEIN bus before applying the LOAD signal. The CE must be held Low during LOAD operation.</p> <p><b>Note:</b> Delays might take up to three clock cycles (CLK) to be applied. During this time, input data should not change to ensure output data does not glitch.</p>
EN_VTC	TX FPGA	Input	Asynchronous	<p>Enable voltage, temperature, and process compensation.</p> <p>High: Allows BITSlice_CONTROL to keep delay constant over VT. BITSlice_CONTROL.EN_VTC must be High for VT compensation to be enabled.</p> <p>Low: VT compensation is disabled.</p> <p>When TIME mode is used, the EN_VTC signal must be pulled High during initial built-in self-calibration (BISC).</p> <p>When used in COUNT mode, the EN_VTC signal must be pulled Low.</p> <p>When bit slices are used in both COUNT and TIME mode in a nibble, EN_VTC must be pulled High for the bit slices used in TIME mode, and pulled High or Low for those used in COUNT mode.</p>

Table 2-26: TX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
CNTVALUEIN[8:0]	TX FPGA	Input	CLK	The CNTVALUEIN bus is used for dynamically changing the loadable tap value. The 9-bit value at the CNTVALUEIN bus is the new tap value the delay line is set to after LOAD. Provide the value on this bus at least one clock cycle before LOAD. The delay line can be changed from 1 to 8 taps at a time. <b>Note:</b> When changing delays using the VT compensation using EN_VTC, only the programmed delay is compensated for. Applications requiring updated output delays to be compensated must use the RIU interface to program the input delays to match the output delays (see <a href="#">Table 2-48</a> and <a href="#">Table 2-49</a> ).
CNTVALUEOUT[8:0]	TX FPGA	Output	CLK	The CNTVALUEOUT pins are used for reporting the current tap value and reading out the amount of taps in the current delay. When EN_VTC is High, CNTVALUEOUT is updated by the BITSlice_CONTROL.
O	I/O TX	Output	PLL_CLK (BITSlice_CONTROL)	Serialized output data from the TX_BITSLICE that should be connected to the output buffer (or bidirectional buffer).
T_OUT	I/O TX	Output	PLL_CLK (when TBYTE_CTL set to TBYTE_IN) otherwise Asynchronous (BITSlice_CONTROL)	3-state output from the TX_BITSLICE that should be connected to the output buffer (or bidirectional buffer). Can be either the combinatorial output when TBYTE_CTL is set to T or the serialized output when TBYTE_CTL is set to TBYTE_IN.

The following RX/TX\_BIT\_CTRL\_OUT and RX/TX\_BIT\_CTRL\_IN pins are 40-bit bus connections between the RXTX\_BITSLICE (RX\_BITSLICE and/or TX\_BITSLICE) and the BITSlice\_CONTROL. Each of these 40-bit buses carries data, clocks, RIU, and status signals between the RXTX\_BITSLICE (RX\_BITSLICE, TX\_BITSLICE), TX\_BITSLICE\_TRI, and BITSlice\_CONTROL and vice versa.

When a bit slice is used, these buses **must** be connected to the appropriate BITSlice\_CONTROL input and output bus.

Example:

When RXTX\_BITSLICE\_2 is used, RX/TX\_BIT\_CTRL\_OUT of that RXTX\_BITSLICE must connect to the BITSlice\_CONTROL RX/TX\_BIT\_CTRL\_IN2, and the RX/TX\_BIT\_CTRL\_IN of the RXTX\_BITSLICE buses must connect to the BITSlice\_CONTROL RX/TX\_BIT\_CTRL\_OUT2 buses.

These buses are made of dedicated routing between the BITSlice\_CONTROL and bit slices and cannot be accessed or used by logic. It is also not possible to connect an ILA or VIO to these buses and viewing the buses in simulation is meaningless because the content and bit names of the buses is not disclosed.

Table 2-26: TX\_BITSLICE Ports (Cont'd)

Port	Function <sup>(1)</sup>	I/O	Synchronous Clock Domain	Description
RX_BIT_CTRL_IN[39:0]		Input	N/A	Input bus from BITSlice_CONTROL
RX_BIT_CTRL_OUT[39:0]		Output	N/A	Output bus to BITSlice_CONTROL
TX_BIT_CTRL_IN[39:0]		Input	N/A	Input bus from BITSlice_CONTROL
TX_BIT_CTRL_OUT[39:0]		Output	N/A	Output bus to BITSlice_CONTROL

**Notes:**

1. I/O RX: Connections between the RX\_BITSLICE side of the RXTX\_BITSLICE and the I/O buffers  
 I/O TX: Connections between the TX\_BITSLICE side of the RXTX\_BITSLICE and the I/O buffers  
 RX FPGA: Connections from/to the RX\_BITSLICE side of the RXTX\_BITSLICE and the logic  
 TX FPGA: Connections from/to the TX\_BITSLICE side of the RXTX\_BITSLICE and the logic

### TX\_BITSLICE Attributes

Table 2-27 lists the TX\_BITSLICE attributes.

Table 2-27: TX\_BITSLICE Attributes

Attributes	Values	Default	Type	Description
DATA_WIDTH	4 or 8	8	Decimal	Attribute defining the input width of the parallel-to-serial converter.  This value specifies the width the data requires to be serialized by the parallel-to-serial converter. Set DATA_WIDTH = 2 x BITSlice_CONTROLLER.DIV_MODE.
TBYTE_CTL	TBYTE_IN or T	TBYTE_IN	String	TBYTE_IN: The TBYTE_IN input is used to pass the 3-state information to the T_OUT output. It also requires that the TX_BITSLICE is used together with a TX_BITSLICE_TRI component.  T: The T input is used to pass the 3-state information to the T_OUT output. T requires that the 3-state information is generated in the interconnect logic. See the explanation in <a href="#">TX_BITSLICE_TRI</a> , page 273.
INIT	1'b0 or 1'b1	1'b1	Binary	Defines the initial value of the O port, which is the serialized data output of the TX_BITSLICE.

Table 2-27: TX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
DELAY_TYPE	FIXED VAR_LOAD VARIABLE	FIXED	String	Delay mode of the output delay line. For more information, see <a href="#">Native Output Delay Type Usage, page 220</a> .
DELAY_VALUE	0–1250 (TIME UltraScale) 0–1100 (TIME UltraScale+) 0–511 (COUNT)	0	Decimal	<p><b>Note:</b> For BISC to properly align, set RX_CLK_PHASE_P = RX_CLK_PHASE_N = SHIFT_0.</p> <p>When DELAY_FORMAT is set to TIME mode, the desired value is in ps.</p> <p>UltraScale devices support delays up to 1.25 ns. UltraScale+ devices support up to 1.1 ns.</p> <p>When DELAY_FORMAT is set to COUNT mode, the desired value is in number of taps. For more information, see <a href="#">Native Output Delay Type Usage, page 220</a>. To ensure TX_BITSLICE data alignment, limit COUNT delays to 1.5 UI.</p>
REFCLK_FREQUENCY	200.00–2400.00 (UltraScale) 300.00–2666.67 (UltraScale+)	300.0	1 significant digit float	<p>Specification of reference clock frequency in MHz.</p> <p>This is the frequency of the master_clock that the BITSlice_CONTROL is configured to use. It is used by BISC to calibrate any TIME mode delays. See <a href="#">Clocking in Native Mode, page 302</a> and <a href="#">Built-in Self-Calibration, page 319</a>. As opposed to previous FPGA families, the tap size is not determined by the REFCLK_FREQUENCY, the tap size is defined in the UltraScale device data sheets as <math>T_{ODELAY\_RESOLUTION}</math> [Ref 2] and the REFCLK_FREQUENCY attribute is used by BISC to calibrate the amount of taps to provide the requested delay when DELAY_FORMAT is set to TIME mode.</p>

Table 2-27: TX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
OUTPUT_PHASE_90	TRUE or FALSE	FALSE	String	<p>FALSE: Output O is not phase-shifted.</p> <p>TRUE: Output O is phase-shifted 90 degrees. DELAY_VALUE must be set to 0 when OUTPUT_PHASE_90 = TRUE.</p> <p>The phase shift can be observed when different transmitters are used. In most cases, it is used to shift the generated clock 90 degrees to the generated data (generated data and center-aligned clock).</p>
DELAY_FORMAT	TIME <sup>(1)</sup> COUNT	TIME	String	<p>DELAY_FORMAT can be either TIME or COUNT.</p> <p>When set to TIME, the delay after BISC completes (DLY_RDY goes High) equals the delay given in DELAY_VALUE (specified in ps).</p> <p>BISC uses the REFCLK_FREQUENCY attribute in conjunction with the incoming master clock to determine how many taps are required to achieve the requested TIME value (DELAY_VALUE). This calibration accounts for the process variation in the device. When EN_VTC is High, the delay is calibrated to provide the requested TIME across voltage and temperature.</p> <p>When DELAY_FORMAT is set to COUNT, the value given in DELAY_VALUE is the number of taps required. EN_VTC must be tied Low when using COUNT.</p>

Table 2-27: TX\_BITSLICE Attributes (Cont'd)

Attributes	Values	Default	Type	Description
UPDATE_MODE	ASYNC, SYNC, or MANUAL	ASYNC	String	<p>ASYNC: This is the default and preferred use method. Updates to the delay value are independent of the data being delayed. This mode is the preferred operation mode because it covers the function of both other modes.</p> <p>SYNC: Updates require data transitions to synchronously update the delay with the data edges. This mode is suitable for clocks or data that are always available and switches on a periodic basis.</p> <p>MANUAL: It takes two assertions of LOAD for the new value to take effect. The first LOAD loads the value defined by CNTVALUEIN and the second LOAD must be asserted with an assertion of the CE for the new value to take effect. This is beneficial because you can update the delay when the data becomes idle.</p>
ENABLE_PRE_EMPHASIS	TRUE FALSE	FALSE	String	<p>Used in conjunction with attributes on the bidirectional IOB to enable and disable pre-emphasis. The ENABLE_PRE_EMPHASIS attribute is used in conjunction with IOB to enable the pre-emphasis. See <a href="#">Transmitter Pre-Emphasis in Chapter 1</a>.</p>
IS_CLK_INVERTED	1'b0 or 1'b1	1'b0	Binary	<p>Similar to the <a href="#">IS_RST_INVERTED</a> attribute, but on the CLK path.</p> <p>When IS_CLK_INVERTED = 1, the inverter is used to reverse polarity (invert) the CLK signal.</p> <p>When IS_CLK_INVERTED = 0, the inverter is not used.</p>

Table 2-27: TX\_BITSLICE Attributes (Cont'd)

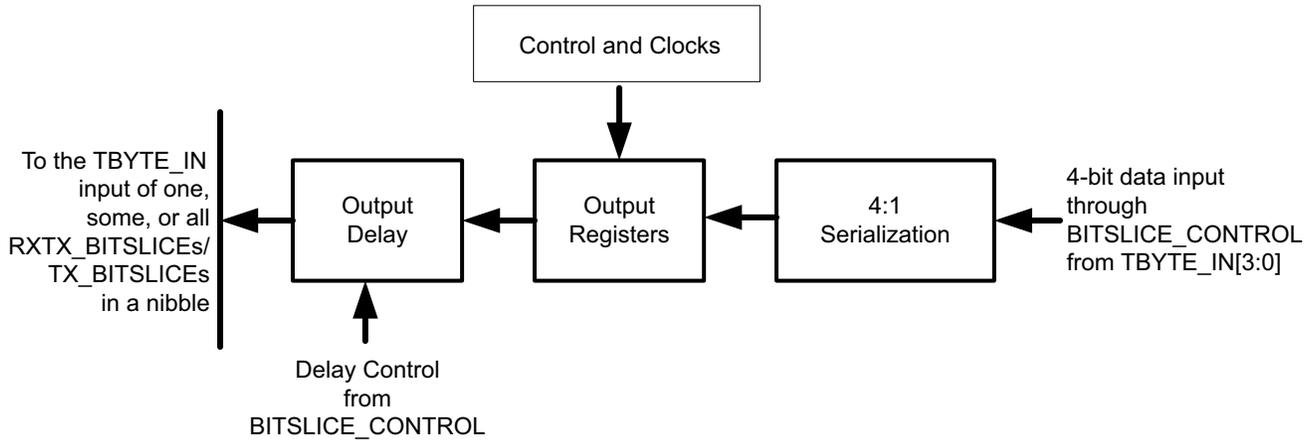
Attributes	Values	Default	Type	Description
IS_RST_DLY_INVERTED	1'b0 or 1'b1	1'b0	Binary	Similar to the <a href="#">IS_RST_INVERTED</a> attribute but on the RST_DLY path. When IS_RST_DLY_INVERTED = 1, the inverter is used to reverse polarity (invert) the RST_DLY signal. When IS_RST_DLY_INVERTED = 0, the inverter is not used.
IS_RST_INVERTED	1'b0 or 1'b1	1'b0	Binary	A selectable local inverter on the reset path can be used to change the polarity of the reset input. When IS_RST_INVERTED = 1, the inverter is used to reverse the polarity (invert) the RST signal. When IS_RST_INVERTED = 0, the inverter is not used. See <a href="#">Figure 2-46</a> .
NATIVE_ODELAY_BYPASS	TRUE or FALSE	FALSE	String	UltraScale+ FPGAs only: Reserved for memory interface generator (MIG). When TRUE, bypass the ODELAY.
SIM_DEVICE	Possible Values: ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2	ULTRASC ALE	String	Sets the device version (ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2)

**Notes:**

1. When in TIME mode, calibration affects the availability of bit slices within the nibble. See [Bank Overview](#) for more information.

## TX\_BITSLICE\_TRI

The TX\_BITSLICE\_TRI is in all respects a bit slice like the TX\_BITSLICE. As the TX\_BITSLICE, it contains an output delay that can continuously be corrected for VT variation by the BITSlice\_CONTROL, high-speed output serializing register and serialization logic for 4:1 data, but it does not have a direct user-accessible parallel data input nor does it have a serial output with access to FPGA pins. The input for this primitive comes through the BITSlice\_CONTROL primitive from the 4-bit TBYTE\_IN bus, so this bit slice is buried inside a nibble. A block diagram of TX\_BITSLICE\_TRI is shown in [Figure 2-52](#).

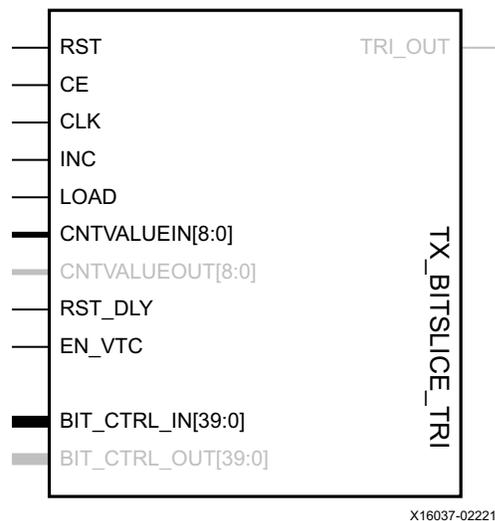


X16351-060916

Figure 2-52: TX\_BITSLICE\_TRI Block Diagram

The TX\_BITSLICE\_TRI can only be used to 3-state bit slices within a nibble. The following [TX\\_BITSLICE\\_TRI Function](#) section shows how the TX\_BITSLICE\_TRI is connected between the BITSlice\_CONTROL and TX\_BITSLICES of a nibble.

The four bits from the BITSlice\_CONTROL are serialized and possibly delayed and fed to and through the TX\_BITSLICE to the 3-state of an output buffer in the IOB. This mechanism provides the ability to 3-state single bits in a serial output stream. The waveform in [Figure 2-55](#) shows the relationship of the TBYTE\_IN input of the BITSlice\_CONTROL to the TX\_BITSLICE.O output and IOB 3-state buffer.



X16037-022216

Figure 2-53: TX\_BITSLICE\_TRI Primitive

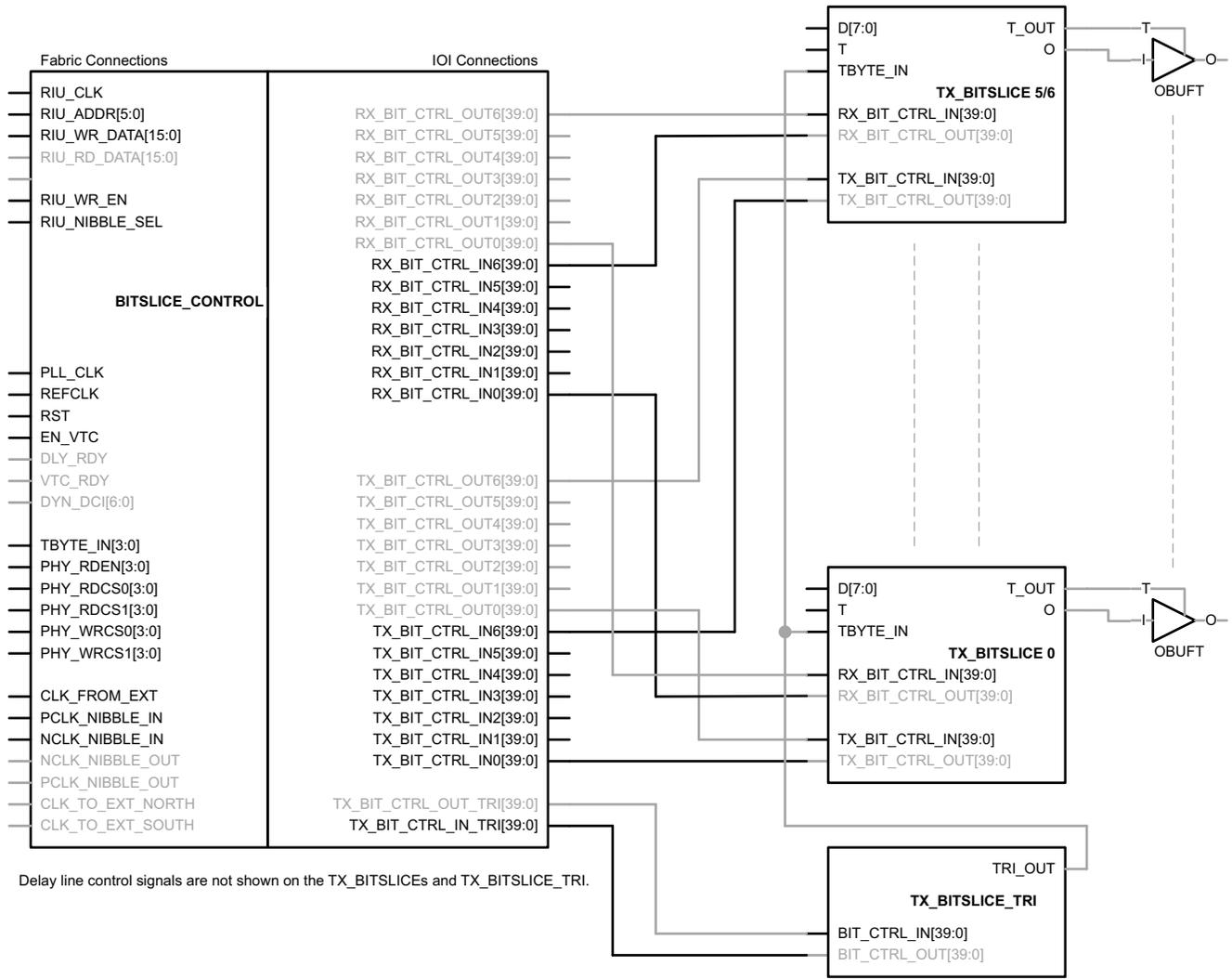
### ***TX\_BITSLICE\_TRI Function***

As mentioned, a TX\_BITSLICE\_TRI is a TX\_BITSLICE but without user data input and serial output. As such, follow the explanation of the [RXTX\\_BITSLICE Transmitter Function, page 215](#) to understand the function of this TX\_BITSLICE\_TRI.

3-state is mostly used for bidirectional data and/or clock/strobe applications. The TX\_BITSLICE\_TRI is only used when the RXTX\_BITSLICE/TX\_BITSLICE attribute TBYTE\_CTL is set to TBYTE\_IN. In that case, the BITSlice\_CONTROL, TX\_BITSLICE\_TRI, and one or more TX\_BITSLICES work together to 3-state dedicated bits in a set of serial data output streams. [Figure 2-54](#) shows how the primitives are interconnected and the waveform in [Figure 2-55](#) shows how signals must be applied in order to 3-state bits in a serial data stream.

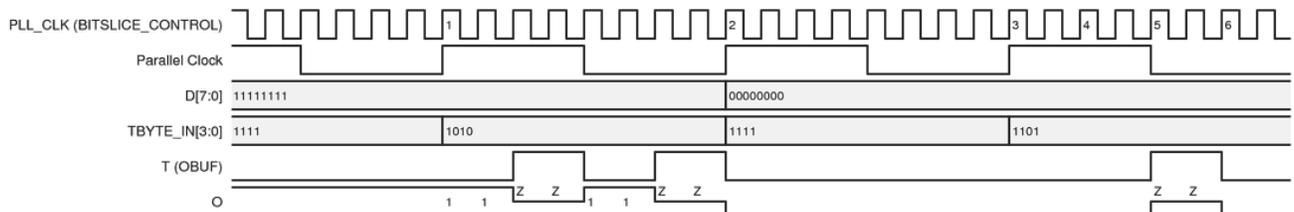
- When the TBYTE\_CTL attribute is set to TBYTE\_IN, the TBYTE\_IN[3:0] inputs of the BITSlice\_CONTROL primitive control the 3-state of all RXTX\_BITSLICES in a nibble. As shown in [Figure 2-56](#), the TBYTE\_IN[3:0] inputs control all nibble output 3-state functions (through the TX\_BITSLICE\_TRI). By using TBYTEIN[3:0] inputs, it is possible to 3-state a single bit in a serial stream.
- When the TBYTE\_CTL attribute is set to T, TX\_BITSLICE\_TRI is not needed and the TBYTE\_IN[3:0] pins can be deasserted Low ([Figure 2-56](#)). When the TBYTE\_CTL attribute is set to T, the 3-state function for that RXTX\_BITSLICE is controlled from interconnect logic. Controlling the 3-state of a RXTX\_BITSLICE from interconnect logic means that it operates as a block, word, or frame 3-state. As shown in [Figure 2-56](#), it is possible to mix TX\_BITSLICES in a nibble with TBYTE\_CTL set to TBYTE\_IN and T. When a nibble has a mix of TBYTE\_IN and T, alignment of serialized data across the bit slices is not guaranteed.

[Figure 2-54](#) shows the required connections when connecting 3-state control using the TX\_BITSLICE\_TRI and T\_BYTE\_IN[3:0] of TX\_BITSLICE.



X16038-022216

Figure 2-54: Connections for a 3-State Path when Using the TBYTE Port



X16039-072516

Figure 2-55: Connections for a 3-State Path when Using TBYTE (DATA\_WIDTH=8)

Notes on Figure 2-55:

- For easier viewing, latency is not shown.



The latency for the TX\_BITSLICE\_TRI is shown in [Figure 2-57](#) and [Figure 2-58](#).

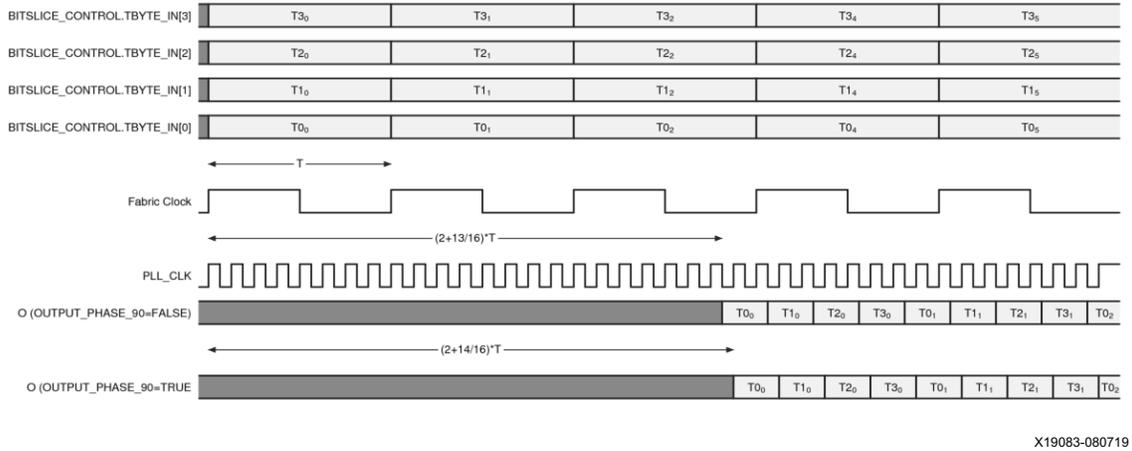


Figure 2-57: 3-State Latency, DATA\_WIDTH = 8

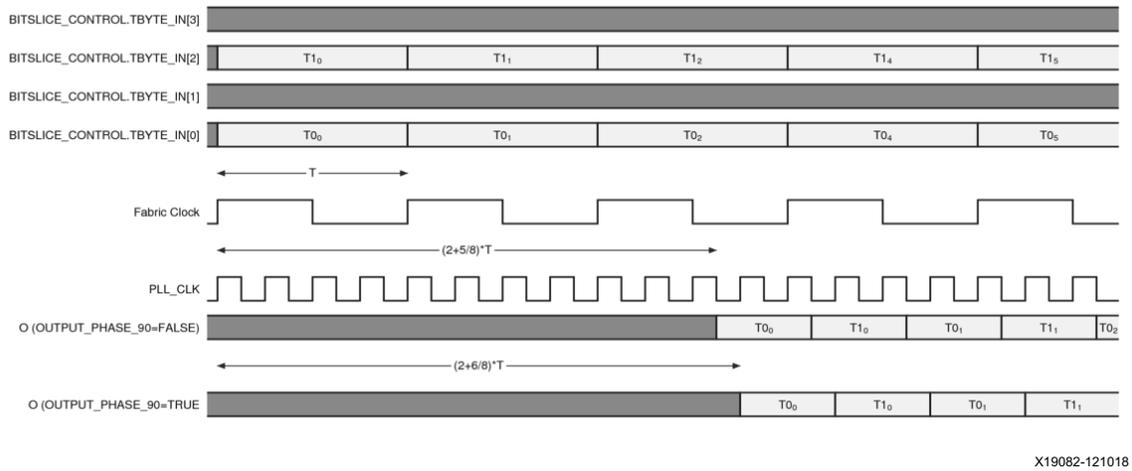


Figure 2-58: 3-State Latency, DATA\_WIDTH = 4

## TX\_BITSLICE\_TRI Ports

Table 2-28 lists the TX\_BITSLICE\_TRI ports.

Table 2-28: TX\_BITSLICE\_TRI Port Descriptions

Port	I/O	Description
RST	Input	Resets the 3-state serialization logic, asynchronous assertion and synchronous deassertion and is active-High. Q resets to zero while RST is asserted. For deterministic bring-up, follow the steps in <a href="#">Native Mode Bring-up and Reset</a> , page 296.
CE	Input	Clock enable for the 3-state delay line register clock.
CLK	Input	Clock input. All control inputs to the DELAY element within the TX_BITSLICE_TRI (LOAD, CE, and INC) are synchronous to this clock input. A clock must be connected to this port when DELAY is configured in VARIABLE or VAR_LOAD. This signal can be locally inverted, and must be supplied by a global or regional clock buffer.  The clock signal connected to this pin must be the same clock signal as the one connected to the RX_CLK and/or CLK of a RXTX_BITSLICE/RX_BITSLICE.
INC	Input	The increment/decrement is controlled by the enable signal (CE). This interface is only available when the delay line is in VARIABLE or VAR_LOAD mode.  As long as CE remains High, the delay line is incremented or decremented by one tap every clock (CLK) cycle. The state of INC determines whether delay line is incremented or decremented: INC = 1 increments; INC = 0 decrements, synchronously to the clock (CLK).  If CE is Low, the delay through the delay line does not change (regardless of the state of INC). When CE goes High, the increment/decrement operation begins on the next positive clock edge. When CE goes Low, the increment/decrement operation ceases on the next positive clock edge.  The programmable delay taps in the delay line primitive wrap around. When the last tap delay is reached (CNTVALUEOUT = 511), a subsequent increment function returns to tap 0. The same applies to the decrement function: decrementing from zero moves to tap 511.
LOAD	Input	When in VAR_LOAD mode, this input loads the value set by the CNTVALUEIN attribute into the delay line. The value present at CNTVALUEIN[8:0] is the new tap value. The LOAD signal is an active-High signal and is synchronous to the input clock signal (CLK). Wait at least one clock cycle after applying a new value on the CNTVALUEIN bus before applying the LOAD signal. The CE must be held Low during LOAD operation.
CNTVALUEIN[8:0]	Input	The CNTVALUEIN bus is used to dynamically change the loadable tap value. The 9-bit value at the CNTVALUEIN is the number of taps required. The new value is to be presented one CLK cycle before LOAD is pulsed High. New CNTVALUEIN values should only be applied when EN_VTC is Low.
CNTVALUEOUT[8:0]	Output	The CNTVALUEOUT pins are used for reporting the current tap value. CNTVALUEOUT should only be sampled when EN_VTC is Low.
RST_DLY	Input	Resets the delay line taps setting to the value provided by the DELAY_VALUE attribute.  Reset port for the delay line in the TX_BITSLICE_TRI.

Table 2-28: TX\_BITSLICE\_TRI Port Descriptions (Cont'd)

Port	I/O	Description
EN_VTC	Input	Enable Voltage Temperature calibration. High: Allows BITSlice_CONTROL to keep delay constant over VT. BITSlice_CONTROL.EN_VTC must be held High for VT compensation to be enabled. Low: VT compensation is disabled. When TIME mode is used, the EN_VTC signal must be pulled High during initial BISC. When COUNT mode is used, the EN_VTC signal must be pulled Low.
BIT_CTRL_IN[39:0]	Input	Input bus from BITSlice_CONTROL. Dedicated pins that must connect directly between the BITSlice_CONTROL and TX_BITSLICE_TRI and to nothing else in the design.
BIT_CTRL_OUT[39:0]	Output	Output bus to BITSlice_CONTROL. Dedicated pins that must connect directly between the BITSlice_CONTROL and TX_BITSLICE_TRI and to nothing else in the design.
TRI_OUT	Output	3-state output (TRI_OUT) outputs to the TBYTE_IN pins of the bit slices.

### TX\_BITSLICE\_TRI Attributes

Table 2-29: TX\_BITSLICE\_TRI Attributes

Attributes	Values	Default	Type	Description
DATA_WIDTH	4, 8	8	Decimal	Attribute defining the input width of the parallel-to-serial converter. This specifies the width the data needs to have to be serialized by the parallel-to-serial converter. This value must match RXTX_BITSLICE/TX_BITSLICE DATA_WIDTH.

Table 2-29: TX\_BITSLICE\_TRI Attributes (Cont'd)

Attributes	Values	Default	Type	Description
DELAY_FORMAT	TIME <sup>(1)</sup> , COUNT	TIME	String	<p>DELAY_FORMAT can be either TIME or COUNT.</p> <p>When set to TIME, the delay after BISC completes (DLY_RDY goes High) equals the delay given in DELAY_VALUE (specified in ps).</p> <p>BISC uses the REFCLK_FREQUENCY attribute in conjunction with the incoming master clock to determine the current tap size and therefore how many taps are required to achieve the requested TIME value (DELAY_VALUE). This calibration accounts for the process variation in the device. When EN_VTC is High, the delay is calibrated to provide the requested TIME across voltage and temperature.</p> <p>When DELAY_FORMAT is set to COUNT, the value given in DELAY_VALUE is the number of taps required. EN_VTC must be tied Low when using COUNT.</p>
DELAY_TYPE	FIXED, VAR_LOAD, VARIABLE	FIXED	String	Delay mode of the input delay line.
DELAY_VALUE	0–1250 (TIME UltraScale) 0–1100 (TIME UltraScale+) 0–511 (COUNT)	0	Decimal	<p>TIME mode: Desired value in ps.</p> <p>UltraScale devices support delays up to 1.25 ns.</p> <p>UltraScale+ devices support up to 1.1 ns.</p> <p>COUNT mode: Desired value in taps.</p>

Table 2-29: TX\_BITSLICE\_TRI Attributes (Cont'd)

Attributes	Values	Default	Type	Description
UPDATE_MODE	ASYNC MANUAL SYNC	ASYNC	String	<p>ASYNC: Updates to the delay value are independent of the data being received. This mode is the preferred operation mode because it covers the function of both other modes.</p> <p>SYNC: Updates require DATAIN transitions to synchronously update the delay with the DATAIN edges. This mode is suitable for clocks or data that are always available and switches on a periodic basis.</p> <p>MANUAL: It takes two assertions of LOAD for the new value to take effect. The first LOAD loads the value defined by CNTVALUEIN and the second LOAD must be asserted with an assertion of the CE for the new value to take effect. This is beneficial because you can update the delay when the data becomes idle.</p>
INIT	1'b1, 1'b0	1'b1	Binary	Defines the initial value of the O port which is the serialized data output of the TX_BITSLICE_TRI.
OUTPUT_PHASE_90	TRUE or FALSE	FALSE	String	<p>The output phase can be chosen to be either 0 or 90 degrees.</p> <p>DELAY_VALUE must be set to 0 when OUTPUT_PHASE_90 = TRUE.</p>

Table 2-29: TX\_BITSLICE\_TRI Attributes (Cont'd)

Attributes	Values	Default	Type	Description
REFCLK_FREQUENCY	200.00–2400.00 (UltraScale) 300.00–2666.67 (UltraScale+)	300.0	1 significant digit float	<p>Specification of reference clock frequency in MHz.</p> <p>The reference clock is the master_clock (PLL_CLK) connected to the BITSLICE_CONTROL. This attribute is used in the BISC to calibrate any TIME mode delays.</p> <p>See <a href="#">Clocking in Native Mode, page 302</a> and <a href="#">Built-in Self-Calibration, page 319</a> in the <a href="#">BITSLICE_CONTROL</a> section.</p> <p>The tap size is not determined by the REFCLK_FREQUENCY.</p> <p>A tap delay range is specified in the UltraScale device data sheets as <math>T_{IDELAY\_RESOLUTION}</math> [Ref 2]. The REFCLK_FREQUENCY attribute is used by the BISC algorithm to calculate the number of taps required for the requested DELAY_VALUE.</p>
IS_CLK_INVERTED	1'b0,1'b1	1'b0	Binary	<p>Specifies whether the CLK pin is active-High or active-Low.</p> <p>Similar to the IS_RST_INVERTED attribute but on the CLK path.</p> <p>When IS_CLK_INVERTED = 1, the inverter is used.</p> <p>When IS_CLK_INVERTED = 0, the inverter is not used.</p>

Table 2-29: TX\_BITSLICE\_TRI Attributes (Cont'd)

Attributes	Values	Default	Type	Description
IS_RST_DLY_INVERTED	1'b0,1'b1	1'b0	Binary	Specifies whether the reset RST_DLY pin is active-High or active-Low. Similar to the IS_RST_INVERTED attribute but on the RST_DLY path. When IS_RST_DLY_INVERTED = 1, the inverter is used. When IS_RST_DLY_INVERTED = 0, the inverter is not used.
IS_RST_INVERTED	1'b0,1'b1	1'b0	Binary	Specifies whether the reset RST pin is active-High or active-Low. There is a selectable local inverter on the reset path that can be used to change the polarity of the reset input. When IS_RST_INVERTED = 1, the inverter is used. When IS_RST_INVERTED = 0, the inverter is not used.
NATIVE_ODELAY_BYPASS	TRUE or FALSE	FALSE	String	UltraScale+ FPGAs only: Reserved for memory interface generator (MIG). When TRUE, bypass the ODELAY.
SIM_DEVICE	Possible Values: ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2	ULTRASCALE	String	Sets the device version (ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2)

**Notes:**

1. When in TIME mode, calibration affects the availability of bit slices within the nibble. See [Bank Overview](#) for more information.

## BITSLICE\_CONTROL

The data and clock/strobe base handling block are the RXTX\_BITSLICE (RX\_BITSLICE and TX\_BITSLICE primitives are derived from this component), which are used per pin or pin pair. The six or seven bit slices within a nibble are all controlled by one BITSLICE\_CONTROL block,

as shown in Figure 2-59. Seven bit slices make an upper nibble and six bit slices assemble a lower nibble.

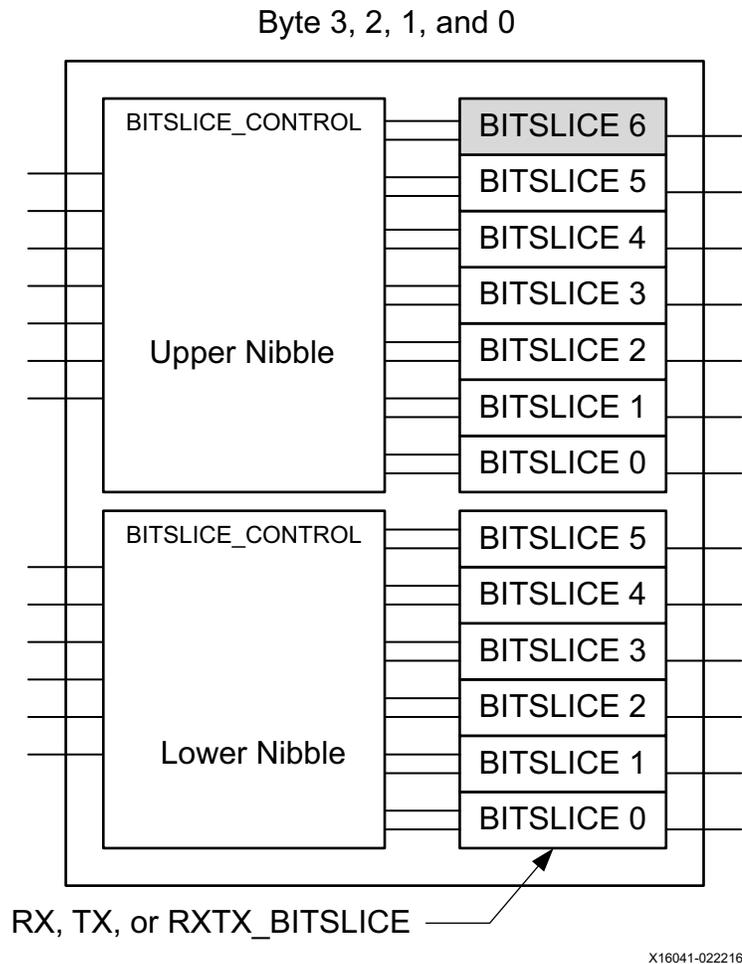
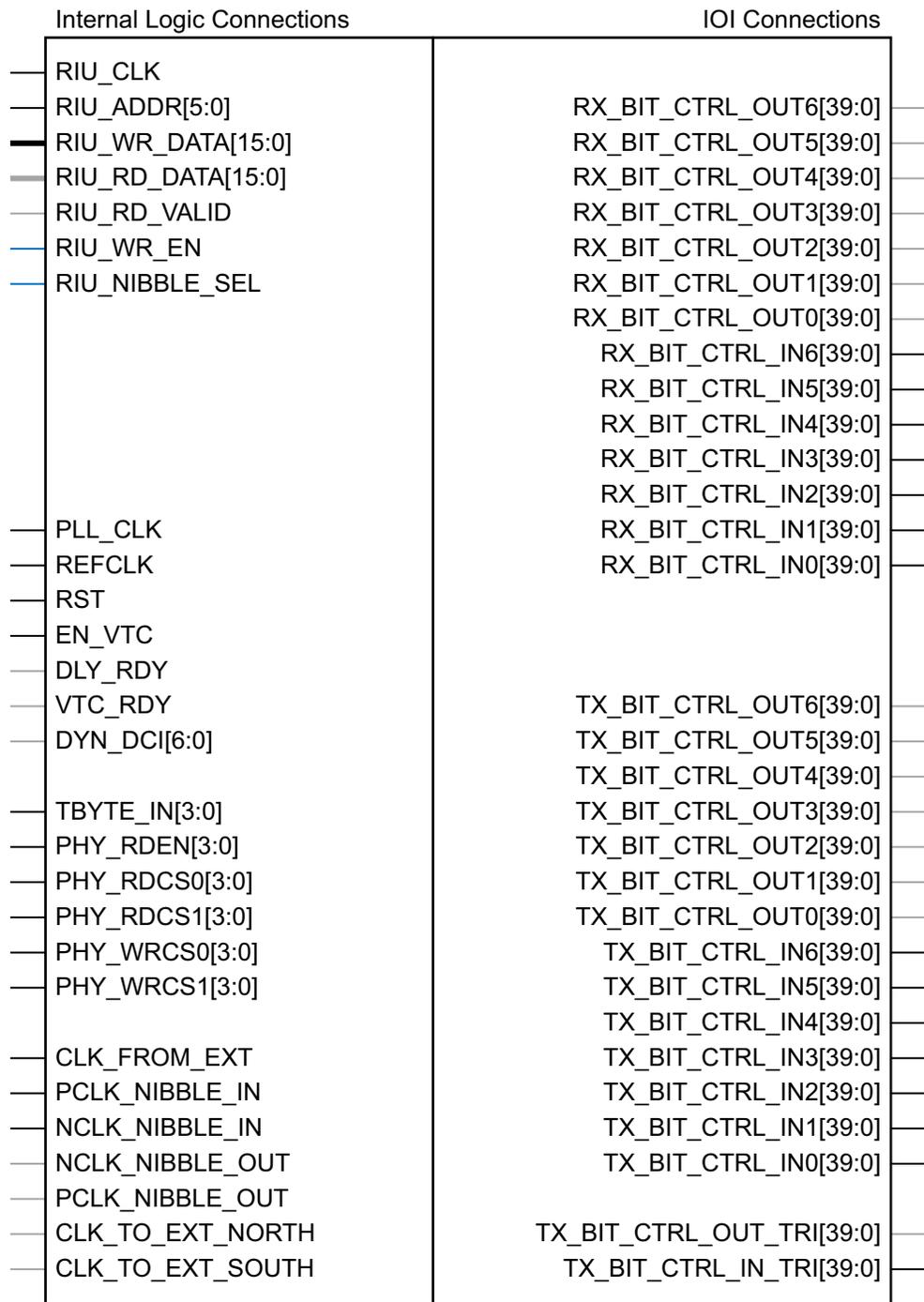


Figure 2-59: BITSLICE\_CONTROL with Respect to Bit Slices

Base functions of the BITSLICE\_CONTROL primitive (Figure 2-60) are to perform built-in self-calibration (BISC), generate clocks for the receiver and transmitter functions in the RXTX\_BITSLICES, control specialized functions such as RX\_ and/or TX\_GATING, and control a set of registers (RIUs) used by the previous summed functions. Each of these functions is discussed separately later in this document. Pins and attributes allow a fair amount of control of the BITSLICE\_CONTROL component, however, full control is obtained through a register interface unit (RIU). The RIU makes the BITSLICE\_CONTROL act as a processor peripheral and gives access to a set of sixty-four 16-bit registers providing access to all the required delay and control values for the nibble group being programmed.



X16040-022216

Figure 2-60: BITSlice\_CONTROL Primitive

Two nibbles can be combined into a byte. A byte contains two BITSlice\_CONTROL components, each having an RIU interface. Both RIU interfaces can be combined using an RIU\_OR component. When the RIU interfaces of both BITSlice\_CONTROLS are combined using a RIU\_OR primitive it looks like a single RIU interface to the interconnect logic.

These aspects of the BITSlice\_CONTROL primitive are discussed later in this chapter:

- [Native Mode Bring-up and Reset, page 296](#)
- [Clocking in Native Mode, page 302](#)
- [Built-in Self-Calibration, page 319](#)
- [Register Interface Unit \(RIU\), page 323](#)

### ***BITSlice\_CONTROL Ports***

Table 2-30 lists the BITSlice\_CONTROL ports.

Table 2-30: **BITSlice\_CONTROL Ports**

Port	I/O	Synchronous Clock Domain	Description
PLL_CLK	Input	Asynchronous	<p>Master clock input for the BITSlice_CONTROL. Set REFCLK_SRC attribute = PLL_CLK.</p> <p>This clock is used by the BISC controller. When SERIAL_MODE = TRUE, it is also used for data and strobe/clock sample clock.</p> <p>This clock must come from one of the two PLLs in the I/O bank where the BITSlice_CONTROL port carrying these pins is located.</p> <p>The PLL connects to the PLL_CLK pin over dedicated, very low jitter routing.</p> <p>Use this PLL_CLK clock input, or the REFCLK clock input, but not both. When the PLL_CLK is used, tie the REFCLK Low.</p>
REFCLK	Input	Asynchronous	<p>Master clock input for the BITSlice_CONTROL. Set REFCLK_SRC attribute = REFCLK. REFCLK is only supported for RX_BITSlice.</p> <p>This clock is used by the BISC controller. When SERIAL_MODE = TRUE, it is also used for data and strobe/clock sample clock.</p> <p>This clock can be generated by a MMCM in the internal logic.</p> <p>Connections to this clock input use clock buffers and route over normal clock routing in the FPGA.</p> <p>Use this REFCLK clock input, or the PLL_CLK clock input, but not both. When the REFCLK is used, tie the PLL_CLK Low.</p> <p>It is recommended to use the PLL_CLK input of the BITSlice_CONTROL. The master clock has very low jitter because it is generated by a PLL.</p>

Table 2-30: BITSlice\_CONTROL Ports (Cont'd)

Port	I/O	Synchronous Clock Domain	Description
RST	Input	Asynchronous	<p>Asynchronous asserted global reset.</p> <p>This reset is best synchronously released, following a dedicated reset sequence.</p> <p>Read the <a href="#">Native Mode Bring-up and Reset, page 296</a> section for more information.</p> <p><b>Note:</b> The reset for all used IDELAYCTRLs/BITSlice_CONTROLS within a bank must be released at the same time due to the cascaded DLY_RDY connections between the BITSlice_CONTROLS. Failure to do so might result in DLY_RDY for one of the IDELAYCTRLs/BITSlice_CONTROLS not asserting.</p>
EN_VTC	Input	RIU_CLK	<p>Enable voltage and temperature control and tracking.</p> <p>Assertion of EN_VTC maintains the delay of the delay lines that are in TIME mode over the V and T changes.</p> <p>After DLY_RDY goes High and initial BISC completion, the EN_VTC signal must be pulled High.</p> <p>There are also EN_VTC pins on the bit slices. For BISC to compensate the delays over VT, the BITSlice.EN_VTC must be held High.</p>
DLY_RDY	Output	Asynchronous	<p>Status bit indicating when BISC finishes initial fixed delay line calibration.</p> <p>This pin is also represented by a RIU register bit.</p>
VTC_RDY	Output	Asynchronous	<p>Status signal indicating when BISC finishes baseline VT calibration and tracking.</p> <p>From now on, BISC continuously compensates the delay lines for voltage and temperature.</p> <p>After asserted, this signal stays High until a hardware reset of the BITSlice_CONTROL or its EN_VTC is toggled Low.</p> <p>This pin is also represented by a RIU register bit. In Component mode, the IDELAYCTRL.RDY signal is the equivalent of this pin.</p>
RIU_CLK	Input	Asynchronous	<p>Clock for the RIU interface peripheral.</p> <p>This clock is independent from all other BITSlice_CONTROL clocks.</p> <p>This clock can be generated by an MMCM or PLL.</p>
RIU_ADDR[5:0]	Input	RIU_CLK	<p>The address input bus provides a register address for the register interface.</p> <p>The address value on this bus specifies the configuration and status bits that are written or read with the next RIU_CLK cycle. When not used, all bits must be assigned zeros.</p>

Table 2-30: BITSlice\_CONTROL Ports (Cont'd)

Port	I/O	Synchronous Clock Domain	Description
RIU_WR_DATA [15:0]	Input	RIU_CLK	<p>This input bus provides data. The value of this bus is written to the register address selected by RIU_ADDR of the register interface. The data is presented in the cycle that RIU_WR_EN and RIU_NIBBLE_SEL are active. The data is captured in a shadow register and written at a later time.</p> <p>RIU_VALID indicates when the RIU port is ready to accept another write. When not used, all bits must be set to zero.</p>
RIU_RD_DATA [15:0]	Output	RIU_CLK	<p>This output bus provides RIU data to the internal logic. The value of this bus is a representation of the register bits addressed by RIU_ADDR. The data is presented in the next cycle when RIU_WR_EN is Low and RIU_NIBBLE_SEL is High, sampled by RIU. For a complete listing of RIU_RD_DATA information, see <a href="#">Register Definitions and Addresses, page 329</a>.</p>
RIU_VALID	Output	RIU_CLK	<p>This signal indicates the status when RIU accesses are made from interconnect logic while the internal BISC state machines are also accessing the RIU registers. During a collision (i.e., an RIU write access from interconnect occurs during a BISC write access), the RIU_VALID signal deasserts. The internal logic write access still succeeds but not until RIU_VALID is asserted. No further action is required from interconnect logic except that no further RIU accesses are possible until RIU_VALID is deasserted High. In addition to collisions, the RIU_VALID asserts when writing to the RL_DLY_RNK[0, 1, 2, 3] registers. These registers are unique because it takes more than two cycles for an RIU write to update them. Therefore, back-to-back accesses to these registers are impossible.</p>
RIU_WR_EN	Input	RIU_CLK	Signal must be High to write a register in an RIU interface.
RIU_NIBBLE_SEL	Input	RIU_CLK	Signal is used to select a nibble RIU in a byte. Must be High to perform write or read.
PHY_RDCS0 [3:0] PHY_RDCS1 [3:0]	Input	PLL_CLK	Memory interface generator (MIG) use only: Rank select.
PHY_WRCS0 [3:0] PHY_WRCS1 [3:0]	Output	PLL_CLK	

Table 2-30: BITSlice\_CONTROL Ports (Cont'd)

Port	I/O	Synchronous Clock Domain	Description
TBYTE_IN[3:0]	Input	PLL_CLK	<p>Nibble/byte group 3-state input.</p> <p>When this input is used, a TX_BITSLICE_TRI primitive must be instantiated and connected to the TX_BIT_CTRL_OUT(IN)_TRI[39:0] buses, and TX_GATING must be set to ENABLE.</p> <p>The nibble provided here is passed through the BITSlice_CONTROL to the TX_BITSLICE_TRI primitive where the bits are serialized and delayed when the output delay line is used. The serial output of the TX_BITSLICE_TRI is passed to the single bit TBYTE_IN input of all used TX_BITSLICES.</p> <p>Read more about this input in the <a href="#">TX_BITSLICE, page 263</a> and <a href="#">TX_BITSLICE_TRI, page 273</a> sections.</p>
PHY_RDEN[3:0]	Input	PLL_CLK	<p>Read enable.</p> <p>Must be tied to 1111 when the RX_GATING attribute is not used.</p>
DYN_DCI[6:0]	Output	Asynchronous	MIG USE ONLY: Direct IOB DCI control.
<p>The following ports are dedicated clock inputs and outputs between two BITSlice_CONTROL components of the same byte or between bytes. The clock routing possibilities are enabled through the setting of attributes. For a discussion about the clocking possibilities between nibbles (inter-nibble) or between bytes (inter-byte) read the <a href="#">Clocking in Native Mode, page 302</a>.</p>			
CLK_FROM_EXT	Input	Asynchronous	Inter-byte clock coming from a neighboring byte BITSlice_CONTROL CLK_TO_EXT_NORTH or CLK_TO_EXT_SOUTH output. When no inter-byte clocking is used or only the CLK_TO_EXT_ pins are used, this pin must be pulled High.
CLK_TO_EXT_NORTH	Output	Asynchronous	Inter-byte clock to the CLK_FROM_EXT input of a neighboring byte BITSlice_CONTROL block located above (north) of this output. Use of this pin is enabled by the EN_CLK_TO_EXT_NORTH attribute.
CLK_TO_EXT_SOUTH	Output	Asynchronous	Inter-byte clock to the CLK_FROM_EXT input of a neighboring byte BITSlice_CONTROL block located below (south) of this output. Use of this pin is enabled by the EN_CLK_TO_EXT_SOUTH attribute.
PCLK_NIBBLE_IN	Input	Asynchronous	<p>Inter-nibble strobe/clock from the other BITSlice_CONTROL in the byte.</p> <p>Each byte contains two nibbles and each nibble has a PCLK_NIBBLE_IN input.</p> <p>Use of this input is enabled by the EN_OTHER_PCLK attribute.</p>

Table 2-30: BITSlice\_CONTROL Ports (Cont'd)

Port	I/O	Synchronous Clock Domain	Description
NCLK_NIBBLE_IN	Input	Asynchronous	Inter-nibble strobe/clock from the other BITSlice_CONTROL in the byte. Each byte contains two nibbles and each nibble has a NCLK_NIBBLE_IN input. Use of this input is enabled by the EN_OTHER_NCLK attribute.
PCLK_NIBBLE_OUT	Output	Asynchronous	Inter-nibble strobe/clock to the other BITSlice_CONTROL in the byte. Each byte contains two nibbles and each nibble has a PCLK_NIBBLE_OUT output. This signal must be connected to PCLK_NIBBLE_IN input of another nibble in the byte.
NCLK_NIBBLE_OUT	Output	Asynchronous	Inter-nibble strobe/clock to the other BITSlice_CONTROL in the byte. Each byte contains two nibbles and each nibble has a NCLK_NIBBLE_OUT output. This signal must be connected to a NCLK_NIBBLE_IN input of another nibble in the byte.
<p>The following RX/TX_BIT_CTRL_OUT and RX/TX_BIT_CTRL_IN pins are 40-bit bus connections between the BITSlice_CONTROL and RXTX_BITSLICE, RX_BITSLICE, or TX_BITSLICE used. Each of these 40-bit buses carries data, clocks, RIU, and status signals between the BITSlice_CONTROL and bit slices.</p> <p>When an RXTX_BITSLICE, RX_BITSLICE, or TX_BITSLICE is used, these buses <b>must</b> be connected to the appropriate BITSlice_CONTROL input and output bus (Figure 2-52).</p> <p>Example:</p> <p>When RX_BITSLICE_0 is used, RX/TX_BIT_CTRL_OUT must connect to the BITSlice_CONTROL RX/TX_BIT_CTRL_IN0 and the RX/TX_BIT_CTRL_IN buses must connect to the BITSlice_CONTROL RX/TX_BIT_CTRL_OUT0 buses. These buses are made of dedicated routing between the BITSlice_CONTROL and bit slices.</p>			
RX_BIT_CTRL_OUTx[39:0]	Output	N/A	Output bus connected to the RX_BIT_CTRL_IN from the bit slice.
RX_BIT_CTRL_INx[39:0]	Input	N/A	Input bus connected to the RX_BIT_CTRL_OUT from the bit slice.
TX_BIT_CTRL_OUTx[39:0]	Output	N/A	Output bus connected to the TX_BIT_CTRL_IN from the bit slice.
TX_BIT_CTRL_INx[39:0]	Input	N/A	Input bus connected to the TX_BIT_CTRL_OUT from the bit slice.
TX_BIT_CTRL_OUT_TRI[39:0]	Output	N/A	Output bus to the TX_BITSLICE_TRI. TX_BIT_CTRL_IN input bus.
TX_BIT_CTRL_IN_TRI[39:0]	Input	N/A	Input bus from the TX_BITSLICE_TRI. TX_BIT_CTRL_OUT output bus.

### **BITSLICE\_CONTROL Attributes**

Table 2-31 lists BITSLICE\_CONTROL attributes. Most of these attributes have an equivalent register bit or bits in the RIU.

Table 2-31: BITSLICE\_CONTROL Attributes

Attribute	Value	Default	Type	Description
EN_OTHER_PCLK	TRUE FALSE	FALSE	String	<p>Enable inter-nibble clocking. When set to TRUE, the PCLK is sourced from the other BITSLICE_CONTROL in the byte.</p> <p>If this is turned on for one BITSLICE_CONTROL, it cannot be turned on for the other BITSLICE_CONTROL in the same byte.</p>
EN_OTHER_NCLK	TRUE FALSE	FALSE	String	<p>Enable inter-nibble clocking. When set to TRUE, the NCLK is sourced from the other BITSLICE_CONTROL in the byte.</p> <p>If this is turned on for one BITSLICE_CONTROL, it cannot be turned on for the other BITSLICE_CONTROL in the same byte.</p>
SERIAL_MODE	TRUE FALSE	FALSE	String	<p>When set to TRUE, the master input clock, PLL_CLK or REFCLK, and the divided versions are used as the sample clock for the deserializer of a bit slice.</p> <p>When set to FALSE, the clock or strobe applied to a BITSLICE_0 is used as sample clock.</p> <p>When only data or data with embedded clock are applied to bit slices, use the SERIAL_MODE. The main function of the bit slices is to sample an incoming data stream with a clock generated from an internal, unrelated to the data source, such as a PLL.</p>

Table 2-31: BITSlice\_CONTROL Attributes (Cont'd)

Attribute	Value	Default	Type	Description
RX_CLK_PHASE_P	SHIFT_0 SHIFT_90	SHIFT_0	String	<p>Shifts the P-edge of the read clock by 0 degrees or 90 degrees relative to the captured data.</p> <p>Data is sampled by a clock in the middle of a bit period. When clock and data arrive at the pin phase-aligned, use a shift by 90 degrees, or else leave this attribute at the default value.</p> <p>When using SHIFT_90, DELAY_VALUE (RX_BITSLICE) or RX_DELAY_VALUE (RXTX_BITSLICE) must be 0.</p>
RX_CLK_PHASE_N	SHIFT_0 SHIFT_90	SHIFT_0	String	<p>Shifts the N-edge of the read clock by 0 degrees or 90 degrees relative to the captured data.</p> <p>Data is sampled by a clock in the middle of a bit period. When clock and data arrive at the pin phase-aligned, use a shift by 90 degrees, or else leave this attribute at the default value.</p> <p>When using SHIFT_90, DELAY_VALUE (RX_BITSLICE) or RX_DELAY_VALUE (RXTX_BITSLICE) must be 0.</p>
INV_RXCLK	TRUE FALSE	FALSE	String	Invert the read or sample CLK applied to BITSlice_0.
TX_GATING	DISABLE ENABLE	DISABLE	String	<p>Write clock gating. For aligned transmitted data, TX_GATING must set to ENABLE and control the TBYTE_IN from interconnect logic.</p> <p>Read <a href="#">Native Mode Bring-up and Reset, page 296</a> for more information.</p> <p><b>Note:</b> TX_GATING = ENABLE does not stop the clock for BITSlice_1 and BITSlice_6.</p> <p>When TX_GATING = ENABLE, TBYTE_IN[3:0] is used to stop the clock for transmit interfaces.</p>

Table 2-31: BITSlice\_CONTROL Attributes (Cont'd)

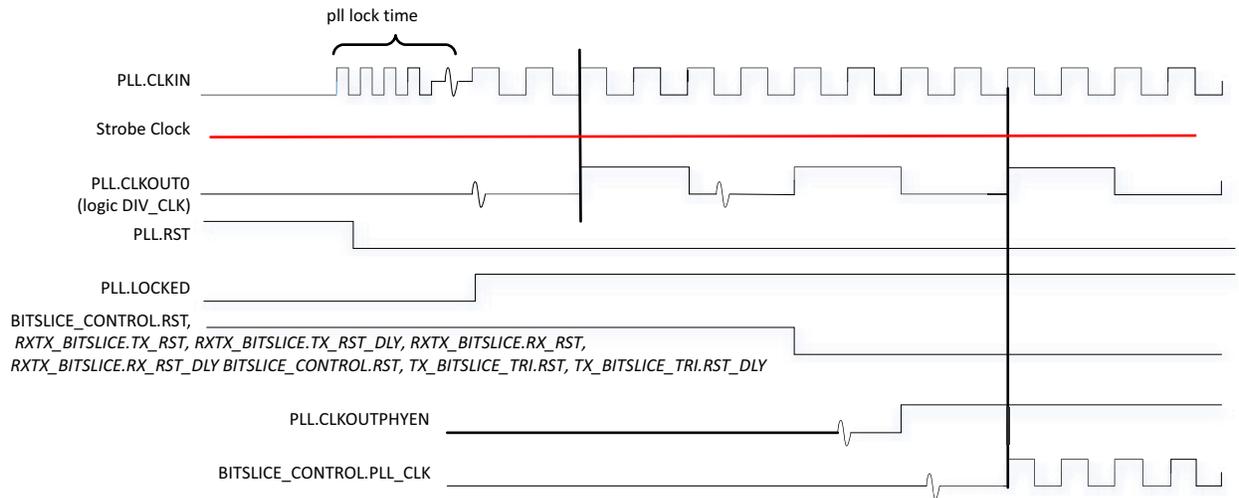
Attribute	Value	Default	Type	Description
RX_GATING	DISABLE ENABLE	DISABLE	String	<p>Enables read strobe/clock gating.</p> <p>The value of this attribute and the mechanism behind it is to gate in the strobe/clock during its preamble. Gate off the strobe/clock immediately following each of its falling edges and enable it afterward.</p> <p>The gating circuit used by the attribute is only available in BITSlice_0 of a nibble because strobe/clock can only be input from the BITSlice_0 location in a nibble.</p> <p>When set to TRUE, the gate is controlled by the BITSlice_CONTROL.PHY_RDEN input.</p>
READ_IDLE_COUNT[5:0]	0 to 63	0	Decimal	<p>Number of clocks after PHY_RDEN deassertion and before turning off ODT termination.</p> <p>MIG USE ONLY.</p>
DIV_MODE	DIV2 DIV4	DIV2	String	<p>Determines how the master clock is divided.</p> <p>When 8-bit mode is used (1:8 serial input), set to DIV4.</p> <p>When 4-bit mode is used, set to DIV2. The FIFO_WRCLK_OUT clock reflects the action of this attribute.</p>
REFCLK_SRC	PLLCLK, REFCLK	PLLCLK	String	<p>When the master clock is the PLL_CLK, this attribute should be set to PLLCLK.</p> <p>When the master clock is the REFCLK input (RX_BITSLICE only), this attribute must be set to REFCLK.</p>
ROUNDING_FACTOR	1, 2, 4, 8, 16, 32, 64, 128	16	Decimal	<p>Rounding factor for BISC.</p> <p>MIG USE ONLY.</p>
CTRL_CLK	EXTERNAL	EXTERNAL	String	<p>Defines the clock source for the RIU interface. Always use the default value, EXTERNAL.</p>

Table 2-31: BITSlice\_CONTROL Attributes (Cont'd)

Attribute	Value	Default	Type	Description
EN_CLK_TO_EXT_NORTH	ENABLE DISABLE	DISABLE	String	Enable inter-byte strobe/clock forwarding to another upper byte BITSlice_CONTROL.
EN_CLK_TO_EXT_SOUTH	ENABLE DISABLE	DISABLE	String	Enable inter-byte strobe/clock forwarding to another lower byte BITSlice_CONTROL.
EN_DYN_ODLY_MODE	TRUE FALSE	FALSE	String	MIG USE ONLY.
SELF_CALIBRATE	ENABLE DISABLE	ENABLE	String	Built-in self-calibration (BISC) enable. When set to ENABLE, BISC runs initial calibration after release of reset. When set to DISABLE, calibration is not run after release of reset.
IDLY_VT_TRACK	TRUE FALSE	TRUE	String	Enables voltage and temperature tracking for all input delays in a nibble.
ODLY_VT_TRACK	TRUE FALSE	TRUE	String	Enables voltage and temperature tracking for all output delays in a nibble.
QDLY_VT_TRACK	TRUE FALSE	TRUE	String	Enables voltage and temperature tracking for the quarter delays in the BITSlice_CONTROL. Quarter delays are used to shift the clock relative to the incoming data.
RXGATE_EXTEND	TRUE FALSE	FALSE	String	MIG USE ONLY.
SIM_DEVICE	Possible Values: ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2	ULTRASCALE	String	Sets the device version (ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2)

## Native Mode Bring-up and Reset

To bring up a design in an UltraScale device using native SelectIO primitives, a specific set of steps must be followed to apply or release the reset. Follow the described steps to ensure all clocks are phase-aligned and related as shown in [Figure 2-61](#) between PLL/MMCM, BITSlice\_CONTROL, and bit slices.



X19013-080719

Figure 2-61: PLL and Reset Bring-up Sequence

In [Figure 2-61](#), PLL.CLKOUTPHY\_EN disables the BITSlice\_CONTROL.PLL\_CLK until after reset has been removed. For detailed descriptions for the clocking requirements, see [Clocking in Native Mode](#).

PLL\_CLK/REFCLK entering BITSlice\_CONTROL should be disabled until all the BITSlice\_CONTROLS and RXTX\_BITSlices are reset and their resets have been safely removed. This ensures a deterministic bring-up of the interface.

At startup of a design or after a reset has been applied to the application in the FPGA, the reset must be released using the following sequence:

### Release Reset

1. Ensure that the SELF\_CALIBRATE attribute is set to ENABLE.
2. On all used RXTX\_BITSlice (RX\_BITSlice, TX\_BITSlice) primitives, hold the EN\_VTC signals High.
3. EN\_VTC of the BITSlice\_CONTROL should be held Low.
4. Use the following sequence to bring the I/O out of reset:
  - a. Release the reset of the PLL/MMCM generating the clocks for the interface.

- b. Keep the CLKOUTPHYEN of the used PLL Low, which disables the CLKOUTPHY high-speed clock to the BITSlice\_CONTROL.PLL\_CLK input. When a MMCM is used, disable the BUFGCE clock buffer delivering the BITSlice\_CONTROL.REFCLK clock.  
**Note:** As shown in [Figure 2-61](#), strobe clocks must be disabled during the reset sequence. For systems that use the input clock as the strobe clock, bitslip will be required. The High-Speed SelectIO wizard provides the bitslip functionality.
  - c. Wait for the PLL/MMCM to reach the LOCKED state.
  - d. Release these reset signals: RXTX\_BITSlice.TX\_RST\_DLY, RXTX\_BITSlice.RX\_RST\_DLY, TX\_BITSlice\_TRI.RST\_DLY, RXTX\_BITSlice.TX\_RST, RXTX\_BITSlice.RX\_RST, TX\_BITSlice\_TRI.RST, and/or BITSlice\_CONTROL.RST.
  - e. Wait at least 64 application clock cycles (PLL/MMCM specification).
  - f. Pull the CLKOUTPHYEN signal of the PLL High, which enables the CLKOUTPHY high-speed PLL output. For a MMCM, enable the BUFGCE to apply the BITSlice\_CONTROL.REFCLK.
5. Continue with the following post-reset sequence:
- a. Wait until the DLY\_RDY of all the used BITSlice\_CONTROL primitives are asserted High by the running BISC controllers.
  - b. After all the DLY\_RDY signals are asserted High, use the RIU\_CLK in a two flip-flop synchronizer circuit to pull the EN\_VTC of the used BITSlice\_CONTROL High. For asynchronous RX designs, BITSlice\_CONTROL.EN\_VTC is tied Low by the High-Speed SelectIO wizard.
  - c. Wait until the BITSlice\_CONTROL.VTC\_RDY status output of the BITSlice\_CONTROL is asserted High. VTC\_RDY being High at this point means that the BISC controller in the BITSlice\_CONTROL primitive is tracking for voltage and temperature compensation.
  - d. Strobe clocks can now be restarted.

**Note:** For systems that cannot stop the strobe clocks during the reset sequence or for systems that have noisy strobes such as unlocked PLLs, bitslip might be required for RX\_BITSlice alignment.

At this point the application in the FPGA logic can be released.

Extra functional mode guidelines after VTC\_RDY is High follow:

- RXTX\_BITSlice transmitters or TX\_BITSlices require that the TBYTE\_IN[3:0] inputs of the BITSlice\_CONTROL are pulled High. Use the VTC\_RDY signal and a two register synchronizer running from the application clock to perform this action.

**Note:** If the TBYTE\_IN bus is used by logic in the FPGA, ensure the designed circuit allows that the guidelines provided above can be applied.

- RXTX\_BITSLICE receivers or RX\_BITSLICES require that the PHY\_RDEN[3:0] inputs of the BITSlice\_CONTROL are pulled High. Use the VTC\_RDY signal and a two register synchronizer running from the application clock to perform this action.

**Note:** Follow the actions described in the FIFO function paragraph of [RXTX\\_BITSLICE, page 198](#) about reading data from the FIFO.

**Note:** For transmit-only interfaces, the PHY\_RDEN[3:0] should be deasserted Low.

Extra functional guidelines for serial mode receivers follow:

- A serial mode receiver only receives data. The data must be sampled by a PLL generated clock (PLL.CLKOUTPHY). In this case it is necessary that you adjust the input delay lines using additional logic to control the RX\_BITSLICE.
- Follow the guidelines in [Native Input Delay Type Usage, page 204](#) and [Native Output Delay Type Usage, page 220](#) to adjust the delay line in a correct manner.

When an application is running in an FPGA, apply the following steps to safely reset the application and allow a correct bring-up afterward:

### Apply Reset

To apply a reset:

1. Assert reset to the PLL.
2. Apply reset to RXTX\_BITSLICE.TX\_RST\_DLY, RXTX\_BITSLICE.RX\_RST\_DLY, TX\_BITSLICE\_TRI.RST\_DLY, RXTX\_BITSLICE.TX\_RST, RXTX\_BITSLICE.RX\_RST, TX\_BITSLICE\_TRI.RST, and/or BITSlice\_CONTROL.RST.
3. Stop strobe clocks.
4. Wait the minimum PLL reset assertion time before releasing the reset. For this timing specification, consult the PLL section of the UltraScale device data sheets [\[Ref 2\]](#). Then follow the steps in [Native Mode Bring-up and Reset, page 296](#) for correct bring-up.

### Bring-up for an Interface Using Multiple Banks

When an interface spans multiple banks, the clocking and bring-up sequence for each bank must be modified to ensure the interfaces start up correctly. When using the High-Speed SelectIO wizard, each bank can be customized by running the HSSIO-Wiz separately and selecting **Enable Ports to Connect Multiple Interfaces**. [Figure 2-62](#) shows an interface that spans two banks. An application clock (APP\_CLK) is used for loading data into the TX\_BITSLICE. As shown in [Figure 2-62](#), the TX\_BITSLICE uses the dedicated clocking from the PLL for the transmit clock. The dedicated PLL clock provides optimal performance for the TX\_BITSLICE. For RX\_BITSLICE, the APP\_CLK is given as FIFO\_RD\_CLK to read the data from FIFO.

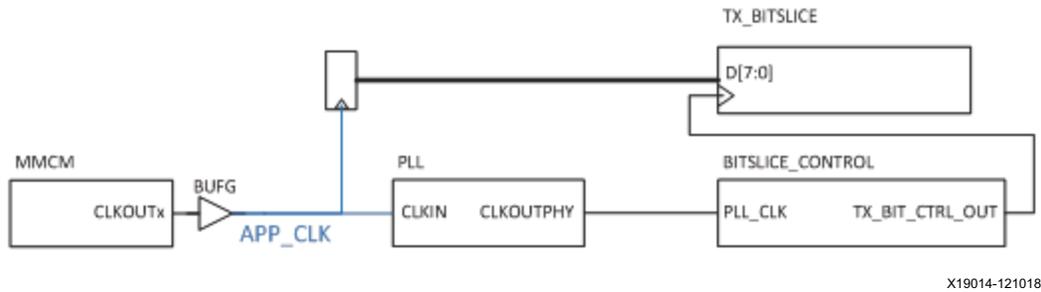


Figure 2-62: TX\_BITSLICE Application Clock

The High-Speed SelectIO wizard can use CLKOUT0/CLKOUT1 for the application clock which can be used when a single bank is used.

In the case of multiple bank interfaces (Figure 2-63), a single clock source is used to drive the APP\_CLK for each of the High-Speed SelectIO wizard cores. Consequently CLKOUT0/CLKOUT1 should not be connected.

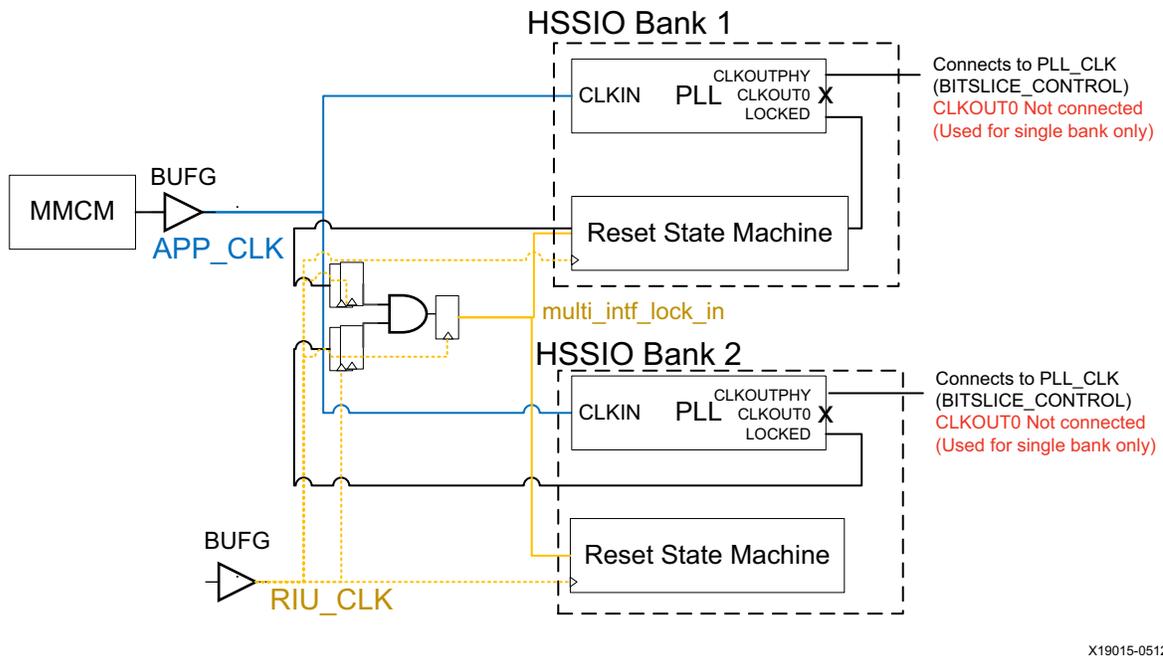


Figure 2-63: Multi-bank Clocking

The High-Speed SelectIO wizard uses the RIU\_CLK for the Reset State Machine. To ensure multi-bank interfaces are aligned, all of the banks should be reset at the same time. The LOCKED outputs from each of the PLLs should be synchronized to the RIU\_CLK domain and logically ANDed together. These changes allow the state machines to be brought up together.

Each bank contains an RST\_SEQ\_DONE status signal. To determine when all of the banks are ready, the RST\_SEQ\_DONE from all of the interfaces should be logically ANDed to create an interface ready (INTF\_RDY) signal. The INTF\_RDY should be synchronized to the APP\_CLK and used to control TBYTE\_IN[3:0] for designs using TX\_BITSLICE. When using the High-Speed SelectIO wizard, the tri\_tbyte#[3:0] inputs should be connected to INTF\_RDY signal. For designs targeting RX\_BITSLICE, the FIFO\_RD\_EN should only be used after the entire interface is ready and INTF\_RDY has gone High.

**Note:** Use the inverted FIFO\_EMPTY signal of the used bit slice farthest away from the bit slice receiving the clock and thus generating the FIFO\_WRCLK\_OUT through a flip-flop to all FIFO\_RD\_EN inputs of used bit slices. Farthest means the bit slice at the end of the clock backbone. See the [FIFO Function](#) section in [Native Primitives](#).

When using the High-Speed SelectIO wizard, the INTF\_RDY is internally synchronized to APP\_CLK.

This is a summary of multi-bank requirements.

Multi-bank clocking changes:

- PLLs for each of the High-Speed SelectIO wizard cores should be driven from a single MMCM clock source to minimize skews between the PLLs. As a result, if three banks were to be used, the MMCM should be placed in the middle I/O bank. Minimizing the clock skews to the different PLLs is more critical than controlling the input clock routing for the MMCM.
- Application clock (APP\_CLK) for each core must be updated to use the MMCM clock for multi-bank clocking.

Reset State Machine

- All PLLs and reset state machines should be reset at the same time.
- LOCKED outputs from all banks must be combined and synchronized to the RIU clock domain. Because the LOCKED signal is an input into the reset state machine, which is driven by the RIU clock domain, the combined multi-bank LOCKED signal must also be in the RIU clock domain.
- The application must wait for RST\_SEQ\_DONE from all banks before enabling the application (INTF\_RDY). This signal should be synchronized to the application clock domain and control TBYTE\_IN[3:0] for transmit applications or FIFO\_RD\_EN for receive applications.

### Bring-up for Multiple Interfaces in a Shared Bank

When a bank contains two different interfaces, the used BITSlice\_CONTROLS within the bank share common control signals requiring the Native Mode Bring-up to start at the same time. Each interface can complete the bring-up sequence at different times. The High-Speed SelectIO wizard must also be modified to ensure the critical steps of the bring-up sequence are synchronized between the interfaces.

Figure 2-64 shows an example design that uses two different interfaces with separate RIU\_CLK connections. When sharing a bank, the RIU\_CLKs should not vary by more than 4x. For example, if RIU\_CLK1 is 200 MHz, then RIU\_CLK2 must be at least 50 MHz.

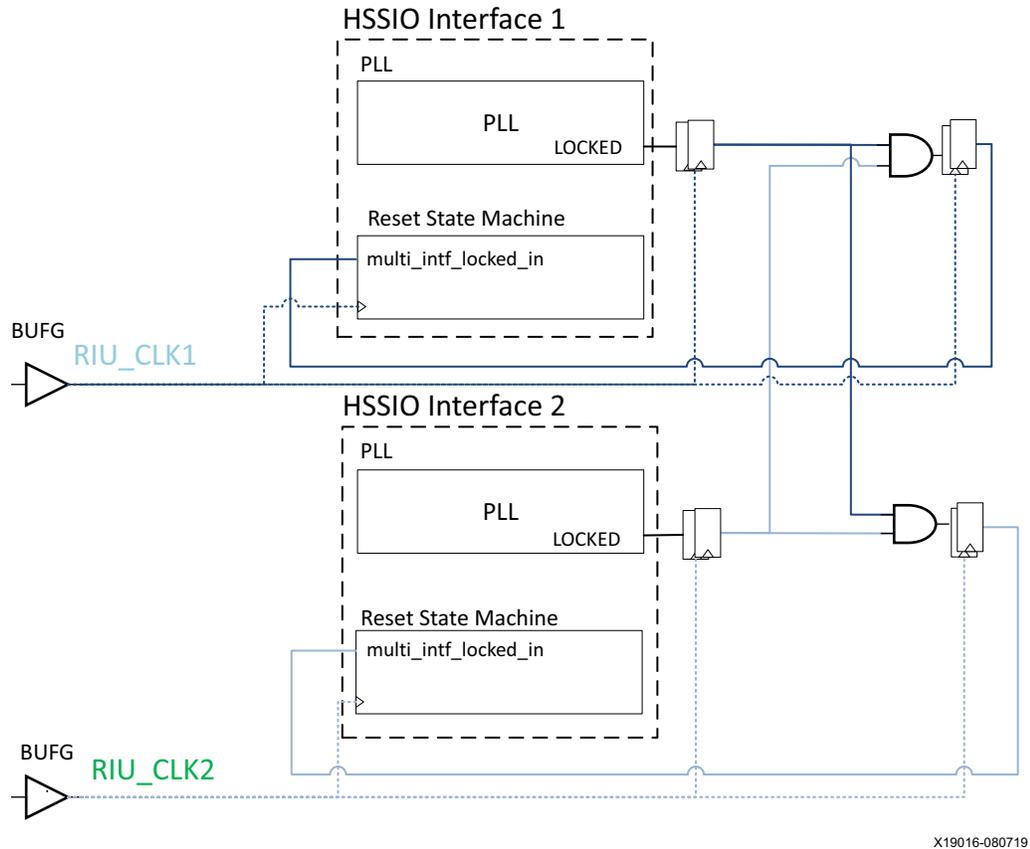


Figure 2-64: Multiple Interfaces in a Shared Bank

As a synchronous state machine, the input data should use the same clock as the state machine. For example, the LOCKED (PLL) signals for both interfaces must be ANDed together and resynchronized to the interface's RIU\_CLK source. Interface 1 should use the RIU\_CLK1 clock domain and interface 2 uses RIU\_CLK2.

Each bank contains an RST\_SEQ\_DONE status signal. To determine when all of the banks are ready, the RST\_SEQ\_DONE from all of the interfaces should be logically ANDed to create an interface ready (INTF\_RDY) signal. The INTF\_RDY should be synchronized to the APP\_CLK and used to control TBYTE\_IN[3:0] for designs using TX\_BITSLICE. When using the High-Speed SelectIO wizard, the tri\_tbyte#[3:0] inputs should be connected to INTF\_RDY signal. For designs targeting RX\_BITSLICE, the FIFO\_RD\_EN should only be used after the entire interface is ready and INTF\_RDY has gone High.

**Note:** Use the inverted FIFO\_EMPTY signal of the used bit slice farthest away from the bit slice receiving the clock and thus generating the FIFO\_WRCLK\_OUT through an optional flip-flop to all FIFO\_RD\_EN inputs of used bit slices. Farthest means the bit slice at the end of the clock backbone. See the [FIFO Function](#) section in [Native Primitives](#).

## Clocking in Native Mode

The pins and attributes of the native I/O primitives involved in clocking are described in this section.

**Table 2-32: Pins and Attributes of Native I/O Primitives Involved in Clocking**

Pin or Attribute	I/O	Description
<b>BITSLICE_CONTROL Pins</b>		
PLL_CLK	Input	High-speed clock delivered by a PLL (CLKOUTPHY) in the same I/O bank and routed over dedicated resources. Normally, this clock is the same frequency as the required data rate (example: 1 GHz clock for a 1 Gb/s data rate). For designs using SERIAL mode, this clock is the serial DDR data sampling clock and thus equal to 1/2 the data rate (for example, for 1 Gb/s data rate, the DDR clock is 500 MHz).
REFCLK	Input	Clock delivered by an MMCM or PLL, not necessarily in the same I/O bank as the one using the BITSLICE_CONTROL components. This clock arrives at the BITSLICE_CONTROL over normal clock routing of the FPGA and uses BUFG and BUFGCE clock buffers.
<ul style="list-style-type: none"> <li>• The PLL_CLK or REFCLK are referred to as the BITSLICE_CONTROL master clock.</li> <li>• This master clock is selected by the REFCLK_SRC attribute.</li> <li>• The clock source, PLL_CLK or REFCLK, is mutually exclusive (one or the other but not both).</li> </ul>		
CLK_FROM_EXT	Input	This input is part of the inter-byte clocking structure. It is a clock routed over dedicated routing in the BITSLICE_CONTROL BITSLICE structure and comes from the CLK_TO_EXT_NORTH or CLK_TO_EXT_SOUTH outputs of a BITSLICE_CONTROL in a neighboring byte. When not used, tie High.
CLK_TO_EXT_NORTH CLK_TO_EXT_SOUTH	Output	This is part of the inter-byte clocking structure. It is a copy of the data sample clock that is forwarded over dedicated routing resources to a neighboring byte BITSLICE_CONTROL or CLK_FROM_EXT clock input.
PCLK_NIBBLE_IN NCLK_NIBBLE_IN	Input	These inputs are part of the inter-nibble clocking structure and are routed over dedicated routing resources to the N(P)CLK_NIBBLE_OUT between the upper and lower nibbles within a byte.
PCLK_NIBBLE_OUT NCLK_NIBBLE_OUT	Output	These outputs are part of the inter-nibble clocking structure and are routed over dedicated routing resources to the N(P)CLK_NIBBLE_IN between the upper and lower nibbles within a byte.
<b>BITSLICE_CONTROL Attributes</b>		
REFCLK_SRC		Determines what master clock input is used.
DIV_MODE		Determines the division factor of the master clock in the BITSLICE_CONTROL. When 4 bits are used, set to DIV2. When 8 bits are used, set to DIV4.
SELF_CALIBRATE		Determines whether or not the clock is going to be tuned to the captured data and tracked over voltage and temperature.

Table 2-32: Pins and Attributes of Native I/O Primitives Involved in Clocking (Cont'd)

Pin or Attribute	I/O	Description
IDLY_VT_TRACK ODLY_VT_TRACK QDLY_VT_TRACK		Turn VT tracking on or off per type of delay line. By default these attributes are turned on.
RX_CLK_PHASE_N RX_CLK_PHASE_P		Shifts the internal capture clock by 90 degrees (or not). When data and clock arrive phase-aligned, this attribute can be set to SHIFT_90. When data and clock arrive 90-degrees shifted, use SHIFT_0.
EN_CLK_TO_EXT_NORTH EN_CLK_TO_EXT_SOUTH		Enable inter-byte clocking to the north or south BITSlice_CONTROL component.
EN_OTHER_NCLK EN_OTHER_PCLK		Set the direction of the inter-nibble clocking. Section <a href="#">Clocking in Native Mode, page 302</a> has details on how inter-nibble clocking allows clocks or strobes to be shared within a byte.
<b>RXTX_BITSlice Pins</b>		
FIFO_WRCLK_OUT	Output	This is a copy of the internal FIFO write clock. The frequency of this clock is the data sample clock divided by the factor of the DIV_MODE attribute. The data sample clock can be the provided REFCLK or PLL_CLK or can be the clock or strobe provided at a BITSlice_0.
FIFO_RD_CLK	Input	This is a clock provided by a MMCM, PLL, or other. This clock must have the same frequency as the internal bit slice FIFO write clock, but most likely it has a different phase.
<b>RXTX_BITSlice Attributes</b>		
OUTPUT_PHASE_90		When set to TRUE, the transmitter output is phase-shifted over 90 degrees. The phase shift can easily be observed when different transmitters are used. This attribute is most often used to shift the generated clock 90 degrees to the generated data.
RX_DATA_WIDTH TX_DATA_WIDTH		This attribute sets the width of the serial-to-parallel and parallel-to-serial converter. It must correspond to the DIV_MODE attribute of the BITSlice_CONTROL. When set to 8, DIV_MODE must be set to 4 or conversely, when DATA_WIDTH is set to 4, DIV_MODE must be set to 2.
RX_DATA_TYPE		Set to DATA when the bit slice receiver is used to capture only data. Set to DATA_AND_CLOCK (only for BITSlice_0) when the clock can be used as a sample clock for the data (SERIAL_MODE = FALSE) and that clock is also sampled as data. Set to SERIAL when bit slice receive data is captured by PLL_CLK.

Table 2-32: Pins and Attributes of Native I/O Primitives Involved in Clocking (Cont'd)

Pin or Attribute	I/O	Description
RX_REFCLK_FREQUENCY TX_REFCLK_FREQUENCY		This attribute must be set to the frequency applied to the BITSlice_CONTROL master clock input (PLL_CLK or REFCLK).
There are no clocks or clock-related pins related to the bit slice transmitter. The transmitter in the RXTX_BITSLICE uses the BITSlice_CONTROL master clock (PLL_CLK or REFCLK) to transmit data. The transmit data rate is equal to the BITSlice_CONTROL master clock frequency. For example, when the master clock frequency is 1000 MHz, the transmitted data rate is equal to 1 Gb/s.		

### Receive Clocking

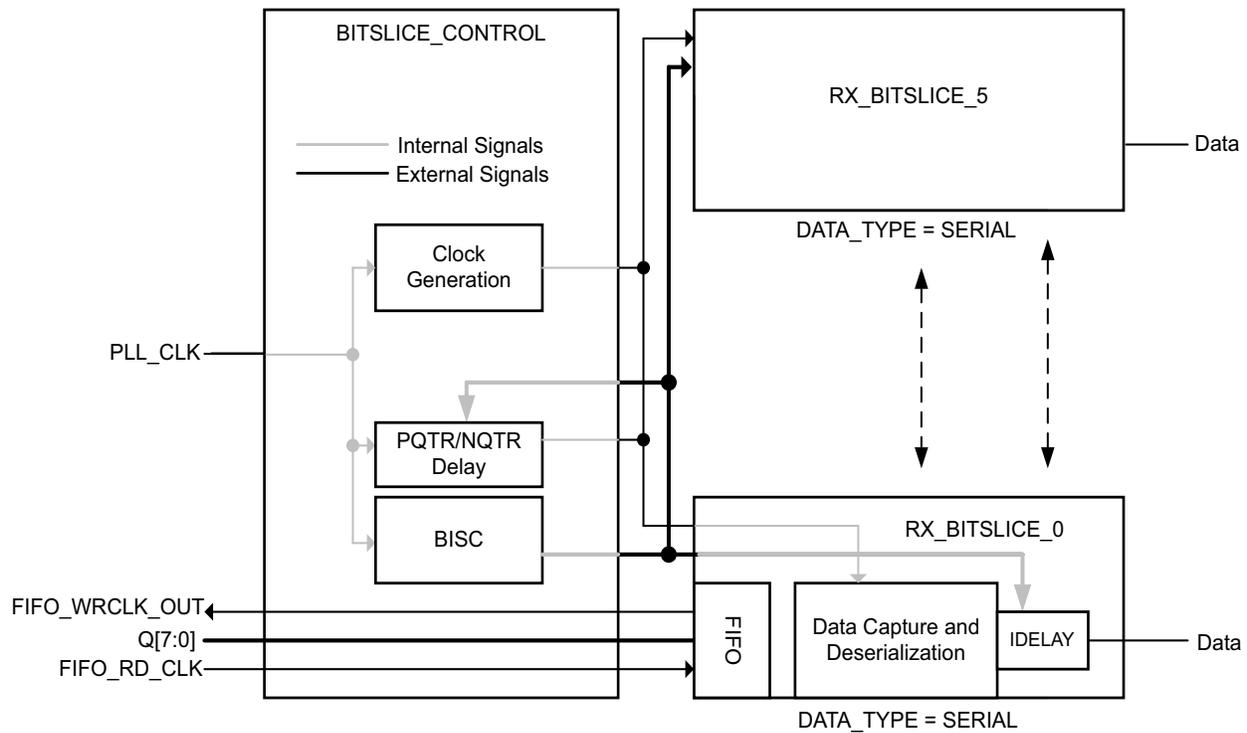
The RXTX\_BITSLICE has two distinct clock/strobe sources, initiated by an attribute or RIU register bit to capture data. The attribute or register bit impacts the functioning of the BITSlice\_CONTROL primitive SERIAL\_MODE = TRUE/FALSE.

- When the attribute SERIAL\_MODE is set to TRUE, the received data is captured using the applied master clock (PLL\_CLK or REFCLK). This clock is the DDR capture clock normally running at half the data. Logic in the BITSlice\_CONTROL regenerates and divides the master clock to form required clocks in the receiver.
- When the attribute SERIAL\_MODE is set to FALSE, then the received data is captured using a clock or strobe forwarded with the data. An attribute (INV\_RXCLK) can enable an inverter in this receiver clock path. The clock applied to the BITSlice\_CONTROL master clock input is used by the BISC controller for input delay line calibration and must have the same frequency as the received data rate.

### SERIAL\_MODE = TRUE

As shown in [Figure 2-65](#), this setup that can be used when:

- Only the data is received from a connected component.
- The received data contains an embedded clock as in SGMII and some other protocols that are normally fed to a GTH or GTY high-speed serial transceiver.
- The clock delivered with the data is not a bit clock but a frame or system-synchronous clock.



X16049-071617

Figure 2-65: Data Capture in Serial Mode

When data is received, a clock is needed to capture that data. In each of the above cases, BITSLICE\_CONTROL master clock (PLL\_CLK or REFCLK) is used to capture the data.

The PLL has dedicated high speed and low jitter connections to the BITSLICE\_CONTROL primitive. When a MMCM is used, clocks are routed over global FPGA clock routing, and clock buffers BUFG or BUFGCE must be inserted in the clock path causing more jitter. Using a PLL allows capturing higher data rates than when using a MMCM.

The PLL used to generate the data-capture clock must be in the same I/O bank as the receiving RXTX\_BITSLICES. In this mode, all inputs of a nibble can be used as data inputs. When the input data is differential, then bit 6 of the upper nibble cannot be used.

The frequency of the clock is equal to a DDR clock that should be used to capture the received data. For example, receiving a data stream of 1 Gb/s requires that the frequency of the master clock is 500 MHz.

A clock generator in the BITSLICE\_CONTROL using DATA\_WIDTH and DIV\_MODE attributes ensures that all necessary clocks for serial-to-parallel conversion of the data are generated.

For example, capturing 8-bit wide data requires DATA\_WIDTH to be set to 8 and DIV\_MODE to 4. The captured and serial-to-parallel converted data is written into the RX\_BITSLICE output FIFO at a rate of the master\_clock/4.

The FIFO\_WRCLK\_OUT pin of BITSlice\_0 in each nibble provides a copy of the clock used to write data, internal to the RXTX\_BITSlice, in the FIFO. This FIFO\_WRCLK\_OUT or a clock of the same frequency generated from a PLL or MMCM can then be used as FIFO\_RD\_CLK.



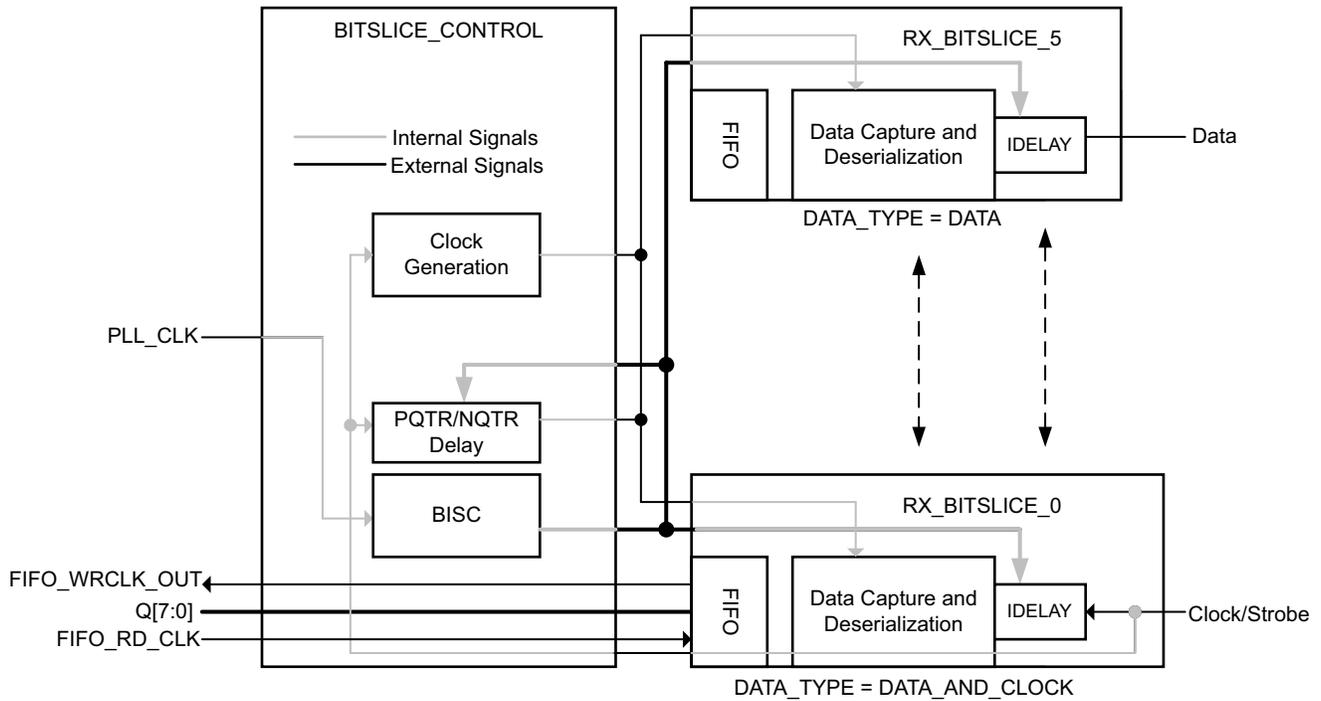
**CAUTION!** When a nibble uses the attribute `SERIAL_MODE=TRUE`, a `BITSlice_0` **must** be instantiated into the design and the `DATA_TYPE` must be set to `SERIAL`. Even when `BITSlice_0` is not used in the design, it **must** be connected to an I/O buffer or else the Vivado tools error out. For the upper nibble, `BITSlice_0` is the equivalent of `BITSlice_6` for a byte group. See [Figure 2-3](#) for an explanation of bitslice numbering within a byte and nibble.

### **SERIAL\_MODE = FALSE**

[Figure 2-66](#) shows the setup to be used with all kinds of source synchronous interfaces. These interfaces provide data with related clock or strobe. The `BITSlice_0` received clock or strobe is used to capture the received data in the other `RX_BITSlices` of the nibble, byte, or entire I/O bank. The clock supplied to the `BITSlice_CONTROL` master clock input is used to calibrate the input delay lines, and its frequency must be equal to the received data rate.

This kind of interface requires connecting the clock or strobe to the `BITSlice_0` of a nibble. These pins are labeled as dedicated byte clock (DBC), as quad byte clock (QBC), or as a dual purpose pin (GC/QBC). The dual purpose GC/QBC clock input allows the received bit clock to be used to capture the data bits in the other bit slices of the nibble, byte, or I/O bank, but can also be used as clock input for the PLL to generate a valid `BITSlice_CONTROL` master clock.

While `BITSlice_0` is used as clock input, the other bit slices of the nibble can be used for data capture. A `BITSlice_0` used as clock input can capture the clock as if it is a data pattern. This can be useful for many kinds of control functionality in a design.



X16050-071617

Figure 2-66: Data Capture in Non-Serial Mode

Only BITSlice\_0 in a nibble can be used as clock/strobe input. When more bit slices than a nibble are used to capture data, inter-nibble and/or inter-byte clocking must be used. By using this technique, clock/strobe can be forwarded to the entire I/O bank. For example, if a connected device delivers 16 data channels with a single clock, this requires multiple nibbles/bytes to capture data while a single BITSlice\_0 must be defined as the input of the forwarded clock. The clock needs inter-nibble and inter-byte clocking to serve all data channels. The supplied data sample clock or strobe is tuned and maintained over voltage and temperature by the BISC controller in the BITSlice\_CONTROL primitive.



**TIP:** A used BITSlice\_0 within a nibble or a combinatorial signal connected to the associated I/O pin passing through the BITSlice\_0 is not available to the application in the FPGA until after the DLY\_RDY output pin of the BITSlice\_CONTROL or RDY output pin of IDELAYCTRL in the nibble used by the BITSlice\_0 is HIGH. BITSlice\_0 is used for calibration and is only available after calibration has completed. The Vivado tool generates a Critical Warning that can be demoted by using the XDC Constraint:

```
set_property UNAVAILABLE_DURING_CALIBRATION TRUE [get_ports <port name>]
```

There is one exception: In an I/O bank, there is one differential QBC/GC pin set or two single-ended QBC/GC pins. Those can be used to carry a clock to a MMCM or PLL while BISC is running and while the possibly connected BITSlice\_0 is unavailable.

The single received clock, in BITSlice\_0, can be used to capture data in the nibble, byte, or entire I/O bank. Assuming a lower and upper nibble are used, and clocking is passed from the lower nibble to the upper nibble, then the BITSlice\_0 of the lower nibble can carry the input/sample clock while the BITSlice\_0 of the upper nibble can be a normal data input.

DATA\_WIDTH and DIV\_MODE attributes set a clock generator in the BITSlice\_CONTROL to generate all necessary clock divisions for serial-to-parallel conversion and RX\_BITSLICE FIFO write operations.

The received clock/strobe is passed through, tuned, and forwarded by the BITSlice\_CONTROL, and is used to capture received data in the other bit slices. The data is parallelized by a divided version (DIV\_MODE) of the capture clock and written into the RX\_BITSLICE FIFO. The received clock/strobe that is passed through, tuned, and forwarded by the BITSlice\_CONTROL can be used in RX\_BITSLICE\_0 to capture an image of the clock. This clock data can be used in the general device logic. The image of the clock is presented in a data format to the interconnect logic and can be used for any design-related functionality.

To read data from the FIFO, a clock must be connected to the FIFO\_RD\_CLK input. This clock should have the same frequency as the data sample clock divided by the DIV\_MODE parameter. The FIFO\_RD\_CLK can be generated by a PLL or a MMCM. The same PLL that is used for the high speed PLL\_CLK can be used to generate this FIFO\_RD\_CLK.

The FIFO\_RD\_CLK can also be sourced from the FIFO\_WRCLK\_OUT of a BITSlice\_0 when this bit slice is used as sample clock input. The BITSlice\_0 used with FIFO\_WRCLK\_OUT must be in the same I/O bank as the clocked FIFOs.

When SELF\_CALIBRATE is TRUE, the received clock is tuned in the BITSlice\_CONTROL by the BISC controller to 90 degrees or 0 degrees depending on RX\_CLK\_PHASE\_P and RX\_CLK\_PHASE\_N attribute values. The BISC controller also compensates the data and clock delay mismatches arriving at the RX\_BITSLICE input, but it does not compensate for delays outside the general device logic. The BISC controller runs on the applied master clock, PLL\_CLK, or REFCLK input of the BITSlice\_CONTROL. Because the master clock has nothing to do with capturing data, its frequency rate should be set to be equal to the received data rate.

As an extra function, BISC can continuously track voltage and temperature variations to maintain the clock and data timing relationship over V and T. When input delay elements are used in the data input path, BISC tracks for voltage and temperature compensation.

One nibble contains 6 (lower) or 7 (upper) bit slices resulting in the following possible amounts of I/Os:

Single-Ended I/Os	Differential I/Os
1 clock input	1 clock input
5 or 6 data inputs	2 data inputs

One byte combines an upper and a lower nibble resulting in the following possible amounts of I/Os:

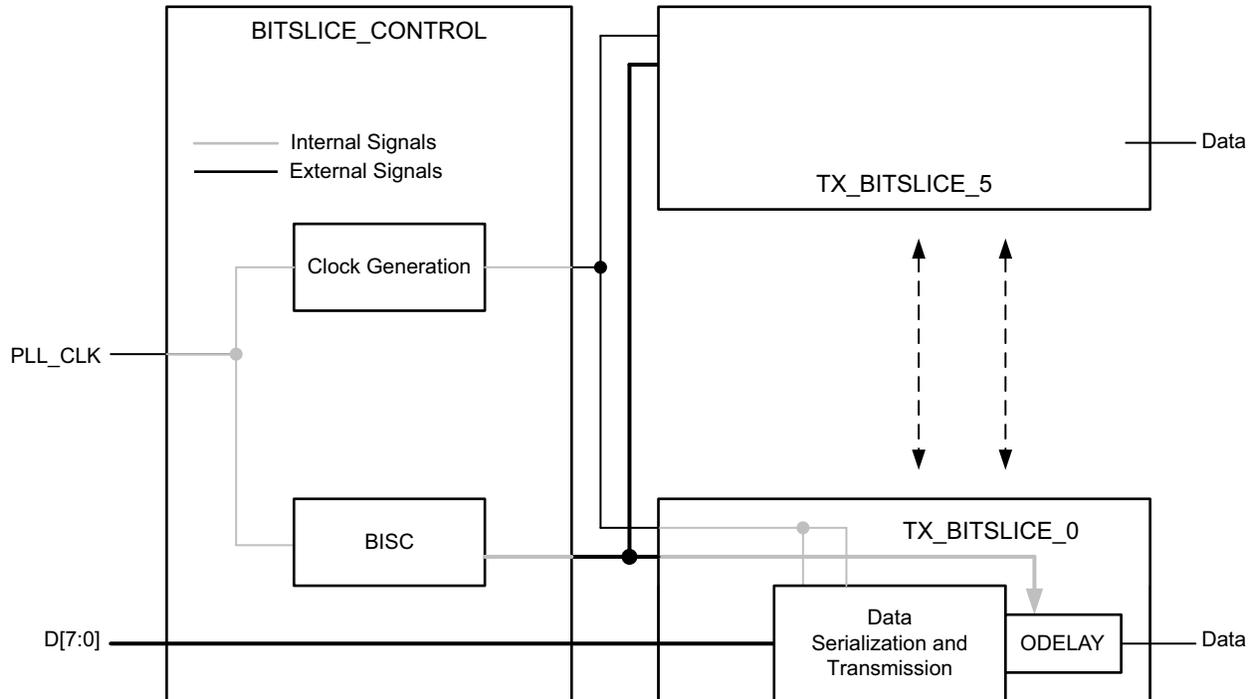
Single-Ended I/Os	Differential I/Os
1 clock input	1 clock input
12 data inputs	5 data inputs

One I/O bank is equal to a combination of four bytes, resulting in the following possible amounts of I/Os:

Single-Ended I/Os	Differential I/Os
1 clock input	1 clock input
Up to 51 data inputs	Up to 23 data inputs

**Transmit Clocking**

When transmitting data, the master input clock of the BITSlice\_CONTROL is used to shift data out of the transmitter (Figure 2-67). The frequency of this clock determines the serial bit rate of the data. Data must be presented at the RXTX\_BITSLICE transmitter inputs at a clock running at the master clock divided by the DIV\_MODE and/or DATA\_WIDTH attribute setting.



X16051-050917

Figure 2-67: Data Transmission

Notes on [Figure 2-67](#):

- When a serial data stream of 1 Gb/s is required, then the PLL in the same I/O bank as the TX\_BITSLICES must deliver a 1 GHz clock at the PLL\_CLK input of the BITSlice\_CONTROL.
- When the TX\_BITSLICES are operated in 8-bit mode (DATA\_WIDTH = 8), the data must be presented at the D inputs of the TX\_BITSLICE with a 125 MHz clock (1 GHz/DIV\_MODE = 4).



---

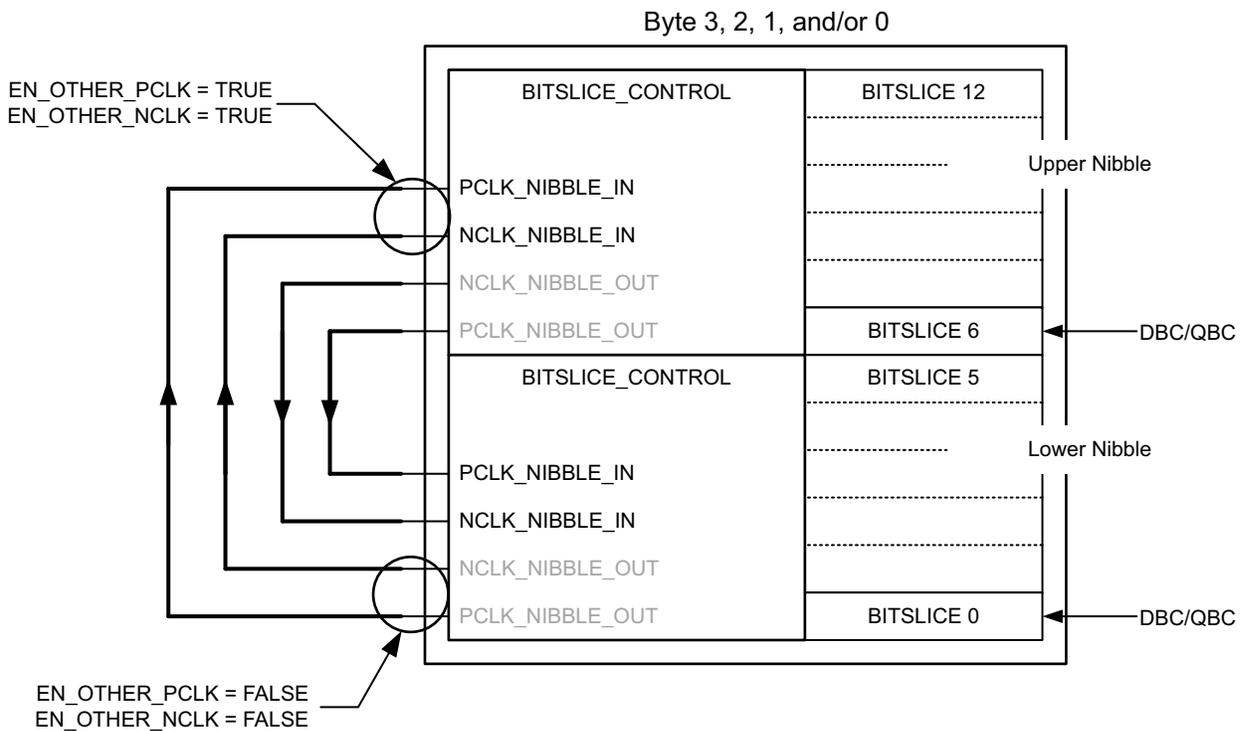
**TIP:** Using the `OUTPUT_PHASE_90` attribute allows the serial output of a transmitter bit slice to be phase-shifted by 90 degrees. This function is normally used to generate center-aligned interface clocks. When using `OUTPUT_PHASE_90`, `DELAY_VALUE` should not be used.

---

### Inter-Nibble Clocking

Each nibble has a possible clock input into BITSlice\_0. Two neighboring nibbles can share one of these clock's inputs and increase the number of possible data inputs by joining together as a byte ([Figure 2-68](#)).

One nibble passes the clock from its BITSlice\_0 input to the other nibble through dedicated inter-nibble clock routing from P(N)CLK\_NIBBLE\_OUT to the clock inputs at the other nibble P(N)CLK\_NIBBLE\_IN. This routing is enabled through attributes (EN\_OTHER\_P(N)\_CLK) set at the BITSlice\_CONTROLS of both nibbles joined together as byte.



X16052-022216

Figure 2-68: Inter-Nibble Clocking

Using Figure 2-68 as an example, assuming the lower nibble's BITSlice\_0 is setup as a clock input (DATA\_TYPE = DATA\_AND\_CLOCK) and the upper nibble BITSlice\_0 is used as data input, the attributes of both nibbles should be set as follows:

Upper nibble

- EN\_OTHER\_PCLK = TRUE
- EN\_OTHER\_NCLK = TRUE

Lower nibble

- EN\_OTHER\_PCLK = FALSE
- EN\_OTHER\_NCLK = FALSE

The clock passes through the lower BITSlice\_0 to the P(N)CLK\_NIBBLE\_OUT and into the upper nibble P(N)CLK\_NIBBLE\_IN inputs for clocking of the bit slices in the upper nibble.

When BITSlice\_0 of the upper nibble is used as a clock input, pass the clock to the lower nibble by using the P(N)CLK\_NIBBLE\_OUT pins of the upper nibble and the P(N)CLK\_NIBBLE\_IN pins of the lower nibble, and the attributes should be set as follows:

Upper nibble

- EN\_OTHER\_PCLK = FALSE
- EN\_OTHER\_NCLK = FALSE

Lower nibble

- EN\_OTHER\_PCLK = TRUE
- EN\_OTHER\_NCLK = TRUE

**Note:** It is possible to enable inter-nibble clock routing in both directions so that bit slices in both nibbles can be used by one of the clocks connected to the byte. In other words, it is possible to capture data in the lower nibble at the clock applied to the upper nibble BITSlice\_0 while at the same time data can be captured in the upper nibble at the clock connected to the lower nibble BITSlice\_0.



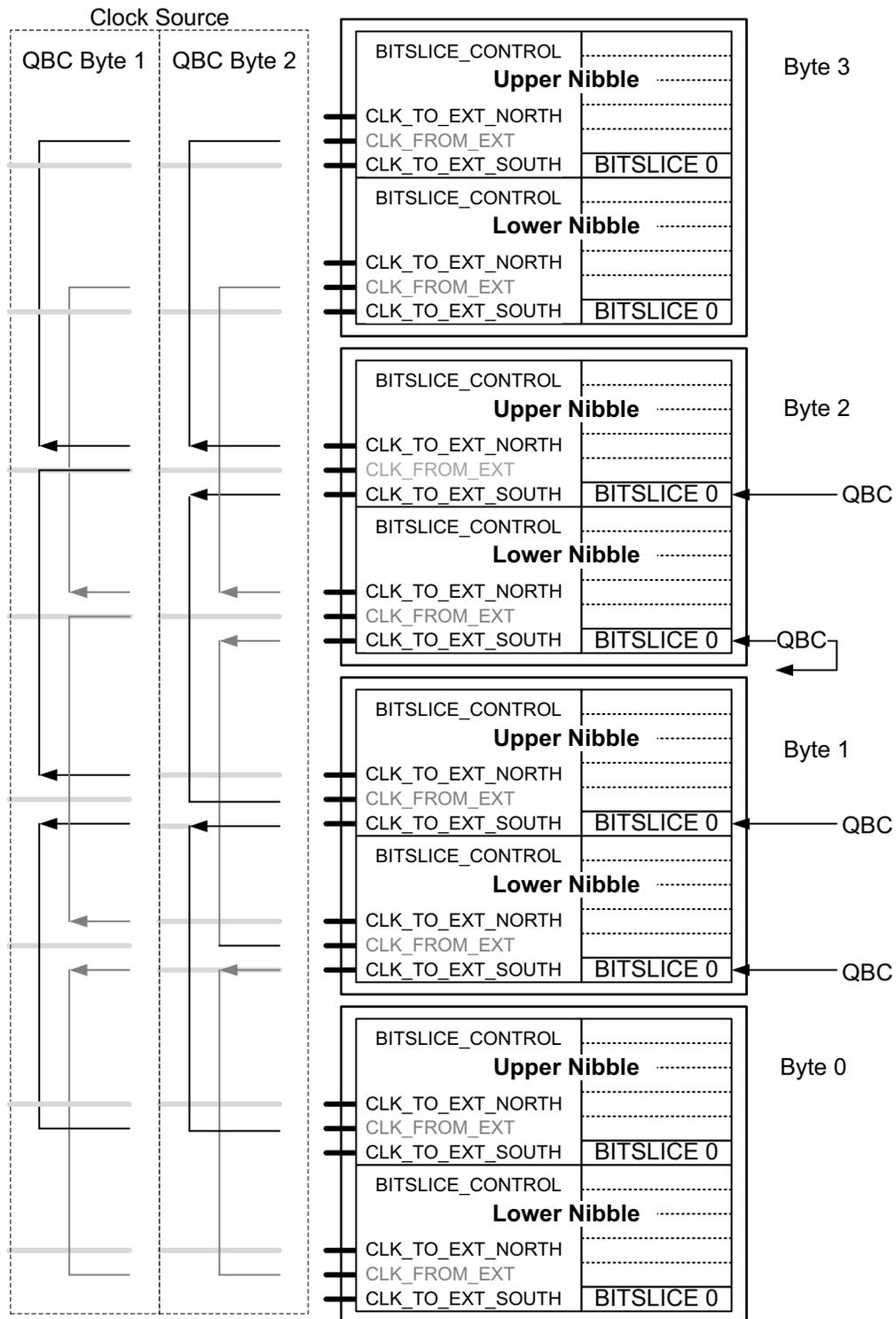
---

**TIP:** When using multiple nibbles in a design, always connect the inter-nibble clocks, as shown in [Figure 2-68](#). When inter-nibble clocking is necessary, enable or disable an attribute.

---

### Inter-Byte Clocking

Inter-byte clocking allows clock sharing between a clock arriving at a BITSlice\_0 of one nibble and the nibbles in the same position of other bytes ([Figure 2-68](#)). Sharing of the input or sample clock is done through CLK\_TO\_EXT\_NORTH(SOUTH) output pins and CLK\_FROM\_EXT input pins in BITSlice\_CONTROL components of other bytes. Inter-byte clocking is started by setting attributes EN\_CLK\_TO\_EXT\_NORTH(SOUTH).

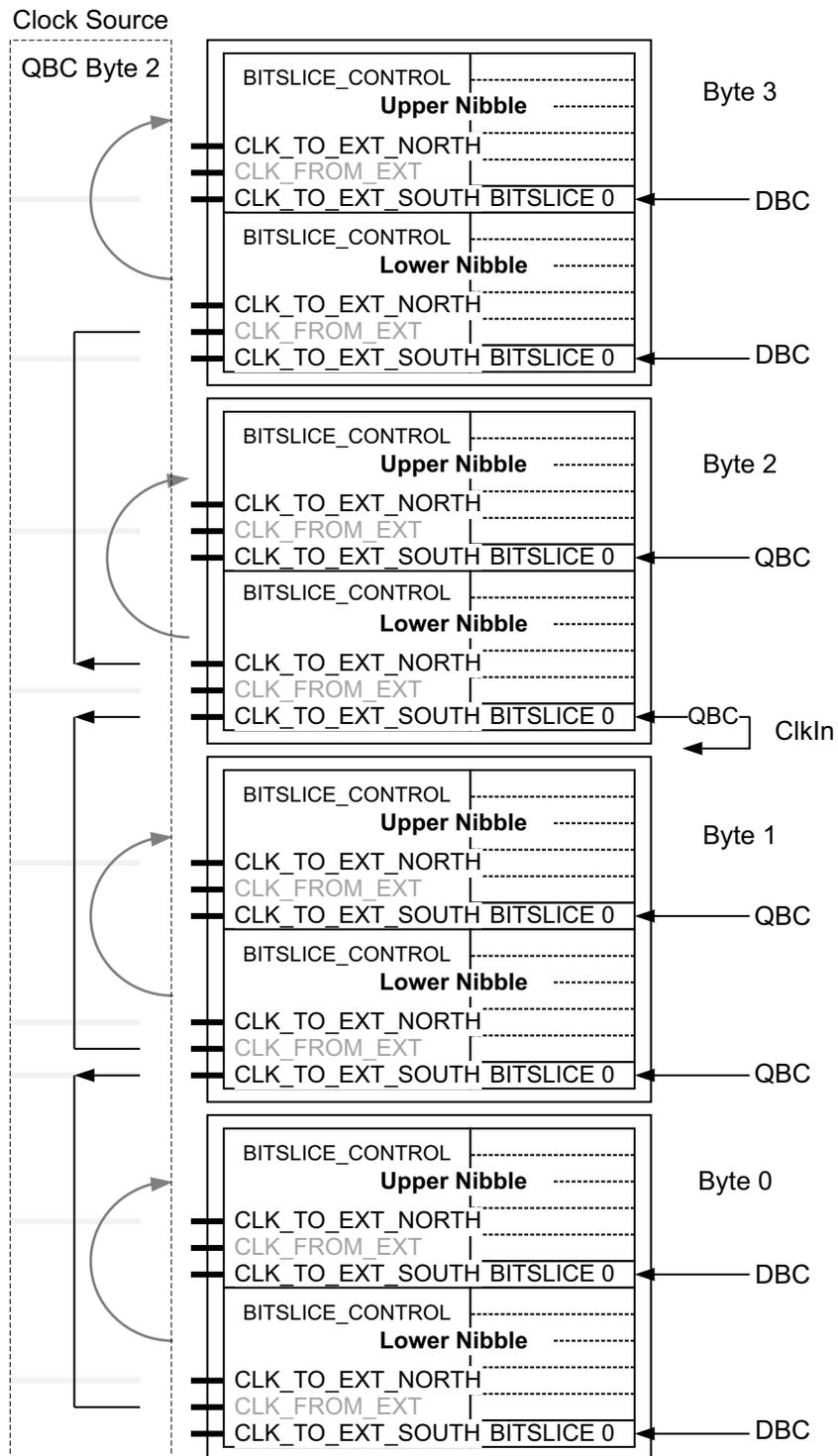


X16053-022216

Figure 2-69: Inter-Byte Clocking

Note: Figure 2-69 and Figure 2-70 do not show BITSlice\_6 of the upper nibble.

Inter-nibble and inter-byte clocking can be combined to serve all bit slices in an I/O bank with the same clock (see Figure 2-70).



X16054-060916

Figure 2-70: Combination of Inter-Nibble and Inter-Byte Clocking

Notes on [Figure 2-70](#).

- Assume that byte\_2, lower nibble, RX\_BITSLICE\_0 is set up as a clock or strobe input.
- Byte\_2, upper nibble must be inter-nibble clock-enabled.
- Because the lower nibble of byte\_2 is used as clock input, and all nibbles are used in multi-byte setup, for all bytes the lower nibble must supply the clock to the upper nibble for inter-nibble clocking.
- Byte\_3, lower nibble, gets a clock from the CLK\_TO\_EXT\_NORTH output of the lower nibble of byte 2 on its CLK\_FROM\_EXT input.
- The lower byte south (down) of byte\_2, lower nibble supplies a clock from the CLK\_TO\_EXT\_SOUTH output to the CLK\_FROM\_EXT input of the lower nibble of byte 1.
- To reach the lower nibble of byte 0, the CLK\_TO\_EXT\_SOUTH output of byte 1 must be connected to the CLK\_FROM\_EXT input of the lower nibble of byte 0. A jump over is made inside byte 1 between the CLK\_FROM\_EXT and CLK\_TO\_EXT\_SOUTH pins. As such, all receivers in all nibbles are configured to receive data from a forwarded version (via inter-byte/nibble) of the strobe sourced from the byte\_2 upper nibble.

The attribute setups for this example are listed in [Table 2-33](#) and [Table 2-34](#).

**Table 2-33: Attributes for Inter-Byte Clocking Example**

Byte	Attribute	Type
Byte 3	EN_CLK_TO_EXT_NORTH	DISABLE
	EN_CLK_TO_EXT_SOUTH	DISABLE
Byte 2	EN_CLK_TO_EXT_NORTH	ENABLE
	EN_CLK_TO_EXT_SOUTH	ENABLE
Byte 1	EN_CLK_TO_EXT_NORTH	DISABLE
	EN_CLK_TO_EXT_SOUTH	ENABLE
Byte 0	EN_CLK_TO_EXT_NORTH	DISABLE
	EN_CLK_TO_EXT_SOUTH	DISABLE

**Notes:**

1. Unused CLK\_FROM\_EXT pins must be tied High.

For all nibbles, inter-nibble clocking must be enabled as listed in [Table 2-34](#).

**Table 2-34: Enabling Inter-Nibble Clocking**

Nibble	Attribute	Type
Upper Nibble	EN_OTHER_NCLK	TRUE
	EN_OTHER_PCLK	TRUE
Lower Nibble	EN_OTHER_NCLK	FALSE
	EN_OTHER_PCLK	FALSE

### Inter-byte Clocking Precautions

For the following or similar conditions, the arriving clock in the lower nibble must be passed via inter-byte clock routes to the lower nibble of an upper or lower byte and then routed in the byte to the upper nibble by inter-nibble clocking paths as shown in [Figure 2-71](#):

- It is assumed that the BISC controller in the BITSlice\_CONTROL primitive is turned on by SELF\_CALIBRATE = ENABLE.
- The received clock arrives at the lower nibble BITSlice\_0 of byte\_2 of the I/O bank.
- The bit slices used to capture data are placed in the upper nibble of byte\_0 in the I/O bank.

In these cases, observe this condition:

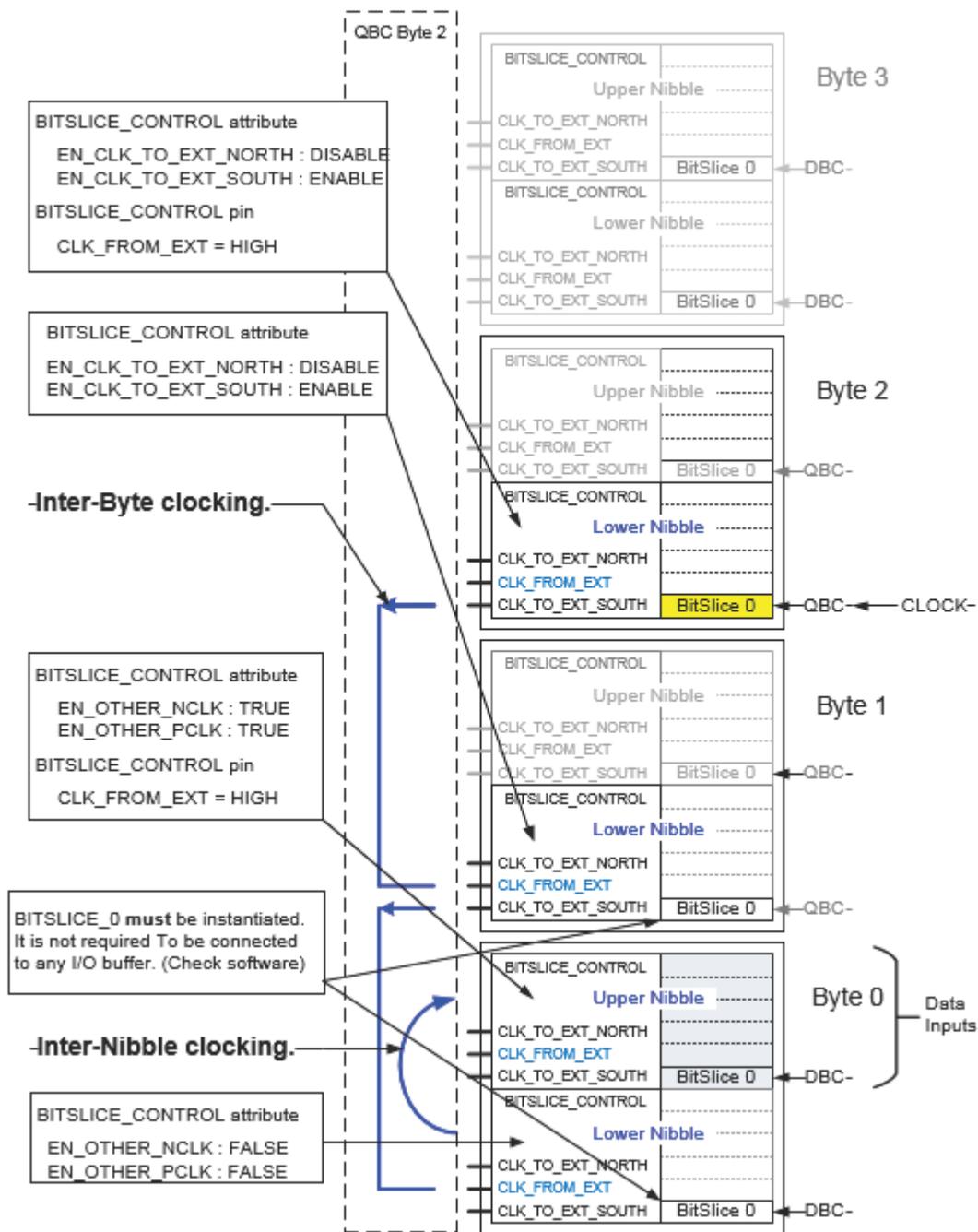
- For a design using inter-byte clocking and SELF\_CALIBRATE is enabled, a BITSlice\_0 must be instantiated in the nibble receiving the inter-byte clock.
- The BITSlice\_0 that needs to be instantiated in the above-mentioned case must be configured with the attribute DATA\_TYPE set to DATA.
- The instantiated BITSlice\_0 can be used for data capture.
- When the instantiated BITSlice\_0 is not used at all, you **must** connect an input buffer to the bit slice for correct software behavior.



---

**CAUTION!** All bit slices in a nibble with an instantiated BITSlice\_0 used to pass a clock input from CLK\_FROM\_EXT use the clock of the CLK\_FROM\_EXT input as a data capture clock. The same applies when the CLK\_FROM\_EXT is routed by inter-nibble clocking to an upper or lower nibble. For the upper nibble, BITSlice\_0 is the equivalent of BITSlice\_6 for a byte group. See [Figure 2-3](#) for an explanation of bitslice numbering within a byte and nibble.

---



X16956-060916

Figure 2-71: Inter-byte Clocking with BITSlice\_0 Bypass

Example 1:

- The clock arrives in the lower nibble of byte\_2 and data inputs are placed in the upper nibble of byte\_0. No other bit slices in the I/O bank are used.

- The clock must pass from CLK\_TO\_EXT\_SOUTH of the lower nibble in byte\_2 as a pass-through in the lower nibble of byte\_1 and then route to the CLK\_FROM\_EXT in the lower nibble of byte\_0.
- Because no bit slices are used in the lower nibble of byte\_1, a RXTX\_BITSLICE or RX\_BITSLICE and BITSlice\_CONTROL must be instantiated in bit slice position zero of the nibble. The bit slice positioned in bit slice zero must be configured with DATA\_TYPE = DATA.
- The instantiated bit slice and BITSlice\_CONTROL do not need any connection except for the PLL\_CLK, RIU\_CLK, and input delay line CLK.
- It might be necessary to LOC the bit slice and BITSlice\_CONTROL in the FPGA I/O architecture.
- The inter-nibble clock must be used to route the clock from the lower nibble into the upper nibble of byte\_0.

#### Example 2:

- This same situation is similar to example 1, but the clock arrives in the upper nibble of byte\_2.
- The clock must pass from CLK\_TO\_EXT\_SOUTH of the upper nibble in byte\_2 as a pass-through in the upper nibble of byte\_1 and then route to the CLK\_FROM\_EXT in the upper nibble of byte\_0.
- Because no bit slices are used in the upper nibble of byte\_1, a RXTX\_BITSLICE or RX\_BITSLICE and BITSlice\_CONTROL must be instantiated in bit slice position zero of the nibble.
- The instantiated bit slice and BITSlice\_CONTROL do not need any connection except for the PLL\_CLK, RIU\_CLK, and input delay line CLK.
- It might be necessary to LOC the bit slice and BITSlice\_CONTROL in the FPGA I/O architecture.
- No inter-nibble clock is needed because the inter-byte clock arrives at the upper nibble of byte\_0.

#### Example 3:

- This same situation is similar to example 1, but all bit slices in all nibbles of bytes\_2, \_1, and \_0 are used except BITSlice\_0 in the lower nibble of byte\_1.
- The inter-nibble clock must be used to route the clock from the lower nibble into the upper nibble of byte\_2.
- The inter-byte clock must be used to route the clock from the CLK\_TO\_EXT\_SOUTH of the lower nibble in byte\_2 to the CLK\_FROM\_EXT of the lower nibble of byte\_1.
- Because BITSlice\_0 of that nibble is not used, one must be instantiated.

- A BITSlice\_CONTROL does not need to be instantiated because one is used for the other bit slices in the nibble.
- The inter-nibble clock must be used to route the clock from the lower to the upper nibble of byte\_1.
- The instantiated BITSlice\_0 does not need a LOC attribute because the BITSlice\_CONTROL is already used for the other bit slices in the nibble.
- The inter-byte clock must be used to route the clock from the CLK\_TO\_EXT\_SOUTH of the lower nibble in byte\_1 to the CLK\_FROM\_EXT of the lower nibble of byte\_0.
- The inter-nibble clock must be used to route the clock from the lower to the upper nibble of byte\_0.

### Built-in Self-Calibration

The built-in self-calibration (BISC) block is a digital control and calibration block inside the BITSlice\_CONTROL component based on a delay-locked loop (DLL) circuit and on a digital delay-line phase detector. The BISC controller calculates the desired tap values for the digital delay lines and keeps track of these values over voltage and temperature drift. By default, when the correct attributes are set after instantiation of the BITSlice\_CONTROL primitive, the BISC controller reports the status of DLY and VTC back to the logic after tuning the used delay lines (Figure 2-72).

The BISC controller registers can also be accessed through the register interface unit (RIU). This gives you full control over the BISC process. It is possible to initiate or influence a BISC run and read back or change registers the BISC controller has filled or modified.

The EN\_VTC signal of any connected bit slice with a TIME delay that needs to be calibrated must have an individual EN\_VTC signal asserted (High) during initial self-calibration and after DLY\_RDY is High during VT calibration.

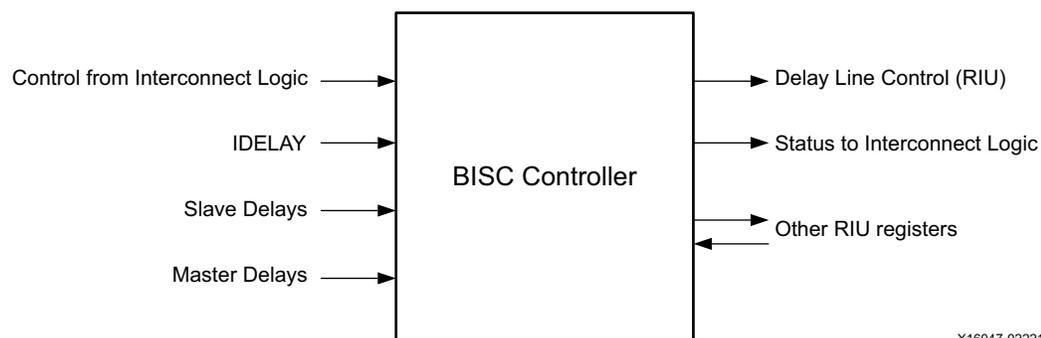


Figure 2-72: Block Diagram of the BISC Controller and Connections

## Ports and Attributes Influencing BISC

Table 2-35 highlights the BITSlice\_CONTROL ports and attributes involved in the BISC process.

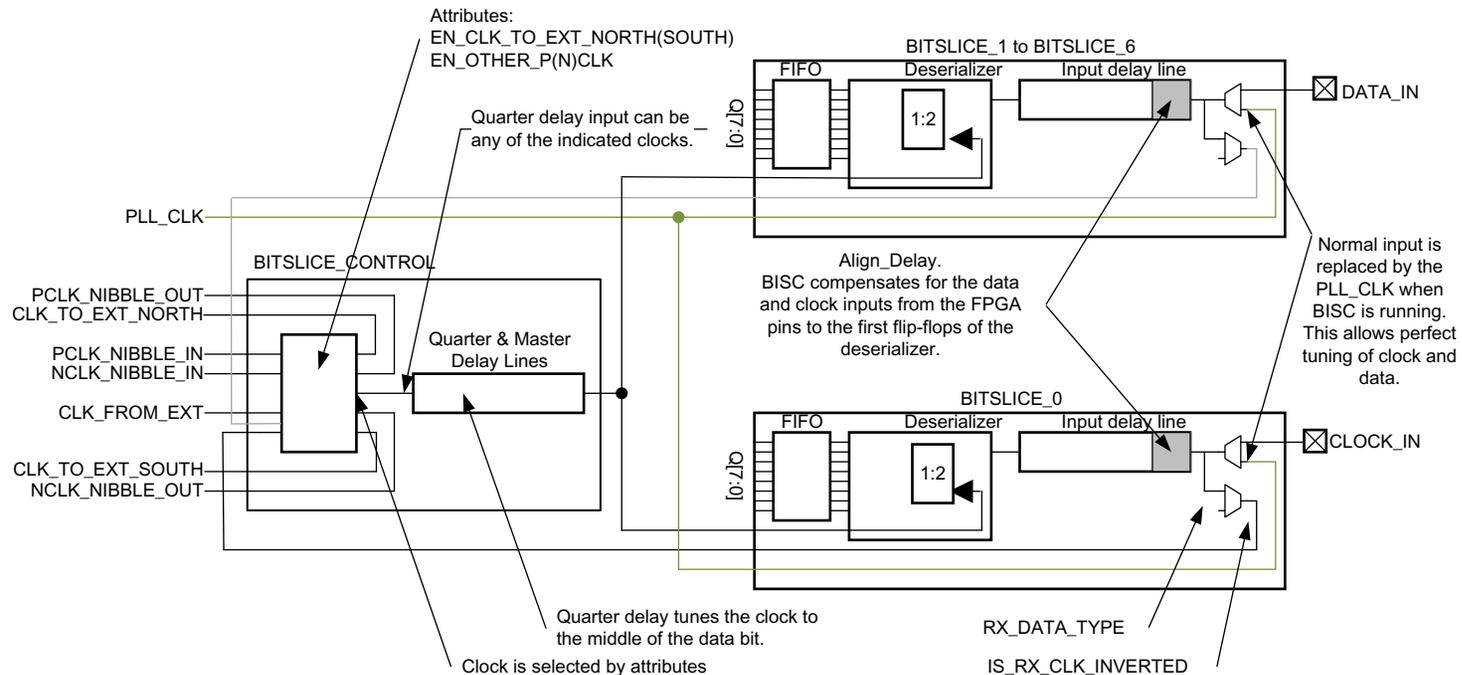
Table 2-35: BITSlice\_CONTROL Ports and Attributes for the BISC Process

Pins	I/O	Type	Description
<b>Logic Control</b>			
EN_VTC	In	Data	Enable VT tracking.
<b>Status</b>			
VTC_RDY	Out	Data	Nibble ready for VT calibration
DLY_RDY	Out	Data	Nibble delay line calibration is complete.
<b>RIU</b>			
RIU_CLK	In	Clock	Clock from interconnect logic. The RIU clock must be connected in for the BISC process to complete.
RIU_ADDR[5:0]	In	Data	Register address.
RIU_WR_DATA[15:0]	In	Data	Data write to register.
RIU_RD_DATA[15:0]	Out	Data	Data read from register.
RIU_VALID	Out	Data	Status indicating if BISC is accessing RIU registers.
RIU_WR_EN	In	Enable	Register write enable (active-High).
RIU_NIBBLE_SEL	In	Data	Nibble in byte select. This signal must be High to perform read/write to the nibble.
<b>Attributes</b>			
IDLY_VT_TRACK			Enable input delay line VT tracking.
ODLY_VT_TRACK			Enable output delay line VT tracking.
QDLY_VT_TRACK			Enable slave quarter delay VT tracking.
ROUNDING_FACTOR			Value to scale the VT tracking. This attribute has a default value that normally does not need to be changed.
SELF_CALIBRATE			Start a self-calibrate cycle.
RIU Registers			Read the RIU paragraph.

### BISC Calibration Steps

The BISC controller tunes the delay line in TIME mode of each connected bit slice separately. BISC starts with the bit slice in position 0 and works its way up the nibble. When done, the DLY\_RDY status signal is pulled High.

When the SELF\_CALIBRATE attribute is set or when the RIU CALIB\_CTRL register CALIBRATE (bit\_0) and/or CALIBRATE\_EN (bit\_3:10) are used, the BISC controller runs three basic steps when tuning the delay lines. All steps are explained in the subsequent sections and shown in Figure 2-73.



X16790-071917

Figure 2-73: Delay Calibration by BISC

When bit slice delay lines are used in COUNT mode, the BISC calibration cycle runs but ignores the calibration and VT compensation for the primitives in COUNT mode delay lines.

In TIME mode, the auto-VT tracking can be turned on or off using attributes (I, O, Q DLY\_VT\_TRACK) or using bits in the RIU CALIB\_CNTRL register.

### Step 1: Alignment

Alignment is the first step in the BISC process and is necessary to maximize the data eye in the bit slices by removing internal skew and compensating for the internal skew between clock and data insertion delays of input paths to first capture flip-flops.

**Note:** PCB (trace) delay, package influences on the input signals, and deskewing of output signals are not handled by the BISC.

This delay is called *Align\_Delay* and is used in the calculations provided in the “Native Delay Mode Usage” paragraph. This delay typically takes between 45 and 65 taps of the input delay line.

This step also manages possible duty cycle distortion (DCD) and the initial voltage and temperature calibration of the data and capture clock signals.

### Step 2: Delay Calibration

This step walks through the input and/or output delay of each used bit slice to calculate the number of taps required to provide the delay requested by DELAY\_VALUE attribute. These calculated delay taps are stored in the RIU registers (ODELAYxx and IDELAYxx). The same is done for the BITSlice\_CONTROL available quarter delay lines (PQTR/NQTR) to provide the 90-degree equivalent delay when RX\_CLK\_PHASE\_P(N) = SHIFT\_90. These values are stored in the RIU PQTR and NQTR registers.

The BISC controller signals the interconnect logic when the delay line calibration mechanism for all used bit slices is done by asserting the DLY\_RDY signal.

### Step 3: Continuous VT Tracking

When turned on by an attribute or RIU register, at this step the BISC controller starts a continuous operation of the last step in the calibration part of BISC, VT tracking.

The automated tracking uses a round-robin scheme to keep every used delay line up to date without interfering or disrupting the normal operation mode of that bit slice.

During calibration, the BISC can modify or write to certain RIU registers. The register values altered by BISC include TX\_DATA\_PHASE, BS\_DQ\_EN, BS\_DQS\_EN, EN\_PDQS, EN\_NDQS, INVERT\_RX\_CLK, SERIAL\_MODE, TX\_GATE, and RX\_GATE.

Each nibble has its own BITSlice\_CONTROL component and therefore its own BISC controller. When multiple nibbles or bytes are used and have shared clocks through the inter-nibble or inter-byte clock resources, each nibble can calibrate the bit slices used in that nibble. Each nibble can require different times to finish the same calibration step due to different environments and configurations. Through inter-nibble or inter-byte communication, each nibble should not proceed to the next calibration step until all nibbles finish the current calibration step.

**Note:** The reset for all used BITSlice\_CONTROLS within a bank must be released at the same time due to the DLY\_RDY connections between the BITSlice\_CONTROLS. For example, if a bank has two different interfaces, both interfaces should be controlled by a single reset to ensure calibration completes. Failure to do so might result in DLY\_RDY for one of the interfaces not asserting.

### Register Interface Unit (RIU)

Using the register interface unit (RIU), the BITSlice\_CONTROL primitive can be turned into a processor peripheral block. The RIU interface is a set of 64 read/write, 16-bit registers, acting as a dynamically accessible processor peripheral interface providing full control over every function and feature of a nibble: control of all input, output, 3-state, and all delay lines (input, output and quarter), voltage and temperature (VT) tracking, clocking options, and built-in self-calibration (BISC). The RIU interface is represented in the BITSlice\_CONTROL component as shown in [Figure 2-74](#).

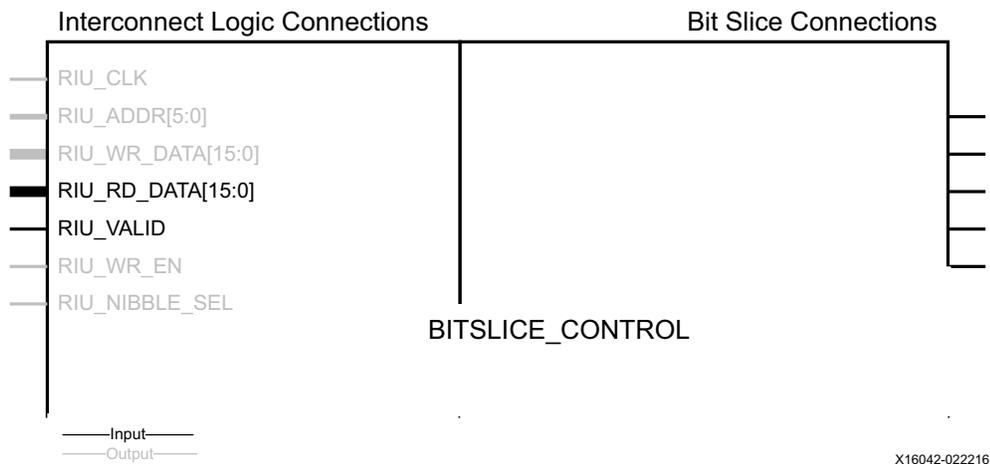


Figure 2-74: RIU in the BITSlice\_CONTROL

Each nibble has its own BITSlice\_CONTROL and therefore its own RIU interface. Two nibbles can be combined in a byte, so a byte can have two RIU interfaces. To allow easy control of both RIU interfaces in a byte, a RIU\_OR primitive exists.

The RIU\_OR primitive combines both nibble RIU interfaces of a byte into a single RIU interface. [Figure 2-75](#) and [Table 2-36](#) show the RIU\_OR primitive and its pins. A possible setup joining two nibble RIU into a byte-wide RIU using the RIU\_OR primitive is shown in [Figure 2-76](#). The RIU\_NIBBLE\_SEL pin of each RIU is used as the MSB address, putting the upper nibble in the upper address space.

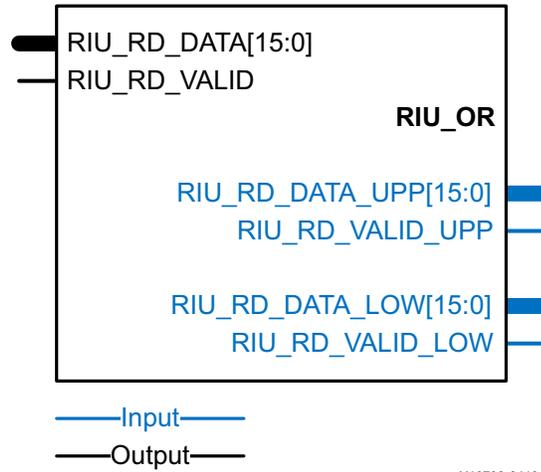


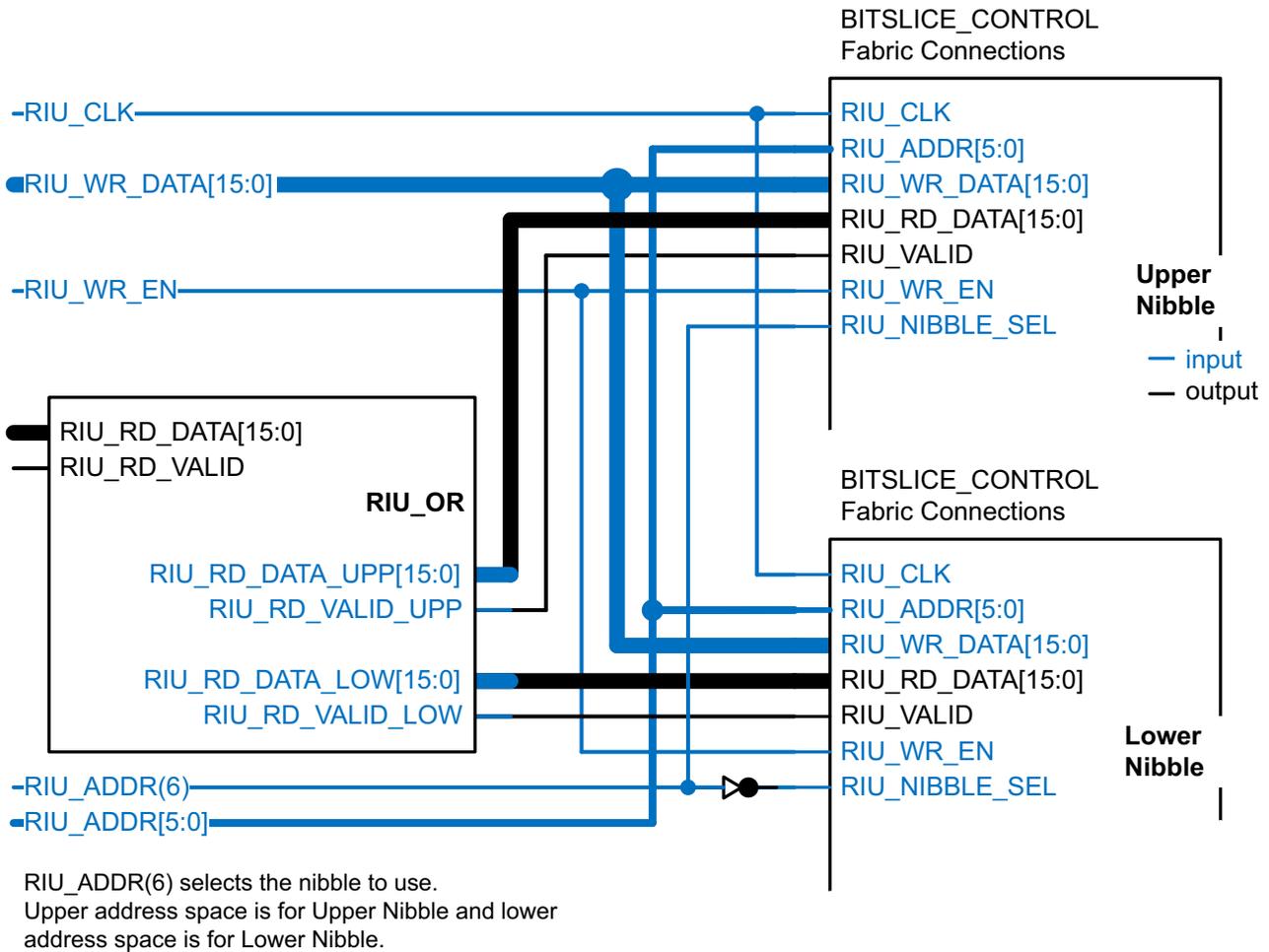
Figure 2-75: RIU Primitive

Table 2-36: RIU\_OR Ports

Port	I/O	Description
RIU_RD_DATA_UPP[15:0]	Input	Connect to RIU_RD_DATA of the upper nibble BITSlice_CONTROL.
RIU_RD_DATA_LOW[15:0]	Input	Connect to RIU_RD_DATA of the lower nibble BITSlice_CONTROL.
RIU_RD_VALID_UPP	Input	Connect to RIU_VALID of the upper nibble BITSlice_CONTROL.
RIU_RD_VALID_LOW	Input	Connect to RIU_VALID of the lower nibble BITSlice_CONTROL.
RIU_RD_DATA[15:0]	Output	Combined RIU data bus to interconnect logic.
RIU_RD_VALID	Output	Combined RIU read valid signal to interconnect logic.

Table 2-37: RIU\_OR Attribute

Attribute	Values	Default	Type	Description
SIM_DEVICE	ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2	ULTRASCALE	String	Sets the device version (ULTRASCALE, ULTRASCALE_PLUS, ULTRASCALE_PLUS_ES1, ULTRASCALE_PLUS_ES2)

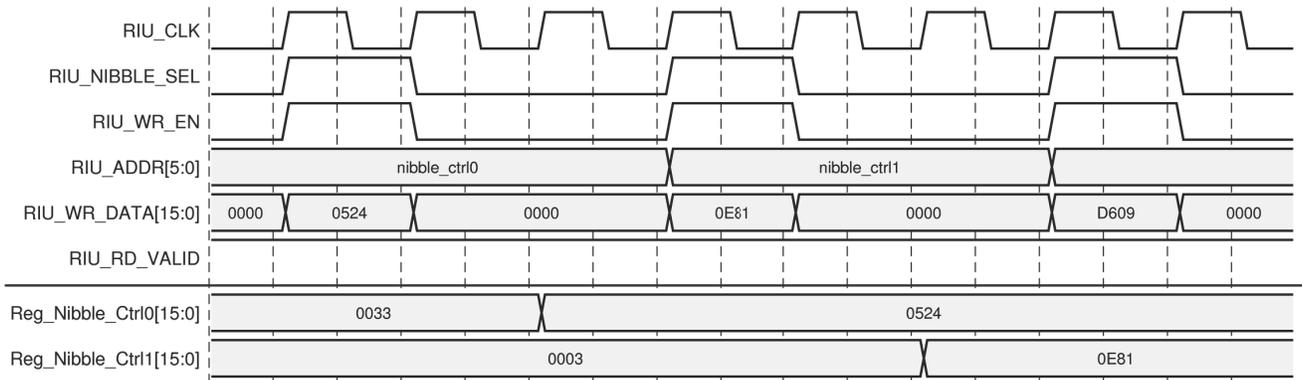


X16793-041316

Figure 2-76: RIU\_OR Combining Two RIU Ports

**RIU Write Actions**

A RIU register write action to a RIU register is when RIU\_WR\_EN, RIU\_NIBBLE\_SEL, and RIU\_ADDR are asserted (Figure 2-77). The data is written into the RIU registers two clocks after RIU\_WR\_EN. Interconnect logic can issue RIU write only when RIU\_RD\_VALID is High.



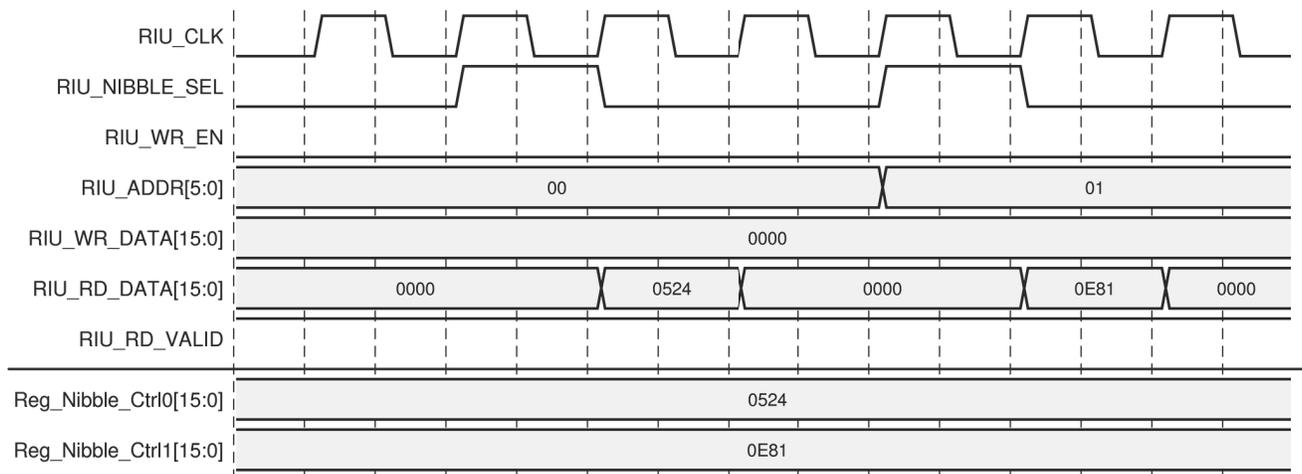
X16043-062316

Figure 2-77: RIU Write

Writing to the RIU registers requires the RIU logic to arbitrate between BISC accesses and access from the interconnect logic. BISC access to RIU registers always has precedence over interconnect logic access, hence an interconnect logic transaction is stored and resumed after BISC access.

**RIU Read Actions**

In the RIU read, data from the RIU is driven out on the **RIU\_RD\_DATA** bus that depends on **RIU\_ADDR** and the **RIU\_NIBBLE\_SEL** signal (Figure 2-78). When **RIU\_NIBBLE\_SEL** is asserted, data is presented one clock cycle later on the **RIU\_RD\_DATA** bus.



X16044-022216

Figure 2-78: RIU Read

Figure 2-79 shows a back to back write and read operation.

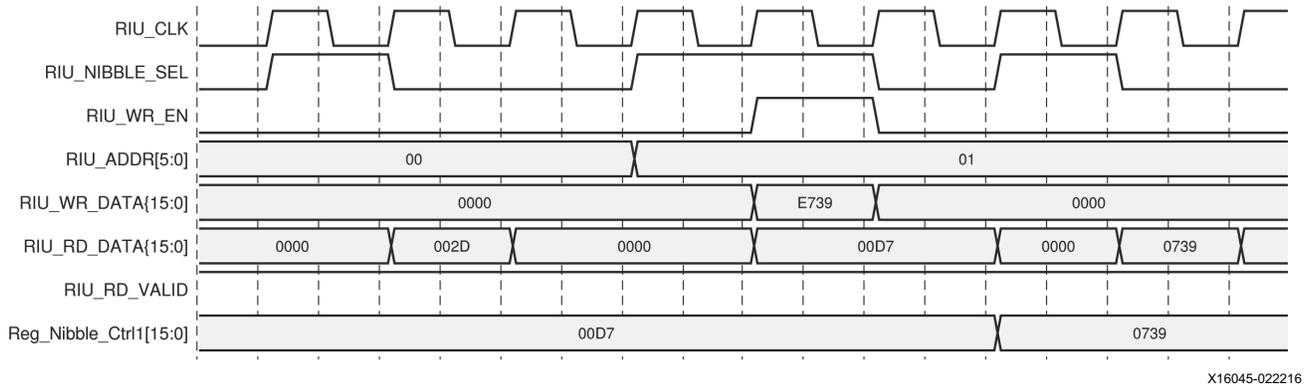


Figure 2-79: RIU Read Modify Write

In Figure 2-79, registers Nibble\_Ctrl0 (0x00) and Nibble\_Ctrl1 (0x01) are accessed. RIU\_RD\_DATA is shown as 0x0000, 0x002D, and 0x00D7 during read and write cycles and the content of the RIU\_WR\_DATA bus is 0xE739. The contents of register Nibble\_Ctrl1 shows 0x0739 after two cycles of latency and the data appears on the RIU\_RD\_DATA bus with one clock cycle latency.

### RIU Ports

Table 2-38 lists register interface unit ports.

Table 2-38: RIU Ports

Pins	I/O	Type	Description
RIU_CLK	Input	Clock	Clock from interconnect logic. Clock for the RIU interface peripheral. This clock is independent of all other clocks of the BITSlice_CONTROL. The RIU clock needs to be connected when BISC is enabled.
RIU_ADDR[5:0]	Input	Data	Register address. The address input bus provides a register address for the register interface. The address value on this bus specifies the configuration bits that are written or read with the next RIU_CLK cycle. When not used, all bits must be assigned zeros.

Table 2-38: RIU Ports (Cont'd)

Pins	I/O	Type	Description
RIU_WR_DATA[15:0]	Input	Data	<p>Data write to register.</p> <p>This input bus provides data. The value of this bus is written to the configuration cells of the register interface. The data is presented in the cycle that RIU_WR_EN and RIU_NIBBLE_SEL are active. The data is captured in a shadow register and written at a later time.</p> <p>RIU_VALID indicates when the RIU port is ready to accept another write. When not used, all bits must be set to zero.</p>
RIU_RD_DATA[15:0]	Output	Data	<p>Data read from register.</p> <p>This output bus provides RIU data.</p> <p>The value of this bus is a representation of the register bits addressed by RIU_ADDR. See <a href="#">Register Definitions and Addresses, page 329</a>.</p> <p>The data is presented in the next cycle when RIU_NIBBLE_SEL is active and RIU_WR_EN is 0.</p> <p>When not used, this output bus must be left floating.</p>
RIU_VALID	Output	Data	<p>Status indicating if BISC is accessing RIU registers.</p> <p>This signal indicates the status when RIU accesses are made from the interconnect logic while the internal BISC state machines also are accessing the RIU registers.</p> <p>During a collision (i.e., an RIU write access occurs during a BISC write access), the RIU_VALID signal is deasserted. The interconnect logic write access still succeeds, but not until RIU_VALID is asserted. No further action is required on the interconnect logic side, except that no further RIU accesses are possible until RIU_VALID is asserted. In addition to collisions, the RIU_VALID deasserts when writing to the RL_DLY_RNK0, RL_DLY_RNK1, RL_DLY_RNK2, or RL_DLY_RNK3 registers. These registers are unique because it takes more than two cycles for an RIU write to update them. Therefore back-to-back accesses to these registers are impossible.</p>
RIU_WR_EN	Input	Enable	<p>Register write enable (active-High).</p> <p>This signal and RIU_NIBBLE_SEL must be asserted High to write a register in an RIU interface.</p>
RIU_NIBBLE_SEL	Input	Data	<p>Nibble in byte select.</p> <p>An I/O bank is constructed from four bytes. Each byte contains two nibbles. Each nibble contains a BITSlice_CONTROL component for control of all RX or TX BITSlices of the nibble. This signal is used to select a nibble RIU in a byte.</p>

### Register Definitions and Addresses

The following tables provide additional details for the RIU register definitions. Many of the settings are used for memory applications (MIG) and are provided for completeness. For information on how the Memory IP uses the RIU registers, see *UltraScale Architecture-Based FPGAs Memory IP LogiCORE IP Product Guide* (PG150) [Ref 14].

Table 2-39 through Table 2-61 list register bit descriptions.

Table 2-39: Register Bit Description (NIBBLE\_CTRL0)

NIBBLE_CTRL0													ADDR: 0x00			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default					0	0	0	0		0	0	0	0	0	1	1
Access					R/W	R/W	R	R/W		R/W	R/W	R/W	R/W	R/W	R/W	R/W
15:12	Reserved															
11	DIS_DYN_MODE_RX: Disable gate delay dynamic mode (MIG) or enable receive delay line updates using RIU. See <i>UltraScale Architecture-Based FPGAs Memory IP LogiCORE IP Product Guide</i> (PG150) for details [Ref 14].															
10	DIS_DYN_MODE_TX: Disable output delay dynamic mode (MIG) or enable transmit delay lines updates using RIU.															
9	GT_STATUS: Monitor gate placement relative to strobe/clock.															
8	CLR_GATE: Used for strobe/clock gate training. Resets the gating logic.															
7	Reserved															
6	RXGATE_EXTEND: Enable preamble extension for DQS_BIAS.															
5	RX_GATE: Enable receive strobe/clock gating.															
4	TX_GATE: Enable transmit clock gating.															
3	SERIAL_MODE: Set to 1 to enable PLL_CLK as a sample clock. This mode is used for data sampling of a serial bitstream such as SGMI. BISC manipulates this bit during initial BISC calibration. So if SERIAL_MODE is selected using the RIU register bit instead of the attribute setting, you must set this bit again after DLY_RDY assertion of High. Otherwise the PHY will not be in serial mode operation.															
2	INVERT_RX_CLK: Invert the clock path from IOB to RX_BITSLICE. This is for clock path through read DQS_IN.															
1	EN_NDQS: Set to 1 to pass a source-synchronous clock NCLK_NIBBLE_OUT from the other nibble's DQS gating circuit through NQTR slave delay. Set to 0 to pass a clock from the present nibble's DQS gating circuit through the NQTR slave delay.															
0	EN_PDQS: Set to 1 to pass a source-synchronous clock PCLK_NIBBLE_OUT from the other nibble's DQS gating circuit through PQTR slave delay. Set to 0 to pass a clock from the present nibble's DQS gating circuit through the PQTR slave delay.															

Table 2-40: Register Bit Description (NIBBLE\_CTRL1)

NIBBLE_CTRL1													ADDR: 0x01			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default					0	0	0	0	0	0	0	0	0	0	1	1
Access					R/W	R/W	R/W	R/W								
<b>15:12</b>	Reserved															
<b>11:2</b>	TX_DATA_PHASE: When set, shifts the output data by 90 degrees. [11]: Master data bit slice [10]: Master clock bit slice [9]: 3-state bit slice [8:2]: TX_BITSLICES															
<b>1</b>	RX_CLK_PHASE_N: When set, shifts any input clock or strobe after entering the bit slice from the I/O by 90 degrees. Leaving this bit at 0 maintains a 0 degrees phase shift clock/strobe.															
<b>0</b>	RX_CLK_PHASE_P: When set, shifts any input clock or strobe after entering the bit slice from the I/O by 90 degrees. Leaving this bit at 0 maintains a 0 degrees phase shift clock/strobe.															

Table 2-41: Register Bit Description (CALIB\_CTRL)

CALIB_CTRL													ADDR: 0x02			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Access	R	R/W	R/W	R	R	R/W	R/W	R/W	R/W							
<b>15</b>	PAUSE_RDY: VT tracking state machine pause indication.															
<b>14</b>	DIS_VTTRACK_QTR: Enable or disable auto VT tracking for the slave PQTR/NQTR delays.															
<b>13</b>	BSC_RESET: Software reset for BISC.															
<b>12</b>	PHY_RDY: PHY calibration complete status. This is the RIU equivalent of the VTC_RDY signal.															
<b>11</b>	FIXDLY_RDY: Fixed delay calibration complete status. This is the RIU equivalent of the DLY_RDY signal.															
<b>3:10</b>	CALIBRATE_EN: Inject the reference clock/PLL CLK into per-bit datapaths for the receive channels to perform self-calibration. CALIBRATE_EN[6:0]: Per RX_BITSLICE. CALIBRATE_EN(7): Master.															
<b>2</b>	DIS_VTTRACK_OBIT: Enable or disable auto VT tracking for all output delay lines.															
<b>1</b>	DIS_VTTRACK_IBIT: Enable or disable auto VT tracking for all input delay lines.															
<b>0</b>	CALIBRATE: Turn self-calibration on or off.															

Table 2-42: Register Bit Description (BS\_CTRL)

BS_CTRL													ADDR: 0x05			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access	R	R/W	R/W	R	R	R/W	R/W	R/W	R/W							
15:9	IFIFO_BYPASS: Bypass the FIFO of the selected bit slice and pass data directly to the interconnect logic. (1 = bypass, 0 = use FIFO). <i>No longer supported.</i>															
8	MON_RESET: Reset the monitor DLL (active-High).															
7	BS_RESET_TRI: Reset the 3-state bit slice (active-High). If TX_OUTPUT_PHASE_90 is used, TBYTE_IN must be 0x0 when BS_RESET is being performed.															
6:0	BS_RESET: Reset the selected bit slice(s) (active-High). If TX_OUTPUT_PHASE_90 is used, TBYTE_IN must be 0x0 when BS_RESET is being performed.															

Table 2-43: Register Bit Description (IODELAY\_INC\_BCAST\_CTRL)

IODELAY_INC_BCAST_CTRL													ADDR: 0x06			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default							0	0	0	0	0	0	0	0	0	0
Access							R/W	R/W	R/W	R/W						
15	Reserved															
9	BCAST_SEL: Broadcast input or output delay lines (1 = input delay, 0 = output delay).															
8	BCAST_INC: Broadcast INC or DEC (1 = INC, 0 = DEC).															
7	BCAST_EN: Broadcast enable of fine delay adjustment to delay line [0:6] (1 = enable, 0 = disable).															
6:0	BCAST_MASK_IDLY[0:6]: Disable broadcast of INC/DEC to selected delay line (1 = disable, 0 = enable). <b>Note:</b> BISC continuously increments and/or decrements the delay during self-calibration so be careful before writing to input delay lines.															

Table 2-44: Register Bit Description (PQTR)

PQTR													ADDR: 0x07			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default								0	0	0	0	0	0	0	1	1
Access								R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
15	INC: Increment. See <a href="#">Table 2-47</a> .															
14	DEC: Decrement. See <a href="#">Table 2-47</a> .															
13	CRSE: See <a href="#">Table 2-47</a> . Coarse delay increment or decrement of 8 taps. Might cause capture clock to glitch when used for PQTR adjustments. Issue a BS_RESET (ADDR=0x05) to fix alignment after any coarse jump.															
12:9	Reserved															
8:0	PQTR: P-side quarter delay of 0 to 511 taps.															

Table 2-45: Register Bit Description (NQTR)

NQTR													ADDR: 0x08			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default	0	0	0					0	0	0	0	0	0	0	1	1
Access	W	W	W					R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
15	INC: Increment. See <a href="#">Table 2-47</a> .															
14	DEC: Decrement. See <a href="#">Table 2-47</a> .															
13	CRSE: See <a href="#">Table 2-47</a> . Coarse delay increment or decrement of 8 taps. Might cause capture clock to glitch when used for PQTR adjustments. Issue a BS_RESET (ADDR = 0x05) to fix alignment after any coarse jump.															
12:9	Reserved															
8:0	NQTR: N-side quarter delay of 0 to 511 taps.															

Table 2-46: Register Bit Description (MON)

MON													ADDR: 0x09			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default	0	0	0				0	0	0	0	0	0	0	0	0	0
Access	W	W	W				R/W	R/W	R/W	R/W						
15	INC: Increment. See <a href="#">Table 2-47</a> .															
14	DEC: Decrement. See <a href="#">Table 2-47</a> .															
13	CRSE: See <a href="#">Table 2-47</a> .															
12:10	Reserved															
9:0	MON: Monitor delay of 0 to 1023 taps.															

Table 2-47: RIU Delay Adjustments (PQTR, NQTR, MON)

INC	DEC	CRSE	RIU Action
0	0	X	Load RIU_WR_DATA[8:0] into delay.
1	1	X	Load RIU_WR_DATA[8:0] into delay.
0	1	0	Decrement delay by 1 tap.
0	1	1	Decrement delay by 8 taps.
1	0	0	Increment delay by 1 tap.
1	0	1	Increment delay by 8 taps.

Table 2-48: Register Bit Description (ODELAYxx)

ODELAYxx													ADDR: 0x0A–0x11 <sup>(1)</sup>			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default	0	0						0	0	0	0	0	0	0	0	0
Access	W	W						R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
15	INC: Increment delay. See <a href="#">Table 2-50</a> .															
14	DEC: Decrement delay. See <a href="#">Table 2-50</a> .															
13:9	Reserved															
8:0	Output delay line: Tap value between 0 and 511. Fine delay adjustment of write data bits. Can be used for per-bit deskew or DDR write leveling.															

**Notes:**

1. ADDR: 0x0A is the output delay in the TX\_BITSLICE\_TRI of the nibble. ADDR: 0x0B to 0x11 are the output delays in the TX\_BITSLICES of the nibble.

Table 2-49: Register Bit Description (IDELAYxx)

IDELAYxx													ADDR: 0x12–0x18			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
Access	W	W		R/W	R/W	R/W	R/W									
15	INC: Increment delay. See <a href="#">Table 2-50</a> .															
14	DEC: Decrement delay. See <a href="#">Table 2-50</a> .															
13	Reserved															
12:9	RX_DCC: Input delay line duty cycle correction.															
8:0	Input delay line: Tap value between 0 and 511. Fine delay adjustment of read data bits. Can be used for per-bit deskew and placement of each data bit relative to the sample clock.															

Table 2-50: RIU Delay Adjustments (IDELAY, ODELAY)

INC	DEC	RIU Action
0	0	Load RIU_WR_DATA[8:0] into delay.
0	1	Decrement delay by 1 tap.
1	0	Increment delay by 1 tap.
1	1	Load RIU_WR_DATA[8:0] into delay.

Table 2-51: Register Bit Description (PQTR\_ALIGN, NQTR\_ALIGN, MON\_ALIGN, IODELAY\_ALIGN)

*_ALIGN													ADDR: 0x19–0x22			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default										0	0	0	0	0	0	0
Access										R/W	R/W	R/W	R/W	R/W	R/W	R/W
17:7	Reserved															
6:0	_ALIGN: Stores the align value after calibration. During initial calibration (when SELF_CALIBRATE is set to ENABLE), BISC programs this register to match the data delay to the sample clock/strobe delay.															

Table 2-52: Register Bit Description (PQTR\_RATIO, NQTR\_RATIO, IODELAY\_RATIO)

*_RATIO													ADDR: 0x23–0x2B			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
15:0	_RATIO: Stores the ratio value after calibration. Used by BISC when EN_VTC of BISTSLICE_CONTROL is asserted High. BISC calculates the baseline for Voltage and Temperature compensation.															

Table 2-53: Register Bit Description (WL\_DLY\_RNK)

WL_DLY_RNK													ADDR: 0x2C–0x2F			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default			0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access			R/W	R/W	R/W	R/W										
15:14	Reserved															
13	WL_TRAIN: Set to 1 to put bit slices into write leveling mode. 3-states the data bits while allowing the strobe/clock bits to drive the output buffers. This bit is only present at address 0x2C and does not exist for 0x2D, 0x2E, and 0x2F registers.															
12:9	WL_DLY_CRSE: Coarse delay adjustment of 1/2 PLL_CLK period on write data/strobe/clock versus clock.															
8:0	WL_DLY_FINE: Fine delay adjustment on write data/strobe/clock versus clock.															

Table 2-54: Register Bit Description (RL\_DLY\_RNK)<sup>(2)</sup>

RL_DLY_RNK													ADDR: 0x30–0x33			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default				0	0	0	0	0	0	0	0	0	0	0	0	0
Access				R/W	R/W	R/W	R/W									
15:13	Reserved															
12:9	RL_DLY_CRSE: Coarse delay adjustment of $1/2$ PLL_CLK period on read strobe/clock gate.															
8:0	RL_DLY_FINE: Fine delay adjustment on read strobe/clock gate.															

**Notes:**

- Reserved for memory interface generator (MIG). For memory designs using the RL\_DLY\_RNK, the PLL clock source (CLKIN for the PLL connected to BITSlice\_CONTROL) and RIU\_CLK (BITSlice\_CONTROL) must be sourced from the same MMCM with the same phase shift to ensure asynchronous transfers are not corrupted.

Table 2-55: Register Bit Description (RD\_IDLE\_COUNT)

RD_IDLE_COUNT													ADDR: 0x34			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default											0	0	0	0	0	0
Access											R/W	R/W	R/W	R/W	R/W	R/W
15:6	Reserved															
5:0	Number of clocks (frequency of PLL_CLK/DATA_WIDTH) after PHY_RDEN deassertion and before turning off ODT termination in IOB.															

Table 2-56: Register Bit Description (RL\_DLY\_RATIO)

RL_DLY_RATIO													ADDR: 0x35			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
15:0	RL_DLY_RATIO: This stores the RATIO value after calibration. Used by BISC for strobe/clock gating and VT tracking.															

Table 2-57: Register Bit Description (RL\_DLY\_QTR)

RL_DLY_QTR													ADDR: 0x36			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default								0	0	0	0	0	0	0	0	0
Access								R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
15:9	Reserved															
8:0	RL_DLY_QTR: Set from 0 to 511 taps of fine delay. Determines the 90-degree delay on DQS/clock. Used for DQS gating and VT tracking.															

Table 2-58: Register Bit Description (DBG\_WR\_STATUS)

DBG_WR_STATUS													ADDR: 0x37			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
15:0	DBG_WR_STATUS: Debug status for write.															

Table 2-59: Register Bit Description (DBG\_RW\_INDEX)

DBG_RW_INDEX													ADDR: 0x38			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
15:8	DBG_RW_INDEX: Multiplexer selection address for debug status write. 0x00: No write. 0x01–0x7F: Write status to BISC. 0x80–0xFF: Write status to other modules.															
7:0	DBG_RD_INDEX: Multiplexer selection address for debug status read. 0x00–0x7F: Read status from BISC. 0x80–0xFF: Read status from other modules.															

Table 2-60: Register Bit Description (DBG\_RD\_STATUS)

DBG_RD_STATUS													ADDR: 0x39			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
15:0	DBG_RD_STATUS: Debug status for read.															

Table 2-61: Register Bit Description (DFD\_CTRL)

DFD_CTRL													ADDR: 0x3A			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Default																0
Access																R/W
15:1	Reserved															
0	DBG_CT_START_EN: DFD debug counter start.															

**Notes:**

- There is no address decoding and no registers for addresses ranging from 0x3B to 0x3F.

Table 2-62: RIU Registers and Their Corresponding Attributes

Attribute	Register	Default	Description	Position	Bits
EN_OTHER_PCLK	NIBBLE_CTRL0	FALSE	Set to 1 to pass a source synchronous clock from the other nibble's strobe/clock gating circuit through the PQTR slave delay. Set to 0 to pass a clock from the present nibble's strobe/clock gating circuit through the PQTR slave delay.	0	EN_PDQS
EN_OTHER_NCLK	NIBBLE_CTRL0	FALSE	Set to 1 to pass a source synchronous clock from the other nibble's strobe/clock gating circuit through the NQTR slave delay. Set to 0 to pass a clock from the present nibble's strobe/clock gating circuit through the NQTR slave delay.	1	EN_NDQS
INV_RXCLK	NIBBLE_CTRL0	FALSE	Invert clock path from IOB to RX_BITSLICE.	2	INVERT_RX_CLK
SERIAL_MODE	NIBBLE_CTRL0	FALSE	Set to 1 to put bit slice read paths into SERIAL_MODE. This mode is used for sampling a serial data stream such as SGMII. If serial mode is selected by setting SERIAL_MODE=1 (NIBBLE_CTRL0, Bit 3), the SERIAL_MODE setting must be reassigned after DLY_RDY = 1. The SERIAL_MODE setting reverts to the programmed settings when DLY_RDY is asserted.	3	SERIAL_MODE
TX_GATING	NIBBLE_CTRL0	FALSE	Disable clock gating in write clock path.	4	TX_GATE
RX_GATING	NIBBLE_CTRL0	FALSE	Disable clock gating in read clock path.	5	RX_GATE
RXGATE_EXTEND	NIBBLE_CTRL0	FALSE	BQS bias enable.	6	RXGATE_EXTEND
RX_CLK_PHASE_P	NIBBLE_CTRL1	SHIFT_0	Apply a phase shift of 90 degrees to the received clock.	0	RX_CLK_PHASE_P
RX_CLK_PHASE_N	NIBBLE_CTRL1	SHIFT_0	Apply a phase shift of 90 degrees to the received clock.	1	RX_CLK_PHASE_N
TX_OUTPUT_PHASE_90	NIBBLE_CTRL1	FALSE	Per transmitter delays output phase by 90 degrees when set to TRUE.	2:11	TX_DATA_PHASE

Table 2-62: RIU Registers and Their Corresponding Attributes (Cont'd)

Attribute	Register	Default	Description	Position	Bits
SELF_CALIBRATE	CALIB_CTRL	ENABLE	Turn self-calibration (BISC) on or off.	0	CALIBRATE
IDLY_VT_TRACK	CALIB_CTRL	TRUE	Enable VT tracking for input delay lines.	1	DIS_VTTRACK_IBIT
ODLY_VT_TRACK	CALIB_CTRL	TRUE	Enable VT racking for output delay lines.	2	DIS_VTTRACK_OBIT

# High Density I/O Resources

## Introduction to High Density I/O Banks

High-density (HD) I/O banks are SelectIO resources designed to support a wide range of I/O standards with voltages ranging from 1.2V to 3.3V. HD I/Os are optimized for single-ended, voltage-referenced, and pseudo-differential I/O standards operating at data rates of up to 250 Mb/s. Limited support for true differential inputs (with external termination) is also available to support LVDS and LVPECL clock inputs. HD I/Os also contain interface logic including registers and static delay lines to support asynchronous, system synchronous, and clock-based source synchronous interfaces. [Table 3-1](#) highlights the features supported in HD I/O banks.



**IMPORTANT:** This chapter only describes HD I/O banks in Zynq™ UltraScale+™ devices, Kintex™ UltraScale+ devices, Artix™ UltraScale+ devices, and some Virtex™ UltraScale+ devices. For HD I/O banks in Spartan UltraScale+ devices, refer to Spartan UltraScale+ FPGAs SelectIO Resources User Guide (UG861) [Ref 4].

Table 3-1: Supported Features in the HD I/O Banks

Features	HD I/O Bank Support
3.3V I/O standards	LVTTTL and LVCMOS
2.5V I/O standards	LVCMOS and LVDS/SUB_LVDS <sup>(1)</sup>
1.8V I/O standards	LVCMOS, SSTL <sup>(2)(3)</sup> , and HSTL <sup>(2)(3)</sup>
1.5V I/O standards	LVCMOS, SSTL <sup>(2)(3)</sup> , and HSTL <sup>(2)(3)</sup>
1.35V I/O standards	SSTL <sup>(2)(3)</sup>
1.2V I/O standards	LVCMOS, SSTL <sup>(2)(3)</sup> , and HSTL <sup>(2)(3)</sup>
LVDS and LVPECL	Inputs supported (with external termination). <sup>(1)</sup>
V <sub>REF</sub>	Internal V <sub>REF</sub> supported in HD I/O banks (no external V <sub>REF</sub> ).
Maximum data rate	250 Mb/s DDR
Output drive strength control	Supported
Output slew rate control	Supported
Pull-up, pull-down, and keeper	Supported

**Table 3-1: Supported Features in the HD I/O Banks (Cont'd)**

Features	HD I/O Bank Support
ILOGIC for SDR and DDR interfaces	Supported
OLOGIC for SDR and DDR interfaces	Supported
ZHOLD (static delay for zero hold)	Supported
Internal differential termination (DIFF_TERM)	Not supported
Digitally controlled impedance (DCI) and DCI cascading	Not supported
ISERDES, OSERDES	Not supported
Programmable delay (IDELAY, ODELAY)	Not supported
DQS_BIAS	Not supported

**Notes:**

1. Does not support differential termination or LVDS/SUB\_LVDS outputs. These features are supported in high-performance I/O (HP I/O) banks available in the same device.
2. Optional 50Ω on-die input termination is supported for SSTL and HSTL inputs.
3. SSTL, HSTL, and HSUL support enables legacy and chip-to-chip interfaces. There is no support for interfaces to DRAM memory devices (DDR3, DDR4, LPDDR2, or LPDDR3).

## HD I/O Bank Resources

Every HD I/O bank contains 24 I/O pins. Within each bank are four clock-capable pin pairs which provide access to clocking resources or can be used as user I/O when not being used as input clocks. When defined as single-ended standards, HD I/O pins support input, output, and bidirectional operating modes. Paired I/O pins can be used to support differential standard functionality. For pseudo-differential standards, like DIFF\_SSTL15, input, output, and bidirectional support is available. True differential standards, like LVDS\_25, can only function as an input buffer.

## HD I/O Bank Features

HD I/O banks support the following features.

### **Output Slew Rate Control**

SLEW settings of FAST and SLOW (default) are supported. The FAST slew rate setting enables higher performance interfaces, while the SLOW slew rate often provides an improved signal quality for many applications.

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

```
set_property SLEW FAST|SLOW [get_ports port_name]
```

## Output Drive Strength

For LVCMOS and LVTTTL output buffers, the DRIVE attribute can be used to specify the load that the driver can safely drive to valid logic levels (in mA). The allowed values for the DRIVE attribute are shown in [Table 3-2](#).

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

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

## Uncalibrated Input Termination

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

```
set_property ODT RTT_48 [get_ports port_name]
```

to create a Thevenin equivalent to the circuit shown in [Figure 3-1](#)

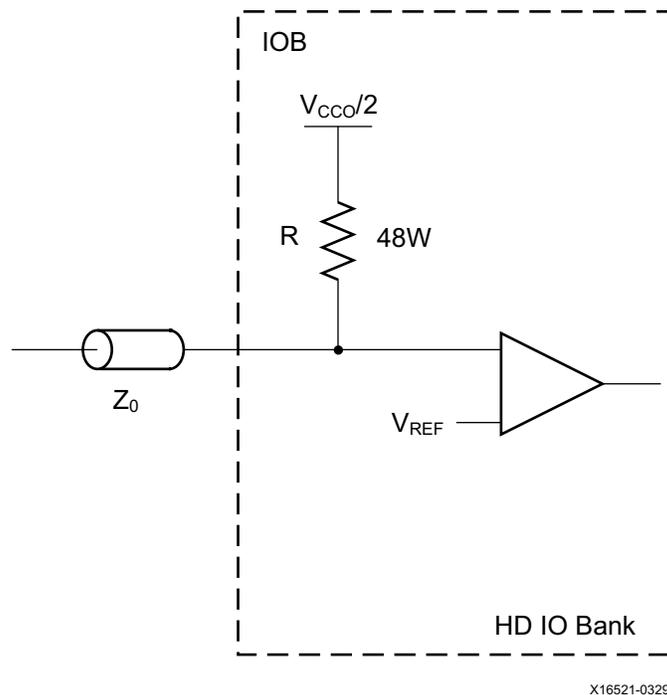


Figure 3-1: Input Termination to  $V_{CCO}$  using Single Termination

Uncalibrated split ODT is supported on all voltage-referenced I/O standards. [Table 3-2](#) defines the specific standards that are supported to use with this feature. The user-programmable input termination setting is implemented with a  $96\Omega$  pull-up and  $96\Omega$  pull-down to provide the equivalent  $48\Omega$  to  $V_{CCO}/2$  termination.

### **Internal $V_{REF}$**

The  $V_{REF}$  supply required by the SSTL, HSTL, HSUL, and POD standards is generated inside the device, removing the need to provide a  $V_{REF}$  supply rail on the PCB or allocate a package pin for  $V_{REF}$ . The internally generated  $V_{REF}$  is sourced from the  $V_{CCO}$  supply rail.

The constraint INTERNAL\_VREF is assigned on a bank-wide basis. The INTERNAL\_VREF attribute uses the following syntax in the XDC file:

```
set_property INTERNAL_VREF voltage [get_iobanks bank_number]
```

Example 1: INTERNAL\_VREF for Bank 84 using HSTL\_II (1.5V), which requires a 0.75V reference voltage, uses the following constraint:

```
set_property INTERNAL_VREF 0.75 [get_iobanks 84]
```

Example 2: INTERNAL\_VREF for Bank 65 using HSTL\_II\_18 (1.8V), which requires a 0.9V reference voltage, uses the following constraint:

```
set_property INTERNAL_VREF 0.90 [get_iobanks 65]
```

The rules for using INTERNAL\_VREF in an HD bank are:

- One value of  $V_{REF}$  can be set for the bank.
- INTERNAL\_VREF can only be set to the nominal reference voltage value of a given I/O standard.
- Valid settings of INTERNAL\_VREF are listed, but must be selected to half of the  $V_{CCO}$  level. Valid levels in HD I/O banks are 0.60V, 0.675V, 0.75V, and 0.9V.

### **Pull-up, Pull-down, and Keeper**

I/O buffers (input, output, and bidirectional) support a user programmable option that enables a weak pull-up resistor, a weak pull-down resistor, or a weak *keeper* circuit.

PULLTYPE attribute has the following possible values:

- NONE
- PULLUP
- PULLDOWN
- KEEPER

These attributes use the following syntaxes in the XDC file:

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

### **IBUFDISABLE**

The input buffer can be dynamically disabled using a control signal from the interconnect logic to drive the IBUFDISABLE port of the buffer primitive. Disabling the input buffer can help reduce power consumption when the input bus is idle.

The IBUF\_IBUFDISABLE, IBUFDS\_IBUFDISABLE, and IBUFDS\_DIFF\_OUT\_IBUFDISABLE primitives can all be used in the HD I/O. See [SelectIO Interface Primitives](#) for more details on these primitives.

### **INTERMDISABLE**

When ODT is used, the INTERMDISABLE port can be used to disable the ODT termination when a reduction of power is desired. The IBUF\_INTERMDISABLE, IOBUF\_INTERMDISABLE, IBUFDS\_DIFF\_OUT\_INTERMDISABLE, IBUFDS\_INTERMDISABLE, and IOBUFDS\_DIFF\_OUT\_INTERMDISABLE primitives can all be used in the HD I/O. See [SelectIO Interface Primitives](#) for more details on these primitives.

## **HD I/O Supported Standards**

The SelectIO pins can be configured to various I/O standards.

- Single-ended I/O standards (LVTTTL and LVCMOS)
- Voltage-referenced I/O standards (SSTL, HSTL, and HSUL)
- Pseudo-differential I/O standards (differential SSTL and differential HSTL)
- True-differential inputs (LVDS, LVPECL)

[Table 3-2](#) lists the HD I/O supported standards with their respective features.

**Table 3-2: HD I/O Supported Standards**

I/O Standard	Driver Features		Receiver Features
	DRIVE	SLEW	ODT
LVC MOS12 <sup>(1)(2)</sup>	4, 8, 12	SLOW, FAST	–
LVC MOS15 <sup>(1)(2)</sup>	4, 8, 12, 16	SLOW, FAST	–
LVC MOS18 <sup>(1)(2)</sup>	4, 8, 12, 16	SLOW, FAST	–
LVC MOS25 <sup>(1)(2)</sup>	4, 8, 12, 16	SLOW, FAST	–
LVC MOS33 <sup>(1)(2)</sup>	4, 8, 12, 16	SLOW, FAST	–
LVTTTL <sup>(1)(2)</sup>	4, 8, 12, 16	SLOW, FAST	–
HSUL_12 <sup>(1)</sup>	–	SLOW, FAST	–
HSTL_I <sup>(1)</sup>	–	SLOW, FAST	RTT_48

Table 3-2: HD I/O Supported Standards (Cont'd)

I/O Standard	Driver Features		Receiver Features
	DRIVE	SLEW	ODT
HSTL_I_18 <sup>(1)</sup>	–	SLOW, FAST	RTT_48
SSTL18_I <sup>(1)</sup>	–	SLOW, FAST	RTT_48
SSTL18_II <sup>(1)</sup>	–	SLOW, FAST	RTT_48
SSTL15 <sup>(1)</sup>	–	SLOW, FAST	RTT_48
SSTL15_II <sup>(1)</sup>	–	SLOW, FAST	RTT_48
SSTL135 <sup>(1)</sup>	–	SLOW, FAST	RTT_48
SSTL135_II <sup>(1)</sup>	–	SLOW, FAST	RTT_48
SSTL12 <sup>(1)</sup>	–	SLOW, FAST	RTT_48
DIFF_HSTL_I <sup>(1)</sup>	–	SLOW, FAST	RTT_48
DIFF_HSTL_I_18 <sup>(1)</sup>	–	SLOW, FAST	RTT_48
DIFF_SSTL18_I <sup>(1)</sup>	–	SLOW, FAST	RTT_48
DIFF_SSTL18_II <sup>(1)</sup>	–	SLOW, FAST	RTT_48
DIFF_SSTL15 <sup>(1)</sup>	–	SLOW, FAST	RTT_48
DIFF_SSTL15_II <sup>(1)</sup>	–	SLOW, FAST	RTT_48
DIFF_SSTL135 <sup>(1)</sup>	–	SLOW, FAST	RTT_48
DIFF_SSTL135_II <sup>(1)</sup>	–	SLOW, FAST	RTT_48
DIFF_SSTL12 <sup>(1)</sup>	–	SLOW, FAST	RTT_48
DIFF_HSUL_12	–	SLOW, FAST	–
LVPECL (receiver only) <sup>(1)</sup>	N/A	N/A	–
LVDS_25 (receiver only) <sup>(1)(3)</sup>	N/A	N/A	–
SUB_LVDS (receiver only) <sup>(1)</sup>	N/A	N/A	–
SLVS_400_25 (receiver only) <sup>(1)</sup>	N/A	N/A	–

**Notes:**

1. Maximum frequency of operation is 250 Mb/s DDR.
2. For 4 mA drive strength, the maximum frequency of operation is limited to 125 Mb/s.
3. There is not a specific requirement on the  $V_{CCO}$  bank voltage on LVDS\_25 inputs provided the  $V_{CCO}$  level is high enough to ensure the pin voltage aligns to the  $V_{in}$  spec in the *Recommended Operating Conditions* table of the specific UltraScale+ device data sheet [Ref 2].

---

## HD I/O Interface Logic

HD I/O pins contain an I/O interface logic (IOI) block that enables various I/O interfaces. HD I/O IOI consists of an OLOGIC and ILOGIC block dedicated for each pin.

Supported interfaces include the following:

- Asynchronous (or combinatorial) input and output interfaces
- System synchronous interfaces with SDR registers in the IOI and/or interconnect logic. Supported flip-flop primitives include the following.
  - 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
- Source synchronous interfaces with DDR registers in the IOI and/or interconnect logic. Supported primitives include IDDRE1 and ODDRE1.

### ZHOLD

The ILOGIC block supports an optional static uncompensated zero hold (ZHOLD) delay line on inputs to compensate for clock insertion delay. The ZHOLD feature is optimized to compensate for the clock insertion delays when the clocking path is directly sourced from a BUFG/BUFGCE, which is sourced in the same bank or on an adjacent bank. ZHOLD is enabled by default unless the clock source is a MMCM/PLL or unless the IOBDELAY attribute is set in the XDC.



---

**IMPORTANT:** *ZHOLD might not be appropriate for all applications, so consult the timing report to verify the impact to a specific clocking scheme.*

---

### DDR Inputs (IDDRE1)

UltraScale devices have dedicated registers in the ILOGIC block to implement input DDR registers. This feature is used by instantiating the IDDRE1 primitive. The IDDRE1 primitive supports these modes of operation:

- [OPPOSITE\\_EDGE Mode, page 150](#)
- [SAME\\_EDGE Mode, page 151](#)
- [SAME\\_EDGE\\_PIPELINED Mode, page 152](#)

The SAME\_EDGE and SAME\_EDGE\_PIPELINED mode data is presented into the interconnect logic on the same clock edge. These modes are implemented using the DDR\_CLK\_EDGE attribute.

Figure 2-7 shows a block diagram of the IDDRE1 primitive. Table 2-1 lists the IDDRE1 ports and Table 2-2 lists the IDDRE1 attributes.

To ensure output registers are forced to use the IOB resources, use the following syntax in the XDC:

```
set_property IOB TRUE [get_ports portname]
```

## DDR Outputs (ODDRE1)

UltraScale devices have registers in the OLOGIC block to implement output DDR registers for both data and 3-state control. When data and 3-state paths are both used in HD I/O, both data and 3-state control are required to either both use (or both not use) the output DDR register. For example, you cannot have a design that uses an output DDR register on the datapath without an output DDR register on the 3-state control path.

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.

There is only one clock input to the ODDRE1 primitive. Falling-edge data is clocked by a locally inverted version of the input 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, saving CLB and clock resources, and increasing performance. This mode is also supported for 3-state control. The timing diagram of the output DDR is shown in Figure 2-8. Figure 2-9 shows a block diagram of the ODDRE1 primitive. The ODDRE1 block in HD bank differs from the XP bank in that SR pin deassertion occurs immediately with no register delay. Compared to simulation, ODDRE1 in HD banks comes out of reset 3 clock cycles early. Table 2-3 lists the ODDRE1 ports and Table 2-4 lists the ODDRE1 attributes.

To ensure output registers are forced to use the IOB resources, use the following syntax in the XDC. For data:

```
set_property IOB TRUE [get_cell <cell_name>]
```

For tristate:

```
set_property IOB_TRI_REG value [get_cells <cell_name>]
```

# Termination Options for Simultaneous Switching Noise Analysis

## Termination Options

The Vivado™ Design Suite can perform simultaneous switching noise (SSN) analysis for each design, taking into account the actual I/O standards and options assigned to the I/O pins in the target device and package.

For each output pin, there is the option to specify whether or not termination is present on the board. The off-chip termination field automatically populates with the default terminations for each I/O standard, if one exists.

[Table A-1](#) lists all of the default terminations for each of the I/O standards supported by UltraScale™ devices when using the SSN predictor tool within the Vivado Design Suite. For each I/O pin in the design, you can specify whether to use these terminations, or to have no termination.

**Table A-1: Default Terminations for SSN Noise Analysis by I/O Standard**

I/O Standard	Drive	Termination Option
BLVDS_25	–	Near Series 165Ω, Near Differential 140Ω, and Far Differential 100Ω
DIFF_HSTL_I	–	Far $V_{TT}$ 40Ω
DIFF_HSTL_I_12	–	Far $V_{TT}$ 40Ω
DIFF_HSTL_I_DCI_12	–	Far $V_{TT}$ 40Ω
DIFF_HSTL_I_18	–	Far $V_{TT}$ 50Ω
DIFF_HSTL_I_DCI	–	Far $V_{TT}$ 40Ω
DIFF_HSTL_I_DCI_18	–	Far $V_{TT}$ 50Ω
DIFF_HSTL_II	–	Near $V_{TT}$ 50Ω & Far $V_{TT}$ 50Ω
DIFF_HSTL_II_18	–	Near $V_{TT}$ 50Ω & Far $V_{TT}$ 50Ω
DIFF_HSUL_12	–	None
DIFF_HSUL_12_DCI	–	None
DIFF_POD10	–	Far $V_{CCO}$ 40Ω
DIFF_POD10_DCI	–	Far $V_{CCO}$ 40Ω

Table A-1: Default Terminations for SSN Noise Analysis by I/O Standard (Cont'd)

I/O Standard	Drive	Termination Option
DIFF_POD12	–	Far $V_{CCO}$ 40 $\Omega$
DIFF_POD12_DCI	–	Far $V_{CCO}$ 40 $\Omega$
DIFF_SSTL12	–	Far $V_{TT}$ 40 $\Omega$
DIFF_SSTL12_DCI	–	Far $V_{TT}$ 40 $\Omega$
DIFF_SSTL135	–	Far $V_{TT}$ 40 $\Omega$
DIFF_SSTL135_DCI	–	Far $V_{TT}$ 40 $\Omega$
DIFF_SSTL135_R	–	Far $V_{TT}$ 40 $\Omega$
DIFF_SSTL15	–	Far $V_{TT}$ 40 $\Omega$
DIFF_SSTL15_DCI	–	Far $V_{TT}$ 40 $\Omega$
DIFF_SSTL15_R	–	Far $V_{TT}$ 50 $\Omega$
DIFF_SSTL18_I	–	Far $V_{TT}$ 50 $\Omega$
DIFF_SSTL18_I_DCI	–	Far $V_{TT}$ 50 $\Omega$
DIFF_SSTL18_II	–	Near $V_{TT}$ 50 $\Omega$ & Far $V_{TT}$ 50 $\Omega$
HSLVDCI_15	–	None
HSLVDCI_18	–	None
HSTL_I	–	Far $V_{TT}$ 40 $\Omega$
HSTL_I_12	–	Far $V_{TT}$ 40 $\Omega$
HSTL_I_DCI_12	–	Far $V_{TT}$ 40 $\Omega$
HSTL_I_18	–	Far $V_{TT}$ 50 $\Omega$
HSTL_I_DCI	–	Far $V_{TT}$ 40 $\Omega$
HSTL_I_DCI_18	–	Far $V_{TT}$ 50 $\Omega$
HSTL_II	–	Near $V_{TT}$ 50 $\Omega$ & Far $V_{TT}$ 50 $\Omega$
HSTL_II_18	–	Near $V_{TT}$ 50 $\Omega$ & Far $V_{TT}$ 50 $\Omega$
HSUL_12	–	None
HSUL_12_DCI	–	None
LVC MOS12	2	None
LVC MOS12	4	None
LVC MOS12	6	None
LVC MOS12	8	None
LVC MOS12	12	Far $V_{TT}$ 50 $\Omega$
LVC MOS15	2	None
LVC MOS15	4	None
LVC MOS15	6	None
LVC MOS15	8	None
LVC MOS15	12	Far $V_{TT}$ 50 $\Omega$

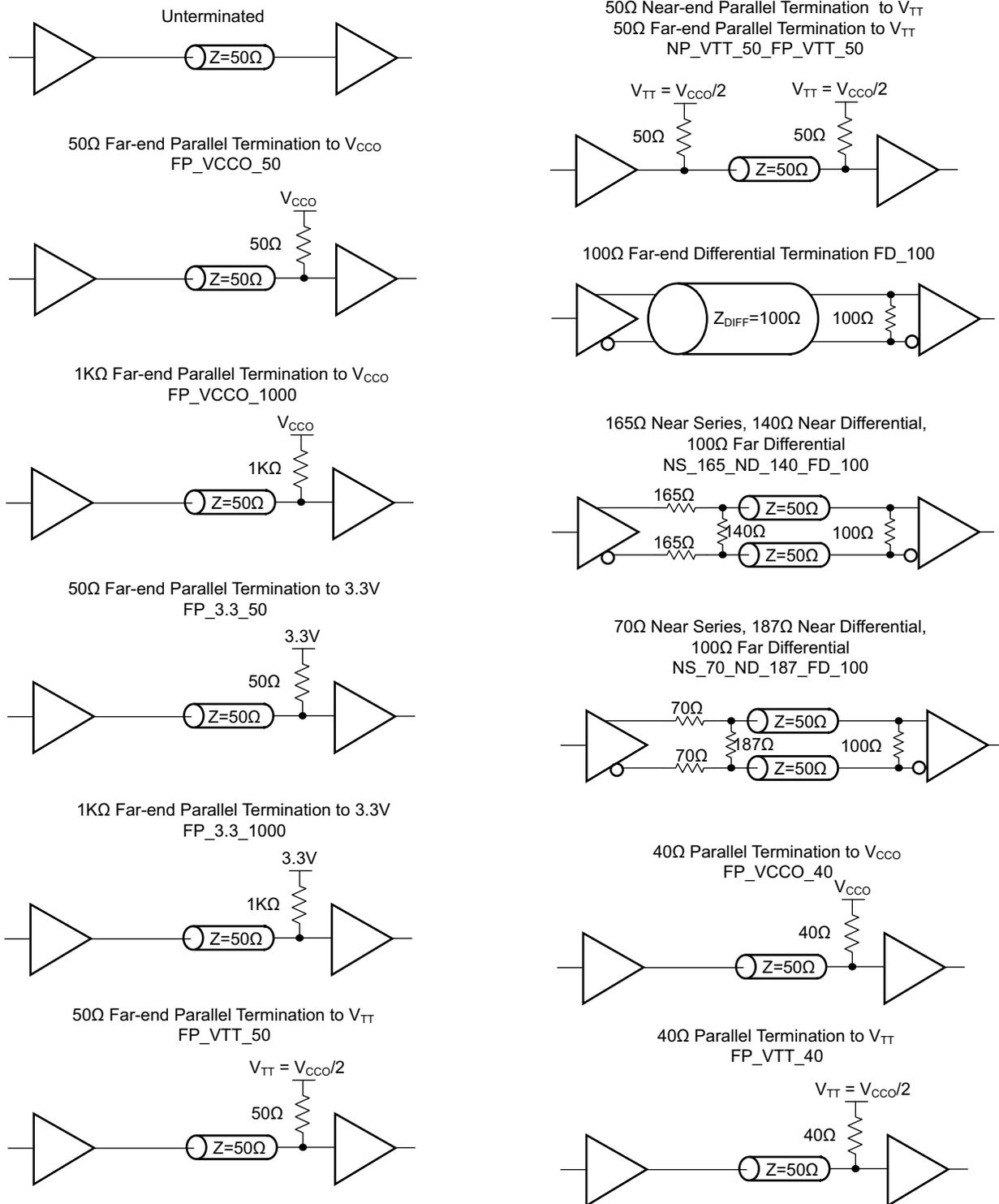
Table A-1: Default Terminations for SSN Noise Analysis by I/O Standard (Cont'd)

I/O Standard	Drive	Termination Option
LVC MOS15	16	Far $V_{TT}$ 50 $\Omega$
LVC MOS18	2	None
LVC MOS18	4	None
LVC MOS18	6	None
LVC MOS18	8	None
LVC MOS18	12	Far $V_{TT}$ 50 $\Omega$
LVC MOS18	16	Far $V_{TT}$ 50 $\Omega$
LVC MOS25	4	None
LVC MOS25	8	None
LVC MOS25	12	Far $V_{TT}$ 50 $\Omega$
LVC MOS25	16	Far $V_{TT}$ 50 $\Omega$
LVC MOS33	4	None
LVC MOS33	8	None
LVC MOS33	12	Far $V_{TT}$ 50 $\Omega$
LVC MOS33	16	Far $V_{TT}$ 50 $\Omega$
LVDCI_15	–	None
LVDCI_18	–	None
LVDS	–	Far Differential 100 $\Omega$
LVDS_25	–	Far Differential 100 $\Omega$
LVDS_25_PE	–	Far Differential 100 $\Omega$
LVDS_PE	–	Far Differential 100 $\Omega$
LVTTL	4	None
LVTTL	8	None
LVTTL	12	Far $V_{TT}$ 50 $\Omega$
LVTTL	16	Far $V_{TT}$ 50 $\Omega$
MINI_LVDS_25	–	Far Differential 100 $\Omega$
POD10	–	Far $V_{CCO}$ 40 $\Omega$
POD10_DCI	–	Far $V_{CCO}$ 40 $\Omega$
POD12	–	Far $V_{CCO}$ 40 $\Omega$
POD12_DCI	–	Far $V_{CCO}$ 40 $\Omega$
PPDS_25	–	Far Differential 100 $\Omega$
RSDS_25	–	Far Differential 100 $\Omega$
SSTL12	–	Far $V_{TT}$ 40 $\Omega$
SSTL12_DCI	–	Far $V_{TT}$ 40 $\Omega$
SSTL135	–	Far $V_{TT}$ 40 $\Omega$

Table A-1: Default Terminations for SSN Noise Analysis by I/O Standard (Cont'd)

I/O Standard	Drive	Termination Option
SSTL135_DCI	–	Far $V_{TT}$ 40Ω
SSTL135_R	–	Far $V_{TT}$ 40Ω
SSTL15	–	Far $V_{TT}$ 40Ω
SSTL15_DCI	–	Far $V_{TT}$ 40Ω
SSTL15_R	–	Far $V_{TT}$ 50Ω
SSTL18_I	–	Far $V_{TT}$ 50Ω
SSTL18_I_DCI	–	Far $V_{TT}$ 50Ω
SSTL18_II	–	Near $V_{TT}$ 50Ω & Far $V_{TT}$ 50Ω
TMDS_33	–	Far 3.3V 50Ω
SUB_LVDS	–	Far Differential 100Ω

Figure A-1 illustrates each of these terminations.



X17678-081616

Figure A-1: Default Terminations

# Additional Resources and Legal Notices

---

## Finding Additional Documentation

### Technical Information Portal

The AMD Technical Information Portal is an online tool that provides robust search and navigation for documentation using your web browser. To access the Technical Information Portal, go to <https://docs.amd.com>.

### Documentation Navigator

Documentation Navigator (DocNav) is an installed tool that provides access to AMD Adaptive Computing documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the AMD Vivado™ IDE, select **Help > Documentation and Tutorials**.
- On Windows, click the **Start** button and select **Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter `docnav`.

**Note:** For more information on DocNav, refer to the Documentation Navigator User Guide ([UG968](#)).

### Design Hubs

AMD 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.
- Go to the [Design Hubs](#) web page.

---

## Support Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see [Support](#).

---

## References

These documents and links provide supplemental material useful with this guide:

1. UltraScale and UltraScale+ product overviews:

*UltraScale Architecture and Product Overview* ([DS890](#))

*Zynq UltraScale+ MPSoC Overview* ([DS891](#))

*Zynq UltraScale+ RFSoc Data Sheet: Overview* ([DS889](#))

2. UltraScale and UltraScale+ device data sheets:

*Kintex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* ([DS892](#))

*Virtex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* ([DS893](#))

*Kintex UltraScale+ FPGAs Data Sheet: DC and AC Switching Characteristics* ([DS922](#))

*Virtex UltraScale+ FPGAs Data Sheet: DC and AC Switching Characteristics* ([DS923](#))

*Zynq UltraScale+ MPSoC Data Sheet: DC and AC Switching Characteristics* ([DS925](#))

*Zynq UltraScale+ RFSoc Data Sheet: DC and AC Switching Characteristics* ([DS926](#))

*Artix UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* ([DS931](#))

3. UltraScale and UltraScale+ device packaging and pinout guides:

*UltraScale and UltraScale+ FPGAs Packaging and Pinouts Product Specification* ([UG575](#))

*Zynq UltraScale+ Device Packaging and Pinouts Product Specification User Guide* ([UG1075](#))

4. *Spartan UltraScale+ FPGAs SelectIO Resources User Guide* ([UG861](#))

5. *UltraScale Architecture Configuration User Guide* ([UG570](#))

6. *UltraScale Architecture Libraries Guide* ([UG974](#))

7. *Vivado Design Suite Properties Reference Guide* ([UG912](#))

8. Electronic Industry Alliance JEDEC web site at [www.jedec.org](http://www.jedec.org)
9. *Vivado Design Suite User Guide: System-Level Design Entry* (UG895)
10. *UltraScale Architecture Clocking Resource User Guide* (UG572)
11. *Bitslip in Logic Application Note* (XAPP1208)
12. *UltraFast Design Methodology Guide for the Vivado Design Suite* (UG949)
13. *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085)
14. *UltraScale Architecture-Based FPGAs Memory IP LogiCORE IP Product Guide* (PG150)

## Revision History

The following table shows the revision history for this document.

Date	Version	Revision
01/14/2025	1.16	<p><b>Chapter 1:</b> Added Spartan UltraScale+ devices to <a href="#">Introduction to the UltraScale Architecture</a>. Added note about Spartan UltraScale+ devices to <a href="#">I/O Tile Overview</a>.</p> <p><b>Chapter 3:</b> Updated important note at the beginning of the chapter.</p>
08/31/2023	1.15	<p>Added Artix UltraScale+ devices throughout.</p> <p><b>Chapter 1:</b> Updated IBUFDS_DIFF_OUT bullet in <a href="#">SelectIO Interface Primitives</a>.</p> <p><b>Chapter 2:</b> In <a href="#">VAR_LOAD Mode</a>, updated bullet about Align_Delay. In <a href="#">Release Reset</a>, updated <a href="#">step d</a>.</p> <p><b>Chapter 3:</b> Added sentence about clock-capable pin pairs to <a href="#">HD I/O Bank Resources</a>.</p>
09/01/2022	1.14	<p><b>Chapter 1:</b> Updated note about DQS_BIAS in <a href="#">Table 1-55</a>.</p> <p><b>Chapter 2:</b> Updated description of ODDR_MODE in <a href="#">Table 2-10</a>. Updated descriptions of CLK, LOAD, and RST in <a href="#">Table 2-11</a> and <a href="#">Table 2-15</a>. Updated possible values for REFCLK_FREQUENCY in <a href="#">Table 2-12</a>. Added <a href="#">Figure 2-20</a>. Added <a href="#">Equation 2-4</a>. Updated values for REFCLK_FREQUENCY and UPDATE_MODE in <a href="#">Table 2-16</a>. Updated <a href="#">step 6a</a> and <a href="#">step 4a</a> in <a href="#">VAR_LOAD Mode</a>. Added tip about IDELAYCTRLs and BITSlice_CONTROLS to <a href="#">IDELAYCTRL</a>. Added note to description of RST in <a href="#">Table 2-17</a> and <a href="#">Table 2-30</a>. Updated <a href="#">step c</a> and added <a href="#">step d</a> to <a href="#">Release Reset</a>. Updated third paragraph with description of designs with multiple nibbles being used within a bank in <a href="#">Mixing Native and Non-Native Mode I/O in a Nibble</a>. Changed all instances of Align_Value to Align_Delay. Added <a href="#">Equation 2-11</a> and <a href="#">Equation 2-16</a>. Updated values of RX_REFCLK_FREQUENCY and TX_REFCLK_FREQUENCY in <a href="#">Table 2-23</a>. Updated values of REFCLK_FREQUENCY in <a href="#">Table 2-25</a>, <a href="#">Table 2-27</a>, and <a href="#">Table 2-29</a>.</p> <p><b>Chapter 3:</b> In <a href="#">Table 3-1</a>, added /SUB_LVDS to 2.5V I/O standards and updated maximum data rate to 250 Mb/s DDR. Updated first paragraph in <a href="#">HD I/O Interface Logic</a>.</p>

Date	Version	Revision
10/22/2021	1.13	<p><b>Chapter 1:</b> Added description of output-only IOBs to <a href="#">V<sub>REF</sub></a>. Updated <a href="#">step 2 in DCI I/O Standard Support</a>. Updated JEDEC specifications in <a href="#">LVCMOS</a>. Added note about changing drive setting to <a href="#">Table 1-22</a>. Updated first paragraph in <a href="#">HSTL_I_DCI</a>, <a href="#">HSTL_I_DCI_12</a>, and <a href="#">HSTL_I_DCI_18</a>. Updated <a href="#">Figure 1-54</a> and <a href="#">Figure 1-58</a>. Added description of alternate bias and termination to <a href="#">Receiver Termination</a>.</p> <p><b>Chapter 2:</b> Added sentence about SDR input and output registers to <a href="#">Simple Registered Inputs and Outputs</a>. Updated description of SR in <a href="#">Table 2-3</a>. Updated description of RST in <a href="#">Table 2-6</a> and <a href="#">Table 2-9</a>. Updated description of ODDR_MODE in <a href="#">Table 2-10</a>. Updated description of CNTVALUEOUT[8:0] in <a href="#">Table 2-11</a>. Updated delay line procedures in <a href="#">VAR_LOAD Mode, page 185</a> and <a href="#">VAR_LOAD Mode, page 221</a>. Updated descriptions of CNTVALUEOUT[8:0] and RST in <a href="#">Table 2-15</a>. Added description of RXTX_EN_VTC to first paragraph of <a href="#">FIXED Mode, page 204</a> and <a href="#">FIXED Mode, page 220</a>. Updated description of RXTX_BITSLICE.EN_VTC port in <a href="#">VARIABLE Mode, page 204</a>. In <a href="#">FIFO Function</a>, clarified that flip-flop for FIFO_RD_EN is optional and only required when timing cannot be met. Updated descriptions of RX_CNTVALUEOUT[8:0] and TX_CNTVALUEOUT[8:0] in <a href="#">Table 2-22</a>. Updated descriptions of CNTVALUEOUT[8:0] and CNTVALUEOUT_EXT[8:0] in <a href="#">Table 2-24</a>. Updated description of CNTVALUEOUT[8:0] in <a href="#">Table 2-26</a> and <a href="#">Table 2-28</a>. Updated description of TBYTE_IN[3:0] in <a href="#">Table 2-30</a>. Updated description of TX_GATING in <a href="#">Table 2-31</a>. Updated <a href="#">step 5b in Release Reset, page 296</a>. Updated description of bits 7 and 6:0 in <a href="#">Table 2-42</a>. Updated description of bit 13 in <a href="#">Table 2-44</a> and <a href="#">Table 2-45</a>.</p> <p><b>Chapter 3:</b> Added POD to list of standards in <a href="#">Internal V<sub>REF</sub>, page 342</a>.</p>
08/28/2019	1.12	<p><b>Chapter 1:</b> Updated fifth paragraph in <a href="#">Introduction to the UltraScale Architecture</a>. Updated <a href="#">I/O Tile Overview</a>. Added DQS_BIAS to <a href="#">Table 1-48</a>.</p> <p><b>Chapter 2:</b> Added note about delay ratio register to <a href="#">VARIABLE Mode</a>. Added note about VT compensation to TX_CNTVALUEIN[8:0] description in <a href="#">Table 2-22</a>. Added note about changing delays using VT compensation to CNTVALUEIN[8:0] description in <a href="#">Table 2-26</a>.</p> <p><b>Appendix B:</b> Added <a href="#">Zynq UltraScale+ RFSoc Data Sheet: Overview (DS889)</a> to <a href="#">References</a>.</p>
07/02/2019	1.11	<p><b>Chapter 1, SelectIO Interface Resources.</b> Updated N and P sides under <a href="#">DQS_BIAS</a>. Added LVDS_PRE_EMPHASIS and EQUALIZATION attributes to <a href="#">Table 1-75</a>, and the attributes are explained for Vivado™ Design Suite version 2019.1.1.</p> <p><b>Chapter 2, SelectIO Interface Logic Resources.</b> Added a note to the CE port and to the CE and LOAD ports in <a href="#">Table 2-11</a>, <a href="#">Table 2-15</a>, <a href="#">Table 2-24</a>, and <a href="#">Table 2-26</a>. Updated Count mode in the TX_DELAY_VALUE attribute of <a href="#">Table 2-23</a>. Updated I/O in <a href="#">Figure 2-30</a>. Updated <a href="#">DQS_BIAS</a>. Updated <a href="#">Receiver Setup</a>. Updated <a href="#">Figure 2-30</a> title.</p>
01/16/2019	1.10	<p><b>Chapter 1, SelectIO Interface Resources.</b> Updated <a href="#">LVDS and LVDS_25</a>.</p> <p><b>Chapter 2, SelectIO Interface Logic Resources.</b> Clarified port descriptions for Component Primitives. Removed the IDELAYCTRL reference from the COUNT mode reset sequence on <a href="#">page 174</a>. Under <a href="#">Release Reset</a>, EN_VTC applies to IDELAYS/ODELAYS. Under <a href="#">Native Input Delay Type Usage</a>, clarified that the EN_VTC applies to RXTX_BITSLICE. In <a href="#">Table 2-25</a>, corrected IS_CLK_INVERTED/IS_RST_DLY_INVERTED attributes. Clarified that the 8 tap restriction is the limit and delays can be changed from 1 to 8 taps.</p>

Date	Version	Revision
06/12/2018	1.9	<p><a href="#">Chapter 1, SelectIO Interface Resources</a>. Updated <a href="#">DQS_BIAS</a>. Starting with Vivado™ Design Suite 2018.1, the DQS_BIAS attribute must be set on the port, not on the cell. Added notes that drivers and receivers must be at the same voltage level. Added the last row (OUTPUT_IMPEDANCE) to <a href="#">Table 1-48</a>. Deleted <a href="#">Figure 2-20: COUNT Mode with Fast Updates</a>.</p> <p><a href="#">Chapter 2, SelectIO Interface Logic Resources</a>. Updated <a href="#">Table 2-4</a>. Clarified reserved bits and defaults in <a href="#">Table 2-39</a> through <a href="#">Table 2-61</a>.</p> <p><a href="#">Chapter 3, High Density I/O Resources</a>. Added the <a href="#">ZHOLD</a> heading. Updated <a href="#">DDR Outputs (ODDRE1)</a>.</p>
02/07/2018	1.8	<p>In <a href="#">Chapter 2</a>, updated BITSlice and waveform information. Updated <a href="#">Figure 2-2</a>, <a href="#">Figure 2-12</a>, <a href="#">Figure 2-13</a>, <a href="#">Figure 2-15</a>, and <a href="#">Figure 2-20</a>. Changed RXTX_BITSLICE to ISERDES in <a href="#">Figure 2-21</a>. Added Count mode with fast update information and <a href="#">Figure 2-23</a>. Updated <a href="#">Figure 2-20</a> and <a href="#">Figure 2-25</a>. Updated <a href="#">Figure 2-38</a>, <a href="#">Figure 2-41</a>, and <a href="#">Figure 2-42</a>. The Values and Description changed in <a href="#">Table 2-18</a>. The <a href="#">Component Mode Reset Sequence</a> was updated. Added <a href="#">Table 2-19</a>, Bidirectional Support by I/O Bank. Updated <b>Caution</b> on <a href="#">page 306</a> and <a href="#">Tip</a> on <a href="#">page 307</a>. Updated <b>Caution</b> on <a href="#">page 316</a>. Updated <a href="#">Table 2-22</a> for ports RX_RST and FIFO_RD_EN and TX_RST. Updated <a href="#">Table 2-22</a> attribute RX_REFCLK_FREQUENCY and TX_REFCLK_FREQUENCY. Updated <a href="#">Table 2-24</a> port RST. Updated <a href="#">Table 2-26</a> port RST. Updated <a href="#">Table 2-28</a> port RST. Updated <a href="#">Table 2-29</a> attribute REFCLK_FREQUENCY. Updated the REFCLK port and BITSlice_CONTROL port REFCLK in <a href="#">Table 2-30</a>. Updated the REFCLK_SRC attribute and BITSlice_CONTROL attribute in <a href="#">Table 2-31</a>. Updated <a href="#">Table 2-38</a> port RIU_WR_EN. Updated <a href="#">Table 2-40</a>, <a href="#">Table 2-58</a>, and <a href="#">Table 2-60</a>.</p>

Date	Version	Revision
07/28/2017	1.7	<p>This book was updated for UltraScale™ and UltraScale+™ devices. The SIM_DEVICE attribute was added to several bit slice attribute tables.</p> <p>Chapter 2: Extensive clarifications were made, including clarifications for BITSlice_0 restrictions. Updated delay line procedures when multiple updates are required for component primitives and native primitives. For component primitives, updated primitive port descriptions and attributes. For native primitives, updated primitive port descriptions and attributes.</p> <p>Additionally, added IS_C_INVERTED, IS_CB_INVERTED in <a href="#">Table 2-2</a>. Added IS_D1_INVERTED, IS_D2_INVERTED in <a href="#">Table 2-4</a>. Added waveforms to describe latencies (<a href="#">Figure 2-12</a>, <a href="#">Figure 2-13</a>). Updated CLK_B description (<a href="#">Table 2-6</a>). Added IS_CLK_INVERTED, IS_CLK_B_INVERTED, IS_RST_INVERTED and SIM_DEVICE (<a href="#">Table 2-7</a>). Added OSERDES latency (<a href="#">Figure 2-14</a>). Updated T description in <a href="#">Table 2-9</a>. Added IS_CLK_INVERTED, IS_CLKDIV_INVERTED, IS_RST_INVERTED, and SIM_DEVICE (<a href="#">Table 2-10</a>). Updated REFCLK_FREQUENCY requirements for TIME mode in the <a href="#">IDELAYE3</a> section. Updated descriptions for RST, EN_VTC, DELAY_VALUE, and DELAY_FORMAT (<a href="#">Table 2-12</a>). Added SIM_DEVICE (<a href="#">Table 2-12</a>). Updated delay procedures for updating VARIABLE and VAR_LOAD to describe multiple updates (DELAY_TYPE descriptions in IDELAY, ODELAY, and <a href="#">Native Input Delay Type Usage</a>). Updated descriptions for RST and EN_VTC (<a href="#">Table 2-15</a>). Updated DELAY_VALUE for UltraScale+ devices (<a href="#">Table 2-16</a>). Added SIM_DEVICE (<a href="#">Table 2-16</a>). Updated Variable Mode waveform (<a href="#">Figure 2-25</a>) for multiple updates. Added <a href="#">Table 2-18</a>. Removed DELAY_TYPE=FIXED from <a href="#">Component Mode Reset Sequence</a>. Updated <a href="#">Figure 2-27</a> to show BUFGCE_DIV usage from a single MMCM clock. Added MMCM descriptions for clock outputs and <a href="#">Figure 2-28</a>. Updated REFCLK_FREQUENCY requirements in <a href="#">Mixing Native and Non-Native Mode I/O in a Nibble</a>. Added latency waveforms for RX_BITSLICE (<a href="#">Figure 2-38</a>, <a href="#">Figure 2-39</a>) and TX_BITSLICE (<a href="#">Figure 2-41</a>, <a href="#">Figure 2-42</a>,). Updated <a href="#">Figure 2-43</a> for multiple updates. Updated descriptions and added TX_OUTPUT_PHASE_90 restriction for <a href="#">Table 2-23</a>. Added UltraScale+ device values and added SIM_DEVICE to <a href="#">Table 2-23</a>. Added UltraScale+ device values for <a href="#">Table 2-25</a>. Added SIM_DEVICE to <a href="#">Table 2-25</a>. Updated T, TBYTE_IN descriptions for <a href="#">Table 2-26</a>. Added UltraScale+ device values for <a href="#">Table 2-27</a>. Added TX_OUTPUT_PHASE_90 restriction for <a href="#">Table 2-27</a>. Added latency figures for TX_BITSLICE_TRI (<a href="#">Figure 2-57</a>, <a href="#">Figure 2-58</a>), Added UltraScale+ device values and TX_OUTPUT_PHASE_90 restriction for <a href="#">Table 2-29</a>. Added SIM_DEVICE (<a href="#">Table 2-29</a>, <a href="#">Table 2-31</a>). Highlighted strobe operation during reset sequence should either be held in reset (updated <a href="#">Figure 2-61</a>) or bitslip is required. Added Native Mode Bring-up when using multiple banks and when a bank uses multiple interfaces. Added SIM_DEVICE to <a href="#">Table 2-23</a>. Added RIU Registers 0x37, 0x38, and 0x39 (<a href="#">Table 2-58</a>, <a href="#">Table 2-59</a>, and <a href="#">Table 2-60</a>).</p>

Date	Version	Revision
10/25/2016	1.6	<p>Chapter 1: Updated <a href="#">DCI—Only Available in the HP I/O Banks</a>. Updated sections in <a href="#">Uncalibrated Input Termination in I/O Banks</a>, <a href="#">IBUF_IBUFDISABLE</a>, <a href="#">IBUF_INTERMDISABLE</a>, <a href="#">IBUFDS_DIFF_OUT_IBUFDISABLE</a>, <a href="#">IBUFDS_DIFF_OUT_INTERMDISABLE</a> and many more.</p> <p>Chapter 2: This chapter was rewritten for clarification. In the <a href="#">Component Primitives</a> section, replaced IDELAY_CTRL with IDELAYCTRL. Reorganized <a href="#">Native Primitives</a> section and updated sub-sections with many clarifications. Added Synchronous Clock Domain columns to port tables <a href="#">Table 2-22</a>, <a href="#">Table 2-24</a>, <a href="#">Table 2-26</a>, and <a href="#">Table 2-30</a>.</p> <p><b>Note:</b> Because the new Chapter 2 organization changed or deleted previous figure and table numbers, references in subsequent rows of this revision history table were accurate as of the date the version was printed.</p> <p>Chapter 3: Updated <a href="#">HD I/O Bank Features</a> and <a href="#">HD I/O Interface Logic</a>.</p>
11/24/2015	1.5	<p>Added the Virtex™ UltraScale+™ family, the Kintex™ UltraScale+ family, and Zynq™ UltraScale+ MPSoCs to this user guide.</p> <p>Chapter 1: Added <a href="#">IBUFDS_DPHY</a>, <a href="#">OBUFDS_DPHY</a>, and <a href="#">MIPI D-PHY</a> sections. Updated the OBUFDS_DPHY slew rate in <a href="#">Table 1-75</a>. Added the MIPI_DPHY_DCI standard with Note 5 to <a href="#">Table 1-77</a>. Added the MIPI_DPHY_DCI standard with Note 6 to <a href="#">Table 1-78</a> and changed the MIPI slew rate to FAST.</p> <p>Chapter 2: Updated <a href="#">Figure 2-59</a>, <a href="#">Figure 2-69</a>, and <a href="#">Figure 2-70</a>.</p> <p>Added <a href="#">Chapter 3, High Density I/O Resources</a> and all references to HD I/O.</p>
11/03/2015	1.4	<p><b>Note:</b> Table and figure numbers were accurate for the 1.4 version.</p> <p>Chapter 1: Added <a href="#">Internal Differential Termination Behavior in Differential I/O Standards</a> section.</p> <p>Chapter 2: Updated the description in the IDELAYE3 section. Updated the RST port description in <a href="#">Table 2-11</a> and <a href="#">Table 2-15</a>. Updated the Q[7:0] description in <a href="#">Table 2-18</a> and <a href="#">Table 2-22</a>. Reversed the arrow direction for DATAOUT in <a href="#">Figure 2-22</a>. Added TX_RST to <a href="#">Figure 2-34</a>. In the introduction to <a href="#">Figure 2-36</a>, changed T_BYTE_IN to T_BYTE_IN[3:0]. Updated RIU_VALID pin descriptions in <a href="#">Table 2-26</a> and <a href="#">Table 2-28</a>. In <a href="#">Table 2-33</a>, bypass 15:9 is no longer supported. Updated Component Mode Reset Sequence. Updated Native Mode Reset Sequence and removed the Native Mode BITSlice Sequence figure. Updated <a href="#">Figure 2-12</a>, OSERDES Used in SDR Mode. Updated FIFO. Resequenced <a href="#">Table 2-18</a>, <a href="#">Table 2-20</a>, <a href="#">Table 2-22</a>, and <a href="#">Table 2-24</a> to match their preceding figures. Updated data type in <a href="#">Figure 2-47</a>. Updated BITSlice numbering in <a href="#">Figure 2-50</a>.</p>
05/29/2015	1.3	<p><b>Note:</b> Table and figure numbers were accurate for the 1.3 version.</p> <p>In Chapter 1: Updated Supply Voltages for the SelectIO Pins section. Added State of I/Os During and After Configuration section. Updated Special DCI Requirements in Some Banks. Corrected <a href="#">Figure 1-28</a>. Updated the VREF and Internal VREF sections. Updated the Transmitter Pre-Emphasis and LVDS Transmitter Pre-Emphasis sections. Added DATA_RATE section. Added Note 6 to <a href="#">Table 1-51</a>. Added slew to <a href="#">Table 1-52</a> and <a href="#">Table 1-53</a>. Updated <a href="#">Table 1-55</a> (added Note 4 and Note 5). Updated <a href="#">Table 1-56</a> and added <a href="#">Table 1-57</a>. Added clarification to the text before the following tables and updated the tables: <a href="#">Table 1-59</a>, <a href="#">Table 1-61</a>, <a href="#">Table 1-63</a>, <a href="#">Table 1-65</a>, <a href="#">Table 1-67</a>, <a href="#">Table 1-69</a>, <a href="#">Table 1-71</a>, <a href="#">Table 1-73</a>, and <a href="#">Table 1-77</a>.</p> <p>A complete rewrite of Chapter 2 including adding sections on the Register Interface Unit, Built-In Self-Calibration (BISC), and Clocking Considerations.</p>

Date	Version	Revision
08/18/2014	1.2	<p><b>Note:</b> Table, figure, and page numbers were accurate for the 1.2 version.</p> <p>Clarified sections of the SelectIO Resources Introduction and the IBUF_ANALOG description under SelectIO Primitives. Removed RTT_NONE from some possible values for ODT for split-termination DCI on page 28 and page 32. Added Note 1 to Table 1-12. Updated the description under HSUL_12 and DIFF_HSUL_12. Revised the HSUL_12 ODT description in Table 1-48. Moved Table 1-52 and Table 1-53. Added Note 3 to Table 1-55.</p> <p>Updated REFCLK_FREQUENCY in Table 2-12. Updated REFCLK in Table 2-17. Revised the DDR modes in Table 2-5. Updated REFCLK_FREQUENCY in Table 2-16. Removed the DDR 2:1 ratio in Table 2-8. In Table 2-27, updated CTRL_CLK. Updated REFCLK_FREQUENCY in Table 2-19.</p>

Date	Version	Revision
05/08/2014	1.1	<p><b>Note:</b> Table and figure numbers were accurate for the 1.1 version. Added features to Table 1-1 and Note 3. Revised the Differences from Previous Generations section. Added clarification to various sections with regards to the OUTPUT_IMPEDANCE attribute. Updated the default for the DCIUpdateMode option to ASREQUIRED. An example discussion added below Table 1-9. Removed VREF tuning from the IBUFDSE3 and IOBUFDSE3 primitives. Added IBUF_ANALOG, IOBUF_INTERMDISABLE, and IBUFDS_DIFF_OUT_INTERMDISABLE to SelectIO Primitives, page 42. Throughout Chapter 1, removed IBUFG (clock input buffer) and updated Figure 1-18, removed IBUFGDS (differential clock input buffer) and updated Figure 1-22, and removed IBUFGDS_DIFF_OUT (differential clock input buffer with complementary outputs) and updated Figure 1-23.</p> <p>Updated the descriptions and some figures and tables: IBUF_IBUFDISABLE, IBUF_INTERMDISABLE, IBUFE3, IBUFDS_DIFF_OUT_IBUFDISABLE, IBUFDS_IBUFDISABLE, IBUFDS_INTERMDISABLE, IBUFDSE3, IOBUF_DCIEN, IOBUFE3, IOBUFDS, IOBUFDS_DCIEN, IOBUFDS_DIFF_OUT, IOBUFDS_DIFF_OUT_DCIEN, IOBUFDS_INTERMDISABLE, IOBUFDS_DIFF_OUT_INTERMDISABLE, IOBUFDSE3, HPIO_VREF, IBUF_LOW_PWR Attribute, Output Slew Rate Attributes, Differential Termination Attribute, Internal VREF, DQS_BIAS, Transmitter Pre-Emphasis, LVDS Transmitter Pre-Emphasis, Receiver EQUALIZATION, LVDCI (Low-Voltage Digitally Controlled Impedance), HSLVDCI (High-Speed LVDCI), HSTL (High-Speed Transceiver Logic), Table 1-49, Table 1-50, Table 1-52, Table 1-53, Table 1-56, and Figure 1-83.</p> <p>Added IBUFDS_DIFF_OUT_IBUFDISABLE, IOBUF_INTERMDISABLE, Source Termination Attribute (OUTPUT_IMPEDANCE), Table 1-13, Table 1-14, and VREF_CNTR.</p> <p>Added the MEDIUM attribute to the HP I/O bank primitives in Table 1-20, Table 1-21, Table 1-22, Table 1-24, Table 1-36, Table 1-37, Table 1-44, Table 1-45, Table 1-48, Table 1-51, and Table 1-78. Updated columns in Table 1-55. Added clarifications to the DQS_BIAS discussion on page 127. Removed SUB_LVDS_25 and replaced with SUB_LVDS on page 132 and throughout the remaining tables including Table A-1. Removed attributes from Table 1-73. Updated the discussion in Rules for Combining I/O Standards in the Same Bank. Added Note 3 and Note 4 to Table 1-77. Added Note 5 to Table 1-78. See Chapter 2 revisions on next page.</p> <p>Updated Figure 2-2. Updated the IDELAYE3 and ODELAYE3 discussions. Updated Table 2-17 and Table 2-1. In Table 2-12, Table 2-16, Table 2-19, and Table 2-21, clarified descriptions for DELAY_VALUE (DELAY_VALUE_EXT), DELAY_FORMAT, and UPDATE_MODE. In Table 2-7, updated the DATA_WIDTH description. Updated the SerDes Output Data Bits to Use in Table 2-5. Added Type column to Table 2-16 and Table 2-4. In Table 2-28, updated the RIU_VALID port description and the port widths and descriptions for the BIT_CTRL ports. In Table 2-27, updated the SERIAL_MODE description, the READ_IDLE_COUNT[5:0] default value, the ROUNDING_FACTOR type, CTRL_CLK, and added new attributes: SELF_CALIBRATE, IDLY_VT_TRACK, ODLY_VT_TRACK, QDLY_VT_TRACK, and RXGATE_EXTEND. Updated Figure 2-24. Removed the CLK_OUT port from Table 2-18 and updated RX_BIT_CTRL_IN[39:0], through TX_BIT_CTRL_OUT[39:0]. In Table 2-19, updated values for DELAY_VALUE, REFCLK_FREQUENCY, DATA_WIDTH, and added the UPDATE_MODE_EXT attribute. Updated Figure 2-29. In Table 2-20, updated the BITSlice_CONTROL ports. Table 2-21, updated values for DELAY_VALUE, REFCLK_FREQUENCY, and added the ENABLE_PRE_EMPHASIS attribute.</p>

Date	Version	Revision
12/10/2013	1.0	Initial Xilinx release.

## Please Read: Important Legal Notices

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes. THIS INFORMATION IS PROVIDED "AS IS." AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

### 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 2013–2025 Advanced Micro Devices, Inc. AMD, the AMD Arrow logo, Artix, Kintex, Spartan, UltraScale, UltraScale+, Virtex, Vivado, Zynq, and combinations thereof are trademarks of Advanced Micro Devices, Inc. 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.