# Versal ACAP PHY for PCI Express v1.0

## *LogiCORE IP Product Guide*

### Vivado Design Suite

**PG345 (v1.0) October 15, 2021**

Xilinx 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.

XILINX®

# Table of Contents

# Introduction

The Xilinx® Versal™ ACAP PHY for PCIe® IP is a building block IP that allows for a PCI Express® MAC to be built as soft IP in the device fabric. The Versal™ ACAP PCIe PHY IP design cannot be migrated to UltraScale™ or UltraScale+™ parts.

***Note*:** IP supports Vivado® IP integrator flow. The GT Quads are always outside the PHY for PCIe IP and connected in IP integrator block design.

## Features

### Versal™ ACAP PHY for PCI Express® Features

- Gen1 (2.5 GT/s), Gen2 (5.0 GT/s), Gen3 (8.0 GT/s), and Gen4 (16.0 GT/s) speeds are supported.
- Versal devices support:
  - 2.5 GT/s, 5.0 GT/s and 8.0 GT/s line rates with x1, x2, x4, x8, and x16 lane operation.

    ***Note*:** x16 lane width support is dependent on available GTYP in the selected device/package.
  - 16.0 GT/s line rate with x1, x2, x4, x8 lane operation.
- GT Quad is always outside in the PHY IP and is available in IP integrator block design on opening the example design.
- Supports PIPE low power state of P0s only at Gen1 or Gen2 speeds. PIPE P0s is equivalent to MAC LTSSM state of L0s.
- Supports synchronous and asynchronous applications.
- Rate change between Gen1 and Gen2 is a fixed datapath implementation.
- Transceiver reset sequence is part of the GT Quad in Versal devices as compared to UltraScale and UltraScale+.
- There is no option to choose the GT Quad location in the PHY IP GUI. You are expected to LOC the suitable GT Quad in the top level constraints file (XDC).

# IP Facts

| LogiCORE™ IP Facts Table | |
|---|---|
| **Core Specifics** | |
| Supported Device Family[1] | Versal™ ACAP |
| Supported User Interfaces | N/A |
| Resources | Performance and Resource Use |
| **Provided with Core** | |
| Design Files | Verilog |
| Example Design | Verilog |
| Test Bench | Verilog |
| Constraints File | Xilinx® Design Constraints (XDC) |
| Simulation Model | Verilog |
| Supported S/W Driver | N/A |
| **Tested Design Flows[2]** | |
| Design Entry | Vivado® Design Suite |
| Simulation | For supported simulators, see the Xilinx Design Tools: Release Notes Guide. |
| Synthesis | Vivado Synthesis |
| **Support** | |
| Release Notes and Known Issues | Master Answer Records: 72289 |
| All Vivado IP Change Logs | Master Vivado IP Change Logs: 72775 |
| Xilinx Support web page | |

**Notes:**

1. For a complete list of supported devices, see the Vivado® IP catalog.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

XILINX

*Chapter 2*

# Overview

## Navigating Content by Design Process

Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. All Versal™ ACAP design process Design Hubs can be found on the Xilinx.com website. This document covers the following design processes:

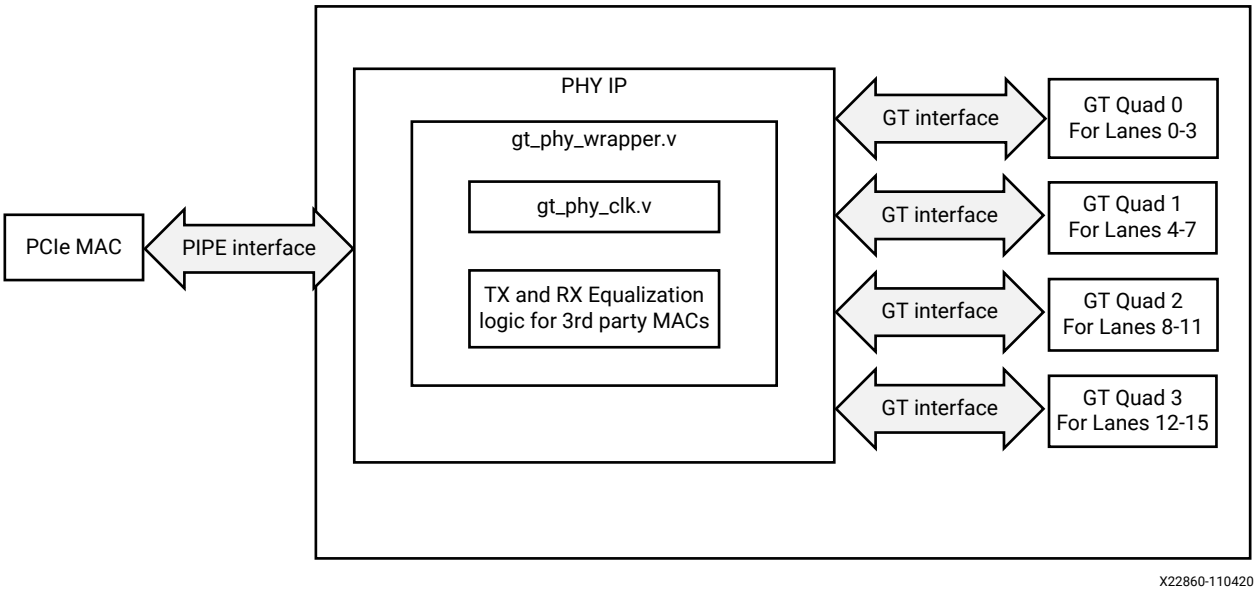
- **System and Solution Planning:** Identifying the components, performance, I/O, and data transfer requirements at a system level. Includes application mapping for the solution to PS, PL, and AI Engine. Topics in this document that apply to this design process include:
  - Chapter 3: Product Specification
- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, functional simulation, and evaluating the Vivado® timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
  - Chapter 4: Designing with the Core
  - Chapter 5: Design Flow Steps
- **Board System Design:** Designing a PCB through schematics and board layout. Also involves power, thermal, and signal integrity considerations.

## Core Overview

This section provides a detailed description of the features, attributes, and signals associated with the PIPE interfaces of the Versal™ ACAP PHY for PCIe® IP.

The following figure is a high-level PCIe PHY IP block diagram.

*Figure 1:* **PCIe PHY IP Block Diagram**




- Lane 0 is always the master.
- Clock block shared by all lanes.
- The Versal ACAP PHY IP communicates with the PCIe MAC through the PIPE interface.

# Feature Summary

## Versal ACAP PHY for PCI Express Support

The following table summarizes the recommended default features supported for different line rates.

*Table 1:* **Default Features Supported**

| Features | Gen1 | Gen2 | Gen3 | Gen4 |
|---|---|---|---|---|
| Line Rate | 2.5 GT/s | 5 GT/s | 8 GT/s | 16 GT/s |
| PCLK Frequency | 125 MHz | 250 MHz | 250 MHz | 500 MHz |
| 2-Byte Data Width | Yes | Yes | | |
| 4-Byte Data Width | | | Yes | Yes |
| 8-Byte Data Width | | | | |
| LPM/DFE Mode | LPM | LPM | DFE [1] | DFE [1] |
| RX Elastic Buffer with Clock Correction | Yes | Yes | Yes | Yes |
| 8b/10b Encoder and Decoder | Yes | Yes | | |

*Table 1:* **Default Features Supported** *(cont'd)*

| Features | Gen1 | Gen2 | Gen3 | Gen4 |
|---|---|---|---|---|
| 128b/130b Encoder and Decoder | | | Yes | Yes |
| N_FTS | 255 | 255 | 255 | 255 |
| Versal™ Device GT Support | GTY | GTY | GTY | GTY |

**Notes:**

1. PLL TYPE is always LCPLL and TXPROGDIV CLOCK SOURCE is always RPLL.

# Unsupported Features

## Versal ACAP PHY for PCI Express Unsupported Features

The following features are not supported in the core:

- Powering down of lane 0 (master) is not supported.
- Per-lane power down is not supported.
- PIPE low power state of P0s is not supported when the max speed is configured as Gen3 or Gen4.
- PIPE low power state of P2 is not supported.
- Bypassing the RX elastic buffer is not supported.
- Preserving of Gen3/Gen4 equalization settings is not supported after the rate change.
- PCIe PHY does not check or monitor for PIPE protocol errors.
- Checking or monitoring for PIPE protocol errors is not supported

# Licensing and Ordering

This Xilinx® LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License.

Information about other Xilinx® LogiCORE™ IP modules is available at the Xilinx Intellectual Property page. For information about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

XILINX

*Chapter 3*

# Product Specification

## Versal ACAP PHY for PCI Express

The Versal™ ACAP PHY for PCI Express® core internally does not instantiate the GTY transceiver like the UltraScale™ or UltraScale+™ PCIe PHY IP. GT Quad in Versal ACAP is always external to the PCIe PHY IP and can also connect to the PCIe PHY IP. GT Quad IP in the Vivado IP using block automation or on opening the example design of the PCIe PHY IP, you can get an integrated design where PCIe PHY IP and GT Quad are connected with the required connections. Versal ACAP GT Quad is highly configurable and tightly integrated with the programmable logic resources.

## Performance and Resource Use

Resources required for the Versal™ ACAP PHY for PCIe® IP are mentioned in the table below. These values are generated using Vivado® Design Suite for the supported devices.

*Table 2:* **Device Utilization**

| Family | Speed | Lane | LUTs | FFs | LUT-FF Pairs |
|---|---|---|---|---|---|
| Versal™ ACAP PHY for PCIe® | Gen1 | x1 | 169 | 116 | 29 |
| | Gen2 | x1 | 169 | 116 | 36 |
| | Gen3 | x1 | 169 | 116 | 36 |
| | Gen4 | x1 | 169 | 116 | 36 |
| | Gen1 | x2 | 277 | 206 | 46 |
| | Gen2 | x2 | 277 | 206 | 48 |
| | Gen3 | x2 | 277 | 206 | 48 |
| | Gen4 | x2 | 277 | 206 | 48 |
| | Gen1 | x4 | 511 | 386 | 85 |
| | Gen2 | x4 | 509 | 386 | 84 |
| | Gen3 | x4 | 509 | 386 | 84 |
| | Gen4 | x4 | 509 | 386 | 84 |
| | Gen1 | x8 | 973 | 745 | 165 |
| | Gen2 | x8 | 973 | 745 | 171 |
| | Gen3 | x8 | 973 | 745 | 171 |
| | Gen4 | x8 | 973 | 745 | 171 |
| | Gen1 | x16 | 1893 | 1465 | 318 |
| | Gen2 | x16 | 1894 | 1465 | 335 |
| | Gen3 | x16 | 1894 | 1465 | 335 |

# Minimum Device Requirements

*Table 3:* **GTY-Based Soft PHY for PCIe Maximum Configurations (Versal Prime, Versal AI Core, Versal AI Edge)**

| Speed Grade | -1 | -1 | -2 | -2 | -2 | -3 |
|---|---|---|---|---|---|---|
| **Voltage Grade** | **L (0.70V)** | **M (0.80V)** | **L (0.70V)** | **M (0.80V)** | **H (0.88V)** | **H (0.88V)** |
| Gen1 (2.5 GT/s per lane) | x16 | x16 | x16 | x16 | x16 | x16 |
| Gen2 (5 GT/s per lane) | x16 | x16 | x16 | x16 | x16 | x16 |
| Gen3 (8 GT/s per lane) | x16 | x16 | x16 | x16 | x16 | x16 |
| Gen4 (16 GT/s per lane) | x8 | x8 | x8 | x8 | x8 | x8 |

# Port Descriptions

The following tables describe the supported PIPE signals by the PCIe PHY IP. For additional details, refer to the PIPE specification. The signals described in this section are based on a single lane application. Signals can be per-lane, or per-design. If not indicated in the description, the default is per-design. Per-design indicates that one signal controls all lanes (0 to N-1 Lane).

A per-lane signal on the PCIe PHY IP is in a form of {LaneN-1[Width-1:0], ...Lane1 [Width-1:0], Lane0[Width-1:0]}.

## Clock and Rest Signals Interface Ports

*Table 4:* **Clock and Reset Signals**

| Port Name | Width | I/O | Clock | Description |
|---|---|---|---|---|
| pcie_refclk | 1 | Input | refclk | Reference clock for fabric logic. The recommended reference clock is 100 MHz. This clock is expected to be free running and stable. This reference clock can be either synchronous or asynchronous. In synchronous mode, the PPM is 0. In asynchronous mode, the PPM is up to ±300 or 600 PPM worst case.The refclk differential pair are connected to the IBUFDS in the PHY IP. One of the IBUFDS_GTE5 outputs are connected to the soft modules in PHY_IP -(Phy_Wrapper). Other output is connected to the BUFG_GT. The BUFG_GT output (phy_gtrefclk) is used as a Reference clock for GT_Quads.<br>• 100 MHz (default)<br>• 125 MHz<br>• 250 MHz |
| phy_gtrefclk | 1 | Input | refclk | Reference clock for GT_QUADS. This clock must be driven directly from an IBUFDS_GTE5. Same definition and frequency as phy_refclk. |
| sys_reset | 1 | Input | Asynchronous | When logic Low, this signal resets the PHY. This must be connected to PCIe PERST_N. Polarity is Active Low. |
| phy_coreclk | 1 | Output | coreclk | Core clock options:<br>• 250 MHz<br>• 500 MHz |
| phy_userclk | 1 | Output | userclk | User clock options:<br>• 62.5 MHz<br>• 125 MHz<br>• 250 MHz<br>• 500 MHz<br>phy_userclk is edge-aligned and phase-aligned to phy_coreclk. |
| phy_userclk2 | 1 | Output | userclk | This clock is same as Core clock or user clock depending on the configuration. No user option is added at present. Not recommended to use. |

*Table 4:* **Clock and Reset Signals** *(cont'd)*

| Port Name | Width | I/O | Clock | Description |
|---|---|---|---|---|
| phy_mcapclk | 1 | Output | mcapclk | This clock is same as phy_userclk. No user option is added at present. Not recommended to use. |
| phy_pclk | 1 | Output | pclk | Not configurable by user. Operations frequencies are:<br>• 125 MHz: Gen1 operating speed<br>• 250 MHz: Gen2 and Gen3 operating speed<br>• 500 MHz: Gen4 operating speed<br>phy_pclk is edge-aligned, but not phase-aligned to phy_coreclk and phy_userclk. |

# TX Data Signals

*Table 5:* **TX Data Signals**

| Port Name | Width | I/O | Clock Domain | Description |
|---|---|---|---|---|
| phy_txdata[63:0] | 64 | Input | pclk | Parallel data input. Bits [63:32] are used for Gen4 only and must be ignored in Gen1, Gen2, and Gen3. Bits[31:16] are used for Gen3 only and must be ignored in Gen1 and Gen2. Per-lane. |
| phy_txdatak[1:0] | 2 | Input | pclk | Indicates whether TXDATA is control or data for Gen1 and Gen2 only. Per-lane.<br>• 0b: Data<br>• 1b: Control |
| phy_txdata_valid | 1 | Input | pclk | This signal allows the MAC to instruct the PHY to ignore TXDATA for one PCLK cycle. When logic High, this indicates the PHY will use TXDATA. When logic Low, this indicates the PHY will not use TXDATA for one PCLK cycle. Gen3 and Gen4 only. Per-lane. |
| phy_txstart_block | 1 | Input | pclk | This signal allows the MAC to tell the PHY the starting byte for a 128b block. The starting byte for a 128b block must always start at bit [0] of TXDATA. Gen3 and Gen4 only. Per-lane. |
| phy_txsync_header[1:0] | 2 | Input | pclk | Provide the sync header for the PHY to use the next 130b block. The PHY reads this value when the txstart_block is asserted. Gen3 and Gen4 only. Per-lane. |
| phy_tx[p/n] | 1 | Output | Serial | The differential transmitter outputs. Per-lane. |

# RX Data Signals

*Table 6:* **RX Data Signals**

| Port Name | Width | I/O | Clock Domain | Description |
|---|---|---|---|---|
| phy_rx[p/n] | 1 | Input | Serial | The differential receiver inputs to the PHY. Per-lane. |

*Table 6:* **RX Data Signals** *(cont'd)*

| Port Name | Width | I/O | Clock Domain | Description |
|---|---|---|---|---|
| phy_rxdata[63:0] | 64 | Output | pclk | PIPE data output from receiver. Bits[63:32] are used for Gen4 only and must be ignored in Gen1, Gen2, and Gen3. Bits[31:16] are used for Gen3 only and must be ignored in Gen1 and Gen2. Per-lane. |
| phy_rxdatak[1:0] | 2 | Output | pclk | Indicates whether RXDATA is control or data. Gen1 and Gen2 only. Per-lane.<br>• 0b: Data<br>• 1b: Control |
| phy_rxdata_valid | 1 | Output | pclk | This signal allows the PHY to instruct the MAC to ignore RXDATA for one pclk cycle. When logic High, this indicates to use RXDATA. When logic Low, this indicates to ignore RXDATA for one pclk cycle. Gen3 and Gen4 only. Per-lane. |
| phy_rxstart_block[1:0] | 2 | Output | pclk | This signal allows the PHY to tell the MAC the starting byte for a 128b block.<br>• 00b: Data with no start<br>• 01b: A block starts at lower 32 bits<br>• 10b: A block starts at upper 32 bits, inactive when operating at Gen3 speed.<br>• 11b: Illegal value when operating at Gen3 speed.<br>Gen3 and Gen4 only. Per-lane. |
| phy_rxsync_header[1:0] | 2 | Output | pclk | Provide the sync header for the MAC to use the next 128b block. The MAC reads this value when the RXSTART_BLOCK is asserted. Gen3 and Gen4 only. Per-lane. |

# Command Signals Interface Ports

*Table 7:* **Command Signals**

| Port Name | Width | I/O | Clock Domain | Description |
|---|---|---|---|---|
| phy_txdetectrx | 1 | Input | pclk | Tells the PHY to perform receiver detection when this signal is logic High and POWERDOWN is in P1 low power state. Receiver detection is complete when phystatus asserts for one pclk cycle. The status of receiver detection is indicated in rxstatus when phystatus is logic High for one pclk cycle.<br>• rxstatus = 000b: Receiver not Present<br>• rxstatus = 001b: Receiver Present |
| phy_txelecidle | 1 | Input | pclk | Forces the tx[p/n] to electrical idle when this signal is logic High. During electrical idle, tx[p/n] are driven to the DC common mode voltage. Per-lane. |

*Table 7:* **Command Signals** *(cont'd)*

| Port Name | Width | I/O | Clock Domain | Description |
|---|---|---|---|---|
| phy_txcompliance | 1 | Input | pclk | Sets the running disparity to negative when this signal is logic High. Used when transmitting the PCIe compliance pattern. Per-lane. |
| phy_rxpolarity | 1 | Input | pclk | Requests the PHY to perform polarity inversion on the received data when this signal is logic High. Per-lane. |
| phy_powerdown[1:0] | 2 | Input | pclk | Request PHY to enter power saving state or return to normal power state. Power management is complete when PHYSTATUS asserts for one PCLK cycle.<br>• 00b: P0, normal operation.<br>• 01b: P0s, power saving state with low recovery time latency.<br>• 10b: P1, power saving state with longer recovery time latency.<br>• 11b: P2, lowest power state.<br>P2 not supported. |
| phy_rate[1:0] | 2 | Input | pclk | Request the PHY to perform a dynamic rate change. Rate change is complete when PHYSTATUS asserts for one PCLK cycle. rxvalid, rxdata, and rxstatus must be ignored while the PHY is in rate change.<br>• 00b: Gen1<br>• 01b: Gen2<br>• 10b: Gen3<br>• 11b: Gen4<br>In the simulation mode (PHY_SIM_EN = TRUE), PHY status assertion takes about 45 us for Gen3 speed change. |

# Status Signals Interface Ports

*Table 8:* **Status Signals**

| Name | Width | Direction | Clock Domain | Description |
|---|---|---|---|---|
| phy_rxvalid | 1 | Output | pclk | Indicates symbol lock and valid data on rxdata when logic High. This signal must be ignored during reset and rate change Gen1 and Gen2 only. Per-lane. |
| phy_phystatus | 1 | Output | pclk /Asynchronous | Used to communicate completion of several PIPE operations including reset, receiver detection, power management, and rate change. Except for reset, this signal indicates done when asserted for one pclk cycle. This signal is held High and asynchronous during reset. In error situations, such as PHY not responding with PHYSTATUS, the MAC should perform the necessary error recovery. Per-lane. |

*Table 8:* **Status Signals** *(cont'd)*

| Name | Width | Direction | Clock Domain | Description |
|---|---|---|---|---|
| phy_phystatus_rst | 1 | Output | pclk /Asynchronous | Similar to phystatus, except this port is used to communicate completion of reset only. This signal is HIGH immediately upon reset. After the PHY and GT resets are complete,this signal transitions from High to Low. |
| phy_rxelecidle | 1 | Output | Asynchronous | RXELECIDLE = High indicates RX electrical idle detected. Gen1 and Gen2 only. Per-lane. |
| phy_rxstatus[2:0] | 3 | Output | pclk | Encodes RX status and error codes for the RX data. Per-lane.<br>• 000b: Received data OK<br>• 001b: 1 SKP added<br>• 010b: 1 SKP removed<br>• 011b: Receiver detected<br>• 100b: 8b/10b (Gen1/Gen2) or 128b/130b (Gen3/Gen4) decode error<br>• 101b: Elastic buffer overflow<br>• 110b: Elastic buffer underflow<br>• 111b: Receive disparity error (Gen1/ Gen2) |
| phy_ready | 1 | Output | Asynchronous | Indicates Master Lane PHY GT is ready. |

# TX Driver Signal Interface Ports

*Table 9:* **TX Driver Signals for Gen1 and Gen2**

| Name | Width | Direction | Clock Domain | Description |
|---|---|---|---|---|
| phy_txmargin[2:0] | 3 | Input | pclk | Selects TX voltage levels. The recommendation is to set this port to 000b for the normal operating voltage range.<br>• 000b: Programmable (default)<br>• 001b: Programmable<br>• 010b: Programmable<br>• 011b: Programmable<br>• 100b: Programmable<br>• 101b: Programmable<br>• 110b: Programmable<br>• 111b: Programmable |
| phy_txswing | 1 | Input | pclk | Controls TX voltage swing level. Gen1 and Gen2 only.<br>• 0b: Full swing (default)<br>• 1b: Low swing |
| phy_txdeemph | 1 | Input | pclk | Selects TX de-emphasis. Gen1 and Gen2 only.<br>• 0b: -6.0 dB de-emphasis<br>• 1b: -3.5 dB de-emphasis (default) |

# TX Equalization Interface Ports for Third-Party MAC

The Gen3/Gen4 TX and RX equalization defined here is different from the PIPE specification. The custom Gen3/Gen4 equalization scheme described here must be used for any third-party PCIe controllers. For more details, refer to Equalization Sequences.

*Table 10:* **TX Equalization Signals for Gen3 and Gen4**

| Name | Width | I/O | Clock Domain | Description |
|---|---|---|---|---|
| phy_txeq_ctrl[1:0] | 2 | Input | pclk | TX equalization control. Must set back to 00b when txeq_done = 1b is detected. Gen3 and Gen4 speeds only. Per-lane.<br>• 00b: Idle<br>• 01b: TX preset<br>• 10b: TX coefficient<br>• 11b: TX query |
| phy_txeq_preset[3:0] | 4 | Input | pclk | Set the TX equalization to one of the defined preset when txeq_ctrl = 01b. Must use txeq_ctrl to change the preset, otherwise the default preset 0100b is used. Gen3 and Gen4 speeds only. Per-lane. |

*Table 10:* **TX Equalization Signals for Gen3 and Gen4**

| Preset | Pre-shoot (dB) | De-emphasis (dB) |
|---|---|---|
| 0000b | 0 | 6 |
| 0001b | 0 | 3.5 |
| 0010b | 0 | 4.5 |
| 0011b | 0 | 2.5 |
| 0100b | 0 | 0 |
| 0101b | 2 | 0 |
| 0110b | 2.5 | 0 |
| 0111b | 3.5 | 6 |
| 1000b | 3.5 | 3.5 |
| 1001b | 3.5 | 0 |
| 1010b | 0 | 9.5 |
| Others | Reserved | |

*Table 10:* **TX Equalization Signals for Gen3 and Gen4** *(cont'd)*

| Name | Width | I/O | Clock Domain | Description |
|---|---|---|---|---|
| phy_txeq_coeff[5:0] | 6 | Input | pclk | Set the TX equalization to a custom coefficient when txeq_control = 10b. Three consecutive pclk cycles are required to register the new 18-bit TX coefficient.<br>• The first pclk cycle is used to register pre-cursor.<br>• The second pclk cycle is used to register main-cursor.<br>• The third pclk cycle is used to register post-cursor.<br>Gen3 and Gen4 speeds only. Per-lane. |
| phy_txeq_fs[5:0] | 6 | Output | pclk | Indicates the full swing of the TX driver. Static value based on characteristics of TX driver. .Gen3 and Gen4 speeds only. |
| phy_txeq_lf[5:0] | 6 | Output | pclk | Indicates the low frequency of the TX driver. Static value based on characteristics of TX driver. Gen3 and Gen4 speeds only. |
| phy_txeq_new_coeff[17:0] | 18 | Output | pclk | Shows the status of the current TX equalization coefficient. Gen3 and Gen4 speeds only. Per-lane.<br>• [17:12]: Pre-cursor.<br>• [11:6]: Main-cursor.<br>• [5:0]: Post-cursor. |
| phy_txeq_done | 1 | Output | pclk | This port is High when TXEQ is equalization done. Single cycle done indicator for `txeq_control`. Gen3 and Gen4 speeds only. Per-lane. |
| dbg_phy_txeq_fsm | 3 | Output | pclk | Tell TX EQ FSM state:<br>FSM_IDLE = 3'd0<br>FSM_PRESET = 3'd1<br>FSM_COEFF = 3'd2<br>FSM_REMAP = 3'd3<br>FSM_QUERY = 3'd4<br>FSM_DONE = 3'd5 |

# Interface Ports RX Equalization Signals for Third-Party MAC

*Table 11:* **RX Equalization Signals for Gen3 and Gen4**

| Name | Width | I/O | Clock Domain | Description |
|---|---|---|---|---|
| phy_rxeq_ctrl[1:0] | 2 | Input | pclk | RX equalization control. Must set back to 00b when rxeq_done = 1b detected. Gen3 and Gen4 speeds only. Per-lane.<br>• 00b: Idle<br>• 01b: Reserved<br>• 10b: RX EQ<br>• 11b: RX EQ Bypass |

*Table 11:* **RX Equalization Signals for Gen3 and Gen4** *(cont'd)*

| Name | Width | I/O | Clock Domain | Description |
|---|---|---|---|---|
| phy_rxeq_preset | 3 | Input | pclk | Not used |
| phy_rxeq_txpreset[3:0] | 4 | Input | pclk | Link partner status for TX preset. Gen3 and Gen4 speeds only. Per-lane. |
| phy_rxeq_preset_sel | 1 | Output | pclk | This output port serves indications as Coefficient or preset when rxeq_done = 1b. Gen3 and Gen4 speeds only. Per-lane.<br>• 0b: Coefficient<br>• 1b: Preset |
| phy_rxeq_lffs | 6 | Input | pclk | Not used |
| phy_rxeq_lffs_sel | 1 | Output | pclk | It will be '1' when phy_rxeq_ctrl is 2'10/2'b11. |
| phy_rxeq_new_txcoeff[17:0] | 18 | Output | pclk | This is presented to the link partner to request new TX coefficient or preset. Valid only when RXEQ_DONE is High. When indicating preset, only the lower four bits are valid. Gen3 and Gen4 speeds only. Per-lane. |
| phy_rxeq_adapt_done | 1 | Output | pclk | RX equalization adaptation done. Single PCLK cycle done indicator for rxeq_control = 10b and 11b. If both rxeq_adapt_done and rxeq_done are High, then RX equalization is successfully done. If rxeq_adapt_done is Low and rxeq_done is High, then RX equalization must be requested again. Gen3 and Gen4 speeds only. Per-lane. |
| phy_rxeq_done | 1 | Output | pclk | RX equalization done. Single pclk cycle done indicator for rxeq_control. Must set pipe_rxeq_control back to 00b when pipe_rxeq_done = High is detected. RX equalization must be re-initiated if rxeq_adapt_done is not High. Gen3 and Gen4 speeds only. Per-lane. |
| dbg_phy_rxeq_fsm | 3 | Output | pclk | Tell RX EQ FSM state:<br>FSM_IDLE = 3'd0<br>FSM_PRESET = 3'd1<br>FSM_TXCOEFF = 3'd2<br>FSM_ADAPT = 3'd3<br>FSM_DONE = 3'd4 |

# PIPE RX Margin (To GT Quad)

*Table 12:* **PIPE RX Margin (To GT Quad)**

| Name | Width | Direction | Clock Domain | Description |
|---|---|---|---|---|
| pipe_rx_margin_req_req | 1 | output | pclk | RX Margin Response Request. When 1b, data presented on `pipe_rx_margin_req_{*}` signals to a GT Quad is valid. MAC drives `pipe_rx_marginn_req_req` to 1b then waits for `pipe_rx_marginn_req_ack` to be driven 1b, before driving it to 0b. |
| pipe_rx_margin_req_ack | 1 | input | pclk | RX Margin Request Ack. 1b indicates GT Quad has accepted data presented on `pipe_rx_margin_req_{*}` signals to a GT Quad, and 1b in response to `pipe_rx_margin_req_req` being 1b. `pipe_rx_margin_req_ack` is driven to 0b, when `pipe_rx_margin_req_req` transitions to 0b. |
| pipe_rx_margin_req_lane_num | 2 | output | pclk | RX Margin Request Lane Number. Physical Lane Number in a GT Quad for which data has been received in the range 0H-3H. |
| pipe_rx_margin_req_cmd | 4 | output | pclk | RX Margin Request. This is the Margin Command received in CSKPOS on Upstream Port or Margin Command from Lane Margin Control Register on Downstream Port. |
| pipe_rx_margin_req_payload | 8 | output | pclk | RX Margin Request Payload. This is the RX Margin Payload received in CSKPOS on Upstream Port or Step Margin Value. |

# PIPE RX Margin (To MAC)

*Table 13:* **PIPE RX Margin (To MAC)**

| Name | Width | Direction | Clock Domain | Description |
|---|---|---|---|---|
| pipe_rx_margin_res_req | 1 | output | pclk | RX Margin Response Request. When 1b, data presented on `pipe_rx_margin_res_{*}` signals to MAC is valid. GT Quad drives `pipe_rx_margin_req_req` to 1b then waits for `pipe_rx_margin_res_ack` to be driven 1b, before driving it to 0b. |

*Table 13:* **PIPE RX Margin (To MAC)** *(cont'd)*

| Name | Width | Direction | Clock Domain | Description |
|---|---|---|---|---|
| pipe_rx_margin_res_ack | 1 | input | pclk | RX Margin Response Ack. 1b indicates MAC has accepted data presented on `pipe_rx_margin_res_{*}` signals to a GT Quad, and 1b in response to `pipe_rx_margin_res_req` being 1b. `pipe_rx_margin_res_ack` is driven to 0b, when `pipe_rx_margin_res_req` transitions to 0b. |
| pipe_rx_margin_res_lane_num | 2 | output | pclk | RX Margin Response Lane Number. Physical Lane Number in a GT Quad for which data has been received in the range 0H-3H. |
| pipe_rx_margin_res_cmd | 4 | output | pclk | RX Margin Response Command. This is the command in response to request generated by GT Quad. |
| pipe_rx_margin_res_payload | 8 | output | pclk | RX Margin Response Payload. This is payload returned by GT Quad in response. |

# TX Equalization Signals for Gen3 and Gen4 (for Xilinx MAC)

*Table 14:* **TX Equalization Signals for Gen3 and Gen4 (for Xilinx MAC)**

| Name | Width | Direction | Clock Domain | Description |
|---|---|---|---|---|
| phy_txprecursor | 5 | input | pclk | Transmitter pre-cursor TX pre-emphasis control. Link Equalization settings. Per-Lane. |
| phy_txmaincursor | 7 | input | pclk | Transmitter post-cursor TX pre-emphasis control. Link Equalization settings. Per-Lane. |
| phy_txpostcursor | 5 | input | pclk | Transmitter main-cursor coefficients. Link Equalization settings. Per-Lane. |

# Miscellaneous Ports for Xilinx PCIe MAC

*Table 15:* **Misc Ports for Xilinx PCIe MAC**

| Name | Width | Direction | Clock Domain | Description |
|---|---|---|---|---|
| pcie_ltssm_state | 6 | input | pclk | PCIe Link Training and Status State Machine. Per-Link. |

*Table 15:* **Misc Ports for Xilinx PCIe MAC** *(cont'd)*

| Name | Width | Direction | Clock Domain | Description |
|---|---|---|---|---|
| pcie_link_reach_target | 1 | input | pclk | Indicates the Reach Length Target of current link. Per-Link.<br>• 1'b0: Short Reach<br>• 1'b1: Long Reach |

**Notes:**

1. Refer to *Versal ACAP Transceivers Wizard LogiCORE IP Product Guide* (PG331) for APB related signals.

# Designing with the Core

This section includes guidelines and additional information to facilitate designing with the core.

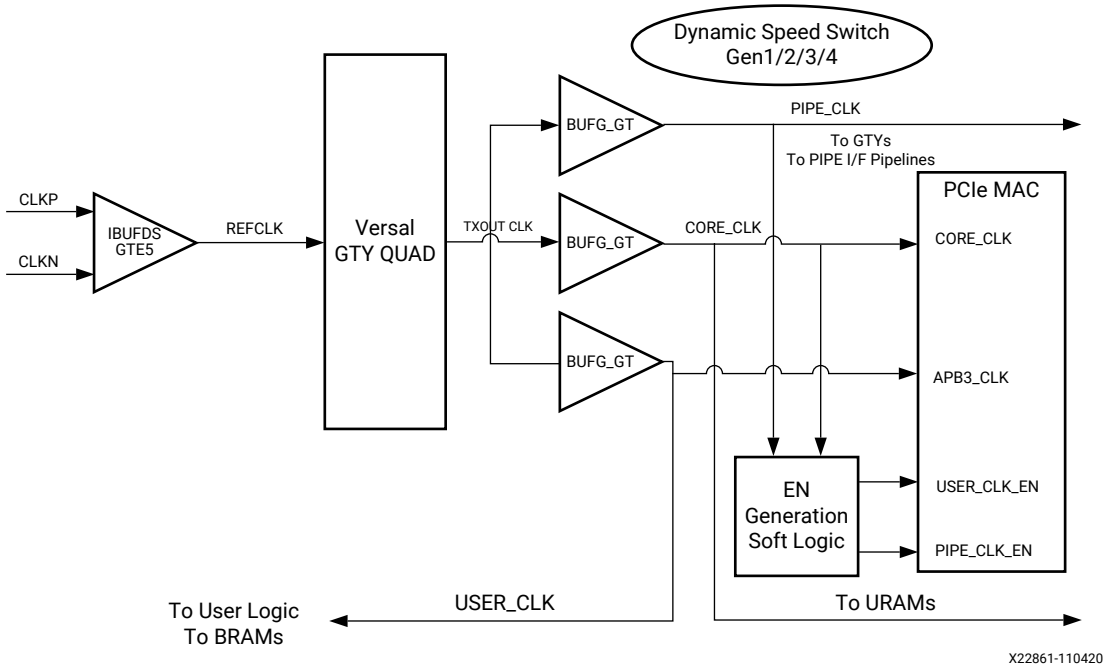
***Note*:** This section describes the guidelines for the Versal™ ACAP PHY for PCIe® core. Versal Premium ACAP guidelines are scheduled for a future release.

## Clocking

- The Versal™ ACAP PHY for PCI Express® IP GTY can be configured to support PCIe® applications with 100 MHz, 125 MHz, or 250 MHz reference clock.
- The reference clock can be synchronous or asynchronous.
- The `phy_pclk` is the primary clock for the PIPE interface, FPGA fabric, and GTY [TX/ RX] `usrclk` and [TX/RX] `usrclk2`.
- In addition to `phy_pclk`, there are other clocks (`phy_coreclk`, `phy_userclk2`, `phy_userclk`) available to support the PCIe MAC.
- `BUFG_GTs` are used to generate these clocks, so MMCM will not be required.
- To use the reference clock for FPGA fabric, another `BUFG_GT` must be used.
- The source of the GTY reference clock must come directly from IBUFDS_GTE5.

The following figure shows clocking architecture for the Versal ACAP PCIe PHY IP configured for x2 lane width.

*Figure 2:* **Clock Architecture**




- **PIPE_CLK (`phy_pclk`):** Provided to Versal device GTY to clock the PIPE interface. PIPE Clock must be 125 MHz for Gen1 operation or 250 MHz for Gen2/Gen3 and 500 MHz for Gen4 operation. Note that the PIPE interface data width is 16 bits for Gen1 or Gen2 operation, 32 bits for Gen3, and Gen4 operation. PIPE Clock frequency input to the block is switched dynamically based on the current selected speed of operation is accomplished using a `BUFG_GT` resource from the FPGA global clocking infrastructure.
- **CORE_CLK (`phy_coreclk`) :** CORE_CLK is the dominant clock domain in the PCIe block, `core_clk` is also used to drive UltraRAMs interfaced with the Hard Block.
- **USER_CLK (`phy_userclk`):** Clocks the non- AXI4 ST user interfaces. The frequency can be 62.5, 125, 250 or 500 MHz, depending on the data rate, number of lanes and Transaction Interface width.
- **USER_CLK2 (`phy_userclk2`):** Clocks the AXI4 ST user interfaces. External to the block `user_clk2` is the same as (and driven by the same `BUFG_GT`) either `core_clk` or `user_clk` depending on the configuration. Internal to the block `user_clk2` is created using `core_clk` and `user_clk_en` and similarly matches either `core_clk` or `user_clk`.

# Resets

Reset sequence procedure is moved inside the GT QUAD for the Versal devices.

# MAC Requirements

For the MAC to function properly with the PCIe PHY IP, it is necessary that the MAC satisfies these requirements:

- RX lane-to-lane `deskew` must be handled by the PCIe MAC.
- PCIe MAC should not depend on `rxelecidle` low-to-high transition.
- RX electrical idle entry must be inferred by the PCIe MAC.
- TX SKP OS must be 16 symbols for Gen3.

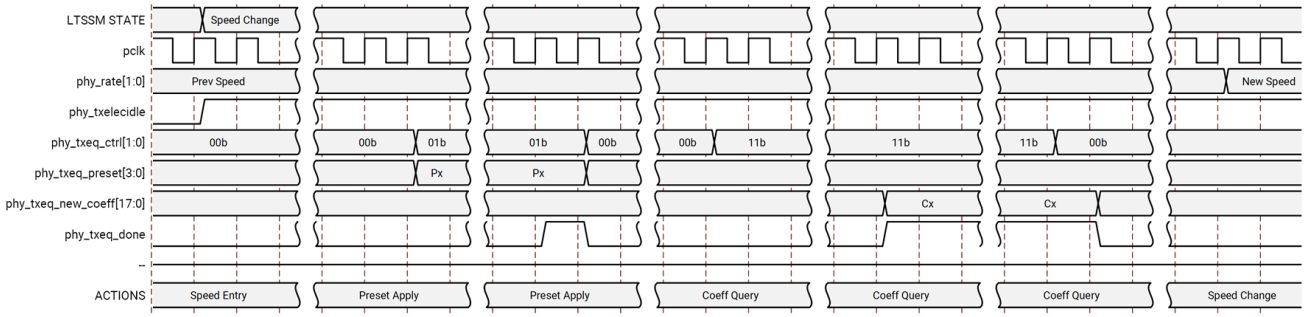
# Equalization Sequences

This section describes the equalization sequences of the PCIe PHY IP.

### Preset Apply During Speed Change

The following figure shows the TX Equalization interface signals during the speed change. It includes Preset Apply (`phy_txeq_ctrl = 2'b01`) and Coefficient Query (`phy_txeq_ctrl = 2'b11`). The speed change is performed in LTSSM Recovery.Speed and LTSSM Polling.Compliance states. Preset Apply step must be performed after asserting `phy_txelecidle` and before driving `phy_rate` to the new speed.

*Figure 3:* **TX Equalization Interface Signals during Speed Change (Preset Apply)**

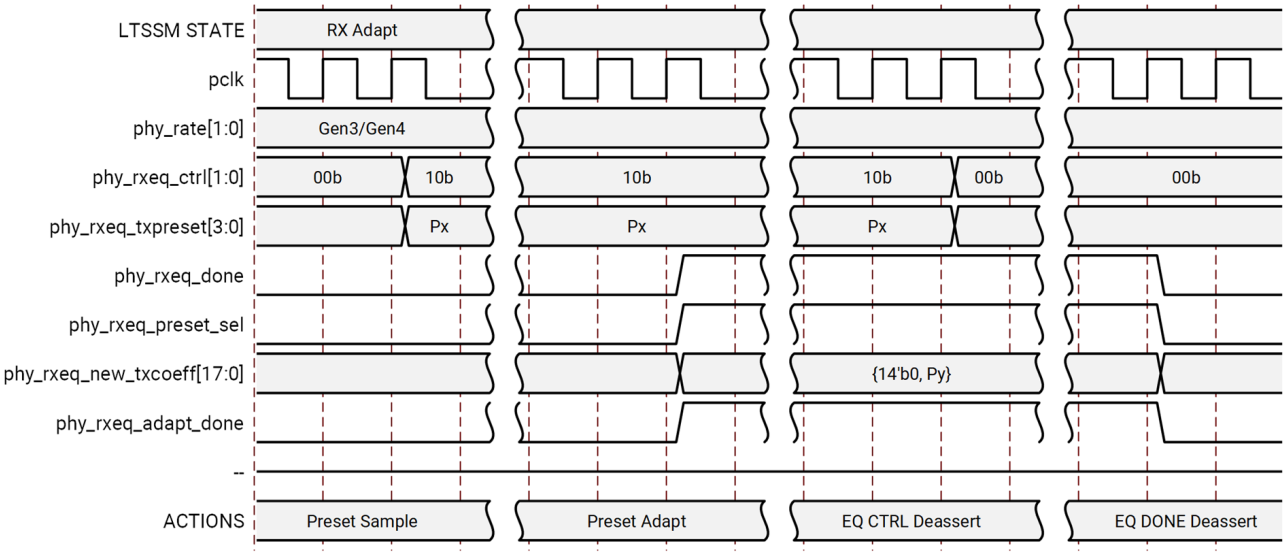



### RX Adapt

The following figures describe the RX Equalization interface signals during the RX Adapt. RX Adapt is performed in Phase 2 of the LTSSM Recovery. Equalization state for the Upstream Port and Phase 3 of the LTSSM Recovery. Equalization state for the Downstream Port. It is composed of two steps: a) New Proposal, and b) Adaptation, in that order.

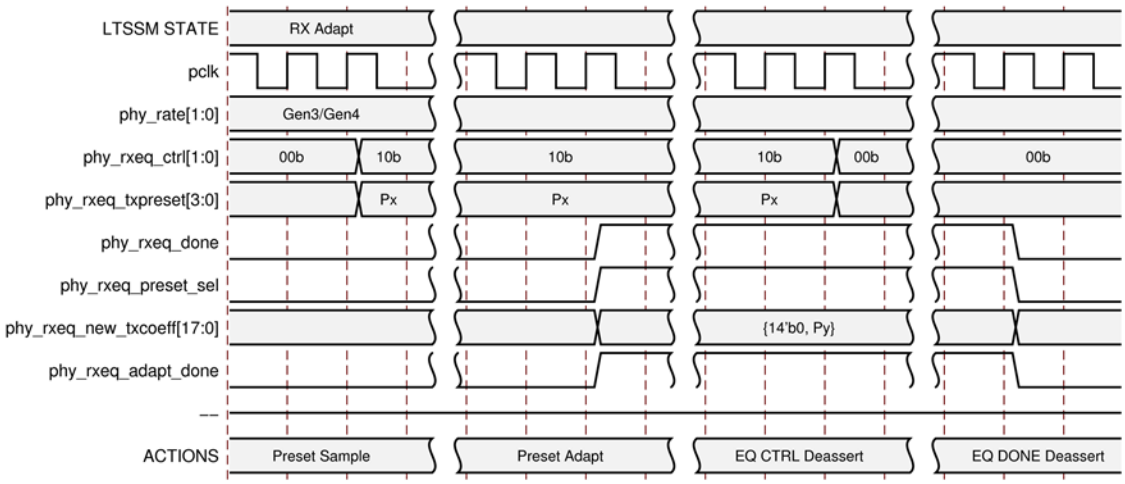
The following figure shows the New Proposal step where RX purposes a new preset (`phy_rxeq_done` = `1'b1` and `phy_rxeq_adapt_done` = `1'b0`) upon a TX preset request (`phy_rxeq_txpreset` with `phy_rxeq_ctrl`= `2'b10`).

*Figure 4:* **RX Equalization Interface Signals during RX Adapt (New Proposal)**




The following figure shows the Adaptation step where RX adapts the preset (`phy_rxeq_done` = `1'b1` and `phy_rxeq_adapt_done` = `1'b1`) which TX requests (`phy_rxeq_txpreset` with `phy_rxeq_ctrl`= `2'b10`).

*Figure 5:* **RX Equalization Interface Signals during RX Adapt (Adapted)**

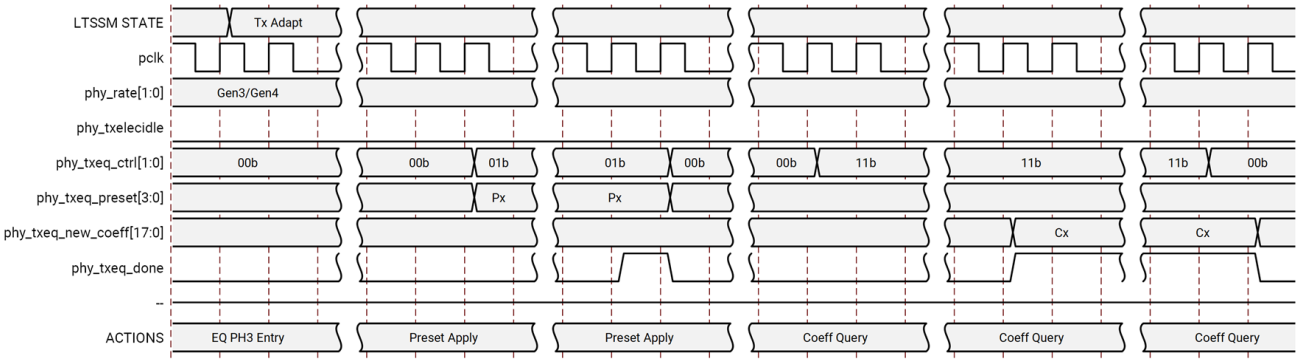


### TX Adapt

The following figures describe the TX Adapt steps. TX Adapt is performed in Phase 3 of the LTSSM Recovery. Equalization state for the Upstream Port and Phase 2 of the LTSSM Recovery. Equalization state for the Downstream Port.

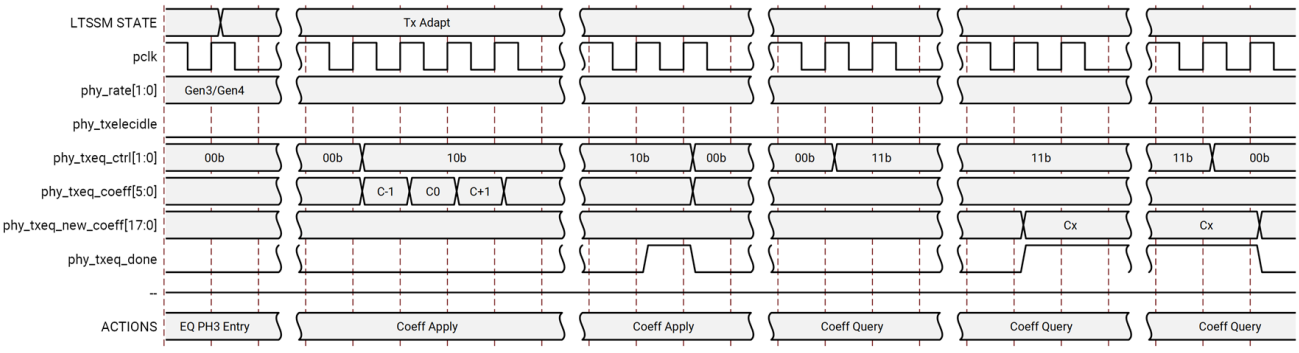
The following figure shows the TX Adapt step when receiving Preset. It includes Preset Apply (`phy_txeq_ctrl = 2'b01`) and Coefficient Query (`phy_txeq_ctrl = 2'b11`).

*Figure 6:* **TX Equalization Interface Signals during TX Adapt (Preset)**




The following figure shows the TX Adapt step when receiving Coefficients. It includes Coefficient Apply (`phy_txeq_ctrl =2'b10`) and Coefficient Query (`phy_txeq_ctrl = 2'b11`).

*Figure 7:* **TX Equalization Interface Signals during TX Adapt (Coefficients)**



# Design Flow Steps

This section describes customizing and generating the core, constraining the core, and the simulation, synthesis, and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
- *Vivado Design Suite User Guide: Designing with IP* (UG896)
- *Vivado Design Suite User Guide: Getting Started* (UG910)
- *Vivado Design Suite User Guide: Logic Simulation* (UG900)

## Customizing and Generating the Core

This section includes information about using Xilinx® tools to customize and generate the core in the Vivado® Design Suite.

***Note:*** This section describes guidelines and options for the Versal™ ACAP PHY for PCI Express® core. Versal Premium ACAP guidelines are scheduled for a future release

### Targeted Supported Devices

The main purpose of this IP is to enable you to generate the PHY wrappers, which can be used with any Versal™ devices.

### Customizing the Core

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

- Versal platform PCIe PHY IP has to be used in the Vivado IP integrator canvas.
- Double-click the PHY IP and configure to the desired configuration.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) and the *Vivado Design Suite User Guide: Getting Started* (UG910).

***Note*:** Figures in this chapter are illustrations of the Vivado Integrated Design Environment (IDE). The layout depicted here might vary from the current version.

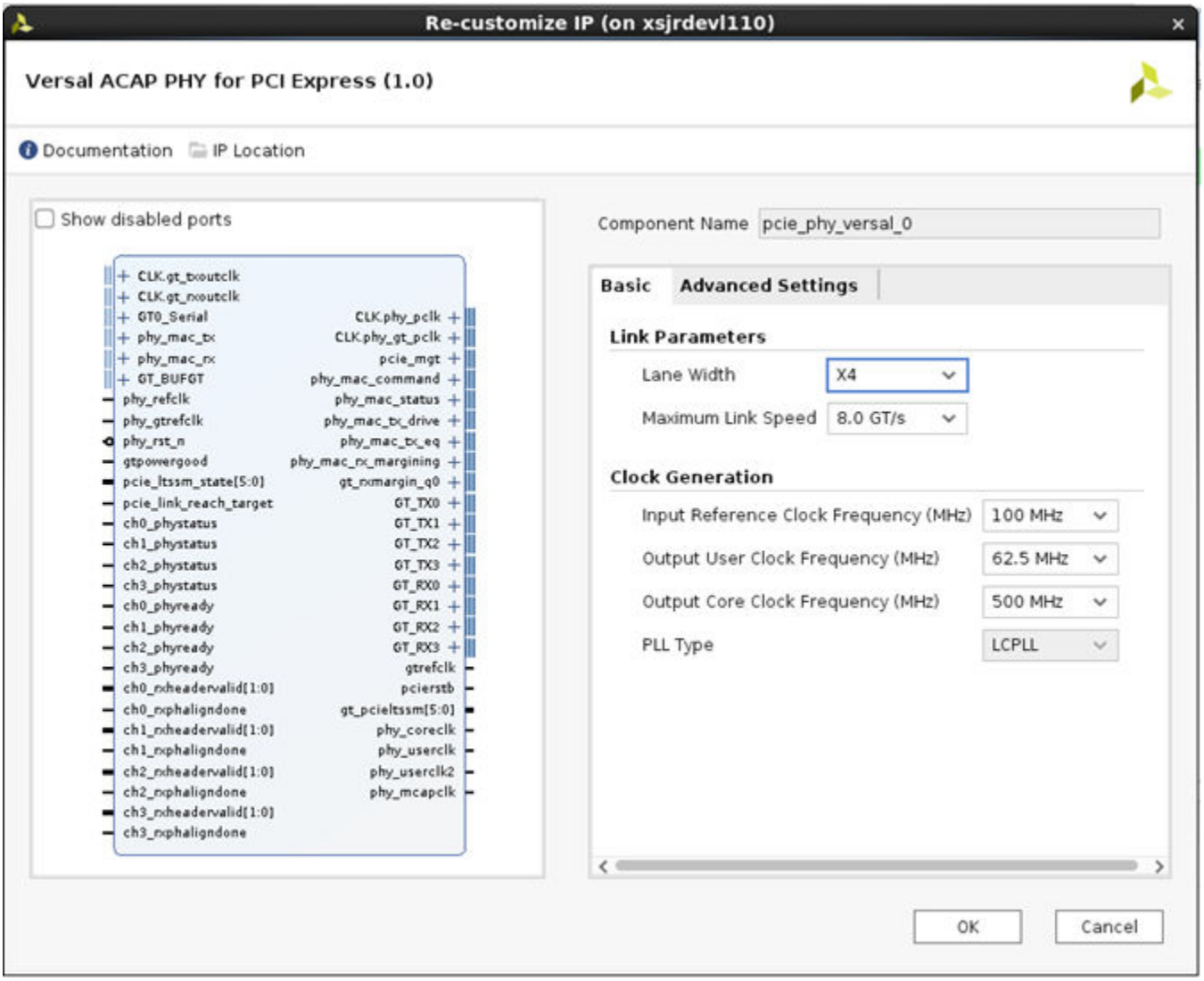
The Customize IP dialog box for the PCIe® PHY IP consists of the following tabs:

1. Basic Tab
2. Advanced Settings Tab

## *Basic Tab*

The initial customization screen is used to define the basic parameters for the core, including the component name, reference clock frequency, lane width, and speed.

*Figure 8:* **Basic Tab**



- **Component Name:** It is the base name of the output files generated for the core. The name must begin with a letter and can be composed of these characters: a to z, 0 to 9, and "_."
- **Link Width:** The core requires the selection of the initial lane width. Supported lane widths are x1, x2, x4, x8, and x16.
- **Maximum Link Speed:** The core allows you to select the Maximum Link Speed supported by the device. Supported link speeds are:
  - 2.5 Gb/s, 5.0 Gb/s, 8.0 Gb/s, and 16.0 Gb/s. The 16.0 Gb/s is available only for lane widths of x1, x2, x4, and x8.
- **Input Reference Clock Frequency:** Selects the input frequency of the reference clock provided on `sys_clk`. It is the GT REFCLK frequency for the IP. Supported values are 100 MHz, 125 MHz, and 250 MHz. For important information about clocking, see Clocking.
- **Output User Clock Frequency:** Selects the frequency of the output USERCLK that can be used by the PCIe MAC.

*Table 16:* **User Clock Options with Xilinx PCIe MAC**

| Speed | Lane | User Clock (in MHz) |
|---|---|---|
| Gen1 | x1 | 62.5, 125, 250 |
| | x2 | 62.5, 125, 250 |
| | x4 | 62.5, 125, 250 |
| | x8 | 62.5, 125, 250 |
| | x16 | 62.5, 125, 250 |
| Gen2 | x1 | 62.5, 125, 250 |
| | x2 | 62.5, 125, 250 |
| | x4 | 62.5, 125, 250 |
| | x8 | 62.5, 125, 250 |
| | x16 | 62.5, 125, 250 |
| Gen3 | x1 | 62.5, 125, 250 |
| | x2 | 62.5, 125, 250 |
| | x4 | 62.5, 125, 250 |
| | x8 | 62.5, 125, 250 |
| | x16 | 62.5, 125, 250, 500 |
| Gen4 | x1 | 125, 250 |
| | x2 | 125, 250 |
| | x4 | 125, 250 |
| | x8 | 125, 250, 500 |

For important information about clocking the core, see Clocking.

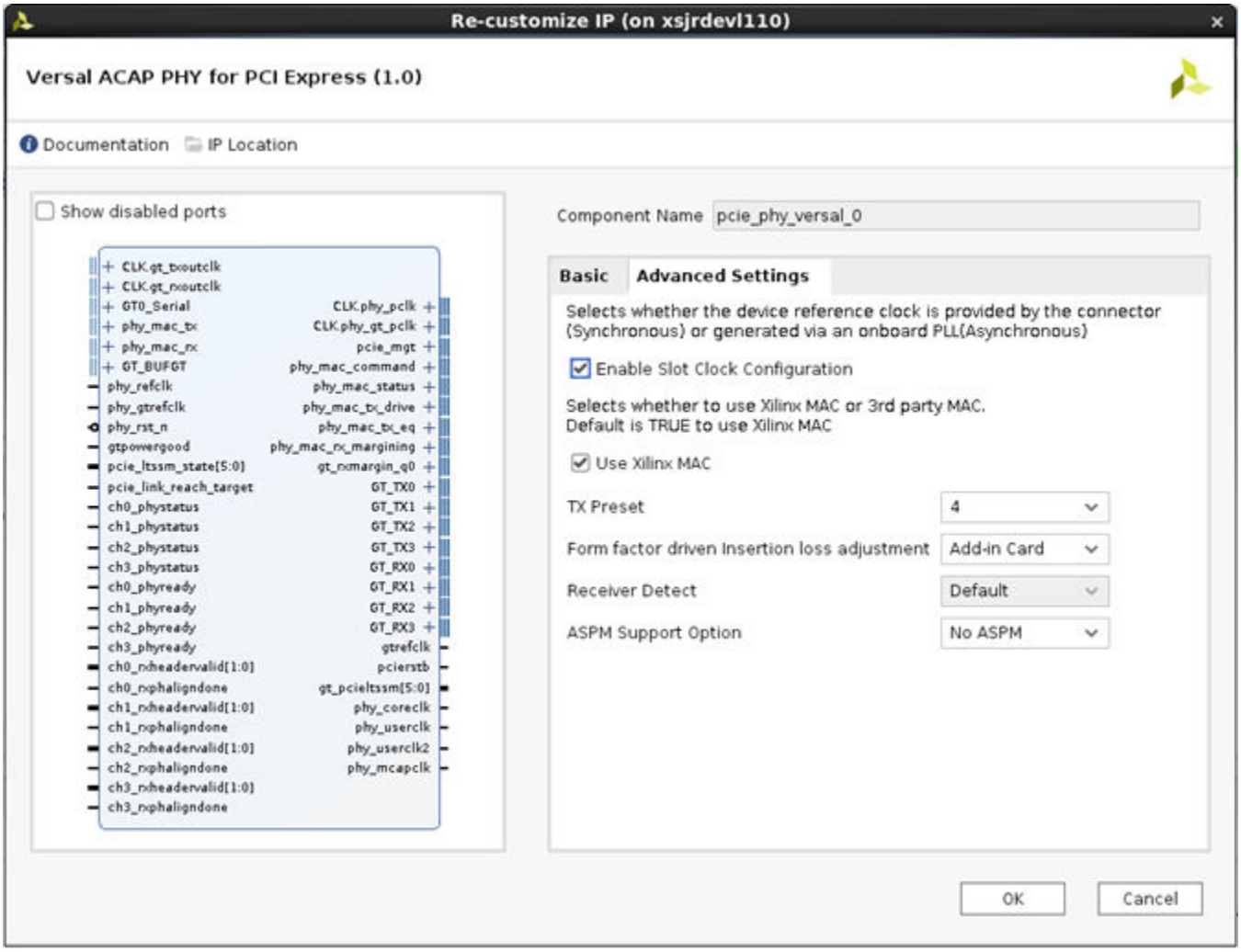
- **Output Core Clock Frequency:** Selects the frequency of the output `coreclk` that can be used by the PCIe MAC. 250 MHz is supported for all configurations. There is support for 500 MHz for x16 Gen3 (8.0 Gb/s) and for x8 Gen4 (16.0 Gb/s) configurations. For important information about clocking the core, see Clocking.
- **PLL Type:** Selects the PLL type for GTs used. For all speed PCIe PHY IP uses LCPLL as PLL TYPE. But TXPROGDIV CLOCK SOURCE uses RPLL internally.

*Table 17:* **PLL Type**

| Link Speed | PLL Type | Description |
|---|---|---|
| 2.5 GT/s | LCPLL | The default is LCPLL. |
| 5.0 GT/s | LCPLL | The default is LCPLL. |
| 8.0 GT/s | LCPLL | The default is LCPLL. |
| 16.0 GT/s | LCPLL | The default is LCPLL. |

### *Advanced Settings Tab*

*Figure 9:* **Advanced Settings Tab**



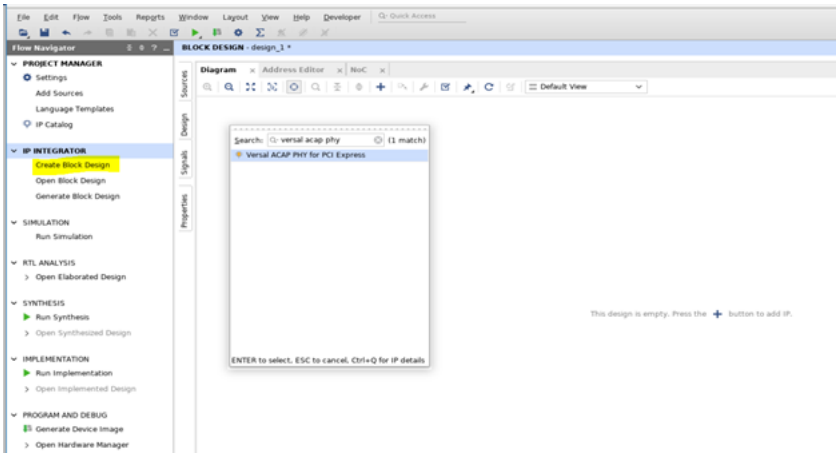

- **Enable Slot Clock Configuration:** When this option is selected, the link is synchronously clocked, that is, whether the device reference clock would be provided synchronously by the connector or asynchronously through an onboard PLL.
- **TX Preset:** It is not advisable to change the default value of 4. Preset value of 5 might work better on some systems.
- **Use Xilinx MAC:** Select this option if the connected PCIe MAC to the Versal PHY IP is a Xilinx MAC.Selecting this option removes soft TX and RX equalization modules which are not required for Xilinx PCIe MACs. De-selecting the option will enable TX and RQ equalization modules required for third party PCIe MACs. For third-party PCIe MACs, de-select this option to enable the TX and RX equalization modules needed.

- **Form factor driven Insertion loss adjustment:** Indicates the transmitter to receiver insertion loss at the Nyquist frequency depending on the form factor selection. There are three available options: Chip to Chip, Add-in Card, and Backplane in the menu, corresponds to 5 dB, 15 dB, and 20 dB insertion loss, respectively. Also, this parameter internally sets LPM mode for the Chip to Chip option, and DFE mode for all others in the GTs.
- **Receiver Detect:** Indicates the type of Receiver Detect Default or Falling Edge. For more information about this option, see the *Versal ACAP GTY and GTYP Transceivers Architecture Manual* (AM002).
- **ASPM Support Option:** The available options are No_ASPM, L0s Supported and L1 Supported. Select the option that is the same as that supported in the MAC. ASPM L0s is available only for Gen1 and Gen2 configurations.

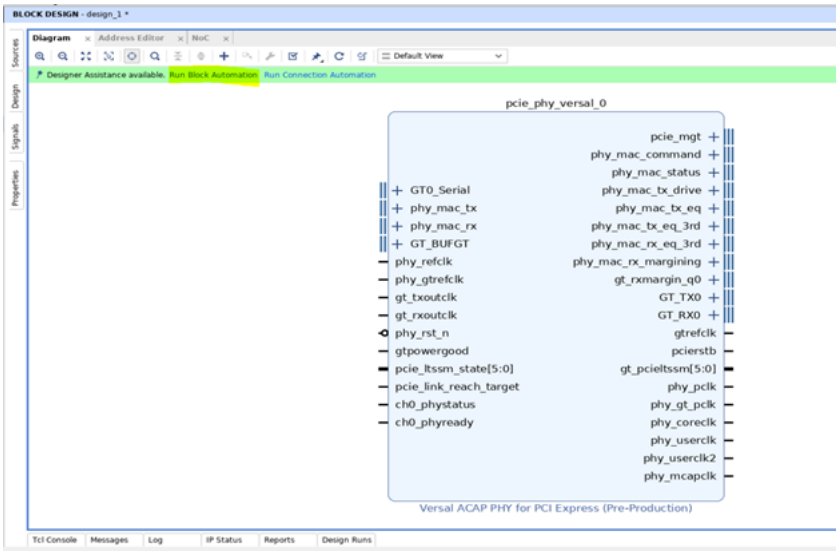
# Vivado IP Integrator Block Automation

The Versal™ ACAP PHY for PCI Express IP supports IP integrator block automation to easily connect the GT_QUADs on IPI Canvas. The following steps shows using the IP integrator block automation.

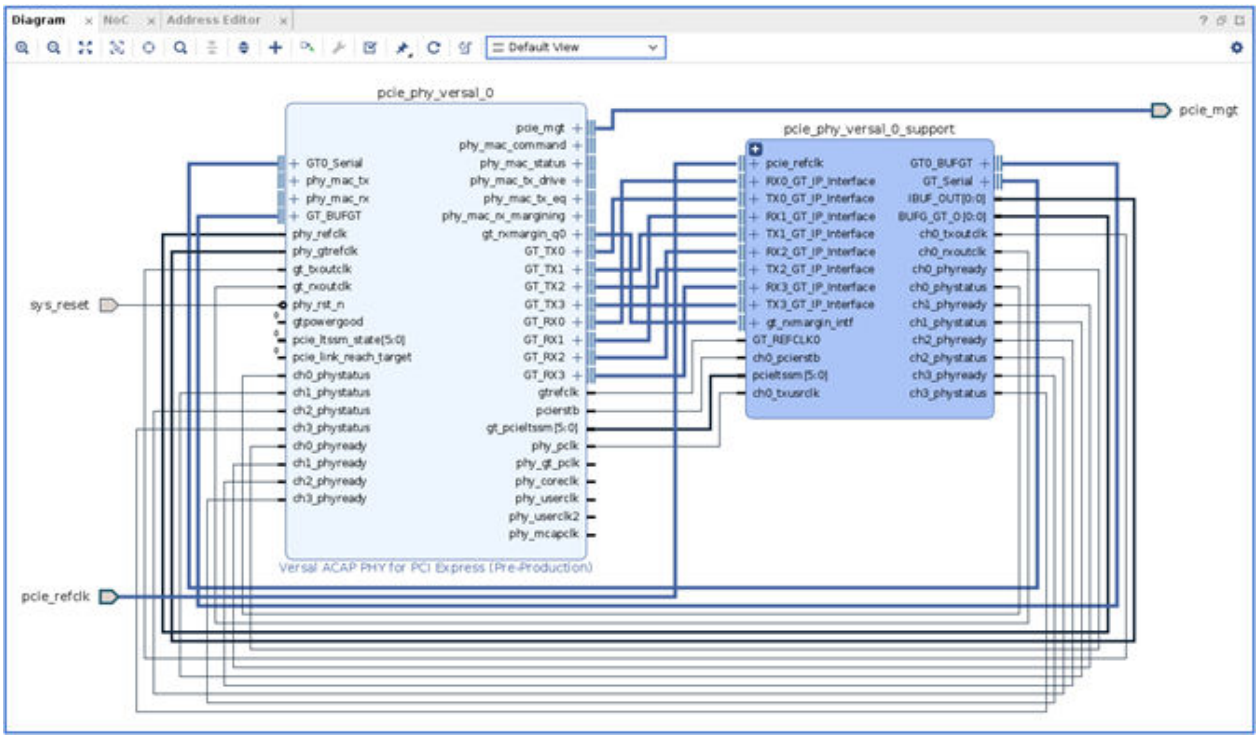
1. Create an IP Integrator block design from Vivado project as shown in the following figure.




2. Add the Versal ACAP PHY for PCI Express IP to the canvas as shown in following figure.




3. Configure the Versal ACAP PHY for PCI Express IP by opening the IP GUI (double click on IP symbol) using steps mentioned in Customizing the Core section.
4. Click on **Run Block Automation** from Designer Assistance BAR as shown in above figure.
5. The Versal ACAP PHY for PCI Express IP will be connected to GT Quads (inside pcie_phy_versal_0_support) and the required ports will be made external as shown in the following figure.



## Output Generation

For details on the output generation of the design, see the *Vivado Design Suite User Guide: Designing with IP* (UG896).

# Constraining the Core

This section contains information about constraining the core in the Vivado® Design Suite.

### Required Constraints

The PCIe PHY IP solution requires the specification of timing and other physical implementation constraints to meet specified performance requirements. These constraints are provided in a Xilinx Design Constraints (XDC) file. Pinouts and hierarchy names in the generated XDC correspond to the provided example design.

**IMPORTANT!** *If the example design top file is not used, copy the IBUFDS_GTE5 instance for the reference clock, IBUF Instance for* `sys_rst`*, and also the timing constraints associated with them into your local design top.*

You should provide the location constraint for `sys_rst` and `sys_rst_override` pins based on the development board in use.

In addition, you should also provide GT location constraints. For more information, see Appendix A: GT Selection and Pin Planning.

Constraints provided with the integrated block solution have been tested in hardware and provide consistent results. Constraints can be modified, but modifications should only be made with a thorough understanding of the effect of each constraint. Additionally, support is not provided for designs that deviate from the provided constraints.

### Device, Package, and Speed Grade Selections

The device selection portion of the XDC informs the implementation tools which part, package, and speed grade to target for the design.

The device selection section always contains a part selection line, but can also contain part or package-specific options. An example part selection line follows:

```
CONFIG PART = XCVC1902-VSVA2197
```

### Clock Frequencies

This section is not applicable for this IP core.

### Clock Management

This section is not applicable for this IP core.

### Clock Placement

This section is not applicable for this IP core.

### Banking

This section is not applicable for this IP core.

### Transceiver Placement

This section is not applicable for this IP core.

### I/O Standard and Placement

This section is not applicable for this IP core.

## Simulation

For comprehensive information about Vivado® simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900).

## Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896).

# Example Design

This chapter contains information about the example design provided in the Vivado® Design Suite.

The example design is a design assistant to validate the functionality of the PCIe® PHY IP core. It ensures the connectivity, clock connections, reset sequence, and enablement of rate change based on your selection.

## Overview

The example design operation is explained below:

- The example design expects the system reset to be received from the link partner.
- You have an additional option to override the system reset using `sys_rst_override` provided in the example design. This can be connected to any I/O on the board, such as a switch pin.
- The TX and RX electrical idle is High at this point.
- The design waits for the reset sequence to finish. For more information on the reset sequence, see Resets.
- The transceiver provides the `phystatus_rst` which indicates that the PHY is ready. Make sure the PCIe MAC is connected to this output from the PHY.
- The design now waits for `phystatus` on all lanes.
- Based on your selection of `phyrate` (either Gen1, Gen2, Gen3, or Gen4), the design changes to the desired speed.

## Simulating the Example Design

The example design provides a quick method to simulate and observe the behavior of the core generated using the Vivado® Design Suite.

***Note:*** To view a complete example of the PCIe PHY IP solution with a PCIe MAC, see *Versal ACAP Integrated Block for PCI Express LogiCORE IP Product Guide* (PG343), Example Design section. It offers a supplemental PCIe PHY IP example design that showcases Versal ACAP Integrated block for a PCIe example design integrating with the Versal ACAP PHY IP.

The currently supported simulators are:

- Vivado simulator (default)
- Mentor Graphics Questa Advanced Simulator

The simulator uses the example design test bench and test cases provided along with the example design for the design configuration.

For any project (Versal™ ACAP PHY for PCIe® IP core) generated out of the box, the simulations using the default Vivado simulator can be run as follows:

1. In the Sources Window, right-click the example project file (`.xci`), and select **Open IP Example Design**.

   The example project is created.

2. In the Flow Navigator (left-hand pane), under Simulation, right-click **Run Simulation** and select **Run Behavioral Simulation**.

   **IMPORTANT!** *The post-synthesis and post-implementation simulation options are not supported for the Versal™ ACAP PHY for PCIe®.*

   After the Run Behavioral Simulation Option is running, you can observe the compilation and elaboration phase through the activity in the Tcl Console, and in the Simulation tab of the Log Window.

3. In Tcl Console, type the run all command and press Enter. This runs the complete simulation as per the test case provided in example design test bench.

   After the simulation is complete, the result can be viewed in the Tcl Console.

In Vivado IDE, change the simulation settings as follows:

1. In the Flow Navigator, under Simulation, select **Simulation Settings**.
2. Set the Target simulator to QuestaSim/ModelSim Simulator, Incisive Enterprise Simulator (IES), or Verilog Compiler Simulator.
3. In the simulator tab, select **Run Simulation > Run behavioral simulation**.
4. When prompted, click **Yes** to change and then run the simulator.

The simulation environment provided with the PCIe PHY IP core performs rate change operation in the same sequence as described in the Overview. The simulation environment has the target speed set to Gen3.

Appendix A

# GT Selection and Pin Planning

This appendix provides guidance on gigabit transceiver (GT) selection for Versal™ devices and some key recommendations that should be considered when selecting the GT locations. The GT locations for Versal devices can be customized through the IP customization wizard. This appendix provides guidance for CPM, PL PCIe® and PHY IP based solutions. In this guide, the IP related guidance is of primary importance, while the other related guidance might be relevant and is provided for informational purposes.

A GT Quad is comprised of four GT lanes. GT Quad and ref clock locations for CPM4 are in fixed locations depending on the desired link configuration (see GT Quad Locations). When selecting GT Quads for the PL PCIe-based solution Xilinx® recommends that you use the GT Quad most adjacent to the integrated block. While this is not required, it will improve place, route, and timing for the design.

- Link widths of x1, x2, and x4 require one bonded GT Quad and should not split lanes between two GT Quads.
- A link width of x8 requires two adjacent GT Quads that are bonded and are in the same SLR.
- A link width of x16 requires four adjacent GT Quads that are bonded and are in the same SLR.
- PL PCIe blocks should use GTs adjacent to the PCIe block where possible.
- CPM has a fixed connectivity to GTs based on the CPM configuration.

For GTs on the **left side of the device**, PCIe lane 0 is placed in the bottom-most GT of the bottom-most GT Quad. Subsequent lanes use the next available GTs moving vertically up the device as the lane number increments. This means that the highest PCIe lane number uses the top-most GT in the top-most GT Quad that is used for PCIe.

For GTs on the **right side of the device**, PCIe lane 0 is placed in the top-most GT of the top-most GT Quad. Subsequent lanes use the next available GTs moving vertically down the device as the lane number increments. This means that the highest PCIe lane number uses the bottom-most GT in the bottom-most GT Quad that is used for PCIe.

The PCIe reference clock uses GTREFCLK0 in the PCIe lane 0 GT Quad for x1, x2, x4, and x8 configurations. For x16 configurations the PCIe reference clock should use GTREFCLK0 on a GT Quad associated with lanes 8-11. This allows the clock to be forwarded to all 16 PCIe lanes.

The PCIe reset pins for CPM designs must connect to one of specified pins for each of the two PCIe controllers. The PCIe reset pin for PL PCIe and PHY IP designs can be connected to any compatible PL pin location, or the CPM PCIe reset pins when the corresponding CPM PCIe controller is not in use. This is summarized in the table below.

*Table 18:* **PCIe Controller Reset Pin Locations**

| Versal PCIe Controller | Versal Reset Pin Location |
|---|---|
| CPM PCIe Controller 0 | PS MIO 18 |
| | PMC MIO 24 |
| | PMC MIO 38 |
| CPM PCIe Controller 1 | PS MIO 19 |
| | PMC MIO 25 |
| | PMC MIO 39 |
| PL PCIe Controllers | Any compatible single-ended PL I/O pin. |
| Versal ACAP PHY IP | Any compatible single-ended PL I/O pin. |

# GT Locations

## Assigning GT Locations

Unlike in UltraScale+™ and previous devices implementations where direct assignment of GTs are not possible in the user constraints, in Versal ACAP implementations the GTs are external to the PCIe® IP and hence the GT assignment can be done in the user constraints. The GT locations are assigned in Quad granularity. (Details of lane 0 placement and lane ordering is found in the GT Selection section. Changing the lane 0 location or lane ordering from the default setting is not supported. Below is an example of assigning GT locations in a user constraint file.

***Note*:** The gt_quad instances should be assigned contiguously.

```
set_property LOC GTY_QUAD_X0Y6   [get_cells $gt_quads -filter NAME=~*/
gt_quad_3/*]
set_property LOC GTY_QUAD_X0Y5   [get_cells $gt_quads -filter NAME=~*/
gt_quad_2/*]
set_property LOC GTY_QUAD_X0Y4   [get_cells $gt_quads -filter NAME=~*/
gt_quad_1/*]
set_property LOC GTY_QUAD_X0Y3   [get_cells $gt_quads -filter NAME=~*/
gt_quad_0/*]
```

# GT Quad Locations

The following table identifies the PCIe lane0 GT Quad(s) that can be used for each PCIe controller location. The Quad shown in bold is the most adjacent or suggested GT Quad for each PCIe lane0 location.

*Table 19:* **GT Locations**

| Device | Package | PCIe Blocks | GT QUAD (X16) | GT QUAD (X8) | GT QUAD (X4 and Below) |
|---|---|---|---|---|---|
| XCVC1902 | VIVA1596 | CPM Controller 0 | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y3** |
| | | CPM Controller 1 | N/A | **GTY_QUAD_X0Y5** | **GTY_QUAD_X0Y5** |
| | | X0Y2 | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y5**<br>GTY_QUAD_X0Y4<br>GTY_QUAD_X0Y3 | GTY_QUAD_X0Y6<br>**GTY_QUAD_X0Y5**<br>GTY_QUAD_X0Y4<br>GTY_QUAD_X0Y3 |
| | | X0Y1 | **GTY_QUAD_X0Y3** | GTY_QUAD_X0Y5<br>GTY_QUAD_X0Y4<br>**GTY_QUAD_X0Y3** | GTY_QUAD_X0Y6<br>GTY_QUAD_X0Y5<br>**GTY_QUAD_X0Y4**<br>GTY_QUAD_X0Y3 |
| | | X1Y2 | **GTY_QUAD_X1Y5** | **GTY_QUAD_X1Y5**<br>GTY_QUAD_X1Y4<br>GTY_QUAD_X1Y3 | **GTY_QUAD_X1Y5**<br>GTY_QUAD_X1Y4<br>GTY_QUAD_X1Y3<br>GTY_QUAD_X1Y2 |
| | | X1Y0 | **GTY_QUAD_X1Y5** | GTY_QUAD_X1Y5<br>GTY_QUAD_X1Y4<br>**GTY_QUAD_X1Y3** | GTY_QUAD_X1Y5<br>GTY_QUAD_X1Y4<br>GTY_QUAD_X1Y3<br>**GTY_QUAD_X1Y2** |
| XCVC1902 | VSVA2197 | CPM Controller 0 | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y3** |
| | | CPM Controller 1 | N/A | **GTY_QUAD_X0Y5** | **GTY_QUAD_X0Y5** |
| | | X0Y2 | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y5**<br>GTY_QUAD_X0Y4<br>GTY_QUAD_X0Y3 | GTY_QUAD_X0Y6<br>**GTY_QUAD_X0Y5**<br>GTY_QUAD_X0Y4<br>GTY_QUAD_X0Y3 |
| | | X0Y1 | **GTY_QUAD_X0Y3** | GTY_QUAD_X0Y5<br>GTY_QUAD_X0Y4<br>**GTY_QUAD_X0Y3** | GTY_QUAD_X0Y6<br>GTY_QUAD_X0Y5<br>**GTY_QUAD_X0Y4**<br>GTY_QUAD_X0Y3 |
| | | X1Y2 | **GTY_QUAD_X1Y6** | **GTY_QUAD_X1Y6**<br>GTY_QUAD_X1Y5<br>GTY_QUAD_X1Y4 | GTY_QUAD_X1Y6<br>**GTY_QUAD_X1Y5**<br>GTY_QUAD_X1Y4<br>GTY_QUAD_X1Y3 |
| | | X1Y0 | **GTY_QUAD_X1Y3** | **GTY_QUAD_X1Y3**<br>GTY_QUAD_X1Y2<br>GTY_QUAD_X1Y1 | GTY_QUAD_X1Y3<br>**GTY_QUAD_X1Y2**<br>GTY_QUAD_X1Y1<br>GTY_QUAD_X1Y0 |

*Table 19:* **GT Locations** *(cont'd)*

| Device | Package | PCIe Blocks | GT QUAD (X16) | GT QUAD (X8) | GT QUAD (X4 and Below) |
|---|---|---|---|---|---|
| XCVC1902 | VSVD1760 | CPM Controller 0 | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y3** |
| | | CPM Controller 1 | N/A | **GTY_QUAD_X0Y5** | **GTY_QUAD_X0Y5** |
| | | X0Y2 | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y5**<br>GTY_QUAD_X0Y4<br>GTY_QUAD_X0Y3 | GTY_QUAD_X0Y6<br>**GTY_QUAD_X0Y5**<br>GTY_QUAD_X0Y4<br>GTY_QUAD_X0Y3 |
| | | X0Y1 | **GTY_QUAD_X0Y3** | GTY_QUAD_X0Y5<br>GTY_QUAD_X0Y4<br>**GTY_QUAD_X0Y3** | GTY_QUAD_X0Y6<br>GTY_QUAD_X0Y5<br>**GTY_QUAD_X0Y4**<br>GTY_QUAD_X0Y3 |
| | | X1Y2 | N/A | **GTY_QUAD_X1Y4** | **GTY_QUAD_X1Y4**<br>GTY_QUAD_X1Y3 |
| | | X1Y0 | N/A | **GTY_QUAD_X1Y4** | GTY_QUAD_X1Y4<br>**GTY_QUAD_X1Y3** |
| XCVM1802 | VFVC1760 | CPM Controller 0 | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y3** |
| | | CPM Controller 1 | N/A | **GTY_QUAD_X0Y5** | **GTY_QUAD_X0Y5** |
| | | X0Y2 | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y5**<br>GTY_QUAD_X0Y4<br>GTY_QUAD_X0Y3 | GTY_QUAD_X0Y6<br>**GTY_QUAD_X0Y5**<br>GTY_QUAD_X0Y4<br>GTY_QUAD_X0Y3 |
| | | X0Y1 | **GTY_QUAD_X0Y3** | GTY_QUAD_X0Y5<br>GTY_QUAD_X0Y4<br>**GTY_QUAD_X0Y3** | GTY_QUAD_X0Y6<br>GTY_QUAD_X0Y5<br>**GTY_QUAD_X0Y4**<br>GTY_QUAD_X0Y3 |
| | | X1Y2 | **GTY_QUAD_X1Y6** | **GTY_QUAD_X1Y6**<br>GTY_QUAD_X1Y5<br>GTY_QUAD_X1Y4 | GTY_QUAD_X1Y6<br>**GTY_QUAD_X1Y5**<br>GTY_QUAD_X1Y4<br>GTY_QUAD_X1Y3 |
| | | X1Y0 | **GTY_QUAD_X1Y3** | **GTY_QUAD_X1Y3**<br>GTY_QUAD_X1Y2<br>GTY_QUAD_X1Y1 | GTY_QUAD_X1Y3<br>**GTY_QUAD_X1Y2**<br>GTY_QUAD_X1Y1<br>GTY_QUAD_X1Y0 |

*Table 19:* **GT Locations** *(cont'd)*

| Device | Package | PCIe Blocks | GT QUAD (X16) | GT QUAD (X8) | GT QUAD (X4 and Below) |
|---|---|---|---|---|---|
| XCVM1802 | VSVA2197 | CPM Controller 0 | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y3** |
| | | CPM Controller 1 | N/A | **GTY_QUAD_X0Y5** | **GTY_QUAD_X0Y5** |
| | | X0Y2 | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y5**<br>GTY_QUAD_X0Y4<br>GTY_QUAD_X0Y3 | GTY_QUAD_X0Y6<br>**GTY_QUAD_X0Y5**<br>GTY_QUAD_X0Y4<br>GTY_QUAD_X0Y3 |
| | | X0Y1 | **GTY_QUAD_X0Y3** | GTY_QUAD_X0Y5<br>GTY_QUAD_X0Y4<br>**GTY_QUAD_X0Y3** | GTY_QUAD_X0Y6<br>GTY_QUAD_X0Y5<br>**GTY_QUAD_X0Y4**<br>GTY_QUAD_X0Y3 |
| | | X1Y2 | **GTY_QUAD_X1Y6** | **GTY_QUAD_X1Y6**<br>GTY_QUAD_X1Y5<br>GTY_QUAD_X1Y4 | GTY_QUAD_X1Y6<br>**GTY_QUAD_X1Y5**<br>GTY_QUAD_X1Y4<br>GTY_QUAD_X1Y3 |
| | | X1Y0 | **GTY_QUAD_X1Y3** | **GTY_QUAD_X1Y3**<br>GTY_QUAD_X1Y2<br>GTY_QUAD_X1Y1 | GTY_QUAD_X1Y3<br>**GTY_QUAD_X1Y2**<br>GTY_QUAD_X1Y1<br>GTY_QUAD_X1Y0 |
| XCVM1802 | VSVD1760 | CPM Controller 0 | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y3** |
| | | CPM Controller 1 | N/A | **GTY_QUAD_X0Y5** | **GTY_QUAD_X0Y5** |
| | | X0Y2 | **GTY_QUAD_X0Y3** | **GTY_QUAD_X0Y5**<br>GTY_QUAD_X0Y4<br>GTY_QUAD_X0Y3 | GTY_QUAD_X0Y6<br>**GTY_QUAD_X0Y5**<br>GTY_QUAD_X0Y4<br>GTY_QUAD_X0Y3 |
| | | X0Y1 | **GTY_QUAD_X0Y3** | GTY_QUAD_X0Y5<br>GTY_QUAD_X0Y4<br>**GTY_QUAD_X0Y3** | GTY_QUAD_X0Y6<br>GTY_QUAD_X0Y5<br>**GTY_QUAD_X0Y4**<br>GTY_QUAD_X0Y3 |
| | | X1Y2 | N/A | **GTY_QUAD_X1Y4** | **GTY_QUAD_X1Y4**<br>GTY_QUAD_X1Y3 |
| | | X1Y0 | N/A | **GTY_QUAD_X1Y4** | GTY_QUAD_X1Y4<br>**GTY_QUAD_X1Y3** |

*Appendix B*

# Upgrading

This appendix is not applicable for the first release of the core.

Appendix C

# Debugging

This appendix includes details about resources available on the Xilinx® Support website and debugging tools.

## Finding Help on Xilinx.com

To help in the design and debug process when using the core, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support. The Xilinx Community Forums are also available where members can learn, participate, share, and ask questions about Xilinx solutions.

### Documentation

This product guide is the main document associated with the core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx® Documentation Navigator. Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

### Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use keywords such as:

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

### *Master Answer Record for the Core*

Xilinx Answer 72289.

## Technical Support

Xilinx provides technical support on the Xilinx Community Forums for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To ask questions, navigate to the Xilinx Community Forums.

# Debug Tools

There are many tools available to address Versal™ ACAP PHY for PCIe® design issues. It is important to know which tools are useful for debugging various situations.

## Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx® devices.

The Vivado logic analyzer is used to interact with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908).

XILINX

*Appendix D*

# Additional Resources and Legal Notices

## Xilinx Resources

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

## Documentation Navigator and Design Hubs

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

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

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

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

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

## References

These documents provide supplemental material useful with this guide:

1. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
2. *Vivado Design Suite User Guide: Designing with IP* (UG896)
3. *Vivado Design Suite User Guide: Getting Started* (UG910)
4. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
5. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
6. *Vivado Design Suite User Guide: Implementation* (UG904)
7. *Versal ACAP GTY and GTYP Transceivers Architecture Manual* (AM002)
8. *Versal ACAP Transceivers Wizard LogiCORE IP Product Guide* (PG331)
9. *Versal ACAP Integrated Block for PCI Express LogiCORE IP Product Guide* (PG343)
10. *Versal ACAP CPM Mode for PCI Express Product Guide* (PG346)
11. *Versal ACAP CPM DMA and Bridge Mode for PCI Express Product Guide* (PG347)

# Revision History

The following table shows the revision history for this document.

| Section | Revision Summary |
|---|---|
| **10/15/2021 Version 1.0** | |
| Minimum Device Requirements | New section. |
| **11/24/2020 Version 1.0** | |
| Appendix A: GT Selection and Pin Planning | New appendix. |
| **07/16/2020 Version 1.0** | |
| Initial release | N/A |

# Please Read: Important Legal Notices

(including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

This document contains preliminary information and is subject to change without notice. Information provided herein relates to products and/or services not yet available for sale, and provided solely for information purposes and are not intended, or to be construed, as an offer for sale or an attempted commercialization of the products and/or services referred to herein.

**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**