Spartan UltraScale+ FPGAs SelectIO Resources

User Guide

UG861 (v1.0) December 23, 2024

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 Resource Overview

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 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 16 nm 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.

AMD 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 16 nm 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.

I/O Tile Overview

The AMD Spartan™ UltraScale+™ devices provide high-performance (HP), high-density (HD), and XP5IO offerings. Although HP I/O and HD I/O resources are similar to other UltraScale+ device resources, the Spartan UltraScale+ family has subtle differences, which along with the unique XP5IO warrants this unique SelectIO™ user guide specific to the Spartan UltraScale+ family.

***Note*:** For all other UltraScale+ families, refer to *UltraScale Architecture SelectIO Resources User Guide* (UG571).

- The HP I/O banks are designed to meet the performance requirements of high-speed memory and other chip-to-chip interfaces with voltages between 1.0V and 1.8V.
- The HD I/O banks are designed to support low-speed interfaces with voltages between 1.2V and 3.3V.
- The XP5IO banks are designed to enhance I/O support for LPDDR5 interfaces as well as improving performance on high-speed interfaces like MIPI D-PHY. XP5IO supports supplies between 1.0V and 1.5V.

Spartan UltraScale+ devices contain different combinations of HP, HD, and XP5IO banks, and not all types of banks are supported in all devices. The *UltraScale Architecture and Product Data Sheet: Overview* (DS890) documents the available number of each type of bank for all devices.

The Spartan UltraScale+ family has high-performance I/O banks (HP and XP5IO) with MIPI D-PHY capabilities and the corresponding logic resources. They also have high-density I/Os (HD I/Os) with corresponding logic resources.

- Chapter 1: SelectIO Interface Resource Overview 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 IOB Technology and Supported Standards describes features related to the drive and receive structures and the supported standards by bank type.
- Chapter 3: HP I/O Bank SelectIO Interface Logic Resources describes the I/O logic resources available in these devices for HP I/Os.
- Chapter 4: HD I/O Interface Logic Resources describes the electrical and logical features of HD I/Os.
- Chapter 5: XP5IO I/O Interface Logic Resources gives an overview of the XP5IO high-speed interface resources.

The following table highlights the features supported in the HP, HD, and XP5IO banks. See *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) for details on the performance and other electrical requirements of the HP, HD, and XP5IO banks.

Table 1: **Supported Features in the HD, HP, and XP5IO Banks**

Feature	HP I/O Banks	HD I/O Banks	XP5IO Banks
3.3V I/O standards[1]	N/A	Supported	N/A
2.5V I/O standards[1]	N/A	Supported	N/A
1.8V I/O standards[1]	Supported	Supported	N/A
1.5V I/O standards[1]	Supported	Supported	Supported
1.35V I/O standards[1]	Supported	Supported	Supported
1.2V I/O standards[1]	Supported	Supported	Supported
1.0V I/O standards	Supported	Not supported	Supported
LVDS signaling[2]	Supported at 1.8V	Supported (input only)	Supported at 1.5V
Digitally-controlled impedance (DCI)	Supported	Not supported	Supported
Internal V_{REF}	Supported	Supported	Supported
Internal differential termination (DIFF_TERM)	Supported	Not supported	Supported
Input delay	Supported	Not supported	Supported
Output delay	Supported	Not supported	Supported
Dedicated serialization resource	Supported (8:1, 4:1)	Not supported	Supported (8:1, 4:1, 2:1)
Dedicated deserialization resources	Supported (1:8, 1:4)	Not supported	Supported (1:8, 1:4, 1:2)

Table 1: **Supported Features in the HD, HP, and XP5IO Banks** *(cont'd)*

Feature	HP I/O Banks	HD I/O Banks	XP5IO Banks
Transmitter pre-emphasis	Supported	Not supported	Supported
Receiver equalization	Supported	Not supported	Supported
Receiver offset control	Supported	Not supported	Not Supported
Receiver V_{REF} scan	Supported	Not supported	Supported
MIPI D-PHY	Supported	Not supported	Supported
LPDDR5 compatibility	Not supported	Not supported	Supported

Notes:

1. The I/O Bank Type column in Table 61 shows the specific I/O standards that are available in the HP I/O banks.
2. Although LVDS is generally considered a 2.5V I/O standard, it is supported at 1.8V in HP I/O banks and 1.5V in XP5IO banks.

Differences from Previous Generations

Spartan 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 for HP I/O and HD I/O include:

- Each HP I/O bank contains 52 SelectIO interface pins, HD banks contain 42 SelectIO interface pins, and XP5IO banks contain 66 interface pins.
- HP and XP5IO banks add support for pseudo-open-drain (POD) logic standards.
- Series output termination control is available in HP and XP5IO I/O banks for improved signal integrity and ease of board design.
- Internal or external V_{REF} is available in HP I/O banks, while only internal V_{REF} is available in HD and XP5IO I/O banks.
- Pre-emphasis is available for several standards in HP and XP5IO I/O banks. Pre-emphasis reduces inter-symbol interference and minimizes the effects of transmission line losses.
- Linear equalization on several V_{REF} -based receivers (in HP and XP5IO 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 available in HP and XP5IO I/O banks. DCI requires one reference resistor per bank in HP and one per device in XP5IO. The values of the driver or input termination are determined by the OUTPUT_IMPEDANCE and on-die termination (ODT) attributes, respectively.
- Uncalibrated ODT (input termination) is available in both HP and HD I/O banks.
- A SLEW value of *MEDIUM* is supported in HP and XP5IO I/O banks.

- The DCITERMDISABLE port can control both DCI and non-DCI on-die input termination features in HP and XP5IO 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.
- In HP I/O, 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.
- MIPI D-PHY transmitter and receiver functions are supported in the HP and XP5IO I/Os.

Note: XP5IO are a new I/O type introduced in Spartan UltraScale+ devices. They are not referenced in this section.

Differences Between Bank Types

As noted in I/O Tile Overview, the Spartan UltraScale+ family consists of different bank types: HP I/O, HD I/O, XP5IO. The bank types are optimized with different features, supply levels, applications, and performance levels. The following descriptions call out fundamental differences in the bank type to help define an appropriate bank type selection for a given application.

HP I/O Bank

The HP I/O bank is intended for higher performance interfaces that are powered from 1.0V to 1.8V. This includes higher performance interface standards like LVDS, POD, SSTL, and MIPI. The HP I/O banks have basic register resources but also contain a PHY with a robust feature set that helps accommodate high-performance fourth generation memory interfaces as well as high-performance MIPI D-PHY interfaces. The HP I/O bank supports high-speed interfaces by leveraging several features: a dedicated PHY with clock-data phase alignment features, calibrated internal termination, receive CTLE equalization, pre-emphasis, and support for several I/O standards used in DDR4/LPDDR4 memory, as well as DPHY.

HD I/O Bank

The HD I/O bank is primarily intended to accommodate lower-speed interfaces that might require basic register function with a wide range of voltage support levels (1.2V to 3.3V). The HD I/O banks support basic SDR/DDR dedicated registers, uncalibrated termination, and zero hold (ZHOLD) which can be used to preserve clock alignment.

XP5IO Bank

The XP5IO bank is unique to the Spartan UltraScale+ architecture. It operates at a supply range between 1.0V and 1.5V and adds LPDDR5 interface support to the Spartan UltraScale+ family. In addition, the XP5IO contains a dedicated PHY which can be used for source-synchronous applications like MIPI D-PHY. XP5IO banks are intended for high-speed PHY operation limited to MIPI D-PHY and LPDDR5 interfaces. There are no IOL resources like IDDR, ODDR, or IODELAY, though the PHY can be bypassed to access fabric resources from the IOB.

Table 2: **Feature Differences by Bank Type**

Feature	HP I/O Banks	HD I/O Banks	XP5IO Banks
Supported Supply Voltage Range	1.0V to 1.8V	1.2V to 3.3V	1.0V to 1.5V
PHY Support	ISERDES/OSERDES	Not Supported	PHY
Memory Support	DDR4	Not Supported	LPDDR5
MIPI Support	D-PHY	Not Supported	D-PHY
Internal Termination	Calibrated and Uncalibrated (40Ω, 48Ω, 60Ω), Differential Input Termination	Uncalibrated 48 Ω Termination Only	Calibrated (40Ω, 48Ω, 60Ω), Differential Input Termination
Equalization	CTLE	Not Supported	CTLE
LVDS Signaling Support	Input and Output (1.8V)	Input only externally terminated LVDS support	Input and Output (1.5V)
Data Alignment Features	High Speed Dynamic Phase Alignment through dedicated PHY	ZHOLD	High Speed Dynamic Phase Alignment through dedicated PHY
Dedicated Clock Managers	PLL/MMCM	Not Supported	PLL/MMCM

Chapter 2

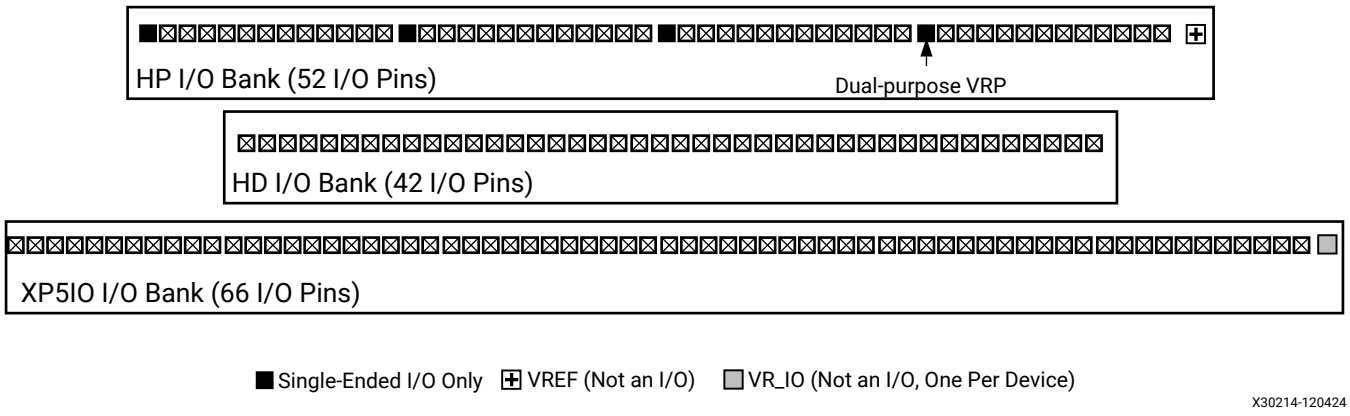
SelectIO IOB Technology and Supported Standards

All Spartan 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, and the ability to internally generate a reference voltage (INTERNAL_VREF). Each bank type in Spartan UltraScale+ device might not support all features and IOSTANDARDS. Implementation details for HD and HP I/O banks are provided in this section, while a summary of resources for the XP5IO are provided at the end of this section.

Bank Differences

The following figure shows an overview of the HP, HD, and XP5IO banks to illustrate bank differences.

Figure 1: **Bank Diagrams**



The following figure illustrates the general IOB structure that applies to all three bank types.

Figure 2: **Single-Ended (Only) IOB Diagram**

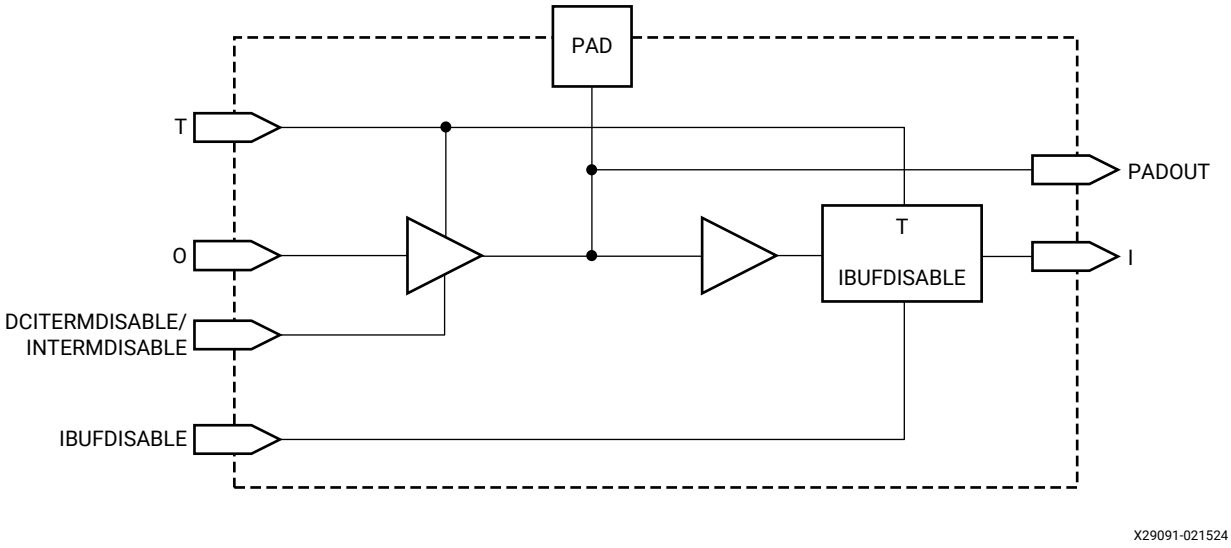
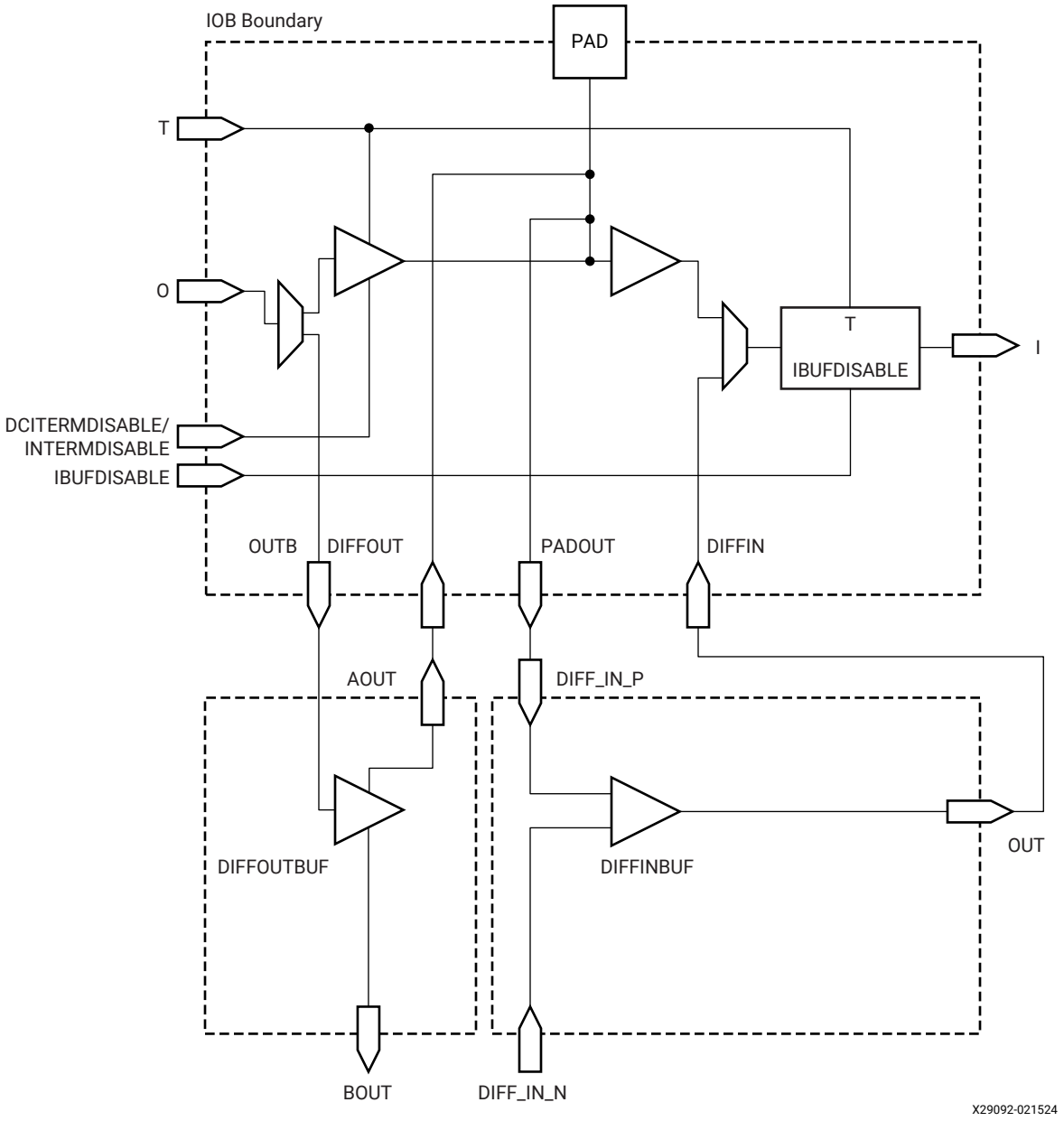


Figure 3: **Standard IOB Diagram**



HP I/O Bank Overview

With some exceptions, each HP 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 HP I/O SelectIOresource 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, HSTL, SSTL, HSUL, and POD.

- Differential I/O standards are, for example, 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. Each HP bank contains an optional VREF pin which can be used to provide an external VREF source to inputs that leverage VREF to define their switching thresholds. Figure 1 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. Figure 2 shows the single-ended (only) HP I/O block (IOB) and its connections to the internal logic and the device pad. Figure 3 shows the standard HP IOB.

Each HP 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, RDES, 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 3: HP I/O Bank SelectIO Interface Logic Resources for more information.

HD I/O Bank Overview

HD I/O banks are grouped into 42 I/O pins capable of either 42 single-ended signals, or 21 differential signals. HD I/O support both single-ended and differential signaling capabilities and have input, output, and 3-state driver conditions. HD I/O only support internal VREF and uncalibrated termination so there are no dedicated pins, and each non-supply pin can function as a single-ended or differential I/O. Within a bank there are two HDGC pin pairs which can function as a normal I/O but also have dedicated connections to the global clock network. HDGC pins can function in both differential or single-ended signal standards. When used as a single-ended signaling standard, only the positive side of the differential pair can connect directly to the global clocking network.

The HD I/O SelectIO pins can be configured to various I/O standards, both single-ended and differential.

- Single-ended I/O standards are, for example, LVCMOS, LVTTL, HSTL, and SSTL.
- Differential I/O standards that support drive and receive capabilities are differential HSTL and SSTL.
- Differential I/O standards that are input only are LVDS and LVPECL.

The IOB diagrams above shows the standard HD IOB but do not support the DCITERMDISABLE port, only INTERMDISABLE. When not configured, I/O drivers are 3-stated, and I/O receivers are weakly pulled-down.

Each HD IOB has a direct connection to input and output register resources for clock, data, and 3-state control. See Chapter 4: HD I/O Interface Logic Resources for more information.

XP5IO I/O Bank Overview

The XP5IO I/O banks (when present) are optimized to provide Spartan UltraScale+ devices with a high-speed PHY that can interface with high-speed interfaces used in LPDDR5 and MIPI D-PHY interfaces. The XP5IO I/O bank is grouped into 11 nibbles of six I/O pins, giving a total of 66 I/O pins capable of either 66 single-ended signals, or 33 differential signals. XP5IO I/O support both single-ended and differential signaling capabilities and have input, output, and 3-state driver conditions. XP5IO I/O only support internal VREF and require a single termination reference (VR) for all XP5IO banks in a device. A bank contains four clock capable I/O (CCIO) which can function as a normal I/O but also have dedicated connections to the global clock network. CCIO pins can function in both differential or single-ended signal standards. When used as a single-ended signaling standard, only the positive side of the differential pair can connect directly to the global clocking network.

The XP5IO SelectIO pins can be configured to various I/O standards, both single-ended and differential, powered from 1.0V to 1.5V:

- Single-ended I/O standards are LVCMOS, HSTL, SSTL, POD, and LVSTL.
- Differential I/O standards that support drive and receive capabilities are differential HSTL, SSTL, POD, LVSTL, LVDS, and MIPI_DPHY.

There are no dedicated low-speed register resources in the XP5IO, so when the PHY is bypassed, only direct access to programmable fabric resources is available, leaving the XP5IO only suitable for very low performance when not used with the PHY. See Chapter 5: XP5IO I/O Interface Logic Resources for more information.

SelectIO Interface General Guidelines

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

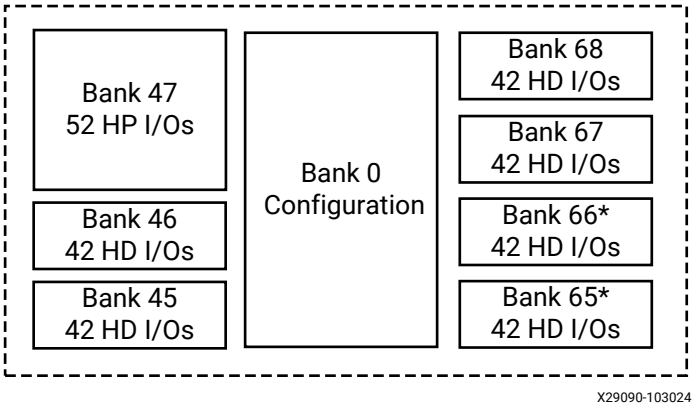
I/O Bank Rules

HP I/O banks consist of 52 IOBs, while HD I/O banks consist of 42 pin banks. The number and distributions of banks depends upon the device size and the package pinout. In the *UltraScale Architecture and Product Data Sheet: Overview* (DS890), the total number of available I/O is listed by device type. The following figure is an example of a typical floorplan. The *UltraScale and UltraScale+ FPGAs Packaging and Pinouts Product Specification* (UG575) includes information on the I/O banks for each device/package combination. The following figure illustrates one example

of a Spartan UltraScale+ device (SU35P). Not all devices have the same bank types available, but bank 0 is dedicated to configuration and cannot be used for general-purpose interfaces using logic described in this user guide. Devices might have one (bank 65) or two dual-purpose banks (bank 65 and bank 66) which can be used as user logic if not used as part of the configuration interface.

IMPORTANT! *Dual-purpose banks (like bank 65 and bank 66 in the following figure) contain pins that might be used as part of the configuration interface. Bank 66 might not always have dual-purpose configuration functionality in all devices. Some restrictions may apply. See Spartan UltraScale+ FPGAs Configuration User Guide (UG860) for more information on these restrictions.*

Figure 4: **Bank Diagram Example**



Supply Voltages for the SelectIO Pins

V_{CCO}

The V_{CCO} supply is the primary power supply of the I/O circuitry. The V_{CCO} (V) columns in Table 60 provide the V_{CCO} requirements for each of the supported I/O standards, and illustrate the V_{CCO} requirements for inputs and outputs as well as the optional internal differential termination circuit.

All V_{CCO} pins for a given XP5IO, HP, or HD I/O bank must be connected to the same external voltage supply on the board, and as a result, all of the I/O within a given I/O bank must share the same V_{CCO} level. The V_{CCO} 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.*

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 HP 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)). HD and XP5IO I/O banks only support INTERNAL_VREF so there are no V_{REF} pins in HDIO or XP5IO I/O banks. An internally generated reference voltage is enabled by using the INTERNAL_VREF constraint. HDIO only support a V_{REF} of 50% V_{CCO} for a given bank, while XP5IO can have unique V_{REF} levels on a nibble wide resolution. 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.

IMPORTANT! *In HP I/O 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 Ω or 1 K Ω resistor.*

In HP I/O 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 Ω or 1 KΩ 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 (HD 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 *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930).

V_{CCAUX_HDIO}, V_{CCAUX_HPIO}, and V_{CCAUX_XP5IO}

The auxiliary I/O (V_{CCAUX_HDIO}, V_{CCAUX_HPIO}, and V_{CCAUX_XP5IO}) voltage supply rail provides power to the I/O circuitry.

V_{CCINT_IO}

This is an internal supply for HP and XP5IO I/O banks. Connect to the V_{CCBRAM} voltage supply rail.

State of I/Os During and After Configuration

Spartan UltraScale+ devices have pins dedicated to the configuration functions contained in I/O bank 0. There are also I/O pins in bank 65/66 (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.

During configuration, the PUDC_B (active-Low) input enables internal pull-up resistors on the SelectIO pins after power-up and during configuration, including multipurpose configuration pins when not used for the selected configuration mode. When PUDC_B is Low, internal pull-up resistors are enabled on each SelectIO pin. When PUDC_B is High, internal pull-up resistors are disabled on each SelectIO pin. PUDC_B must be tied either directly, or via a ≤ 1 kΩ resistor, to VCCO_0 or GND.

DCI 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 the impedance inside the I/O to calibrate to an external precision reference resistor placed on the VRP (HP) or VR (XP5IO) pin. This compensates for changes in I/O impedance due to process variation. HP banks can continuously adjust the impedance to compensate for variations of temperature and supply voltage fluctuations. Each HP bank that uses DCI requires a DCI reference VRP pin, while there is only one VR pin that is shared across all XP5IO banks (when present).

IMPORTANT! *In HP banks for DCI I/O standards that are used in the bank, the external reference resistor (R_{VRP}) should be 240 Ω.*

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 HD 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 Ω.*

To implement DCI in an HP I/O:

1. Assign one of the DCI I/O standards in an HP I/O bank (see Table 4).
2. Connect the VRP multi-function pin to a precision resistor (240 Ω) 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.

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.

In HP I/O, 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 HP 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 in HP I/O banks. 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 in HP I/O during normal operation of the design. This primitive is required in a design when DCIUpdateMode is set to QUIET (see DCIUpdateMode Configuration Option). See the *UltraScale Architecture Libraries Guide* (UG974) for more details on the DCIRESET primitive.

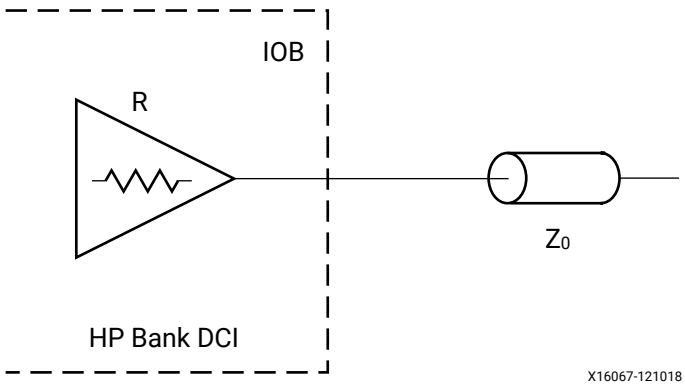
Controlled Impedance Driver (Source Termination)

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.

The following figure illustrates a controlled impedance driver inside a device.

Figure 5: **Controlled Impedance Driver**



In HP banks, the use of controlled impedance DCI drivers is triggered by specific standards supporting the controlled impedance driver as shown in the following table.

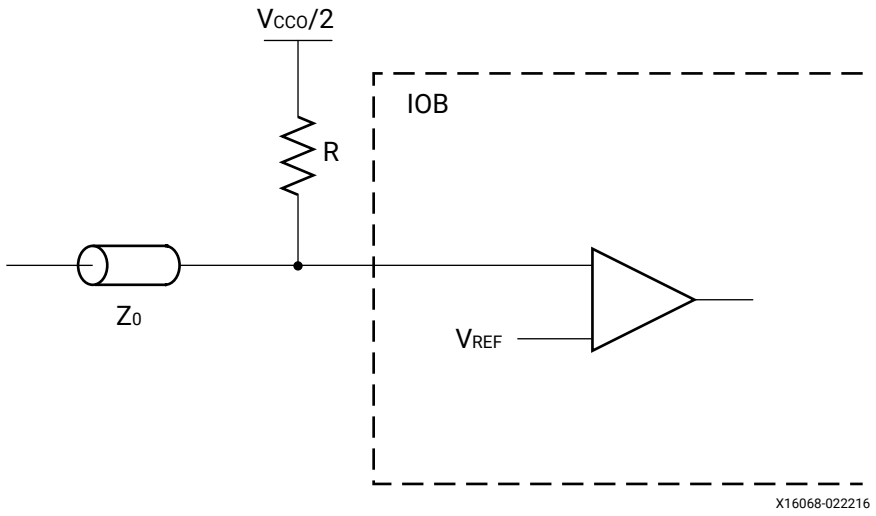
Table 3: **All DCI I/O Standards Supporting Controlled Impedance Driver**

HSTL	DIFF_HSTL	LVDCI	HSUL	DIFF_HSUL	SSTL18	DIFF_SSTL
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 $V_{CCO}/2$)

Some I/O standards (HSTL and SSTL) require an input termination resistance (R) to a V_{TT} voltage of $V_{CCO}/2$ (see the following figure).

Figure 6: **Input Termination to V_{CCO} / 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_{CCO}, the other to GND. Split-termination DCI provides an equivalent termination to $V_{CCO}/2$ using this method. The 2R termination resistance is set by programming the ODT attribute. The resistors to V_{CCO} and GND are equal to twice the value set by ODT. For example, to achieve the Thevenin equivalent parallel-termination circuit of approximately 50 Ω to $V_{CCO}/2$, a 240 Ω 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.

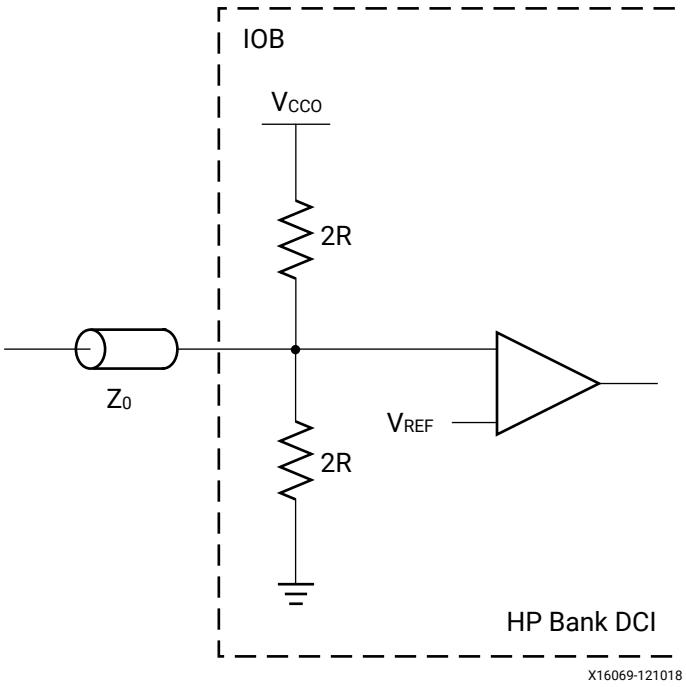
In HP banks, the following IOSTANDARDS must be used to trigger split input termination.

Table 4: **HP DCI I/O Standards Supporting Split-Termination DCI**

HSTL	DIFF_HSTL	SSTL	DIFF_SSTL
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
		SSTL135_DCI	DIFF_SSTL135_DCI
		SSTL12_DCI	DIFF_SSTL12_DCI

The following figure illustrates split-termination DCI.

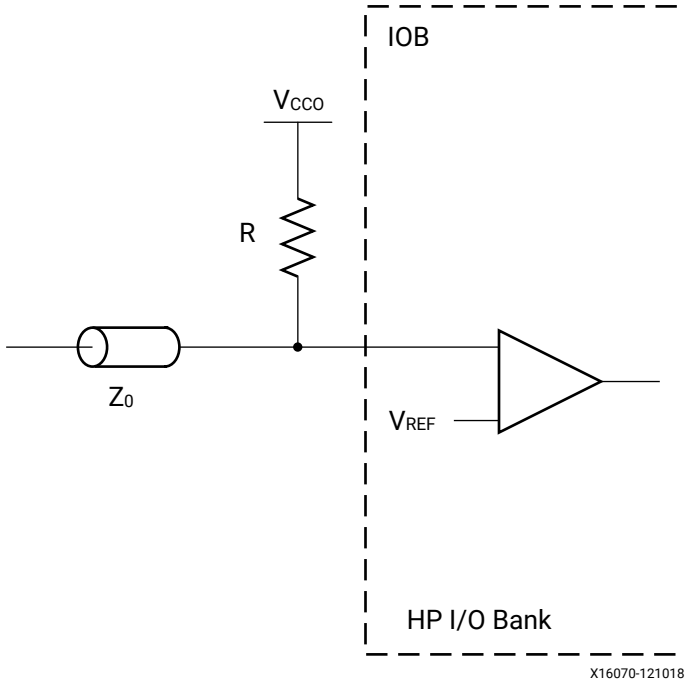
Figure 7: **Input Termination to V_{CCO}/ 2 Using Split-Termination DCI (where R = Z_0)**



Single-Termination DCI

Some I/O standards (POD10, POD12, HSUL_12, and DIFF_HSUL_12) require an input termination resistance (R) to a VTT voltage of V_{CCO} (see the following figure).

Figure 8: **Input Termination to V_{CCO} without DCI (where R = Z_0)**



HP I/O DCI input standards supporting single termination are shown in the following table.

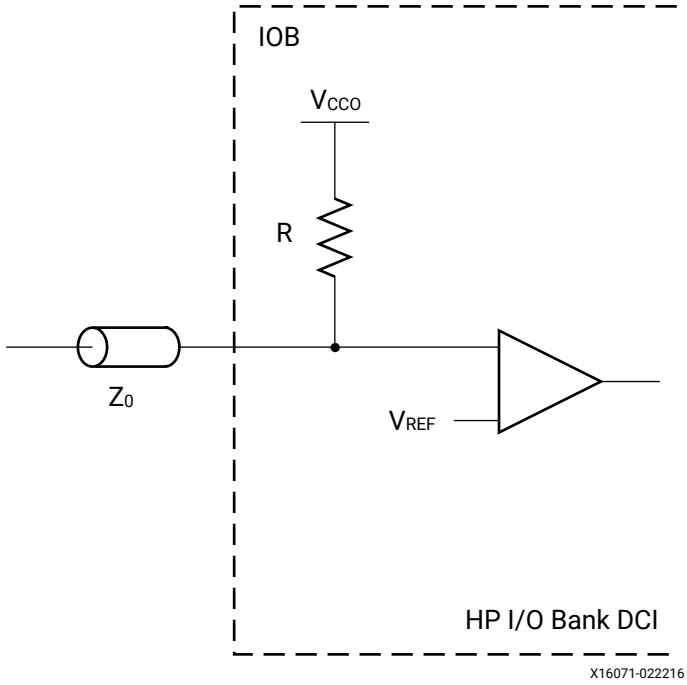
Table 5: **All DCI I/O Standards Supporting Single-Termination DCI**

POD	DIFF_POD	HSUL/DIFF_HSUL
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_{CCO} internally as shown in the following figure. The 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 9: **Input Termination to V_{CCO} using Single-Termination DCI (where R = Z_0)**



For example, to achieve single termination of approximately 50 Ω to V_{CCO} for the POD12_DCI standard, a 240 Ω external precision resistor is required at the VRP pin, and ODT must be set to the value of RTT_48.

In XP5IO, single termination to V_{CCO} is supported in POD and HSUL through the ODT attribute. In addition to single termination to V_{CCO}, XP5IO support several standards that leverage single termination to ground with the LVSTL I/O standards. Enabling an ODT attribute enables single termination to ground.

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 Ω to $V_{CCO}/2$ requires 100 Ω 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 Ω. 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 $V_{CCO}/2$ 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_{CCO} 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 SSTL I/O configurations with 3-state support for internal input split-termination. Those T_DCI standards are not supported in Spartan UltraScale+ devices. However, in HP banks, internally terminated SSTL12 standards are capable of supporting similar bidirectional configurations. For example, in 7 series FPGAs, SSTL12_T_DCI has the same functionality as SSTL12_DCI in HP I/O banks.

DCI I/O Standard Support

DCI supports the standards shown in the following table.

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

LVDCI	HSTL	DIFF_HSTL	SSTL	DIFF_SSTL
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			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 Ω, 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 Ω). 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 HP and HD I/O Banks

The HD I/O and HP I/O banks have an optional uncalibrated input on-chip split-termination feature for HSTL and SSTL standards. For HP I/O banks, a single-termination feature also exists for POD and HSUL standards that is 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_{CCO} and the other to GND, providing a Thevenin equivalent termination circuit of half the resistor value to the mid-point V_{CCO} /2 for HSTL and SSTL standards. A single resistor terminates to V_{CCO} 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 HD 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 in HP I/O banks for split-termination standards (HSTL and SSTL) are RTT_40, RTT_48, RTT_60, or RTT_NONE, while HD I/O banks only support RTT_48 termination.
- Possible values in HP I/O banks for ODT for single-termination POD standards are RTT_40, RTT_48, RTT_60, or RTT_NONE.
- Possible values in HP I/O banks 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.

The following table shows a list of I/O standards that support the uncalibrated termination in HP I/O banks.

***Note*:** HD I/O banks only support ODT internal termination of RTT_48.

Table 7: **I/O Standards that Support Uncalibrated Termination in HP I/O Banks**

HSTL	DIFF_HSTL	SSTL	DIFF_SSTL	POD/HSUL	DIFF_POD
HSTL_I	DIFF_HSTL_I	SSTL18_I	DIFF_SSTL18_I	POD12	DIFF_POD12
HSTL_I_18	DIFF_HSTL_II	SSTL18_II	DIFF_SSTL135	POD10	DIFF_POD10
	DIFF_HSTL_I_18	SSTL135	DIFF_SSTL15	HSUL_12	DIFF_HSUL_12
	DIFF_HSTL_II_18	SSTL15	DIFF_SSTL12		
		SSTL12			

The following table shows the I/O standards that support uncalibrated termination in HD I/O banks.

Table 8: **I/O Standards that Support Uncalibrated Termination in HD I/O Banks (RTT_48 Only)**

HSTL	SSTL	DIFF_HSTL	DIFF_SSTL
HSTL_I	SSTL18_I	DIFF_HSTL_I	DIFF_SSTL18_I
HSTL_I_18	SSTL15	DIFF_HSTL_I_18	DIFF_SSTL135
	SSTL135		DIFF_SSTL15
	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 a 40 Ω, 48 Ω, or 60 Ω 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.

The following table shows a list of I/O standards that support uncalibrated source termination in HP I/O banks.

***Note*:** HD I/O banks do not support source termination control.

Table 9: **I/O Standards that Support Uncalibrated Source Termination in HP I/O Banks**

HSTL	DIFF_HSTL	SSTL	DIFF_SSTL	POD/HSUL	DIFF_POD
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 ±35 mV). This feature can be accessed through IBUFE3, IBUFDSE3, IOBUFE3, and IOBUFDSE3 primitives as shown in the following two figures. Offset calibration requires building control logic into your interconnect logic design.

Figure 10: **Offset Calibration Connections for Single-Ended I/O Standards**

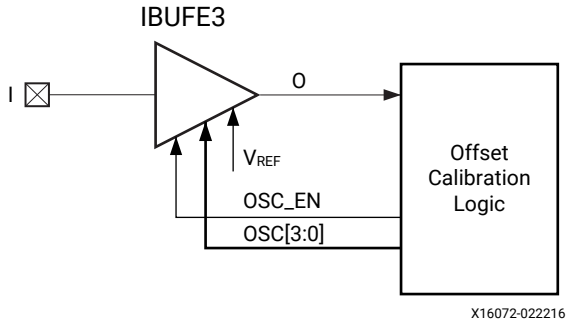
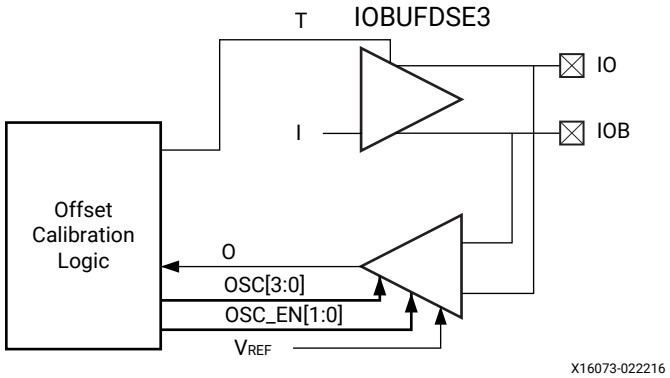


Figure 11: **Offset Calibration Connections for Differential I/O Standards**



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.
3. 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. The following table shows the approximate amount of offset cancellation provided by each setting of OSC.

Table 10: **Approximate Amount of Offset Cancellation for Each Setting of OSC**

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

4. 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.
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 the following table.

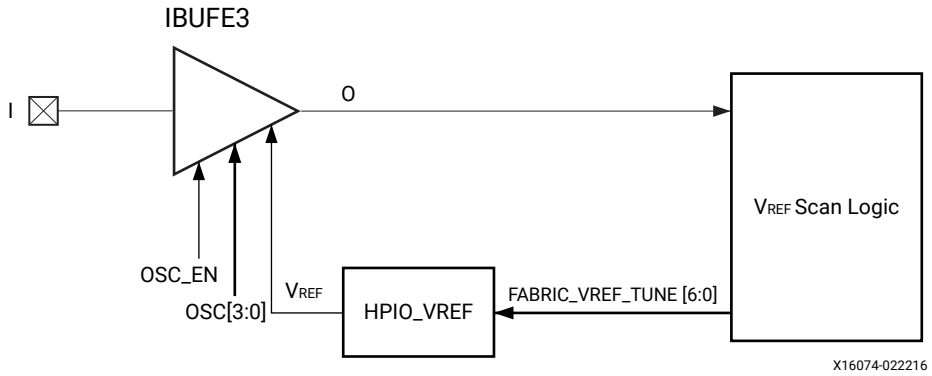
Table 11: **I/O Standards Supporting Receiver Offset Control**

POD	DIFF_POD
POD12	DIFF_POD12
POD12_DCI	DIFF_POD12_DCI

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

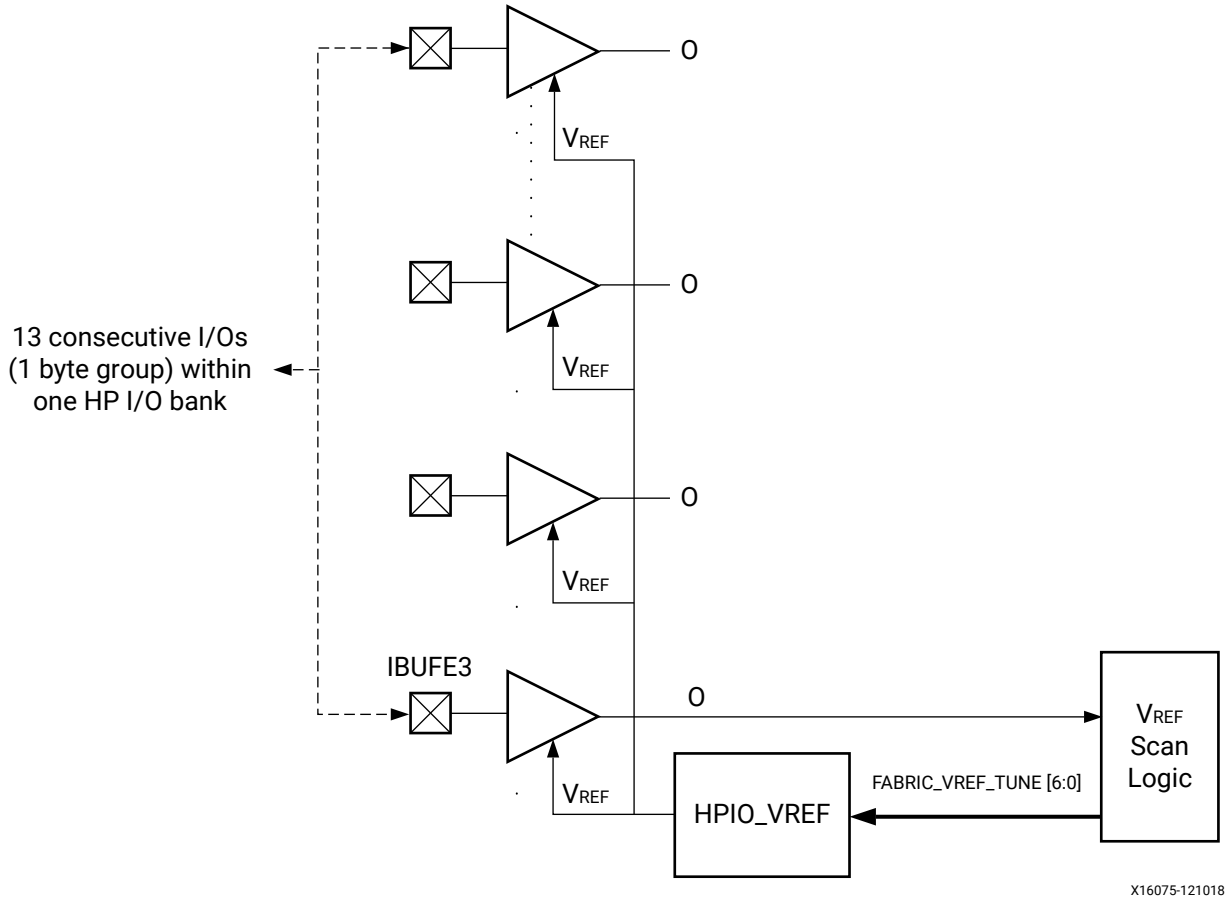
An optional V_{REF} scan feature in HP I/O banks helps to fine tune the internal V_{REF} of input buffers to maximize the performance for a subset of I/O standards. This feature can be accessed through the IBUFE3 and IOBUFE3 primitives with the HPIO_VREF primitive as shown in the following figure. V_{REF}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

Figure 12: **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 the following figure. 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 13: **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 here:

- 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 the following table.

Table 12: **Approximate V_{REF} Value as a Result of Using the V_{REF} Scan Function**

FABRIC_TUNE_VREF[6:0]	V_{REF} (% of V_{CCO})	
	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%
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%

Table 12: **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_{CCO})	
	VREF_CNTR = FABRIC_RANGE1	VREF_CNTR = FABRIC_RANGE2
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%
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 IOB Interface Primitives

The Vivado Design Suite library includes an extensive list of primitives supporting many I/O standards available in the IOB 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 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)
- 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))
- 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_IBUFDISABLE (differential input buffer with buffer disable control)
- 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.)
- 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))

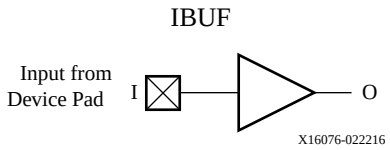
- IOBUFDS_DIFF_OUT (differential bidirectional buffer with complementary outputs from the input buffer)
- IOBUFDS_DIFF_OUT_DCIEN (differential bidirectional buffer with complementary outputs from the input buffer with on-die input termination disable controls and input buffer disable controls (HP I/O banks only))
- 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.)
- 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).

IBUF

Signals used as inputs must use an input buffer (IBUF). The generic IBUF primitive is shown in the following figure.

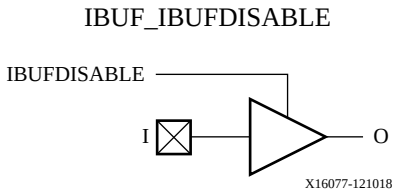
Figure 14: **Input Buffer Primitive (IBUF)**



IBUF_IBUFDISABLE

The IBUF_IBUFDISABLE primitive shown in the following figure 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 15: **Input Buffer with Input Buffer Disable (IBUF_IBUFDISABLE)**

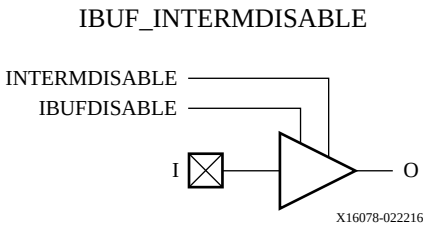


The IBUF_IBUFDISABLE primitive can disable the input buffer and force the O output to the internal logic to a logic Low when the IBUFDISABLE signal is asserted High. The USE_IBUFDISABLE attribute must be set to TRUE, the IBUFDISABLE port must be controlled, and SIM_DEVICE set to ULTRASCALE_PLUS for this primitive to have the expected behavior that is specific to the Spartan UltraScale+ devices. 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 LVTTL.

IBUF_INTERMDISABLE

The IBUF_INTERMDISABLE primitive shown in the following figure is available in the HD 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_PLUS 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 HP and HD I/O Banks for more details about this feature.

Figure 16: **Input Buffer with Input Buffer Disable and On-Die Input Termination Disable (IBUF_INTERMDISABLE)**

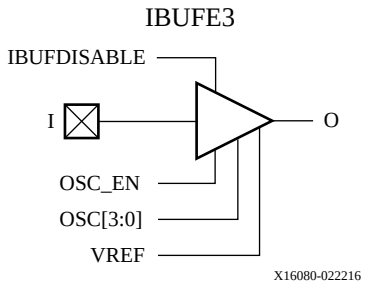


The IBUF_INTERMDISABLE primitive can disable the input buffer and force the O output to the internal logic to a logic-Low when the IBUFDISABLE signal is asserted High. The IBUF_INTERMDISABLE primitive further allows the termination legs to be disabled whenever the INTERMDISABLE signal is asserted High. 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 LVTTL.

IBUFE3

The input buffer (IBUFE3) primitive (shown in the following figure) 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 with IBUFE3.

Figure 17: **IBUFE3 Primitive—Input Buffer with Offset Calibration and V_{REF} Tuning (HP I/O Banks Only)**

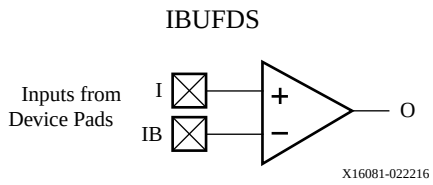


IBUFDS

The usage and rules corresponding to the differential primitives are similar to the single-ended SelectIO primitives. Differential SelectIOprimitives 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.

The following figure shows the differential input buffer primitive.

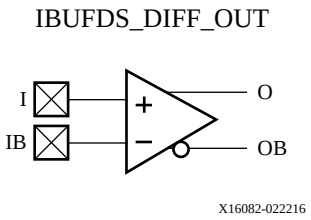
Figure 18: **Differential Input Buffer Primitive (IBUFDS)**



IBUFDS_DIFF_OUT

The following figure shows the differential input buffer primitive with complementary outputs (O and OB).

Figure 19: **Differential Input Buffer Primitive with Complementary Outputs (IBUFDS_DIFF_OUT)**

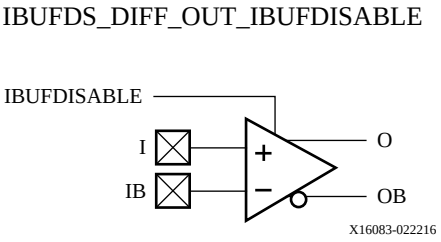


IMPORTANT! *When this primitive is used for a clock signal, the OB output to the interconnect logic has no direct connection to a clock buffer.*

IBUFDS_DIFF_OUT_IBUFDISABLE

The IBUFDS_DIFF_OUT_IBUFDISABLE primitive shown in the following figure is a differential input buffer with complementary differential outputs.

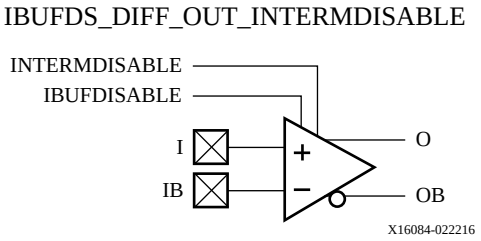
Figure 20: **Differential Input Buffer with Complementary Outputs and Input Buffer Disable (IBUFDS_DIFF_OUT_IBUFDISABLE)**



IBUFDS_DIFF_OUT_INTERMDISABLE

The IBUFDS_DIFF_OUT_INTERMDISABLE primitive shown in the following figure is available in the HD 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 HP and HD I/O Banks for more details.

Figure 21: **Differential Input Buffer with Complementary Outputs, Input Path Disable, and On-Die Input Termination Disable (IBUFDS_DIFF_OUT_INTERMDISABLE)**

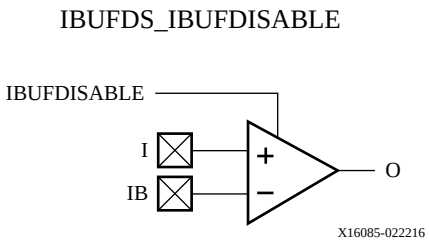


If the I/O is using any on-die receiver termination features (uncalibrated), this primitive disables the termination legs whenever the INTERMDISABLE signal is asserted High.

IBUFDS_IBUFDISABLE

The IBUFDS_IBUFDISABLE primitive shown in the following figure 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 22: **Differential Input Buffer with Input Buffer Disable (IBUFDS_IBUFDISABLE)**

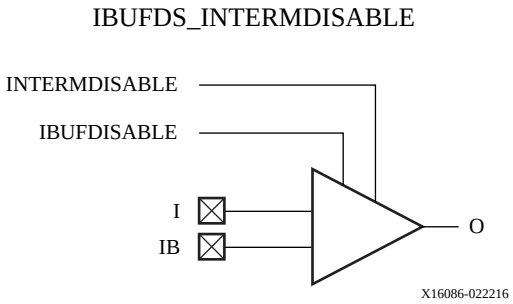


The IBUFDS_IBUFDISABLE primitive can disable the input buffer and force the O output to the internal logic to a logic-Low when the IBUFDISABLE signal is asserted High. The USE_IBUFDISABLE attribute must be set to TRUE, the IBUFDISABLE port must be controlled, and SIM_DEVICE set to ULTRASCALE_PLUS for this primitive to have the expected behavior that is specific to the Spartan UltraScale+ devices. This feature can be used to reduce power whenever the I/O is idle.

IBUFDS_INTERMDISABLE

The IBUFDS_INTERMDISABLE primitive (shown in the following figure), available in the HD 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 HP and HD I/O Banks for more details.

Figure 23: **Differential Input Buffer with Input Buffer Disable and On-Die Input Termination Disable (IBUFDS_INTERMDISABLE)**

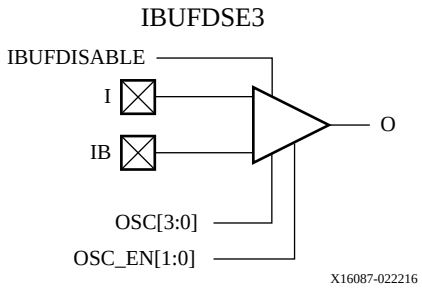


The IBUFDS_INTERMDISABLE primitive can disable the input buffer and force the O output to a logic-Low when the IBUFDISABLE signal is asserted High. The USE_IBUFDISABLE attribute must be set to TRUE, the IBUFDISABLE port must be controlled, and SIM_DEVICE set to ULTRASCALE for this primitive to have the 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 (see the following figure). 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_{REF} scan feature is not supported with this primitive.

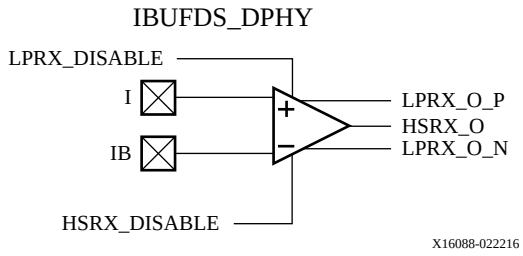
Figure 24: **IBUFDSE3 Primitive—Differential Input Buffer with Offset Calibration (HP I/O Banks Only)**



IBUFDS_DPHY

The differential input buffer primitive (IBUFDS_DPHY) is only supported in the HP I/O banks. 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.

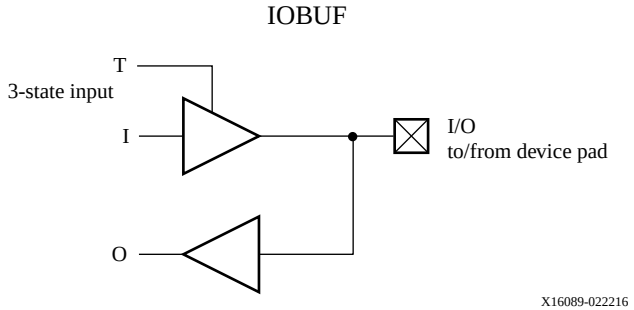
Figure 25: **Differential Input Buffer Primitive (IBUFDS_DPHY)**



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. The following figure 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.

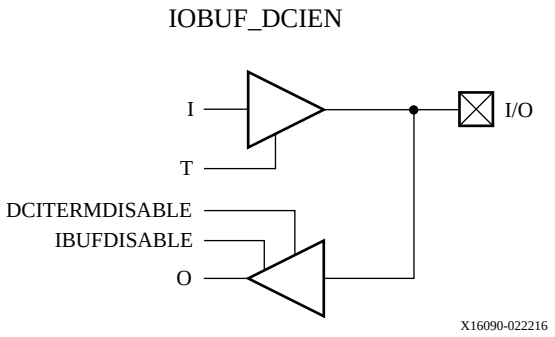
Figure 26: **Input/Output Buffer Primitive (IOBUF)**



IOBUF_DCIEN

The IOBUF_DCIEN primitive shown in the following figure 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_DCIEN primitive also has a DCITERMDISABLE port that can be used to manually disable the optional on-die receiver termination features (uncalibrated and DCI). See DCI in the HP I/O Banks and Uncalibrated Input Termination in HP and HD I/O Banks for more details.

Figure 27: **Input/Output Buffer DCI Enable Primitive (IOBUF_DCIEN)**

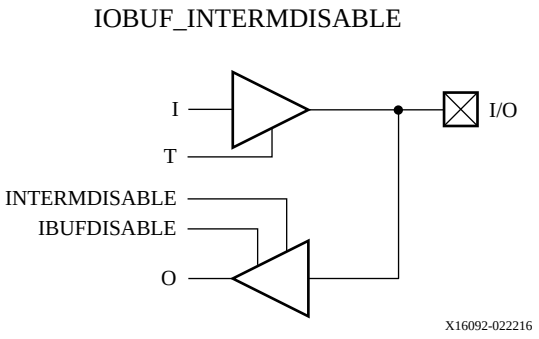


The IOBUF_DCIEN primitive can disable the input buffer and force the O output to the internal logic to a logic-Low when the IBUFDISABLE signal is asserted High and output buffer is 3-stated (T = High). If the I/O is using any on-die receiver termination features (uncalibrated and DCI), this primitive disables the termination legs whenever the DCITERMDISABLE signal is asserted High and the output buffer is 3-stated (T = 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. The USE_IBUFDISABLE attribute must be set to TRUE, the IBUFDISABLE port must be controlled, and SIM_DEVICE set to ULTRASCALE_PLUS for this primitive to have the expected behavior that is specific to the Spartan UltraScale+ devices. 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 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 the following figure is available in the HD 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 HP and HD I/O Banks for more details.

Figure 28: **Bidirectional Buffer with Input Path Disable and On-Die Input Termination Disable (IOBUF_INTERMDISABLE)**

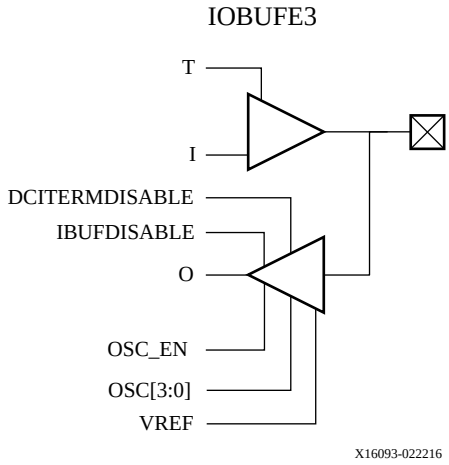


The IOBUF_INTERMDISABLE primitive can disable the input buffer and force the O output to the internal logic to a logic-Low when the IBUFDISABLE signal is asserted High and the output buffer is 3-stated (T = High). If the I/O is using the on-die receiver termination feature (uncalibrated), this primitive disables the termination legs whenever the INTERMDISABLE signal is asserted High and the output buffer is 3-stated (T = High). When the output buffer is 3-stated (T = 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_PLUS for this primitive to have the expected behavior that is specific to the Spartan UltraScale+ devices. When the output buffer is not 3-stated (T = 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 (see the following figure). 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 with IOBUFE3.

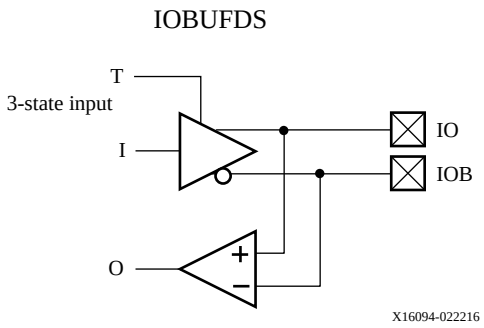
Figure 29: **IOBUFE3 Primitive—Bidirectional I/O Buffer with Offset Calibration and V_{REF} Tuning (HP I/O Banks Only)**



IOBUFDS

The following figure 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.

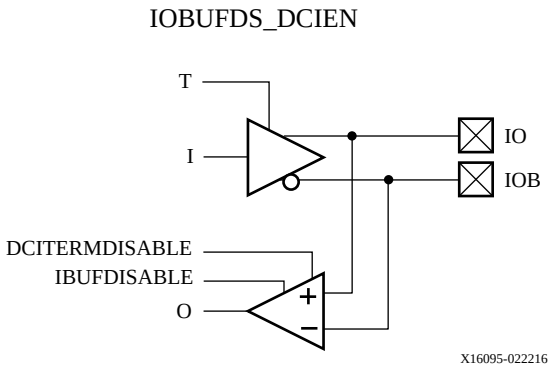
Figure 30: **Differential Input/Output Buffer Primitive (IOBUFDS)**



IOBUFDS_DCIEN

The IOBUFDS_DCIEN primitive shown in the following figure 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_DCIEN 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 in the HP I/O Banks and Uncalibrated Input Termination in HP and HD I/O Banks for more details.

Figure 31: **Differential Bidirectional Buffer with Input Buffer Disable and On-Die Input Termination Disable (IOBUFDS_DCIEN)**



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).

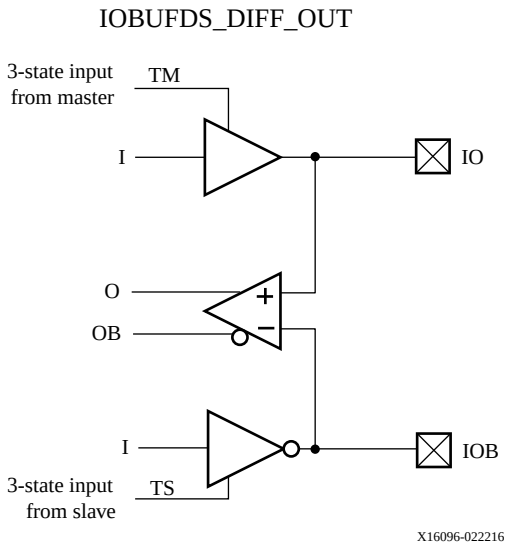
When the output buffer is 3-stated (T = High), the input buffer and any on-die receiver termination (uncalibrated or DCI) are controlled by IBUFDISABLE and DCITERMDISABLE, respectively. When the output buffer is not 3-stated (T = Low), the input buffer and any on-die receiver termination (uncalibrated 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

The following figure 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.

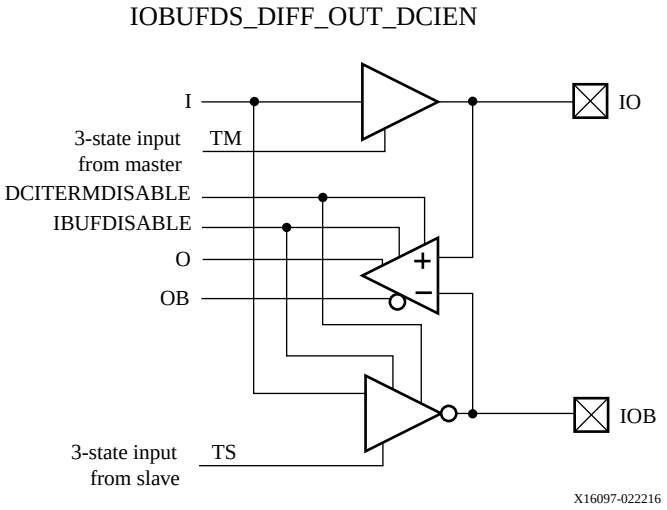
Figure 32: **Differential Input/Output Buffer Primitive with Complementary Outputs for the Input Buffer (IOBUFDS_DIFF_OUT)**



IOBUFDS_DIFF_OUT_DCIEN

The IOBUFDS_DIFF_OUT_DCIEN primitive shown in the following figure 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 in the HP I/O Banks and Uncalibrated Input Termination in HP and HD 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 33: **Differential Bidirectional Buffer with Complementary Outputs, Input Path Disable, and On-Die Input Termination Disable (IOBUFDS_DIFF_OUT_DCIEN)**



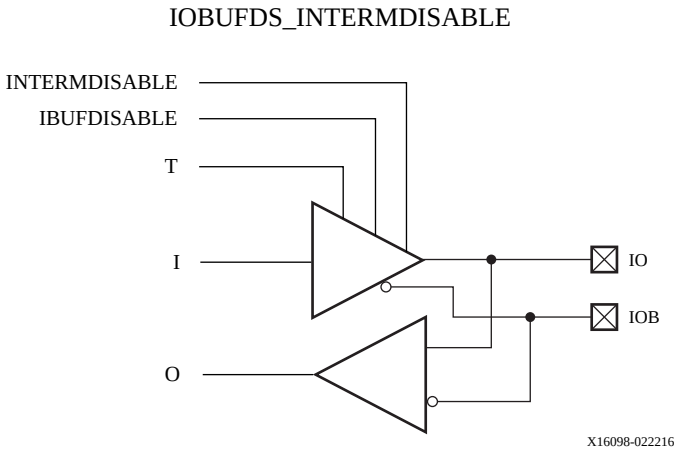
If the I/O is using any on-die receiver termination features (uncalibrated or DCI), this primitive disables the termination legs whenever the DCITERMDISABLE signal is asserted High and the output buffer is 3-stated. When the output buffer is 3-stated (T = High), any on-die receiver termination (uncalibrated 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 the following figure) is available in the HD 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 HP and HD I/O Banks for more details on this feature.

Figure 34: **Differential Bidirectional Buffer with Input Buffer Disable and On-Die Input Termination Disable (IOBUFDS_INTERMDISABLE)**



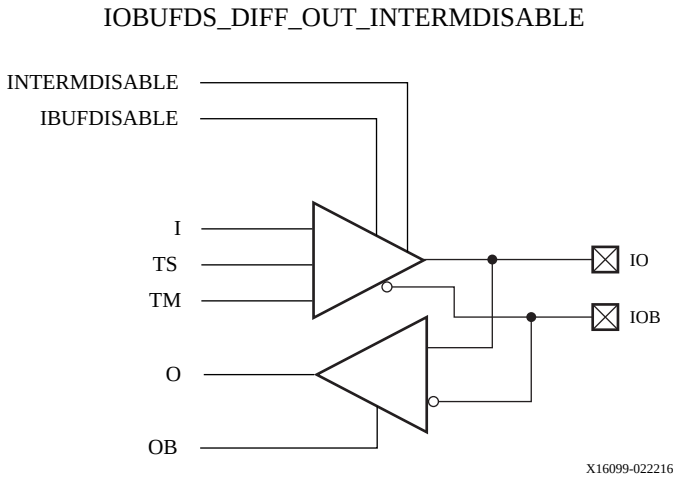
The IOBUFDS_INTERMDISABLE primitive can disable the input buffer and force the O output to the internal logic to a logic-Low when the IBUFDISABLE signal is asserted High and the output buffer is 3-stated (T = High). The USE_IBUFDISABLE attribute must be set to TRUE, the IBUFDISABLE port must be controlled, and SIM_DEVICE set to ULTRASCALE_PLUS for this primitive to have the expected behavior that is specific to the Spartan UltraScale+ devices. 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 the following figure) is available in the HD 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 HP and HD 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 35: **Differential Bidirectional Buffer with Complementary Outputs, Input Buffer Disable, and On-Die Input Termination Disable (IOBUFDS_DIFF_OUT_INTERMDISABLE)**

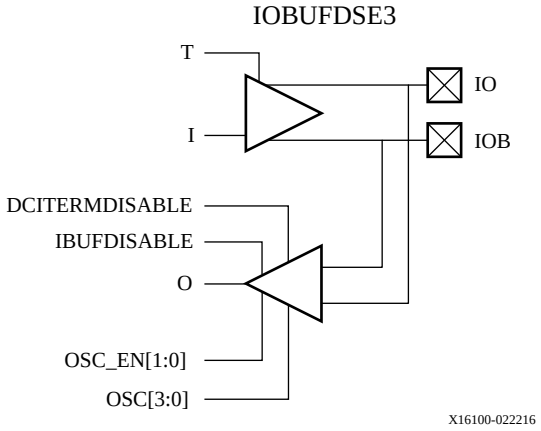


If the I/O is using the on-die receiver termination features, this primitive disables the termination legs whenever the INTERMDISABLE signal is asserted High and the output buffer is 3-stated. When the output buffer is 3-stated (T = High), any on-die receiver termination is controlled by INTERMDISABLE. 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.

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.

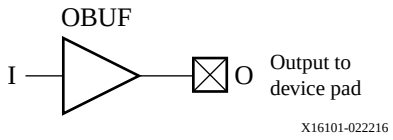
Figure 36: **IOBUFDSE3 Primitive—Differential Bidirectional I/O Buffer with Offset Calibration (HP I/O Banks Only)**



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 the following figure.

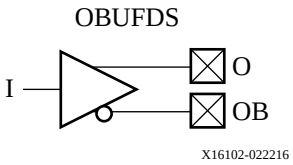
Figure 37: **Output Buffer Primitive (OBUF)**



OBUFDS

The following figure shows the differential output buffer primitive.

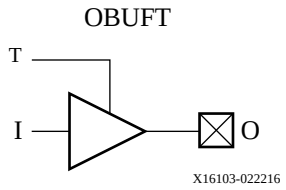
Figure 38: **Differential Output Buffer Primitive (OBUFDS)**



OBUFT

The generic 3-state output buffer OBUFT, shown in the following figure, typically implements 3-state outputs or bidirectional I/O.

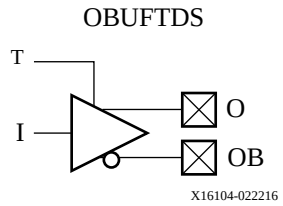
Figure 39: **3-State Output Buffer Primitive (OBUFT)**



OBUFTDS

The following figure shows the differential 3-state output buffer primitive.

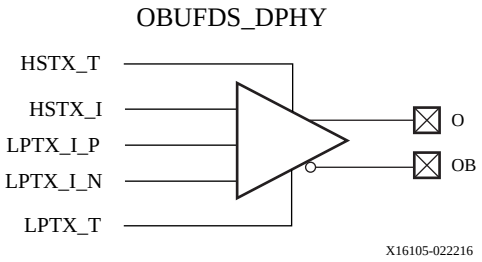
Figure 40: **Differential 3-State Output Buffer Primitive (OBUFTDS)**



OBUFDS_DPHY

The differential output buffer primitive (OBUFDS_DPHY) is only supported in the HP I/O banks. 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.

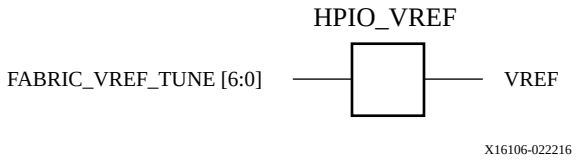
Figure 41: **Differential Output Buffer Primitive (OBUFDS_DPHY)**



HPIO_VREF

The HPIO_VREF primitive is only supported in HP I/O banks (see the following figure). 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 with either the IBUFE3 or IOBUFE3 primitives.

Figure 42: **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).

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 *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930); however, Table 60 lists the IOSTANDARD support by I/O bank type (HP and HD). The IOSTANDARD attribute uses the following syntax in the XDC file:

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

IBUF_LOW_PWR Attribute (HP I/O Only)

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 ≥ 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).

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) 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, LVTTL, 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 LVTTL 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 the following table.

Table 13: **Allowed Values for the DRIVE Attribute**

Standard	HD 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[1]
LVCMOS15	4, 8, or 12	12	2, 4, 6, 8, or 12	12
LVCMOS18	4, 8, or 12	12	2, 4, 6, 8, or 12	12
LVCMOS25	4, 8, or 12	12	N/A	N/A
LVCMOS33	4, 8, or 12	12	N/A	N/A
LVTTL	4, 8, or 12	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).

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. In HP I/O banks the IOSTANDARD selection dictates whether the ODT is using calibrated (DCI) or uncalibrated termination.

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 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 (HD I/O only support 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 (HP I/O banks only) 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 (HP I/O banks only) support the differential I/O standards when used as inputs. These attributes are used to turn the built-in, 100 Ω, 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_{CCO} of the I/O bank must be connected to 1.8V for HP I/O banks to provide 100 Ω of effective differential termination. DIFF_TERM and DIFF_TERM_ADV are only available for inputs and can only be used with the appropriate V_{CCO} 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 in the references or the *UltraScale Architecture Libraries Guide* (UG974) 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 the 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 Spartan 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_{CCO}.

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 46 using HSTL_I (1.5V), which requires a 0.75V reference voltage, uses the following constraint:

```
set_property INTERNAL_VREF 0.75 [get_iobanks 46]
```

Example 2: INTERNAL_VREF for Bank 65 using HSTL_I_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 below. For HD I/O, internal V_{REF} must be set to ½ V_{CCO}:
  - 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 with the HPIO_VREF primitive.

Within an HP I/O 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_V_{REF} or V_{REF} scan is used in a bank, connect the dedicated external V_{REF} pins to GND through a 500 Ω or 1 KΩ resistor.

DQS_BIAS (HP I/O Banks only)

DQS_BIAS (HP I/O banks only) 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_{CCO} and the P side of the buffer to ground. For LVDS inputs, DQS_BIAS provides a DC bias of $V_{CCO}/2$ 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 with the PULLTYPE attribute set to PULLUP, PULLDOWN, or KEEPER in the same port.
- FALSE (default)

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 (HP I/O banks only) allows pre-emphasis on the drivers for certain I/O standards. This attribute must be used 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 40 Ω is listed in the following table.

Table 14: **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 40 Ω.	RDRV_240[1]	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 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 the following table.

Table 15: **Typical Pre-Emphasis Gain when Using the LVDS_PRE_EMPHASIS Attribute**

Attribute	Value	Estimated Gain (dB)
LVDS_PRE_EMPHASIS (HP I/O banks only)	TRUE[1]	4

Notes:

1. ENABLE_PRE_EMPHASIS must be set to TRUE.

Receiver EQUALIZATION

The receiver equalization (EQUALIZATION) feature (HP I/O banks only) allows equalization at the receiver for certain I/O standards. The allowed values for the EQUALIZATION attribute are:

- EQ_LEVEL0
- EQ_LEVEL1
- EQ_LEVEL2
- EQ_LEVEL3
- EQ_LEVEL4
- EQ_NONE (Default)

Receiver OFFSET Control

The receiver offset control (OFFSET_CNTRL) feature (HP I/O banks only) 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 HP 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) 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 in the references section.

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.

LVTTL

Table 16: **Available I/O Bank Type**

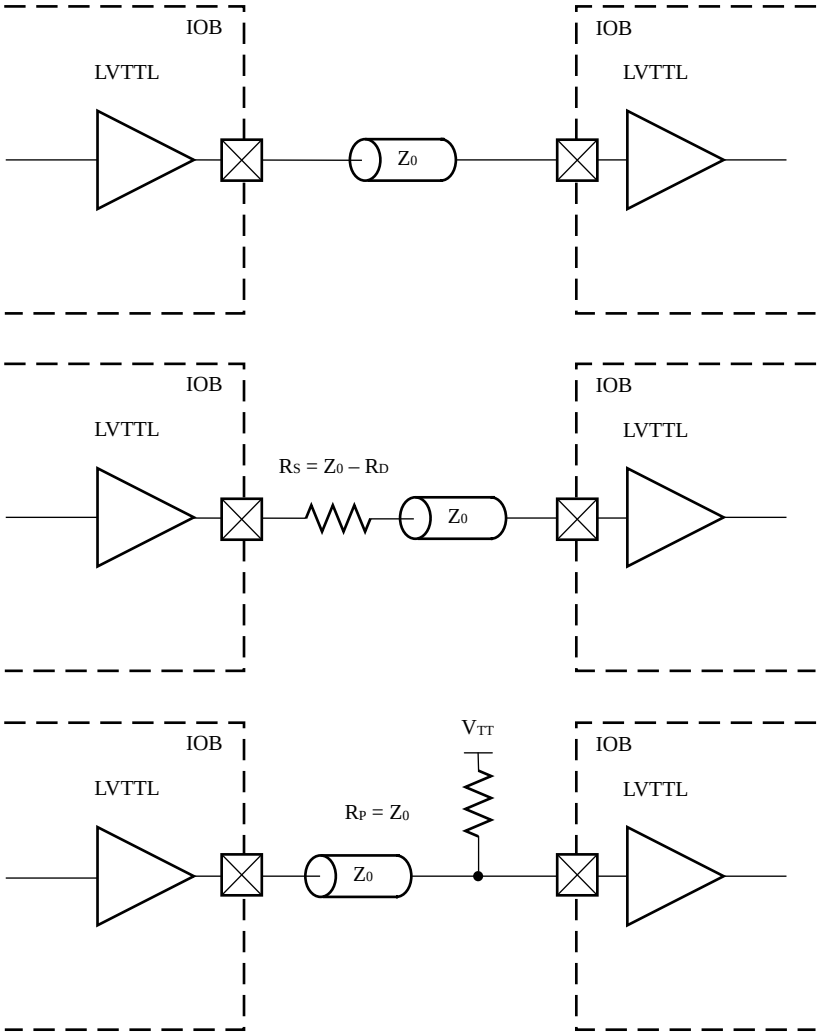
HD	HP
Available	N/A

Low voltage TTL (LVTTL) 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_{CCO}), 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).

Sample circuits illustrating both unidirectional and bidirectional LVTTL termination techniques are shown in the following figures. These two diagrams show examples of source-series and parallel terminated topologies.

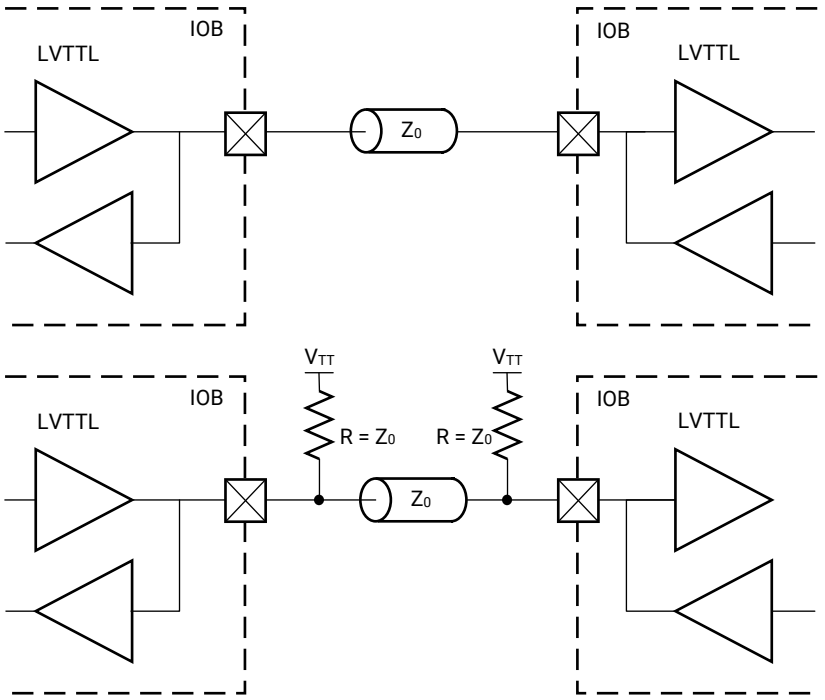
The following figure shows unidirectional terminated topologies.

Figure 43: **LVTTL Unidirectional Termination**



The following figure shows a bidirectional, parallel-terminated topology.

Figure 44: **LVTTL Bidirectional Termination**



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

X16108-022216

The following table details the allowed attributes that can be applied to the LVTTL I/O standard. This standard is only available in the HD I/O banks. Support is implied for primitives that are derivatives of the primitives listed in the following table (for example: *_DIFF_OUT, *_DCIEN, *_IBUFDISABLE, or *_INTERMDISABLE). Refer to SelectIO IOB Interface Primitives for all supported derivatives.

Table 17: **Allowed Attributes for the LVTTL I/O Standards**

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

LVCMOS

Table 18: **Available I/O Bank Type**

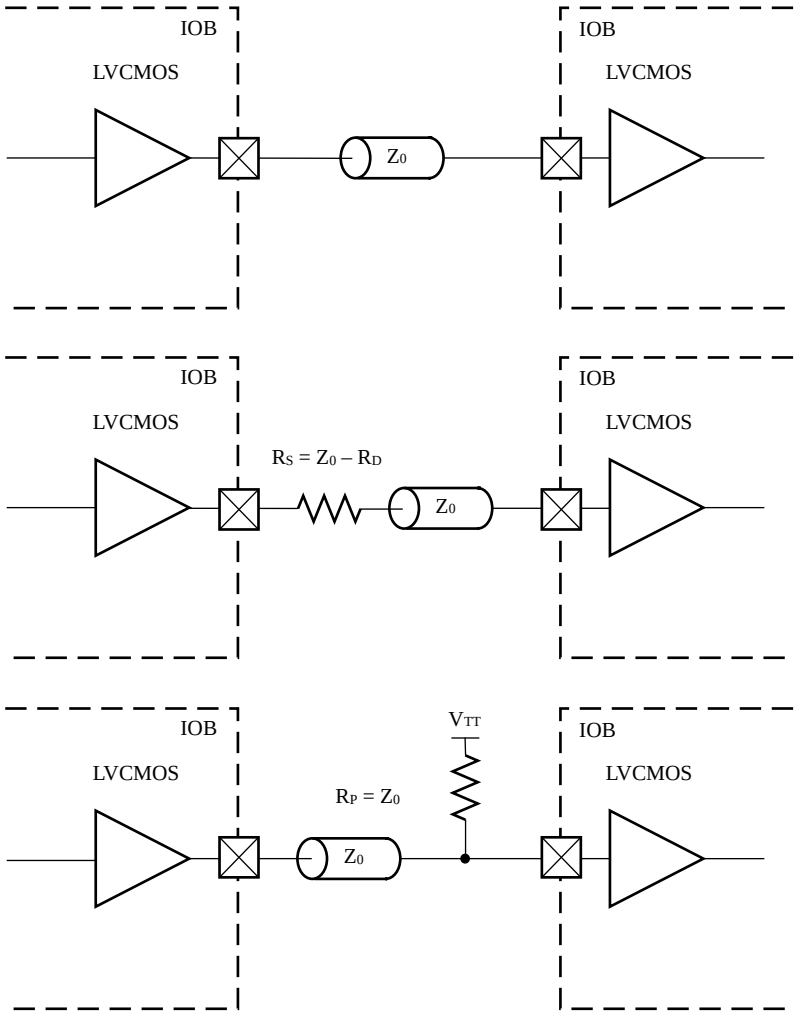
HD	HP
Available	Available

Low voltage CMOS (LVCMOS) 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 (https://www.jedec.org). The LVCMOS standards supported in Spartan UltraScale + devices are: LVCMOS12, LVCMOS15, LVCMOS18, LVCMOS25, and LVCMOS33.

Sample circuits illustrating both unidirectional and bidirectional LVCMOS termination techniques are shown in the following figures. These two diagrams show examples of source-series and parallel terminated topologies.

The following figure shows unidirectional terminated topologies.

Figure 45: **LVCMOS Unidirectional Termination**

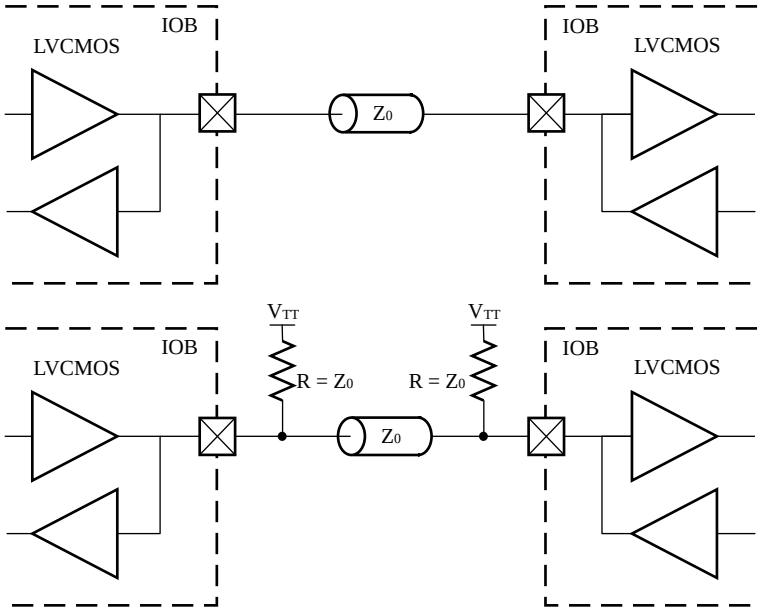


Note: V_{TT} is any voltage from 0V to V_{CCO}

X16109-022216

The following figure shows a bidirectional, parallel-terminated topology.

Figure 46: **LVCMOS Bidirectional Termination**



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

X16110-022216

The following table details the allowed attributes that can be applied to the LVCMOS33 and LVCMOS25 I/O standards. These standards are only available in the HD I/O banks. Support is implied for primitives that are derivatives of the primitives listed in the following table (for example: *_DIFF_OUT, *_DCIEN, *_IBUFDISABLE, or *_INTERMDISABLE). Refer to SelectIO IOB Interface Primitives for all supported derivatives.

Table 19: **Allowed Attributes for the LVCMOS33 and LVCMOS25 I/O Standards in HD I/O Banks**

Attributes	Primitives		
	IBUF	OBUF/OBUFT/IOBUF	
		Allowed Values	Default
IOSTANDARD	LVCMOS33, LVCMOS25	LVCMOS33, LVCMOS25	
DRIVE	N/A	4, 8, or 12	12
SLEW	N/A	FAST or SLOW	SLOW

The following table details the allowed attributes that can be applied to the LVCMOS18 I/O standard. This standard is available in both the HD 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 the following table (for example: *_DIFF_OUT, *_DCIEN, *_IBUFDISABLE, or *_INTERMDISABLE). Refer to SelectIO IOB Interface Primitives for all supported derivatives.

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

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

The following table details the allowed attributes that can be applied to the LVCMOS15 I/O standard. This standard is available in both the HD and HP I/O banks. Support is implied for primitives that are derivatives of the primitives listed in the following table (for example: *_DIFF_OUT, *_DCIEN, *_IBUFDISABLE, or *_INTERMDISABLE). Refer to SelectIO IOB Interface Primitives for all supported derivatives.

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

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

The following table details the allowed attributes that can be applied to the LVCMOS12 I/O standard. This standard is available in both the HD and HP I/O banks. Support is implied for primitives that are derivatives of the primitives listed in the following table (for example: *_DIFF_OUT, *_DCIEN, *_IBUFDISABLE, or *_INTERMDISABLE). Refer to SelectIO IOB Interface Primitives for all supported derivatives.

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

Attributes	Primitives				
	IBUF	OBUF/OBUFT/IOBUF			
		HP I/O Banks		HD I/O Banks	
		Allowed Values	Default	Allowed Values	Default
IOSTANDARD	LVCMOS12	LVCMOS12		LVCMOS12	
DRIVE	N/A	2, 4, 6, 8	12[1]	4, 8, 12	12

Table 22: **Allowed Attributes for the LVCMOS12 I/O Standard** *(cont'd)*

Attributes	Primitives				
	IBUF	OBUF/OBUFT/IOBUF			
		HP I/O Banks		HD I/O Banks	
		Allowed Values	Default	Allowed Values	Default
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 23: **Available I/O Bank Type**

HD	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 240 Ω resistor on the VRP pin. The only valid value of this attribute for LVDCI standards is RDRV_48_48, which corresponds to a 48 Ω setting.

Sample circuits illustrating both unidirectional and bidirectional topologies for a controlled impedance driver are shown in the following figures. The DCI I/O standards supporting a controlled impedance driver are: LVDCI_15 and LVDCI_18.

Figure 47: **Unidirectional Controlled Impedance Driver Topology**

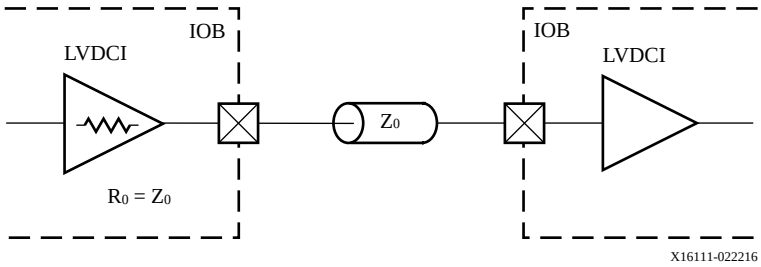
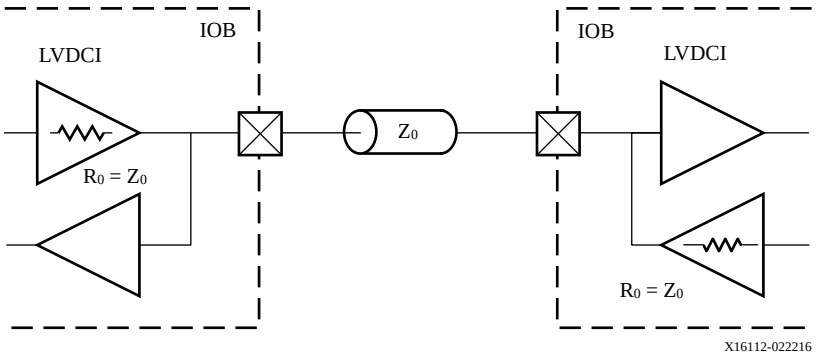


Figure 48: **Bidirectional Controlled Impedance Driver Topology**



The following table 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 the following table (for example: *_DIFF_OUT, *_DCIEN, *_IBUFDISABLE, or *_INTERMDISABLE). Refer to SelectIO IOB Interface Primitives for all supported derivatives.

Table 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 25: **Available I/O Bank Type**

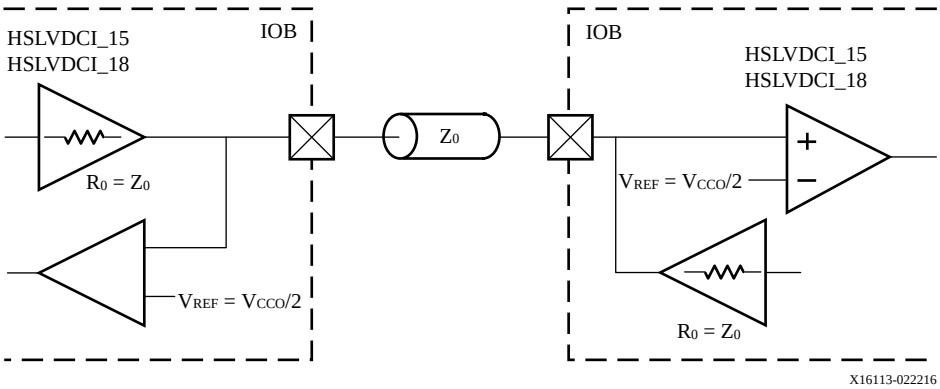
HD	HP
N/A	Available

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 Ω 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 Ω setting.

A sample circuit illustrating bidirectional termination techniques for an HSLVDCI controlled impedance driver is shown in the following figure. The DCI I/O standards supporting a controlled impedance driver with a V_{REF} referenced input are: HSLVDCI_15 and HSLVDCI_18.

Figure 49: **HSLVDCI Controlled Impedance Driver with Bidirectional Termination**



For electrical specifications, see the LVDCI V_{OH} and V_{OL} entries in the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930).

The following table 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 the following table (for example: *_DIFF_OUT, *_DCIEN, *_IBUFDISABLE, or *_INTERMDISABLE). Refer to SelectIO IOB Interface Primitives for all supported derivatives.

Table 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). 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 HD 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 27: **Available I/O Bank Type**

HD	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 40 Ω , 48 Ω, or 60 Ω of driver impedance is available in HP I/O banks only. The default value of the driver output impedance is 48 Ω.

HSTL_I_12

Table 28: **Available I/O Bank Type**

HD	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 40 Ω, 48 Ω, or 60 Ω of driver impedance is available in HP I/O banks. The default value of the driver output impedance is 48 Ω.

HSTL_ I_DCI, HSTL_I_DCI_12, and HSTL_ I_DCI_18

Table 29: **Available I/O Bank Type**

HD	HP
N/A	Available

In Spartan 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 does not provide input split termination in Spartan UltraScale+ devices. SSTL12_DCI provides equivalent functionality.

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

DIFF_HSTL_I and DIFF_HSTL_I_18

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

HD	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 Ω, 48 Ω, or 60 Ω of driver impedance is available in HP I/O banks only. The default value of the driver output impedance is 48 Ω.

DIFF_HSTL_I_DCI and DIFF_HSTL_I_DCI_18

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

HD	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 40 Ω, 48 Ω, or 60 Ω of tuned driver impedance in HP I/O banks. The default value of the driver output impedance is 48 Ω.

DIFF_HSTL_I_12

Available I/O Bank Type

HD	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 40 Ω, 48 Ω, or 60 Ω of driver impedance is available in HP I/O banks. The default value of the driver output impedance is 48 Ω.

DIFF_HSTL_I_12_DCI

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

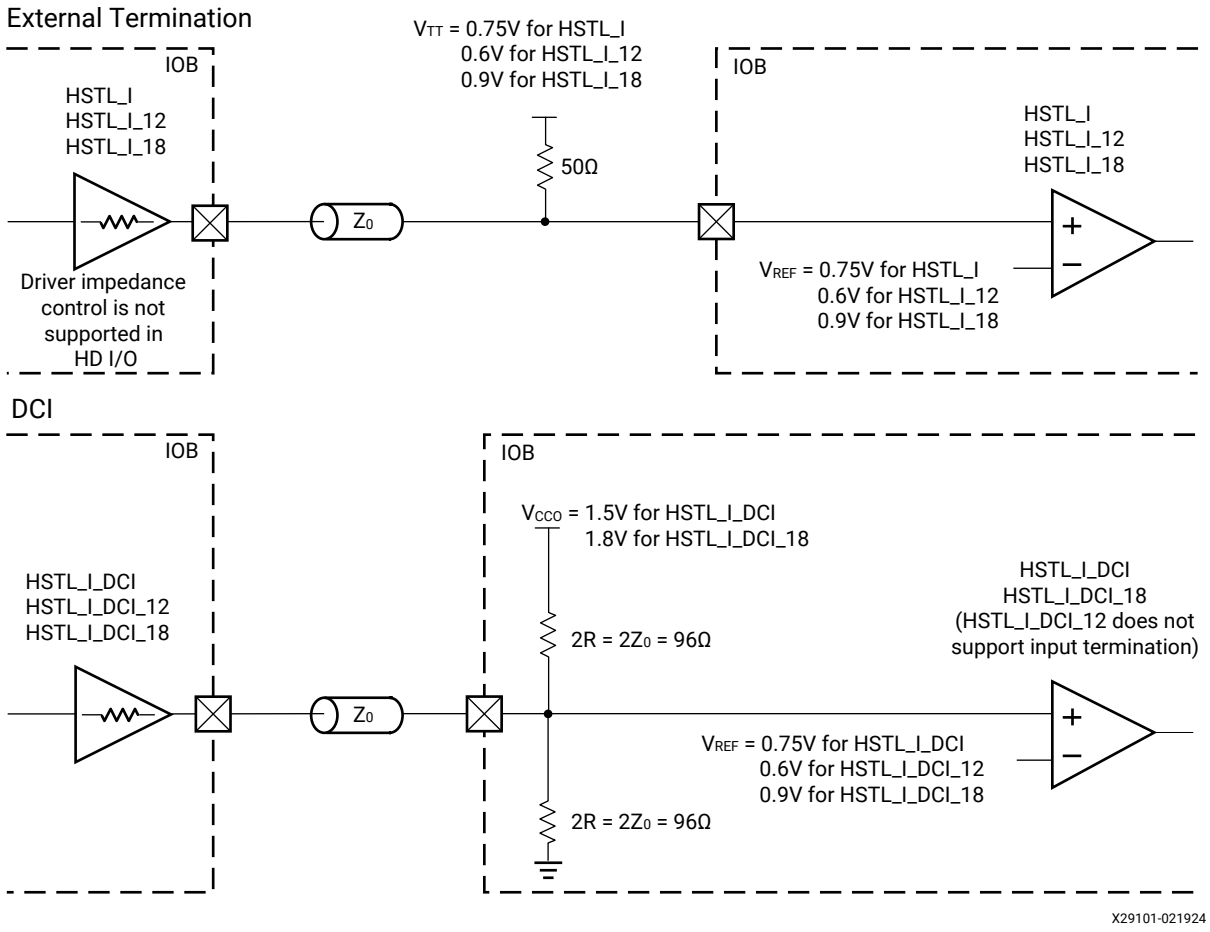
HD	HP
Not Available	Available

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 40 Ω, 48 Ω, or 60 Ω of tuned driver impedance in HP I/O banks. The default value of the driver output impedance is 48 Ω.

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

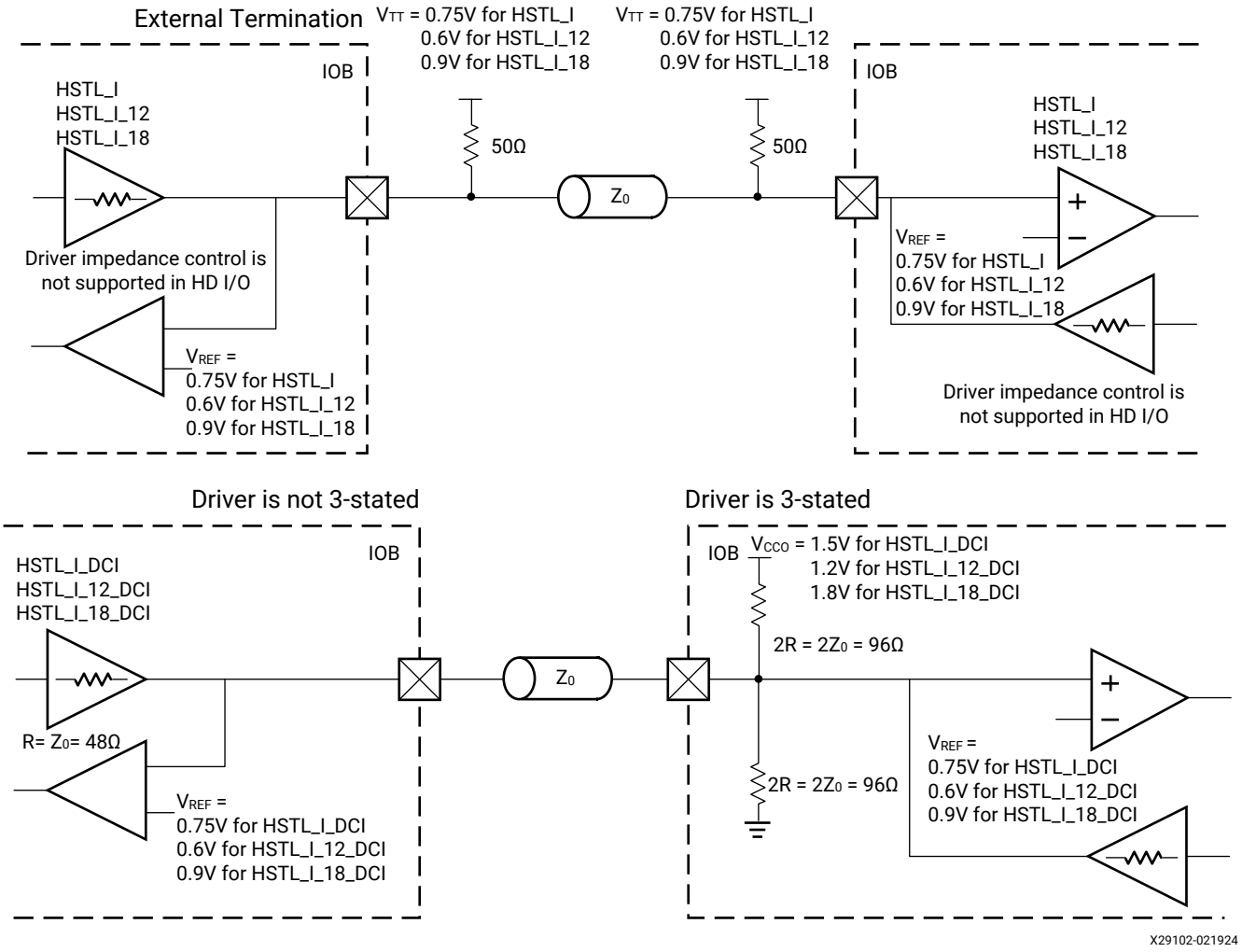
The following figure shows a sample circuit illustrating a termination technique for HSTL class-I for the 1.2V, 1.5V, or 1.8V versions. In a specific circuit, all drivers and receivers must be at the same voltage level (1.2V, 1.5V, or 1.8V); they are not interchangeable (that is, HSTL_I_12 should only interface with HSTL_I_12). Only HP I/O banks support the DCI standards.

Figure 50: **HSTL Class I (1.2V, 1.5V, or 1.8V) Unidirectional Termination**



The following figure 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 (that is, HSTL_I_18 should only interface with HSTL_I_18). Only HP I/O banks support the DCI standards.

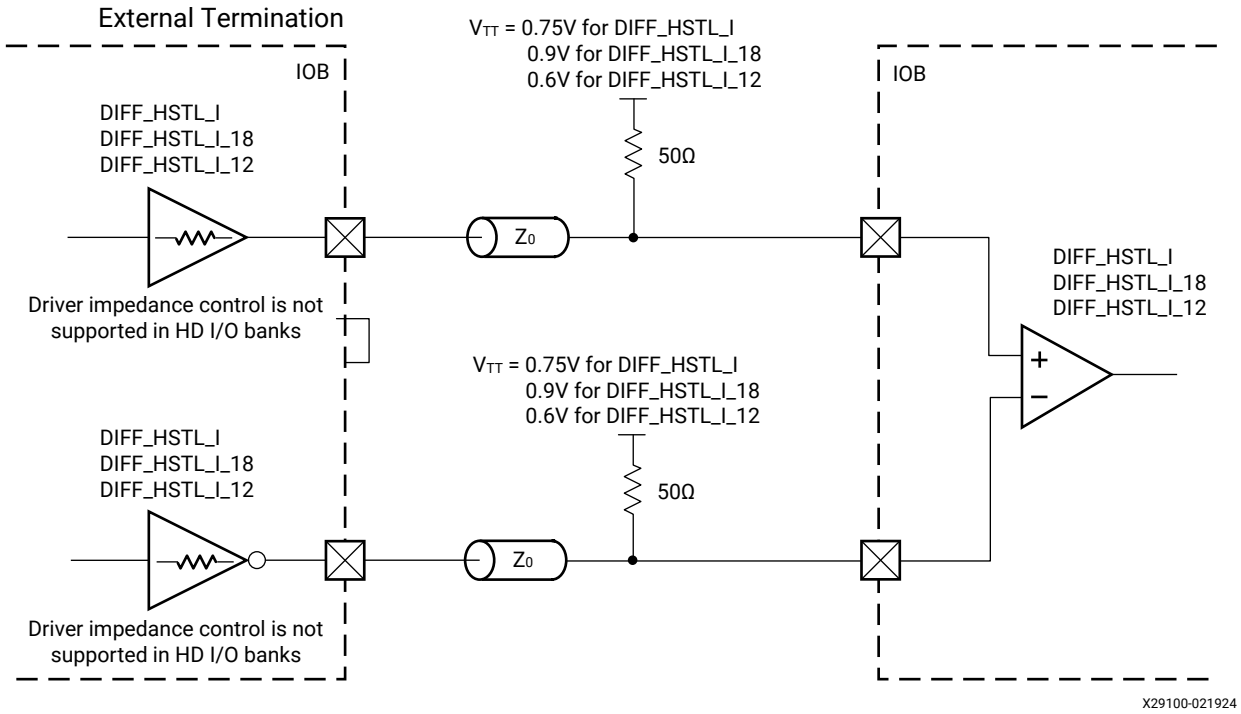
Figure 51: **HSTL Class I (1.2V, 1.5V, or 1.8V) Bidirectional Termination**



Differential HSTL Class I

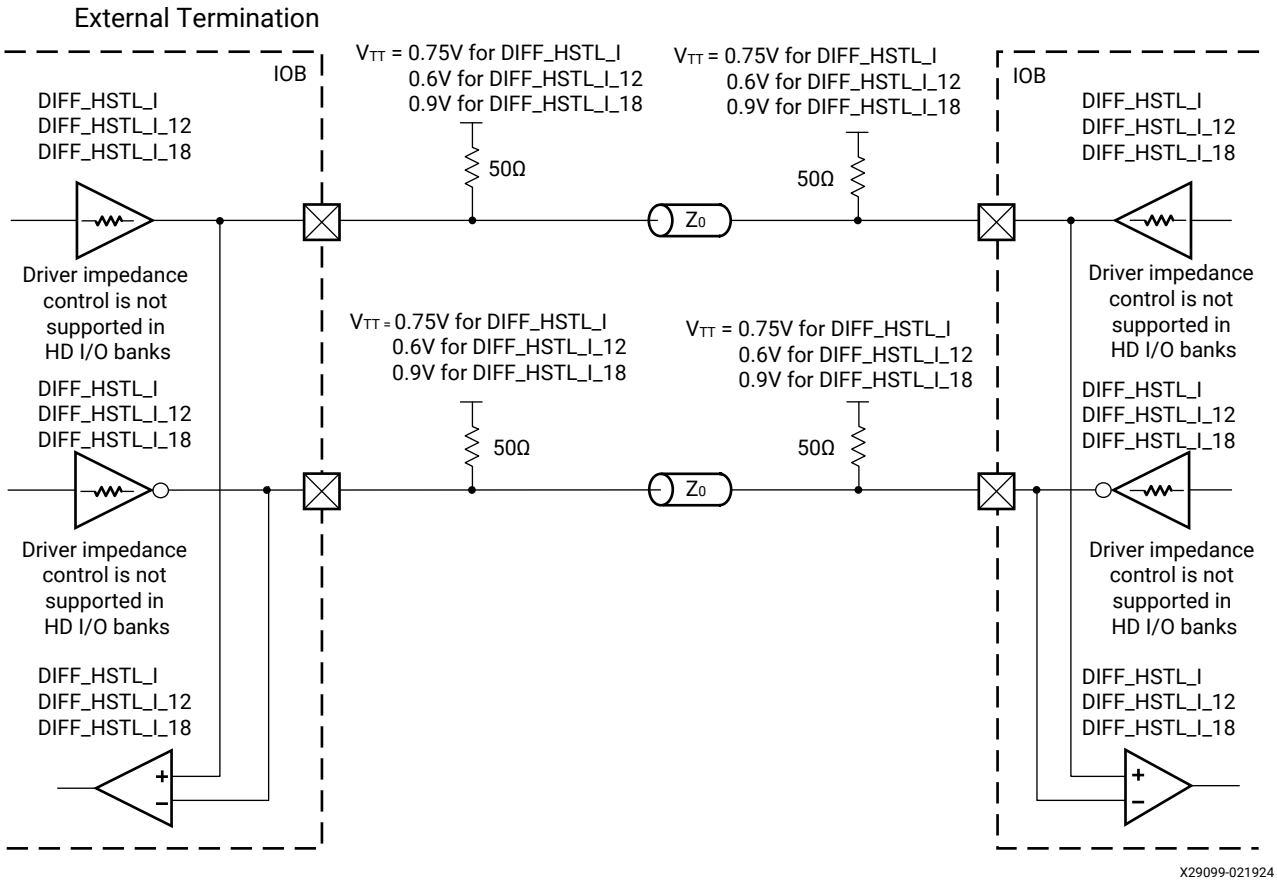
The following figure shows a sample circuit illustrating a termination technique for differential HSTL class-I (1.2V, 1.5V, or 1.8V) with unidirectional termination. In a specific circuit, all drivers and receivers must be at the same voltage level (1.2V, 1.5V, or 1.8V); they are not interchangeable (that is, HDIFF_HSTL_I_18 should only interface with HSTL_I_18).

Figure 52: **Differential HSTL Class I (1.2V, 1.5V, or 1.8V) Unidirectional Termination**



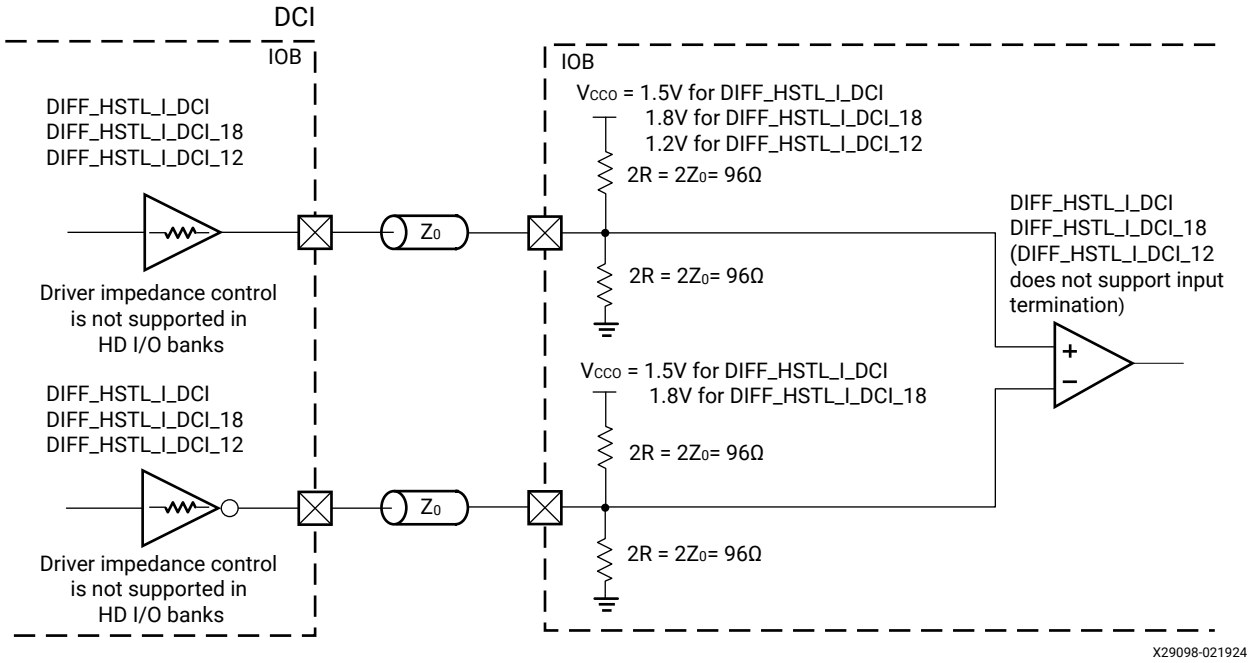
The following figure shows a sample circuit illustrating a termination technique for differential HSTL class-I (1.2V, 1.5V, or 1.8V) with bidirectional termination. In a specific circuit, all drivers and receivers must be at the same voltage level (1.2V, 1.5V, or 1.8V); they are not interchangeable (that is, DIFF_HSTL_I_18 should only interface with DIFF_HSTL_I_18).

Figure 53: **Differential HSTL Class I (1.2V, 1.5V, or 1.8V) Bidirectional Termination**



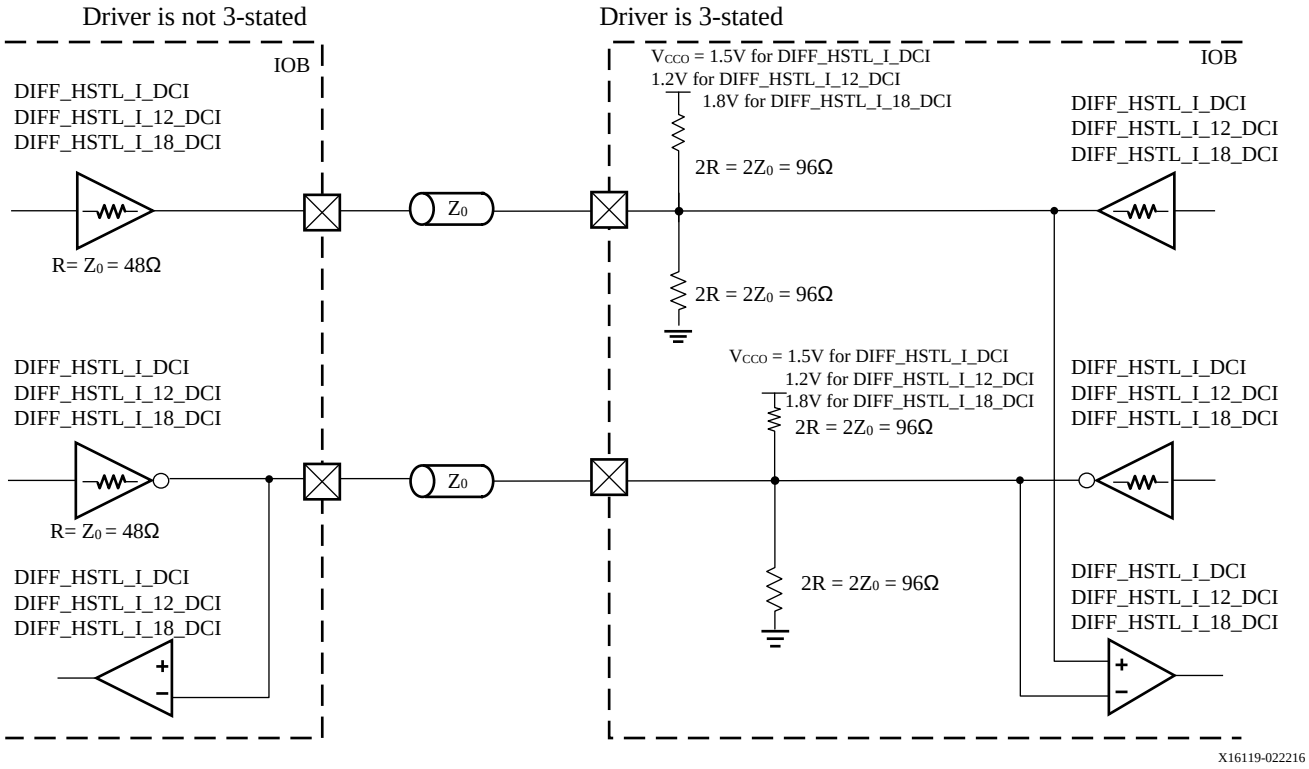
The following figure shows a sample circuit illustrating a termination technique for differential HSTL class-I (1.2V, 1.5V, or 1.8V) with unidirectional DCI termination. In a specific circuit, all drivers and receivers must be at the same voltage level (1.2V, 1.5V, or 1.8V); they are not interchangeable (that is, DIFF_HSTL_I_DCI should only interface with DIFF_HSTL_I_DCI). Only HP I/O banks support these DCI standards.

Figure 54: **Differential HSTL Class I (1.2V, 1.5V, or 1.8V) DCI Unidirectional Termination**



The following figure 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 (that us, DIFF_HSTL_I_DCI should only interface with DIFF_HSTL_I_DCI). Only HP I/O banks support these DCI standards.

Figure 55: **Differential HSTL Class I (1.2V, 1.5V, or 1.8V) DCI Bidirectional Termination**



HSTL Allowed Attributes

The following tables 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 SelectIO IOB Interface Primitives for all supported derivatives.

Table 33: **HSTL Class I Allowed Attributes**

Attributes	IBUF/IBUFE3/IBUFDS/IBUFDSE3				OBUF/OBUFT				IOBUF/IOBUFE3/IOBUFDS/IOBUFDSE3			
	HP I/O		HD I/O		HP I/O		HD I/O		HP I/O		HD 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	N/A	N/A	N/A		N/A		RTT_40 RTT_48 RTT_60 RTT_NONE [1]	RTT_NONE	RTT_48 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 [1]	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	

Table 33: **HSTL Class I Allowed Attributes** *(cont'd)*

Attributes	IBUF/IBUFE3/IBUFDS/IBUFDSE3				OBUF/OBUFT				IOBUF/IOBUFE3/IOBUFDS/IOBUFDSE3			
	HP I/O		HD I/O		HP I/O		HD I/O		HP I/O		HD I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
ODT	RTT_40 RTT_48 RTT_60[2]	RTT_48	N/A		N/A		N/A		RTT_40 RTT_48 RTT_60[1, 2]	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[1]	RDRV_48_48	N/A	
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_48 RTT_NONE	RTT_NONE	N/A		N/A		RTT_40 RTT_48 RTT_60 RTT_NONE[1]	RTT_NONE	RTT_48 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[1]	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	

Table 33: **HSTL Class I Allowed Attributes** *(cont'd)*

Attributes	IBUF/IBUFE3/IBUFDS/IBUFDSE3				OBUF/OBUFT				IOBUF/IOBUFE3/IOBUFDS/IOBUFDSE3			
	HP I/O		HD I/O		HP I/O		HD I/O		HP I/O		HD I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
ODT	RTT_40 RTT_48 RTT_60[2]	RTT_48	N/A		N/A		N/A		RTT_40 RTT_48 RTT_60[1, 2]	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[1]	RDRV_48_48	N/A	

Notes:

1. The allowed bidirectional configuration combinations for driver output impedance (OUTPUT_IMPEDANCE) and ODT are listed in the following table.
2. ODT = RTT_NONE is not a valid setting for DCI I/O standards.

Table 34: **Only Allowed Combinations for Bidirectional Configurations in HP I/O Banks**

OUTPUT_IMPEDANCE	ODT
RDRV_40_40 (40 Ω)	RTT_40
RDRV_40_40 (40 Ω)	RTT_60
RDRV_40_40 (40 Ω)	RTT_NONE
RDRV_48_48 (48 Ω)	RTT_48
RDRV_48_48 (48 Ω)	RTT_NONE
RDRV_60_60 (60 Ω)	RTT_40
RDRV_60_60 (60 Ω)	RTT_60
RDRV_60_60 (60 Ω)	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, 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 HD I/O banks. Both HP and HD 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 — HP I/O only) provides the option of 40 Ω, 48 Ω, or 60 Ω 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 Ω (48 Ω for HD I/O). The optimal drive and termination scheme for any new design is determined through careful signal-integrity analysis. SSTL18 class-II is only available as an input in HD I/O banks. HD I/O banks provide the option of ODT attributes for untuned internal parallel split-termination resistors for the standard.

IMPORTANT! *HD I/O banks only support class-II drivers. SSTL125_II, SSTL15_II, SSTL18_II, DIFF_SSTL15_II, DIFF_SSTL135_II, and DIFF_SSTL18_II are all only supported as input buffers.*

SSTL15 is used for DDR3 SDRAM interfaces and is roughly defined (not by name) in the JEDEC standard JESD79-3E. For this standard, the full-strength driver (SSTL15) is available in both the HP and HD 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 HD 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 Ω , 48 Ω , or 60 Ω 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 Ω (48 Ω for HD I/O banks). 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. For this standard, the full-strength driver (SSTL135) is available in both the HP and HD 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 HD 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 40 Ω, 48 Ω, or 60 Ω 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 Ω (48 Ω for HD I/O banks). The optimal drive and termination scheme for any new design is determined through careful signal-integrity analysis.

SSTL12 supports Micron's next-generation RLDRAM3 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. 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 40 Ω, 48 Ω, or 60 Ω 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 Ω. HD I/O banks only support SSTL12 as an input. The optimal drive and termination scheme for any new design is determined through careful signal-integrity analysis.

Note: HD I/O banks only support SSTL12 as an input. Neither output or bidirectional interfaces support SSTL12 in HD I/O banks.

SSTL18_I, DIFF_SSTL18_I

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

HD	HP
Available	Available

Class-I drivers can be preferred for short, point-to-point board topologies. Parallel end-termination resistors (commonly 50 Ω) to V_{TT} = (V_{CCO}/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_{CCO}/2. The untuned on-die source termination feature (OUTPUT_IMPEDANCE) provides the option of 40 Ω, 48 Ω, or 60 Ω of driver impedance in HP I/O banks. The driver output impedance is set to a default of 40 Ω (48 Ω for HD I/O). The differential (DIFF_) version uses complementary single-ended drivers for outputs, and differential receivers for inputs.

SSTL18_I_DCI, DIFF_SSTL18_I_DCI

Table 36: **Available I/O Bank Type**

HD	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 $V_{CCO}/2$ mid-point level. The source termination feature (OUTPUT_IMPEDANCE) provides the option of 40 Ω, 48 Ω, or 60 Ω tuned driver impedance in HP I/O banks. The driver output impedance is set to a default of 40 Ω. The differential (DIFF_) version uses complementary single-ended drivers for outputs, and differential receivers for inputs.

SSTL15, SSTL135, SSTL12, DIFF_SSTL15, DIFF_SSTL135, DIFF_SSTL12

Table 37: **Available I/O Bank Type**

HD	HP
Available	Available

Parallel end-termination resistors (commonly 50 Ω) 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 40 Ω, 48 Ω, or 60 Ω of driver impedance in HP I/O banks. The driver output impedance is set to a default of 40 Ω (48 Ω or HD I/O). 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 38: **Available I/O Bank Type**

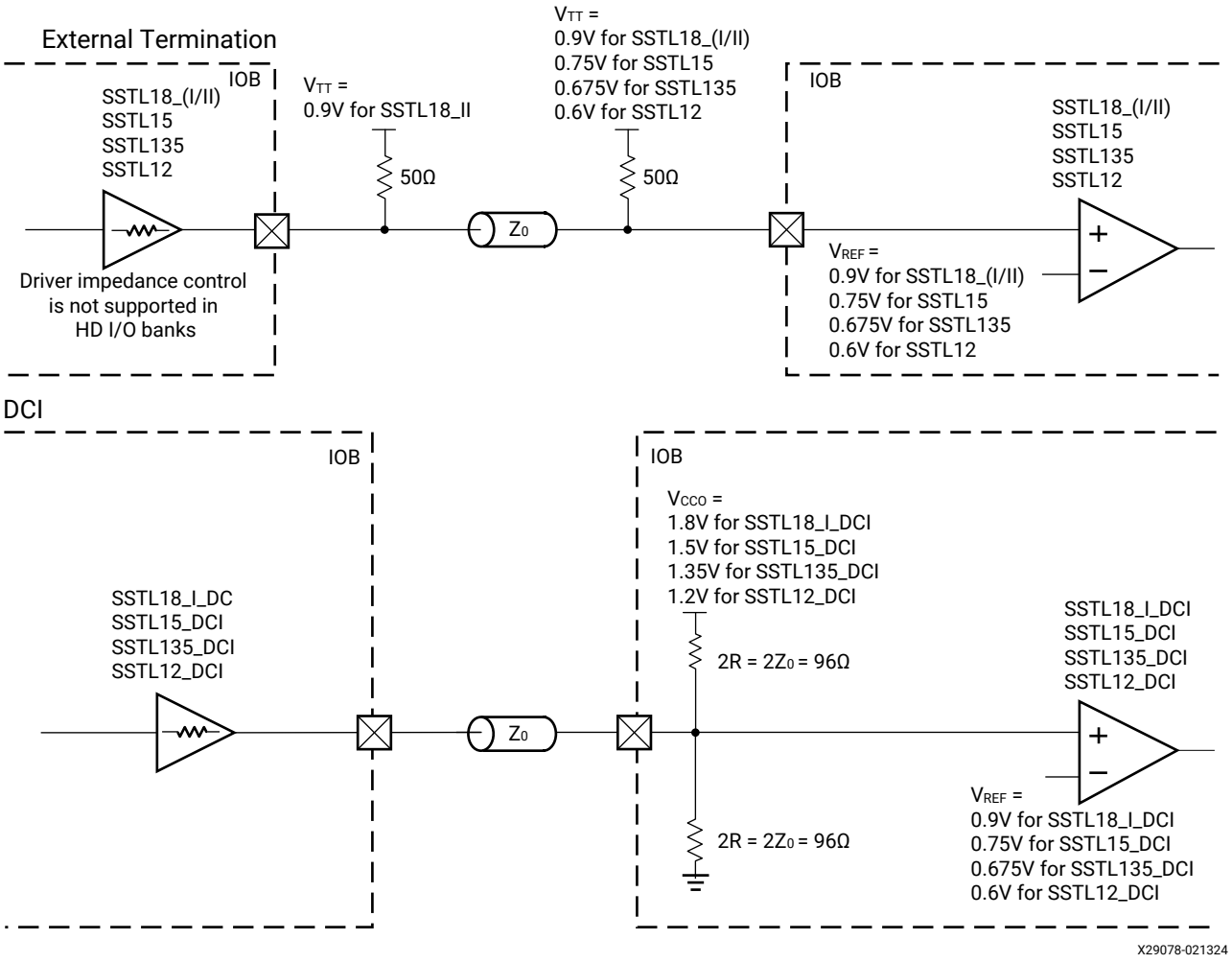
HD	HP
N/A	Available

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 $V_{CCO}/2$ mid-point level. The source termination feature (OUTPUT_IMPEDANCE) provides the option of 40 Ω, 48 Ω, or 60 Ω tuned driver impedance in HP I/O banks. The driver output impedance is set to a default of 40 Ω. The differential (DIFF_) versions use complementary single-ended drivers for outputs, and differential receivers for inputs.

SSTL18, SSTL15, SSTL135, SSTL12

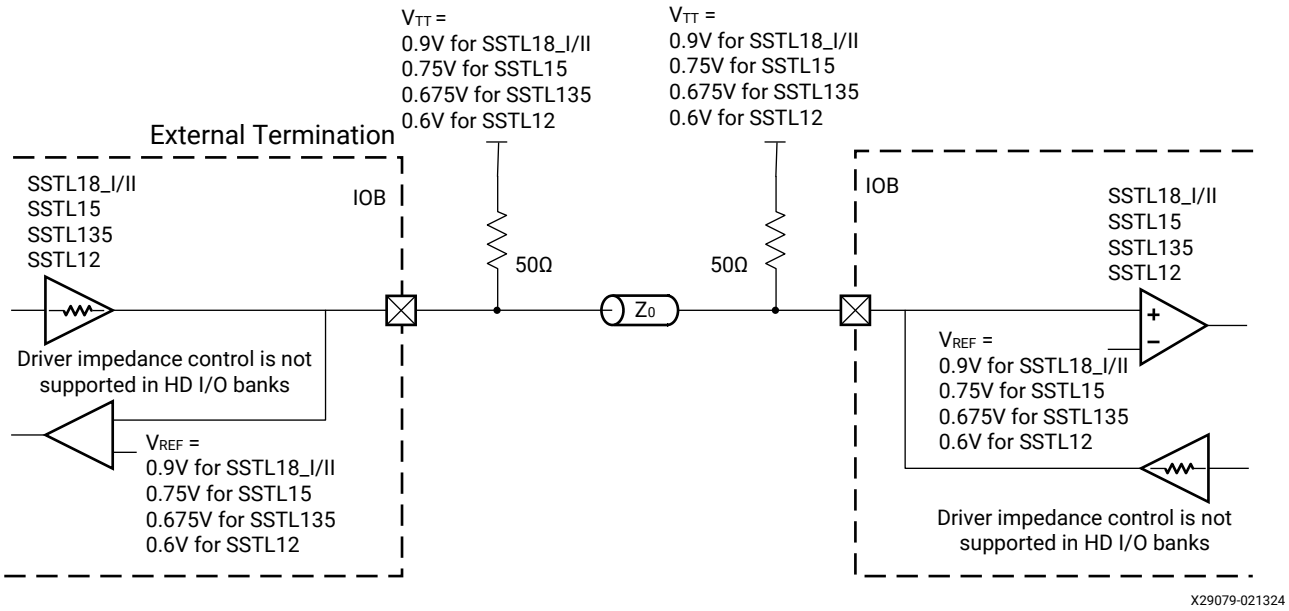
The following figure shows a sample circuit illustrating a unidirectional termination technique for SSTL18, SSTL15, SSTL135, or SSTL12. In a specific circuit, all drivers and receivers must be at the same voltage level (1.8V, 1.5V, 1.35V, or 1.2V); they are not interchangeable (that is, SSTL12 should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

Figure 56: **SSTL18, SSTL15, SSTL135, or SSTL12 Unidirectional Termination**



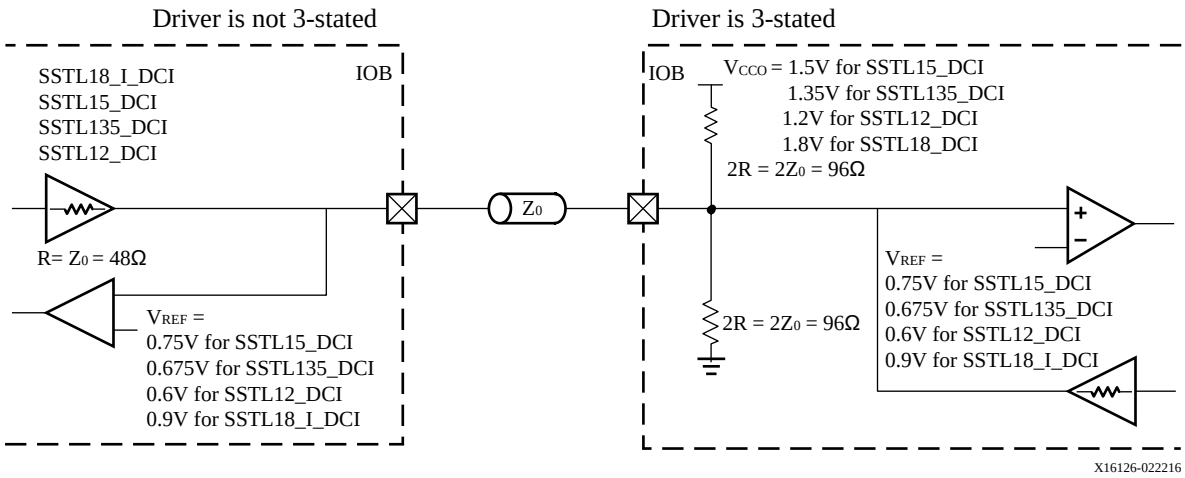
The following figure shows a sample circuit illustrating a bidirectional termination technique for SSTL18, SSTL15, SSTL135, or SSTL12. In a specific circuit, all drivers and receivers must be at the same voltage level (1.8V, 1.5V, 1.35V, or 1.2V); they are not interchangeable (that is, SSTL12 should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

Figure 57: **SSTL18, SSTL15, SSTL135, or SSTL12 Bidirectional Termination**



The following figure 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 (that is, SSTL12_DCI should only interface with SSTL2_DCI). DCI standards are only supported in HP I/O banks.

Figure 58: **SSTL18_DCI, SSTL15_DCI, SSTL135_DCI, or SSTL12_DCI Bidirectional Termination**

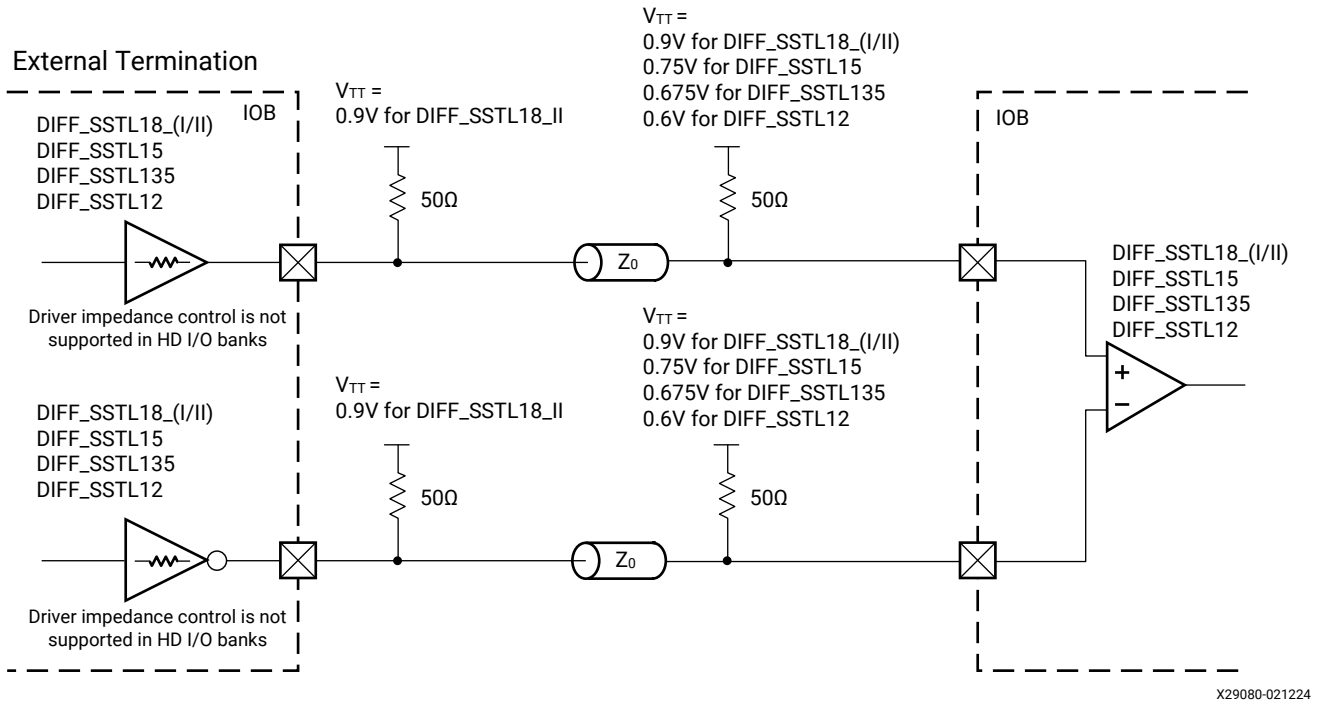


Differential SSTL18, SSTL15, SSTL135, SSTL12

The following figure 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 (that is, DIFF_SSTL12 should only interface with DIFF_SSTL12).

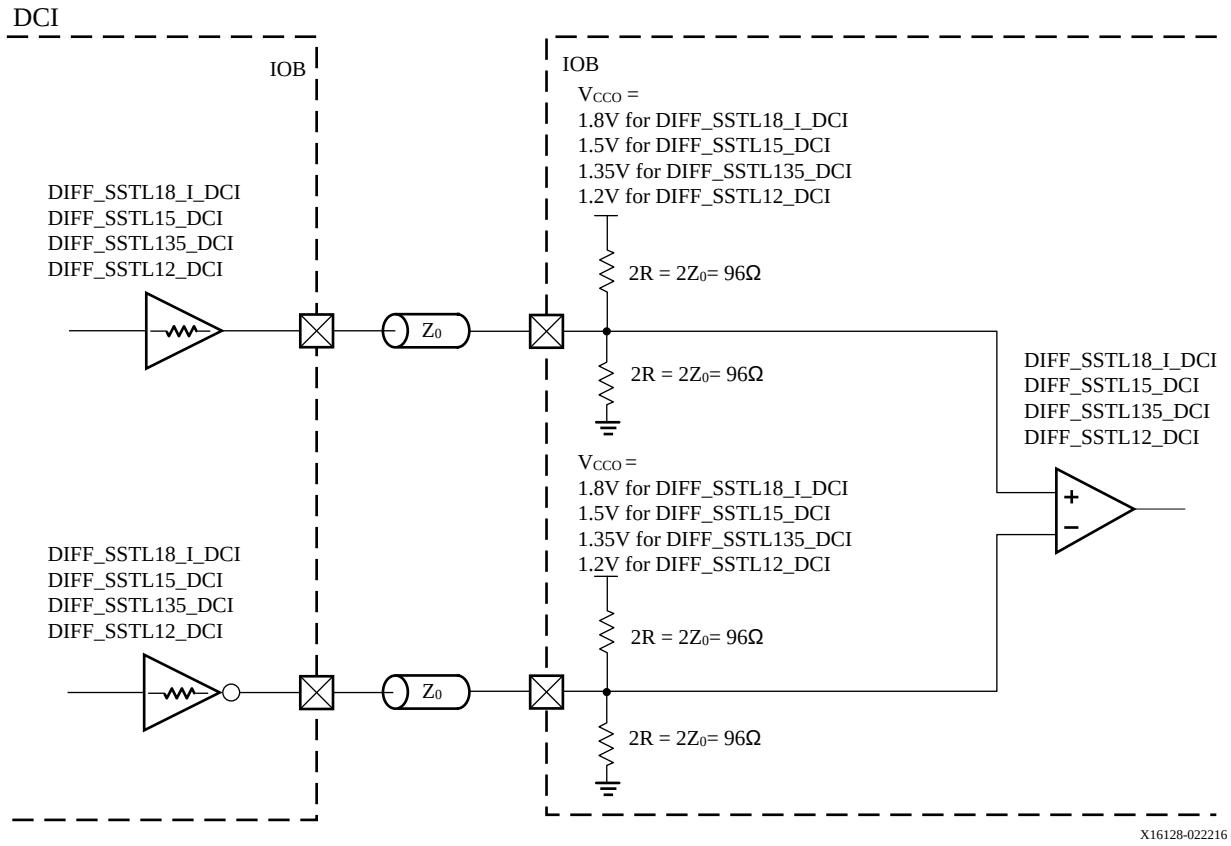
***Note*:** HD I/Os only support DIFF_SSTL12 in receive mode. Output and bidirectional buffers do not support DIFF_SSTL12 in HD I/O banks.

Figure 59: **Differential SSTL18, SSTL15, SSTL135, or SSTL12 Unidirectional Termination**



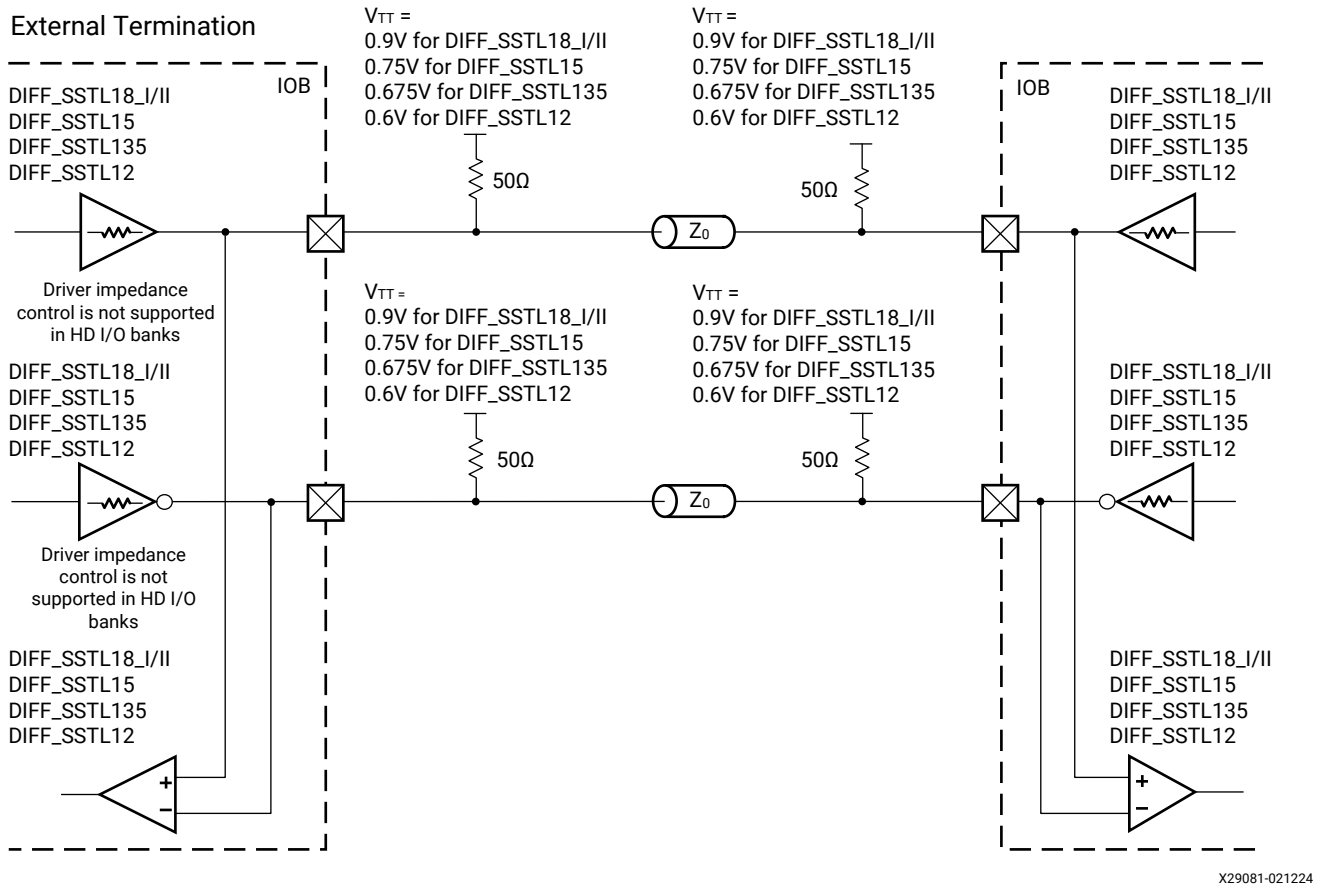
The following figure shows a sample circuit illustrating a termination technique for differential SSTL18, SSTL15, SSTL135, or SSTL12 with unidirectional DCI termination. In a specific circuit, all drivers and receivers must be at the same voltage level (1.8V, 1.5V, 1.35V, or 1.2V); they are not interchangeable (that is, DIFF_SSTL12_DCI should only interface with DIFF_SSTL12_DCI).

Figure 60: **Differential SSTL18, SSTL15, SSTL135, or SSTL12 Unidirectional DCI Termination**



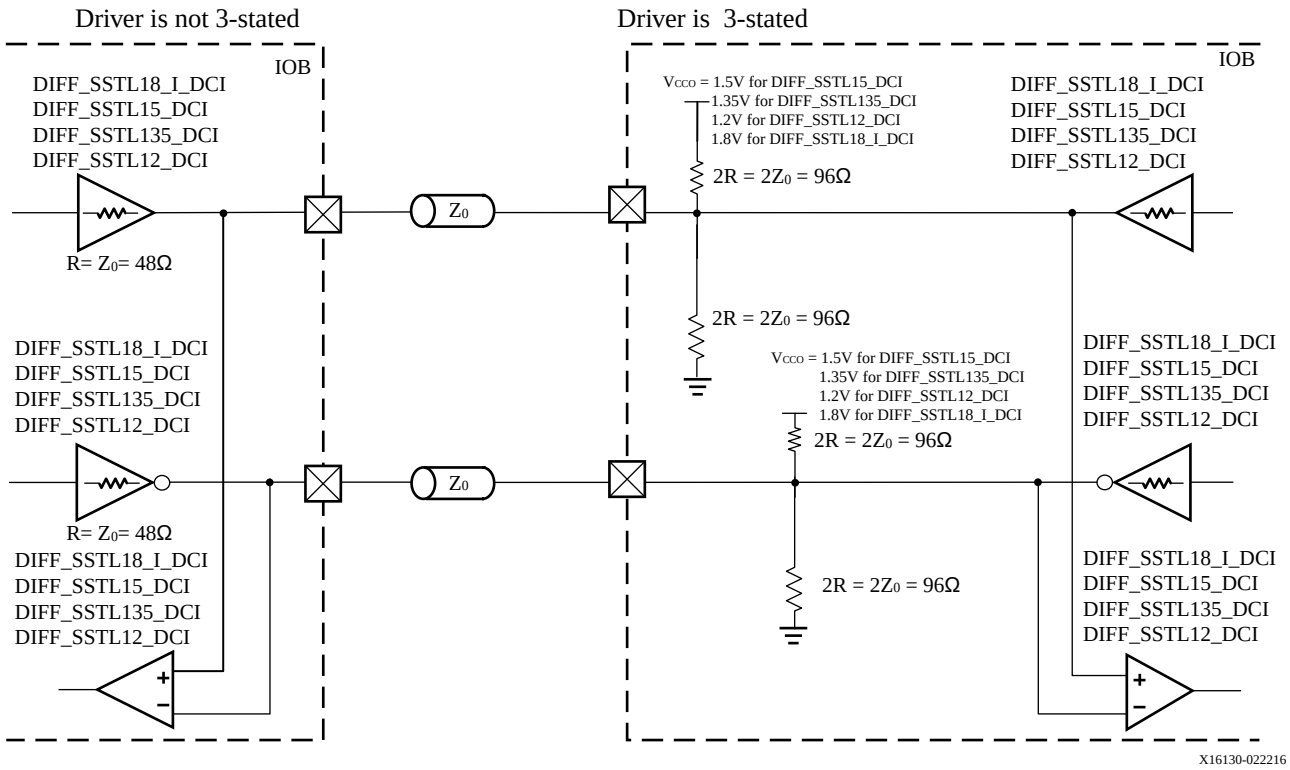
The following figure shows a sample circuit illustrating a termination technique for differential SSTL18, SSTL15, SSTL135, or SSTL12 with bidirectional termination. In a specific circuit, all drivers and receivers must be at the same voltage level (1.8V, 1.5V, 1.35V, or 1.2V); they are not interchangeable (that is, DIFF_SSTL12 should only interface with DIFF_SSTL12).

Figure 61: **Differential SSTL18, SSTL15, SSTL135, or SSTL12 with Bidirectional Termination**



The following figure 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 (that is, DIFF_SSTL12_DCI should only interface with DIFF_SSTL12_DCI). DCI standards are supported only in HP I/Os.

Figure 62: **Differential SSTL18, SSTL15, SSTL135, or SSTL12 with Bidirectional DCI Termination**



The following table lists the allowed attributes for SSTL I/O standards. Support is implied for primitives that are derivatives of the primitives listed in the following table (for example: *_DIFF_OUT, *_DCIEN, *_IBUFDISABLE, or *_INTERMDISABLE). Refer to SelectIO IOB Interface Primitives for all supported derivatives.

Table 39: **SSTL Allowed Attributes**

Attributes	IBUF/IBUFE3/IBUFDS/IBUFDSE3				OBUF/OBUFT				IOBUF/IOBUFE3/IOBUFDS/IOBUFDSE3			
	HP I/O		HD I/O		HP I/O		HD I/O		HP I/O		HD 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 SSTL15 SSTL18_I SSTL135_II SSTL18_II SSTL15_II		SSTL12 SSTL135 SSTL15 SSTL18_I		SSTL135 SSTL15 SSTL18_I		SSTL12 SSTL135 SSTL15 SSTL18_I		SSTL135 SSTL15 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_48 RTT_NONE	RTT_NONE	N/A		N/A		RTT_40 RTT_48 RTT_60 RTT_NONE[1]	RTT_NONE	RTT_48 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[1]	RDRV_40_40	N/A	
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	

Table 39: **SSTL Allowed Attributes** *(cont'd)*

Attributes	IBUF/IBUFE3/IBUFDS/IBUFDSE3				OBUF/OBUFT				IOBUF/IOBUFE3/IOBUFDS/IOBUFDSE3			
	HP I/O		HD I/O		HP I/O		HD I/O		HP I/O		HD I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
ODT	RTT_40 RTT_48 RTT_60	RTT_40	N/A		N/A		N/A		RTT_40 RTT_48 RTT_60[1, 3]	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[1]	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_SSTL135_II DIFF_SSTL18_II DIFF_SSTL15_II		DIFF_SSTL12 DIFF_SSTL135 DIFF_SSTL15 DIFF_SSTL18_I		DIFF_SSTL135 DIFF_SSTL15 DIFF_SSTL18_I		DIFF_SSTL12 DIFF_SSTL135 DIFF_SSTL15 DIFF_SSTL18_I		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[2]	TRUE FALSE	FALSE	N/A	N/A	N/A		N/A		TRUE FALSE	FALSE	TRUE FALSE	FALSE
ODT	RTT_40 RTT_48 RTT_60 RTT_NONE	RTT_NONE	RTT_48 RTT_NONE	RTT_NONE	N/A		N/A		RTT_40 RTT_48 RTT_60 RTT_NONE[1]	RTT_NONE	RTT_48 RTT_NONE	RTT_NONE

Table 39: **SSTL Allowed Attributes** *(cont'd)*

Attributes	IBUF/IBUFE3/IBUFDS/IBUFDSE3				OBUF/OBUFT				IOBUF/IOBUFE3/IOBUFDS/IOBUFDSE3			
	HP I/O		HD I/O		HP I/O		HD I/O		HP I/O		HD 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_40_40	N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60[1]	RDRV_40_40	N/A	
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[2, 4]	TRUE FALSE	FALSE	N/A		N/A		N/A		TRUE FALSE	FALSE	N/A	
ODT	RTT_40 RTT_48 RTT_60[3]	RTT_40	N/A		N/A		N/A		RTT_40 RTT_48 RTT_60[1, 3]	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[1]	RDRV_40_40	N/A	

Notes:

1. The allowed bidirectional configuration combinations for driver output impedance (OUTPUT_IMPEDANCE) and ODT are listed in Table 34.
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.

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. Spartan 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 40: **Available I/O Bank Type**

HD	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_{CCO}. The untuned on-die source termination feature (OUTPUT_IMPEDANCE) provides the option of 40 Ω, 48 Ω, or 60 Ω of driver impedance in HP I/O banks. The driver output impedance is set to a default of 48 Ω.

HSUL_DCI_12 and DIFF_HSUL_12_DCI

Table 41: **Available I/O Bank Type**

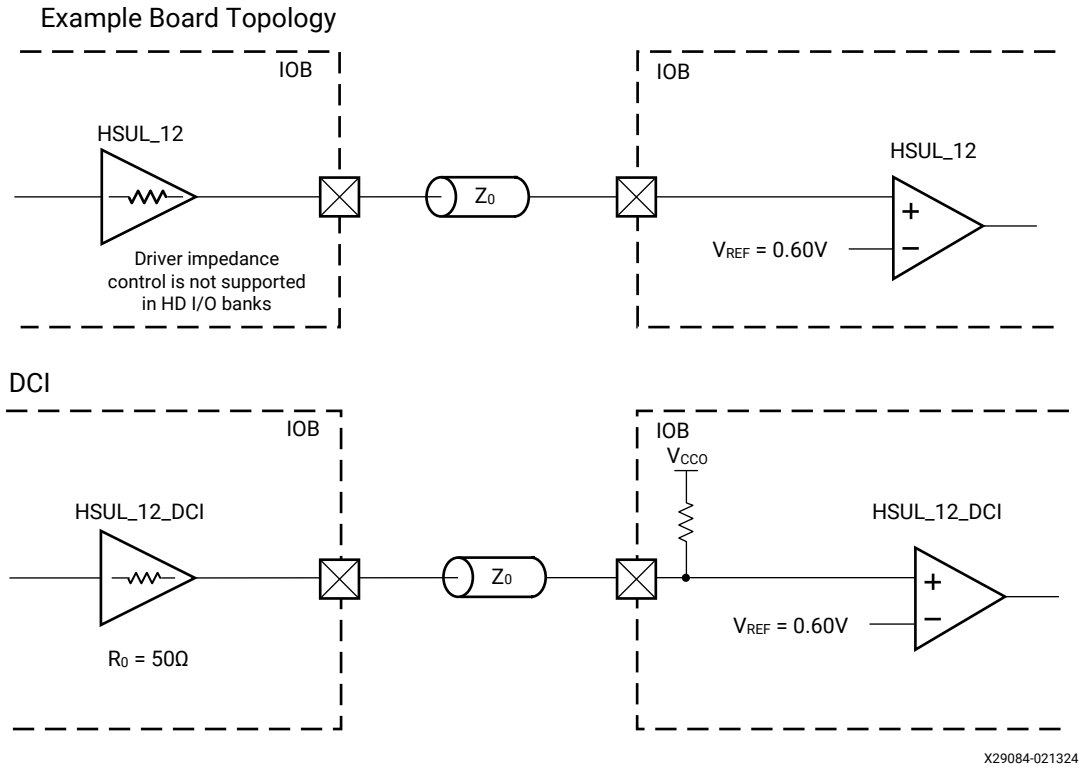
HD	HP
N/A	Available

DCI provides a tuned on-die input single termination to V_{CCO} on the receivers and a tuned on-die source termination option (OUTPUT_IMPEDANCE) of 40 Ω, 48 Ω, or 60 Ω 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 48 Ω.The differential (DIFF_) versions use complementary single-ended drivers for outputs, and differential receivers for inputs.

HSUL_12

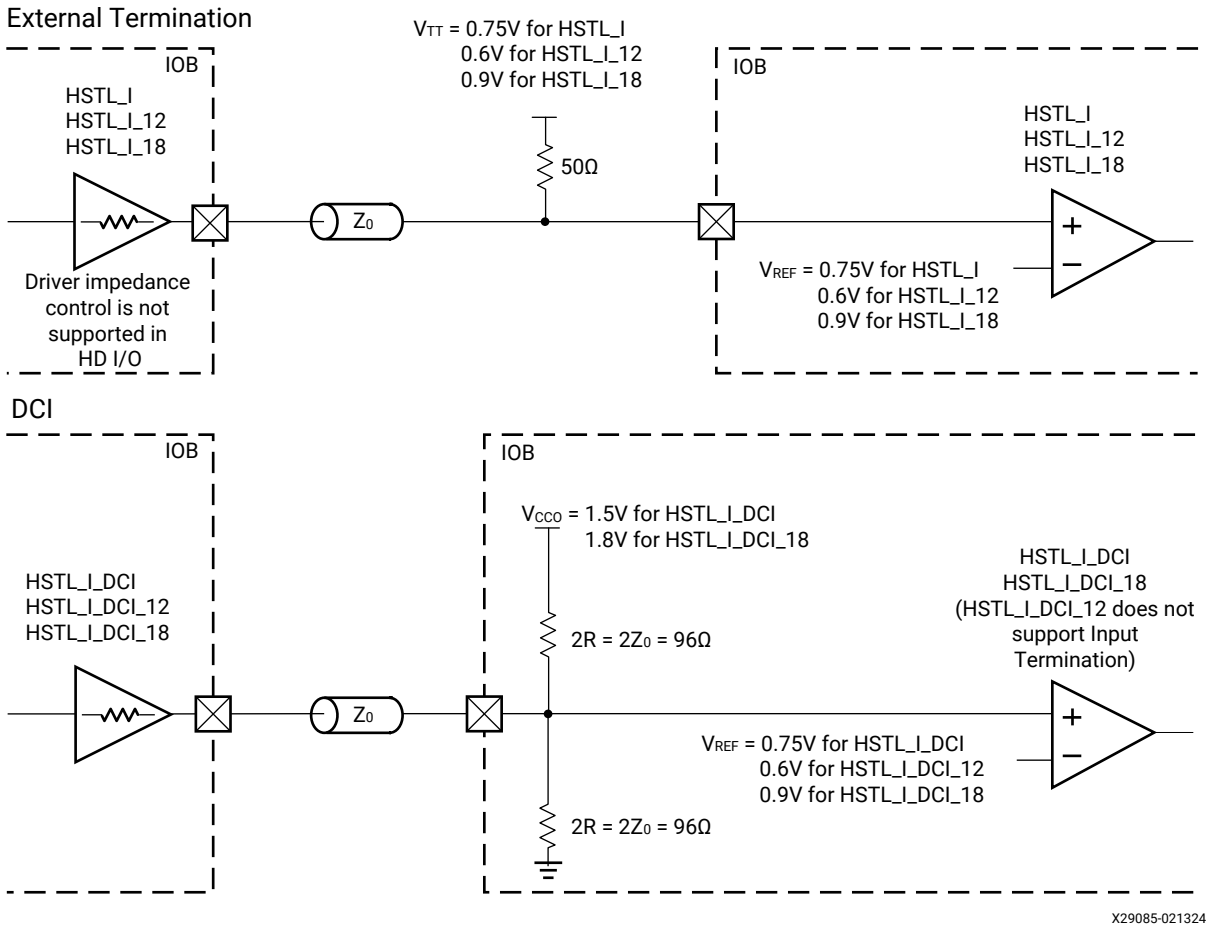
The following figure shows a sample circuit illustrating a unidirectional board topology for HSUL_12. Only HP I/O banks support the DCI version.

Figure 63: **HSUL_12 with Unidirectional Signaling**



The following figure shows a sample circuit illustrating a bidirectional board topology (with no termination) for HSUL_12. Only HP I/O banks support the DCI version.

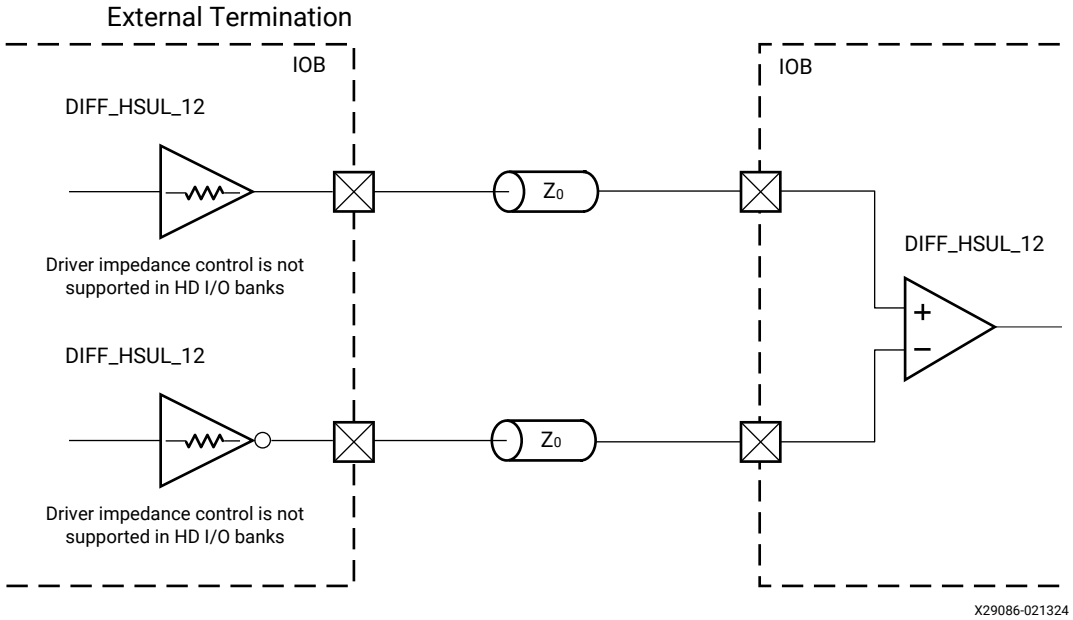
Figure 64: **HSUL_12 with Bidirectional Signaling**



Differential HSUL_12

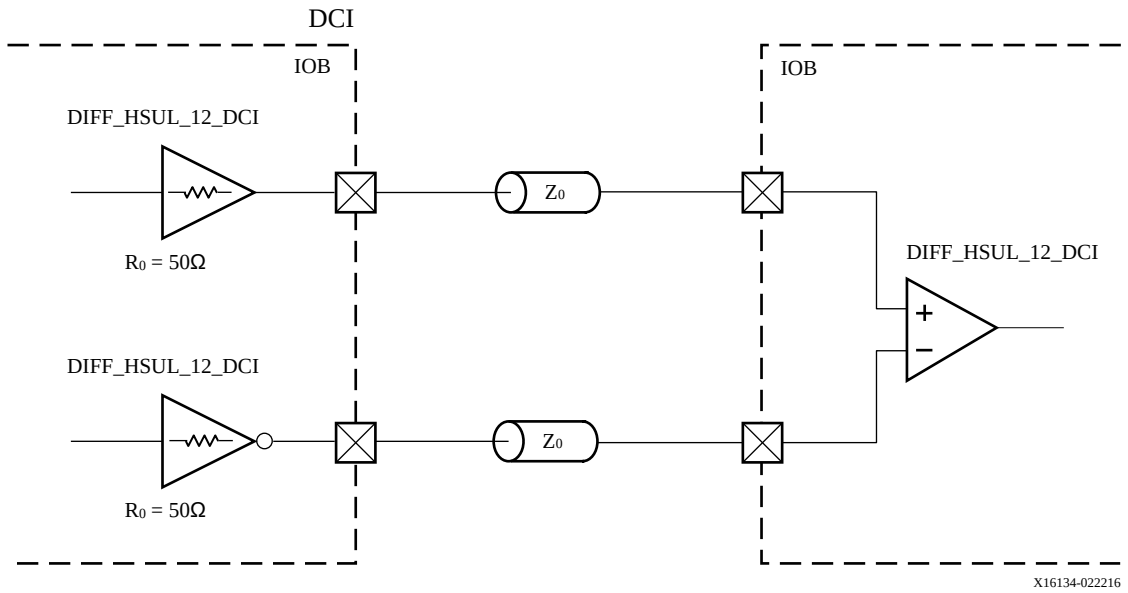
The following figure shows a sample circuit illustrating a board topology for differential HSUL_12 with unidirectional signaling.

Figure 65: **Differential HSUL_12 with Unidirectional Signaling**



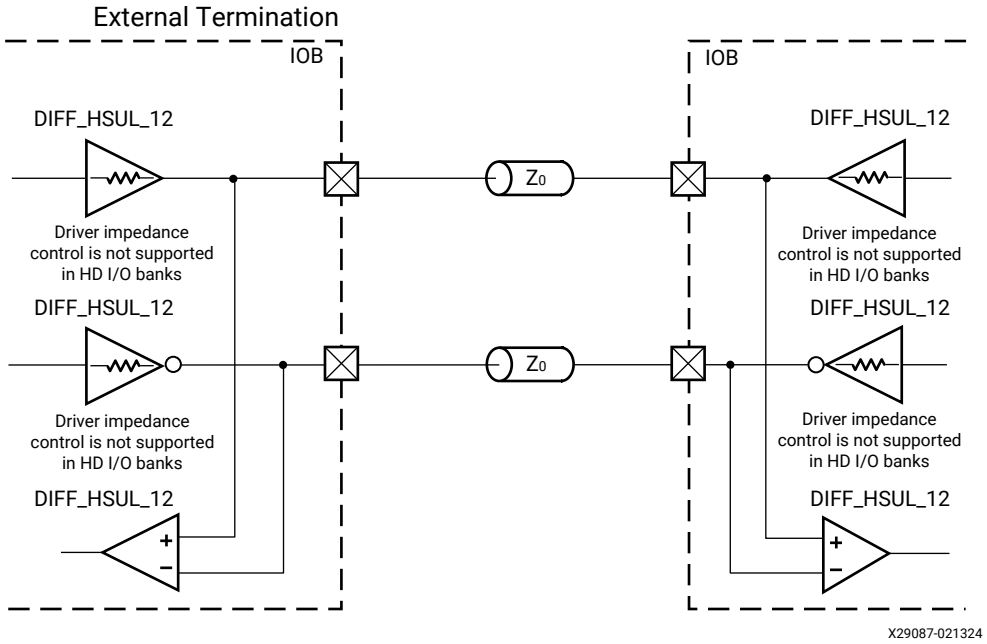
The following figure shows a sample circuit illustrating a board topology for differential HSUL_12 with unidirectional DCI signaling.

Figure 66: **Differential HSUL_12 with Unidirectional DCI Signaling**



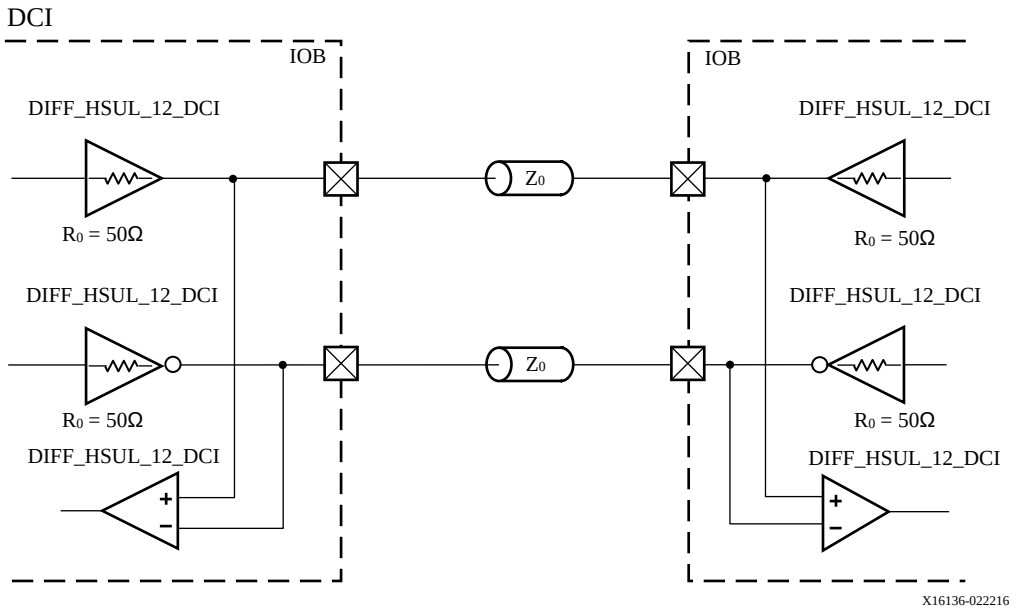
The following figure shows a sample circuit illustrating a board topology for differential HSUL_12 with bidirectional signaling.

Figure 67: **Differential HSUL_12 with Bidirectional Signaling**



The following figure shows a sample circuit illustrating a board topology for differential HSUL_12 with bidirectional DCI signaling.

Figure 68: **Differential HSUL_12 with DCI Bidirectional Signaling**



The following table lists the allowed attributes for HSUL I/O standards. Support is implied for primitives that are derivatives of the primitives listed in the following table (for example: *_DIFF_OUT, *_DCIEN, *_IBUFDISABLE, or *_INTERMDISABLE). Refer to SelectIO IOB Interface Primitives for all supported derivatives.

Table 42: **HSUL Allowed Attributes**

Attributes	IBUF/IBUFE3/IBUFDS/IBUFDSE3				OBUF/OBUFT				IOBUF/IOBUFE3/IOBUFDS/IOBUFDSE3			
	HP I/O		HD I/O		HP I/O		HD I/O		HP I/O		HD 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		N/A		HSUL_12 DIFF_HSUL_12		N/A	

Table 42: **HSUL Allowed Attributes** *(cont'd)*

Attributes	IBUF/IBUFE3/IBUFDS/IBUFDSE3				OBUF/OBUFT				IOBUF/IOBUFE3/IOBUFDS/IOBUFDSE3			
	HP I/O		HD I/O		HP I/O		HD I/O		HP I/O		HD I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
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[1]	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	
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 43: **Available I/O Bank Type**

HD	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_{CCO}. The untuned on-die source termination feature (OUTPUT_IMPEDANCE) provides the option of 40 Ω, 48 Ω, or 60 Ω of driver impedance in HP I/O banks. The driver output impedance is set to a default of 40 Ω. 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 44: **Available I/O Bank Type**

HD	HP
N/A	Available

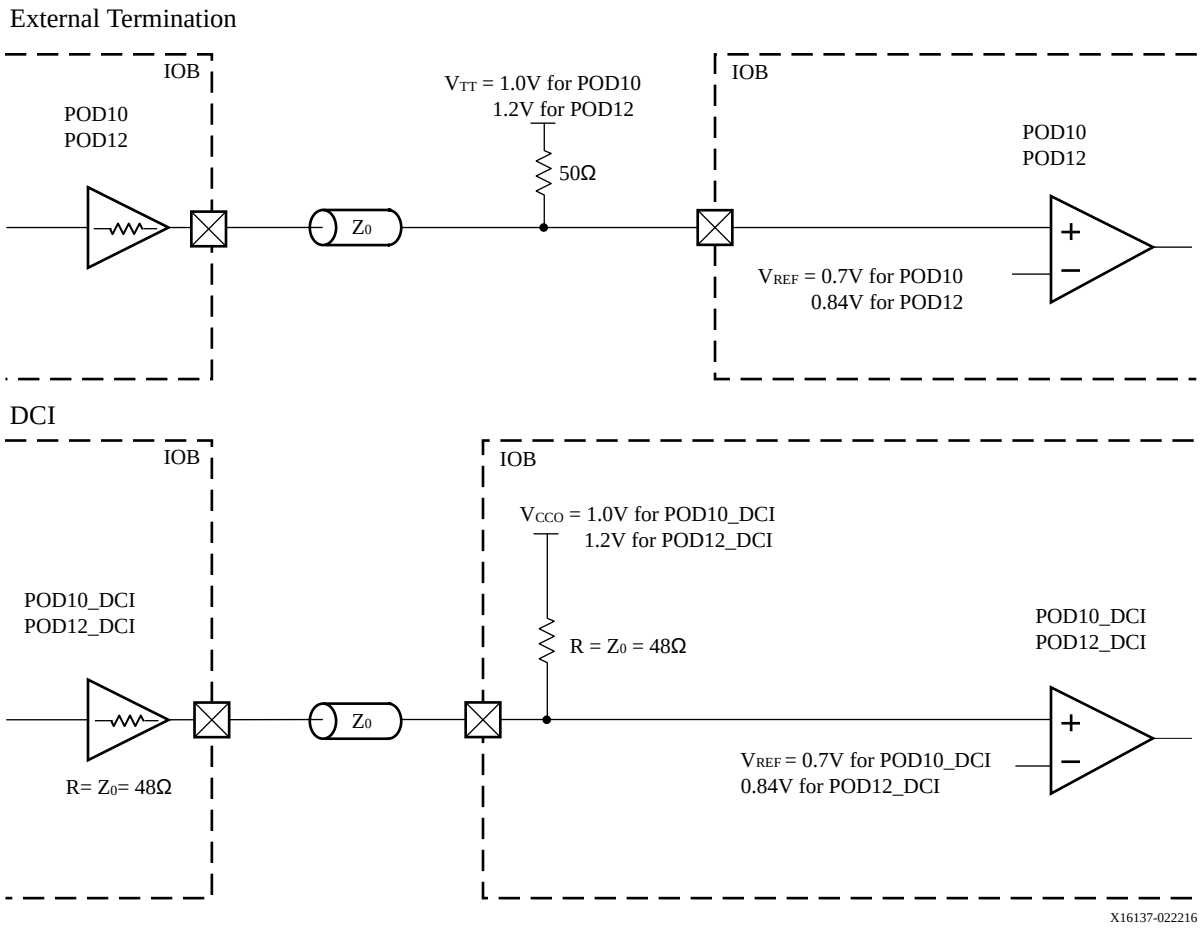
DCI provides a tuned single termination to V_{CCO} at the receiver that matches the ODT attribute setting. The differential (DIFF_) versions use complementary single-ended drivers for outputs, and differential receivers for inputs.

DCI provides a tuned single ODT pull-up to V_{CCO} in the receivers and a source termination option (OUTPUT_IMPEDANCE) of 40 Ω , 48 Ω, or 60 Ω in the driver. The driver output impedance is set to a default of 40 Ω. POD12 standards also have optional EQUALIZATION and OFFSET_CNTRL features in the receivers and PRE_EMPHASIS in the drivers.

POD

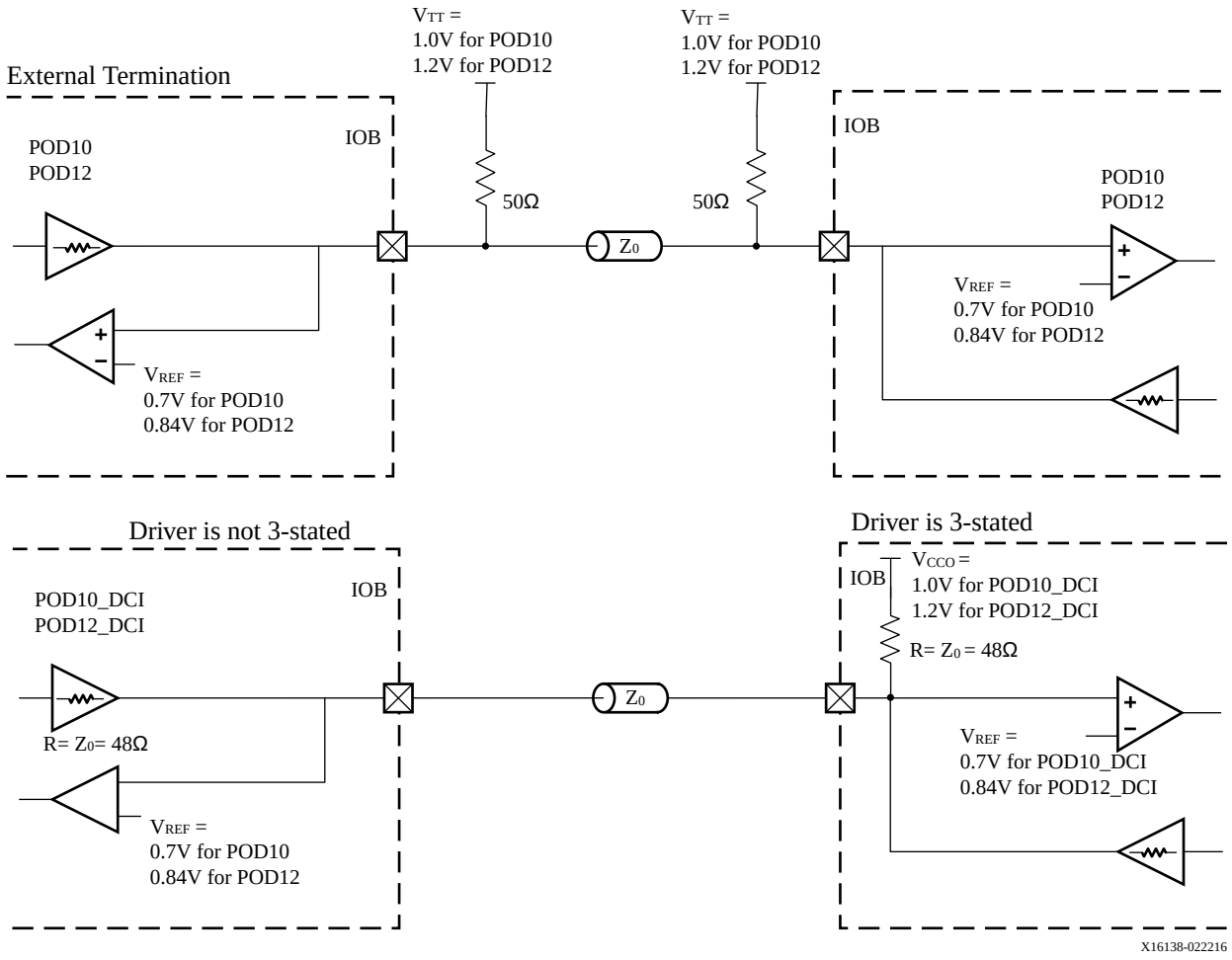
The following figure shows a sample circuit illustrating a unidirectional board topology for POD (1.0V or 1.2V) with matched driver and receiver termination values. Only HP I/O banks support these standards.

Figure 69: **POD with Unidirectional Signaling**



The following figure shows a sample circuit illustrating a termination technique for POD (1.0V or 1.2V) with bidirectional termination and 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 (that is, POD12 should only interface with POD12).

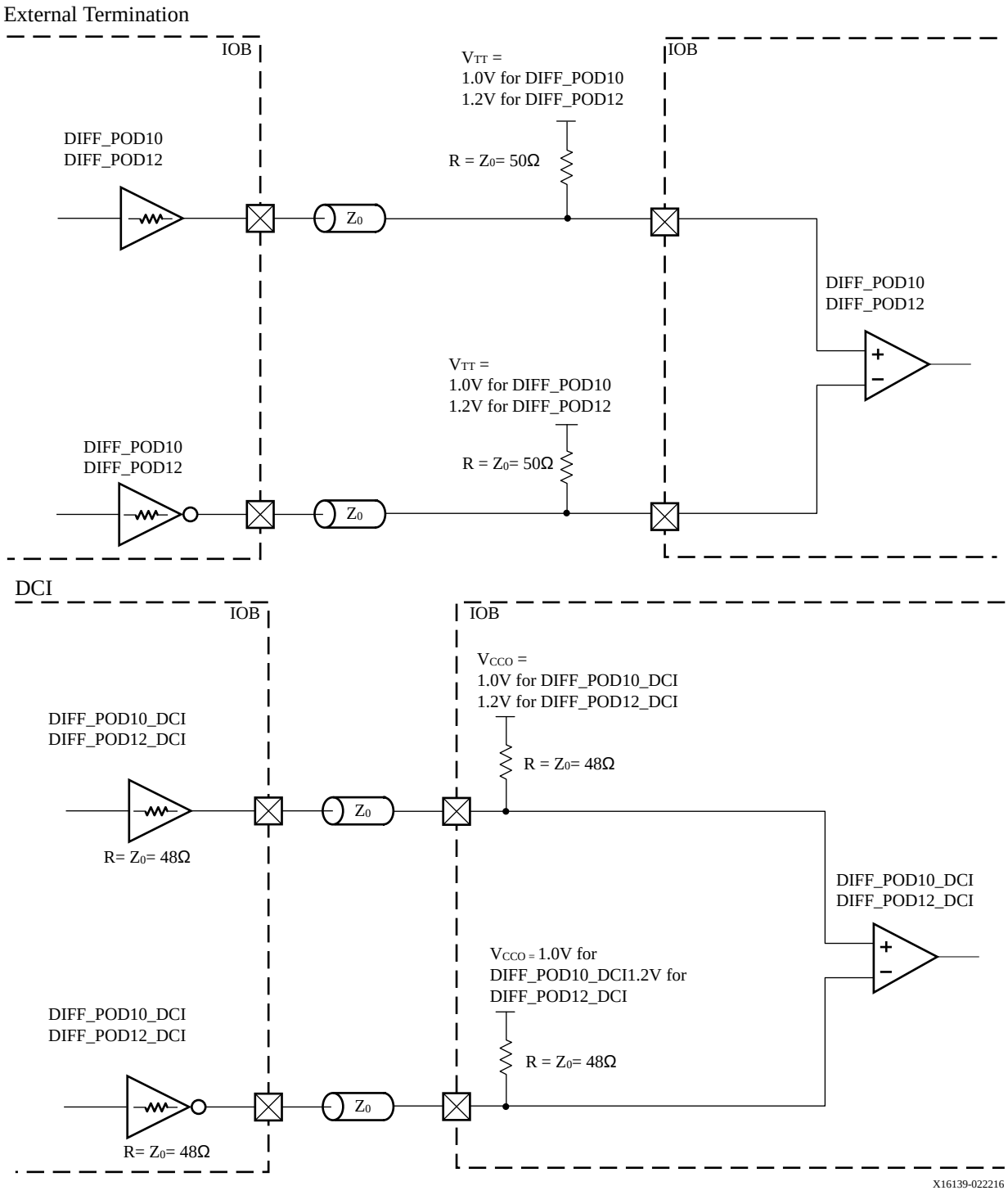
Figure 70: **POD with Bidirectional Signaling**



Differential POD

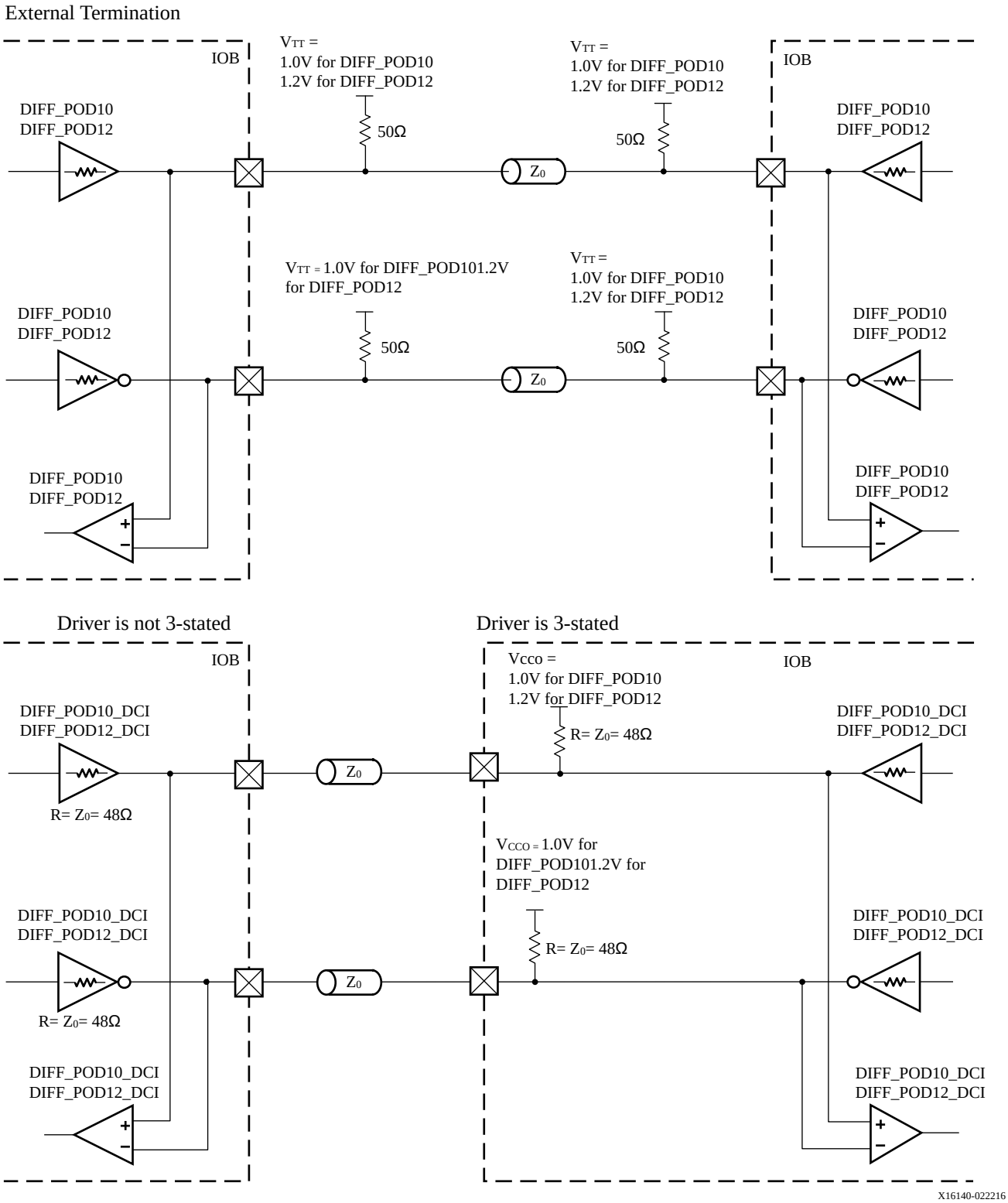
The following figure shows a sample circuit illustrating a termination technique for differential POD (1.0V, 1.2V) with unidirectional termination and with matched driver and receiver termination values. In a specific circuit, all drivers and receivers must be at the same voltage level (1.2V or 1.0V); they are not interchangeable (that is, DIFF_POD12 should only interface with DIFF_POD12).

Figure 71: **Differential POD with Unidirectional Signaling**



The following figure 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 (that is, DIFF_POD10_DCI should only interface with DIFF_POD10_DCI).

Figure 72: **Differential Pod with Bidirectional Signaling**



The following table lists the allowed attributes for POD I/O standards. Support is implied for primitives that are derivatives of the primitives listed in the following table (for example: *_DIFF_OUT, *_DCIEN, *_IBUFDISABLE, or *_INTERMDISABLE). Refer to SelectIO IOB Interface Primitives for all supported derivatives.

Table 45: **POD Allowed Attributes**

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
IOSTANDARD	POD10 DIFF_POD10		POD10 DIFF_POD10		POD10 DIFF_POD10	
SLEW	N/A		FAST MEDIUM SLOW	SLOW	FAST MEDIUM SLOW	SLOW
DQS_BIAS[5]	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[1]	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[1]	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[5]	TRUE FALSE	FALSE	N/A		TRUE FALSE	FALSE
ODT	RTT_40 RTT_48 RTT_60[2]	RTT_40	N/A		RTT_40 RTT_48 RTT_60[12]	RTT_40
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[1]	RDRV_40_40
IOSTANDARD	POD12 DIFF_POD12		POD12 DIFF_POD12		POD12 DIFF_POD12	
SLEW	N/A		FAST MEDIUM SLOW[4]	SLOW	FAST MEDIUM SLOW[3]	SLOW

Table 45: **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
PRE_EMPHASIS	N/A		RDRV_240 RDRV_NONE[4]	RDRV_NONE	RDRV_240 RDRV_NONE[3]	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[5]	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)[3]	RTT_NONE
OUTPUT_IMPEDANCE	N/A		RDRV_40_40 RDRV_48_48 RDRV_60_60[4]	RDRV_40_40	RDRV_40_40 RDRV_48_48 RDRV_60_60[3]	RDRV_40_40
IOSTANDARD	POD12_DCI DIFF_POD12_DCI		POD12_DCI DIFF_POD12_DCI		POD12_DCI DIFF_POD12_DCI	
SLEW	N/A		FAST MEDIUM SLOW[4]	SLOW	FAST MEDIUM SLOW[3]	SLOW
PRE_EMPHASIS[6]	N/A		RDRV_240 RDRV_NONE[4]	RDRV_NONE	RDRV_240 RDRV_NONE[3]	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[56]	TRUE FALSE	FALSE	N/A		TRUE FALSE	FALSE
ODT	RTT_40 RTT_48 RTT_60	RTT_40	N/A		RTT_40 RTT_48 RTT_60[23]	RTT_40

Table 45: **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[4]	RDRV_40_40	RDRV_40_40 RDRV_48_48 RDRV_60_60[3]	RDRV_40_40

Notes:

1. The allowed bidirectional configuration combinations for driver output impedance (OUTPUT_IMPEDANCE) and ODT are listed in Table 34.
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 46.
4. The combinations allowed of driver output impedance (OUTPUT_IMPEDANCE) and PRE_EMPHASIS are listed in Table 47.
5. Only applicable to DIFF_POD I/O standards.
6. This attribute has to be used 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.

The following tables lists the allowed combinations of attributes for POD I/O standards.

Table 46: **Allowed Combinations of OUTPUT_IMPEDANCE, ODT, and PRE_EMPHASIS**

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

Table 47: **Allowed Combinations of OUTPUT_IMPEDANCE and PRE_EMPHASIS**

OUTPUT_IMPEDANCE	SLEW	PRE_EMPHASIS
RDRV_40_40 (40 Ω)	SLOW, MEDIUM, FAST	RDRV_NONE

Table 47: **Allowed Combinations of OUTPUT_IMPEDANCE and PRE_EMPHASIS** *(cont'd)*

OUTPUT_IMPEDANCE	SLEW	PRE_EMPHASIS
RDRV_48_48 (48 Ω)	SLOW, MEDIUM, FAST	RDRV_NONE
RDRV_60_60 (60 Ω)	SLOW, MEDIUM, FAST	RDRV_NONE
RDRV_40_40 (40 Ω)	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 (HP I/O banks only) in the IOBs and the optional internal differential termination feature (HP I/O banks only), the need for external source termination in point-to-point applications is eliminated. Spartan 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_{CCO} 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 HD I/O banks for input-only mode.

Table 48: **Available I/O Bank Type**

HD	HP
LVDS_25 in input only	Available for LVDS only

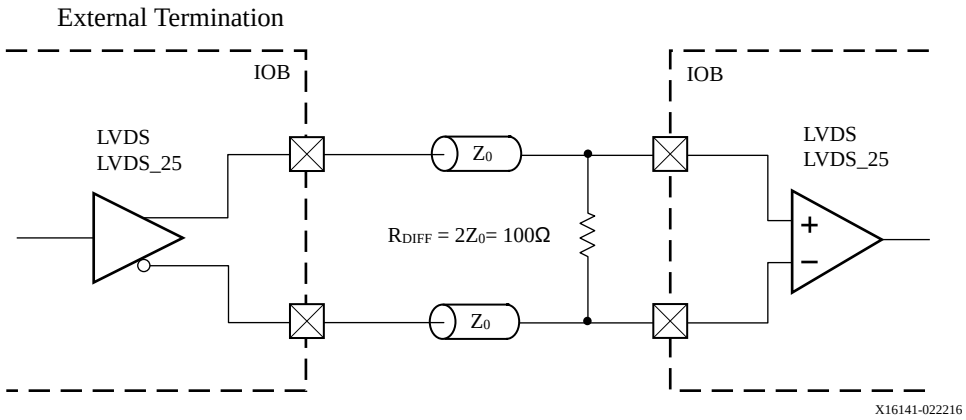
Transmitter Termination

The LVDS transmitter does not require any external termination. The following table 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 (HP I/O Banks Only)

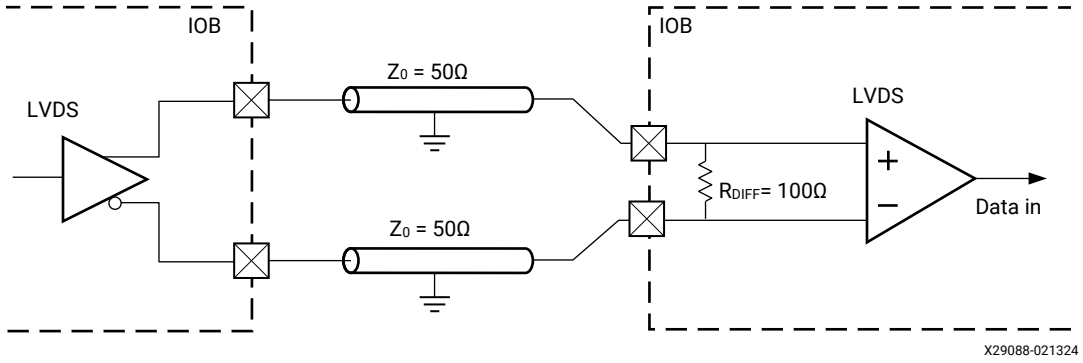
The following figure is an example of differential termination for an LVDS or LVDS_25 receiver on a board with 50 Ω transmission lines.

Figure 73: **LVDS or LVDS_25 Receiver Termination**



The following figure is an example of internal differential termination for an LVDS receiver on a board with 50 Ω transmission lines.

Figure 74: **LVDS with DIFF_TERM Receiver Termination**



The following table lists the allowed attributes for the LVDS I/O standards. Support is implied for primitives that are derivatives of the primitives listed in the following table (for example: *_DIFF_OUT, *_DCIEN, *_IBUFDISABLE, or *_INTERMDISABLE). Refer to SelectIO IOB Interface Primitives for all supported derivatives.

Table 49: **Allowed Attributes for the LVDS I/O Standards**

Attributes	IBUFDS				OBUFDS			
	HP I/O		HD I/O		HP I/O		HD I/O	
	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default
IOSTANDARD	LVDS		LVDS_25		LVDS		N/A	
DQS_BIAS	TRUE FALSE	FALSE	N/A		N/A		N/A	
EQUALIZATION	EQ_LEVEL0 EQ_LEVEL1 EQ_LEVEL2 EQ_LEVEL3 EQ_LEVEL4 EQ_NONE	EQ_NONE	N/A	N/A	N/A		N/A	
LVDS_PRE_EMPHASIS	N/A		N/A		TRUE FALSE	FALSE	N/A	
DIFF_TERM	TRUE FALSE	FALSE	N/A		N/A		N/A	
DIFF_TERM_ADV	TERM_100 TERM_NONE	TERM_NONE	N/A		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 50.
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. This attribute must be used with ENABLE_PRE_EMPHASIS to enable the pre-emphasis function.

Table 50: **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

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). However, these criteria must be met:

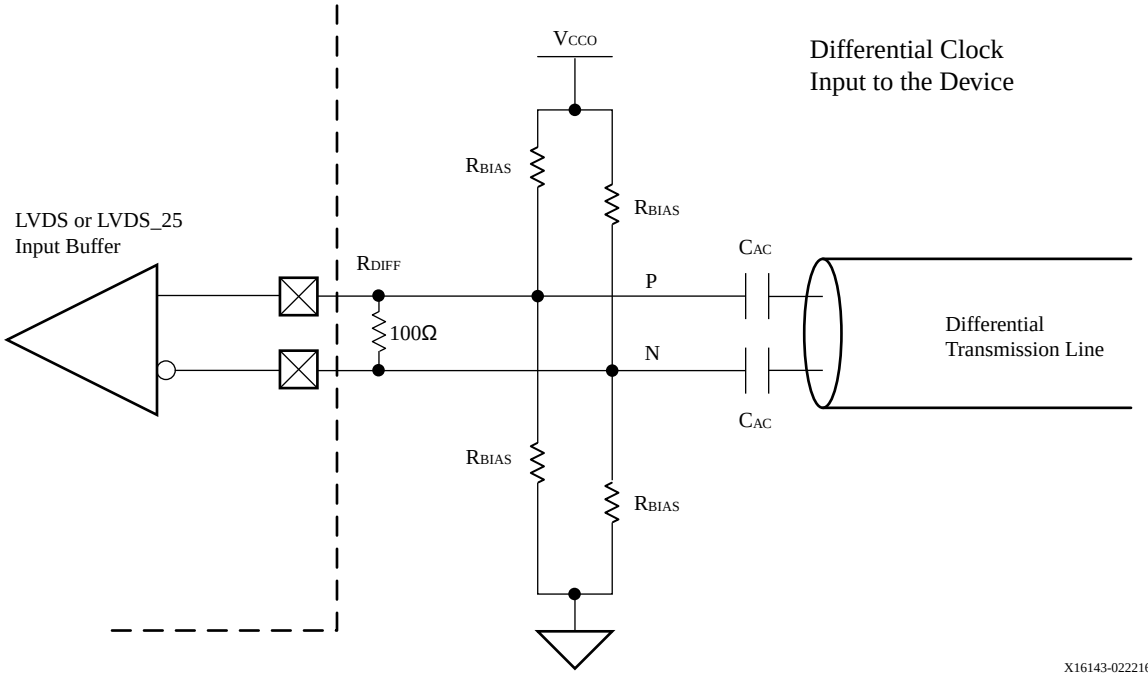
- The optional internal differential termination is not used.
- DIFF_TERM_ADV = TERM_NONE
- DIFF_TERM = FALSE (default).
- The differential signals at the input pins meet the V_{IN} requirements in the Recommended Operating Conditions table of the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930).
- The differential signals at the input pins meet the V_{IDIFF} (min) requirements in the corresponding LVDS or LVDS_25 DC specifications tables of the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930).

One way to accomplish this criteria is to use an external circuit that both AC-couples and DC-biases the input signals. The following figure shows an example circuit for providing an AC-coupled and DC-biased circuit for a differential clock input. R_{DIFF} provides the 100 Ω 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 1.8 V source (typically V_{CCO} or V_{CCAUX}) to ensure the input common-mode voltage for AC-coupled signals is maintained. Resistors in the 1K–100 KΩ 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 *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) 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 Spartan 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.

HD I/O banks do not support any equalization settings.

Figure 75: **Example Circuit for AC-Coupled and External DC-Biased Differential Clock Input**



SUB_LVDS

Table 51: **Available I/O Bank Type**

HD	HP
Available for SUB_LVDS	Available for SUB_LVDS

The following table lists the allowed attributes for the SUB_LVDS I/O standards. Support is implied for primitives that are derivatives of the primitives listed in the following table (for example: *_DIFF_OUT, *_DCIEN, *_IBUFDISABLE, or *_INTERMDISABLE). Refer to SelectIO IOB Interface Primitives for all supported derivatives.

Table 52: **Allowed Attributes for the SUB_LVDS I/O Standards**

Attributes	Primitives		
	IBUFDS		OBUFDS or OBUFTDS
	Allowed Values	Default	
IOSTANDARD	SUB_LVDS		
DIFF_TERM (HP I/O banks only)	TRUE FALSE	FALSE	N/A
DIFF_TERM_ADV (HP I/O banks only)	TERM_NONE TERM_100	TERM_NONE	N/A

SLVS_400

SLVS_400 is supported in HD 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 53: **Available I/O Bank Type**

HD	HP
Available for SLVS_400_25 input only	Available for SLVS_400_18 only

The following table lists the allowed attributes for the SLVS_400 I/O standards. Support is implied for primitives that are derivatives of the primitives listed in the following table (for example: *_DIFF_OUT, *_DCIEN, *_IBUFDISABLE, or *_INTERMDISABLE). Refer to SelectIO IOB Interface Primitives for all supported derivatives.

Table 54: **Allowed Attributes for the SLVS_400 I/O Standards**

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

LVPECL

LVPECL is supported only in HD I/O banks, and is only supported in receive mode.

Table 55: **Available I/O Bank Type**

HD	HP
Input Only	N/A

The following table lists the allowed attributes for the LVPECL I/O standard. Support is implied for primitives that are derivatives of the primitives listed in the following table (for example: *_DIFF_OUT, *_IBUFDISABLE, or *_INTERMDISABLE). Refer to SelectIO IOB Interface Primitives for all supported derivatives.

Table 56: **Allowed Attributes for the LVPECL I/O Standard**

Attributes	Primitives	
	IBUFDS	OBUFDS or OBUFTDS
IOSTANDARD	LVPECL (HD 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. Support for this standard in Spartan UltraScale+ devices is in adherence to the MIPI Alliance interface specifications.

IMPORTANT! *MIPI_DPHY_DCI, as with other DCI standards, requires a 240 Ω 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 57: **Available I/O Bank Type**

HD	HP
N/A	Supported

The following table lists the allowed attributes for the MIPI_DPHY_DCI I/O standard.

Table 58: **Allowed Attributes for the MIPI_DPHY_DCI Standard**

Attributes	IBUFDS_DPHY			OBUFDS_DPHY		
	HP I/O		HD I/O	HP I/O		HD 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

Internal Differential Termination Behavior in Differential I/O Standards in HP I/O Banks

The internal differential termination (100 Ω) 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 the following table.

Table 59: **Internal Differential Termination Behavior In Differential I/O Standards**

Primitive[1]	Driving	3-state/Receiving
OBUFDS	Internal differential termination is ON	N/A
OBUFTDS	Internal differential termination is ON	Internal differential termination is ON.

Table 59: **Internal Differential Termination Behavior In Differential I/O Standards** *(cont'd)*

Primitive[1]	Driving	3-state/Receiving
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:

1. Support is implied for primitives that are derivatives of the primitives listed (for example, *_DIFF_OUT, *_DCIEN, *_IBUFDISABLE, or *_INTERMDISABLE). Refer to SelectIO IOB Interface Primitives 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 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_I inputs

   Incompatible example:

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

   Incompatible example:

   HSTL_I_DCI_18 (VREF = 0.9V) and HSTL_I_DCI (VREF = 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 input and LVCMOS25 output

Incompatible example:

LVCMOS18 output (output V_{CCO} = 1.8 V) and HSTL_I output (input V_{CCO} = 1.5 V)

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.

The following table 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 *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930).

Table 60: **V_{CCO} and V_{REF} Requirements for Each Supported I/O Standard**

I/O Standard	I/O Bank Availability	VCCO (V)			VREF (V)
		Output	Input	Input with DIFF_TERM_ADV and DIFF_TERM Support	Input
LVTTL	HD	3.3	3.3	N/A	N/A
LVCMOS33	HD	3.3	3.3	N/A	N/A
LVCMOS25	HD	2.5	2.5	N/A	N/A
LVCMOS18	HP/HD	1.8	1.8	N/A	N/A
LVCMOS15	HP/HD	1.5	1.5	N/A	N/A
LVCMOS12	HP/HD	1.2	1.2	N/A	N/A
HSUL_12	HP/HD	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/HD	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	HP/HD	1.5	1.5	N/A	0.75
HSTL_I_DCI	HP	1.5	1.5	N/A	0.75
HSTL_I_18	HP/HD	1.8	1.8	N/A	0.90

Table 60: **V_{CCO} and V_{REF} Requirements for Each Supported I/O Standard** *(cont'd)*

I/O Standard	I/O Bank Availability	VCCO (V)			VREF (V)
		Output	Input	Input with DIFF_TERM_ADV and DIFF_TERM Support	Input
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	HP/HD	1.8	1.8	N/A	0.90
SSTL15	HP/HD	1.5	1.5	N/A	0.75
SSTL135	HP/HD	1.35	1.35	N/A	0.675
SSTL12	HP/HD	1.2 (N/A for HDIO)	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	HP/HD	1.5	1.5[2]	N/A	N/A
DIFF_HSTL_I_18	HP/HD	1.8	1.8[2]	N/A	N/A
DIFF_SSTL18_I	HP/HD	1.8	1.8[2]	N/A	N/A
DIFF_SSTL15	HP/HD	1.5	1.5[2]	N/A	N/A
DIFF_SSTL135	HP/HD	1.35	1.35[2]	N/A	N/A
DIFF_SSTL12	HP/HD	1.2 (N/A for HDIO)	1.2[2]	N/A	N/A
DIFF_HSUL_12	HP/HD	1.2	1.2[3]	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

Table 60: **V_{CCO} and V_{REF} Requirements for Each Supported I/O Standard** *(cont'd)*

I/O Standard	I/O Bank Availability	VCCO (V)			VREF (V)
		Output	Input	Input with DIFF_TERM_ADV and DIFF_TERM Support	Input
DIFF_HSUL_12_DCI	HP	1.2	1.2	N/A	N/A
BLVDS_25	HD	N/A	Any	N/A	N/A
LVDS_25	HD	N/A	2.5[1]	N/A	N/A
LVDS	HP	1.8	1.8[1]	1.8	N/A
LVPECL	HD	N/A	Any	N/A	N/A
SLVS_400_18	HP	N/A	1.8[1]	1.8	N/A
SLVS_400_25	HD	N/A	2.5[1]	N/A	N/A
SUB_LVDS	HP/HD	1.8 (N/A for HDIO)	1.8[1]	1.8 (HP Only)	N/A
DIFF_HSTL_I_12	HP	1.2	1.2[2]	N/A	N/A
DIFF_POD10	HP	1.0	1.0[2]	N/A	N/A
DIFF_POD12	HP	1.2	1.2[2]	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

Table 60: **V_{CCO} and V_{REF} Requirements for Each Supported I/O Standard** *(cont'd)*

I/O Standard	I/O Bank Availability	VCCO (V)			VREF (V)
		Output	Input	Input with DIFF_TERM_ADV and DIFF_TERM Support	Input
MIPI_DPHY_DCI	HP	1.2	1.2	1.2	N/A

Notes:

1. 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:
   a. 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.
   b. The differential signals at the input pins meet the V_{IN} requirements in the Recommended Operating Conditions table of the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930).
   c. The differential signals at the input pins meet the V_{IDIFF} and V_{ICM} requirements in the DC Specifications tables in the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930). In some cases, to accomplish this it might be necessary to provide an external circuit to both AC-couple and DC-bias the pins.
2. 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 VCCO input voltage is any allowed voltage.
3. 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.
4. 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 *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930).

The following table summarizes the DRIVE and SLEW attribute options, bidirectional buffer availability, and DCI termination type for each supported I/O standard.

Table 61: **Attribute Options, Bidirectional Buffer Availability, and DCI Termination Type**

I/O Standard	I/O Bank Type	Output Slew				Output Drive				Bidirectional Buffers[1]	Termination Type[2]	
		HD I/O Banks		HP I/O Banks		HD I/O Banks		HP I/O Banks			Input	Output[3]
		Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default			
LVTTL	HD	SLOW FAST	SLOW	N/A		4, 8, 12	12	N/A		Yes	None	None
LVCMOS33	HD	SLOW FAST	SLOW	N/A		4, 8, 12	12	N/A		Yes	None	None

Table 61: **Attribute Options, Bidirectional Buffer Availability, and DCI Termination Type** *(cont'd)*

I/O Standard	I/O Bank Type	Output Slew				Output Drive				Bidirectional Buffers[1]	Termination Type[2]	
		HD I/O Banks		HP I/O Banks		HD I/O Banks		HP I/O Banks			Input	Output[3]
		Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default			
LVCMOS25	HD	SLOW FAST	SLOW	N/A		4, 8, 12	12	N/A		Yes	None	None
LVCMOS18	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	4, 8, 12	12	2, 4, 6, 8, 12	12	Yes	None	None
LVCMOS15	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	4, 8, 12	12	2, 4, 6, 8, 12	12	Yes	None	None
LVCMOS12	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	4, 8	12	2, 4, 6, 8	12	Yes	None	None
HSUL_12	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes		Driver
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

Table 61: **Attribute Options, Bidirectional Buffer Availability, and DCI Termination Type** *(cont'd)*

I/O Standard	I/O Bank Type	Output Slew				Output Drive				Bidirectional Buffers[1]	Termination Type[2]	
		HD I/O Banks		HP I/O Banks		HD I/O Banks		HP I/O Banks			Input	Output[3]
		Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default			
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_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_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	Driver
SSTL18_I	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
SSTL15	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver

Table 61: **Attribute Options, Bidirectional Buffer Availability, and DCI Termination Type** *(cont'd)*

I/O Standard	I/O Bank Type	Output Slew				Output Drive				Bidirectional Buffers[1]	Termination Type[2]	
		HD I/O Banks		HP I/O Banks		HD I/O Banks		HP I/O Banks			Input	Output[3]
		Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default			
SSTL135	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
SSTL12	Both	N/A		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
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_I_18	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
DIFF_SSTL18_I	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver

Table 61: **Attribute Options, Bidirectional Buffer Availability, and DCI Termination Type** *(cont'd)*

I/O Standard	I/O Bank Type	Output Slew				Output Drive				Bidirectional Buffers[1]	Termination Type[2]	
		HD I/O Banks		HP I/O Banks		HD I/O Banks		HP I/O Banks			Input	Output[3]
		Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default			
DIFF_SSTL15	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
DIFF_SSTL135	Both	SLOW FAST	SLOW	SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
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[4]	Driver[4]
DIFF_HSTL_I_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
DIFF_HSTL_I_DCI_18	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
DIFF_SSTL18_I_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
DIFF_SSTL15_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
DIFF_SSTL135_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver

Table 61: **Attribute Options, Bidirectional Buffer Availability, and DCI Termination Type** *(cont'd)*

I/O Standard	I/O Bank Type	Output Slew				Output Drive				Bidirectional Buffers[1]	Termination Type[2]	
		HD I/O Banks		HP I/O Banks		HD I/O Banks		HP I/O Banks			Input	Output[3]
		Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default			
DIFF_SSTL12_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
DIFF_HSUL_12_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Single	Driver
LVDS_25	HD	N/A		N/A		N/A		N/A		No	None	None
LVDS	HP	N/A		N/A		N/A		N/A		Yes[4]	None	None
LVPECL	HD	N/A		N/A		N/A		N/A		No	None	None
SLVS_400_18	HP	N/A		N/A		N/A		N/A		No	None	None
SUB_LVDS	Both	N/A		N/A		N/A		N/A		Yes[4]	None	None
DIFF_HSTL_I_12	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Split	Driver
DIFF_POD10	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Single	Driver
DIFF_POD12	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Single	Driver
DIFF_HSTL_I_DCI_12	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	N/A	Driver
DIFF_POD10_DCI	HP	N/A		SLOW MEDIUM FAST	SLOW	N/A		N/A		Yes	Single	Driver

Table 61: **Attribute Options, Bidirectional Buffer Availability, and DCI Termination Type** *(cont'd)*

I/O Standard	I/O Bank Type	Output Slew				Output Drive				Bidirectional Buffers[1]	Termination Type[2]	
		HD I/O Banks		HP I/O Banks		HD I/O Banks		HP I/O Banks			Input	Output[3]
		Allowed Values	Default	Allowed Values	Default	Allowed Values	Default	Allowed Values	Default			
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	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_{CCO}.
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.
5. The bidirectional configuration on these I/O standards is a fixed impedance structure optimized to 100 Ω differential. They are intended to only be used in point-to-point transmissions that do not have turn around timing requirements.

XP5IO Features and Standard Support

This section is intended to provide a general overview of supported features in the XP5IO. While XP5IO I/O banks contain similar features to HP I/O banks, there are differences between the banks regarding feature and IOSTANDARD support. While many IOSTANDARDS are very similar to HP I/O banks described earlier, the XP5IO adds low-voltage swing terminated logic (LVSTL) standards that are optimized for lower-power memory interfaces. To support the LVSTL family of standards, the XP5IO adds internal ODT support for single to GND, which provides calibrated impedance to ground.

The following list calls out the IOSTANDARDS that are supported in the XP5IO:

- LVCMOS (1.0V, 1.1V, 1.2V, 1.35V, 1.5V): LVCMOS10, LVCMOS11, LVCMOS12, LVCMOS135, LVCMOS15
- SSTL (1.2V, 1.35V, 1.5V): SSTL12, SSTL135, SSTL15
- HSTL (1.2V and 1.5V): HSTL_I_12, HSTL_I_15
- POD (1.0V and 1.2V): POD10, POD12
- HSUL_12 (1.2V): HSUL_12
- MIPI D-PHY (1.2V): MIPI_DPHY
- LVSTL (1.0V, 1.1V, 1.2V): LVSTL_11, LVSTL055_11, LVSTL05_10, LVSTL06_12
- DIFF_SSTL (1.2V, 1.35V, 1.5V): DIFF_SSTL12, DIFF_SSTL135, DIFF_SSTL15
- Differential HSTL (1.2V and 1.5V): DIFF_HSTL_I_12, DIFF_HSTL_I_15
- DIFF_POD (1.0V and 1.2V): DIFF_POD10, DIFF_POD12
- DIFF_LVSTL (1.0V, 1.1V, 1.2V): DIFF_LVSTL_11, DIFF_LVSTL055_11, DIFF_LVSTL05_10, DIFF_LVSTL06_12
- LVDS15

Along with the addition of new IOSTANDARDS, XP5IO includes many features similar to HP, with the addition of or enhancement to others. The following list provides a general list of supported features with an associated brief description:

- DCI: Calibrated impedance control, split, single to V_{CCO} and single to GND internal calibrated termination is available in XP5IO. XP5IO uses a single 240 Ω VR reference resistor for all XP5IO banks.
- Fast, medium, and slow driver SLEW control
- PULLUP, PULLDOWN, KEEPER capability in each IOB
- 100 Ω differential termination for LVDS15 and MIPI_DPHY
- Eight levels of equalization for POD and LVSTL standards

- Internal weak bus-hold bias (DQS_BIAS) and weak bias for AC-coupled signals (DC_BIAS)
- Per-nibble internal VREF tuning (no external VREF pin in XP5IO)

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 LVTTL 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.

Chapter 3

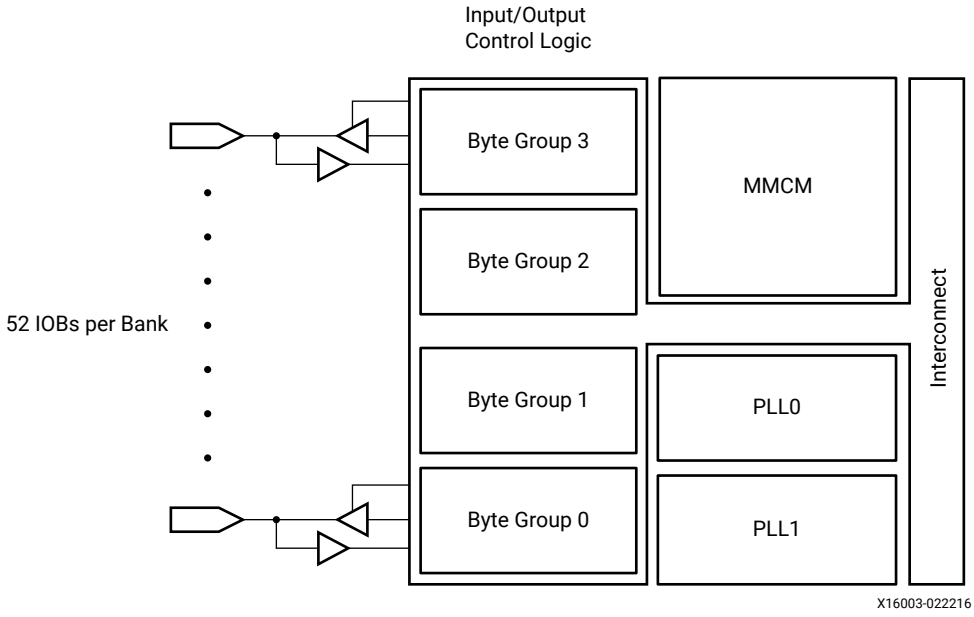
HP I/O Bank SelectIO Interface Logic Resources

HP I/O Bank Overview

Each HP I/O bank contains 52 pins that can be used for input, output, or bidirectional operations using single-ended standards appropriate for the bank. Up to 48 of these pins can be configured as up to 24 differential signaling pin pairs with signaling standards appropriate for 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 the following figure. 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 76: **Bank Overview**



> **TIP:** *Native mode designs have additional restrictions. The High-Speed SelectIOwizard (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.

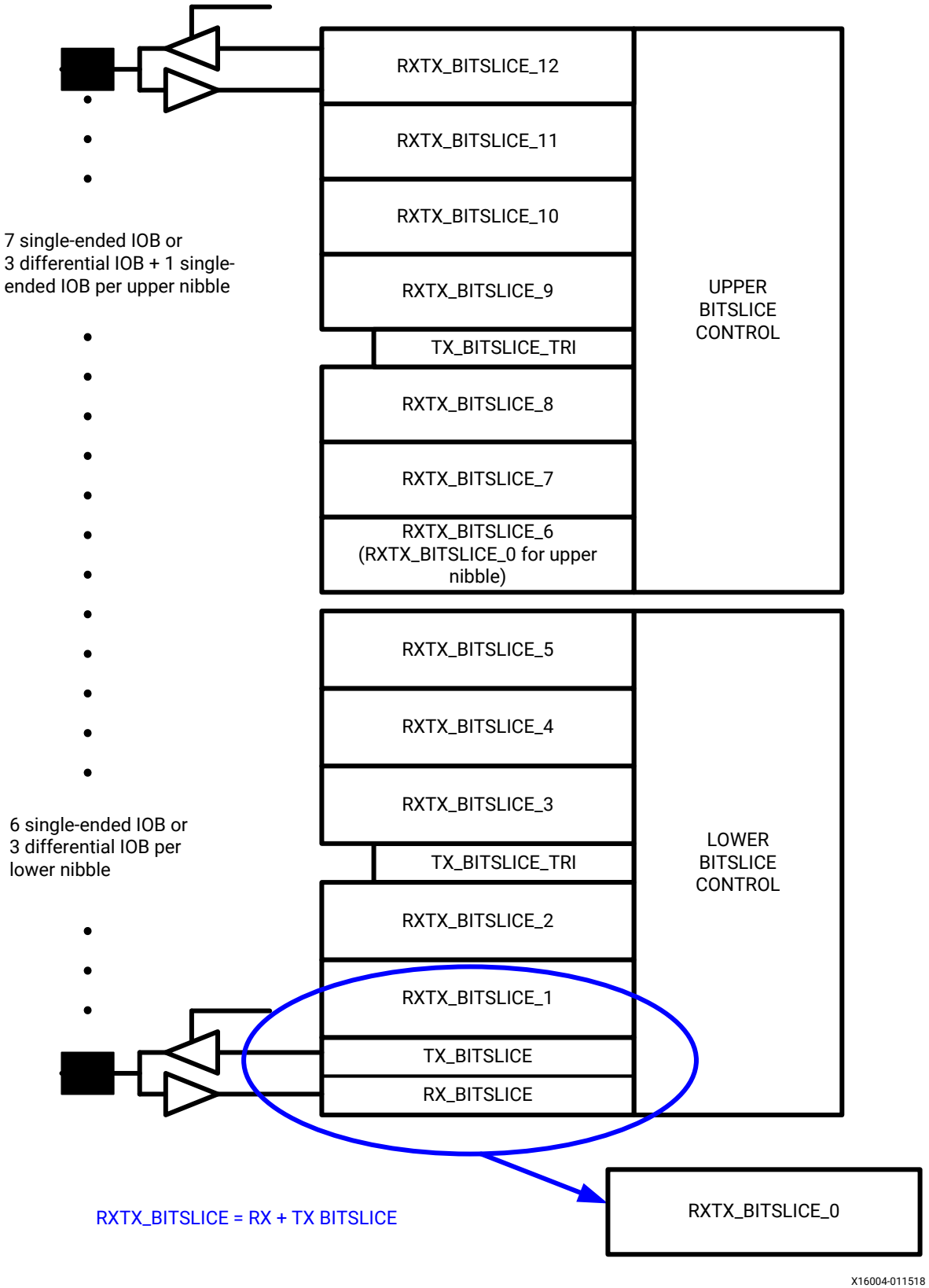
```
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets <clock_net_name>]
```

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 the preceding figure. Each byte group is further sub-divided into two nibble groups, as shown in the following figure. 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 Resources User Guide* (UG572).

Figure 77: **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 in the following figure 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 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

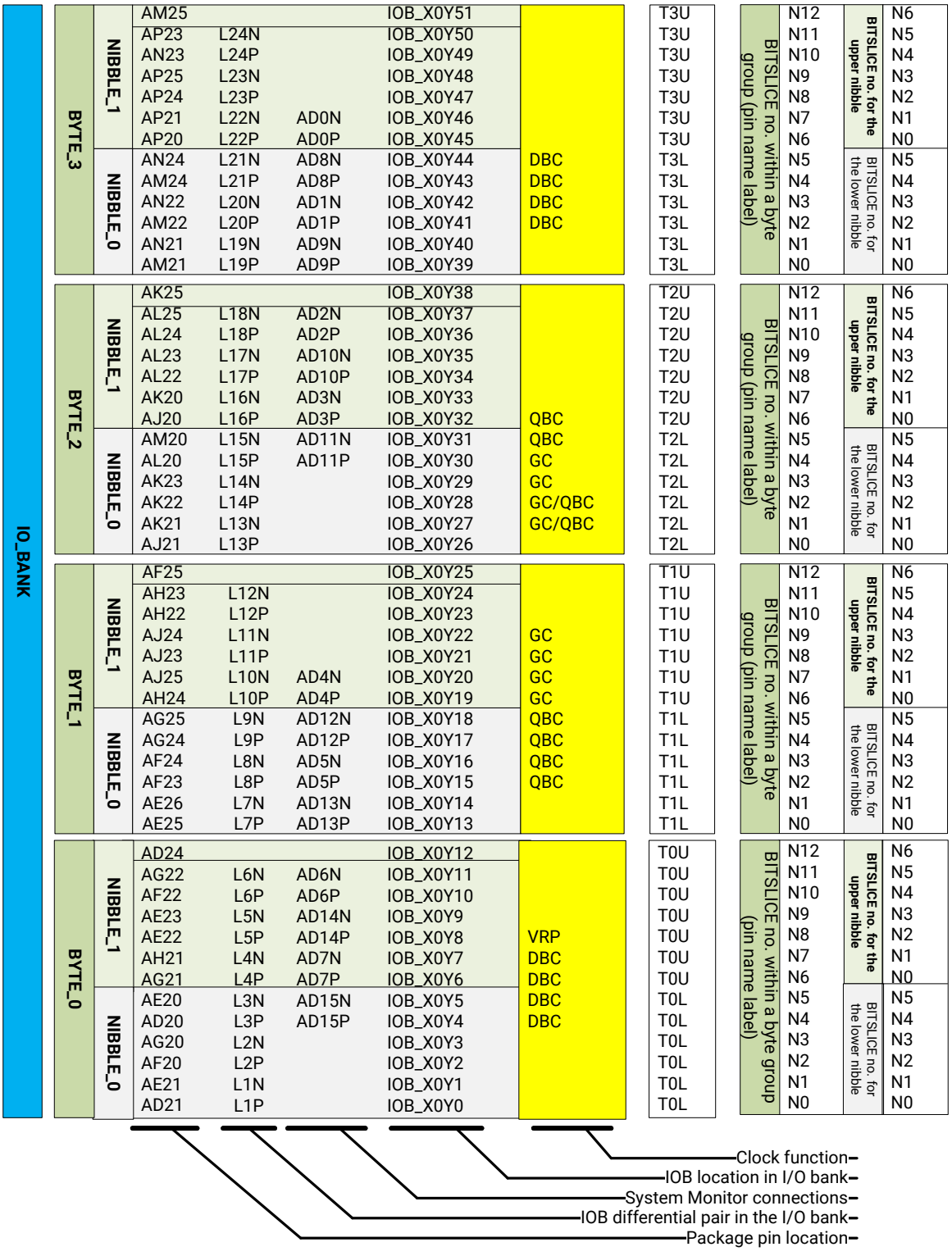
Vivado will issue an error message to indicate input routing and logic associated with BITSLICE_0 within a nibble will be unavailable during the BISC operation. If these restrictions do not affect a design, the DRC can be disabled with the following constraint:

```
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 Spartan UltraScale+ FPGA I/O banks, bytes, and nibbles have the same setup. The following figure shows an example of a pin setting of HP I/O bank. The setup shown can be applied to all I/O banks over the entire FPGA family. Using the following figure simplifies making pin nibble, byte, and I/O bank pin assignments.

Figure 78: **Example I/O Bank**



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 with the IOB = TRUE constraint applied to the flip-flop instance. IS_D_INVERTED is not supported for Spartan UltraScale+ devices 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-flip 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

Spartan 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:

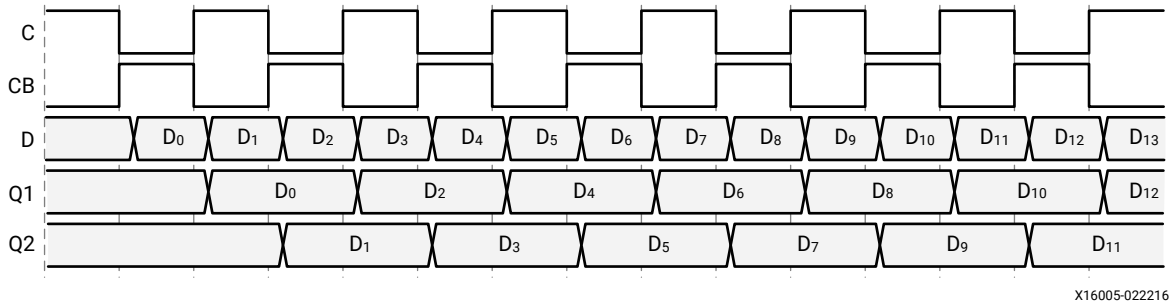
- OPPOSITE_EDGE
- SAME_EDGE
- SAME_EDGE_PIPELINED

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 though 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. The following figure shows the timing diagram of the input DDR using the OPPOSITE_EDGE mode.

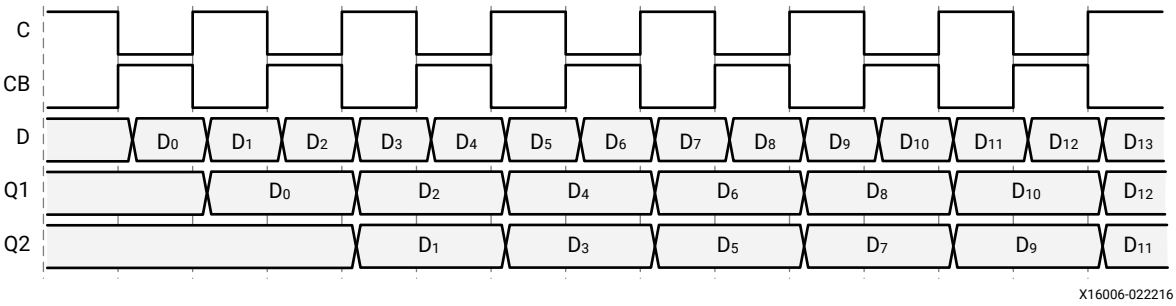
Figure 79: **Input DDR Timing in OPPOSITE_EDGE Mode**



SAME_EDGE Mode

In SAME_EDGE mode, the data is presented into the device logic on the same clock edge. The following figure 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 (do not care), followed by pair (1) and (2) on the next clock cycle.

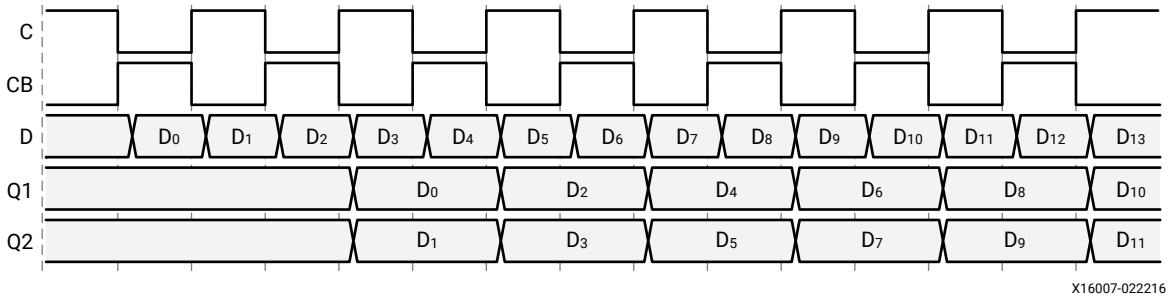
Figure 80: **Input DDR Timing in SAME_EDGE Mode**



SAME_EDGE_PIPELINED Mode

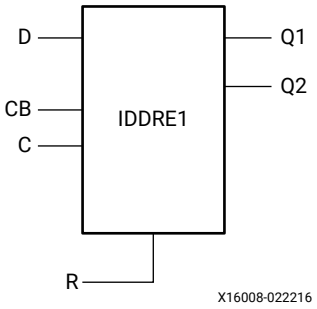
In SAME_EDGE_PIPELINED mode, the data is presented into the device logic on the same clock edge. Unlike the SAME_EDGE mode, the data pair is not separated by one clock cycle. However, additional clock latency is required to remove the separated effect of the SAME_EDGE mode. The following figure shows 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 81: **Input DDR Timing in SAME_EDGE_PIPELINED Mode**



The following figure shows a block diagram of the IDDRE1 primitive.

Figure 82: **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.

The following table lists the IDDRE1 ports.

Table 62: **IDDRE1 Ports**

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

IDDRE1 Attributes

The following table lists the IDDRE1 attributes.

Table 63: **IDDRE1 Attributes**

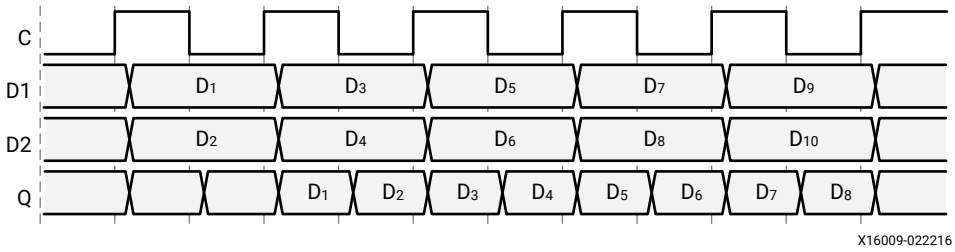
Attribute	Values	Default	Type	Description
DDR_CLK_EDGE	OPPOSITE_EDGE SAME_EDGE SAME_EDGE_PIPELINED	OPPOSITE_EDGE	String	Sets the IDDRE1 mode of operation with respect to clock edge.
IS_C_INVERTED	0 or 1	0	Bit	Sets a local clock inversion for C input when 1.
IS_CB_INVERTED	0 or 1	0	Bit	Sets a local clock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

Spartan 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.

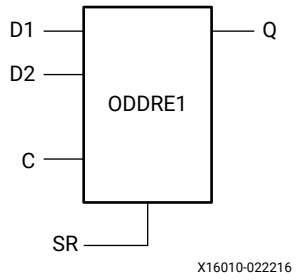
The ODDRE1 primitive supports only the SAME_EDGE mode of operation. The SAME_EDGE mode allows designers to present both data inputs to the ODDRE1 primitive on the rising edge of the ODDRE1 clock, saving CLB and clock resources and increasing performance. This mode is also supported for 3-state control. The timing diagram of the output DDR is shown in the following figure.

Figure 83: **Output DDR Timing**



The following figure shows a block diagram of the ODDRE1 primitive.

Figure 84: **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

The following table lists the ODDRE1 ports.

Table 64: **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

The following table lists the ODDRE1 attributes.

ODDRE1 Attributes

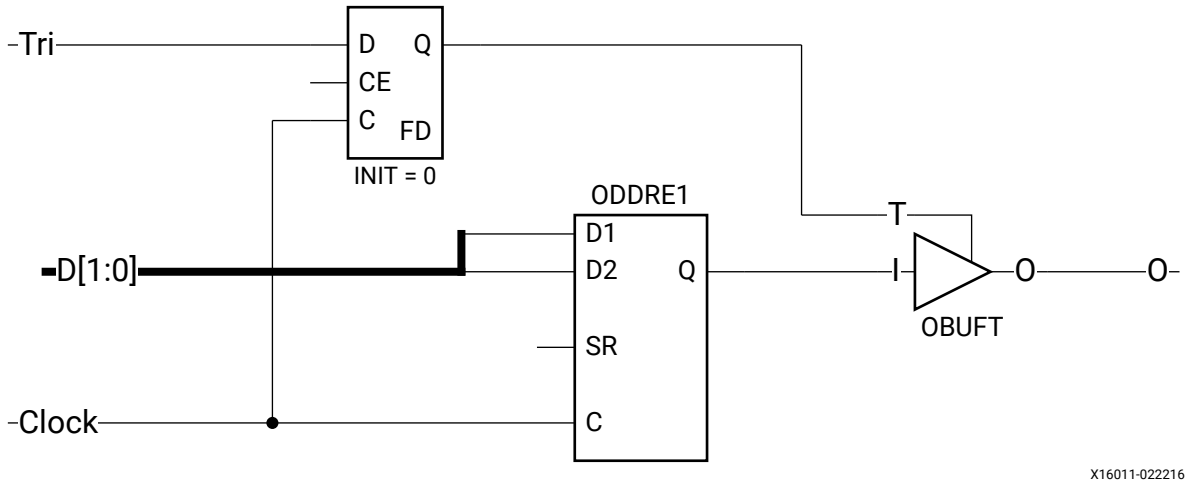
Attribute	Values	Default	Type	Description
SRVAL	0 or 1	0	Bit	Value of Q output configuration after a reset.
IS_C_INVERTED	0 or 1	0	Bit	Local signal inversion.
IS_D1_INVERTED	0 or 1	0	Bit	Not supported ***Note*:** Simulation results will not match hardware if IS_D1_INVERTED is set to 1.

Attribute	Values	Default	Type	Description
IS_D2_INVERTED	0 or 1	0	Bit	Not supported ***Note*:** Simulation results will not match hardware if IS_D2_INVERTED is set to 1.
SIM_DEVICE	ULTRASCALE_PLUS	ULTRASCALE	String	Device family for behavioral simulation.

ODDR with Serialized 3-State

The Spartan UltraScale+ device ODDRE1 solution supports both a single (see the following figure) and a serialized 3-state source shown in the figure next to the following figure.

Figure 85: **ODDR with Internal Logic Flip-Flop 3-State**

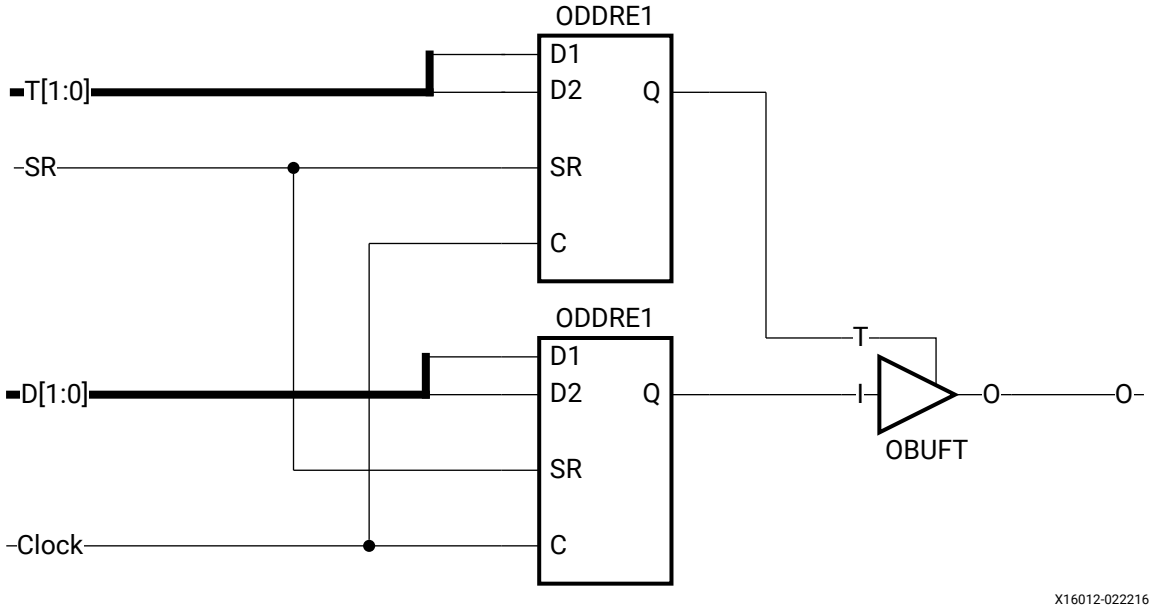


TIP: *To achieve required timing constraints for a design set up as in the preceding figure, it might be necessary to LOC the flip-flop in FPGA logic close to the ODDRE1/OSERDESE3 used.*

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 the following figure can be altered to tie the 3-state D1 and D2 inputs together to a common 3-state.

The following figure 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.

Figure 86: **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 Spartan 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 bitslip 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 the following table.

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

Table 65: **ISERDESE3 Output Connections in SDR and DDR Modes**

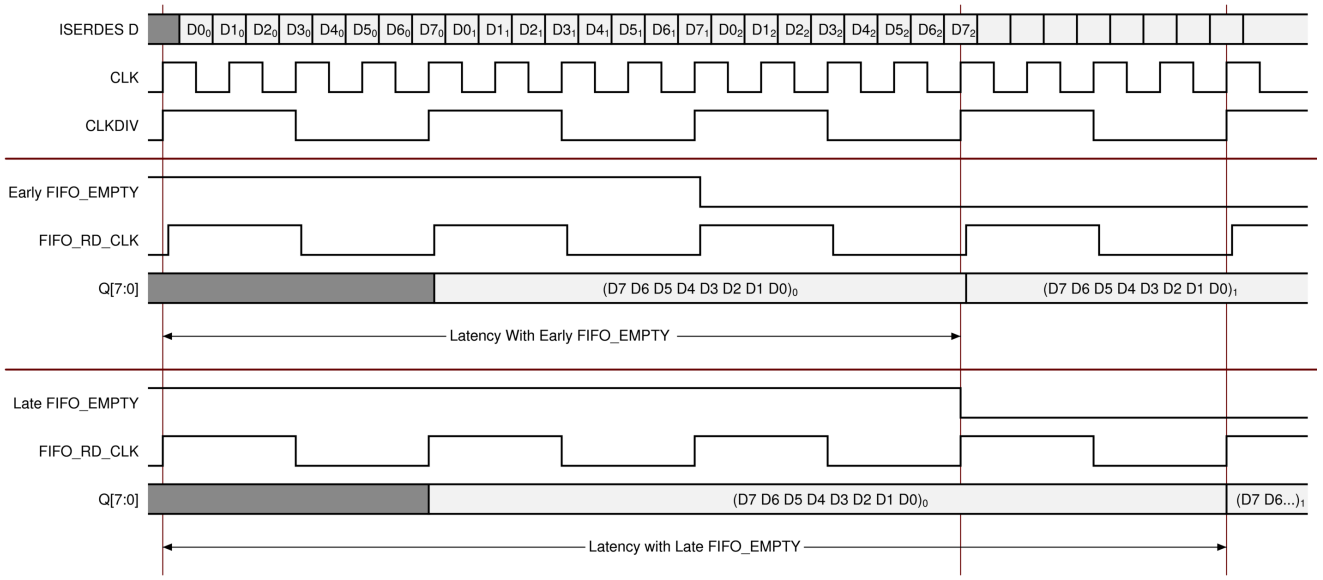
SDR or DDR	Required Ratio	DATA_WIDTH Attribute to Apply to ISERDESE3	SerDes Output Data Bits to Use
DDR	1:8	8	Q7, Q6, Q5, Q4, Q3, Q2, Q1, Q0
DDR	1:4	4	Q3, Q2, Q1, Q0
SDR	1:8	N/A	N/A
SDR	1:4	8	Q6, Q4, Q2, Q0
SDR	1:2	4	Q2, Q0

Other deserialization ratios and word alignment schemes are possible through the use of additional logic resources in the FPGA logic (see *Bitslip in Logic* (XAPP1208). 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 the following figures, 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 103). To ensure the ISERDES is properly aligned after a reset, see Component Mode Reset Sequence.

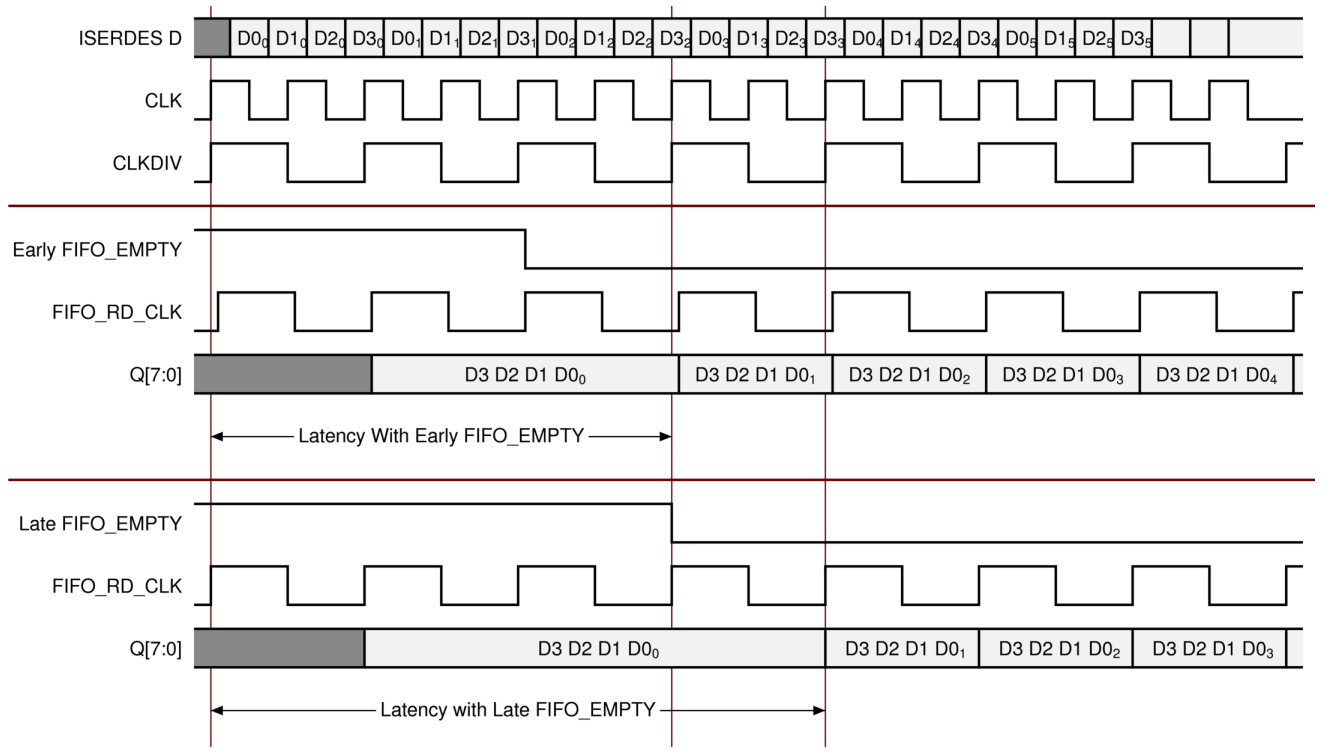
When the clocks are not compensated for, such as when clock-capable inputs are directly connected to clock buffers (BUFG, BUFGCE, BUFGCE_DIV), additional bitslip logic is required. See *Bitslip in Logic* (XAPP1208).

Figure 87: **ISERDES FIFO Latency with DATA_WIDTH = 8**



X19091-121718

Figure 88: **ISERDES FIFO Latency with DATA_WIDTH = 4**



X19090-121718

ISERDESE3 Ports

The following table lists the ISERDESE3 ports.

Table 66: **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

The following table lists the ISERDESE3 attributes.

Table 67: **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.
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_PLUS	ULTRASCALE	String	Device family for behavioral simulation.

OSERDESE3

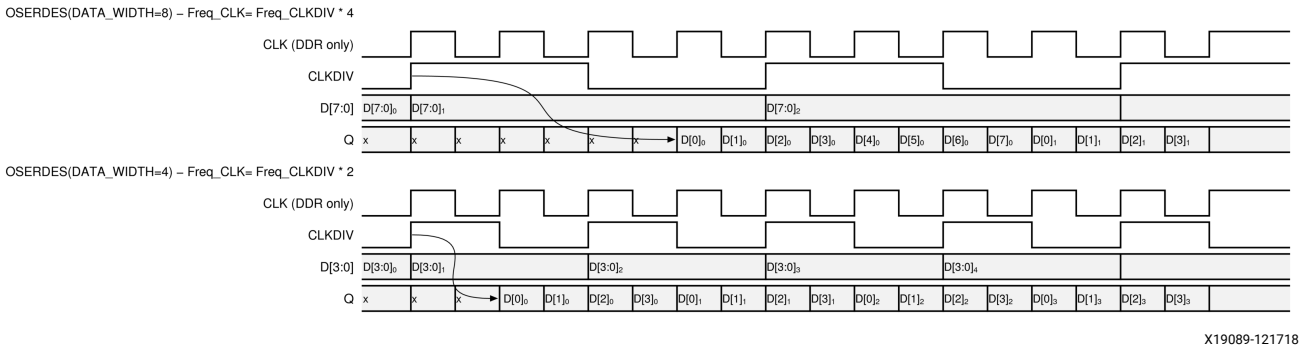
The OSERDESE3 primitive is available to perform output serialization for designs migrating from previous FPGA families or for designs not requiring native mode primitives. The OSERDESE3 in Spartan UltraScale+ devices is a 4- or 8-bit parallel-to-serial converter with specific clocking features to facilitate 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.

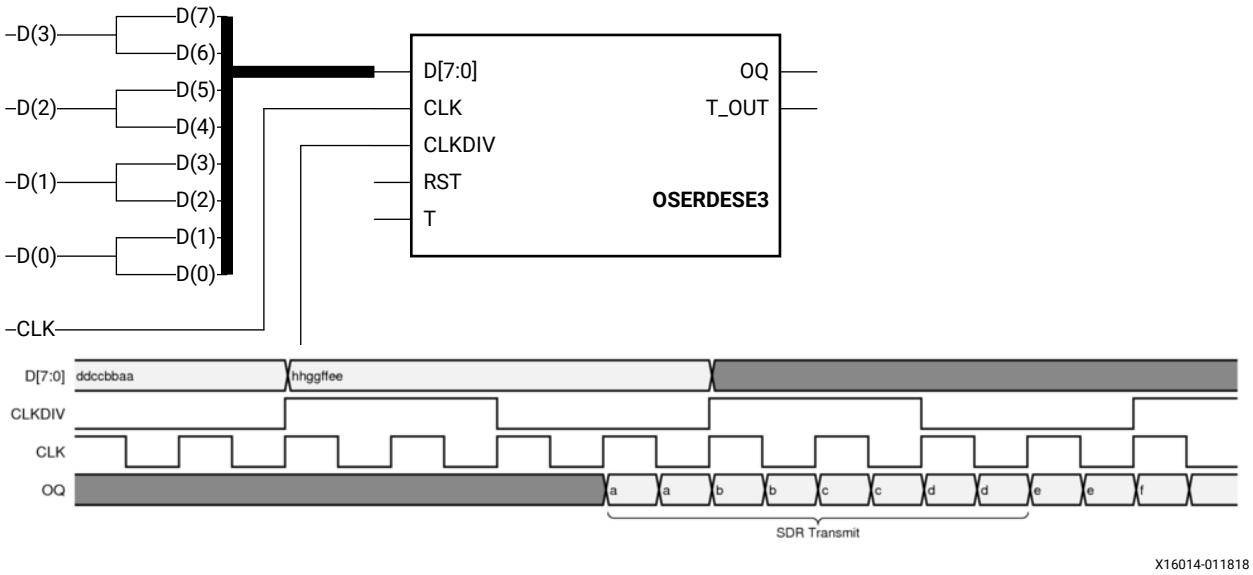
The latency through the OSERDES depends on the DATA_WIDTH setting as shown in the following figure.

Figure 89: **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 the following figure.

Figure 90: **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 the following table.

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

Table 68: **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

The following table lists the OSERDESE3 ports.

Table 69: **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

Table 69: **OSERDESE3 Ports** *(cont'd)*

Port	I/O	Description
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

The following table lists the OSERDESE3 attributes.

Table 70: **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.
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 ODDR with Serialized 3-State. In the ODDRE1 mode, data is connected to D[4,0] and the clock must be connected to CLK. 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_PLUS	ULTRASCALE	String	Device family for behavioral simulation.

IDELAYE3

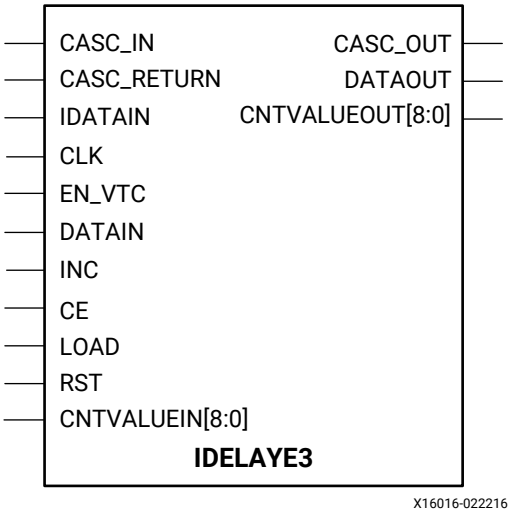
Any input signal except clocks can be delayed using the IDELAYE3 primitive and then either forwarded to the device logic directly, or registered in a simple flip-flop, IDDR, or ISERDESE3, using a single data rate (SDR) clock or a double data rate (DDR) clock inside the input/output interconnect (IOI). 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 *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930). 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 the following figure.

Figure 91: **IDELAYE3 Primitive**



IDELAYE3 Ports

The following table lists the IDELAYE3 ports.

Table 71: **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. ***Note*:** 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.

Table 71: **IDELAYE3 Ports** *(cont'd)*

Port	I/O	Description
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). ***Note*:** 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.
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. • High: Enables IDELAYCTRL to keep delay constant over VT. • Low: VT compensation is disabled. 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

The following table lists the IDELAYE3 attributes.

Table 72: **IDELAYE3 Attributes**

Attribute	Possible Values	Default	Type	Description
DELAY_SRC	DATAIN IDATAIN	IDATAIN	String	For more information, see DELAY_SRC Attribute.

Table 72: **IDELAYE3 Attributes** *(cont'd)*

Attribute	Possible Values	Default	Type	Description
CASCADE	NONE MASTER SLAVE_MIDDLE SLAVE_END	NONE	String	For more information, see CASCADE Attribute.
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 DELAY_TYPE Attribute for further information.
DELAY_VALUE	0–1100 (TIME) 0–511 (COUNT)	0	Decimal	Spartan UltraScale+ devices support up to 1.1 ns. COUNT mode: Desired value in taps.
REFCLK_FREQUENCY	300.00–800.00 (IDELAYCTRL) 300.00–2666.67 BITSLICE_CONTROL)	300.0	1 significant digit float	The REFCLK_FREQUENCY attribute specifies the reference clock of the IDELAYCTRL frequency in MHz. This attribute must 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 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 DELAY_FORMAT Attribute.
UPDATE_MODE	ASYNC SYNC MANUAL	ASYNC	String	For more information, see UPDATE_MODE Attribute.
SIM_DEVICE	ULTRASCALE_PLUS	ULTRASCALE	String	Sets the device version (ULTRASCALE_PLUS)

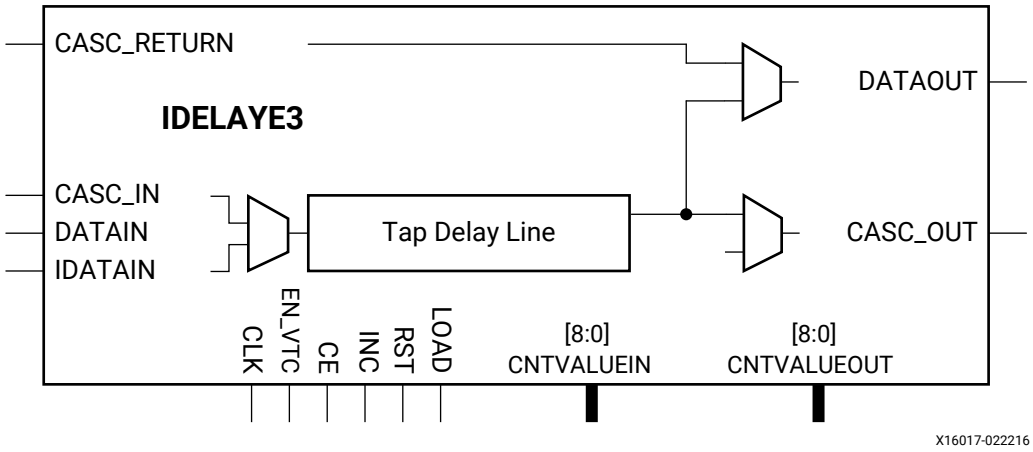
Notes:

1. When in TIME mode, calibration affects the availability of bit slices within the nibble. See HP I/O Bank Overview for more information.

DELAY_SRC Attribute

DELAY_SRC (see the following figure) 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 92: **IDELAYE3 DELAY_SRC Diagram**

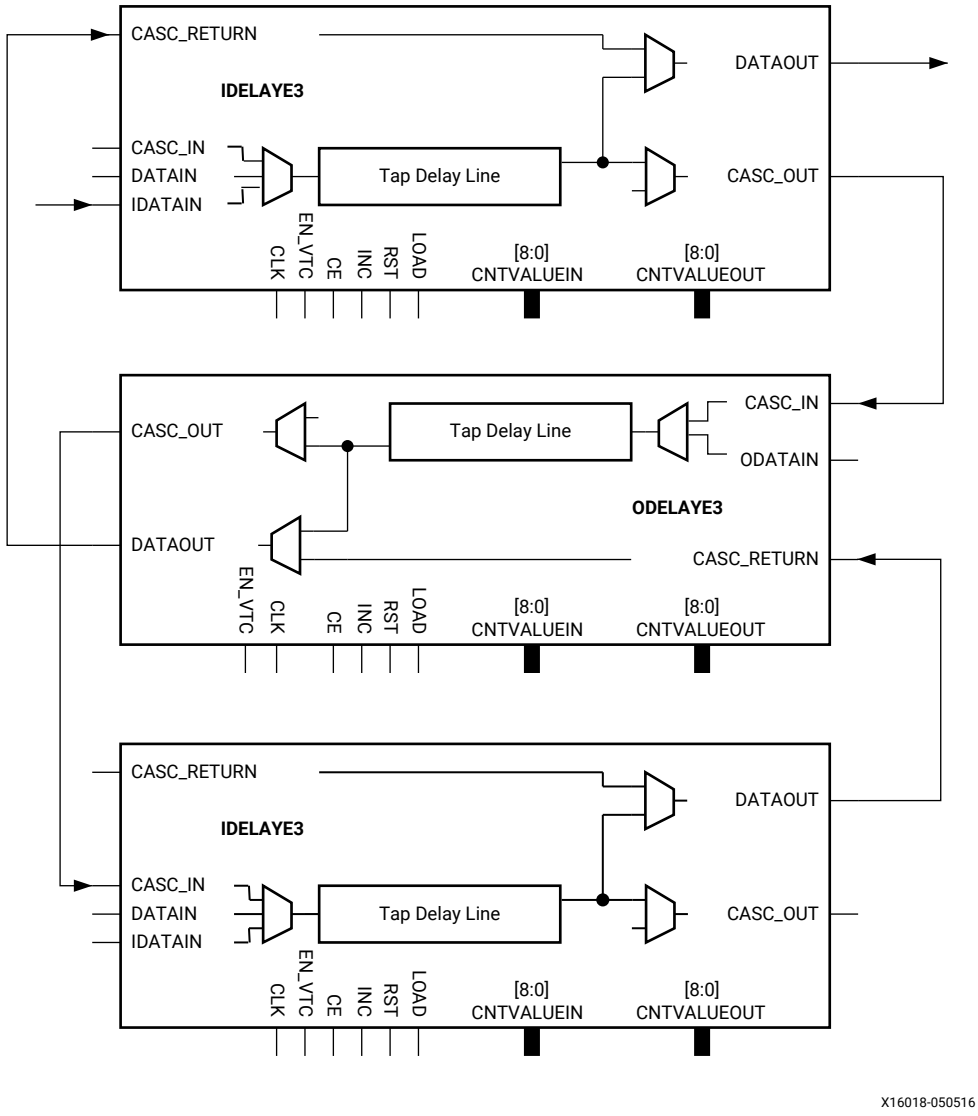


CASCADE Attribute

The CASCADE attribute is set to NONE if the delay line is not cascaded. Cascading is used when a delay greater than 1.10 ns is required. The connections between delay elements are shown in the following figure. 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.

Figure 93: **IDELAYE3 Cascading to Three Slave Delays**



If cascading to achieve a delay > 1.10 ns 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.

DELAY_FORMAT Attribute

The tap size of the IDELAYE3 primitive is defined in the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) as $T_{IDELAY\_RESOLUTION}$.

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 CNTVALUEOU[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 (see the following table). 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 the following figure). 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 the following figure):

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.

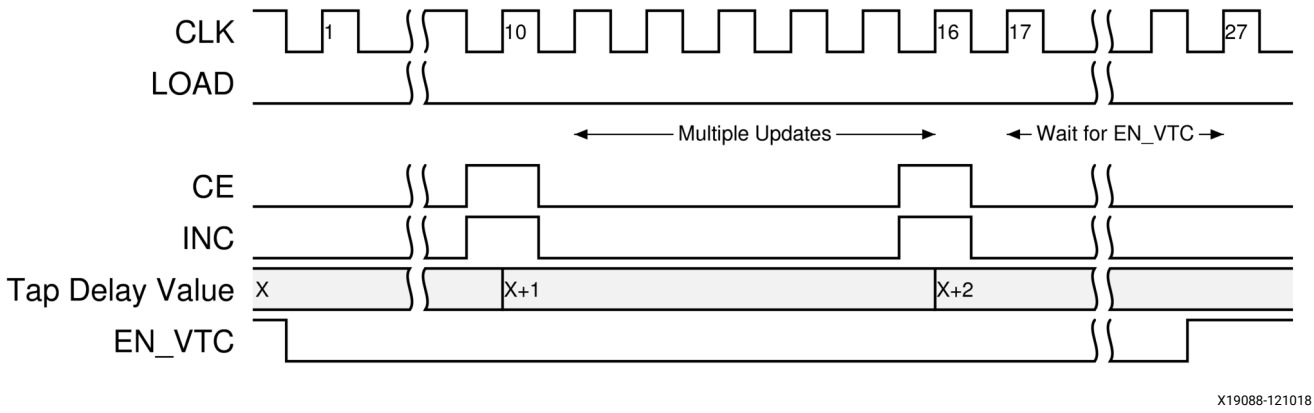
Table 73: **Control Pin when DELAY_TYPE = VARIABLE**[1]

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[2]
0	1	0	1	0	Current value –1 Tap[2]
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 the following figure.

Figure 94: **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.

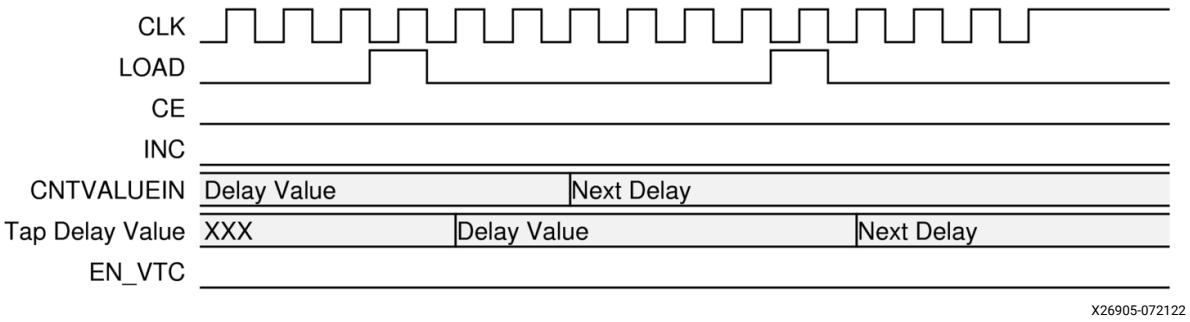
If DELAY_TYPE is VAR_LOAD and DELAY_FORMAT is TIME, the procedure to update the delay line follows (see Figure 97).

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 the following figure):

Figure 95: **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 1100 ps for Spartan UltraScale+ devices.
- The delay range of a single tap is specified in the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930).

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.

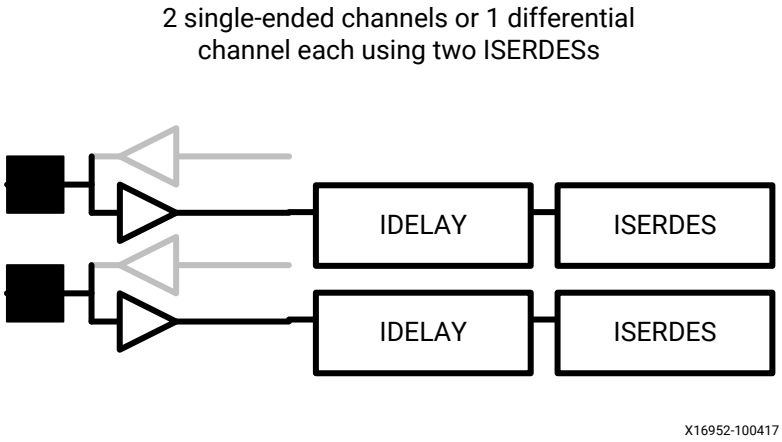
Thus, this is the only parameter a design using COUNT mode must take care of. The value of a measured data, clock, or strobe eye is expressed as an amount of taps without providing the delay this represents. It is thus not necessary to calculate the delay of a single tap, and all 512 provided taps in 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 (see the following figure).

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 (see the following figure).

Figure 96: **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 the following equation:

   $$\text{Total\_Value} - \text{Align\_Delay} = \text{n taps}$$
5. The delay of a single tap is then equal to the following equation:

   $$\text{odd channel DELAY\_VALUE} / \text{n taps} = \text{single tap}$$
6. The new CNTVALUEIN value to write to the delay line or lines used in taps is shown in the following equation for IDELAYs or the second equation for ODELAYs:

CNTVALUEIN <IDELAY> = (wanted delay / single tap) + Align_Delay

CNTVALUEIN <ODELAY> = (wanted delay / single tap)

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 97: **Changing Delay when DELAY_TYPE = VAR_LOAD**

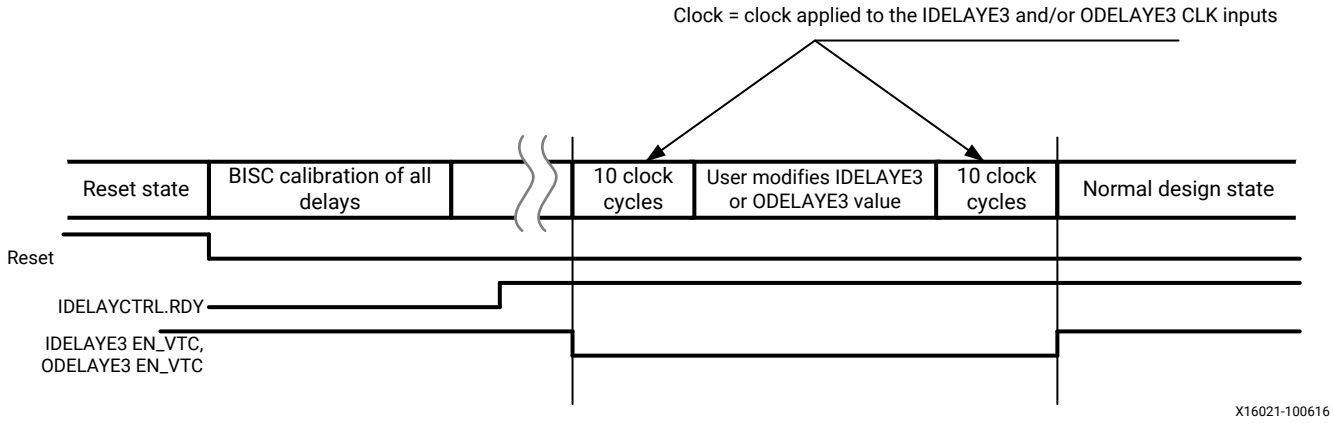


Table 74: **Control Pin when DELAY_TYPE = VAR_LOAD**

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[1]
1	0	1	0	X	Current value – 1	Current value – 1[1]
1	0	0	0	X	No change	No change

Notes:

1. Value depends upon the UPDATE_MODE attribute.

ODELAYE3

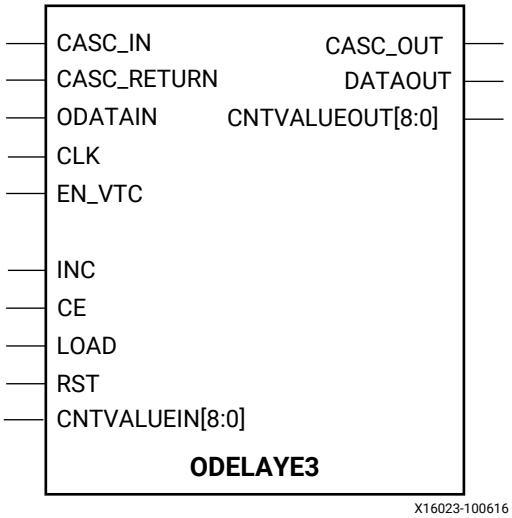
Any output signal can be delayed using the ODELAYE3 primitive, having been either forwarded from the device logic directly or registered in a simple flip-flop or OSERDES using an SDR or DDR clock.

The ODELAYE3 primitive (see the following figure) contains a 512-tap delay line. (See the tap resolution in the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930). 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 98: **ODELAYE3 Primitive**



ODELAYE3 Ports

The following table lists the ODELAYE3 ports.

Table 75: **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. ***Note*:** 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. 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). ***Note*:** 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.
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.

Table 75: **ODELAYE3 Ports** *(cont'd)*

Port	I/O	Description
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. • High: Enables IDELAYCTRL to keep delay constant over VT. • Low: VT compensation is disabled. 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

The following table lists the ODELAYE3 attributes.

Table 76: **ODELAYE3 Attributes**

Attribute	Values	Default	Type	Description
CASCADE	NONE MASTER SLAVE_MIDDLE SLAVE_END	NONE	String	For more information, see CASCADE Attribute.
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 DELAY_TYPE Attribute for further information.
DELAY_VALUE	0–1100 (TIME)	0	Decimal	TIME mode: Desired value in ps. Spartan UltraScale+ devices support up to 1.1 ns. COUNT mode: Desired value in taps. For more information, see DELAY_VALUE Attribute.

Table 76: **ODELAYE3 Attributes** *(cont'd)*

Attribute	Values	Default	Type	Description
REFCLK_FREQUENCY	300.00–800.00 (IDELAYCTRL) 300.00–2666.67 (BITSLICE_CONTROL)	300.0	1 significant digit float	The REFCLK_FREQUENCY attribute specifies the reference clock of the IDELAYCTRL frequency in MHz. This attribute must 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[1] 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 DELAY_FORMAT Attribute.
UPDATE_MODE	ASYNC SYNC MANUAL	ASYNC	String	For more information, see UPDATE_MODE Attribute.
SIM_DEVICE	ULTRASCALE_PLUS	ULTRASCALE	String	Sets the device version (ULTRASCALE_PLUS)

Notes:

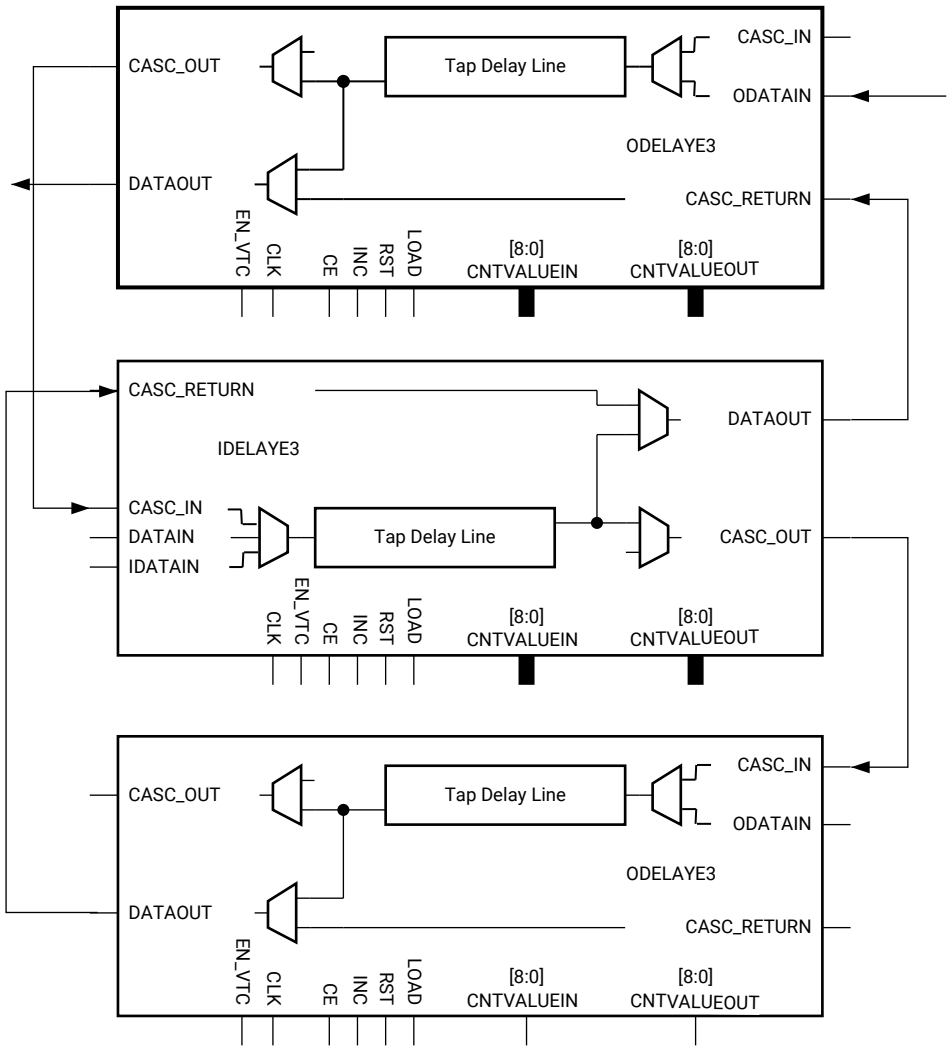
1. When in TIME mode, calibration affects the availability of bit slices within the nibble. See HP I/O Bank Overview for more information.

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 the following figure. 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.

Figure 99: **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) . 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_VALUE Attribute.

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 73). 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 94). The EN_VTC pin should be held Low during the delay change command to ensure that any automatic adjustments are stopped.

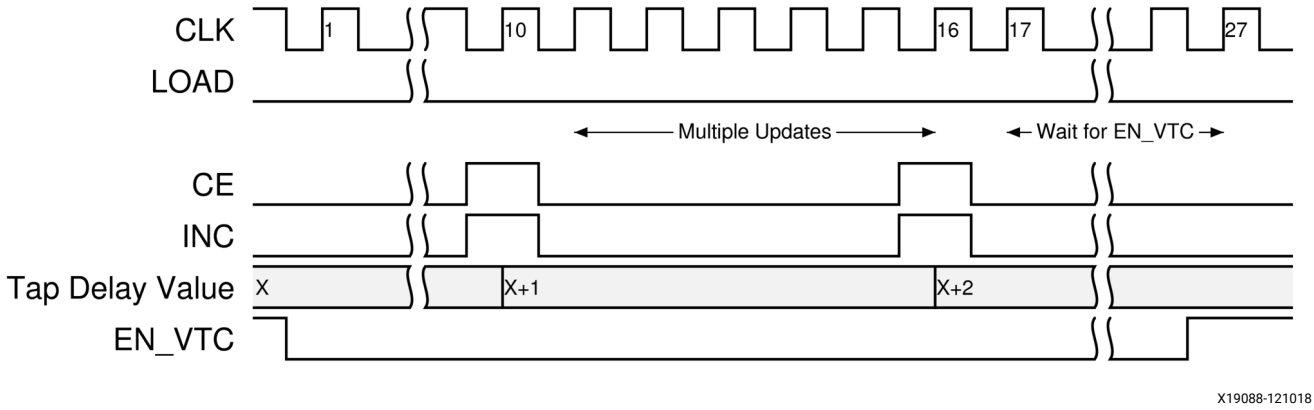
To increment/decrement delay lines when using TIME mode, use the following procedure (see Figure 89):

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 section.

Figure 100: **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 97):

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. 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 eight 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

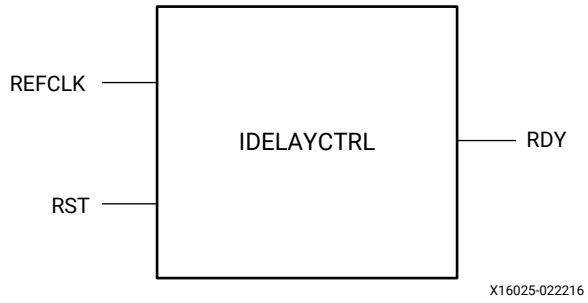
If the IDELAYE3 (or ODELAYE3) primitives are instantiated, the IDELAYCTRL module must be instantiated, except when the DELAY_FORMAT is set to COUNT or when mixing component and native mode in native mode designs (see Mixing Native and Non-Native Mode I/O in a Nibble). There is one IDELAYCTRL module per 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. The following 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 IDLEAYCTRL.RDY pin is asserted High.*

Figure 101: **IDELAYCTRL Module**



IDELAYCTRL Ports

The following table lists the IDELAYCTRL ports.

Table 77: **IDELAYCTRL Ports**

Port	I/O	Type	Description
REFCLK	Input	Clock	Reference clock for delay calibration.
RST	Input	Reset	Active-High asynchronous reset for IDELAYCTRL. ***Note***: 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 78: **IDELAYCTRL Attribute**

Attribute	Values	Default	Type	Description
SIM_DEVICE	7SERIES, ULTRASCALE	ULTRASCALE	String	Set to ULTRASCALE for Spartan 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 in the references section.

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 the previous step is 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.

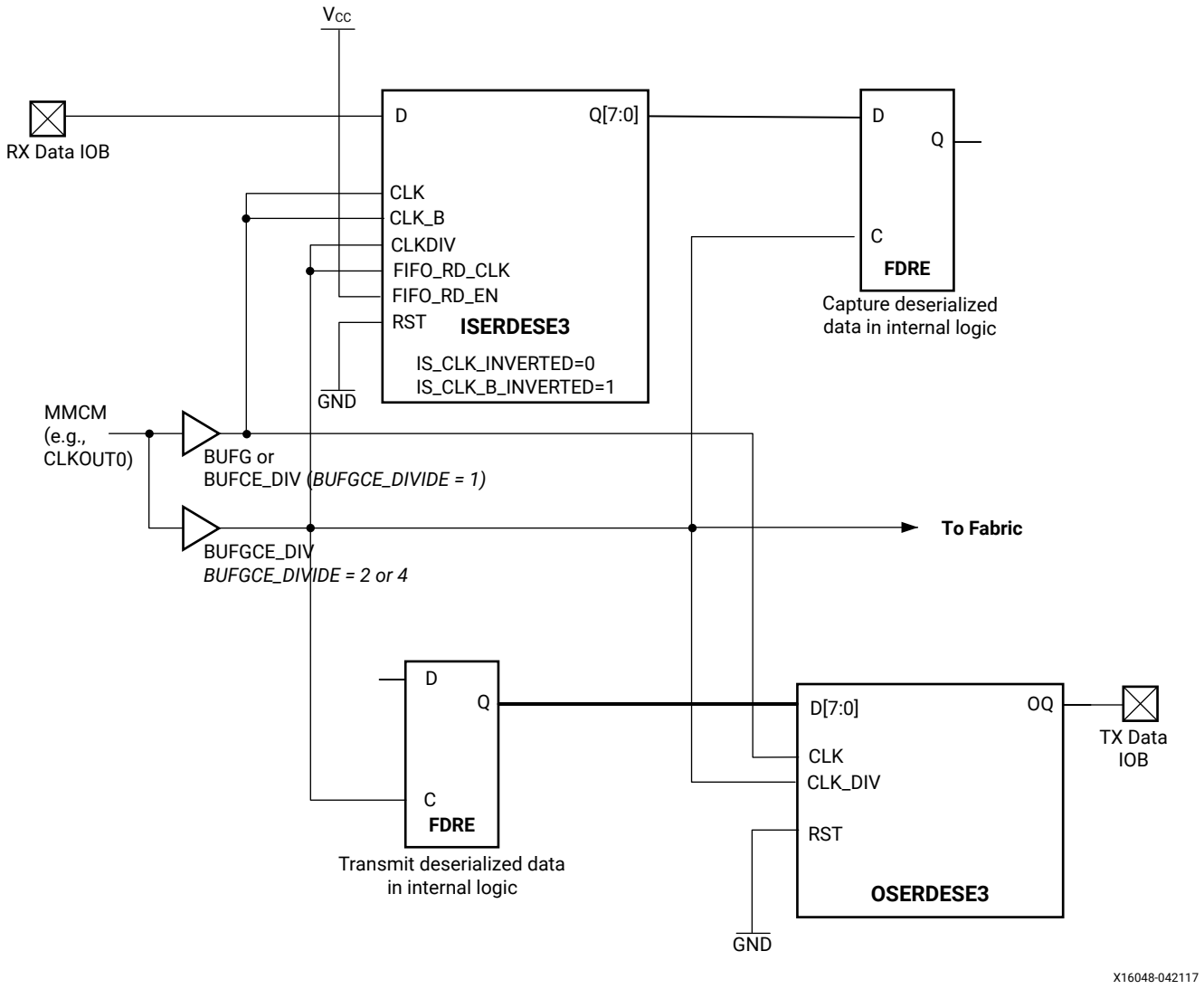
Clocking Considerations Using Component Primitives

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 the following figure.

Figure 102: **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 the previous figure 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 the previous figure. 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 the previous figure, 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 the previous figure, 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 the following figure. 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 103: **MMCM with Separate Clock Outputs**

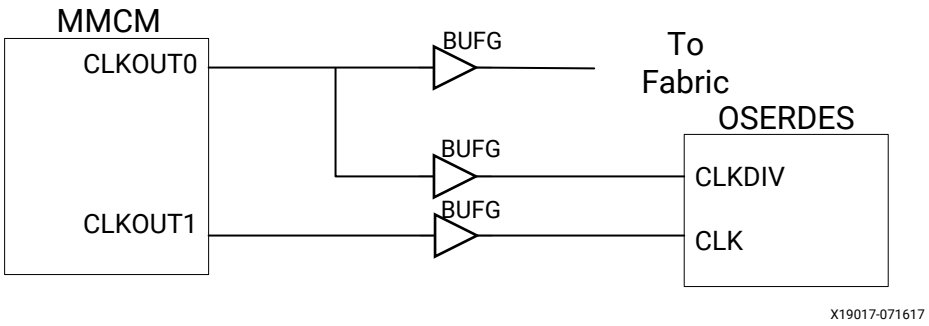
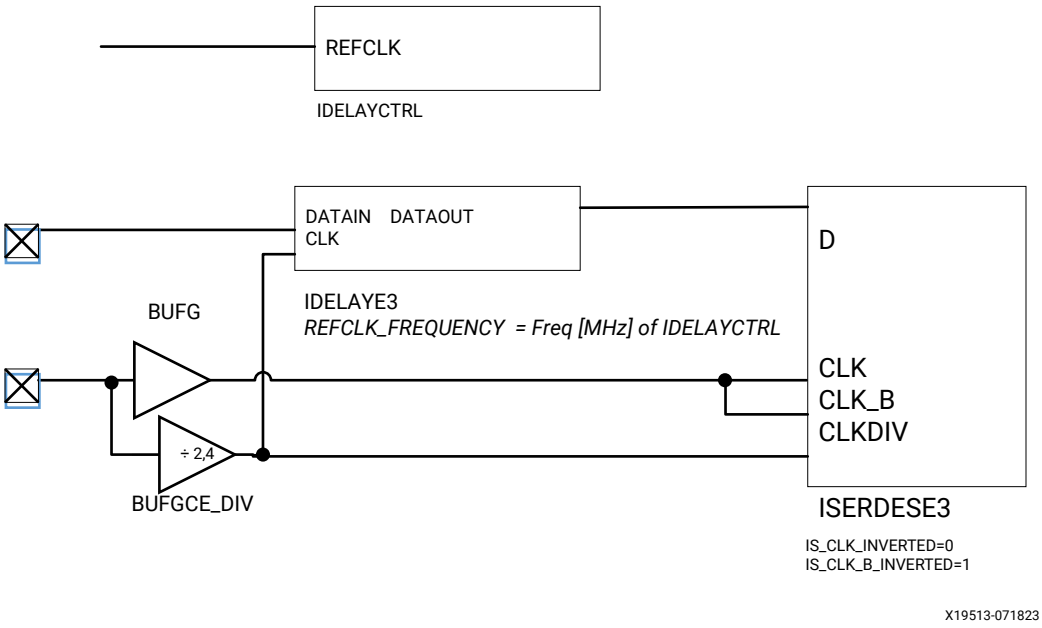


Figure 104: **Clocking Connections for ISERDES with IDELAY (TIME Mode)**



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 the previous figure. 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.

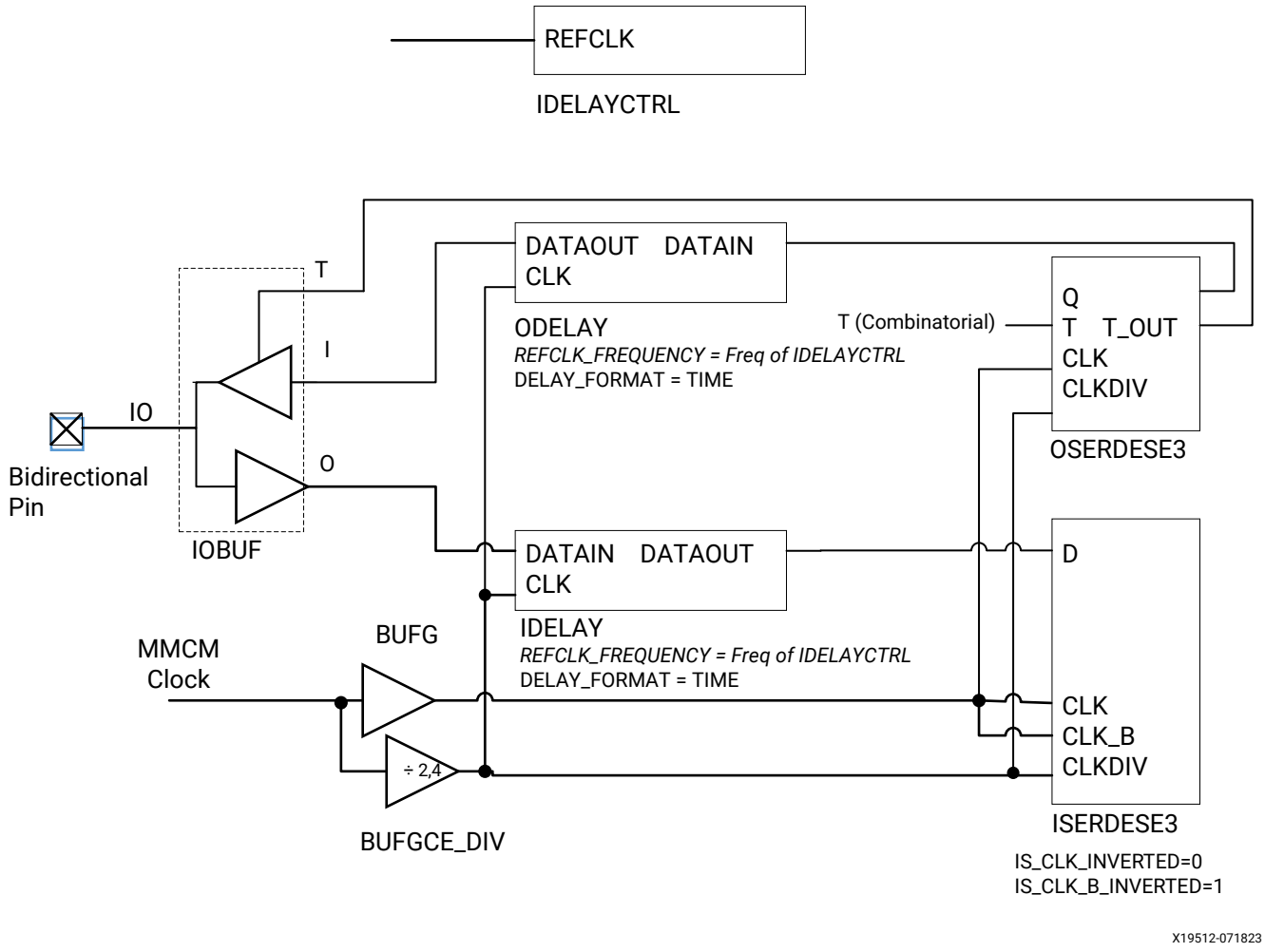
Bidirectional Signaling Using Component Mode

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. 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, the following figure shows an example bidirectional pin with IDELAY and ODELAY. The OSERDES only supports combinatorial 3-state controls for the T input of the IOBUF.

Figure 105: **Bidirectional Signaling Using IDELAY/ODELAY with ISERSDES 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. The following table lists the recommended bidirectional logic. When unsupported, the Vivado tool can borrow resources from additional I/Os.

Table 79: **Bidirectional Support by I/O Bank**

Input Datapath	Output Datapath	Tristate Control	I/O Banks
Fabric Logic	Fabric Logic	Fabric Logic	HDIO/HPIO
IFD	OFD	Fabric Logic	HDIO/HPIO
IFD	OFD	OFD	HDIO
IDDRE1	ODDRE1	Fabric Logic	HPIO
IDDE1	ODDRE1	ODDRE1	HPIO
ISERDES3	OSERDESE3	Fabric Logic	HPIO
ISERDES3	OSERDESE3	OSERDESE3	Not supported

Mixing Native and Non-Native Mode I/O in a Nibble

As described in RXTX_BITSLICE, 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]
```

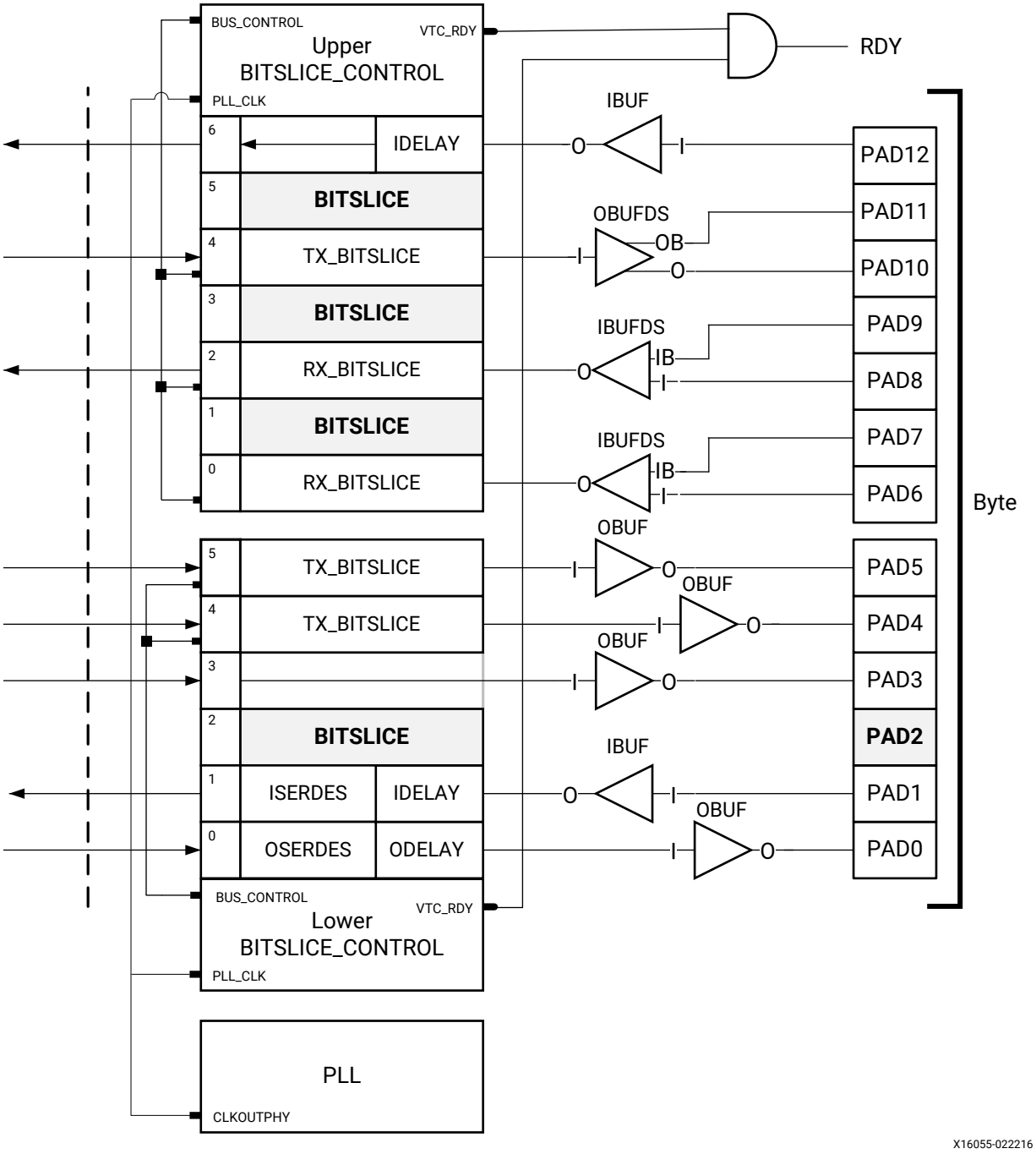
Each nibble requires an IODELAY_GROUP, which corresponds to a BITSLICE_CONTROL or IDELAYCTRL within that nibble. Because each nibble only contains a single IDELAYCTRL or BITSLICE_CONTROL, each nibble can only contain IDELAYs/ODELAYs from a single IODELAY_GROUP.

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 the following figure.

Figure 106: **Example Mixed-Mode Byte**



Example XDC constraints for mixed nibbles depicted in the previous figure 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 the previous figure:

- 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 SelectIObank 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

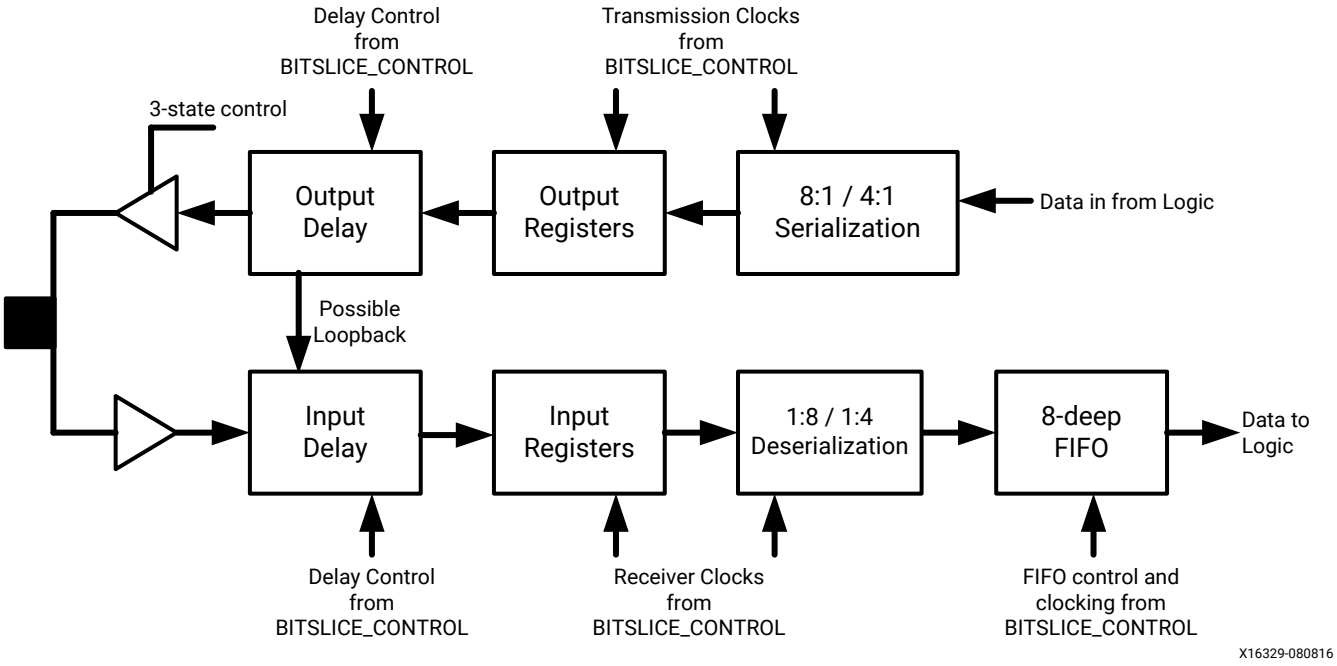
The Native primitives are the fundamental structures from which Component primitives are created. The Component primitives use specific settings of the Native primitives to provide the same functionality from previous FPGA families. With native primitives you can construct component interfaces that run 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 the following figure.

Figure 107: **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 $T_{ODELAY\_RESOLUTION}$. 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 (see Extended Delay Control Signals).

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.

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.

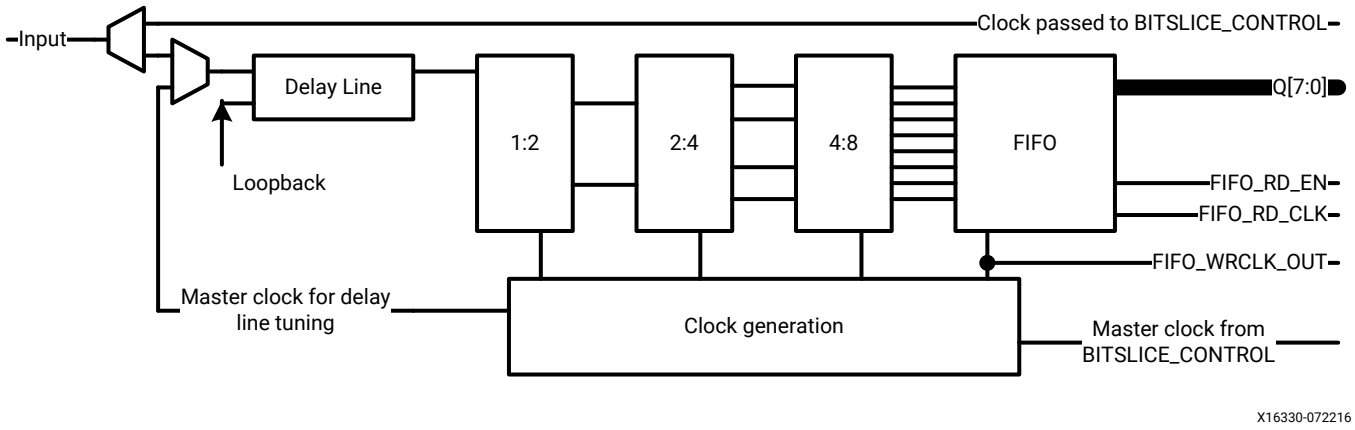
The FIFO writes the 4-bit or 8-bit deserialized data on the rising edge of FIFO_WR_CLK. The FIFO can be read after interpretation of some FIFO status signals from the FPGA logic side. In this way, the FIFO performs the role of clock domain crossing element. See more about the FIFO in FIFO Function.

RXTX_BITSLICE Receiver Function

Receiver Setup

As shown in the following figure, 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.

Figure 108: **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 the following equation:

  $$\text{forwarded clock} / 4 = 156.25\ \text{MHz}$$
- The receiver used in 4-bits requires a FIFO_RD_CLK shown in the following equation:

  $$\text{forwarded clock} / 2 = 312.5\ \text{MHz}$$

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 the following equation:

  $$\text{DDR (PLL/MMCM) clock} / 4 = 156.25\ \text{MHz}$$
- The receiver used in 4-bits requires a FIFO_RD_CLK shown in the following equation:

  $$\text{DDR (PLL/MMCM) clock} / 2 = 312.5\ \text{MHz}$$

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 73). 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 109). 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:

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.

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 80: **RXTX_BITSLICE Control Pin when DELAY_TYPE = VARIABLE**[1]

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

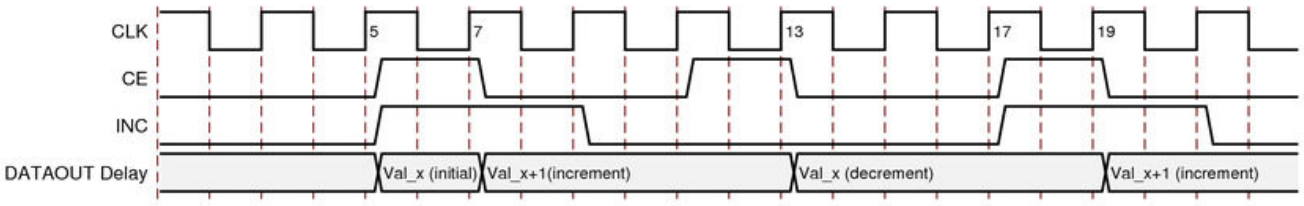
Table 80: **RXTX_BITSLICE Control Pin when DELAY_TYPE = VARIABLE[1]** *(cont'd)*

EN_VTC	CLK	LOAD	CE	INC	Tap Setting
0	1	0	1	1	Current value + 1 tap[2]
0	1	0	1	0	Current value – 1 tap[2]
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 the following figure.

Figure 109: **Variable Mode, UPDATE_MODE = ASYNC**



Notes on the preceding figure:

- 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_x+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_x+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 97):

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, 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.

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 *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930).
- 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 is 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.

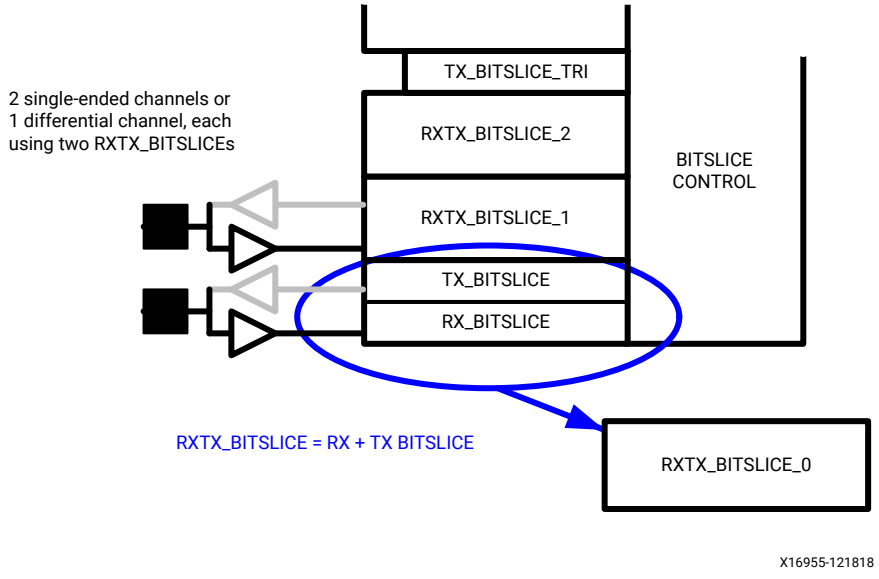
When DELAY_TYPE is VAR_LOAD and DELAY_FORMAT is COUNT, a delay line is used in bare-metal mode because only the depth or amount of taps of the delay line are important (512 in the case of Spartan UltraScale+ devices).

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 (see the following figure).
- 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 (see the following figure).

Figure 110: **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 the following equation:

   $$\text{Total\_Value} - \text{Align\_Delay} = \text{n taps}$$

6. The delay of a single tap is then equal to the following equation:

   $$\text{odd channel DELAY\_VALUE} / \text{n taps} = \text{single tap}$$

7. The new CNTVALUEIN value to write to the delay line or lines used in taps is shown in the following equation for RX_BITSLICEs or the next equation for TX_BITSLICEs:

   $$\text{CNTVALUEIN <RX\_BITSLICE>} = (\text{wanted delay} / \text{single tap}) + \text{Align\_Delay}$$

   $$\text{CNTVALUEIN <TX\_BITSLICE>} = (\text{wanted delay} / \text{single tap})$$

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 (see the following equation):

   $$\text{DELAY\_VALUE/OUTPUT\_DELAY\_CNT} = \text{single tap [ps/tap]}$$

4. The number of taps required for the Align_Delay is shown in the following equation:

   $$\text{INPUT\_DELAY\_CNT} - \text{OUTPUT\_DELAY\_CNT} = \text{Align\_Delay [\# taps]}$$

5. CNTVALUEIN can now be set as needed (the following equation for RX_BITSLICEs or the next equation or TX_BITSLICEs):

   $$\text{CNTVALUEIN <RX\_BITSLICE>} = (\text{wanted delay/single tap}) + \text{Align\_Delay}$$

   $$\text{CNTVALUEIN <TX\_BITSLICE>} = (\text{wanted delay/single tap})$$

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 82 and Table 83.

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.

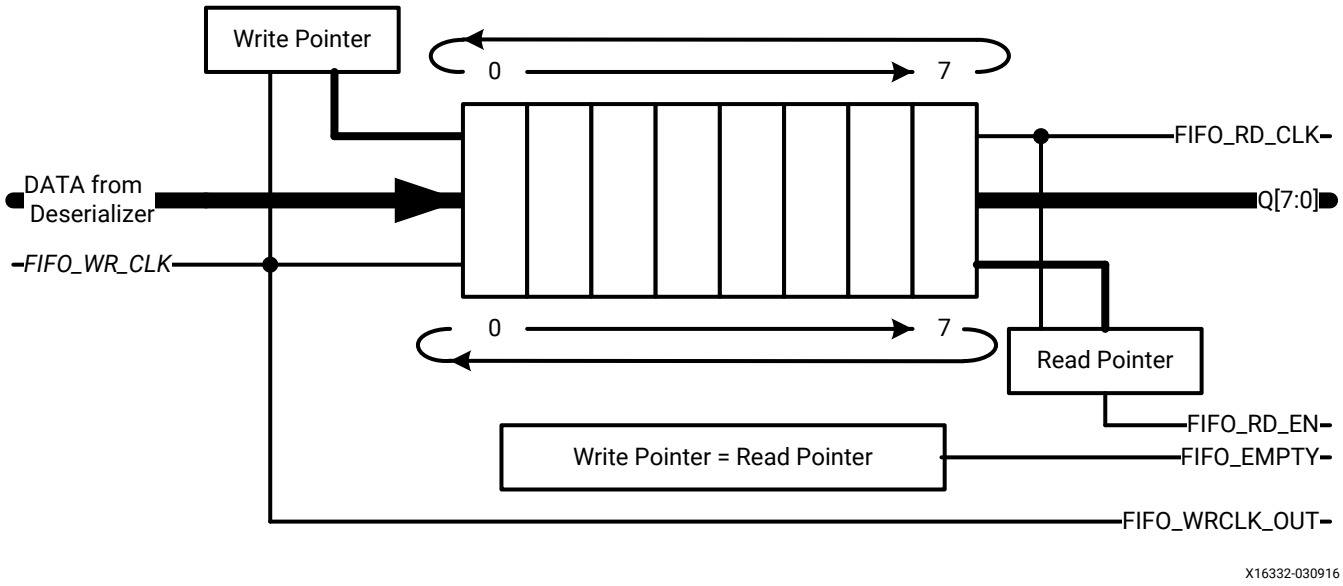
Set FIFO_SYNC_MODE to FALSE to use the FIFO as a clock domain crossing element. This mode allows data captured from the interface clock domain to successfully cross over to the interconnect logic clock domain(s). The typical latency through the FIFO is equal to two clock cycles (three when counting the extra FIFO enable cycles), but depending on the design, the latency can also be eight read clock cycles. The following table describes the behavior of the clock and control inputs of the FIFO.

Table 81: **Behavior of the Clock and Control Inputs of the FIFO**

Clock or Control Input	Behavior
FIFO_WRCLK_OUT	This clock can be used as a read clock for the FIFO. Clocks to capture the data are generated inside the BITSLICE_CONTROL and RXTX_BITSLICE primitives. One of these internally generated clocks, the divided sample clock, is the clock used to write data into the FIFO (call it FIFO_WR_CLK). A copy of this clock is provided as FIFO_WRCLK_OUT output of the bit slice. Each bit slice has a FIFO_WRCLK_OUT output pin but only bit slices in nibble position zero can route and use this clock.
FIFO_RD_CLK	This is the clock used to pull data out of the FIFO. It needs to have the same frequency as the FIFO_WR_CLK, but there does not need to be a phase relationship between FIFO_RD_CLK and FIFO_WRCLK_OUT clocks. The FIFO read clock can be supplied by a FIFO_WRCLK_OUT of a BITSLICE_0 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 (BUFG, BUFGCE, or other).
FIFO_RD_EN	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 FIFO_RD_CLK cycles. This is because the FIFO_RD_EN 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 FIFO_EMPTY status is generated.
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 the previous table is shown in the following figure.

Figure 111: **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 the following figure, 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 the following figure).

Notes:

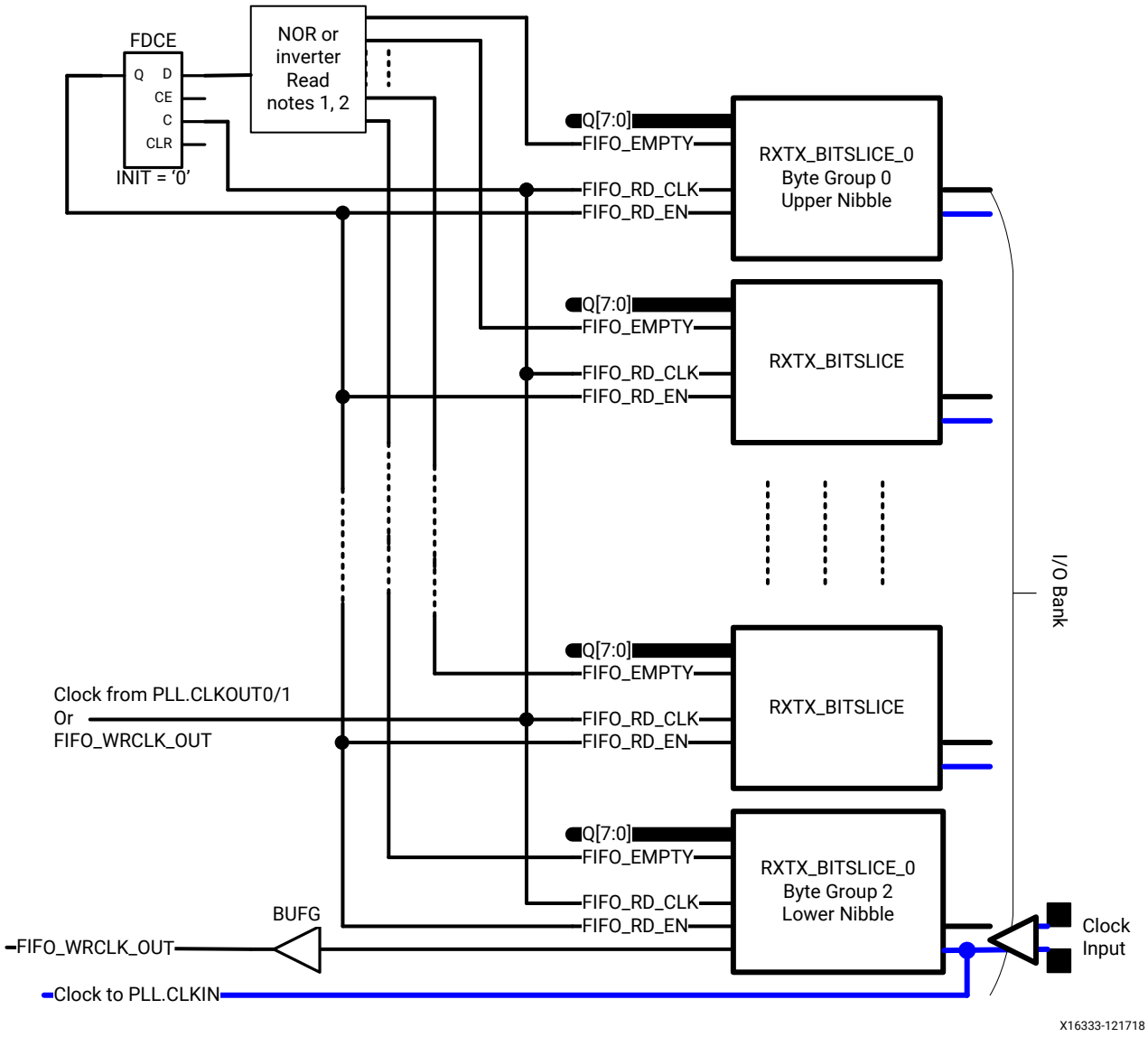
- For static timing purposes, a generated clock should be specified as part of the timing constraint when using FIFO_WRCLK_OUT. As an example, assume a DATA_WIDTH = 4 (RX_BITSLICE) and a sample clock of 500 MHz on a port named rx_clk_in. Additionally, assume the instance rx_clock_bitslice_inst (at nibble 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.

Figure 112: **FIFO as Bit Slice Output Synchronizer (Multiple Clocks or Strobes)**



- **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 BITSLICES in the path to the flip-flop. The optional flip-flop helps close timing.

As shown in the following figures, 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.

Figure 113: **RX_BITSLICE FIFO, DATA_WIDTH = 8**

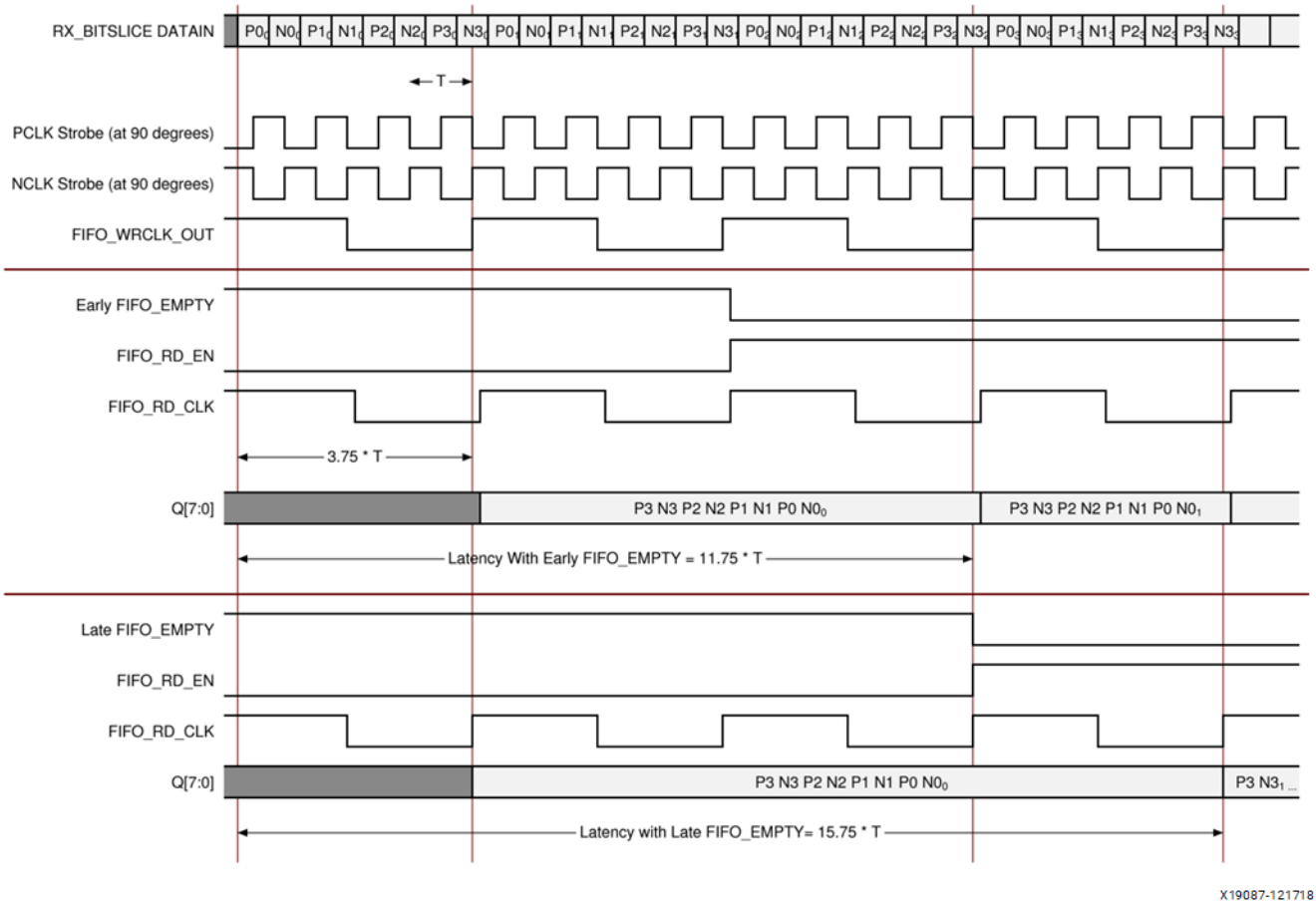
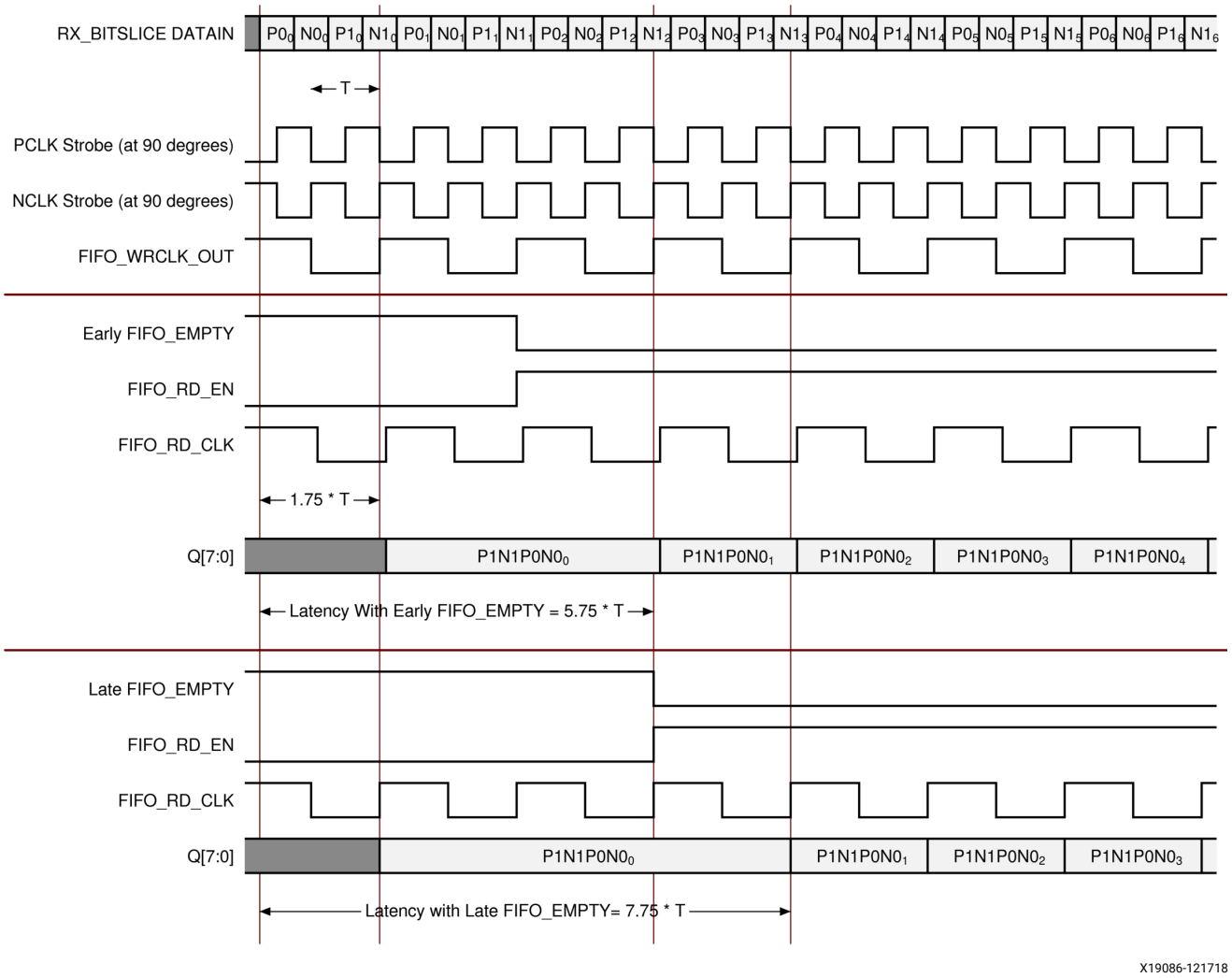


Figure 114: **RX_BITSLICE FIFO, DATA_WIDTH = 4**



RXTX_BITSLICE Transmitter Function

Transmitter Setup

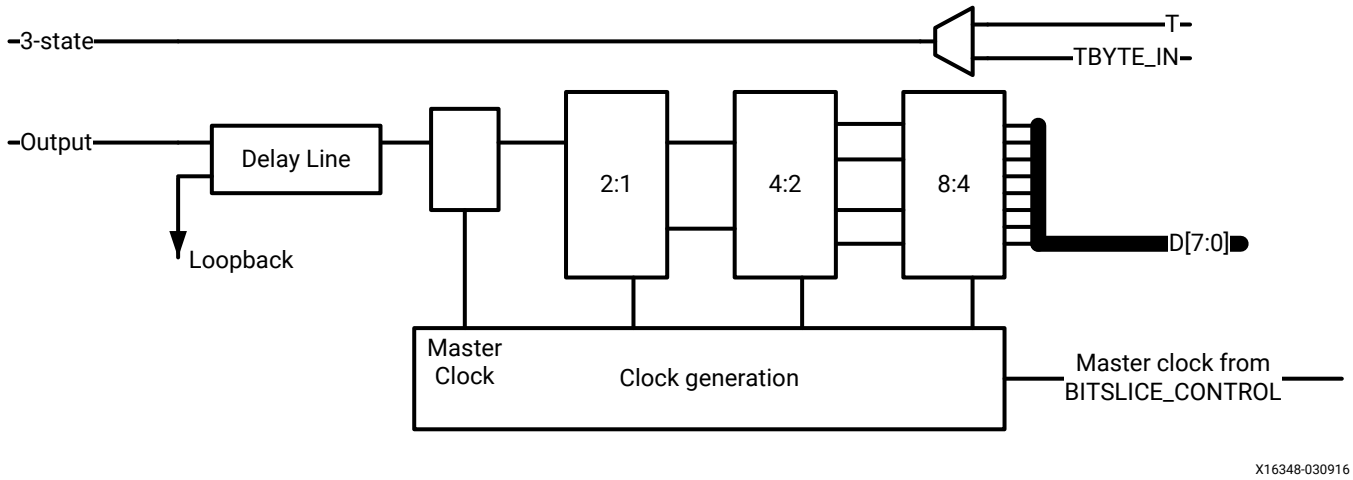
The transmitter in the RXTX_BITSLICE has an 8-bit input parallel register. In 4-bit mode, only bits [3:0] are used to store the data from the logic. Capturing the parallel data and the clocking of registers in the RXTX_BITSLICE happens on internally generated clocks. To create these clocks, the transmitter side of the RXTX_BITSLICE uses the high-speed PLL generated master clock (PLL_CLK). Follow the Native Mode Bring-up and Reset procedure described for the BITSLICE_CONTROL primitive.

The 8-bit input register multiplexes down to a registered 4-bit and then to a registered 2-bit value. Those two bits of data pass through a multiplexer and register into the output delay line. The output delay line connects to an output buffer in the IOB (see the following figure).

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.

Figure 115: **RXTX_BITSLICE Transmitter Block Diagram**



Both 3-state possibilities are passing through the transmitter (see the preceding figure). 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 the following equation for 8-bit (see the following figure):

  $$T + (13/16)T = \text{latency}$$

  where T is the period of the parallel load or interconnect logic clock, and shown in the following equation for 4-bit (see the figure next to the following figure):

  $$T + (5/8)T = \text{latency}$$

- With OUTPUT_PHASE_90 = TRUE, the latency from loading eight parallel bits to a first serial output bit is shown in the following equation for 8-bit (see the following figure):

  $$T + (14/16)T = \text{latency}$$

  where T is the period of the parallel load or interconnect logic clock, and shown in the following equation for 4-bit (see the figure next to the following figure):

  $$1T + (6/8)T = \text{latency}$$

  where T is the period of the parallel load or FPGA logic clock.

Figure 116: **TX_BITSLICE Latency, DATA_WIDTH = 8**

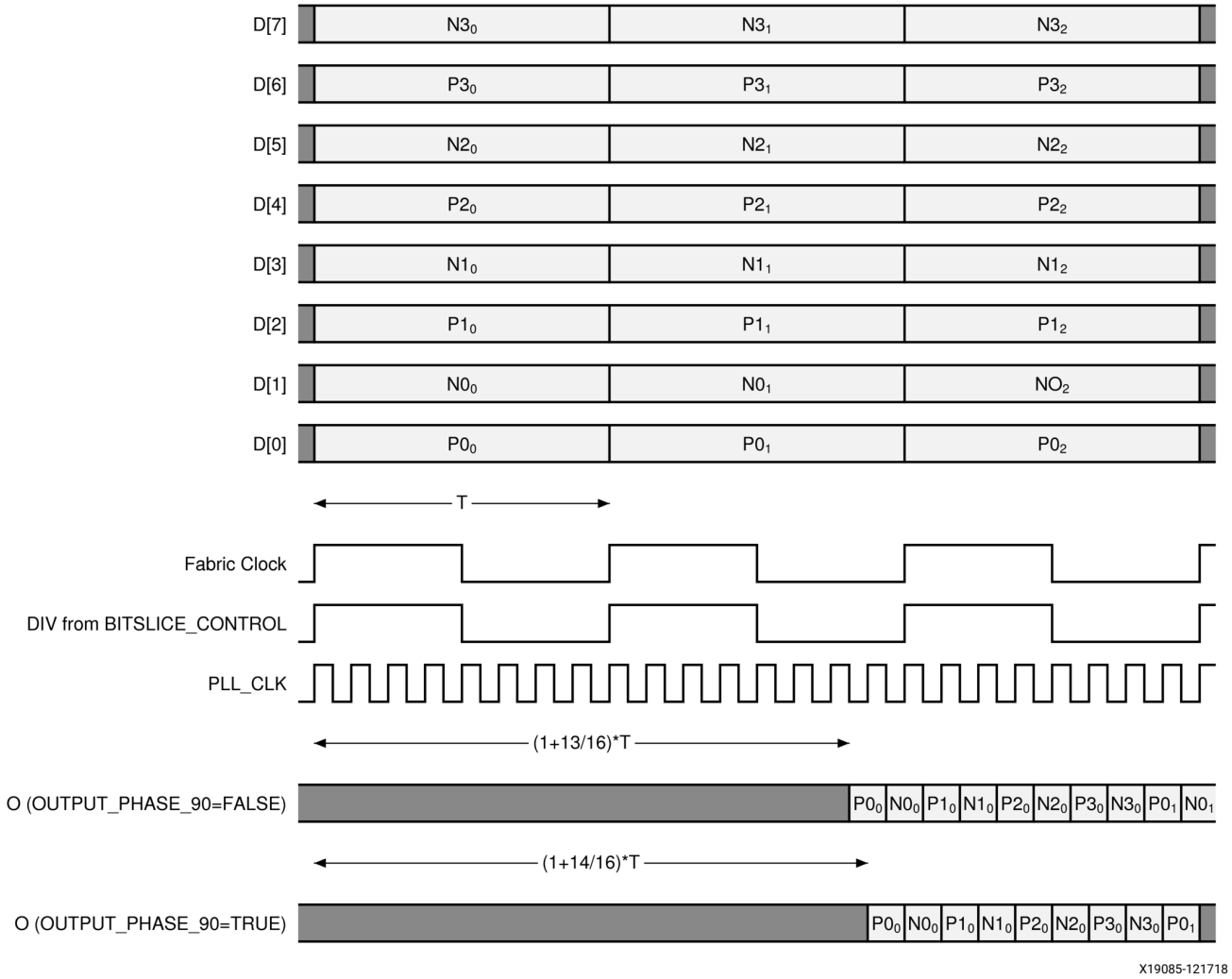
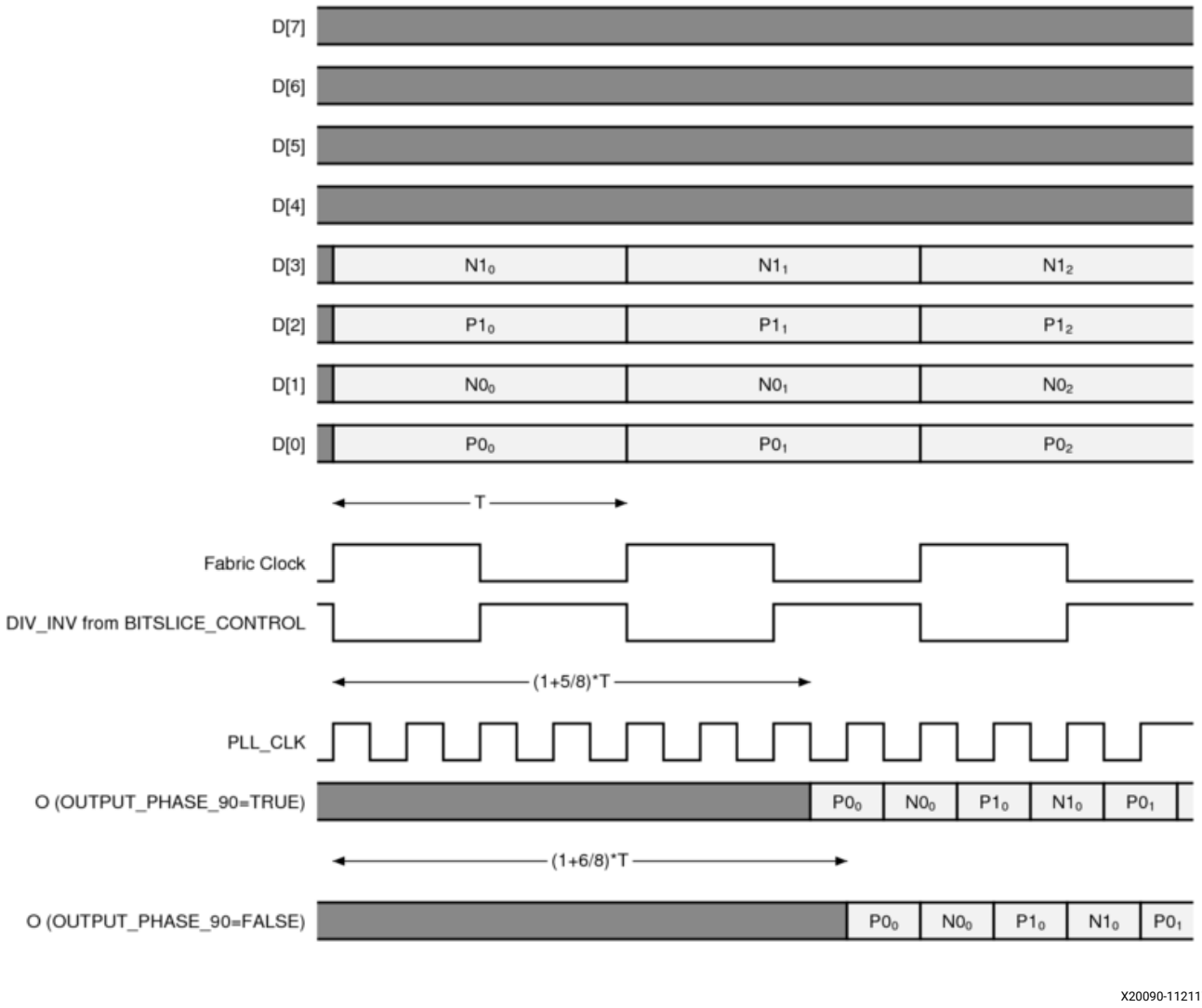


Figure 117: **TX_BITSLICE Latency, DATA_WIDTH = 4**



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 73). 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 the following figure).

The RXTX_BITSLICE.EN_VTC pin should be held Low during the delay change command to ensure that any automatic adjustments are stopped.

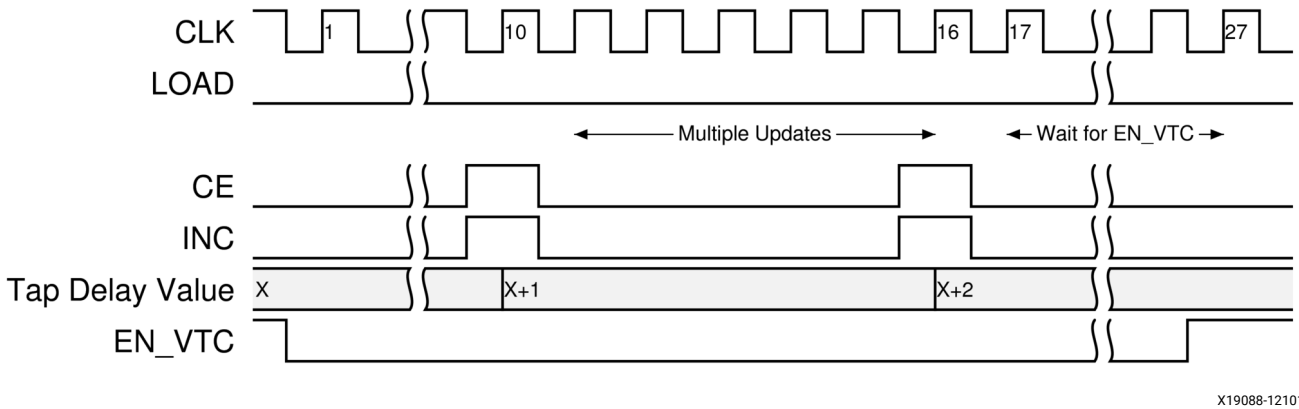
***Note*:** The input and output delays for each bitslice share a delay ratio register (see Table 115). The delay ratio register ensures the input delays are correctly calibrated based on the reference clock frequency. Output delays must match input delays for optimal calibration accuracy for the output 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.

Figure 118: **Variable Mode, UPDATE_MODE = ASYNC**

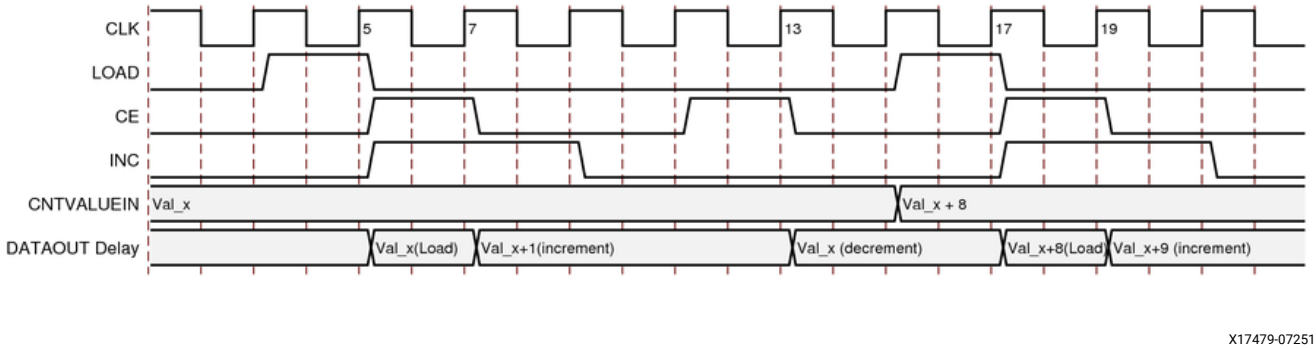


Notes on the preceding figure:

- 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_x+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_x+1.

VAR_LOAD Mode

Figure 119: **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 (see Figure 97):

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

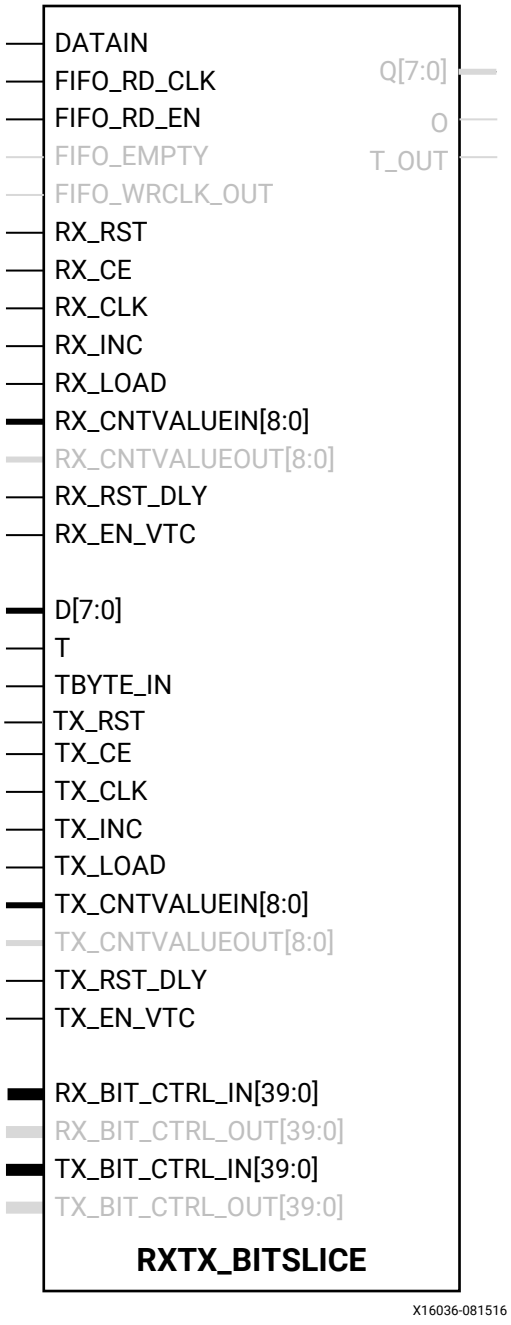
    a. Calculate the difference (Dif_Val) between New_Val and Org_Val and the direction to step.
    b. Make the INC input High or Low to increment or decrement the delay line.
    c. Toggle the CE pin to execute the increment or decrement.
    d. Decrement the Dif_Val and check if it is zero.
        - 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 the following figure. In this figure, black represents inputs and gray represents outputs. The following figure lists the RXTX_BITSLICE ports.

Figure 120: **RXTX_BITSLICE Primitive**



The following table lists the RXTX_BITSLICE ports.

Table 82: **RXTX_BITSLICE Ports**

Port	Function[1]	I/O	Synchronous Clock Domain	Description
DATAIN	I/O RX	Input	Asynchronous	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. The incoming signal from the IOB can be data, clock, or strobe, selected by the DATA_TYPE attribute on the RX_BITSLICE. 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 Clocking in Native Mode in the BITSLICE_CONTROL section for more information. When the incoming signal from the IOB is data only, it can be located at any bit slice position in the nibble.
Q[7:0]	RX FPGA	Output	FIFO_RD_CLK	Deserialized (parallel) output data from the RX FIFO goes to the interconnect logic. 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. ***Note***: 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. If DATA_WIDTH = 8, Q[7:0] represent 8 bits of captured serial data, and route through from DATAIN to Q5 is not available.
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 Native Mode Bring-up and Reset for more information.

Table 82: **RXTX_BITSLICE Ports** *(cont'd)*

Port	Function[1]	I/O	Synchronous Clock Domain	Description
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.
RX_LOAD	RX FPGA	Input	RX_CLK	When in VAR_LOAD mode and RX_UPDATE_MODE=ASYNC, 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.

Table 82: **RXTX_BITSLICE Ports** *(cont'd)*

Port	Function[1]	I/O	Synchronous Clock Domain	Description
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_CNTVALUIN 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.
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 FIFO Function 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 FIFO Function.

Table 82: **RXTX_BITSLICE Ports** *(cont'd)*

Port	Function[1]	I/O	Synchronous Clock Domain	Description
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. ***Note***: The use of this port is only recommended for experienced designers. Additional timing constraints are described in FIFO Function.
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.
T	TX FPGA	Input	Asynchronous	T assigns a combinatorial path through the TX_BITSLICE to the 3-state pin of an output buffer. 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. Each TX_BITSLICE in a nibble has a T input, meaning that there are 13 T inputs for a byte (byte = two nibbles)
TBYTE_IN	TX FPGA	Input	PLL_CLK (BITSLICE_CONTROL)	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.

Table 82: **RXTX_BITSLICE Ports** *(cont'd)*

Port	Function[1]	I/O	Synchronous Clock Domain	Description
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). 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.
TX_RST	TX FPGA	Input	Asynchronous	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. For deterministic bring-up, follow the steps in Native Mode Bring-up and Reset.
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 82: **RXTX_BITSLICE Ports** *(cont'd)*

Port	Function[1]	I/O	Synchronous Clock Domain	Description
TX_INC	TX FPGA	Input	TX_CLK	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. 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.
TX_LOAD	TX FPGA	Input	TX_CLK	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.
TX_EN_VTC	TX FPGA	Input	Asynchronous	Enable voltage, temperature, and process 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 TX_EN_VTC signal must be pulled High during initial BISC. When used in COUNT mode, the TX_EN_VTC signal must be pulled Low. 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.

Table 82: **RXTX_BITSLICE Ports** *(cont'd)*

Port	Function[1]	I/O	Synchronous Clock Domain	Description
TX_CNTVALUEIN[8:0]	TX FPGA	Input	TX_CLK	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. ***Note*:** 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.
TX_CNTVALUEOUT[8:0]	TX FPGA	Output	TX_CLK	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.
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 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 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:

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 FPGA logic.
   TX FPGA: Connections from/to the TX_BITSLICE side of the RXTX_BITSLICE and the FPGA logic.

RXTX_BITSLICE Attributes

The following table lists the RXTX_BITSLICE attributes.

Table 83: **RXTX_BITSLICE Attributes**

<table>
<tr><th>Attributes</th><th>Values</th><th>Default</th><th>Type</th><th>Description</th></tr>
<tr><td>RX_DATA_TYPE</td><td>DATA
DATA_AND_CLOCK SERIAL</td><td>DATA</td><td>String</td><td>Attribute defining the type of signal BITSLICE is receiving (DATA, DATA_AND_CLOCK, or SERIAL) and the capture clock to be used.
SERIAL = When received data must be captured by an unrelated clock (for example SGMII).
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.
DATA = When the received signal contains purely data information.
DATA_AND_CLOCK is only used for bit slices positioned at DBC, QBC or GC pins (bitslice_0).
DATA can be used for all bit slices in a nibble when the received signal contains pure data information.</td></tr>
<tr><td>RX_DATA_WIDTH</td><td>4 or 8</td><td>8</td><td>Decimal</td><td>Note: Because BITSLICE_CONTROL.DIV_MODE must match the data width, TX_DATA_WIDTH and RX_DATA_WIDTH must be same.

Attribute defining the output width of the serial-to-parallel converter.
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:
<table>
<tr><th>RXTX_BITSLICE DATA_WIDTH</th><th>BITSLICE_CONTROL DIV_MODE</th></tr>
<tr><td>4</td><td>2</td></tr>
<tr><td>8</td><td>4</td></tr>
</table></td></tr>
<tr><td>RX_DELAY_FORMAT</td><td>TIME[1] or COUNT</td><td>TIME</td><td>String</td><td>Note: For BISC to properly align RXTX_BITSLICEs, set TX_DELAY_FORMAT= RX_DELAY_FORMAT.

DELAY_FORMAT can be either TIME or COUNT.
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).
BISC uses the RX_REFCLK_FREQUENCY attribute 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.
When set to COUNT, the value given in RX_DELAY_VALUE is the number of taps required.</td></tr>
<tr><td>RX_DELAY_TYPE</td><td>FIXED
VAR_LOAD
VARIABLE</td><td>FIXED</td><td>String</td><td>Delay mode of the input delay line. For more information, see Native Input Delay Type Usage.</td></tr>
</table>

Table 83: **RXTX_BITSLICE Attributes** *(cont'd)*

Attributes	Values	Default	Type	Description
RX_DELAY_VALUE	0–1100 (TIME) 0–511 (COUNT)	0	Decimal	***Note*:** For BISC to properly align, set RX_CLK_PHASE_P = RX_CLK_PHASE_N = SHIFT_0. TIME mode: Desired value in ps. Spartan UltraScale+ devices support up to 1.1 ns. COUNT mode: Desired value in taps. To ensure TX_BITSLICE data alignment, limit COUNT delays to 1.5 UI. For more information, see Native Input Delay Type Usage.
TX_DATA_WIDTH	4 or 8	8	Decimal	Because BITSLICE_CONTROL.DIV_MODE must match the data width, TX_DATA_WIDTH and RX_DATA_WIDTH must be same. Attribute defining the input width of the parallel-to-serial converter. TX_DATA_WIDTH = 2 x BITSLICE_CONTROL.DIV_MODE
TX_DELAY_FORMAT	TIME[1] or COUNT	TIME	String	***Note*:** For BISC to properly calibrate RXTX_BITSLICEs, set TX_DELAY_FORMAT= RX_DELAY_FORMAT. TX_DELAY_FORMAT can be either TIME or COUNT. When set to TIME, the delay after BISC completes (DLY_RDY goes High) equals the delay given in TX_DELAY_VALUE (specified in ps). BISC uses the TX_REFCLK_FREQUENCY attribute 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. When set to COUNT, the value given in TX_DELAY_VALUE is the number of taps required.
TX_DELAY_TYPE	FIXED VAR_LOAD VARIABLE	FIXED	String	Delay mode of the output delay line. For further information, see Native Output Delay Type Usage.
TX_DELAY_VALUE	0–1100 (TIME) 0–511 (COUNT)	0	Decimal	***Note*:** For BISC to properly calibrate RXTX_BITSLICEs, set TX_DELAY_VALUE= RX_DELAY_VALUE and TX_OUTPUT_PHASE_90 = FALSE. TIME mode: Desired value of the delay line in ps. Spartan UltraScale+ devices support up to 1.1 ns. COUNT mode: Desired value of the delay line in taps. To ensure TX_BITSLICE data alignment, limit COUNT delays to 1.5 UI.

Table 83: **RXTX_BITSLICE Attributes** *(cont'd)*

Attributes	Values	Default	Type	Description
RX_REFCLK_FREQUENCY	300.00–2666.67	300.0	1 significant digit float	***Note***: Because there is only a single reference clock for BITSLICE_CONTROL, set TX_REFCLK_FREQUENCY = RX_REFCLK_FREQUENCY. Specification of reference clock frequency in MHz. 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 *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) as TIDELAY_RESOLUTION. 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.
TX_REFCLK_FREQUENCY	300.00–2666.67	300.0	1 significant digit float	***Note***: Because there is only a single reference clock for BITSLICE_CONTROL, set TX_REFCLK_FREQUENCY = RX_REFCLK_FREQUENCY. Specification of reference clock frequency in MHz. 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 *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) as TIDELAY_RESOLUTION. 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.

Table 83: **RXTX_BITSLICE Attributes** *(cont'd)*

Attributes	Values	Default	Type	Description
RX_UPDATE_MODE	ASYNC, SYNC, or MANUAL	ASYNC	String	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. 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. 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.
TX_UPDATE_MODE	ASYNC, SYNC, or MANUAL	ASYNC	String	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. 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. 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.
FIFO_SYNC_MODE	TRUE or FALSE	FALSE	BOOLSTRING	Attribute defining the relationship between FIFO_WRCLK_OUT and FIFO_RD_CLK. Always set this attribute to FALSE. FIFO_SYNC_MODE = TRUE. Reserved for later use. See the Clocking in Native Mode in the BITSLICE_CONTROL section for more information on these clocks.
INIT	`1'b0` or `1'b1`	`1'b1`	Binary	Defines the initial value of the serialized data output of the RXTX_BITSLICE/TX_BITSLICE.
LOOPBACK	TRUE or FALSE	FALSE	BOOLSTRING	FALSE: RXTX_BITSLICE has distinct input (DATAIN) and/or output (O) to the input or output of bidirectional buffers in the IOB. 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.

Table 83: **RXTX_BITSLICE Attributes** *(cont'd)*

Attributes	Values	Default	Type	Description
TBYTE_CTL	TBYTE_IN or T	TBYTE_IN	Decimal	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. 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.
TX_OUTPUT_PHASE_90	TRUE or FALSE	FALSE	String	FALSE: Output of RXTX_BITSLICE/TX_BITSLICE is not phase-shifted. TRUE: Output of RXTX_BITSLICE/TX_BITSLICE is phase-shifted 90 degrees. RX_DELAY_VALUE/ TX_DELAY_VALUE must be set to 0 when TX_OUTPUT_PHASE_90 =TRUE. 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.
ENABLE_PRE_EMPHASIS	TRUE or FALSE	FALSE	String	Used with attributes on the bidirectional IOB to enable and disable pre-emphasis. Pre-emphasis is documented in Transmitter Pre-Emphasis.
IS_RX_CLK_INVERTED	`1'b0` or `1'b1`	`1'b0`	Binary	When 1 reverses polarity (inverts) the RX_CLK signal. Similar to the IS_RX_RST_INVERTED attribute but on the RX_CLK path. When IS_RX_CLK_INVERTED = 1, the inverter is used. When IS_RX_CLK_INVERTED = 0, the inverter is not used.
IS_RX_RST_DLY _INVERTED	`1'b0` or `1'b1`	`1'b0`	Binary	When 1 reverses polarity (inverts) the RX_RST_DLY signal. Similar to the IS_RX_RST_INVERTED attribute but on the RX_RST_DLY path. When IS_RX_RST_DLY_INVERTED = 1, the inverter is used. When IS_RX_RST_DLY_INVERTED = 0, the inverter is not used.
IS_RX_RST_INVERTED	`1'b0` or `1'b1`	`1'b0`	Binary	When 1 reverses polarity (inverts) the RX_RST signal. A selectable local inverter on the reset path that can change the polarity of the reset input. When IS_RX_RST_INVERTED = 1, the inverter is used. When IS_RX_RST_INVERTED = 0, the inverter is not used.

Table 83: **RXTX_BITSLICE Attributes** *(cont'd)*

Attributes	Values	Default	Type	Description
IS_TX_CLK_INVERTED	1'b0 or 1'b1	1'b0	Binary	When 1 reverses polarity (inverts) the TX_CLK signal. This attribute is similar to the IS_RX_RST_INVERTED attribute but on the TX_CLK path. When IS_TX_CLK_INVERTED = 1, the inverter is used. When IS_TX_CLK_INVERTED = 0, the inverter is not used.
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	Reserved for memory interface generator (MIG). When TRUE, bypass the ODELAY.
SIM_DEVICE	ULTRASCALE_PLUS	ULTRASCALE	String	Sets the device version (ULTRASCALE_PLUS)

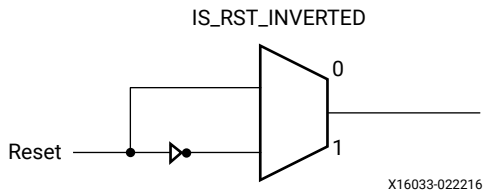
Notes:

1. When in TIME mode, calibration affects the availability of bit slices within the nibble. See HP I/O 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 the following figure.

Figure 121: **IS_RST_INVERTED Attribute**



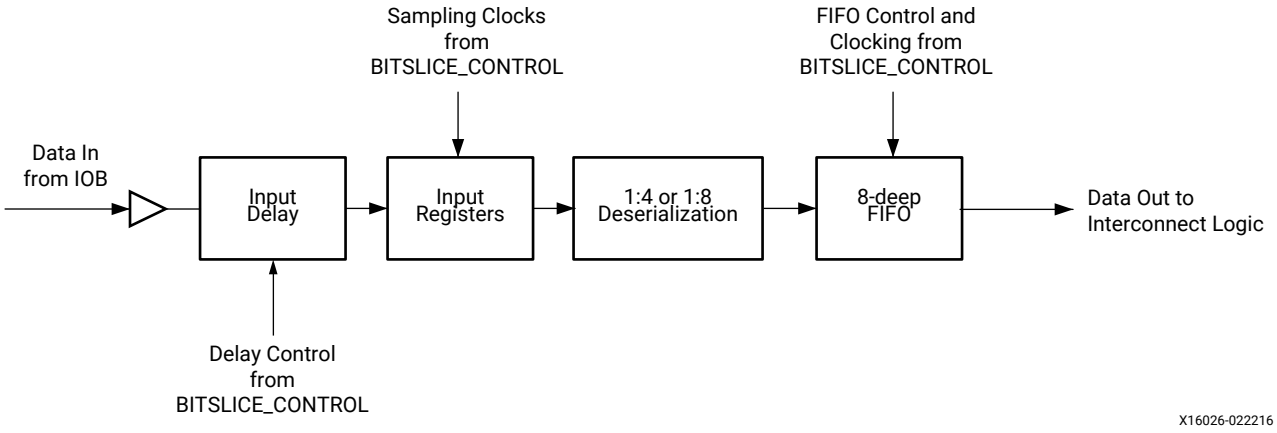
RX_BITSLICE

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 the following figure.

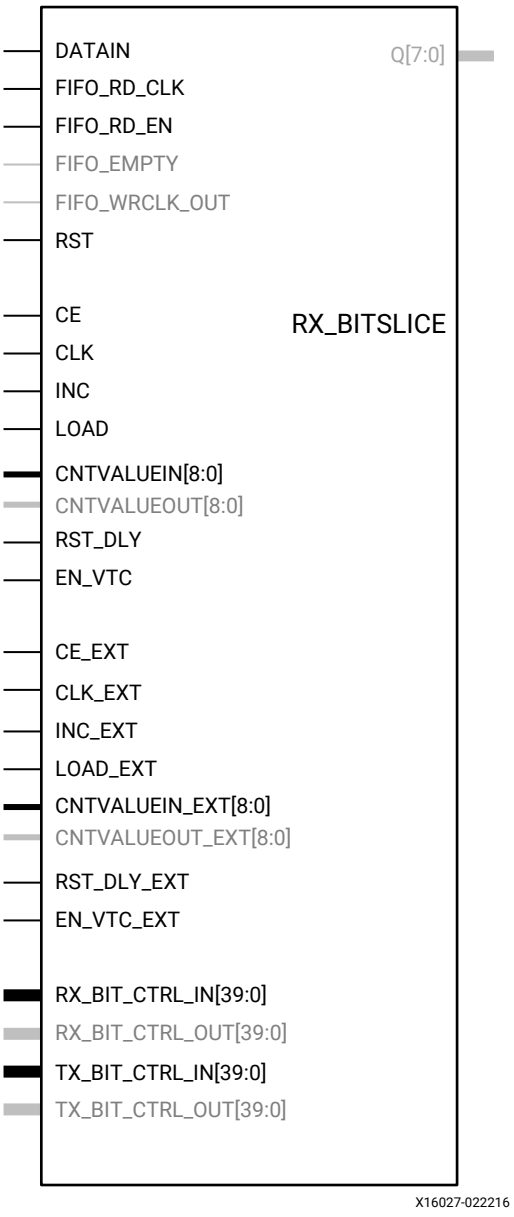
***Note*:** The input buffer is not part of the RX_BITSLICE.

Figure 122: **RX_BITSLICE Block Diagram**



The RX_BITSLICE primitive is shown in the following figure. In this figure, black represents inputs and gray represents outputs. Table 82 lists the RXTX_BITSLICE ports.

Figure 123: **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

The following table lists the RX_BITSLICE ports.

Table 84: **RX_BITSLICE Ports**

Port	Function[1]	I/O	Synchronous Clock Domain	Description
DATAIN	I/O RX	Input	Asynchronous	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. The incoming signal from the IOB can be data, clock, or strobe, selected by the DATA_TYPE attribute on the RX_BITSLICE. 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 Clocking in Native Mode in the BITSLICE_CONTROL section for more information. When the incoming signal from the IOB is data only, it can be located at any bit slice position in the nibble.
Q[7:0]	RX FPGA	Output	FIFO_RD_CLK	Deserialized (parallel) output data from the RX FIFO going to the interconnect logic. 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. ***Note*:** 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. If DATA_WIDTH = 8, Q[7:0] represent 8 bits of captured serial data.
RST	RX FPGA	Input	Asynchronous	Resets the RX_BITSLICE 0 logic, asynchronous assertion, synchronous deassertion, and is active-High. Q resets to zero while RST is asserted. See Native Mode Bring-up and Reset for more information.
CLK	RX FPGA	Input	Asynchronous	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.

Table 84: **RX_BITSLICE Ports** *(cont'd)*

Port	Function[1]	I/O	Synchronous Clock Domain	Description
CE	RX FPGA	Input	CLK	Clock enable for the delay line register clock. ***Note*:** 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	RX FPGA	Input	Asynchronous (synchronous deassertion to CLK)	Reset port for the delay line within the receiver logic. Resets the internal delay line to the value defined in DELAY_VALUE.
INC	RX 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 through the delay line does not change (regardless of the state of INC). 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	RX FPGA	Input	CLK	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. ***Note*:** 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 84: **RX_BITSLICE Ports** *(cont'd)*

Port	Function[1]	I/O	Synchronous Clock Domain	Description
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 EN_VTC signal must be pulled High during initial BISC. When COUNT mode is used, the EN_VTC signal must be pulled Low. 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.
CNTVALUEIN[8:0]	RX FPGA	Input	CLK	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. 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. For RX_BITSLICEs used as clock/strobe, CNTVALUEIN is not supported. Clocking in Native Mode in the BITSLICE_CONTROL section describes how the strobe/clock is tuned using BISC. Provide CNTVALUEIN one clock cycle before LOAD pulse High.
CNTVALUEOUT[8:0]	RX FPGA	Output	CLK	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.
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 FIFO Function in RXTX_BITSLICE.
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.

Table 84: **RX_BITSLICE Ports** *(cont'd)*

Port	Function[1]	I/O	Synchronous Clock Domain	Description
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 FIFO Function.
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 84: **RX_BITSLICE Ports** *(cont'd)*

Port	Function[1]	I/O	Synchronous Clock Domain	Description
INC_EXT	TX FPGA	Input	CLK_EXT	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. 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.
LOAD_EXT	TX FPGA	Input	CLK_EXT	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.
EN_VTC_EXT	TX FPGA	Input	Asynchronous	Enable voltage, temperature, and process 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 EN_VTC_EXT signal must be pulled High during initial BISC. When COUNT mode is used, the EN_VTC_EXT signal must be pulled Low. 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.

Table 84: **RX_BITSLICE Ports** *(cont'd)*

Port	Function[1]	I/O	Synchronous Clock Domain	Description
CNTVALUEIN_EXT[8:0]	TX FPGA	Input	CLK_EXT	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.
CNTVALUEOUT_EXT[8:0]	TX FPGA	Output	CLK_EXT	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.
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 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'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.				
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

The following table lists the RX_BITSLICE attributes.

Table 85: **RX_BITSLICE Attributes**

Attributes	Values	Default	Type	Description
DATA_TYPE	DATA DATA_AND_CLOCK SERIAL	DATA	String	SERIAL = When received data must be captured by a unrelated clock (for example, SGMII). 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. DATA = When the received signal contains purely data information. DATA_AND_CLOCK is only used for bit slices positioned at DBC, QBC or GC pins (BITSLICE_0). DATA can be used for all bit slices in a nibble when the received signal contains pure data information.
DATA_WIDTH	4 or 8	8	Decimal	Attribute defining the output width of the serial-to-parallel converter. 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: RX_BITSLICE DATA_WIDTH: BITSLICE_CONTROL DIV_MODE 4: 2 8: 4
DELAY_FORMAT	TIME[1] or COUNT	TIME	String	DELAY_FORMAT can be either TIME or COUNT. 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 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. 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.
DELAY_TYPE	FIXED VAR_LOAD VARIABLE	FIXED	String	Delay mode of the input delay line. For further information, see Native Input Delay Type Usage.

Table 85: **RX_BITSLICE Attributes** *(cont'd)*

Attributes	Values	Default	Type	Description
DELAY_VALUE	0–1100 (TIME) 0–511 (COUNT)	0	Decimal	***Note***: For BISC to properly align, set OUTPUT_PHASE_90 = FALSE. TIME mode: Desired value in ps. Spartan UltraScale+ devices support delays up to 1.1 ns. COUNT mode: Desired value in taps.
DELAY_FORMAT_EXT	TIME[1] or COUNT	TIME	String	DELAY_FORMAT_EXT can be either TIME or COUNT. Must match DELAY_FORMAT. The attribute value should match DELAY_FORMAT when CASCADE is set to TRUE. 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.
DELAY_TYPE_EXT	FIXED VAR_LOAD VARIABLE	FIXED	String	Delay mode of the extended delay line. For further information, see Extended Delay Control Signals.
DELAY_VALUE_EXT	0–1100 (TIME) 0–511 (COUNT)	0	Decimal	Delay value for extended delay. TIME mode: Desired value in ps. COUNT mode: Desired value in taps. For further information, see Extended Delay Control Signals.

Table 85: **RX_BITSLICE Attributes** *(cont'd)*

Attributes	Values	Default	Type	Description
REFCLK_FREQUENCY	300.00–2666.67	300.0	1 significant digit float	Specification of reference clock frequency in MHz. 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 *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) as TIDELAY_RESOLUTION in the references section. The REFCLK_FREQUENCY attribute is used by the BISC algorithm to calculate the tap size but not affect the tap size. 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 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.
UPDATE_MODE	ASYNC, SYNC, or MANUAL	ASYNC	String	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. 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. 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.

Table 85: **RX_BITSLICE Attributes** *(cont'd)*

Attributes	Values	Default	Type	Description
UPDATE_MODE_EXT	ASYNC, SYNC, or MANUAL	ASYNC	String	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. 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. 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 Extended Delay Control Signals. Value should match UPDATE_MODE value.
FIFO_SYNC_MODE	TRUE (Reserved) or FALSE	FALSE	BOOLSTRING	FALSE: This attribute defines the relationship between FIFO_WRCLK_OUT and FIFO_RD_CLK. Always set this attribute to FALSE. ***Note***: FIFO_SYNC_MODE = TRUE. Reserved for later use. See Clocking in Native Mode in the BITSLICE_CONTROL section for more information on these clocks.
CASCADE	TRUE or FALSE	FALSE	String	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. Consider the use of the attributes for the cascaded output delay line in addition to the master input delay attributes. FALSE: Disables cascading and the extended (_EXT) attributes can be ignored (pull input Low and leave outputs open). See the Extended Delay Control Signals description for more information on delay cascading with the RX_BITSLICE. ***Note***: CASCADE = TRUE has reduced performance and should not be used when performance is critical.

Table 85: **RX_BITSLICE Attributes** *(cont'd)*

Attributes	Values	Default	Type	Description
IS_CLK_INVERTED	1'b0 or 1'b1	1'b0	Binary	Similar to the IS_RST_INVERTED attribute but on the RX_CLK path. When IS_CLK_INVERTED = 1 the inverter is used. When 1, reverses polarity (inverts) the CLK signal. When 0, the inverter is not used.
IS_RST_DLY _INVERTED	1'b0 or 1'b1	1'b0	Binary	Similar to the IS_RST_INVERTED attribute but on the RST_DLY path. When IS_RST_DLY_INVERTED = 1 the inverter is used. When 1, reverses polarity (inverts) the RST_DLY signal. When 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 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	ULTRASCALE_PLUS	ULTRASCALE	String	Sets the device version (ULTRASCALE_PLUS).

Notes:

1. When in TIME mode, calibration affects the availability of bit slices within the nibble. See HP I/O Bank Overview for more information.

Extended Delay Control Signals

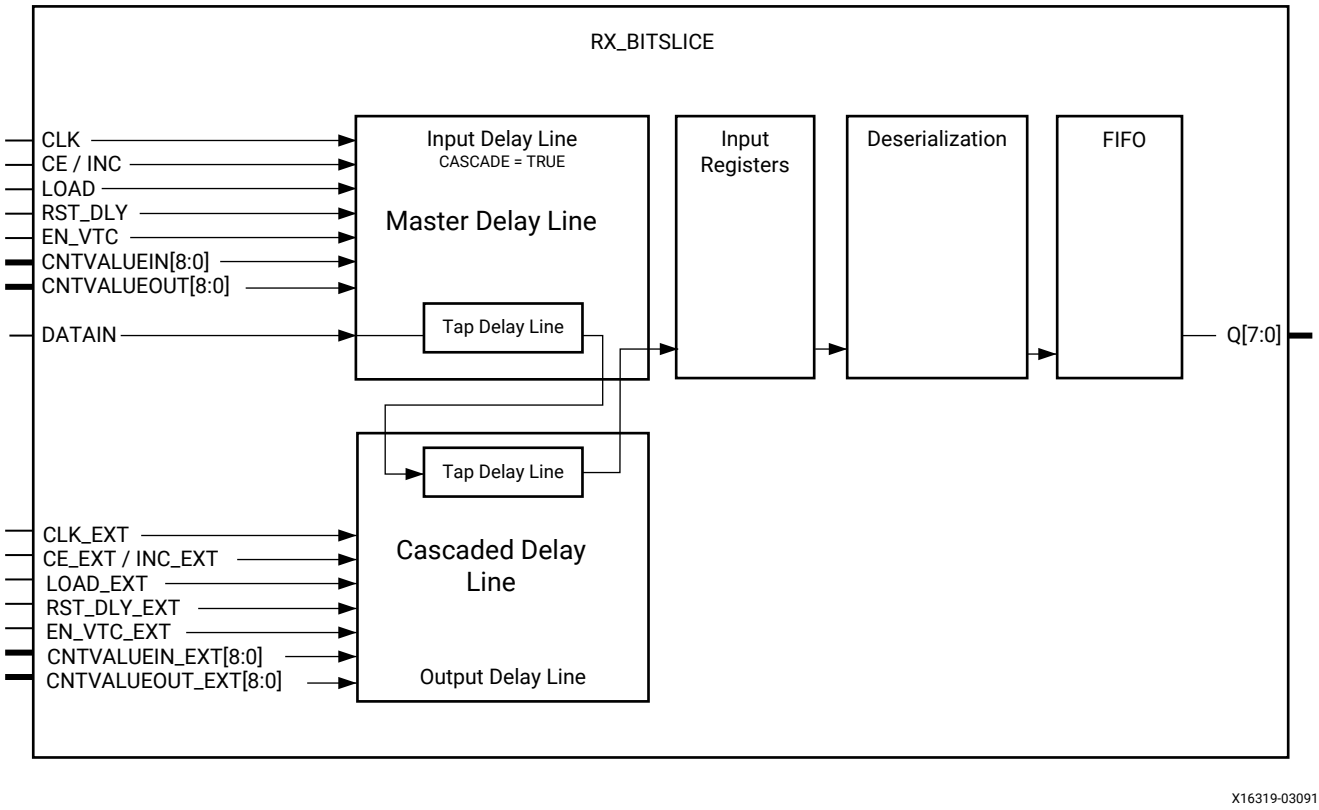
Further information on the CE_EXT, CLK_EXT, EN_VTC_EXT, INC_EXT, RST_DLY_EXT, LOAD_EXT, CNTVALUEIN_EXT[8:0], and CNTVALUEOUT_EXT[8:0] signals is presented in this section.

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 the following figure. 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.

Figure 124: **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_{CC} 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 the following figure.

Figure 125: **TX_BITSLICE Block Diagram**

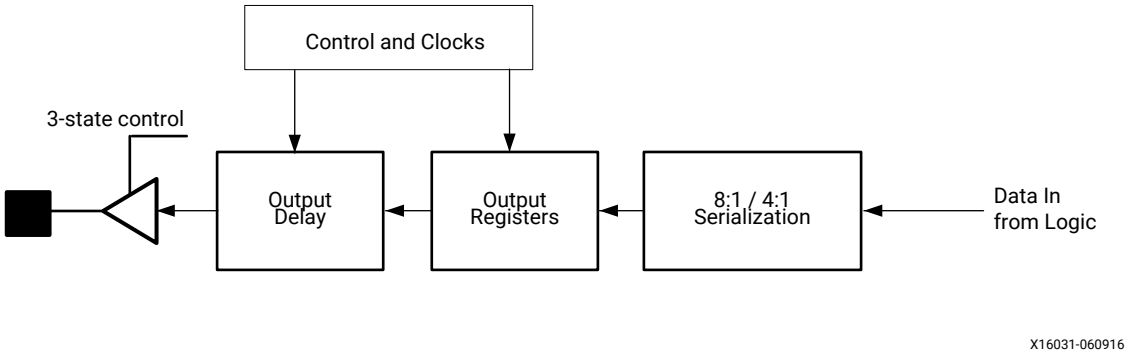
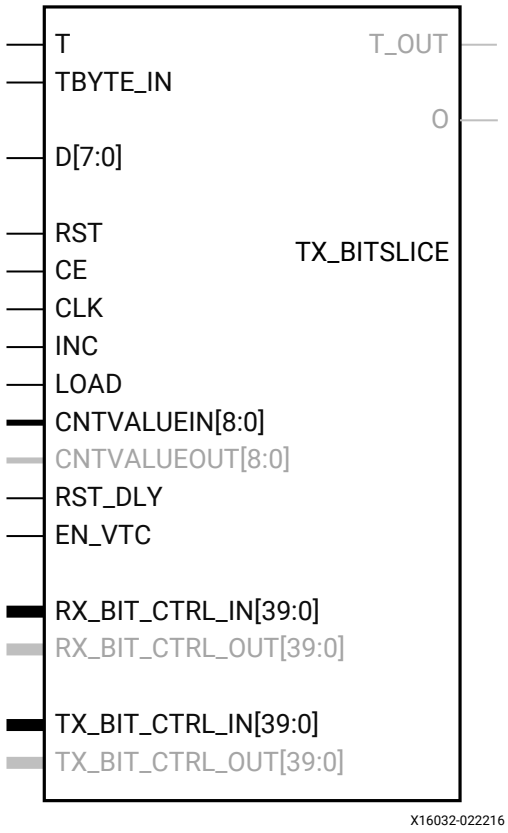


Figure 126: **TX_BITSLICE Primitive**



TX_BITSLICE Function

The function of the TX_BITSLICE is discussed in depth in the earlier section RXTX_BITSLICE Transmitter Function.

TX_BITSLICE Ports

The following table lists the TX_BITSLICE ports.

Table 86: **TX_BITSLICE Ports**

Port	Function[1]	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 86: **TX_BITSLICE Ports** *(cont'd)*

Port	Function[1]	I/O	Synchronous Clock Domain	Description
T	TX FPGA	Input	Asynchronous	T assigns a combinatorial path through the TX_BITSLICE to the 3-state pin of an output buffer. 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. Each TX_BITSLICE in a nibble has a T input, meaning that there are 13 T inputs for a byte (byte = two nibbles). A logic High means the output buffer is 3-stated and a logic Low means the output buffer is not 3-stated. Active-High.
TBYTE_IN	TX FPGA	Input	PLL_CLK (BITSLICE_CONTROL)	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.
RST	TX FPGA	Input	Asynchronous	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. For deterministic bring-up, follow the steps in Native Mode Bring-up and Reset.
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. ***Note*:** 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.

Table 86: **TX_BITSLICE Ports** *(cont'd)*

Port	Function[1]	I/O	Synchronous Clock Domain	Description
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.
LOAD	TX FPGA	Input	CLK	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. ***Note*:** 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.
EN_VTC	TX FPGA	Input	Asynchronous	Enable voltage, temperature, and process 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 EN_VTC signal must be pulled High during initial built-in self-calibration (BISC). When used in COUNT mode, the EN_VTC signal must be pulled Low. 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.

Table 86: **TX_BITSLICE Ports** *(cont'd)*

Port	Function[1]	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. ***Note*:** 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 Table 111 and Table 112).
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.				
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

The following table lists the TX_BITSLICE attributes.

Table 87: **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 TX_BITSLICE_TRI.
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.
DELAY_TYPE	FIXED VAR_LOAD VARIABLE	FIXED	String	Delay mode of the output delay line. For more information, see Native Output Delay Type Usage.

Table 87: **TX_BITSLICE Attributes** *(cont'd)*

Attributes	Values	Default	Type	Description
DELAY_VALUE	0–1100 (TIME) 0–511 (COUNT)	0	Decimal	***Note***: For BISC to properly align, set RX_CLK_PHASE_P = RX_CLK_PHASE_N = SHIFT_0. When DELAY_FORMAT is set to TIME mode, the desired value is in ps. Spartan UltraScale+ devices support up to 1.1 ns. When DELAY_FORMAT is set to COUNT mode, the desired value is in number of taps. For more information, see Native Output Delay Type Usage. To ensure TX_BITSLICE data alignment, limit COUNT delays to 1.5 UI.
REFCLK_FREQUENCY	300.00–2666.67	300.0	1 significant digit float	Specification of reference clock frequency in MHz. 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 Clocking in Native Mode and Built-in Self-Calibration. As opposed to previous FPGA families, the tap size is not determined by the REFCLK_FREQUENCY, the tap size is defined in the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) as $T_{ODELAY\_RESOLUTION}$ 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.

Table 87: **TX_BITSLICE Attributes** *(cont'd)*

Attributes	Values	Default	Type	Description
OUTPUT_PHASE_90	TRUE or FALSE	FALSE	String	FALSE: Output O is not phase-shifted. TRUE: Output O is phase-shifted 90 degrees. DELAY_VALUE must be set to 0 when OUTPUT_PHASE_90 = TRUE. 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).
DELAY_FORMAT	TIME[1] COUNT	TIME	String	DELAY_FORMAT can be either TIME or COUNT. When set to TIME, the delay after BISC completes (DLY_RDY goes High) equals the delay given in DELAY_VALUE (specified in ps). BISC uses the REFCLK_FREQUENCY attribute 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. 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.

Table 87: **TX_BITSLICE Attributes** *(cont'd)*

Attributes	Values	Default	Type	Description
UPDATE_MODE	ASYNC, SYNC, or MANUAL	ASYNC	String	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. 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. 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.
ENABLE_PRE_EMPHASIS	TRUE FALSE	FALSE	String	Used with attributes on the bidirectional IOB to enable and disable pre-emphasis. The ENABLE_PRE_EMPHASIS attribute is used with IOB to enable the pre-emphasis. See Transmitter Pre-Emphasis.
IS_CLK_INVERTED	`1'b0` or `1'b1`	`1'b0`	Binary	Similar to the IS_RST_INVERTED attribute, but on the CLK path. When IS_CLK_INVERTED = 1, the inverter is used to reverse polarity (invert) the CLK signal. When IS_CLK_INVERTED = 0, the inverter is not used.
IS_RST_DLY_INVERTED	`1'b0` or `1'b1`	`1'b0`	Binary	Similar to the IS_RST_INVERTED 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.

Table 87: **TX_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 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 Figure 121.
NATIVE_ODELAY_BYPASS	TRUE or FALSE	FALSE	String	Reserved for memory interface generator (MIG). When TRUE, bypass the ODELAY.
SIM_DEVICE	ULTRASCALE_PLUS	ULTRASCALE	String	Sets the device version (ULTRASCALE_PLUS).

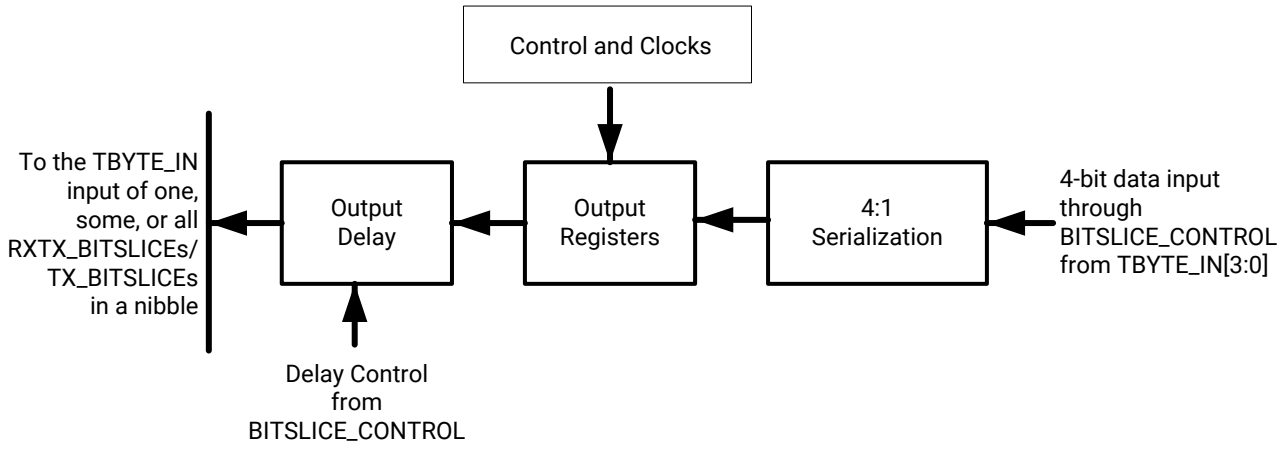
Notes:

1. When in TIME mode, calibration affects the availability of bit slices within the nibble. See HP I/O 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 the following figure.

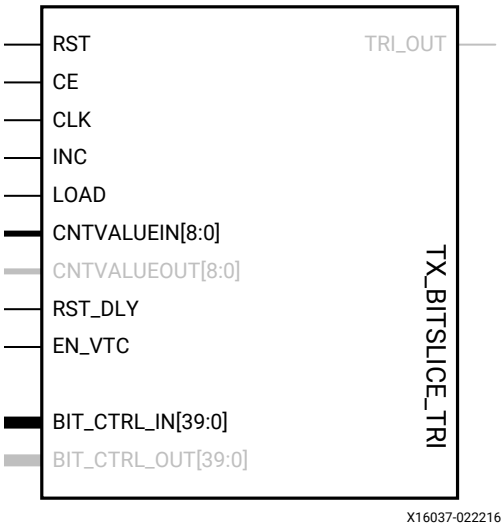
Figure 127: **TX_BITSLICE_TRI Block Diagram**



The TX_BITSLICE_TRI can only be used to 3-state bit slices within a nibble. TX_BITSLICE_TRI Function 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 130 shows the relationship of the TBYTE_IN input of the BITSLICE_CONTROL to the TX_BITSLICE.O output and IOB 3-state buffer.

Figure 128: **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 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 129 shows how the primitives are interconnected and the waveform in Figure 130 shows how signals must be applied 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 131, 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 131). 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 131, 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 129 shows the required connections when connecting 3-state control using the TX_BITSLICE_TRI and T_BYTE_IN[3:0] of TX_BITSLICE.

Figure 129: **Connections for a 3-State Path when Using the TBYTE Port**

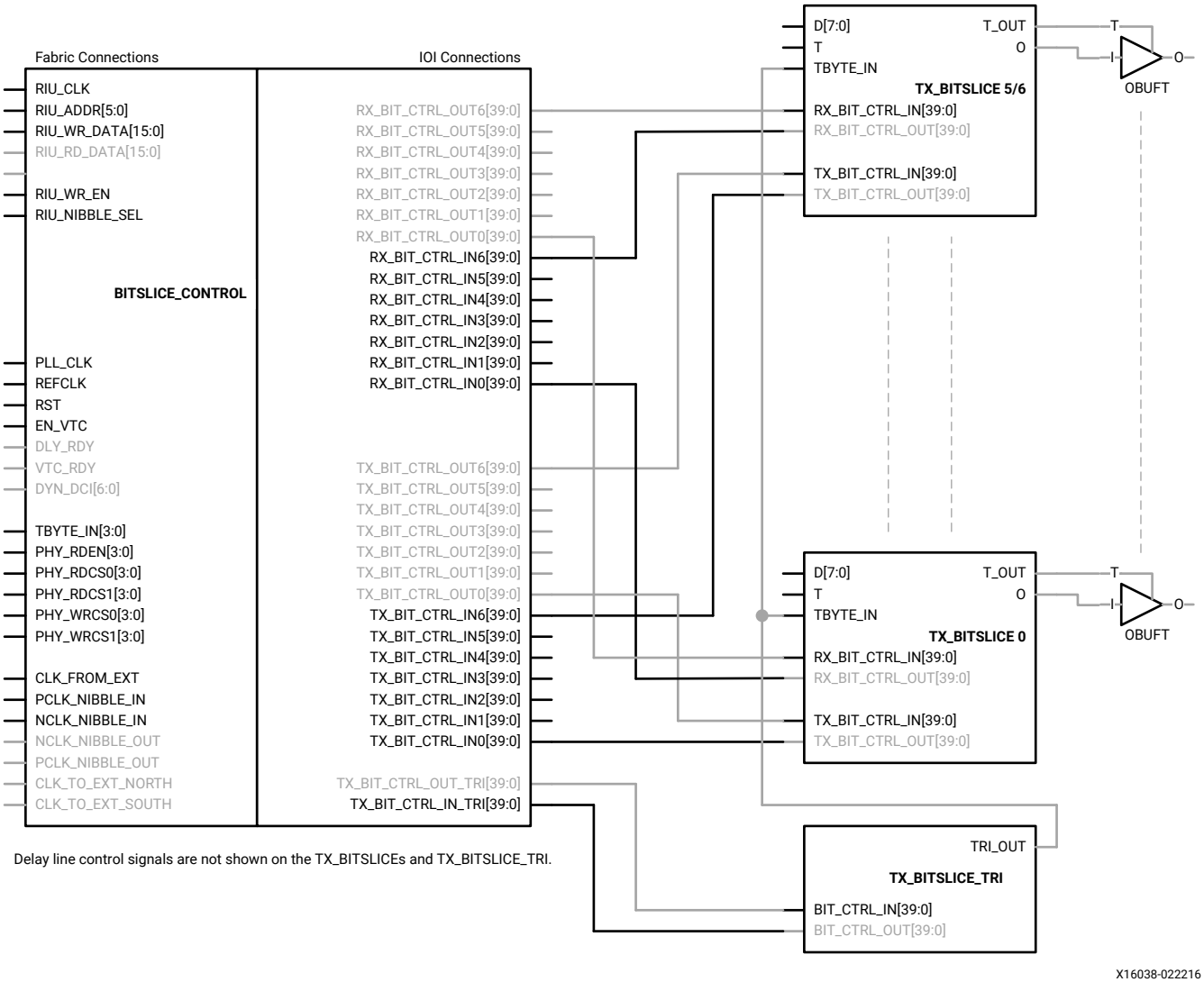
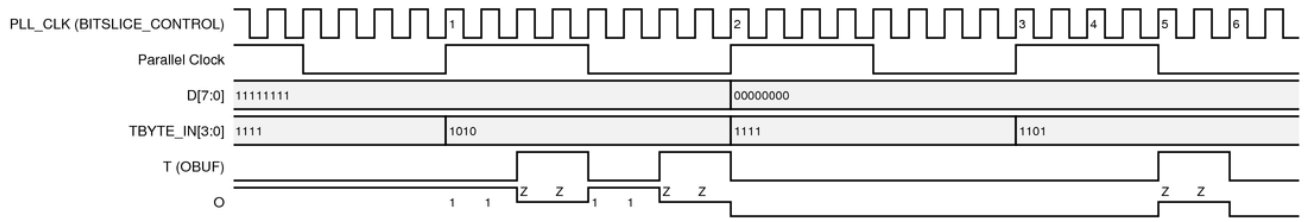


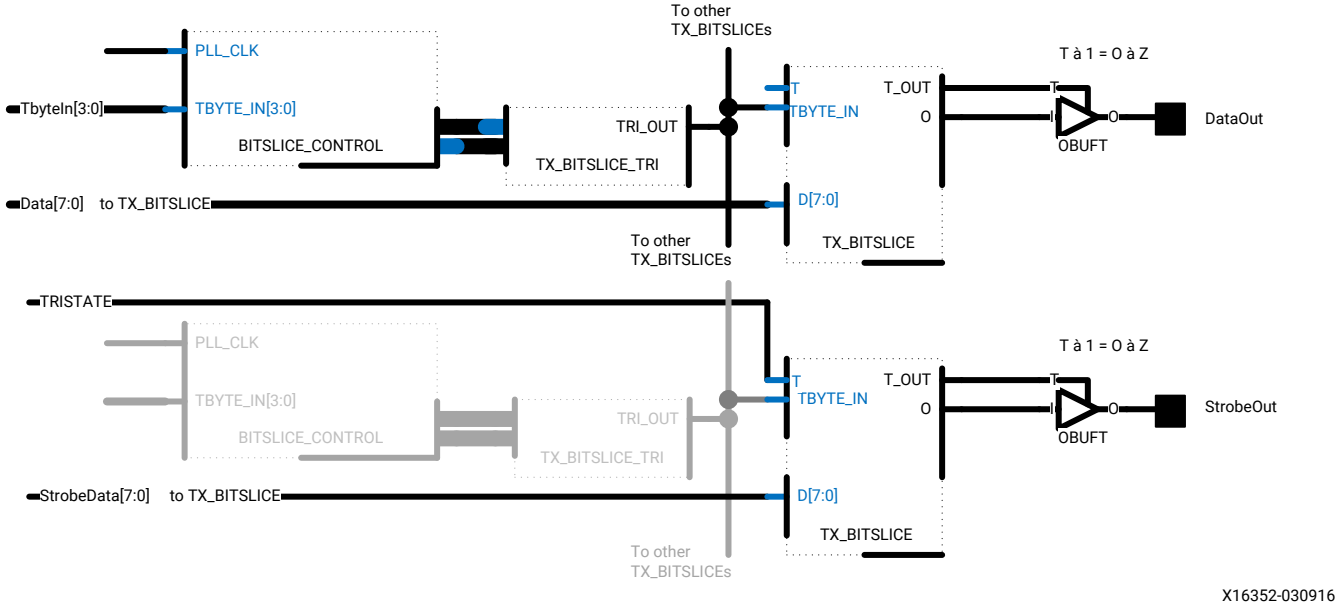
Figure 130: **Connections for a 3-State Path when Using TBYTE (DATA_WIDTH=8)**



Notes on the preceding figure:

- For easier viewing, latency is not shown.
- At the start, BITSLICE_CONTROL.TBYTE_IN is 1111 and the output buffer is not 3-stated. The output of the OBUFT is all ones (11111111).
- At event 1, the TBYTE_IN at the BITSLICE_CONTROL is 1010 while the data input is all High. This causes part of the serial data stream to 3-state. The serial stream out of the output buffer O port is 11ZZ11ZZ.
- At event 2, the parallel data input is all Low, the TBYTE_IN is 1111, and therefore the data stream is not 3-stated.
- At event 3, the TBYTE_IN is 1101 while the parallel input is still zero, therefore the fourth and fifth bits of the parallel word are 3-stated. The D input is all logic Low. The output data is 0000ZZ00.

Figure 131: **Using the TBYTE_CTL Attribute Settings TBYTE_IN or T**



The latency for the TX_BITSLICE_TRI is shown in the following figures.

Figure 132: **3-State Latency, DATA_WIDTH = 8**

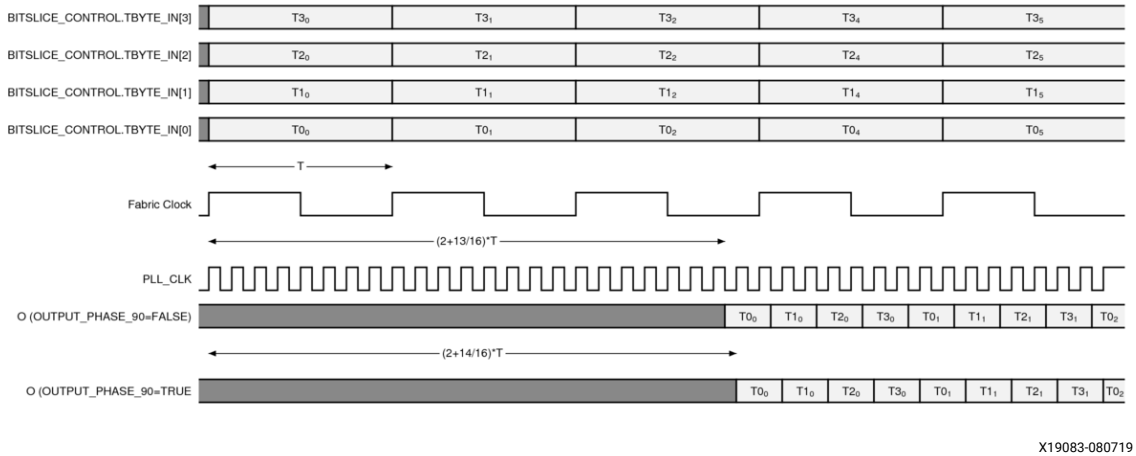
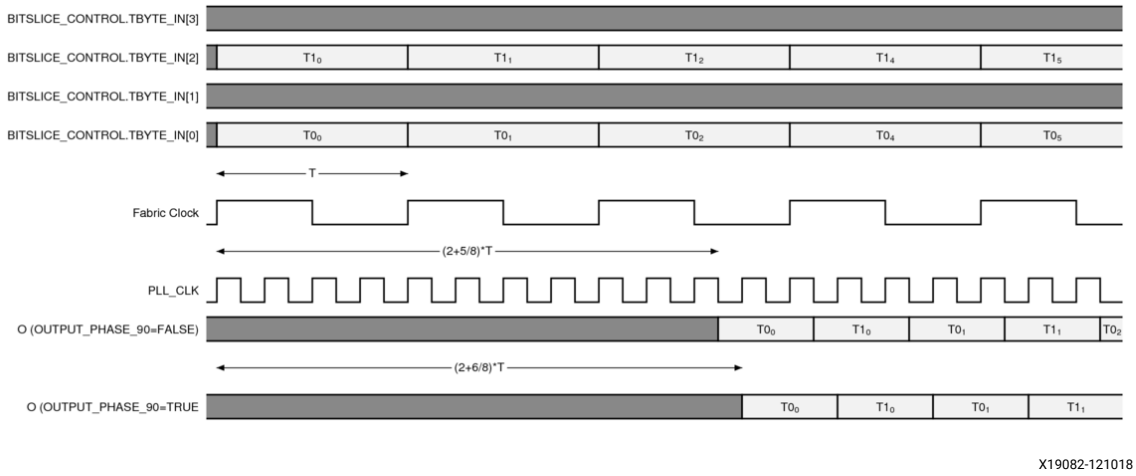


Figure 133: **3-State Latency, DATA_WIDTH = 4**



TX_BITSLICE_TRI Ports

The following table lists the TX_BITSLICE_TRI ports.

Table 88: **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 Native Mode Bring-up and Reset.
CE	Input	Clock enable for the 3-state delay line register clock.

Table 88: **TX_BITSLICE_TRI Port Descriptions** *(cont'd)*

Port	I/O	Description
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.
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.

Table 88: **TX_BITSLICE_TRI Port Descriptions** *(cont'd)*

Port	I/O	Description
TRI_OUT	Output	3-state output (TRI_OUT) outputs to the TBYTE_IN pins of the bit slices.

TX_BITSLICE_TRI Attributes

Table 89: **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.
DELAY_FORMAT	TIME[1], COUNT	TIME	String	DELAY_FORMAT can be either TIME or COUNT. When set to TIME, the delay after BISC completes (DLY_RDY goes High) equals the delay given in DELAY_VALUE (specified in ps). BISC uses the REFCLK_FREQUENCY attribute 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. 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.
DELAY_TYPE	FIXED, VAR_LOAD, VARIABLE	FIXED	String	Delay mode of the input delay line.
DELAY_VALUE	0–1100 (TIME) 0–511 (COUNT)	0	Decimal	TIME mode: Desired value in ps. Spartan UltraScale+ devices support up to 1.1 ns. COUNT mode: Desired value in taps.

Table 89: **TX_BITSLICE_TRI Attributes** *(cont'd)*

Attributes	Values	Default	Type	Description
UPDATE_MODE	ASYNC MANUAL SYNC	ASYNC	String	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. 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. 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.
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	The output phase can be chosen to be either 0 or 90 degrees. DELAY_VALUE must be set to 0 when OUTPUT_PHASE_90 =TRUE.
REFCLK_FREQUENCY	300.00–2666.67	300.0	1 significant digit float	Specification of reference clock frequency in MHz. 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. See Clocking in Native Mode and Built-in Self-Calibration in the BITSLICE_CONTROL section. The tap size is not determined by the REFCLK_FREQUENCY. A tap delay range is specified in the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) as $T_{IDELAY\_RESOLUTION}$. The REFCLK_FREQUENCY attribute is used by the BISC algorithm to calculate the number of taps required for the requested DELAY_VALUE.
IS_CLK_INVERTED	1'b0, 1'b1	1'b0	Binary	Specifies whether the CLK pin is active-High or active-Low. Similar to the IS_RST_INVERTED attribute but on the CLK path. When IS_CLK_INVERTED = 1, the inverter is used. When IS_CLK_INVERTED = 0, the inverter is not used.

Table 89: **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	Reserved for memory interface generator (MIG). When TRUE, bypass the ODELAY.
SIM_DEVICE	ULTRASCALE_PLUS	ULTRASCALE	String	Sets the device version (ULTRASCALE_PLUS)

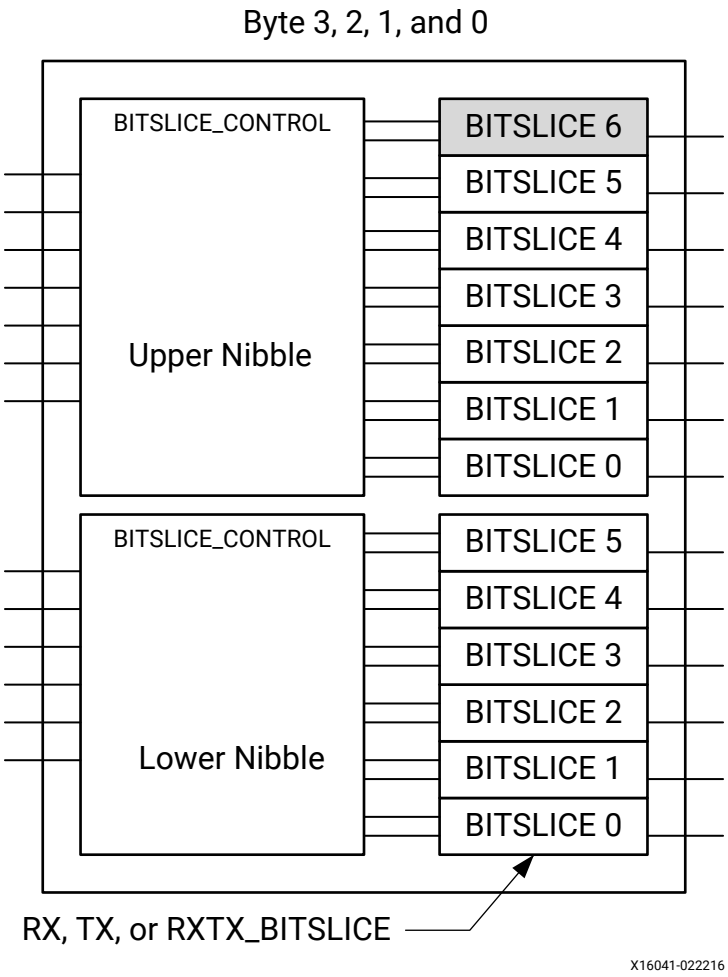
Notes:

1. When in TIME mode, calibration affects the availability of bit slices within the nibble. See HP I/O Bank Overview for more information.

BITSLICE_CONTROL

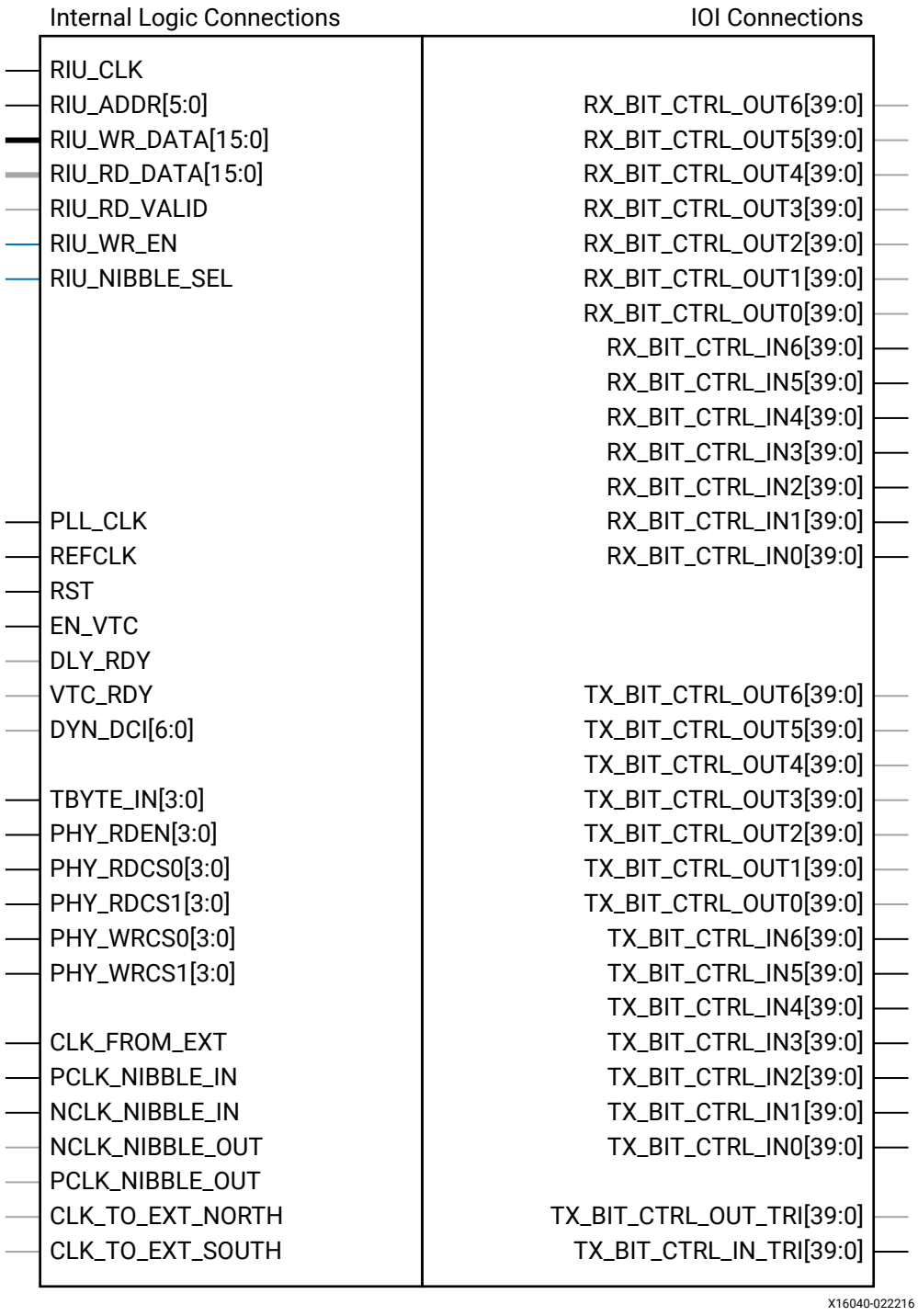
The data and clock/strobe base handling block are the RXTX_BITSLICE (RX_BITSLICE and TX_BITSLICE primitives are derived from this component), which are used per pin or pin pair. The six or seven bit slices within a nibble are all controlled by one BITSLICE_CONTROL block, as shown in the following figure. Seven bit slices make an upper nibble and six bit slices assemble a lower nibble.

Figure 134: **BITSLICE_CONTROL with Respect to Bit Slices**



Base functions of the BITSLICE_CONTROL primitive (see the following figure) 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.

Figure 135: **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
- Clocking in Native Mode
- Built-in Self-Calibration
- Register Interface Unit (RIU)

BITSLICE_CONTROL Ports

The following table lists the BITSLICE_CONTROL ports.

Table 90: **BITSLICE_CONTROL Ports**

Port	I/O	Synchronous Clock Domain	Description
PLL_CLK	Input	Asynchronous	Master clock input for the BITSLICE_CONTROL. Set REFCLK_SRC attribute = PLL_CLK. This clock is used by the BISC controller. When SERIAL_MODE = TRUE, it is also used for data and strobe/clock sample clock. 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. The PLL connects to the PLL_CLK pin over dedicated, very low jitter routing. Use this PLL_CLK clock input, or the REFCLK clock input, but not both. When the PLL_CLK is used, tie the REFCLK Low.
REFCLK	Input	Asynchronous	Master clock input for the BITSLICE_CONTROL. Set REFCLK_SRC attribute = REFCLK. REFCLK is only supported for RX_BITSLICE. This clock is used by the BISC controller. When SERIAL_MODE = TRUE, it is also used for data and strobe/clock sample clock. This clock can be generated by a MMCM in the internal logic. Connections to this clock input use clock buffers and route over normal clock routing in the FPGA. Use this REFCLK clock input, or the PLL_CLK clock input, but not both. When the REFCLK is used, tie the PLL_CLK Low. 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.

Table 90: **BITSLICE_CONTROL Ports** *(cont'd)*

Port	I/O	Synchronous Clock Domain	Description
RST	Input	Asynchronous	Asynchronous asserted global reset. This reset is best synchronously released, following a dedicated reset sequence. Read Native Mode Bring-up and Reset for more information. ***Note*:** 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.
EN_VTC	Input	RIU_CLK	Enable voltage and temperature control and tracking. Assertion of EN_VTC maintains the delay of the delay lines that are in TIME mode over the V and T changes. After DLY_RDY goes High and initial BISC completion, the EN_VTC signal must be pulled High. 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.
DLY_RDY	Output	Asynchronous	Status bit indicating when BISC finishes initial fixed delay line calibration. This pin is also represented by a RIU register bit.
VTC_RDY	Output	Asynchronous	Status signal indicating when BISC finishes baseline VT calibration and tracking. From now on, BISC continuously compensates the delay lines for voltage and temperature. After asserted, this signal stays High until a hardware reset of the BITSLICE_CONTROL or its EN_VTC is toggled Low. This pin is also represented by a RIU register bit. In Component mode, the IDELAYCTRL.RDY signal is the equivalent of this pin.
RIU_CLK	Input	Asynchronous	Clock for the RIU interface peripheral. This clock is independent from all other BITSLICE_CONTROL clocks. This clock can be generated by an MMCM or PLL.
RIU_ADDR[5:0]	Input	RIU_CLK	The address input bus provides a register address for the register interface. 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.
RIU_WR_DATA [15:0]	Input	RIU_CLK	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. RIU_VALID indicates when the RIU port is ready to accept another write. When not used, all bits must be set to zero.

Table 90: **BITSLICE_CONTROL Ports** *(cont'd)*

Port	I/O	Synchronous Clock Domain	Description
RIU_RD_DATA [15:0]	Output	RIU_CLK	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 Register Definitions and Addresses.
RIU_VALID	Output	RIU_CLK	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 (that is, 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.
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	
TBYTE_IN[3:0]	Input	PLL_CLK	Nibble/byte group 3-state input. 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. 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. Read more about this input in the TX_BITSLICE and TX_BITSLICE_TRI sections.
PHY_RDEN[3:0]	Input	PLL_CLK	Read enable. Must be tied to 1111 when the RX_GATING attribute is not used.
DYN_DCI[6:0]	Output	Asynchronous	MIG USE ONLY: Direct IOB DCI control.
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 Clocking in Native Mode.			

Table 90: **BITSLICE_CONTROL Ports** *(cont'd)*

Port	I/O	Synchronous Clock Domain	Description
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	Inter-nibble strobe/clock from the other BITSLICE_CONTROL in the byte. Each byte contains two nibbles and each nibble has a PCLK_NIBBLE_IN input. Use of this input is enabled by the EN_OTHER_PCLK attribute.
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.
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. When an RXTX_BITSLICE, RX_BITSLICE, or TX_BITSLICE is used, these buses must be connected to the appropriate BITSLICE_CONTROL input and output bus (Figure 127). Example: 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.			
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.

Table 90: **BITSLICE_CONTROL Ports** *(cont'd)*

Port	I/O	Synchronous Clock Domain	Description
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

The following table lists BITSLICE_CONTROL attributes. Most of these attributes have an equivalent register bit or bits in the RIU.

Table 91: **BITSLICE_CONTROL Attributes**

Attribute	Value	Default	Type	Description
EN_OTHER_PCLK	TRUE FALSE	FALSE	String	Enable inter-nibble clocking. When set to TRUE, the PCLK is sourced from the other BITSLICE_CONTROL in the byte. If this is turned on for one BITSLICE_CONTROL, it cannot be turned on for the other BITSLICE_CONTROL in the same byte.
EN_OTHER_NCLK	TRUE FALSE	FALSE	String	Enable inter-nibble clocking. When set to TRUE, the NCLK is sourced from the other BITSLICE_CONTROL in the byte. If this is turned on for one BITSLICE_CONTROL, it cannot be turned on for the other BITSLICE_CONTROL in the same byte.
SERIAL_MODE	TRUE FALSE	FALSE	String	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. When set to FALSE, the clock or strobe applied to a BITSLICE_0 is used as sample clock. 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.

Table 91: **BITSLICE_CONTROL Attributes** *(cont'd)*

Attribute	Value	Default	Type	Description
RX_CLK_PHASE_P	SHIFT_0 SHIFT_90	SHIFT_0	String	Shifts the P-edge of the read clock by 0 degrees or 90 degrees relative to the captured data. 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. When using SHIFT_90, DELAY_VALUE (RX_BITSLICE) or RX_DELAY_VALUE (RXTX_BITSLICE) must be 0.
RX_CLK_PHASE_N	SHIFT_0 SHIFT_90	SHIFT_0	String	Shifts the N-edge of the read clock by 0 degrees or 90 degrees relative to the captured data. 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. When using SHIFT_90, DELAY_VALUE (RX_BITSLICE) or RX_DELAY_VALUE (RXTX_BITSLICE) must be 0.
INV_RXCLK	TRUE FALSE	FALSE	String	Invert the read or sample CLK applied to BITSLICE_0.
TX_GATING	DISABLE ENABLE	DISABLE	String	Write clock gating. For aligned transmitted data, TX_GATING must set to ENABLE and control the TBYTE_IN from interconnect logic. Read Native Mode Bring-up and Reset for more information. ***Note***: TX_GATING = ENABLE does not stop the clock for BITSLICE_1 and BITSLICE_6. When TX_GATING = ENABLE, TBYTE_IN[3:0] is used to stop the clock for transmit interfaces.
RX_GATING	DISABLE ENABLE	DISABLE	String	Enables read strobe/clock gating. 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. 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. When set to TRUE, the gate is controlled by the BITSLICE_CONTROL.PHY_RDEN input.
READ_IDLE_COUNT[5:0]	0 to 63	0	Decimal	Number of clocks after PHY_RDEN deassertion and before turning off ODT termination. MIG USE ONLY.

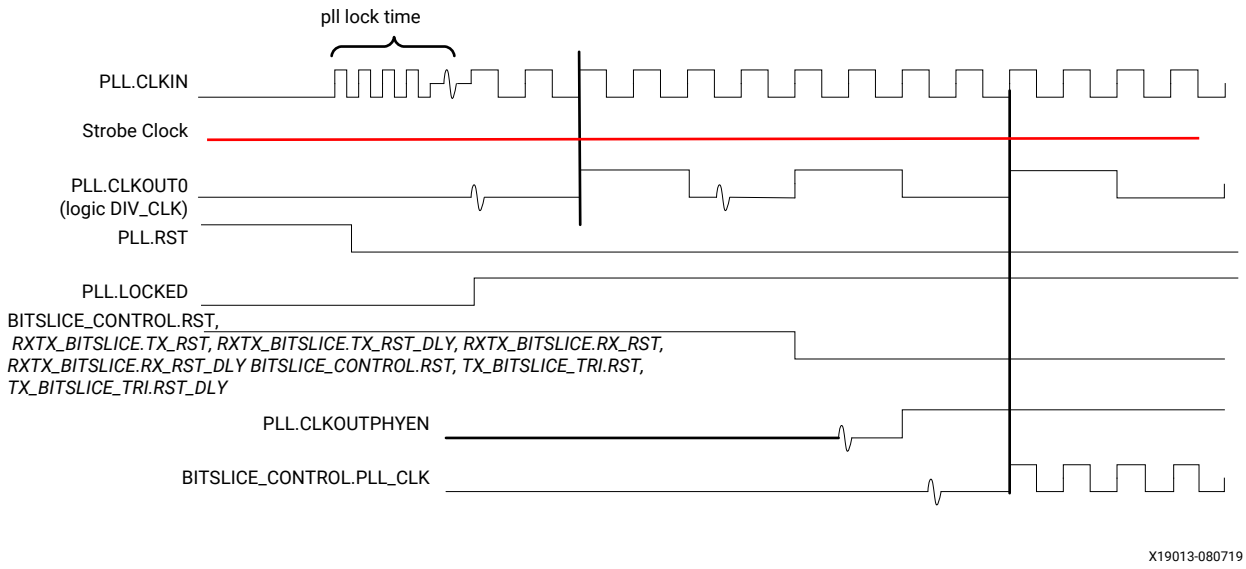
Table 91: **BITSLICE_CONTROL Attributes** *(cont'd)*

Attribute	Value	Default	Type	Description
DIV_MODE	DIV2 DIV4	DIV2	String	Determines how the master clock is divided. When 8-bit mode is used (1:8 serial input), set to DIV4. When 4-bit mode is used, set to DIV2. The FIFO_WRCLK_OUT clock reflects the action of this attribute.
REFCLK_SRC	PLLCLK, REFCLK	PLLCLK	String	When the master clock is the PLL_CLK, this attribute should be set to PLLCLK. When the master clock is the REFCLK input (RX_BITSLICE only), this attribute must be set to REFCLK.
ROUNDING_FACTOR	1, 2, 4, 8, 16, 32, 64, 128	16	Decimal	Rounding factor for BISC. MIG USE ONLY.
CTRL_CLK	EXTERNAL	EXTERNAL	String	Defines the clock source for the RIU interface. Always use the default value, EXTERNAL.
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	ULTRASCALE_PLUS	ULTRASCALE	String	Sets the device version (ULTRASCALE_PLUS)

Native Mode Bring-up and Reset

To bring up a design in a Spartan UltraScale+ device using native SelectIOprimitives, 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 the following figure between PLL/MMCM, BITSLICE_CONTROL, and bit slices.

Figure 136: **PLL and Reset Bring-up Sequence**



In the following figure, 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 136: PLL and Reset Bring-up Sequence, 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 SelectIOwizard 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 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 and Native Output Delay Type Usage 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:

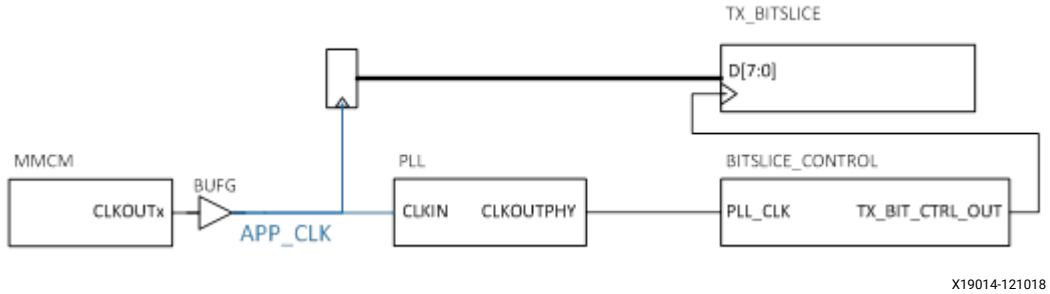
Apply Reset

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 *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930). Then follow the steps in Native Mode Bring-up and Reset 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*. The following figure shows an interface that spans two banks. An application clock (APP_CLK) is used for loading data into the TX_BITSLICE. As shown in the following figure, 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.

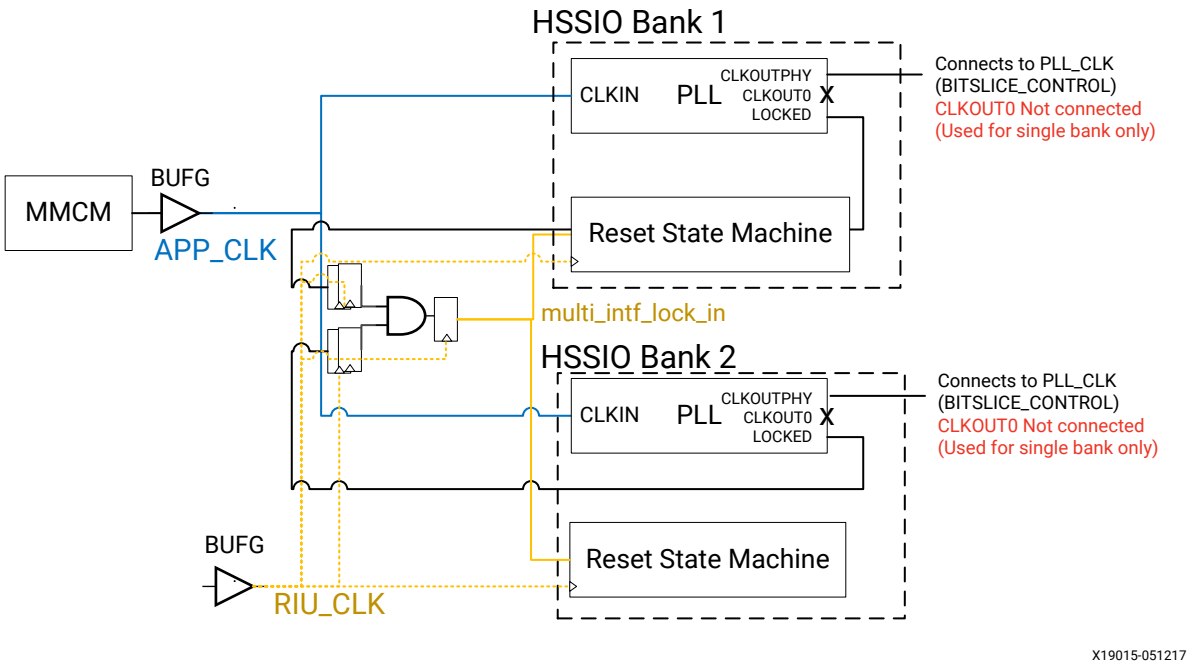
Figure 137: **TX_BITSLICE Application Clock**



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 (see the following figure), 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 138: **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

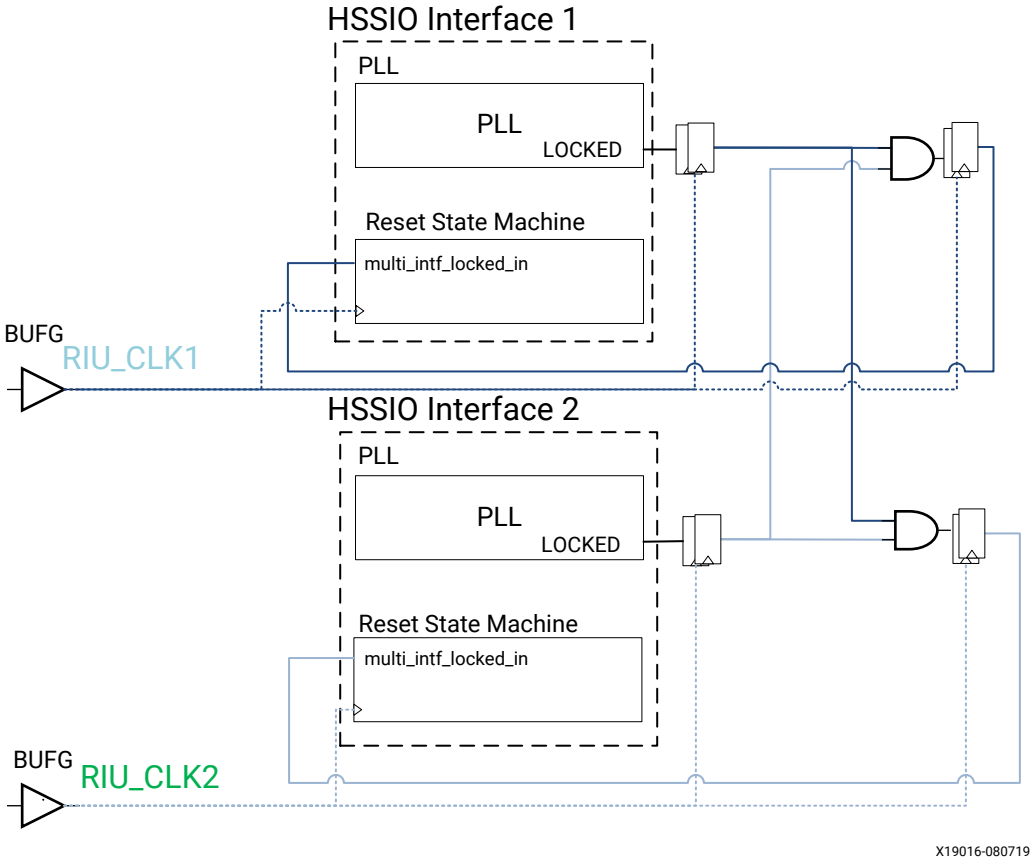
- All PLLs and reset state machines should be reset at the same time.
- LOCKED outputs from all banks must be combined and synchronized to the RIU clock domain. Because the LOCKED signal is an input into the reset state machine, which is driven by the RIU clock domain, the combined multi-bank LOCKED 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.

The following figure 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 139: **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 92: **Pins and Attributes of Native I/O Primitives Involved in Clocking**

Pin or Attribute	I/O	Description
BITSLICE_CONTROL Pins		
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 ½ 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.
• The PLL_CLK or REFCLK are referred to as the BITSLICE_CONTROL master clock. • This master clock is selected by the REFCLK_SRC attribute. • The clock source, PLL_CLK or REFCLK, is mutually exclusive (one or the other but not both).		
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.
BITSLICE_CONTROL Attributes		
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.

Table 92: **Pins and Attributes of Native I/O Primitives Involved in Clocking** *(cont'd)*

Pin or Attribute	I/O	Description
SELF_CALIBRATE		Determines whether or not the clock is going to be tuned to the captured data and tracked over voltage and temperature.
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. Inter-Nibble Clocking has details on how inter-nibble clocking allows clocks or strobes to be shared within a byte.
RXTX_BITSLICE Pins		
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.
RXTX_BITSLICE Attributes		
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.
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.

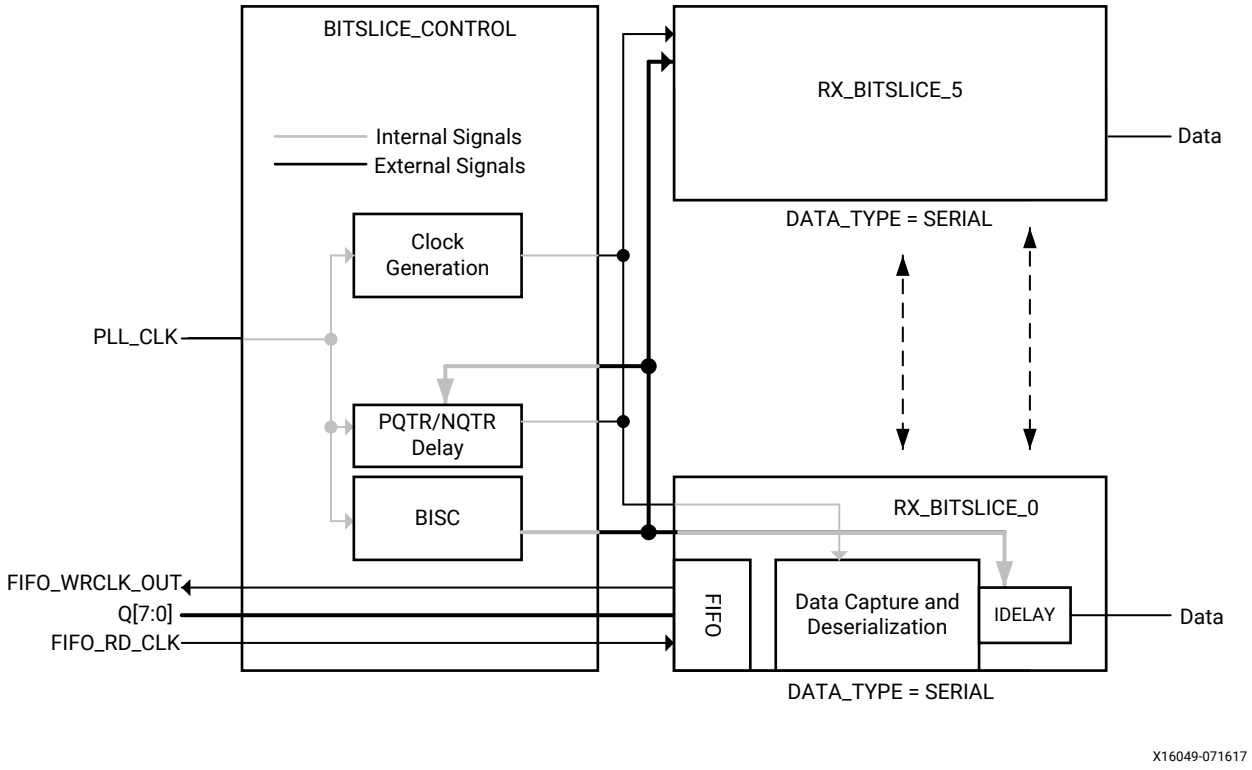
- When the attribute SERIAL_MODE is set to TRUE, the received data is captured using the applied master clock (PLL_CLK or REFCLK). This clock is the DDR capture clock normally running at half the data. Logic in the BITSLICE_CONTROL regenerates and divides the master clock to form required clocks in the 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 the following figure, 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.

Figure 140: **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 78 for an explanation of bitslice numbering within a byte and nibble.*

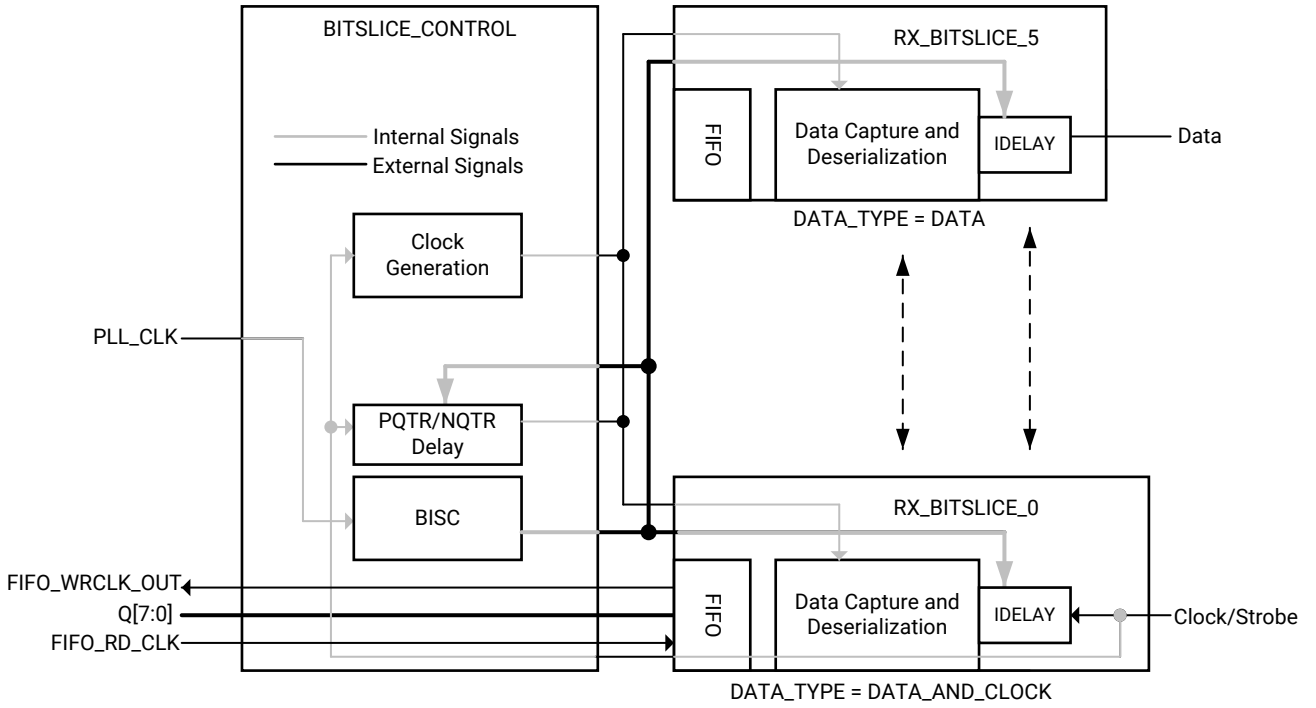
SERIAL_MODE = False

The following figure 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.

Figure 141: **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:

Table 93: **I/Os in One Nibble**

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:

Table 94: **I/Os in One Byte**

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:

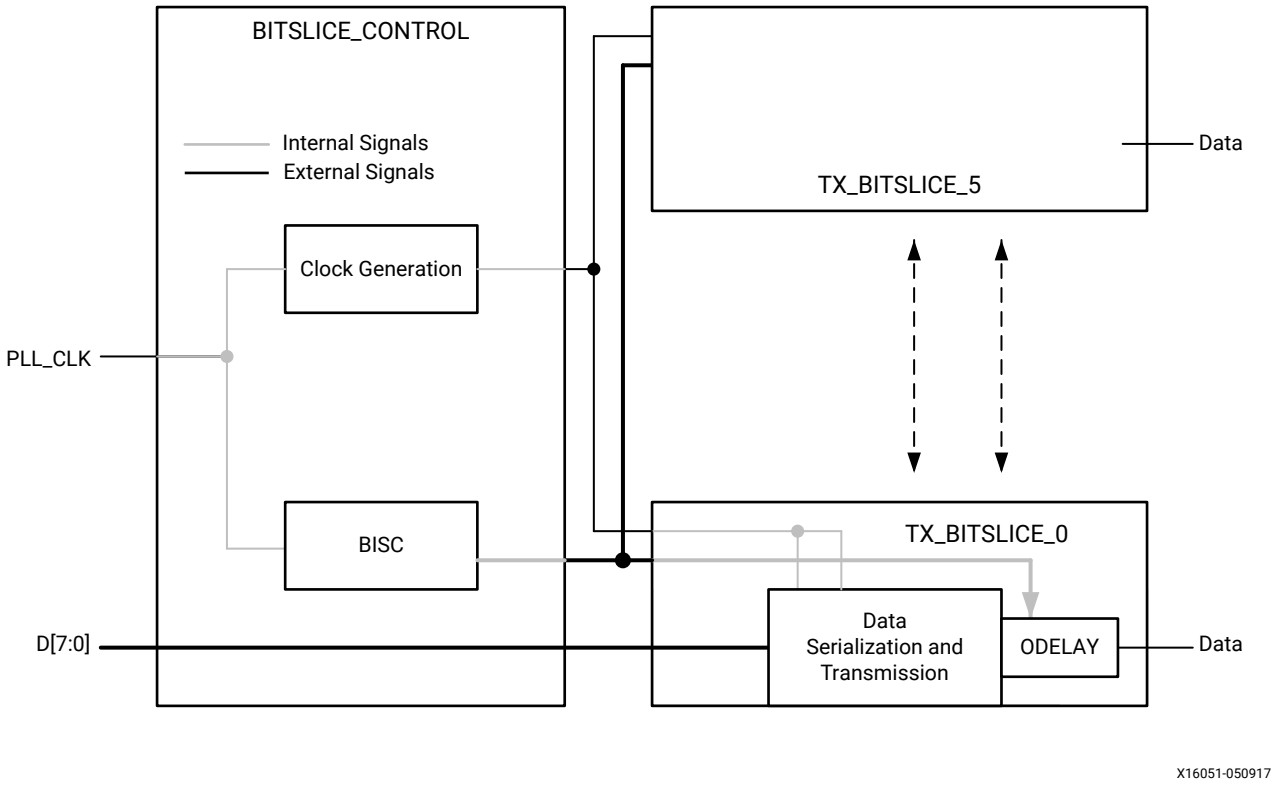
Table 95: **I/Os in One I/O Bank**

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 (see the following figure). 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.

Figure 142: **Data Transmission**



Notes on the preceding figure:

- 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).

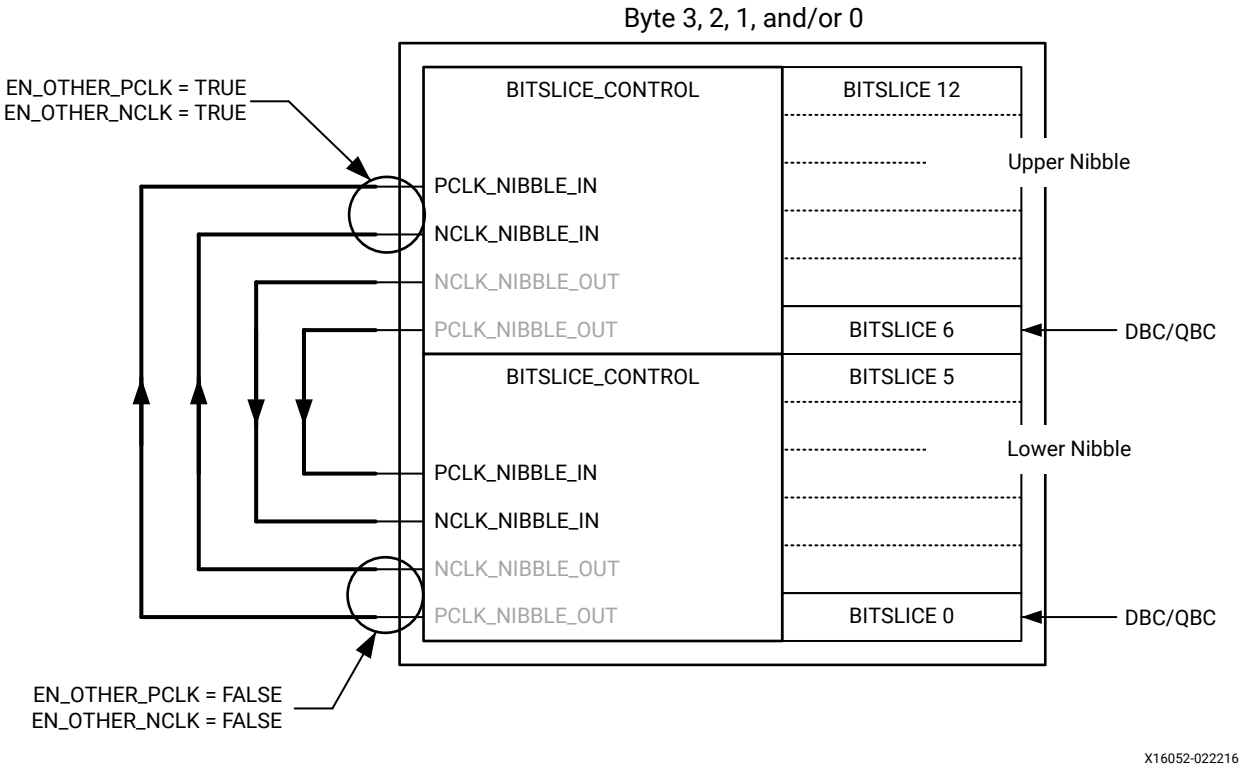
TIP: *Using the OUTPUT_PHASE_90 attribute allows the serial output of a transmitter bit slice to be phase-shifted by 90 degrees. This function is normally used to generate center-aligned interface clocks. When using OUTPUT_PHASE_90, DELAY_VALUE should not be used.*

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 (see the following figure).

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.

Figure 143: **Inter-Nibble Clocking**



Using the previous figure 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

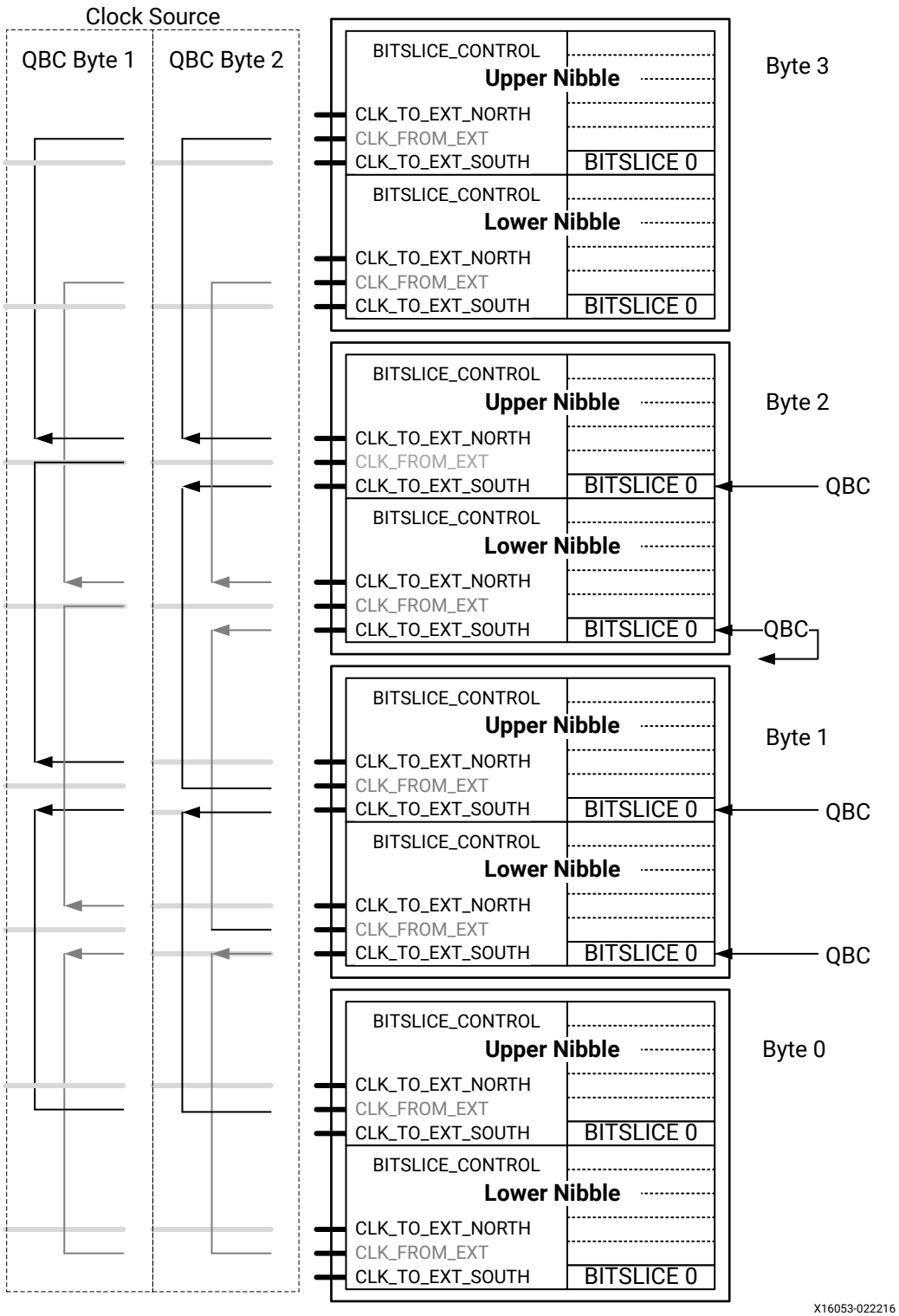
***Note*:** It is possible to enable inter-nibble clock routing in both directions so that bit slices in both nibbles can be used by one of the clocks connected to the byte. In other words, it is possible to capture data in the lower nibble at the clock applied to the upper nibble BITSLICE_0 while at 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 the preceding figure. 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 (see Figure 143). 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).

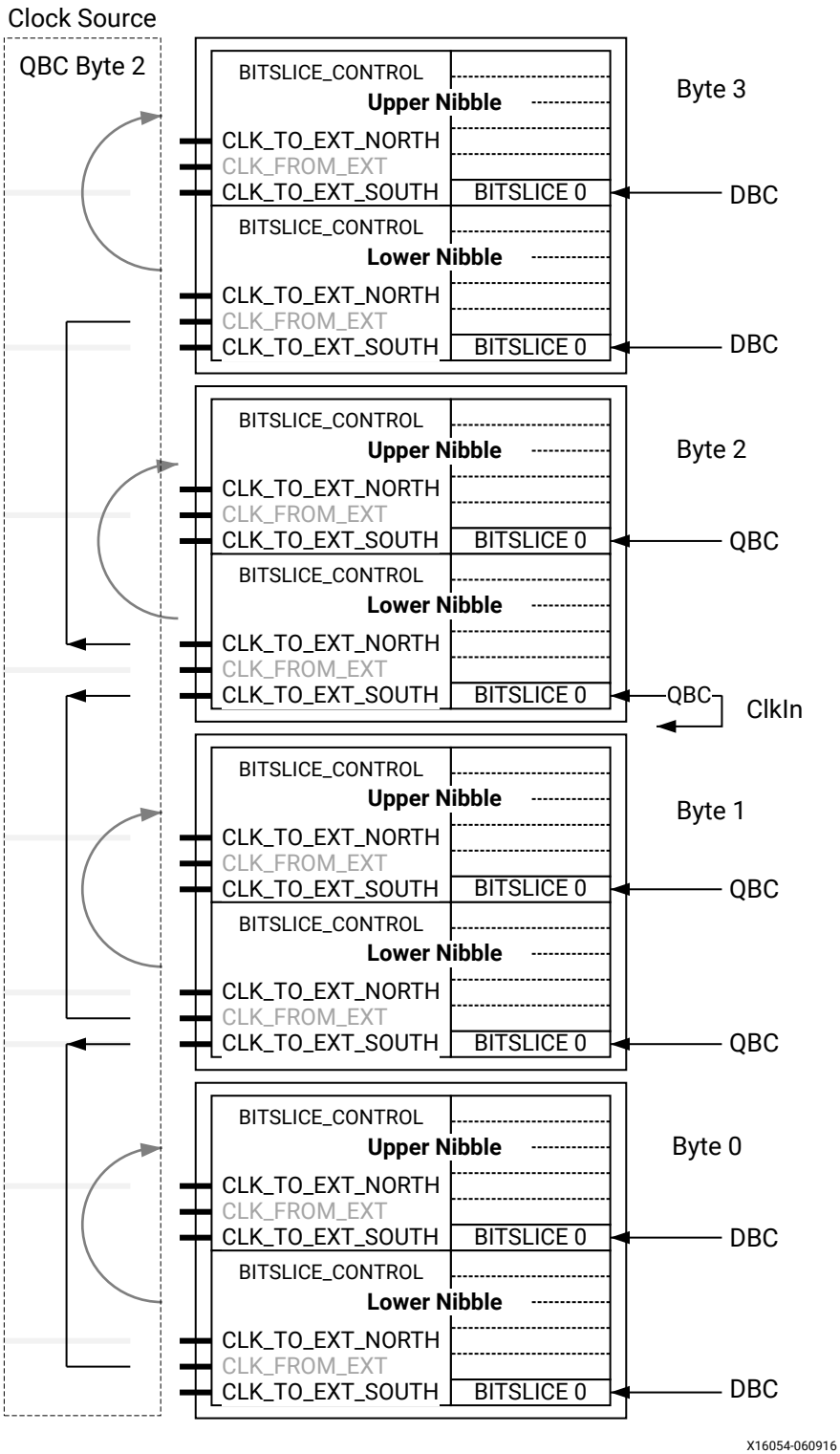
Figure 144: **Inter-Byte Clocking**



***Note*:** The preceding and the following figures 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 the following figure).

Figure 145: **Combination of Inter-Nibble and Inter-Byte Clocking**



Notes on the preceding figure.

- 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 the following tables.

Table 96: **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 the following table.

Table 97: **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 a upper or lower byte and then routed in the byte to the upper nibble by inter-nibble clocking paths as shown in the following figure:

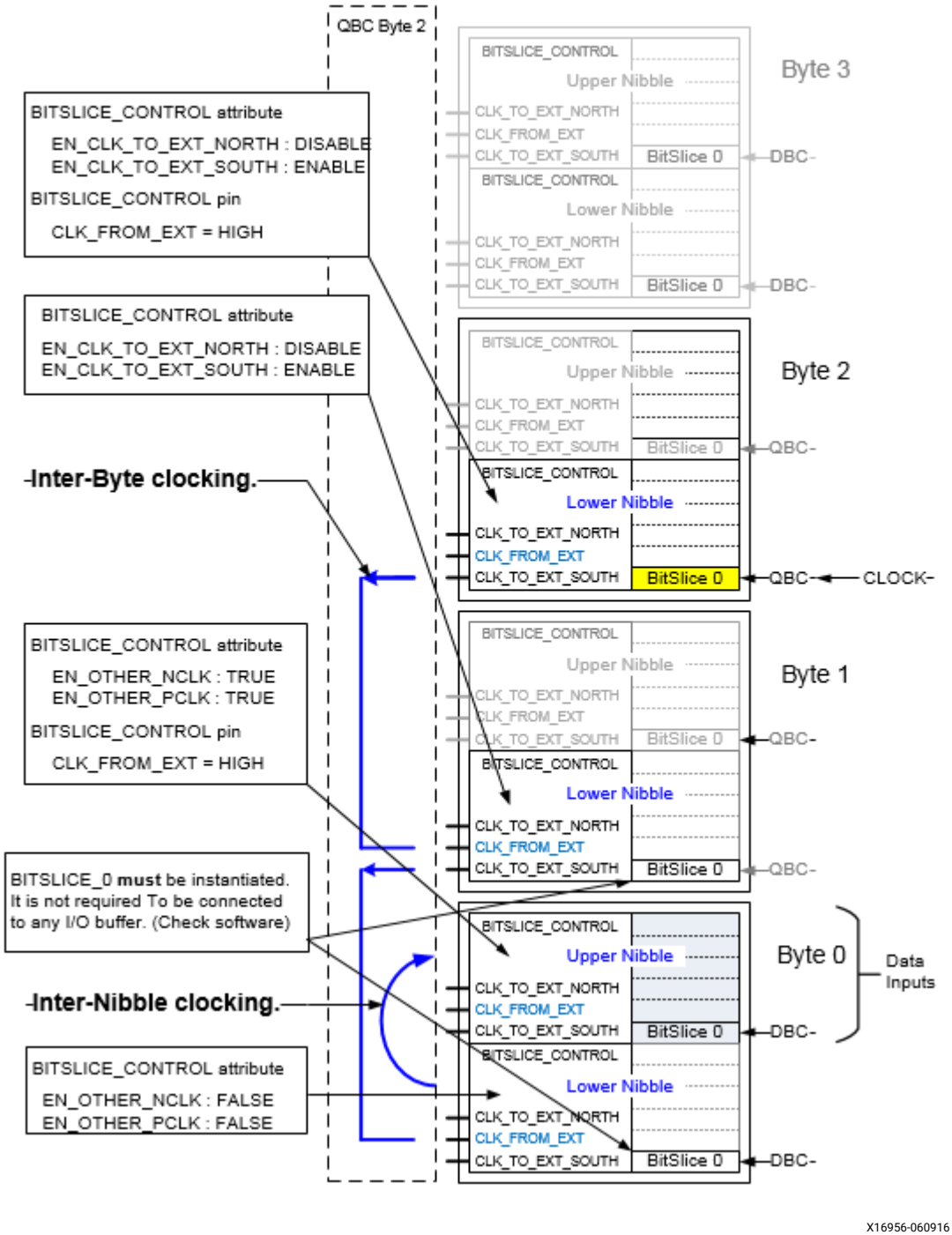
- It is assumed that the BISC controller in the BITSLICE_CONTROL primitive is turned on by SELF_CALIBRATE = ENABLE.
- The received clock arrives at the lower nibble BITSLICE_0 of byte_2 of the I/O bank.
- The bit slices used to capture data are placed in the upper nibble of byte_0 in the I/O bank.

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 78 for an explanation of bitslice numbering within a byte and nibble.*

Figure 146: **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.

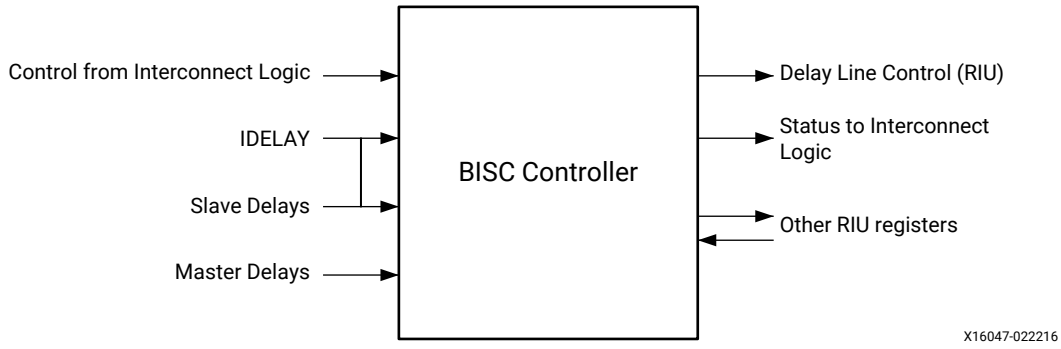
Built-in Self-Calibration

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 (see the following figure).

The BISC controller registers can also be accessed through the register interface unit (RIU). This give 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 147: **Block Diagram of the BISC Controller and Connections**



Ports and Attributes Influencing BISC

The following figure highlights the BITSLICE_CONTROL ports and attributes involved in the BISC process.

Table 98: **BITSLICE_CONTROL Ports and Attributes for the BISC Process**

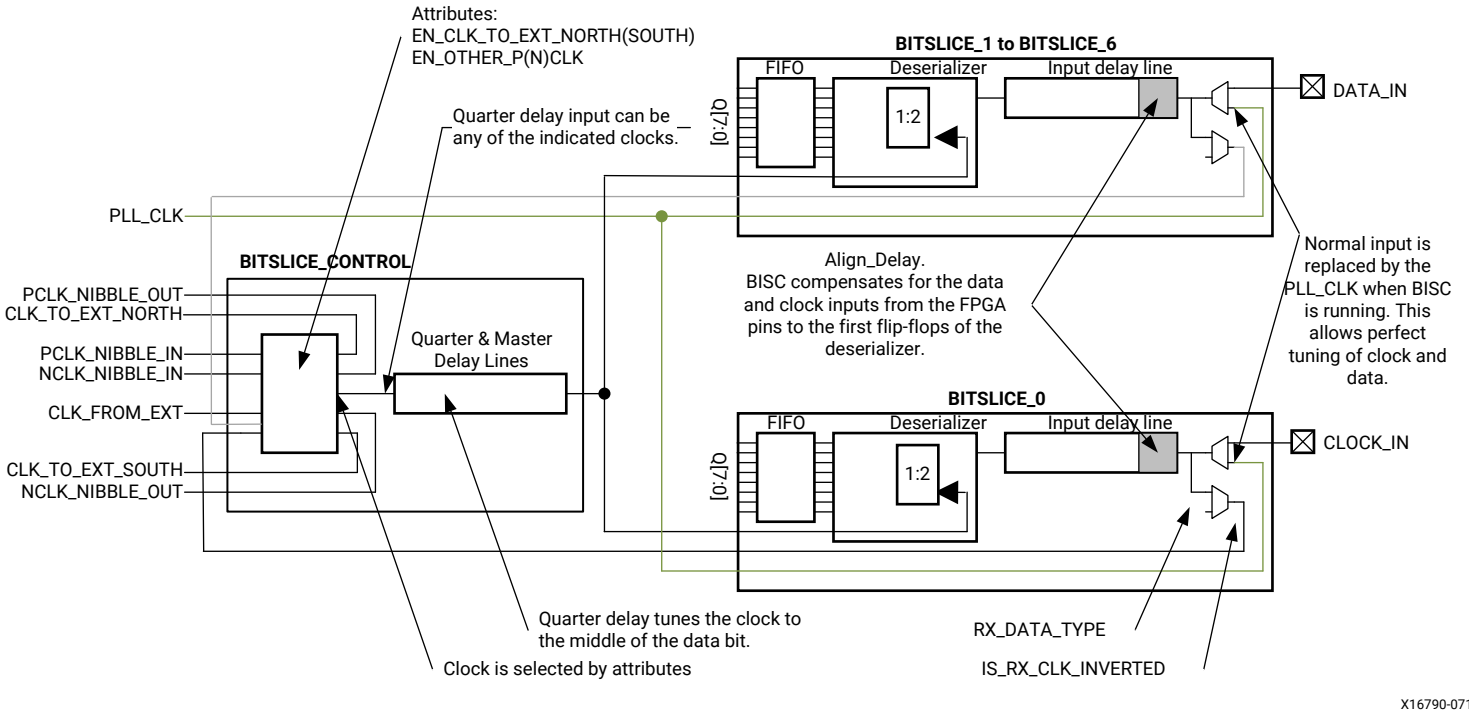
Pins	I/O	Type	Description
Logic Control			
EN_VTC	In	Data	Enable VT tracking.
Status			
VTC_RDY	Out	Data	Nibble ready for VT calibration
DLY_RDY	Out	Data	Nibble delay line calibration is complete.
RIU			
RIU_CLK	In	Clock	Clock from interconnect logic. The RIU clock must be connected in for the BISC process 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.
Attributes			
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 the following figure.

Figure 148: **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.

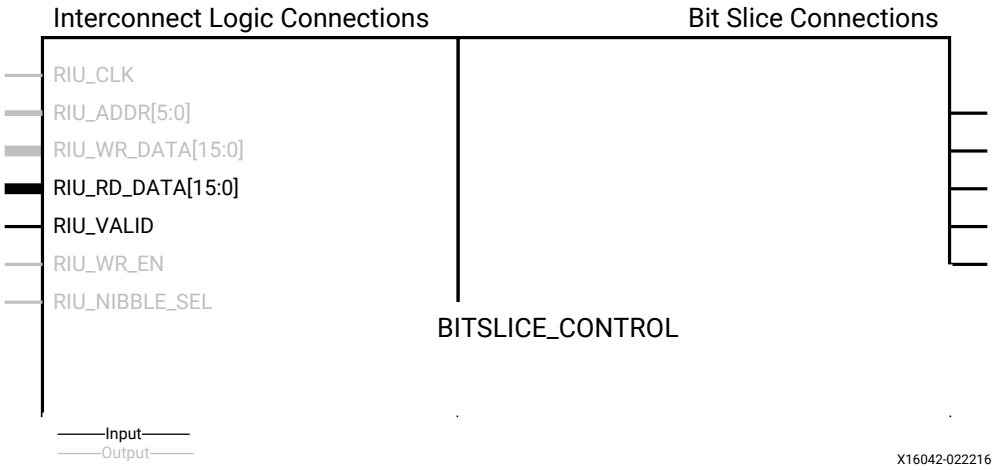
Each nibble has its own BITSLICE_CONTROL component and therefore its own BISC controller. When multiple nibbles or bytes are used and have shared clocks through the inter-nibble or inter-byte clock resources, each nibble can calibrate the bit slices used in that nibble. Each nibble can require different 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 the following figure.

Figure 149: **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. The following figure and table 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 151. The RIU_NIBBLE_SEL pin of each RIU is used as the MSB address, putting the upper nibble in the upper address space.

Figure 150: **RIU Primitive**

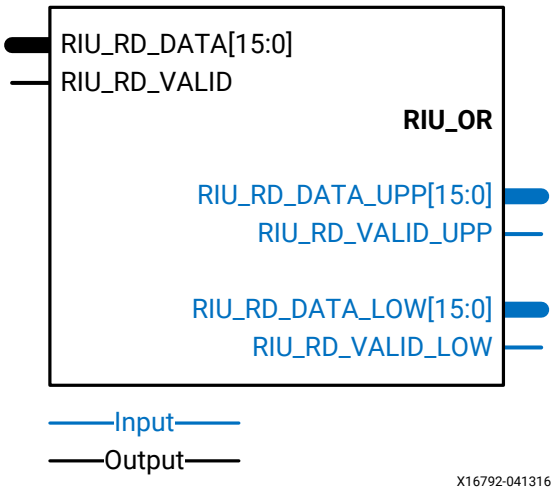


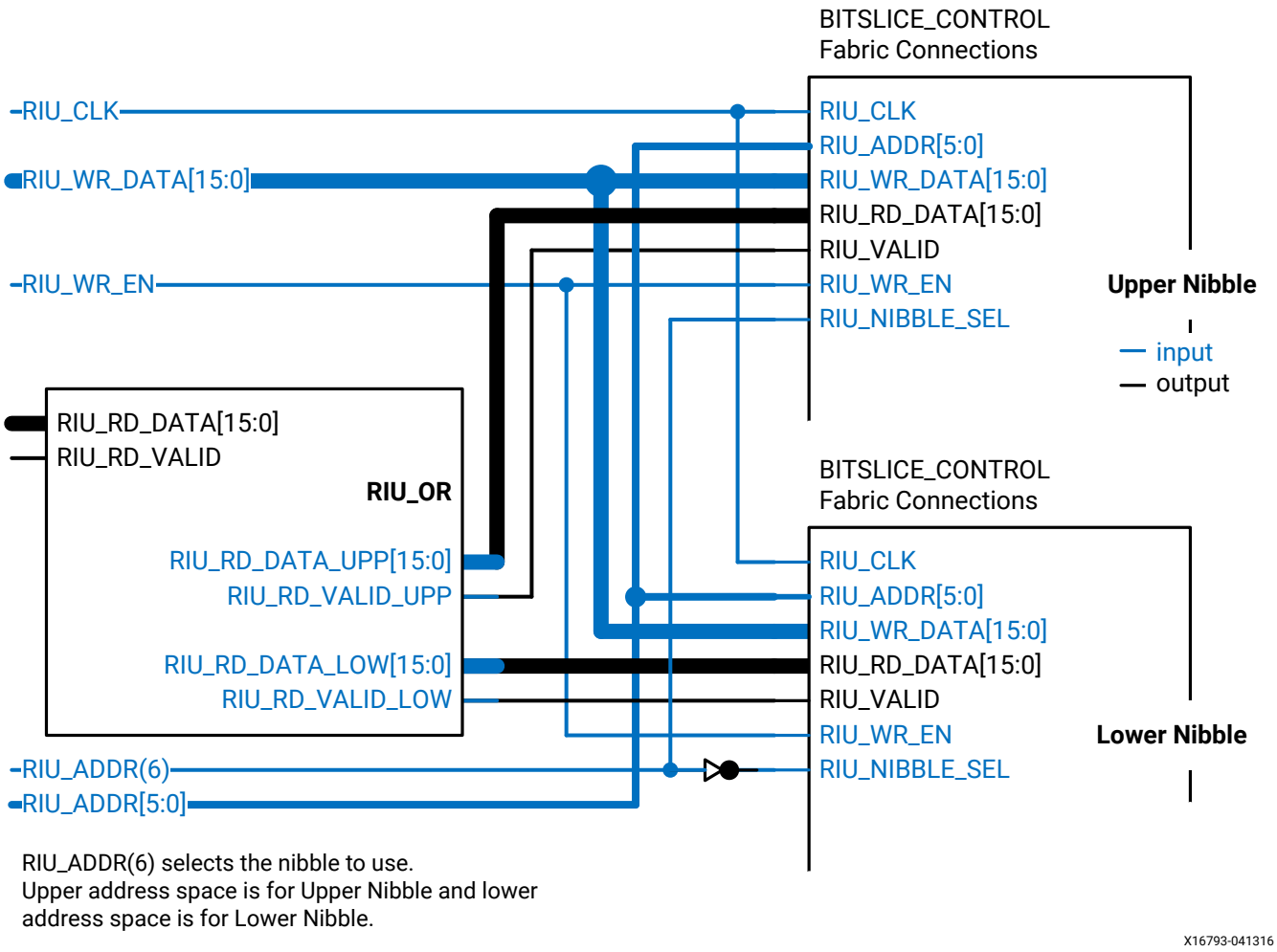
Table 99: **RIU_OR Ports**

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 100: **RIU_OR Attribute**

Attribute	Values	Default	Type	Description
SIM_DEVICE	ULTRASCALE_PLUS	ULTRASCALE	String	Sets the device version (ULTRASCALE_PLUS)

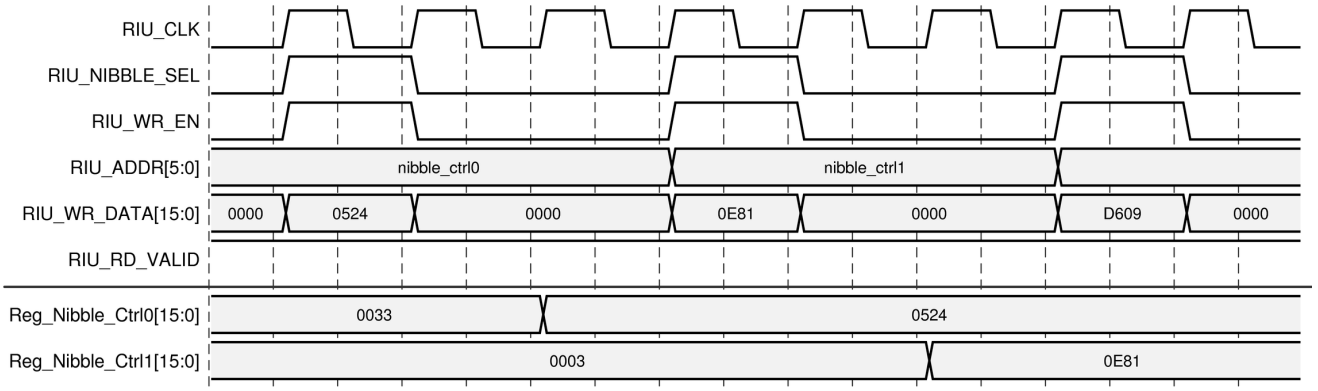
Figure 151: **RIU_OR Combining Two RIU Ports**



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 (see the following figure). 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.

Figure 152: **RIU Write**

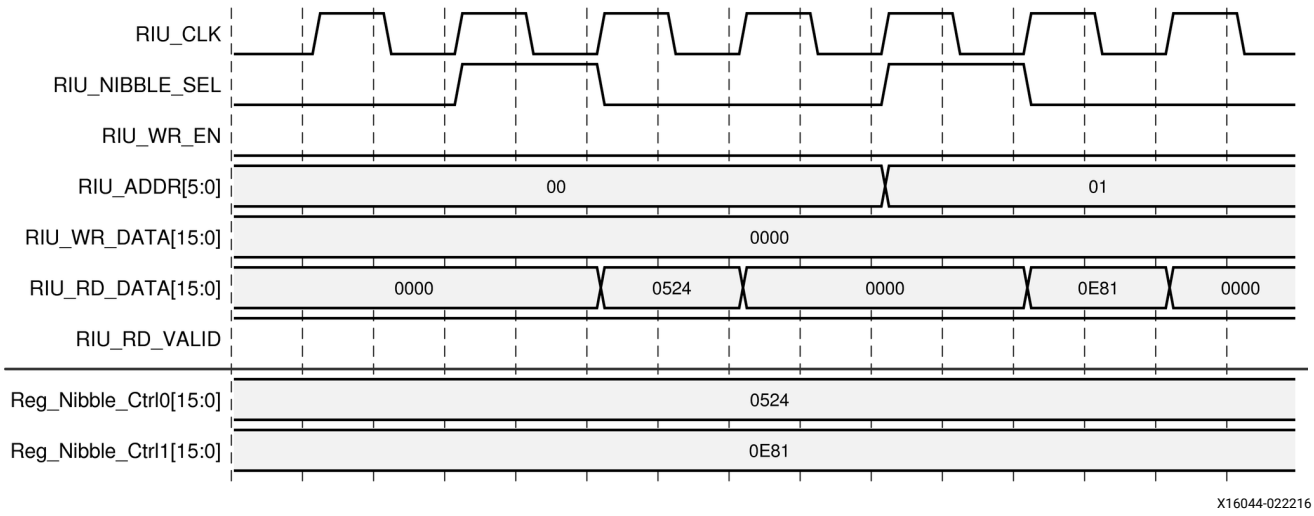


Writing to the RIU registers requires the RIU logic to arbitrate between BISC accesses and access from the interconnect logic. BISC access to RIU registers always has precedence over interconnect logic access, hence an interconnect logic transaction is stored and resumed after BISC access.

RIU Read Actions

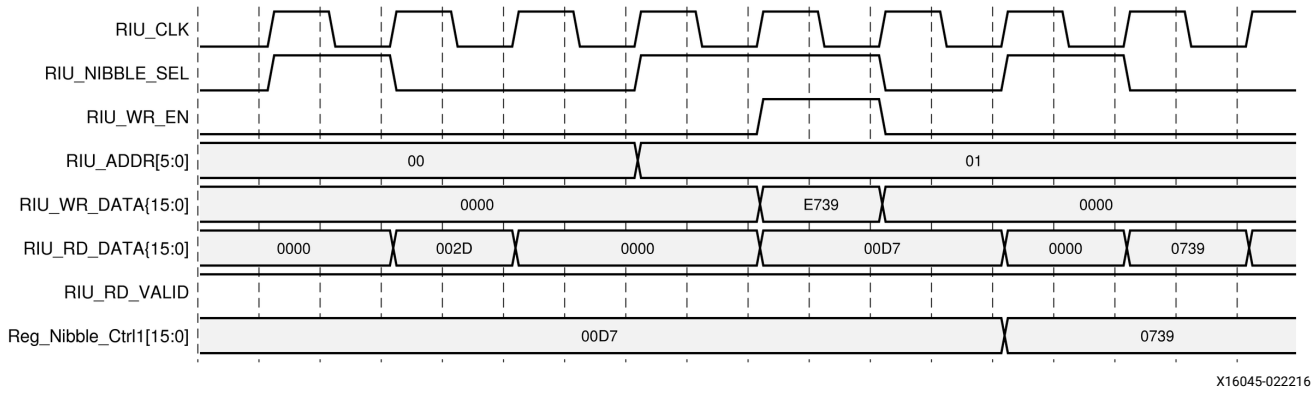
In the RIU read, data from the RIU is driven out on the RIU_RD_DATA bus that depends on RIU_ADDR and the RIU_NIBBLE_SEL signal (see the following figure). When RIU_NIBLE_SEL is asserted, data is presented one clock cycle later on the RIU_RD_DATA bus.

Figure 153: **RIU Read**



The following figure shows a back to back write and read operation.

Figure 154: **RIU Read Modify Write**



In the preceding figure, 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

The following figure lists register interface unit ports.

Table 101: **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.
RIU_WR_DATA[15:0]	Input	Data	Data write to register. 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. RIU_VALID indicates when the RIU port is ready to accept another write. When not used, all bits must be set to zero.

Table 101: **RIU Ports** *(cont'd)*

Pins	I/O	Type	Description
RIU_RD_DATA[15:0]	Output	Data	Data read from register. This output bus provides RIU data. The value of this bus is a representation of the register bits addressed by RIU_ADDR. See Register Definitions and Addresses. The data is presented in the next cycle when RIU_NIBBLE_SEL is active and RIU_WR_EN is 0. When not used, this output bus must be left floating.
RIU_VALID	Output	Data	Status indicating if BISC is accessing RIU registers. 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. During a collision (that is, 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.
RIU_WR_EN	Input	Enable	Register write enable (active-High). This signal and RIU_NIBBLE_SEL must be asserted High to write a register in an RIU interface.
RIU_NIBBLE_SEL	Input	Data	Nibble in byte select. 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.

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).

The following tables list register bit descriptions.

Table 102: **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 *UltraScale Architecture-Based FPGAs Memory IP LogiCORE IP Product Guide* (PG150) for details.															
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.															

Table 102: **Register Bit Description (NIBBLE_CTRL0)** *(cont'd)*

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**
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 103: **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**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**
15:12	**Reserved**															
11:2	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															
1	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.															
0	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 104: **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**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**
15	PAUSE_RDY: VT tracking state machine pause indication.															
14	DIS_VTTRACK_QTR: Enable or disable auto VT tracking for the slave PQTR/NQTR delays.															
13	BSC_RESET: Software reset for BISC.															
12	PHY_RDY: PHY calibration complete status. This is the RIU equivalent of the VTC_RDY signal.															
11	FIXDLY_RDY: Fixed delay calibration complete status. This is the RIU equivalent of the DLY_RDY signal.															
3:10	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.															
2	DIS_VTTRACK_OBIT: Enable or disable auto VT tracking for all output delay lines.															
1	DIS_VTTRACK_IBIT: Enable or disable auto VT tacking for all input delay lines.															
0	CALIBRATE: Turn self-calibration on or off.															

Table 105: **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**	**R/W**	**R/W**	**R/W**	**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). No longer supported.															
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 106: **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**	**R/W**	**R/W**	**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). ***Note*:** BISC continuously increments and/or decrements the delay during self-calibration so be careful before writing to input delay lines.

Table 107: **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 Table 110.
14	DEC: Decrement. See Table 110.
13	CRSE: See Table 110. 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 108: **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 Table 110.															
14	DEC: Decrement. See Table 110.															
13	CRSE: See Table 110. 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 109: **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**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**
15	INC: Increment. See Table 110.															
14	DEC: Decrement. See Table 110.															
13	CRSE: See Table 110.															
12:10	**Reserved**															
9:0	MON: Monitor delay of 0 to 1023 taps.															

Table 110: **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.

Table 110: **RIU Delay Adjustments (PQTR, NQTR, MON)** *(cont'd)*

INC	DEC	CRSE	RIU Action
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 111: **Register Bit Description (ODELAYxx)**

ODELAYxx													ADDR: 0x0A–0x11			
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 Table 113.															
14	DEC: Decrement delay. See Table 113.															
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 112: **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**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**
15	INC: Increment delay. See Table 113.															
14	DEC: Decrement delay. See Table 113.															

Table 112: **Register Bit Description (IDELAYxx)** *(cont'd)*

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**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**
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 113: **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 114: **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																
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 115: **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 116: **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**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**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 117: **Register Bit Description (RL_DLY_RNK)**[1]

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**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**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.															

Table 117: **Register Bit Description (RL_DLY_RNK)[1]** *(cont'd)*

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	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
8:0	RL_DLY_FINE: Fine delay adjustment on read strobe/clock gate.															

Notes:

1. 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 118: **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 119: **Register Bit Description (RL_DLY_RATIO)**

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
RL_DLY_RATIO													ADDR: 0x35			
15:0	RL_DLY_RATIO: This stores the RATIO value after calibration. Used by BISC for strobe/clock gating and VT tracking.															

Table 120: **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 121: **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 122: **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_WR_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 123: **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 124: **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:

1. There is no address decoding and no registers for addresses ranging from 0x3B to 0x3F.

Table 125: **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

Table 125: **RIU Registers and Their Corresponding Attributes** *(cont'd)*

Attribute	Register	Default	Description	Position	Bits
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
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

HD I/O Interface Logic Resources

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 lower data rates. 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. Chapter 2: SelectIO IOB Technology and Supported Standards highlights the features supported in HD I/O banks.

HD I/O pins contain an I/O interface (IOI) logic 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)

Spartan 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
- SAME_EDGE Mode
- SAME_EDGE_PIPELINED Mode

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 82 shows a block diagram of the IDDRE1 primitive. Table 62 lists the IDDRE1 ports and Table 63 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)

Spartan 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 83. Figure 84 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 64 lists the ODDRE1 ports and ODDRE1 Attributes 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>]
```

Chapter 5

XP5IO I/O Interface Logic Resources

XP5IO contains a PHY to support the high-performance I/O interfaces like LPDDR5 and MIPI D-PHY. There are eleven PHY nibbles in an XP5IO bank, with each PHY nibble containing six NIBBLESLICEs that transmit and/or receive data from six individual I/O pins, for a total of 66 pins per bank. Each NIBBLESLICE is composed of a serializer, deserializer, I/O delays, and receiver FIFO. The PHY is equipped with voltage and temperature compensation (VTC) and a mechanism for automatic delay adjustment for optimal data eye centering through the built-in self-calibration (BISC) feature in each PHY nibble. I/O delays can also be controlled through the programmable logic. Control of the PHY features is available through the register interface unit (RIU) in each nibble. Although the PHY can by bypassed in XP5IO, there are no additional I/O resources in the XP5IO and fabric resources must be used.

The XP5IO PHY is used to support the following applications:

- LPDDR5 integrated memory controllers
- MIPI D-PHY v1.2
- Custom PHY-based interfaces

***Note*:** All PHY-based interfaces must be designed using the IP catalog of the Vivado tools.

XP5IO PHY Nibble

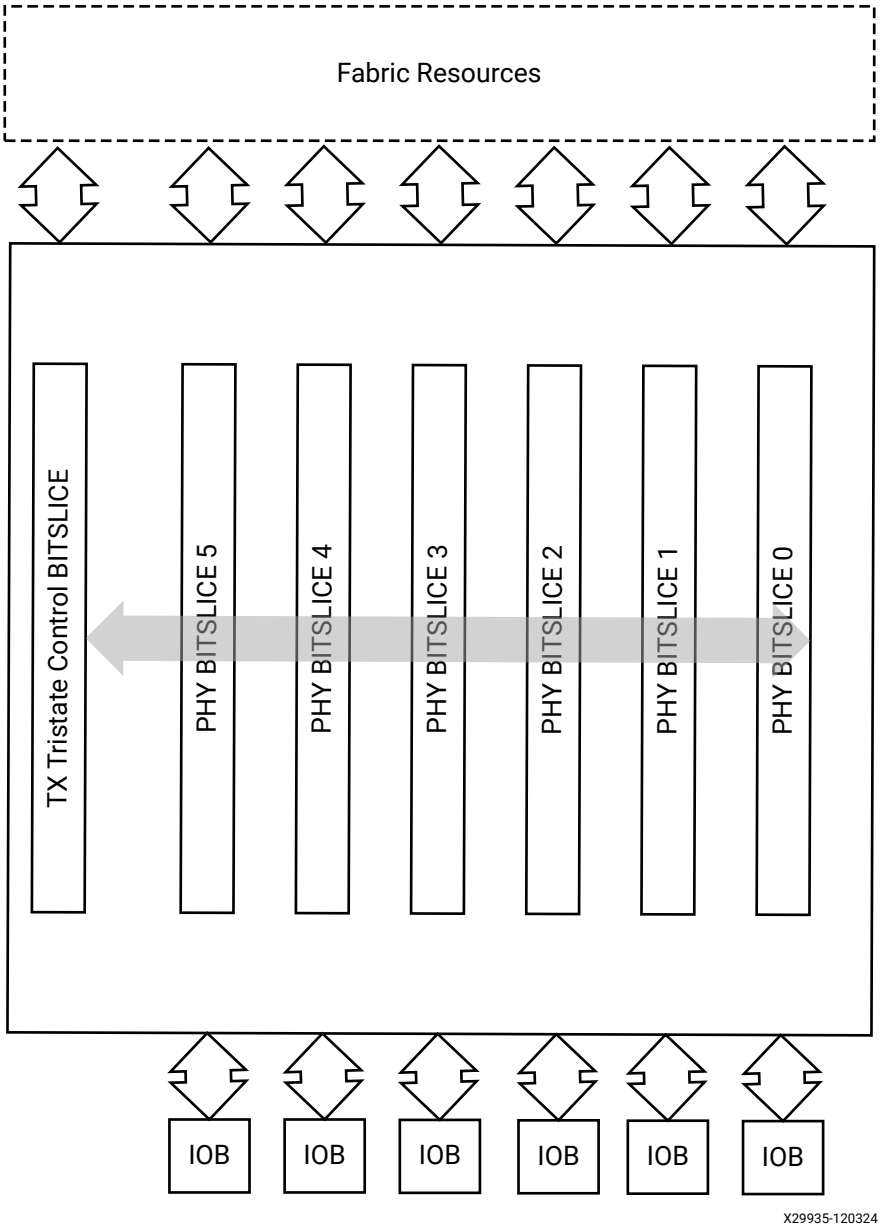
The XP5IO PHY is the high-performance I/O interface for an XP5IO bank. Every XP5IO bank has eleven nibbles. Each nibble is defined as six NIBBLESLICEs and its associated features. NIBBLESLICEs contain input and output logic and are composed of a serializer, deserializer, I/O delays, and a receiver FIFO. NIBBLESLICEs can operate as a transmitter, receiver, or bidirectional circuit. A nibble also performs the following functions/features:

- Built-in self-calibration (BISC) aids in alignment and uses voltage and temperature compensation (VTC) to adjust delay lines
- Generates clocks for the receiver and transmitter functions in the NIBBLESLICEs
- Gives access to the register interface unit (RIU) that provides access to all features of a PHY nibble

- Tristate control
- TX to RX loopback
- Serial mode, which supports receiver interfaces where the clock and data phase relationship is unknown (any interface that is not source-synchronous)

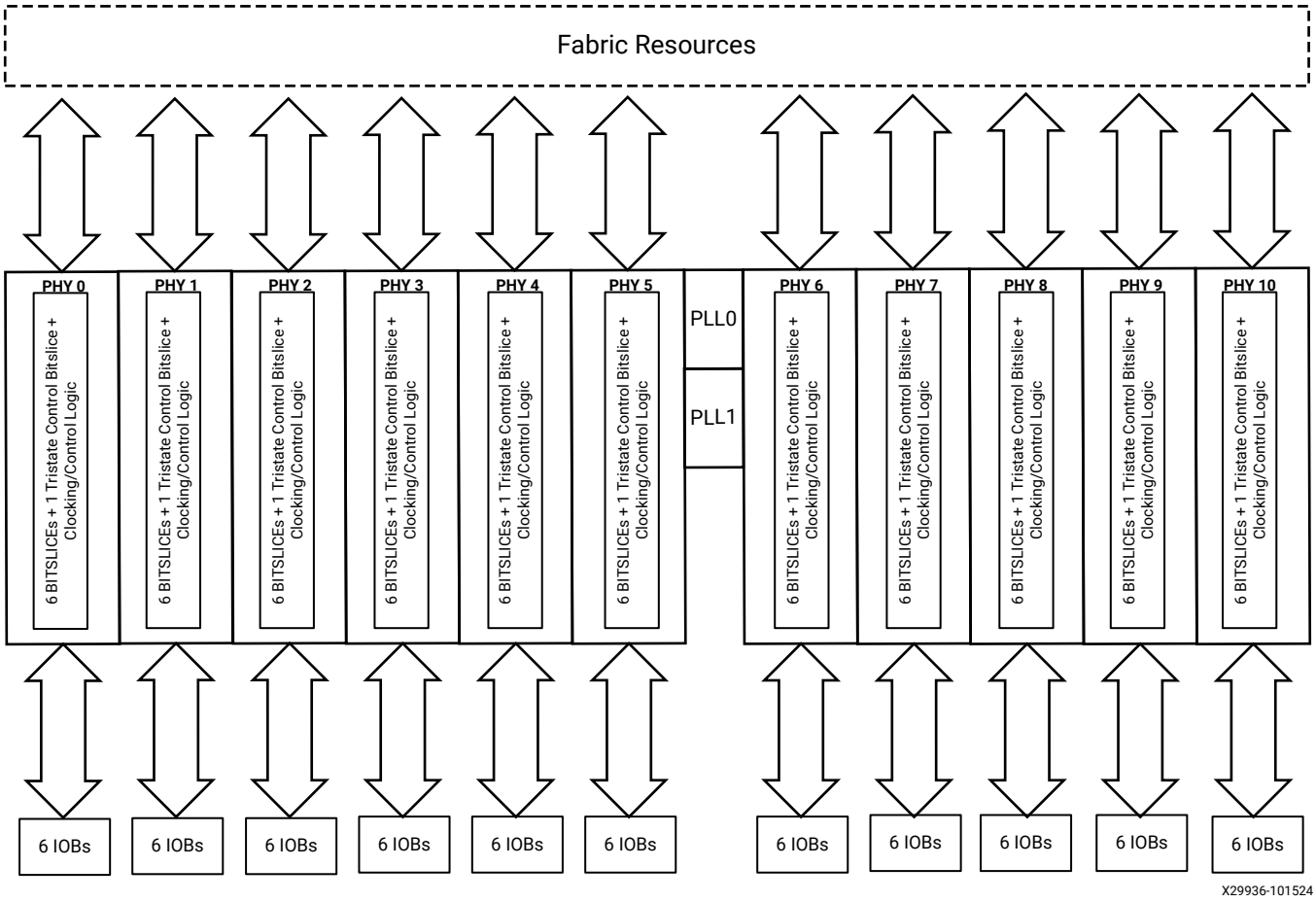
The layout of nibbles in an XP5IO bank is represented in the following figure.

Figure 155: **Relationship Between a Single Nibble, Fabric, and IOB**



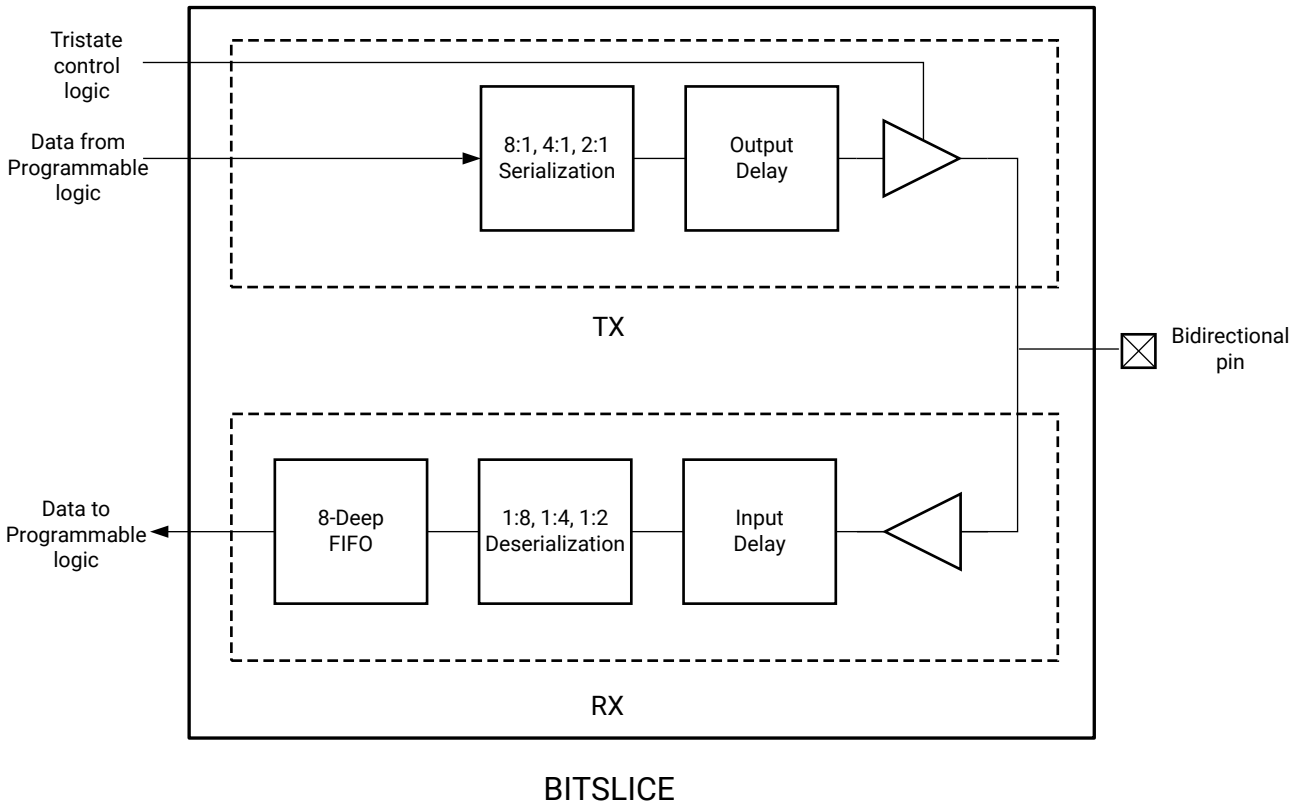
The following figure is a detailed view of the previous figure, representing the relationship between a single nibble XP5IO bank.

Figure 156: **Relationship of Nibbles within an XP5IO Bank**



The following figure shows a NIBBLESLICE.

Figure 157: **XP5IO NIBBLESLICE with TX and RX Datapaths**



AMD

Appendix A

Termination Options for Simultaneous Switching Noise Analysis

Termination Options

The AMD 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.

The following table lists all of the default terminations for each of the I/O standards supported by Spartan 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 126: **Default Terminations for SSN Noise Analysis by I/O Standard**

I/O Standard	Drive	Termination Option
DIFF_HSTL_I	–	Far V_{TT} 40 Ω
DIFF_HSTL_I_12	–	Far V_{TT} 40 Ω
DIFF_HSTL_I_DCI_12	–	Far V_{TT} 40 Ω
DIFF_HSTL_I_18	–	Far V_{TT} 50 Ω
DIFF_HSTL_I_DCI	–	Far V_{TT} 40 Ω
DIFF_HSTL_I_DCI_18	–	Far V_{TT} 50 Ω
DIFF_HSTL_II	–	Near V_{TT} 50 Ω & Far V_{TT} 50 Ω
DIFF_HSUL_12	–	None
DIFF_HSUL_12_DCI	–	None
DIFF_POD10	–	Far V_{CCO} 40 Ω
DIFF_POD10_DCI	–	Far V_{CCO} 40 Ω

Table 126: **Default Terminations for SSN Noise Analysis by I/O Standard** *(cont'd)*

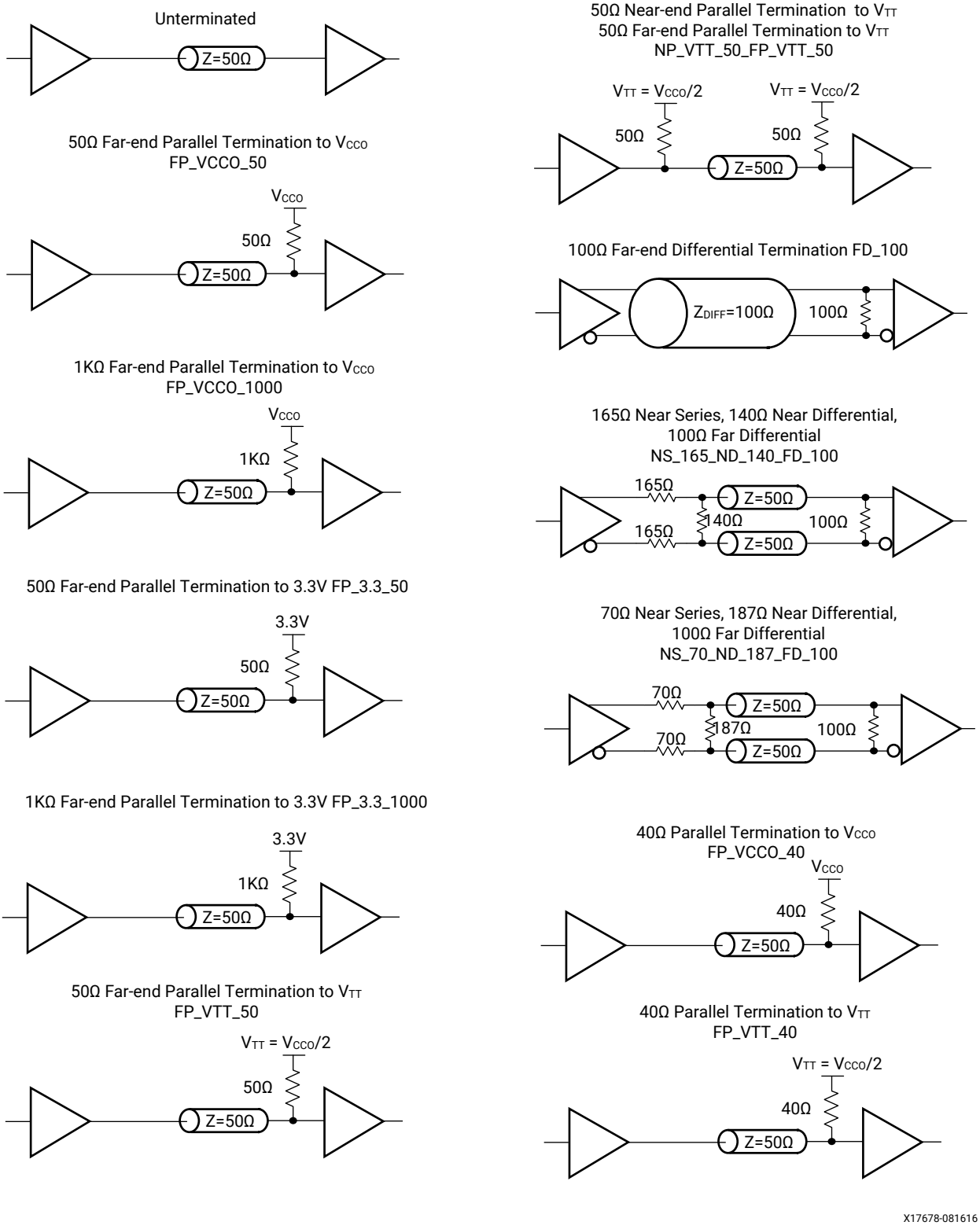
I/O Standard	Drive	Termination Option
DIFF_POD12	–	Far V_{CCO} 40 Ω
DIFF_POD12_DCI	–	Far V_{CCO} 40 Ω
DIFF_SSTL12	–	Far V_{TT} 40 Ω
DIFF_SSTL12_DCI	–	Far V_{TT} 40 Ω
DIFF_SSTL135	–	Far V_{TT} 40 Ω
DIFF_SSTL135_DCI	–	Far V_{TT} 40 Ω
DIFF_SSTL135_R	–	Far V_{TT} 40 Ω
DIFF_SSTL15	–	Far V_{TT} 40 Ω
DIFF_SSTL15_DCI	–	Far V_{TT} 40 Ω
DIFF_SSTL15_R	–	Far V_{TT} 50 Ω
DIFF_SSTL18_I	–	Far V_{TT} 50 Ω
DIFF_SSTL18_I_DCI	–	Far V_{TT} 50 Ω
HSLVDCI_15	–	None
HSLVDCI_18	–	None
HSTL_I	–	Far V_{TT} 40 Ω
HSTL_I_12	–	Far V_{TT} 40 Ω
HSTL_I_DCI_12	–	Far V_{TT} 40 Ω
HSTL_I_18	–	Far V_{TT} 50 Ω
HSTL_I_DCI	–	Far V_{TT} 40 Ω
HSTL_I_DCI_18	–	Far V_{TT} 50 Ω
HSUL_12	–	None
HSUL_12_DCI	–	None
LVCMOS12	2	None
LVCMOS12	4	None
LVCMOS12	6	None
LVCMOS12	8	None
LVCMOS12	12	Far V_{TT} 50 Ω
LVCMOS15	2	None
LVCMOS15	4	None
LVCMOS15	6	None
LVCMOS15	8	None
LVCMOS15	12	Far V_{TT} 50 Ω
LVCMOS15	16	Far V_{TT} 50 Ω
LVCMOS18	2	None
LVCMOS18	4	None
LVCMOS18	6	None
LVCMOS18	8	None
LVCMOS18	12	Far V_{TT} 50 Ω
LVCMOS18	16	Far V_{TT} 50 Ω

Table 126: **Default Terminations for SSN Noise Analysis by I/O Standard** *(cont'd)*

I/O Standard	Drive	Termination Option
LVCMOS25	4	None
LVCMOS25	8	None
LVCMOS25	12	Far V_{TT} 50 Ω
LVCMOS25	16	Far V_{TT} 50 Ω
LVCMOS33	4	None
LVCMOS33	8	None
LVCMOS33	12	Far V_{TT} 50 Ω
LVCMOS33	16	Far V_{TT} 50 Ω
LVDCI_15	–	None
LVDCI_18	–	None
LVDS	–	Far Differential 100 Ω
LVDS_PE	–	Far Differential 100 Ω
LVTTL	4	None
LVTTL	8	None
LVTTL	12	Far V_{TT} 50 Ω
LVTTL	16	Far V_{TT} 50 Ω
MINI_LVDS_25	–	Far Differential 100 Ω
POD10	–	Far V_{CCO} 40 Ω
POD10_DCI	–	Far V_{CCO} 40 Ω
POD12	–	Far V_{CCO} 40 Ω
POD12_DCI	–	Far V_{CCO} 40 Ω
SSTL12	–	Far V_{TT} 40 Ω
SSTL12_DCI	–	Far V_{TT} 40 Ω
SSTL135	–	Far V_{TT} 40 Ω
SSTL135_DCI	–	Far V_{TT} 40 Ω
SSTL135_R	–	Far V_{TT} 40 Ω
SSTL15	–	Far V_{TT} 40 Ω
SSTL15_DCI	–	Far V_{TT} 40 Ω
SSTL15_R	–	Far V_{TT} 50 Ω
SSTL18_I	–	Far V_{TT} 50 Ω
SSTL18_I_DCI	–	Far V_{TT} 50 Ω

The following figure illustrates each of these terminations.

Figure 158: **Default Terminations**



Appendix B

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 provide supplemental material useful with this guide:

1. *UltraScale Architecture and Product Data Sheet: Overview* (DS890)
2. *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930)
3. *UltraScale Architecture SelectIO Resources User Guide* (UG571)
4. *UltraScale and UltraScale+ FPGAs Packaging and Pinouts Product Specification* (UG575)
5. *Spartan UltraScale+ FPGAs Configuration User Guide* (UG860)
6. *Spartan UltraScale+ Libraries Guide* (UG1074)
7. *Vivado Design Suite Properties Reference Guide* (UG912)
8. Electronic Industry Alliance JEDEC web site
9. *Vivado Design Suite User Guide: System-Level Design Entry* (UG895)
10. *UltraScale Architecture Clocking Resources User Guide* (UG572)
11. *UltraFast Design Methodology Guide for FPGAs and SoCs* (UG949)
12. *UltraScale Architecture-Based FPGAs Memory IP LogiCORE IP Product Guide* (PG150)
13. *Bitslip in Logic* (XAPP1208)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
12/23/2024 Version 1.0	
Initial release.	N/A

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