

Versal ACAP CPM Mode for PCI Express

Product Guide

Vivado Design Suite

PG346 (v1.0) July 24, 2020



Table of Contents

Chapter 1: Overview	3
Navigating Content by Design Process.....	3
Introduction to the CPM4.....	4
Use Modes.....	9
Chapter 2: Product Specification	15
Clocking.....	15
Resets.....	16
Port Descriptions.....	17
Register Space.....	17
Chapter 3: Design Flow Steps	19
Customizing and Generating the CIPS IP Core.....	19
Appendix A: GT Selection and Pin Planning	41
CPM4 GT Selection.....	42
CPM4 Additional Considerations.....	44
GT Locations.....	44
Appendix B: PCIe Link Debug Enablement	48
Enabling PCIe Link Debug.....	48
Connecting to PCIe Link Debug in Vivado.....	52
Appendix C: Debugging	54
Finding Help on Xilinx.com.....	54
Appendix D: Additional Resources and Legal Notices	56
Xilinx Resources.....	56
Documentation Navigator and Design Hubs.....	56
References.....	57
Revision History.....	57
Please Read: Important Legal Notices.....	57

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. This document covers the following design processes:

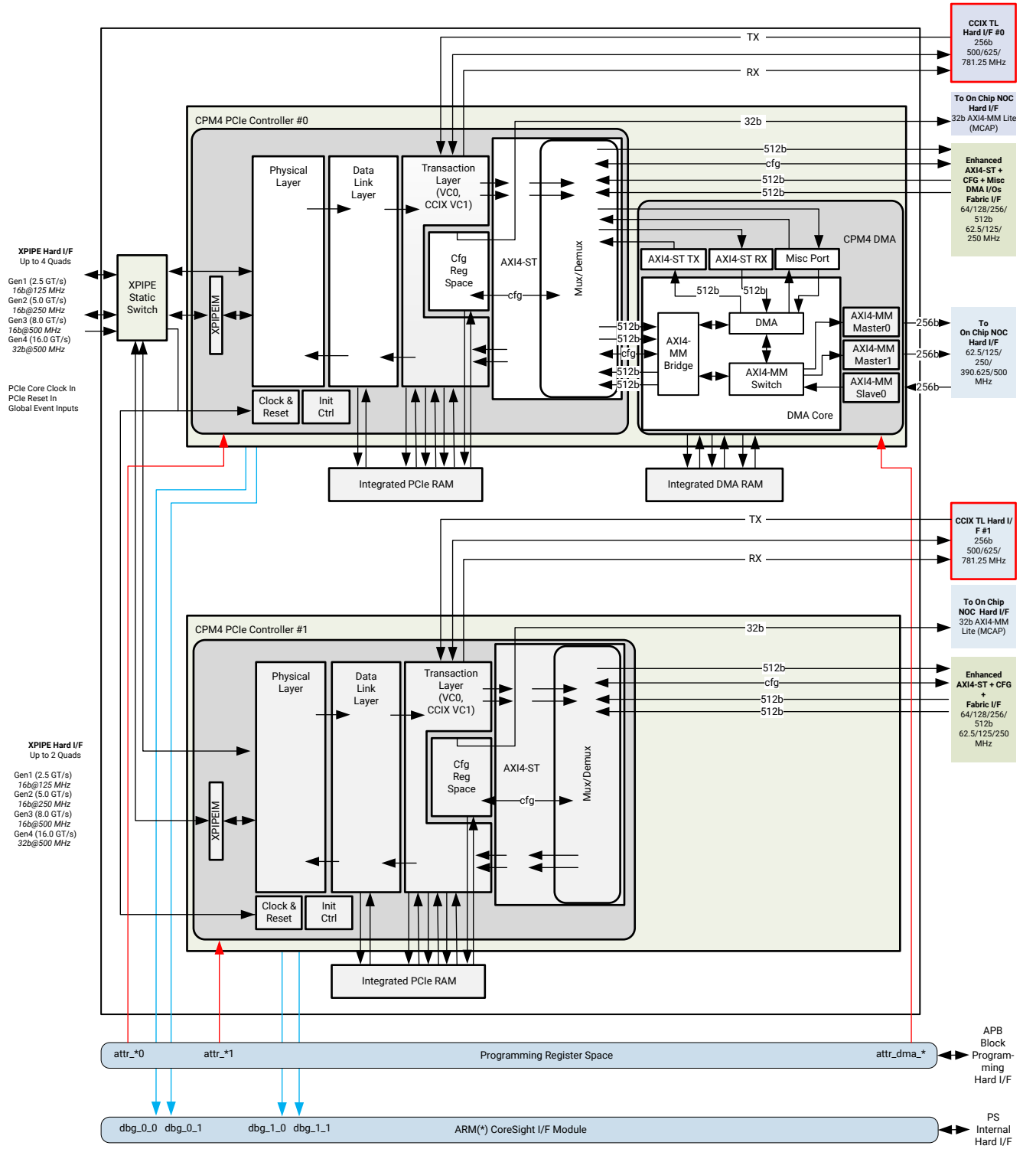
- **System and Solution Planning:** Identifying the components, performance, I/O, and data transfer requirements at a system level. Includes application mapping for the solution to PS, PL, and AI Engine. Topics in this document that apply to this design process include:
 - [Introduction to the CPM4](#)
 - [Use Modes](#)
- **Embedded Software Development:** Creating the software platform from the hardware platform and developing the application code using the embedded CPU. Also covers XRT and Graph APIs. The topic in this document that applies to this design process include:
 - [Register Space](#)
- **Host Software Development:** Developing the application code, accelerator development, including library, XRT, and Graph API use. The topic in this document that applies to this design process include:
 - [Register Space](#)
- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, subsystem functional simulation, and evaluating the Vivado[®] timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
 - [Chapter 3: Design Flow Steps](#)
 - [Appendix A: GT Selection and Pin Planning](#)
 - [Appendix B: PCIe Link Debug Enablement](#)

Introduction to the CPM4

The integrated block for PCIe Rev. 4.0 with DMA and CCIX Rev. 1.0 (CPM4) consists of two PCIe controllers, DMA features, CCIX features, and network on chip (NoC) integration. The Versal™ ACAP CPM Mode for PCI Express enables direct access to the two high-performance, independently customizable PCIe controllers. The CPM4 uses up to 16 Versal device GTY channels over the XPIPE. Application designs can also interface to the CPM4 with soft logic and clocking resources in the programmable logic. All feature references are applicable to both instances of CPM4 PCIe controllers, with the following exceptions:

- CPM4 PCIe controller #0 supports up to x16 operation, and CPM4 PCIe controller #1 supports up to x8 operation.
- CPM4 PCIe controller #1 with up to x8 support is available only when CPM4 PCIe controller #0 is configured with 8 lanes or fewer.
- The CPM4 DMA features are supported only with CPM4 PCIe controller #0. For more information about CPM4 DMA features, see the *Versal ACAP CPM DMA and Bridge Mode for PCI Express Product Guide* (PG347).

Figure 1: CPM4 Sub-Block for PCIe Function (CPM4 PCIe)



X22665-072320

The CPM4 PCIe controllers are designed to the PCI Express Base Specification Revision 4.0 and support the Gen 4 data rate (16 GT/s per lane). They also support the Gen1 (2.5 GT/s per lane), Gen2 (5 GT/s per lane) and Gen3 (8 GT/s per lane) data rates, and can interoperate with components that are compliant with all versions of the PCI Express Base Specification.

The CPM4 PCIe controllers are available through the Vivado IP catalog in the Vivado Integrated Design Environment (IDE). The combination of the CPM4 PCIe controllers, the GTY, and clocking implement all layers of the PCI Express protocol, and the configuration space and controller.

Protocol Layers

The layers of the protocol are the AXI4-Stream layer, the transaction layer, the data link layer and the physical layer, and they are described below.

AXI4-Stream Layer

The AXI4-Stream layer implements Xilinx-specific requirements. In the transmit or outbound direction, the AXI4 layer interfaces the transaction layer with two AXI4-Stream interfaces. In the receive or inbound direction, the transaction layer output is forwarded to two AXI4-Stream interfaces. Application designs can attach to the AXI4-Stream interfaces, exchange information with the Versal™ ACAP CPM Mode for PCI Express encoded as a Xilinx-specific streaming protocol implementation, and run on top of the industry standard AXI4-Stream interface. The CPM4 PCIe controllers support management of up to 256 (extended tag) or 1024 (scaled tag) outstanding customer initiated read requests, as part of the streaming protocol. The AXI4-Stream layer supports:

- Reception and transmission of address translation services (ATS) invalid requests, ATS invalid completions, ATS page requests and ATS PRG response message TLPs, which enable ATS to be implemented in the fabric logic.
- AXI4-Stream interface widths of 64 bits, 128 bits, 256 bits and 512 bits.

Transaction Layer

The transaction layer is the upper layer of the PCI Express architecture, and its primary function is to accept, buffer, and forward transaction layer packets (TLPs). TLPs communicate information with the use of memory, I/O, configuration, and message transactions. To maximize the efficiency of communication between devices, the transaction layer enforces PCI-compliant transaction ordering rules and supports relaxed ordering (RO) of received transactions. The transaction layer also manages TLP buffer space through credit-based flow control. The transaction layer implements built-in tag management for transmitted non-posted transactions. It also implements cut-through forwarding of transactions in the transmit (or outbound) direction.

CCIX Transaction Layer

The Cache Coherent Interconnect for Accelerators (CCIX) transaction layer requirements are implemented by the optional virtual channel 1 (VC1) in the design. Note that VC1 storage is in addition to the PCI Express-compliant virtual channel 0 (VC0) storage. The CCIX transaction layer interfaces with the CCIX protocol layer is implemented externally to the PCIe ports over the CCIX transaction layer (ARM CXS) hard interface. For more information, see the *Versal ACAP CPM CCIX Architecture Manual* (AM016).

Data Link Layer

The data link layer acts as an intermediate stage between the transaction layer and the physical layer. Its primary responsibility is to provide a reliable mechanism for the exchange of information between two components on a link. This includes data exchange (TLPs), error detection and recovery, initialization services and the generation and consumption of data link layer packets (DLLPs). DLLPs are used to transfer information between data link layers of two directly connected components on the link. DLLPs convey information, such as power management, flow control, and TLP acknowledgments. The data link layer supports 32 kilobyte replay buffers and the feature DLLP.

Physical Layer

The physical layer interfaces the data link layer with signaling technology for link data interchange, and is subdivided into the logical sub-block and the electrical sub-block.

- The logical sub-block frames and de-frames TLPs and DLLPs. It also implements the link training and status state machine (LTSSM), which handles link initialization, training, and maintenance. Scrambling and descrambling of data (for Gen1/Gen2/Gen3/Gen4 operation) is also performed in this sub-block.
- The electrical sub-block defines the input and output buffer characteristics that interface the device to the PCIe link. The physical layer also supports lane reversal (for multi-lane designs) and lane polarity inversion, as required by the *PCI Express Base Specification 4.0* (<http://www.pcisig.com/specifications>).

Data exchange with the other components on the link occurs over the serial lines of one or more gigabit transceivers (GTs), which expose parallel interfaces at lower clock frequencies to the PCIe controller. For Gen1, Gen2, Gen3 and Gen4 operation, the physical layer is up-configuration capable in the downstream port mode only.

Standards

The CPM4 block adheres to the following standards:

- PCI Express Base Specification 4.0 Version 1.0, and Errata updates (available at <http://pcisig.com/specifications>).

- Cache Coherent Interconnect for Accelerators (CCIX) Transport Specification 1.0 (available at <http://www.ccixconsortium.com>).

Features

- Support for the following PCI Express architecture components:
 - PCI Express Endpoint, Legacy Endpoint
 - Root Port
 - Switch Upstream and Downstream Ports
- x1, x2, x4, x8 or x16 link widths
- Gen1, Gen2, Gen3 or Gen4 link speeds
- CCIX support in PCI Express and EDR PHY Modes
 - PCI Express support for Gen4x4, and Gen4x8
- Advanced Error Reporting (AER) and End-to-End CRC (ECRC)
- Two PCI Express virtual channels
 - One PCI Express compliant virtual channel, eight traffic classes
 - One CCIX compliant virtual channel
- Support for multiple functions and Single-Root IO Virtualization (SR-IOV)
 - Up to four physical functions
 - Up to 252 virtual functions
- Built-in lane reversal and receiver lane-lane de-skew
- 3 x 64-bit or 6 x 32-bit Base Address Registers (BARs) that are fully configurable
 - Expansion ROM BAR supported
- All Interrupt types are supported:
 - INTx
 - 32 multi-vector MSI capability
 - MSI-X capability with up to 2048 vectors with optional built-in vector tables
- Features that enable high-performance applications include:
 - AXI4-Stream TLP Straddle on Requester Completion Interface
 - Address Translation Services (ATS) and Page Request Interface (PRI) Messaging
 - Atomic Operation Transactions Support

- Transaction Tag Scaling as Completer
- Flow Control Scaling

Use Modes

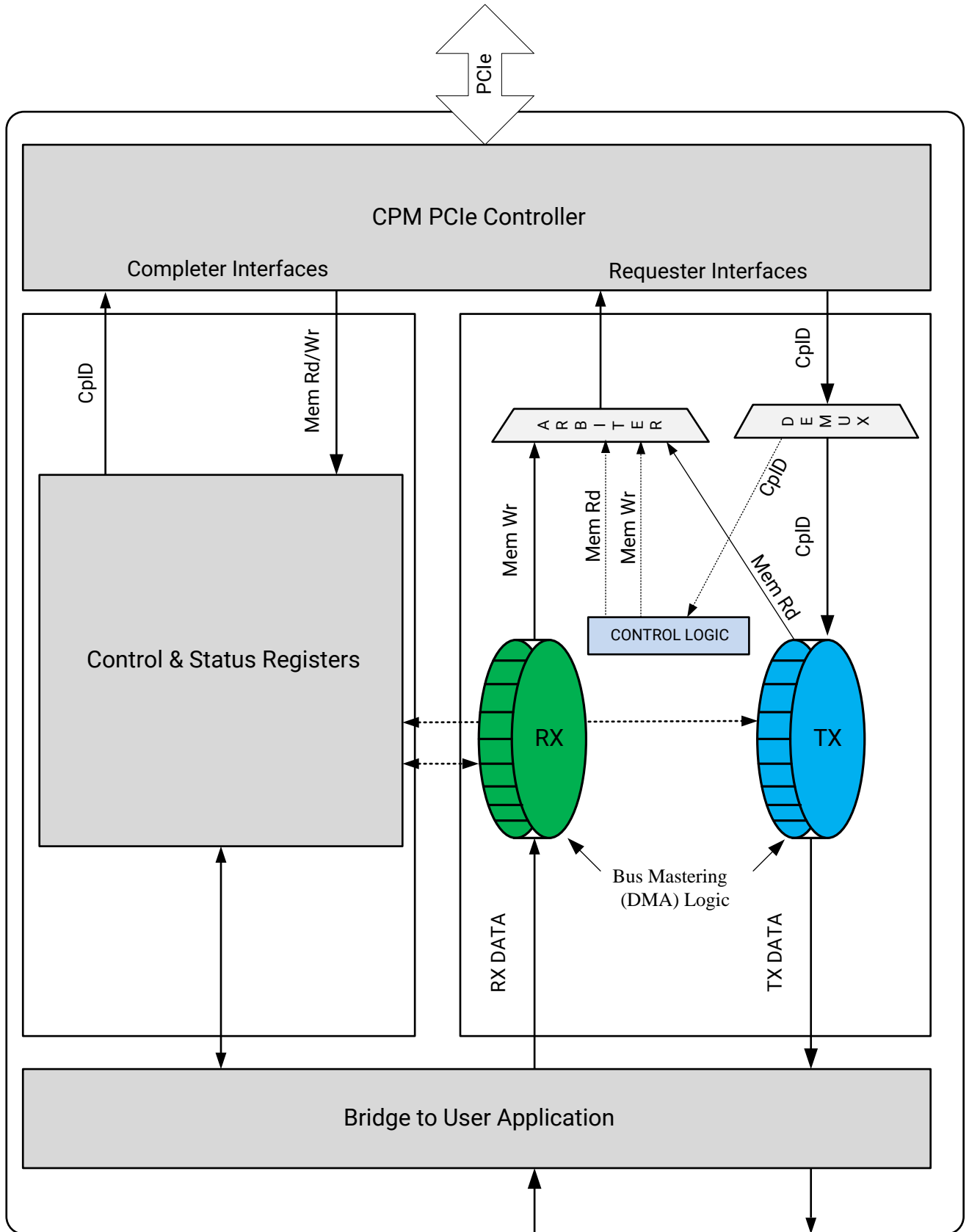
All design use modes support Endpoint, Legacy Endpoint, and Root Port configurations.

PCI Express Endpoint Use Modes

Illustrative Example of Basic Bus Mastering Endpoint

By far the most common use of the Versal™ ACAP CPM Mode for PCI Express is to construct a bus mastering Endpoint using a CPM PCIe controller. This use model is applicable to most applications that interface the Endpoint port on the ACAP (on an add-in card) to a root complex or that switch ASSP downstream port through a PCI Express connector. The following figure shows a block diagram of the bus mastering Endpoint use case.

Figure 2: Basic PCI Express Bus Mastering Endpoint Use Case

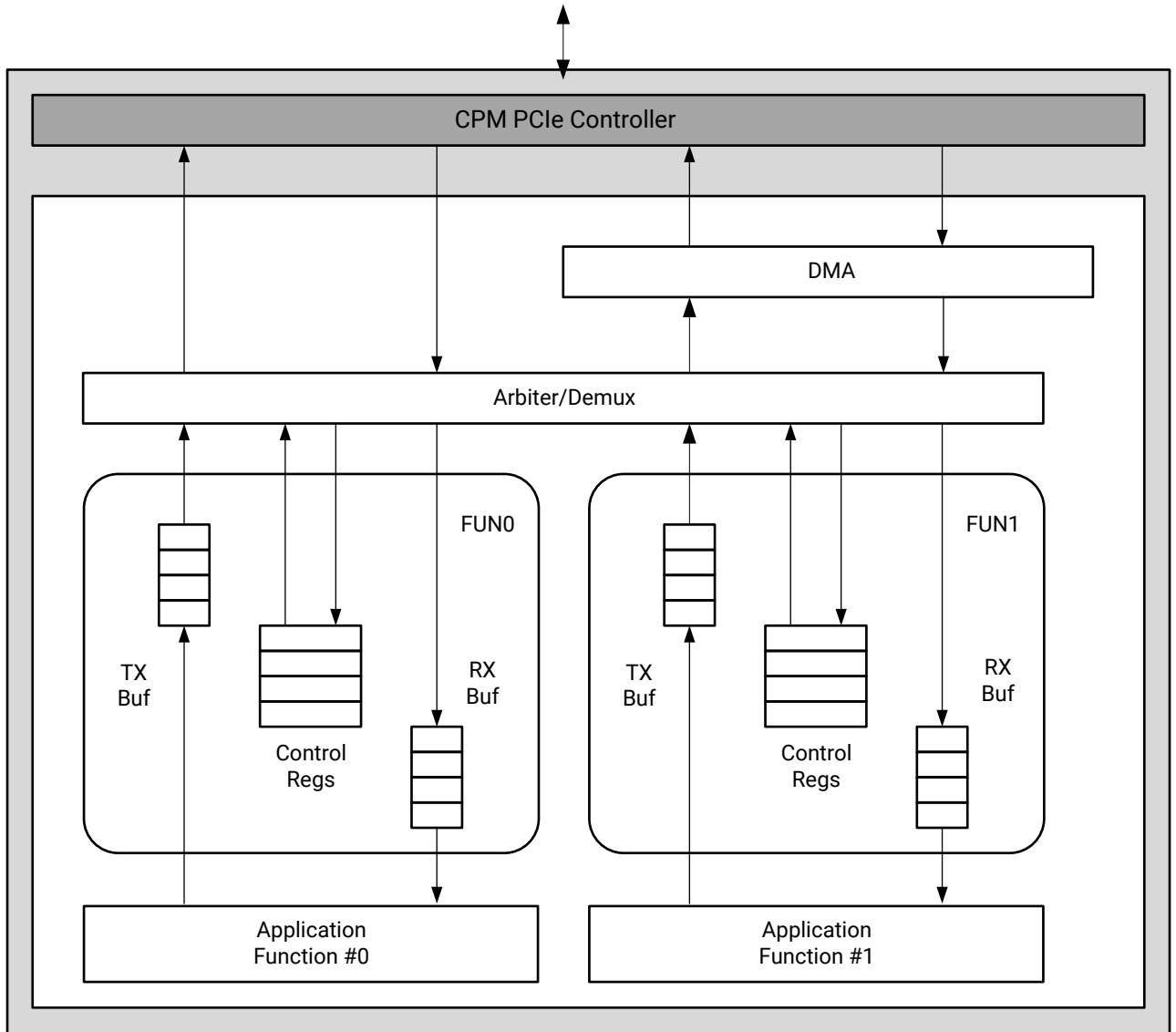


X22666-071620

PCI Express Two Function Endpoint

The following figure shows the architecture of a two-function Endpoint design. The CPM PCIe Controller is configured to enable two built-in function configuration spaces. This use case enables the application device driver to access and control two distinct applications independently. The user logic implements the DMA, control registers and applications.

Figure 3: Illustrative Example of Two Function Endpoint Use Case

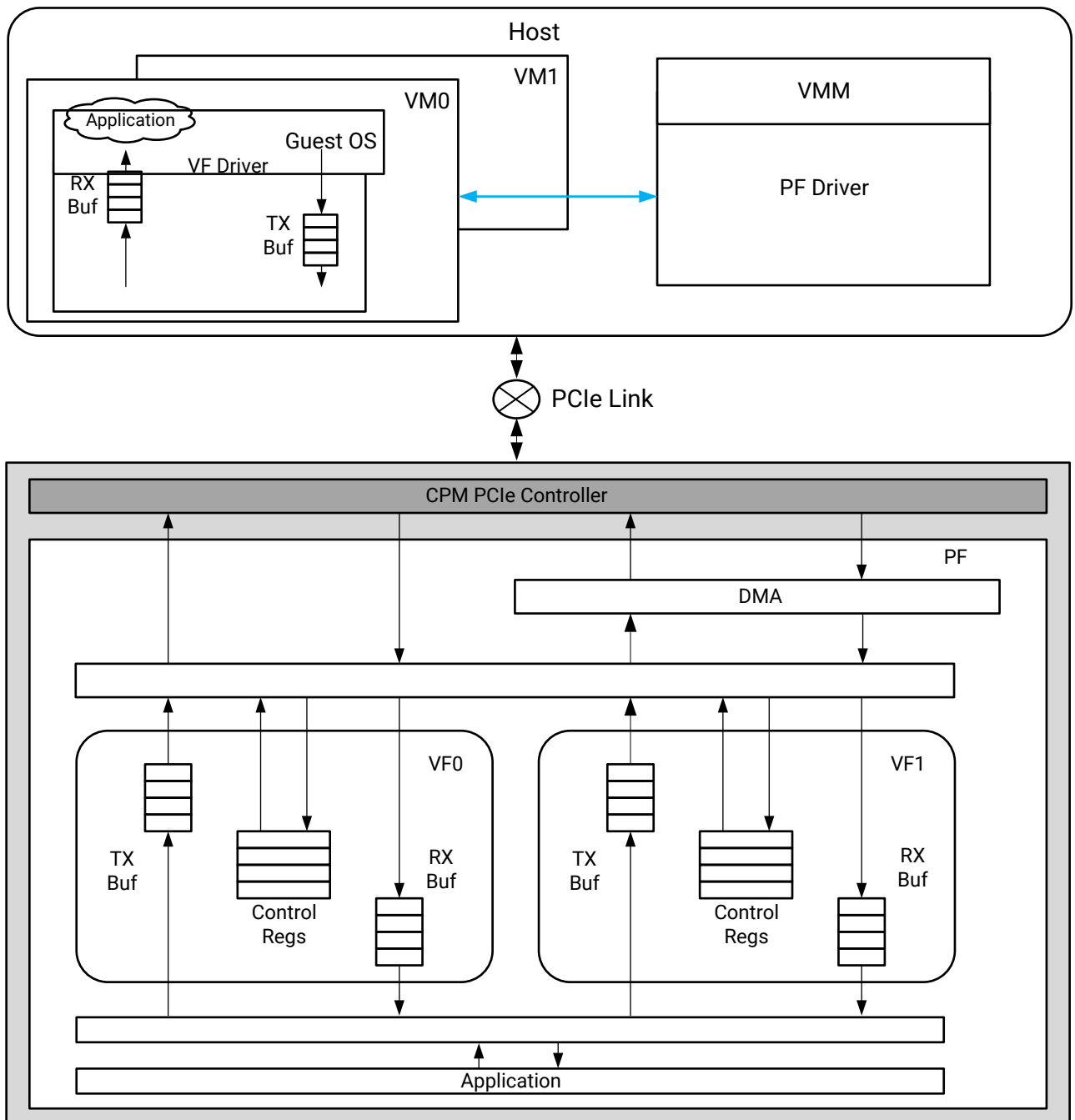


X22667-071620

PCI Express Endpoint with SR-IOV

The following figure shows the CPM PCIe Controller configured as a SR-IOV capable Endpoint, interfacing with the user design. This use case addresses requirements for up to four physical and 252 virtual functions, and minimizes the soft logic requirement to implement an SR-IOV Endpoint.

Figure 4: Illustrative Example of Endpoint with SR-IOV Use Case



X22668-071620

PCI Express Endpoint with AXI4 Memory Mapped Interface

This use case describes a PCI Express Endpoint functional unit that implements AXI4 Memory Mapped (AXI-MM) interfaces. This functional block implements a soft logic bridge between the native AXI4-Stream interface on the CPM PCIe controller and AXI4 Memory Mapped interconnect.

PCI Express Endpoint Using Staged Configuration Flow

This use case addresses the ability to configure the ACAP and bring up the CPM block in less than 100 ms after power to the ACAP is stable. This will be accomplished through a staged configuration flow.

PCI Express Endpoint Using Customizable Tandem Design

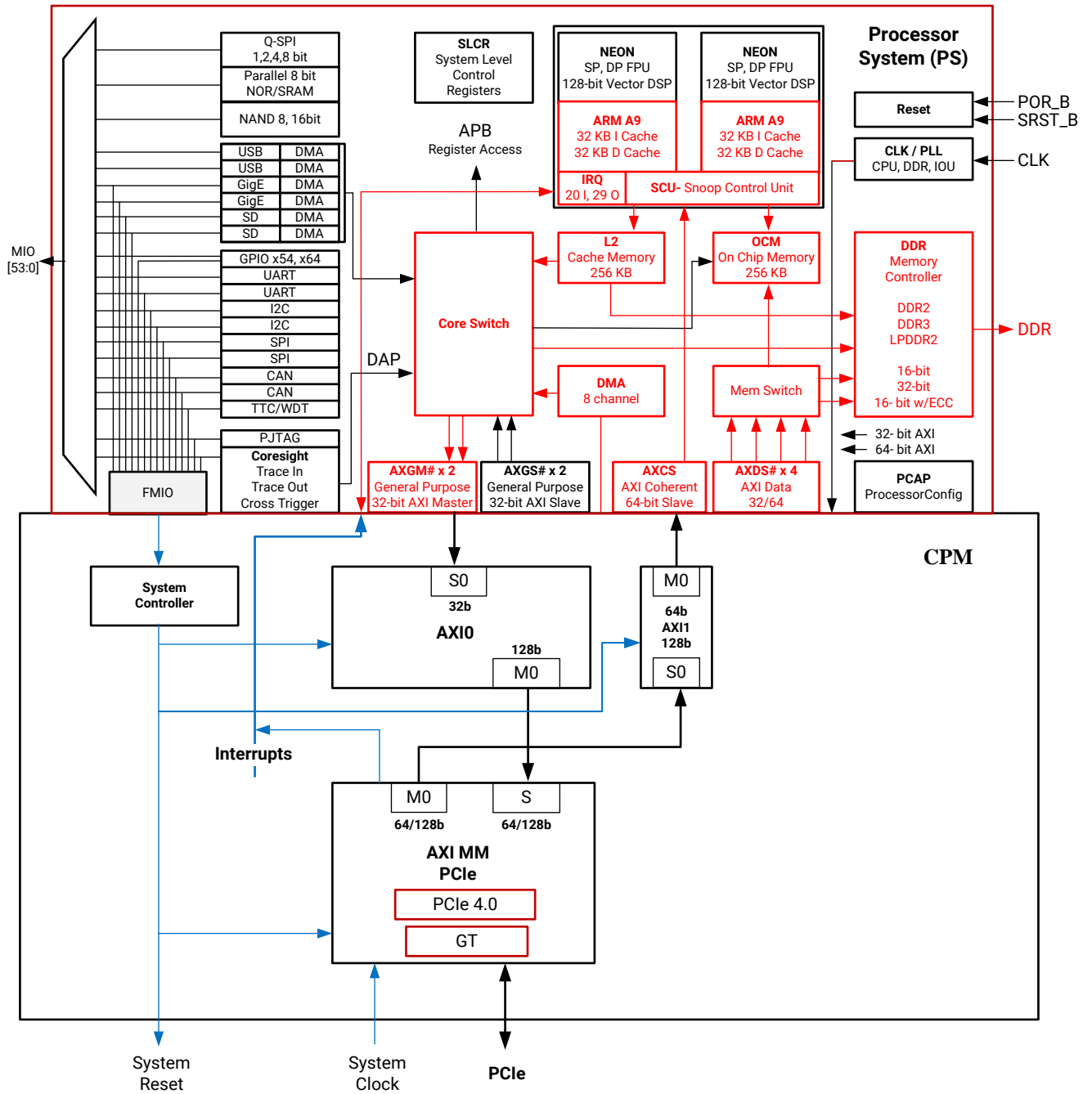
This use case addresses the ability to initially load fully configurable PCI Express protocol solution from a small external ROM, so as to meet the 100 ms configuration requirement. A PCIe link is formed with a Root Complex or Switch component, which is subsequently used to download the design that configures the rest of the ACAP. In this case the PCIe link is used by the user application. A light weight staged configuration flow is used.

PCI Express Root Port Use Mode

Basic PCI Express Root Complex Use Case

The following figure shows a PCI Express Root Complex in the simplest form consisting of a PCI Express Root Port to an AXI4 memory mapped bridge interfaced with the interconnect. The interconnect consists of an Arm[®]-based processor system (PS) containing most of the critical blocks such as CPU, memory controller and other important peripherals. One of the goals of this use case is to minimize ACAP soft logic requirements.

Figure 5: Basic PCI Express Root Complex Use Case



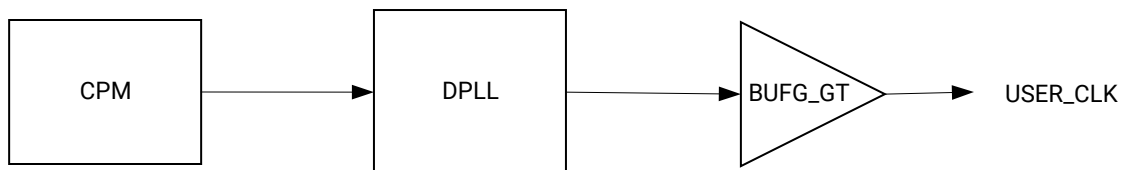
X22670-051319

Product Specification

Clocking

The Versal™ ACAP CPM Mode for PCI Express® requires a 100, 125, or 250 MHz reference clock input. The following figure shows the clocking architecture. The `user_clk` clock is available for use in the fabric logic.

Figure 6: USER_CLK Clocking Architecture



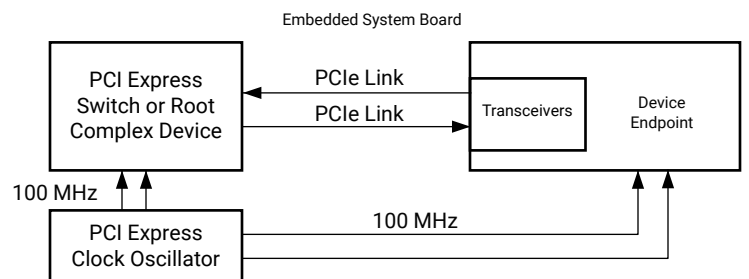
X22710-071520

All user interface signals are timed with respect to the same clock (`user_clk`) which can have a frequency of 62.5, 125, or 250 MHz depending on the configured link speed and width.

Each link partner device shares the same reference clock source. The following figures show a system using a 100 MHz reference clock. Even if the device is part of an embedded system, if the system uses commercial PCI Express root complexes or switches along with typical motherboard clocking schemes, synchronous clocking should still be used.

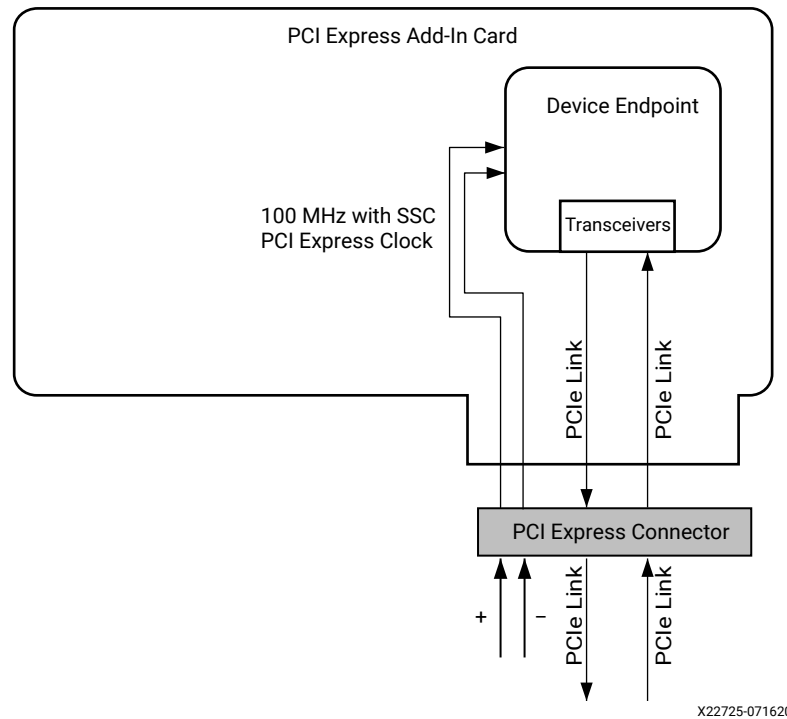
Note: The following figures are high-level representations of the board layout. Ensure that coupling, termination, and details are correct when laying out a board.

Figure 7: Embedded System Using 100 MHz Reference Clock



X22724-051419

Figure 8: Open System Add-In Card Using 100 MHz Reference Clock



Resets

The fundamental resets for the CPM PCIe controllers and associated GTs are `perst0n` and `perst1n`. The resets are driven by the I/O inside the PS. In addition, there is a power-on-reset for CPM driven by the platform management controller (PMC). When both PS and the power-on reset from PMC are released, CPM PCIe controllers and the associated GTs will come out of reset.

After the reset is released, the core attempts to link train and resumes normal operation.

In addition, there is a `user_reset` given from the CPM PCIe controller to the user design present in the fabric logic. Whenever the CPM PCIe block goes through a reset, or there is a link down, the CPM PCIe controller issues a `user_reset` to the user design in the programmable logic (PL) region. After the PCIe link is up, `user_reset` is released for the user design to come out of reset.

Port Descriptions

The interfaces and ports of the Versal™ ACAP CPM Mode for PCIe are similar to those described in the *Versal ACAP Integrated Block for PCI Express LogiCORE IP Product Guide (PG343)*, except that signal names used for this solution begin with the letters `ifcpm`. Detailed information will be provided in a future release.

Register Space

The configuration space is a register space defined by the *PCI Express Base Specification 4.0* (<http://www.pcisig.com/specifications>). The Versal™ ACAP CPM Mode for PCIe supports Xilinx proprietary read/write configuration interfaces into this register space, and supports up to four Physical Functions (PFs) and 252 Virtual Functions (VFs).

The PCI configuration space consists of the following primary parts.

Legacy PCI v4.0 Type 0/1 Configuration Space Header

- Type 0 Configuration Space Header supported for Endpoint configuration
- Type 1 Configuration Space Header supported for Root, Switch Port configuration

Legacy Extended Capability Items

- PCIe Capability
- Power Management Capability
- Message Signaled Interrupt (MSI) Capability
- MSI-X Capability
- Legacy Extend Capabilities

PCIe Extended Capabilities

- Advanced Error Reporting Capability
- Function Level Reset
- ASPM L1 Support
- ASPM L0s Support (supported in Gen1 and Gen2 configurations only)
- Device Serial Number Capability
- Virtual Channel Capability

- ARI Capability (optional)
- SR-IOV Extended Capability Structure
- Configuration Space Extend Capabilities
- Address Translation Services (ATS)
- Page Request Interface (PRI)
- Feature DLLP
- CCIX Transport DVSEC through configuration space extension
- CCIX Protocol DVSEC through configuration space extension
- Transaction Tag Scaling as Requester and Completer
- Flow Control Scaling
- MCAP Interface for Staged Configuration and Dynamic Function eXchange per PCI Express port

Design Flow Steps

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

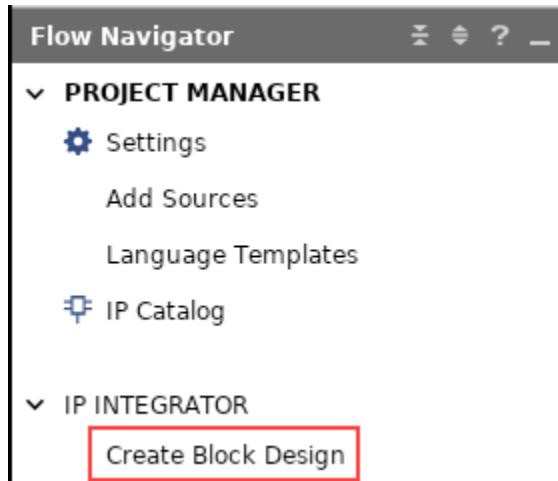
- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* ([UG994](#))
- *Vivado Design Suite User Guide: Designing with IP* ([UG896](#))
- *Vivado Design Suite User Guide: Getting Started* ([UG910](#))
- *Vivado Design Suite User Guide: Logic Simulation* ([UG900](#))

Customizing and Generating the CIPS IP Core

This section includes information about using the Vivado® Design Suite to customize and generate the Control, Interfaces, and Processing System IP core. This section configures the CIPS IP core to access the CPM PCIe controllers directly. For extended information about the CIPS IP core, see the *Control Interface and Processing System IP Product Guide* (PG352).

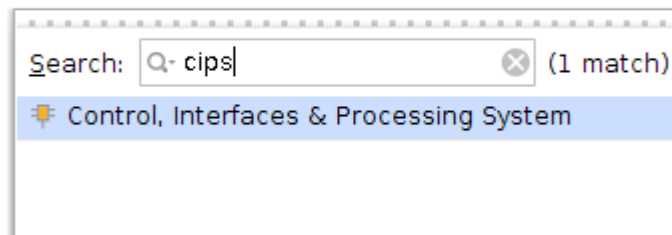
Configuring the CIPS IP Core

1. In the Vivado IDE, select **IP Integrator** → **Create Block Design** from the Flow Navigator, as shown in the following figure.

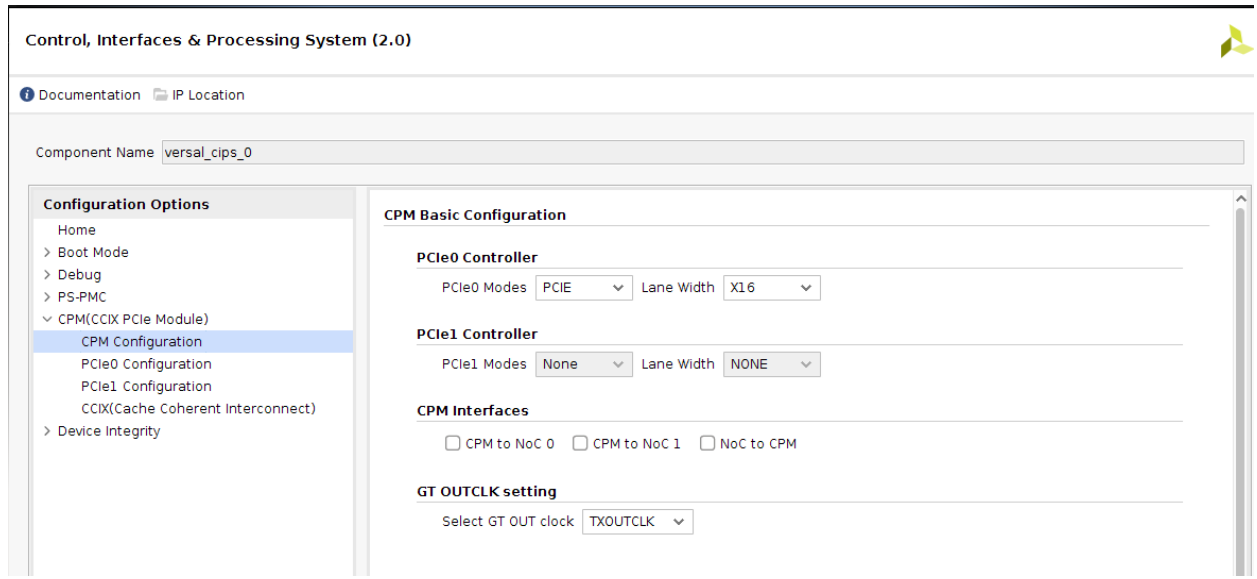


A popup dialog displays to create the block design.

2. Click **OK**. An empty block design diagram canvas opens in the IP integrator.
3. Right-click on the block design canvas and from the context menu select **Add IP**.
4. Search for `cips`.



5. Double-click the **Control, Interface, and Processing System** IP core to customize it.
6. In the Configuration Options pane, expand **CPM**, and click **CPM Configuration**.
The CPM Basic Configuration page displays.



7. Set the PCIe0 Modes to **PCIe**, and select the lane width. This setting enables the PCIe Port 0, and in a later step, you will be configuring the PCIe Port 0.
8. If you require the PCIe Port 1, set the PCIe1 Modes to **PCIe**, and select the lane width.

Available lane width combinations are:

PCIe Port 0	PCIe Port 1
X1, X2, X4, or X8	X1, X2, X4, or X8
X16	Not available

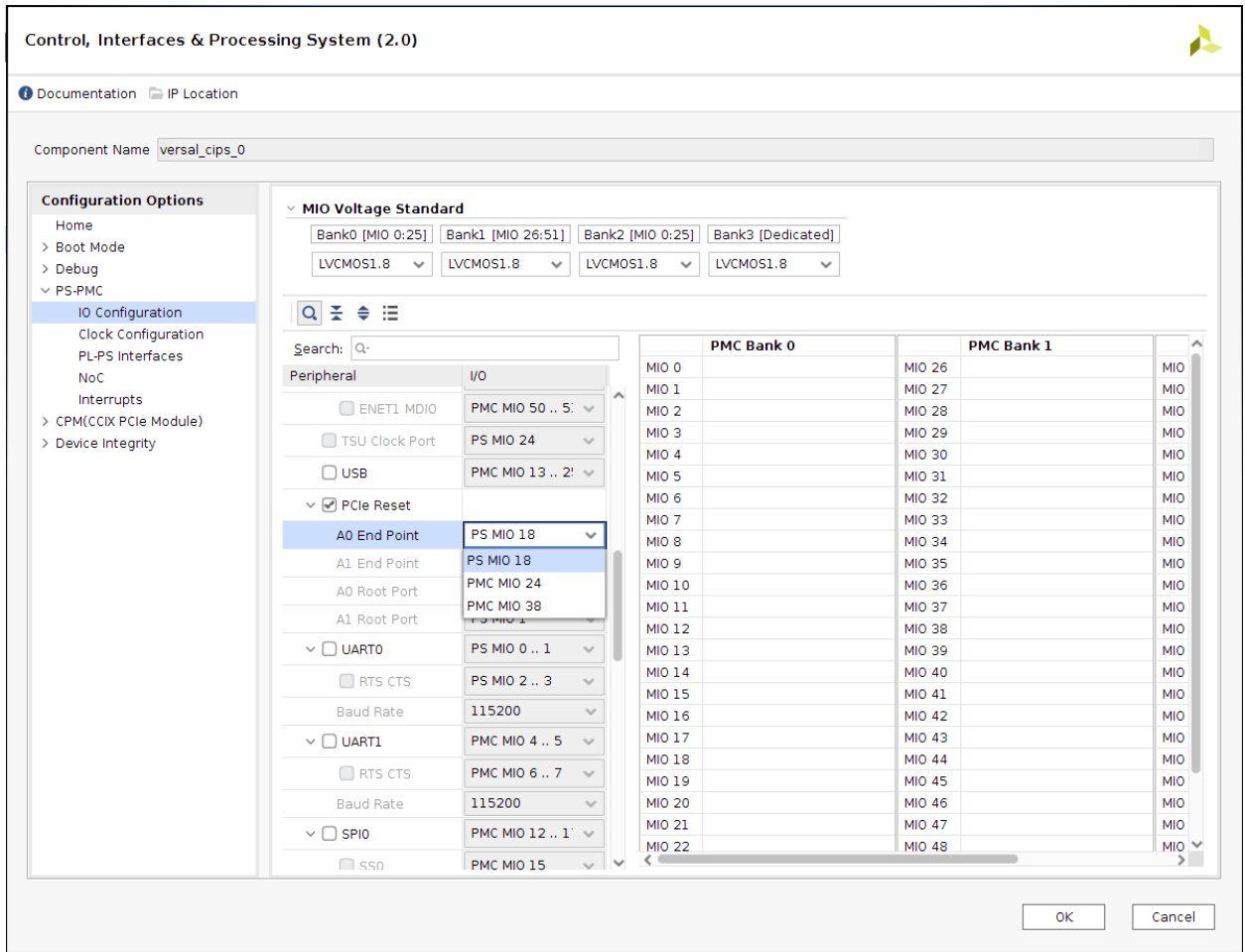
Note: There is no PCIe Port 1-only option available.

Note: PCIe Port 1 is available only if the lane width of PCIe Port 0 is less than or equal to X8.

Note: PCIe Port 1 supports up to X8 when PCIe Port 0 is configured up to X8.

9. Leave the Select GT OUT clock option set to **TXOUTCLK**.
10. In the Configuration Options pane, expand **PS-PMC**, and click **IO Configuration**.

The IO Configuration tab has a list of options to configure the external PCIe Reset options.



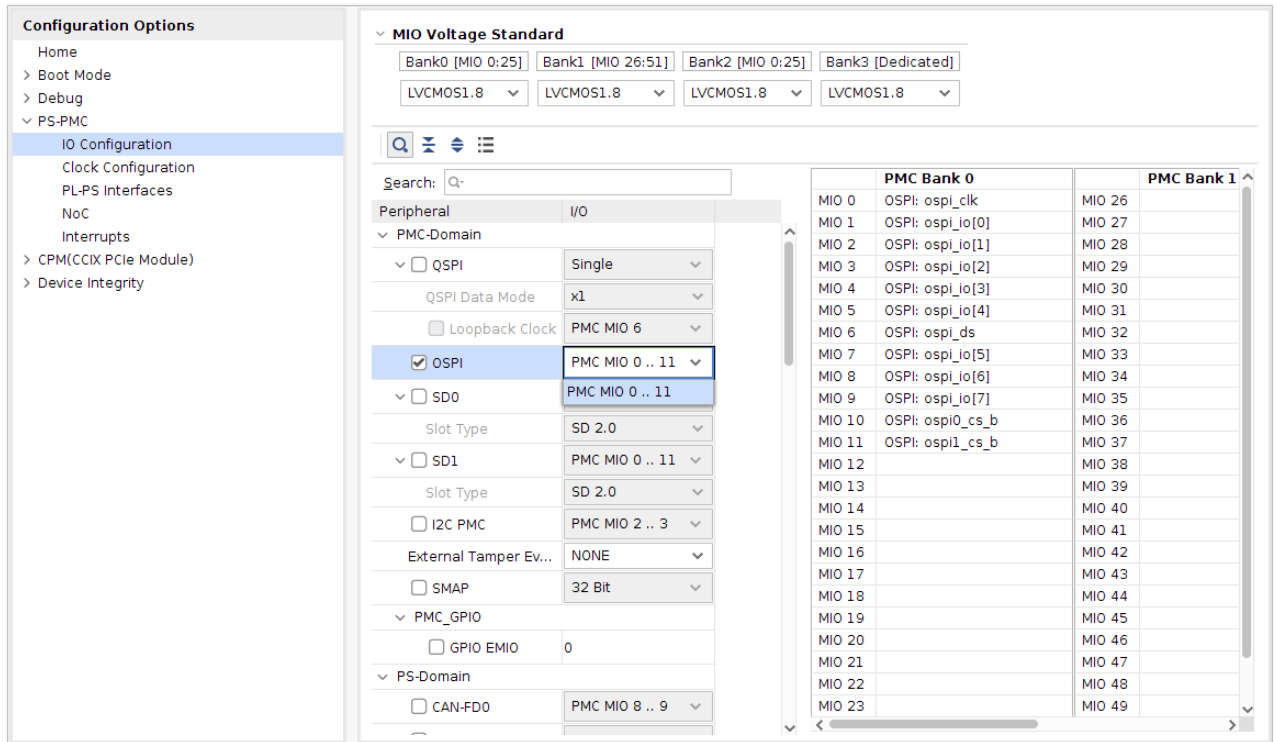
11. Select the **PCIe Reset** option located in the Peripheral column.

Notice that the MIO pin selected in the PCIe reset is automatically connected to the PCIe reset I/O. In the figure below, MIO 38 is connected to the PCIe reset I/O.

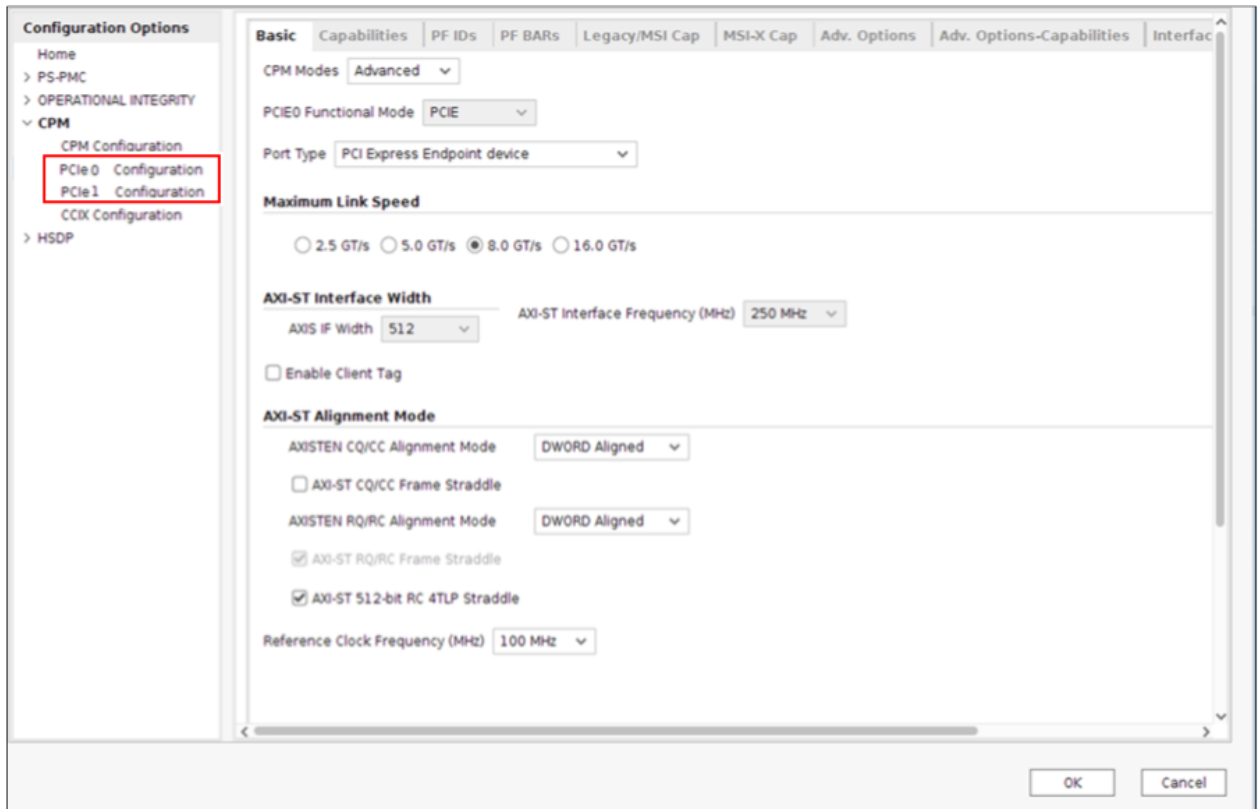
- a. For PCIe Port 0 End Point configuration: Next to A0 End Point, select one of the available MIOs: **PS MIO 18**, **PMC MIO 24**, or **PMC MIO 38**. This selection should match the MIO that is connected to the PCIe reset I/O in the board schematic. In the figure below, PMC MIO 38 is selected to correspond to PCIe reset MIO 38.
- b. If PCIe Port 1 is enabled: Next to A1 End Point, select one of the available MIOs (**PS MIO 19**, **PMC MIO 25**, or **PMC MIO 39**) based on which MIO is connected to the PCIe reset I/O in the board schematic.

12. If the board will boot from serial NOR flash, select the a **QSPI** or **OSPI** option to enable programming of the flash on the board. Select the appropriate MIOs based on availability to match the board schematic.

To set up the boot device, see the *Versal ACAP Technical Reference Manual (AM011)*. If a serial NOR flash boot device will be used, the correct options must be selected to enable the correct MIOs.



13. To enable additional I/O interfaces, such as UART, 12C, and USB IOs, select them here in a similar manner. See the *Versal ACAP Technical Reference Manual (AM011)* for more details on these interfaces.
14. In the Configuration Options pane, expand **CPM**, and click **PCIE 0 Configuration** to customize the PCIe Port 0 for the Versal ACAP CPM Mode for PCI Express core. It offers two modes: Basic, and Advanced. To select a mode, use the CPM Modes drop-down list on the first page of the Customize IP dialog box. The following sections explain the parameters available in each mode.



15. If applicable, in the Configuration Options pane, expand **CPM**, and click **PCIE 1 Configuration** to customize PCIe Port 1.

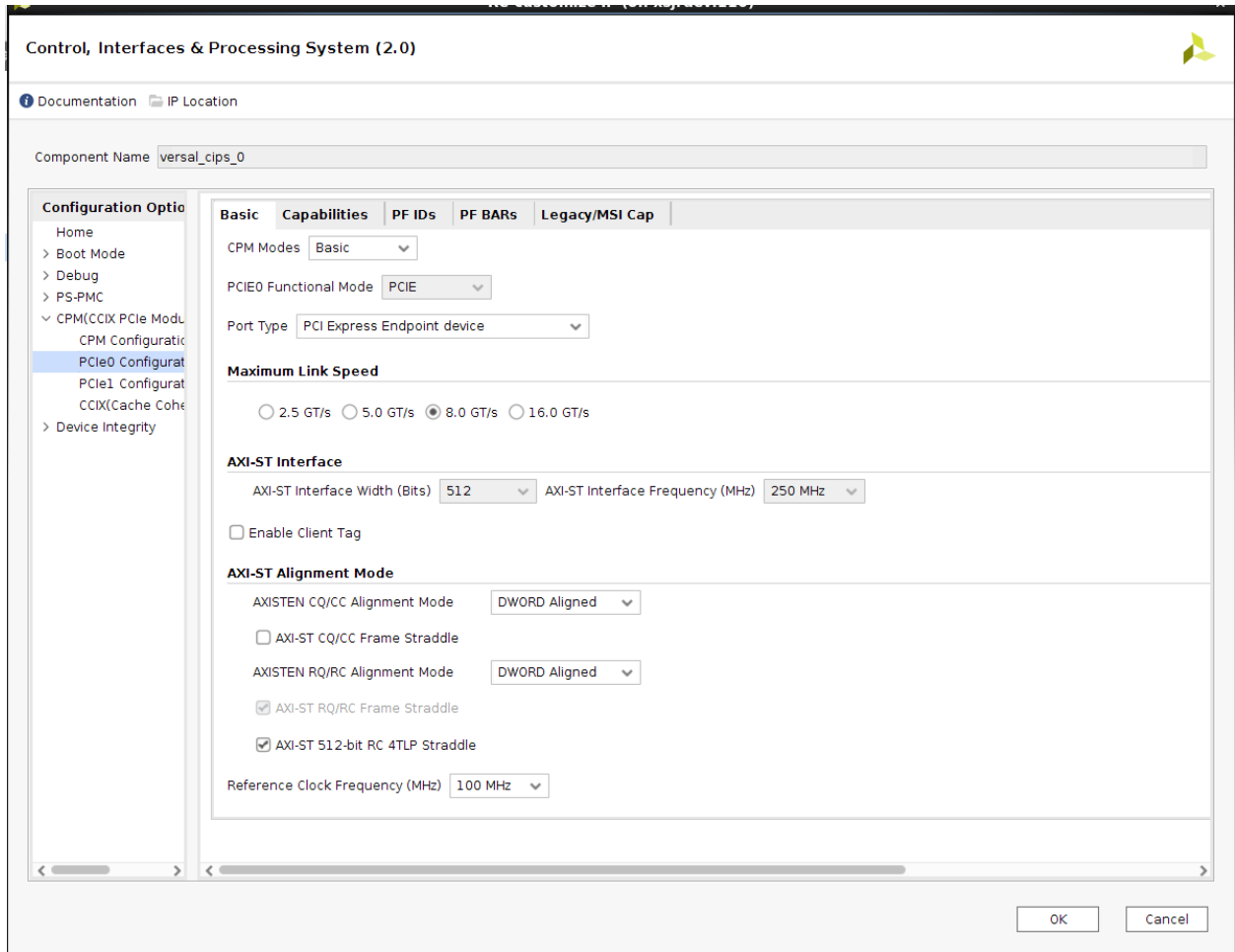
Basic Mode Parameters

The Basic mode parameters are explained in this section.

Basic Tab

The following figure shows the initial customization page, used to set the Basic mode parameters.

Figure 9: Basic Tab



- **Component Name:** 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 “_.”
- **CPM Mode:** Allows you to select the Basic or Advanced mode of the configuration of core.
- **Port Type:** Indicates the PCI Express logical device type.
- **Maximum Link Speed:** The core allows you to select the Maximum Link Speed supported by the device. Higher link speed cores are capable of training to a lower link speed if connected to a lower link speed capable device.
- **AXI-ST Interface Width:** The core allows you to select the Interface Width. The default interface width set in the Customize IP dialog box is the lowest possible interface width.
- **AXI-ST Interface Frequency:** Enables you to specify the AXI-ST Interface frequency.
- **Enable Client Tag:** Enables you to use the client tag.

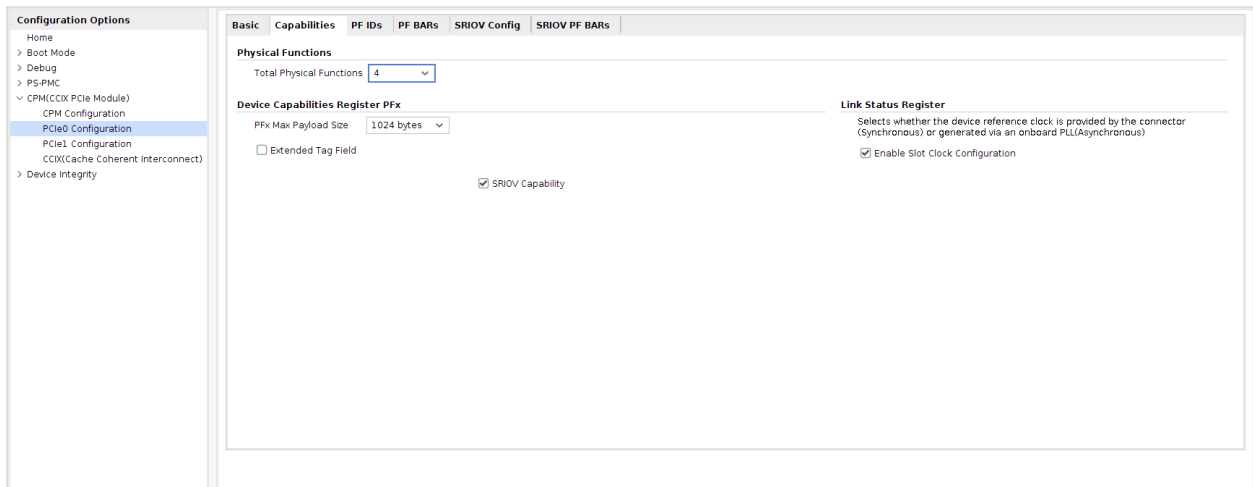
- **AXI-ST Alignment Mode:** When a payload is present, there are two options for aligning the first byte of the payload with respect to the datapath. The options are provided to select the CQ/CC and RQ/RC interfaces.
- **Enable AXI-ST Frame Straddle:** The core provides an option to straddle packets on the requester completion interface when the interface width is 256 bits.
- **AXI-ST CQ/CC Frame Straddle and AXI-ST RQ/RC Frame Straddle:** When 512-bit AXI-ST interface width is selected AXI-ST frame Straddle is supported for CQ, CC, RQ and RC AXI-ST interfaces. Option to select CQ and CC AXI-ST frame straddle together and for RQ and RC interfaces
- **Reference Clock Frequency:** Selects the frequency of the reference clock provided on `sys_clk`.

Related Information

[Clocking](#)

Capabilities Tab

The Capabilities settings are explained in this section as shown in the following figure.



- **Physical Functions:** Enables you to select the number of physical functions. The number of physical functions supported is 4.
- **PFX Max Payload Size:** This field indicates the maximum payload size that the device or function can support for TLPs. This is the value advertised to the system in the Device Capabilities Register.
- **Extended Tag Field:** This field indicates the maximum supported size of the Tag field as a Requester. The options are:
 - When selected, 8-bit Tag field support (256 tags)

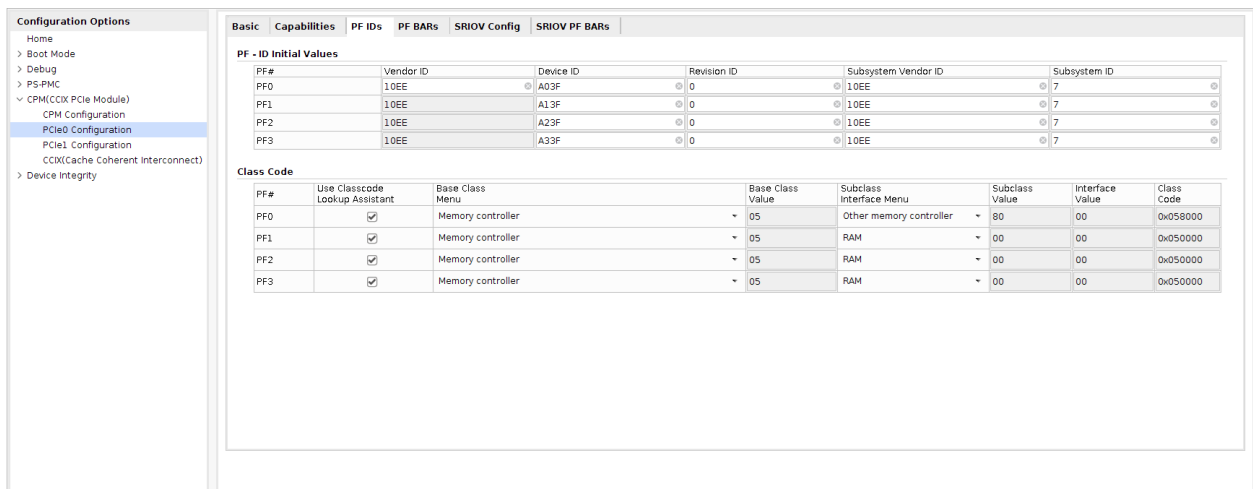
- When deselected, 5-bit Tag field support (32 tags)
- **Enable Slot Clock Configuration:** Enables the Slot Clock Configuration bit in the Link Status register. When you select this option, the link is synchronously clocked.
- **SRIOV Capabilities:** Enables Single Root Port I/O Virtualization (SR-IOV) capabilities. The integrated block implements extended Single Root Port I/O Virtualization PCIe. When this is enabled, SR-IOV is implemented on all the selected physical functions.

Related Information

[Clocking](#)

PF IDs Tab

The following figure shows the Identity Settings parameters.



PF#	Vendor ID	Device ID	Revision ID	Subsystem Vendor ID	Subsystem ID
PF0	10EE	A03F	0	10EE	7
PF1	10EE	A13F	0	10EE	7
PF2	10EE	A23F	0	10EE	7
PF3	10EE	A33F	0	10EE	7

PF#	Use Classcode Lookup Assistant	Base Class Menu	Base Class Value	Subclass Interface Menu	Subclass Value	Interface Value	Class Code
PF0	<input checked="" type="checkbox"/>	Memory controller	05	Other memory controller	80	00	0x058000
PF1	<input checked="" type="checkbox"/>	Memory controller	05	RAM	00	00	0x050000
PF2	<input checked="" type="checkbox"/>	Memory controller	05	RAM	00	00	0x050000
PF3	<input checked="" type="checkbox"/>	Memory controller	05	RAM	00	00	0x050000

- **PF0 ID Initial Values:**
 - **Vendor ID:** Identifies the manufacturer of the device or application. Valid identifiers are assigned by the PCI Special Interest Group to guarantee that each identifier is unique. The default value, 10EEh, is the Vendor ID for Xilinx. Enter a vendor identification number here. FFFFh is reserved.
 - **Device ID:** A unique identifier for the application; the default value, which depends on the configuration selected, is A0<link speed><link width>h. This field can be any value; change this value for the application. The default Device ID parameter is based on:
 - The device family: A for Versal ACAP.
 - EP or RP mode.
 - Link width: 1 for x1, 2 for x2, 4 for x4, 8 for x8, and F for x16.
 - Link speed: 1 for Gen1, 2 for Gen2, 3 for Gen3, and 4 for Gen4.

If any of the above values are changed, the Device ID value will be re-evaluated, replacing the previous set value.



RECOMMENDED: *It is always recommended that the link width, speed, and Device Port type be changed first and then the Device ID value. Make sure the Device ID value is set correctly before generating the IP.*

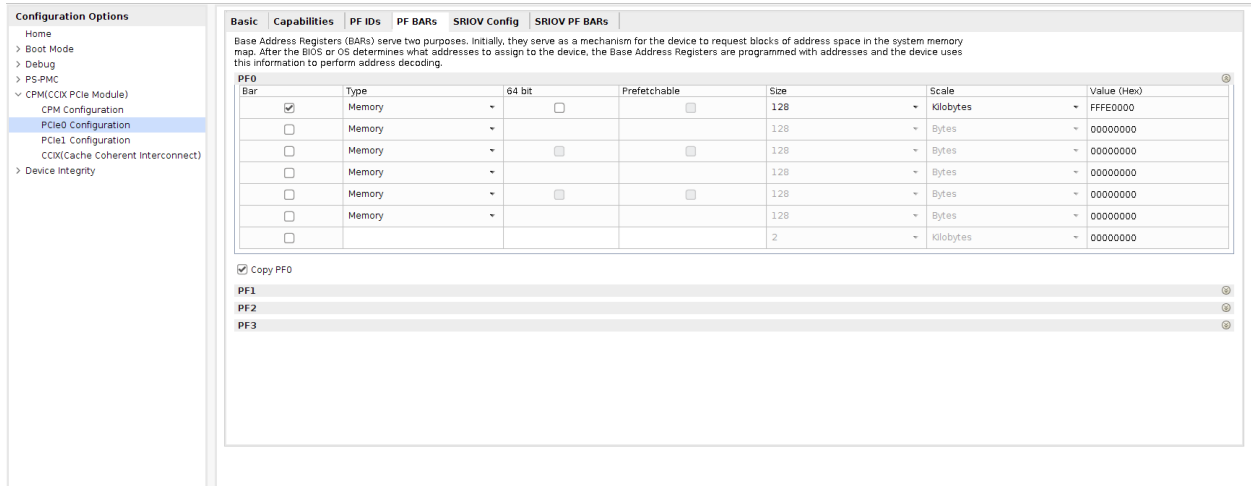
- **Revision ID:** Indicates the revision of the device or application; an extension of the Device ID. The default value is 00h; enter a value appropriate for the application.
- **Subsystem Vendor ID:** Further qualifies the manufacturer of the device or application. Enter a Subsystem Vendor ID here; the default value is 10EEh. Typically, this value is the same as Vendor ID. Setting the value to 0000h can cause compliance testing issues.
- **Subsystem ID:** Further qualifies the manufacturer of the device or application. This value is typically the same as the Device ID; the default value depends on the lane width and link speed selected. Setting the value to 0000h can cause compliance testing issues.
- **Class Code:** The Class Code identifies the general function of a device, and is divided into three byte-size fields:
 - **Base Class:** Broadly identifies the type of function performed by the device.
 - **Sub-Class:** More specifically identifies the device function.
 - **Interface:** Defines a specific register-level programming interface, if any, allowing device-independent software to interface with the device.

Class code encoding can be found at the [PCI-SIG website](#).

- **Class Code Look-up Assistant:** The Class Code Look-up Assistant provides the Base Class, Sub-Class and Interface values for a selected general function of a device. This Look-up Assistant tool only displays the three values for a selected function. You must enter the values in Class Code for these values to be translated into device settings.

PF BARs Tab

The PF BARs tab, shown in the following figure, sets the base address register space for the Endpoint configuration. Each BAR (0 through 5) configures the BAR Aperture Size and Control attributes of the physical function.



- Base Address Register Overview:** In Endpoint configuration, the core supports up to six 32-bit BARs or three 64-bit BARs, and the Expansion read-only memory (ROM) BAR. In Root Port configuration, the core supports up to two 32-bit BARs or one 64-bit BAR, and the Expansion ROM BAR. BARs can be one of two sizes:
 - 32-bit BARs:** The address space can be as small as 128 bytes or as large as 2 gigabytes. Used for Memory or I/O.
 - 64-bit BARs:** The address space can be as small as 128 bytes or as large as 8 Exabytes. Used for Memory only.

All BAR registers share these options:

- Checkbox:** Click the checkbox to enable the BAR; deselect the checkbox to disable the BAR.
- Type:** Bars can either be I/O or Memory.
 - I/O:** I/O BARs can only be 32-bit; the Prefetchable option does not apply to I/O BARs. I/O BARs are only enabled for a Legacy PCI Express Endpoint.
 - Memory:** Memory BARs can be either 64-bit or 32-bit and can be prefetchable. When a BAR is set as 64 bits, it uses the next BAR for the extended address space and makes the next BAR inaccessible.
- Size:** The available Size range depends on the PCIe Device/Port Type and the Type of BAR selected. The following table lists the available BAR size ranges.

Table 1: BAR Size Ranges for Device Configuration

PCIe Device / Port Type	BAR Type	BAR Size Range
PCI Express Endpoint	32-bit Memory	128 bytes (B) – 2 gigabytes (GB)
	64-bit Memory	128 B – 8 Exabytes

Table 1: BAR Size Ranges for Device Configuration (cont'd)

PCIe Device / Port Type	BAR Type	BAR Size Range
Legacy PCI Express Endpoint	32-bit Memory	128 B – 2 GB
	64-bit Memory	128 B – 8 Exabytes
	I/O	16 B – 2 GB

- **Prefetchable:** Identifies the ability of the memory space to be prefetched.
- **Value:** The value assigned to the BAR based on the current selections.
- **Expansion ROM Base Address Register:** If selected, the Expansion ROM is activated and can be sized from 2 KB to 4 GB. According to the PCI Local Bus Specification Revision 3.0 on the [PCI-SIG website](#), the maximum size for the Expansion ROM BAR should be no larger than 16 MB. Selecting an address space larger than 16 MB can cause compliance testing issues.
- **Managing Base Address Register Settings:** Memory, I/O, Type, and Prefetchable settings are handled by setting the appropriate settings for the desired base address register.

Memory or I/O settings indicate whether the address space is defined as memory or I/O. The base address register only responds to commands that access the specified address space. Generally, memory spaces less than 4 KB in size should be avoided. The minimum I/O space allowed is 16 bytes; use of I/O space should be avoided in all new designs.

Prefetchability is the ability of memory space to be prefetched. A memory space is prefetchable if there are no side effects on reads (that is, data is not destroyed by reading, as from a RAM). Byte-write operations can be merged into a single double word write, when applicable.

When configuring the core as an Endpoint for PCIe (non-Legacy), 64-bit addressing must be supported for all BARs (except BAR5) that have the prefetchable bit set. 32-bit addressing is permitted for all BARs that do not have the prefetchable bit set. The prefetchable bit-related requirement does not apply to a Legacy Endpoint. The minimum memory address range supported by a BAR is 128 bytes for a PCI Express Endpoint and 16 bytes for a Legacy PCI Express Endpoint.

- **Disabling Unused Resources:** For best results, disable unused base address registers to conserve system resources. A base address register is disabled by deselecting unused BARs in the Customize IP dialog box.

SRIOV Config Tab

The SRIOV Configuration parameters, as shown in the following figure, are described in this section.

- General SRIOV Config:** This value specifies the offset of the first PF with at least one enabled VF. When ARI is enabled, allowed value is 'd4 or 'd64, and the total number of VF in all PFs plus this field must not be greater than 256. When ARI is disabled, this field will be set to 1 to support 1PF plus 7VF non-ARI SR-IOV configurations only.
- Cap Version:** Indicates the 4-bit SR-IOV Capability version for the physical function.
- Number of PFx VF's:** Indicates the number of virtual functions associated to the physical function. A total of 252 virtual functions are available that can be flexibly used across the four physical functions.
- PFx Dependency Link:** Indicates the SR-IOV Functional Dependency Link for the physical function. The programming model for a device can have vendor-specific dependencies between sets of functions. The Function Dependency Link field is used to describe these dependencies.
- First VF Offset:** Indicates the offset of the first virtual function (VF) for the physical function (PF). PFx offset is always fixed. PF0 resides at offset 0, PF1 resides at offset 1, PF2 resides at offset 2, and PF3 resides at offset 3.

A total of 252 virtual functions are available. They reside at the function number range 4 to 255.

You can select either 4 or 64 for the first VF offset in the customization GUI, if the last VF offset is not more than 255.

- Examples:**

- When the total number of enabled VFs is less than 192, select either 4 or 64 for the first VF offset. The last VF offset will be less than 255.
- When the total number of enabled VFs is more than 192, select 4 for first VF offset. In this case, 64 is not permitted for first VF offset, because the last VF offset will become more the 255.

Virtual functions are mapped sequentially with VFs with PFs taking precedence. For example, if PF0 has two virtual functions and PF1 has three, the following mapping occurs:

The `PFx_FIRST_VF_OFFSET` is calculated by taking the first offset of the virtual function and subtracting that from the offset of the physical function.

```
PFx_FIRST_VF_OFFSET = (PFx first VF offset - PFx offset)
```

In the example above, the following offsets are used:

```
PF0_FIRST_VF_OFFSET = (4 - 0) = 4
PF1_FIRST_VF_OFFSET = (6 - 1) = 5
```

The initial offset for PF1 is a function of how many VFs are attached to PF0 and is defined in the following pseudo code:

```
PF1_FIRST_VF_OFFSET = FIRST_VF_OFFSET + NUM_PF0_VFs - 1
```

Similarly, for other PFs:

```
PF2_FIRST_VF_OFFSET = FIRST_VF_OFFSET + NUM_PF0_VFs + NUM_PF1_VFs - 2
PF3_FIRST_VF_OFFSET =
    FIRST_VF_OFFSET + NUM_PF0_VFs + NUM_PF1_VFs + NUM_PF2_VFs - 3
```

- **VF Device ID:** Indicates the 16-bit Device ID for all virtual functions associated with the physical function.
- **SRIOV Supported Page Size:** Indicates the page size supported by the physical function. This physical function supports a page size of $2^{(n+12)}$, if bit n of the 32-bit register is set.

SRIOV BARs Tab

The SRIOV Base Address Registers (BARs) set the base address register space for the Endpoint configuration. Each BAR (0 through 5) configures the SR-IOV BAR aperture size and SR-IOV control attributes.

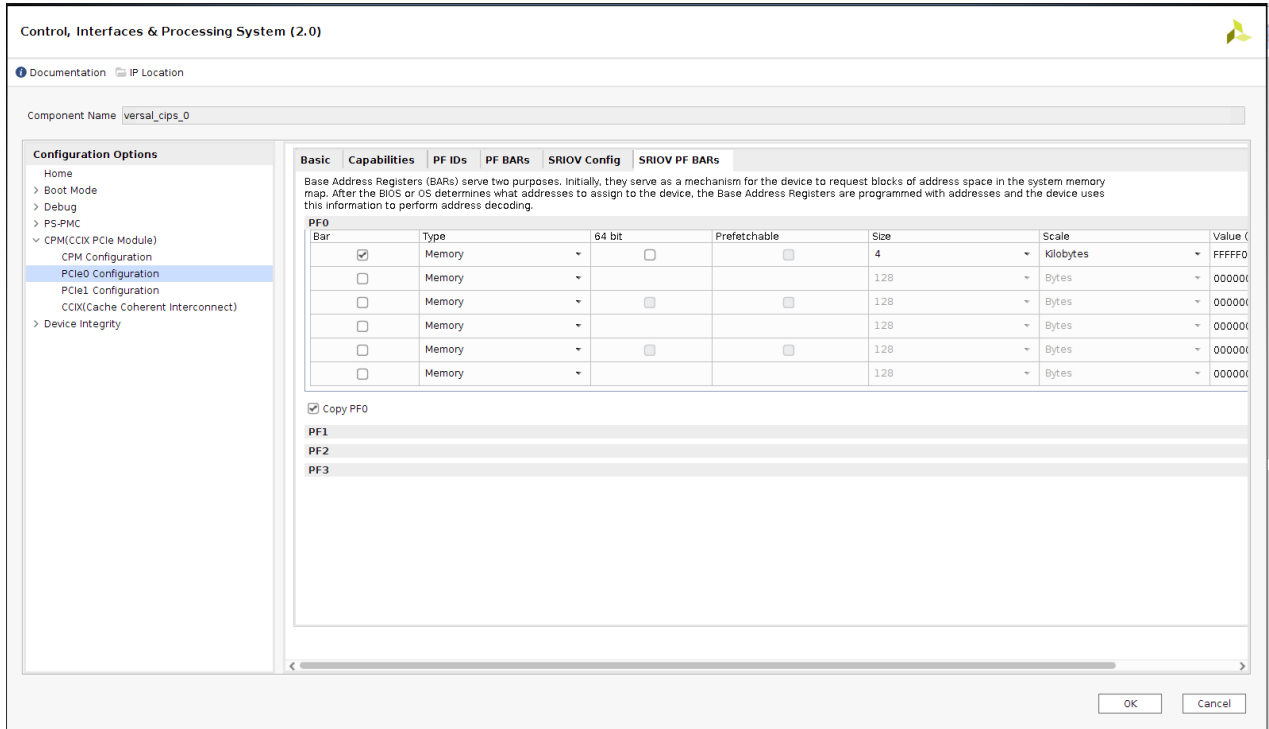


Table 2: Example Virtual Function Mappings

Physical Function	Virtual Function	Function Number Range
PF0	VF0	64
PF0	VF1	65
PF1	VF0	66
PF1	VF1	67
PF1	VF1	68

- SRIOV Base Address Register Overview:** In Endpoint configuration, the core supports up to six 32-bit BARs or three 64-bit BARs. In Root Port configuration, the core supports up to two 32-bit BARs or one 64-bit BAR. SR-IOV BARs can be one of two sizes:
 - 32-bit BARs:** The address space can be as small as 16 bytes or as large as 3 gigabytes. Used for memory to I/O.
 - 64-bit BARs:** The address space can be as small as 128 bytes or as large as 256 gigabytes. Used for memory only.

All SR-IOV BAR registers have these options:

- Checkbox:** Click the checkbox to enable the BAR; deselect the checkbox to disable the BAR.
- Type:** SR-IOV BARs can be either I/O or Memory.

- **I/O:** I/O BARs can only be 32-bit; the Prefetchable option does not apply to I/O BARs. I/O BARs are only enabled for a Legacy PCI Express Endpoint.
- **Memory:** Memory BARs can be either 64-bit or 32-bit and can be prefetchable. When a BAR is set to 64-bits, it uses the next BAR for the extended address space and makes the next BAR inaccessible.
- **Size:** The available size range depends on the PCIe device/port type and the type of BAR selected. The following table lists the available BAR size ranges.

Table 3: SRIOV BAR Size Ranges for Device Configuration

PCIe Device / Port Type	BAR Type	BAR Size Range
PCI Express Endpoint	32-bit Memory	128 bytes – 2 gigabytes
	64-bit Memory	128 bytes – 8 exabytes
Legacy PCI Express Endpoint	32-bit Memory	16 bytes – 2 gigabytes
	64-bit Memory	16 bytes – 8 exabytes
	I/O	16 bytes – 2 gigabytes
Root Port Mode	32-bit Memory	16 bytes – 2 gigabytes
	64-bit Memory	4 bytes – 8 exabytes
	I/O	16 bytes – 2 gigabytes

- **Prefetchable:** Identifies the ability of the memory space to be prefetched.
- **Value:** The value assigned to the BAR based on the current selections.
- **Managing SRIOV Base Address Register Settings:** Memory, I/O, Type, and Prefetchable settings are handled by setting the appropriate Customize IP dialog box settings for the desired base address register.

Memory or I/O settings indicate whether the address space is defined as memory or I/O. The base address register only responds to commands that access the specified address space. Generally, memory spaces less than 4 KB in size should be avoided. The minimum I/O space allowed is 16 bytes. I/O space should be avoided in all new designs.

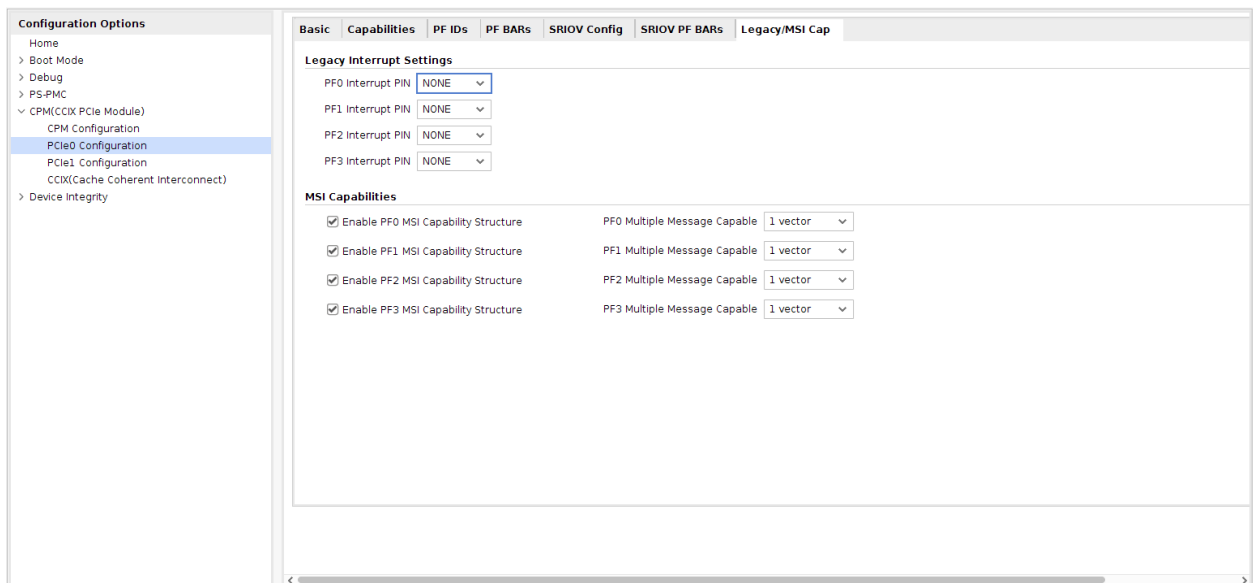
A memory space is prefetchable if there are no side effects on reads (that is, data is not destroyed by reading, as from RAM). Byte-write operations can be merged into a single double-word write, when applicable.

When configuring the core as an Endpoint for PCIe (non-Legacy), 64-bit addressing must be supported for all SR-IOV BARs (except BAR5) that have the prefetchable bit set. 32-bit addressing is permitted for all SR-IOV BARs that do not have the prefetchable bit set. The prefetchable bit related requirement does not apply to a Legacy Endpoint. The minimum memory address range supported by a BAR is 128 bytes for a PCI Express Endpoint and 16 bytes for a Legacy PCI Express Endpoint.

- **Disabling Unused Resources:** For best results, disable unused base address registers to conserve system resources. Disable base address register by deselecting unused BARs in the Customize IP dialog box.

Legacy/MSI Cap Tab

On this page, you set the Legacy Interrupt Settings and MSI Capabilities for all applicable physical and virtual functions. This page is not visible when the **SRIOV Capability** parameter is selected on the Capabilities page.



- **Legacy Interrupt Settings:**
 - **PF0/PF1/PF2/PF3 Interrupt PIN:** Indicates the mapping for Legacy Interrupt messages. A setting of None indicates that no Legacy Interrupts are used.

- **MSI Capabilities:**

- **PF0/PF1/PF2/PF3 Enable MSI Capability Structure:** Indicates that the MSI Capability structure exists.

Note: Although it is possible to not enable MSI or MSI-X, the result would be a non-compliant core. The PCI Express Base Specification requires that MSI, MSI-X, or both be enabled. No MSI capabilities are supported when **MSI-X Internal** is enabled in the [MSI-X Capabilities Tab](#) (Advanced mode), because MSI-X Internal uses some of the MSI interface signals.

- **PF0/PF1/PF2/PF3 Multiple Message Capable:** Selects the number of MSI vectors to request from the Root Complex.
- **Enable MSI Per Vector Masking:** Enables MSI Per Vector Masking Capability of all the Physical functions enabled.

Note: Enabling this option for individual physical functions is not supported.

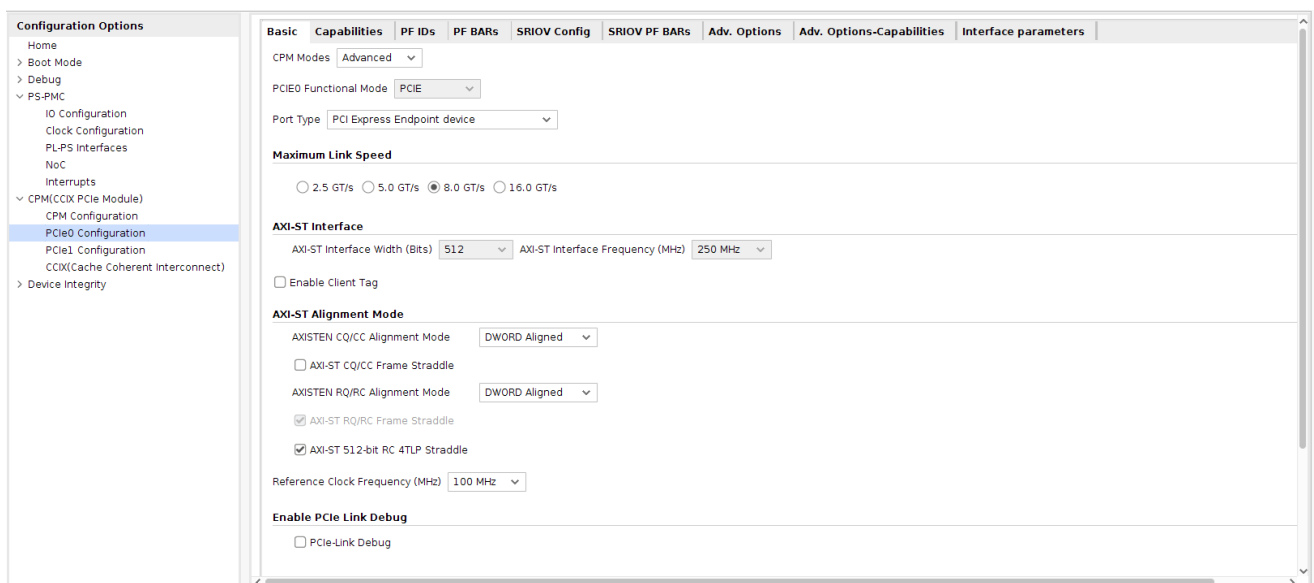
Advanced Mode Parameters

The following parameters appear on different pages of the IP catalog when **Advanced** mode is selected for Mode on the Basic page.

Basic Tab

The Basic page with Advanced mode selected (shown in the following figure) includes additional settings. The following parameters are visible on the Basic page when the **Advanced** mode is selected.

Figure 10: Basic Tab, Advanced Mode

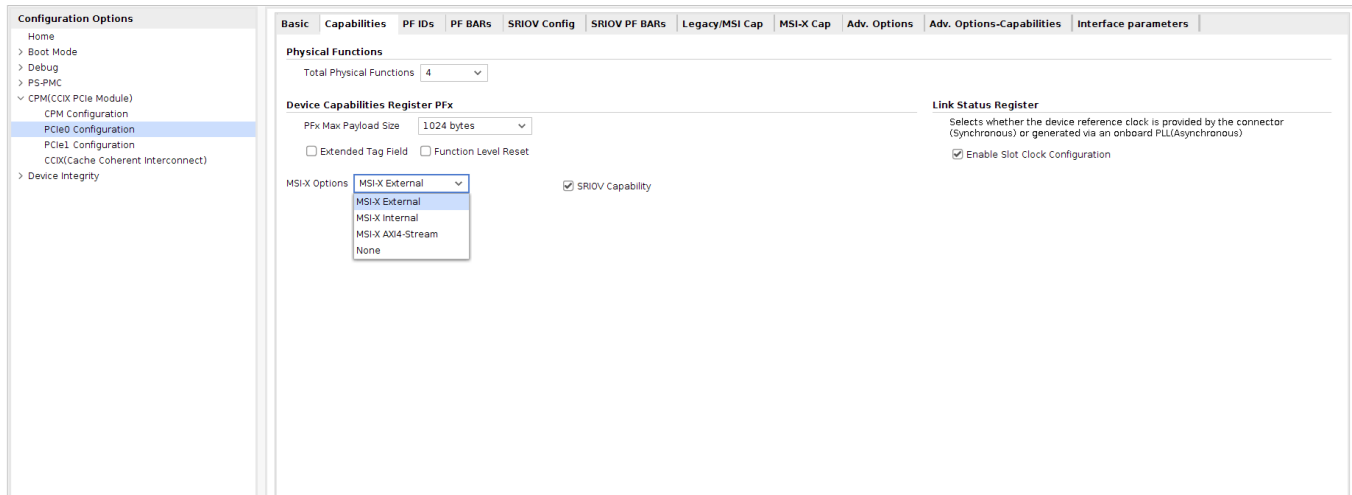


- **PCIe Link Debug:** This enables the link debug option to be activated.

Capabilities Tab

The Capabilities settings for Advanced mode (as shown in the following figure) contains two additional parameters to those for Basic mode and are described below.

Figure 11: Capabilities Tab, Advanced Mode



- MSI-X Options:** To enable MSI-X capabilities, select **Advanced** mode and then select the required options on the Capabilities tab. There are four options to choose from:
 - MSI-X External:** In this mode you need to implement MSI-X External interface driving logic, MSI-X Table and PBA buffers outside the PCIe core. You can configure the MSI-X BARs.
 - MSI-X Internal:** In this mode you need to implement the MSI-X Internal interface driving logic only. MSI-X Table and PBA buffers are built into the PCIe core. You can configure the MSI-X BARs.
 - MSI-X AXI4-Stream:** In this mode user is expected to drive MSI-X interrupts on the AXI4-Stream interface. You can configure the MSI-X BARs.
 - None:** No MSI-X is supported.

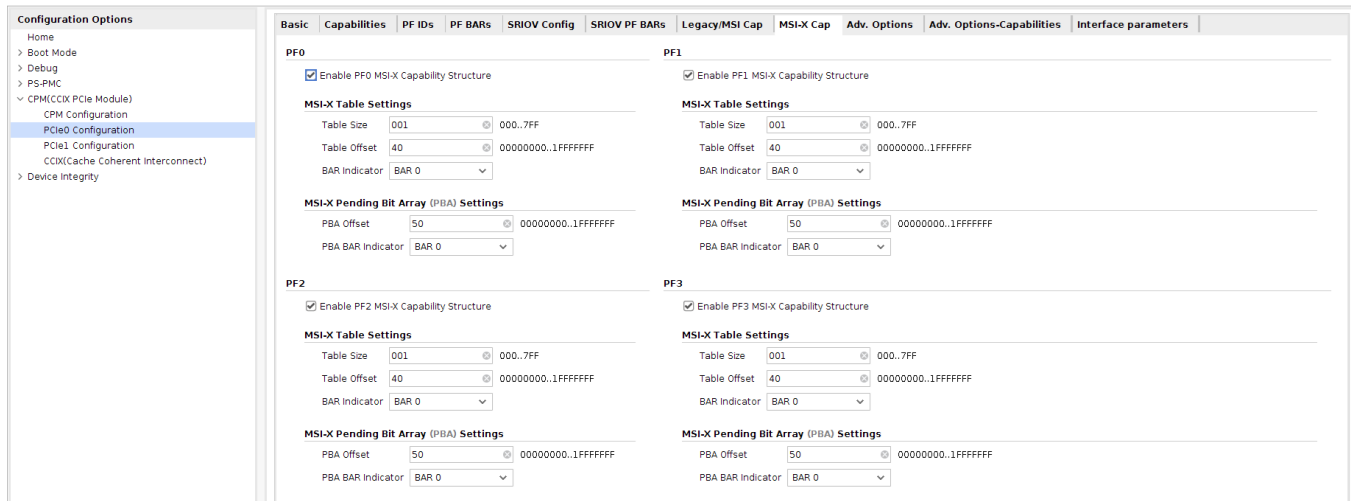
The same MSI-X options are applicable when SRIOV capability is selected.

MSI-X Capabilities Tab

The MSI-X Capabilities parameters, shown in the following figure, are available in Advanced mode only. To enable MSI-X capabilities, select **Advanced** mode and then select the required options on the Capabilities page.

The same MSI-X options are applicable when SRIOV capability is selected.

Figure 12: MSI-X Cap Tab, Advanced Mode



- **Enable MSI-X Capability Structure:** Indicates that the MSI-X Capabilities structure exists.

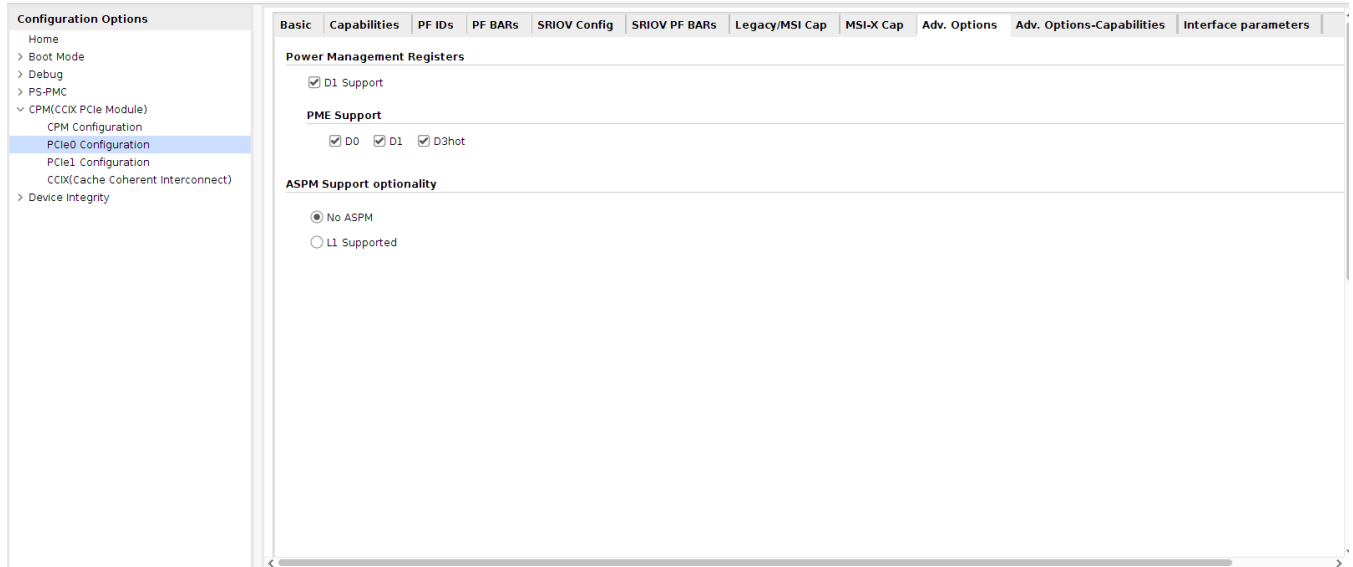
Note: The Capability Structure needs at least one Memory BAR to be configured. You must maintain the MSI-X Table and Pending Bit Array in the application.

- **MSI-X Table Settings:** Defines the MSI-X Table structure.
 - **Table Size:** Specifies the MSI-X Table size. Table Size field is expecting N-1 interrupts (0x0F will configure a table count of 16).
 - **Table Offset:** Specifies the offset from the Base Address Register that points to the base of the MSI-X Table.
 - **BAR Indicator:** Indicates the Base Address Register in the Configuration Space used to map the function in the MSI-X Table onto memory space. For a 64-bit Base Address Register, this indicates the lower DWORD.
- **MSI-X Pending Bit Array (PBA) Settings:** Defines the MSI-X Pending Bit Array (PBA) structure.
 - **PBA Offset:** Specifies the offset from the Base Address Register that points to the base of the MSI-X PBA.
 - **PBA BAR Indicator:** Indicates the Base Address Register in the Configuration Space used to map the function in the MSI-X PBA onto Memory Space.

Adv. Options-1

The Advanced Options tab enables Power Management Registers and ASPM. There is no L0s support when the link speed is not 2.5 Gb/s and 5.0 Gb/s.

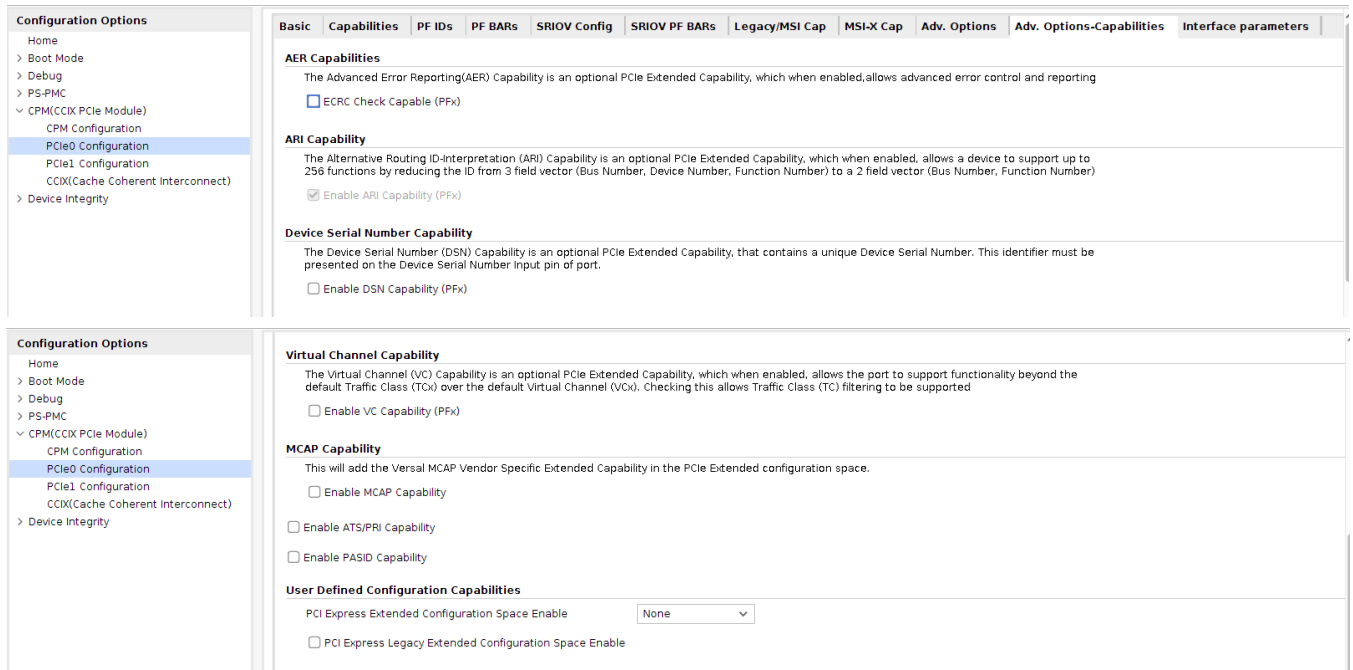
Figure 13: Adv. Options Tab



Adv. Options Capabilities

The Advanced Options tab enables you to choose ECRC check in AER Capability, ARI, DSN, Virtual Channel, MCAP, ATS/PRI, PASID, and PCI Express Extended Config space and PCI Legacy Extended space.

Figure 14: Adv. Options Capabilities Tab



PCI Express Extended Configuration Space has three options:

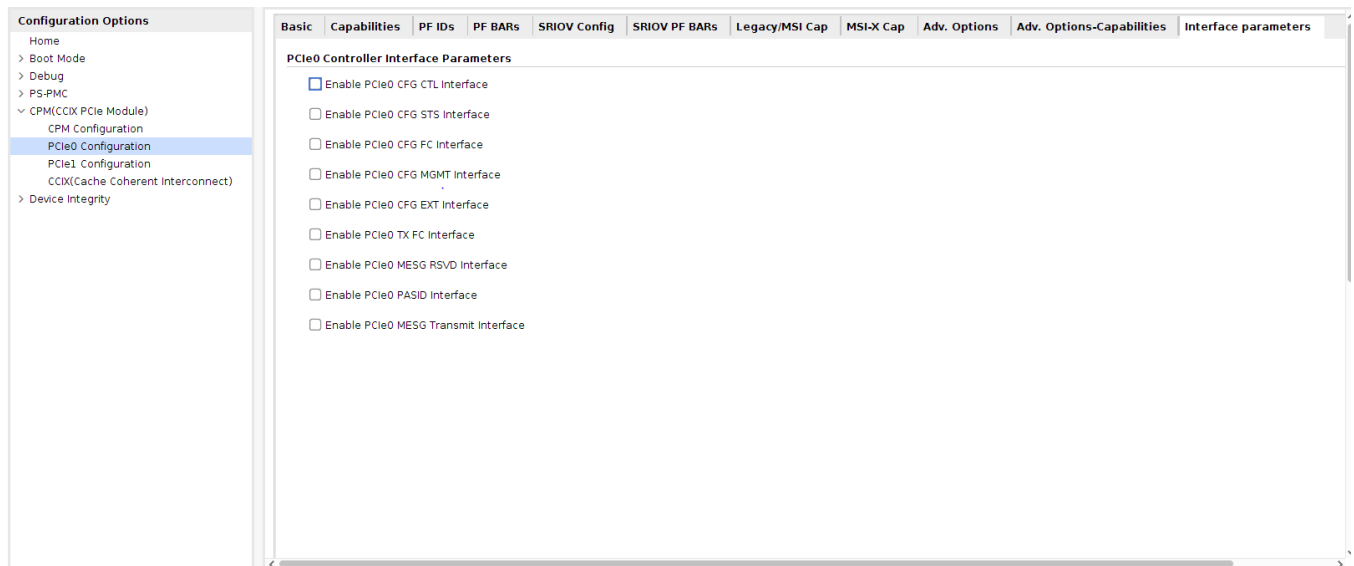
- **None:** No Extended Configuration space required.
- **Extended Small:** PCI Express Extended space available from 0XE00 – 0XFFF.
- **Extended Large:** PCI Express Extended space available from 0X600 – 0XFFF.

Note: TPH Capability is not supported.

Interface Parameters

The Interface Parameters tab enables you to enable/disable interfaces that are not required for your application.

Figure 15: Interface Parameters Tab



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 both CPM and PL PCIe based solutions. For this core, the CPM related guidance is of primary importance, while the PL PCIe related guidance might be relevant and is provided for informational purposes.

A GT Quad is comprised of four GT lanes. When selecting GT Quads for the CPM PCIe controller 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 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 4: 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 PL pin

The following subsections provide more detail about GT selection for Versal devices.

CPM4 GT Selection

The CPM block within Versal devices has a fixed set of GTs that can be used for each of the two PCIe controllers. These GTs are shared between the two PCIe controllers and High Speed Debug Port (HSDP) as such x16 link widths are only supported when a single PCIe controller is in use and HSDP is disabled. When two CPM PCIe controllers or one PCIe controller and HSDP are enabled each link will be limited to a x8 link width. GT Quad allocation for CPM happens at GT Quad granularity and must include all GT Quads from the most adjacent to the CPM to the top-most GT Quad that is in use by the CPM. GT Quads that are used or between GT Quads that are used by the CPM (for either PCIe or HSDP) cannot be share with PL resources even if GTs within the quad are not in use.

CPM in Single Controller Mode

When a single PCIe controller in the CPM is being used and HSDP is disabled, PCIe x1, x2, x4, x8, and x16 link widths are supported. PCIe lane0 is places at the bottom-most GT of the bottom-most GT Quad that is directly above the CPM. Subsequent lanes use the next available GTs moving vertically up the device as the lane number increments. This means the highest PCIe lane number uses the top-most GT in the top-most GT Quad that is used for PCIe. Because the GT locations and lane ordering for CPM is fixed it cannot be modified through IP customization.

As stated previously GT Quad allocation happens at GT Quad granularity and cannot share unused GT Quad resources with the PL. This means that CPM PCIe controller 0 configurations that use x1 or x2 link widths will not use all the GTs within the Quad and that these GTs cannot be used in the PL for additional GT connectivity. Unused GT Quads in this configuration can be used by the PL to implement PL GT based solutions.

When CPM PCIe controller 0 and High Speed Debug Port (HSDP) is enabled, a PCIe link width of x16 cannot be used and the CPM will use all three GT Quads that are directly above the CPM regardless of PCIe link width. In this configuration, these GT Quads are allocated to CPM and cannot be shared with PL resources. CPM PCIe lanes 0-7 will be unchanged in their GT selection and lane ordering. HSDP will use the bottom-most GT that is the third GT Quad away from CPM. This corresponds to the same location as PCIe lane 8 for a x16 link configuration. The fourth GT Quad in this configuration is not used by CPM and can be used to implement PL GT based solutions.

CPM in Dual Controller Mode

When the CPM is configured to use two PCIe controllers, High Speed Debug Port (HSDP) cannot be used because it shares GTs with the two PCIe controllers. Each PCIe controller can support x1, x2, x4 and x8 link widths in this configuration. This configuration will use at least the bottom three GT Quads closest to the CPM. These GT Quads cannot be used by PL resources. If CPM PCIe controller 1 is using a link width of x1, x2, or x4; then CPM uses three GT Quads. In this case the fourth GT Quad can be used by PL resources to implement GT based solutions. If CPM PCIe controller 1 is using a x8 link width, all four GT Quads will be used by the CPM and cannot be used by PL resources.

CPM PCIe controller 0 lane0 is placed at the bottom-most GT of the bottom-most GT Quad that is directly above the CPM. Subsequent lanes use the next available GTs moving vertically up the device as the lane number increments. CPM PCIe controller 0 lane7 connects to the top-most GT in the second GT Quad away from the CPM.

CPM PCIe controller 1 lane0 is placed at the bottom-most GT of third GT Quad above the CPM. Subsequent lanes use the next available GTs moving vertically up the device as the lane number increments. CPM PCIe controller 1 lane7 connects to the top-most GT in the fourth GT Quad away from the CPM.

High Speed Debug Port (HSDP) Only Modes

When the CPM is configured to use the High Speed Debug Port (HSDP) without enabling either PCIe controller, the bottom-most GT in the bottom-most GT Quad closest to CPM should be used. This will allow the CPM to use only one GT Quad and allow the next three GT Quads to be used by PL resources.

HSDP can also be enabled for the bottom-most GT in the third GT Quad up from CPM. In this scenario CPM will use three GT Quads and only use one GT. The remaining unused GTs cannot be used or shared by PL resources. As a result typically HSDP will not be used in this configuration.

CPM4 Additional Considerations

To facilitate migration from UltraScale™ or UltraScale+™ designs, boards may be designed to use either CPM4 or PL PCIe integrated blocks to implement PCIe solutions. When designing a board to use either CPM4 or the PL PCIe hardblock, the CPM4 pin selection and planning guidelines should be followed because they are more restrictive. By doing this a board can be designed that will work for either a CPM4 or PL PCIe implementation. To route the PCIe reset from the CPM4 to the PL for use with a PL PCIe implementation the following parameter must be set in Vivado prior to customizing the CIPS IP.

```
set_param pcw.enlpciereset 1
```

When this parameter is enabled the PCIe reset for each disabled CPM4 PCIe controller will be routed to the PL. The same CPM4 pin selection limitations will apply and the additional PCIe reset output pins will be exposed at the boundary of the CIPS IP. If the CPM4 PCIe controller is enabled, the PCIe reset will be used internal to the CPM4 and will not be routed to the PL for connectivity to PL PCIe controllers.

GT Locations

The following tables identify 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. GT selections and assignment can be modified through the GT customization IP and user constraints.

Table 5: GT Locations

Device	Package	PCIe Blocks	GT QUAD (X16)	GT QUAD (X8)	GT QUAD (X4 and Below)
XCVC1902	VIVA1596	CPM Controller 0	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y3
		CPM Controller 1	N/A	GT_Y_QUAD_X0Y5	GT_Y_QUAD_X0Y5
		X0Y2	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y6 GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3
		X0Y1	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y6 GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3
		X1Y2	GT_Y_QUAD_X1Y5	GT_Y_QUAD_X1Y5 GT_Y_QUAD_X1Y4 GT_Y_QUAD_X1Y3	GT_Y_QUAD_X1Y6 GT_Y_QUAD_X1Y5 GT_Y_QUAD_X1Y4 GT_Y_QUAD_X1Y3 GT_Y_QUAD_X1Y2
		X1Y0	GT_Y_QUAD_X1Y5	GT_Y_QUAD_X1Y5 GT_Y_QUAD_X1Y4 GT_Y_QUAD_X1Y3	GT_Y_QUAD_X1Y6 GT_Y_QUAD_X1Y5 GT_Y_QUAD_X1Y4 GT_Y_QUAD_X1Y3 GT_Y_QUAD_X1Y2
XCVC1902	VSVA2197	CPM Controller 0	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y3
		CPM Controller 1	N/A	GT_Y_QUAD_X0Y5	GT_Y_QUAD_X0Y5
		X0Y2	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y6 GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3
		X0Y1	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y6 GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3
		X1Y2	GT_Y_QUAD_X1Y6	GT_Y_QUAD_X1Y6 GT_Y_QUAD_X1Y5 GT_Y_QUAD_X1Y4	GT_Y_QUAD_X1Y6 GT_Y_QUAD_X1Y5 GT_Y_QUAD_X1Y4 GT_Y_QUAD_X1Y3
		X1Y0	GT_Y_QUAD_X1Y3	GT_Y_QUAD_X1Y3 GT_Y_QUAD_X1Y2 GT_Y_QUAD_X1Y1	GT_Y_QUAD_X1Y3 GT_Y_QUAD_X1Y2 GT_Y_QUAD_X1Y1 GT_Y_QUAD_X1Y0

Table 5: GT Locations (cont'd)

Device	Package	PCIe Blocks	GT QUAD (X16)	GT QUAD (X8)	GT QUAD (X4 and Below)
XCVC1902	VSVD1760	CPM Controller 0	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y3
		CPM Controller 1	N/A	GT_Y_QUAD_X0Y5	GT_Y_QUAD_X0Y5
		X0Y2	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y6 GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3
		X0Y1	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y6 GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3
		X1Y2	N/A	GT_Y_QUAD_X1Y4	GT_Y_QUAD_X1Y4 GT_Y_QUAD_X1Y3
		X1Y0	N/A	GT_Y_QUAD_X1Y4	GT_Y_QUAD_X1Y4 GT_Y_QUAD_X1Y3
XCVM1802	VFVC1760	CPM Controller 0	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y3
		CPM Controller 1	N/A	GT_Y_QUAD_X0Y5	GT_Y_QUAD_X0Y5
		X0Y2	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y6 GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3
		X0Y1	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y6 GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3
		X1Y2	GT_Y_QUAD_X1Y6	GT_Y_QUAD_X1Y6 GT_Y_QUAD_X1Y5 GT_Y_QUAD_X1Y4	GT_Y_QUAD_X1Y6 GT_Y_QUAD_X1Y5 GT_Y_QUAD_X1Y4 GT_Y_QUAD_X1Y3
		X1Y0	GT_Y_QUAD_X1Y3	GT_Y_QUAD_X1Y3 GT_Y_QUAD_X1Y2 GT_Y_QUAD_X1Y1	GT_Y_QUAD_X1Y3 GT_Y_QUAD_X1Y2 GT_Y_QUAD_X1Y1 GT_Y_QUAD_X1Y0

Table 5: GT Locations (cont'd)

Device	Package	PCIe Blocks	GT QUAD (X16)	GT QUAD (X8)	GT QUAD (X4 and Below)
XCVM1802	VSVA2197	CPM Controller 0	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y3
		CPM Controller 1	N/A	GT_Y_QUAD_X0Y5	GT_Y_QUAD_X0Y5
		X0Y2	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y6 GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3
		X0Y1	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y6 GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3
		X1Y2	GT_Y_QUAD_X1Y6	GT_Y_QUAD_X1Y6 GT_Y_QUAD_X1Y5 GT_Y_QUAD_X1Y4	GT_Y_QUAD_X1Y6 GT_Y_QUAD_X1Y5 GT_Y_QUAD_X1Y4 GT_Y_QUAD_X1Y3
		X1Y0	GT_Y_QUAD_X1Y3	GT_Y_QUAD_X1Y3 GT_Y_QUAD_X1Y2 GT_Y_QUAD_X1Y1	GT_Y_QUAD_X1Y3 GT_Y_QUAD_X1Y2 GT_Y_QUAD_X1Y1 GT_Y_QUAD_X1Y0
XCVM1802	VSVD1760	CPM Controller 0	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y3
		CPM Controller 1	N/A	GT_Y_QUAD_X0Y5	GT_Y_QUAD_X0Y5
		X0Y2	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y6 GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3
		X0Y1	GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3	GT_Y_QUAD_X0Y6 GT_Y_QUAD_X0Y5 GT_Y_QUAD_X0Y4 GT_Y_QUAD_X0Y3
		X1Y2	N/A	GT_Y_QUAD_X1Y4	GT_Y_QUAD_X1Y4 GT_Y_QUAD_X1Y3
		X1Y0	N/A	GT_Y_QUAD_X1Y4	GT_Y_QUAD_X1Y4 GT_Y_QUAD_X1Y3

PCIe Link Debug Enablement

The Versal™ ACAP CPM Mode for PCI Express customization provides an option to enable PCIe® Link Debug. Enabling this option will insert a debug core inside the IP core that will be recognized by the Vivado® Hardware Manager and provide PCIe specific debug information and view. The debug view provides information relating to the current link speed, current link width, and LTSSM state transitions, which can facilitate debug of PCIe link related issues.

Note: This appendix provides guidance for both CPM and PL PCIe based solutions. For this core, the CPM related guidance is of primary importance, while the PL PCIe related guidance might be relevant and is provided for informational purposes.

Enabling PCIe Link Debug

Use this guide to enable and connect PCIe Link Debug in a Vivado IP Integrator design. This section only describes the additional connections that should be added to enable PCIe Link Debug in a design. It does not discuss how to properly connect the PCIe enabled IPs to create a working design. The described connectivity and connections should be added to an existing design and IP configuration.

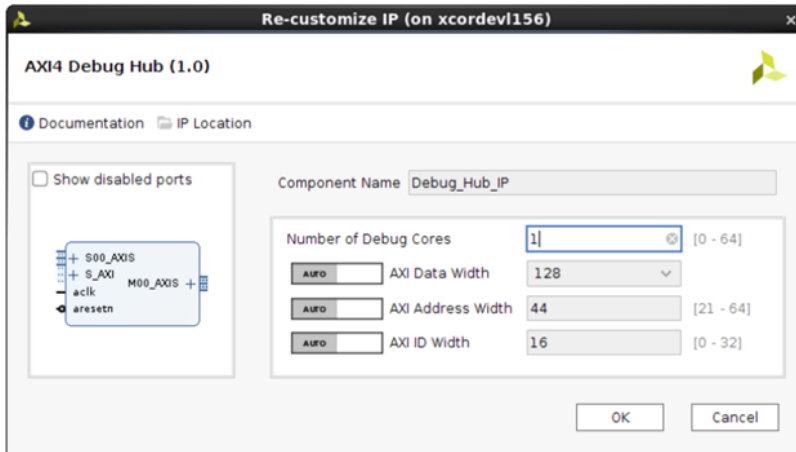
1. Enable this option in the core customization wizard, and select the options in the customization GUI, as shown below. The CPM PCIe cores are customized through the CIPS IP and for PL PCIe cores are customized through the Versal ACAP Integrated Block for PCIe IP.

PCIe-Link Debug

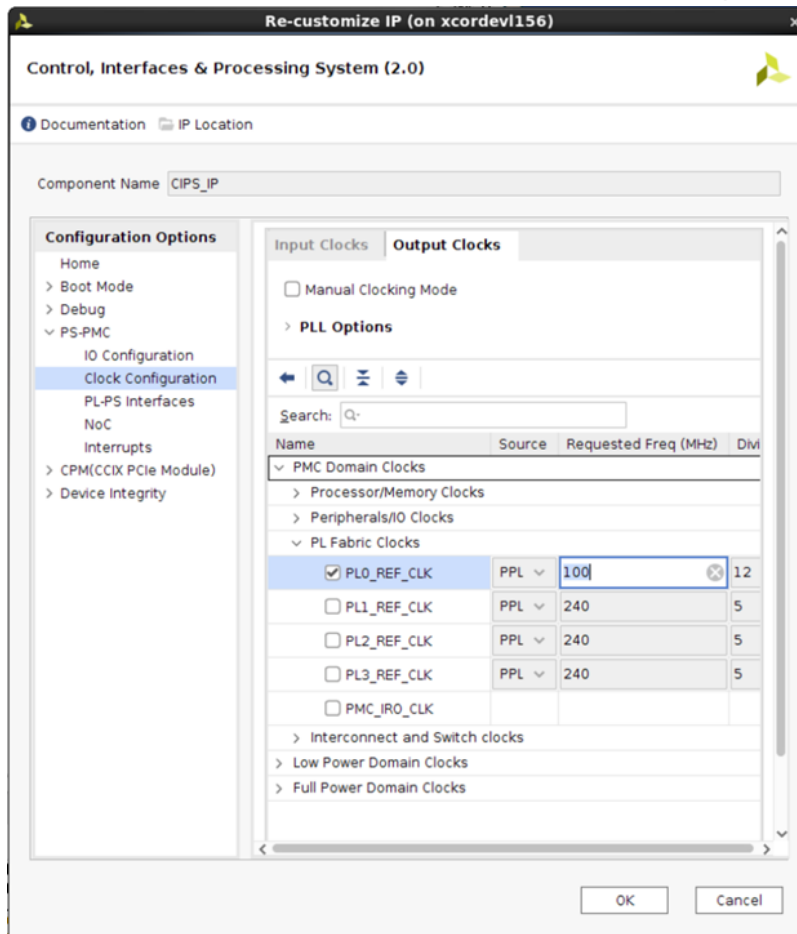
Enable Debug AXI4 Stream Interfaces

This adds the PCIe debug core to the PCIe IP and exposes the debug AXI Stream interfaces and ports. The debug AXI Stream and interface ports should be connected to a Debug Hub IP within the Versal design to enable debug for the design. The PCIe example design provides one implementation of how the Debug Hub IP can be connected in Versal designs. This is also detailed in the description below.

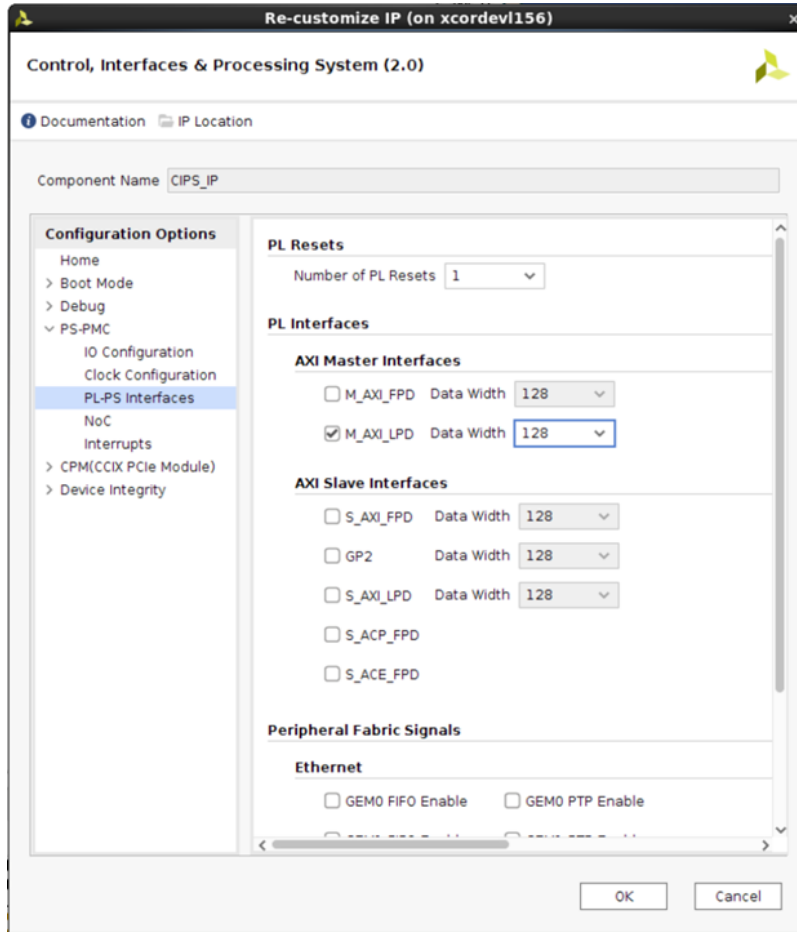
2. Add the Debug Hub IP to the design and use the following configuration options to enable the Debug Hub AXI Memory Mapped interface along with one set of AXI stream interfaces. Additional AXI Stream interfaces can be enabled and connected in your design as desired.



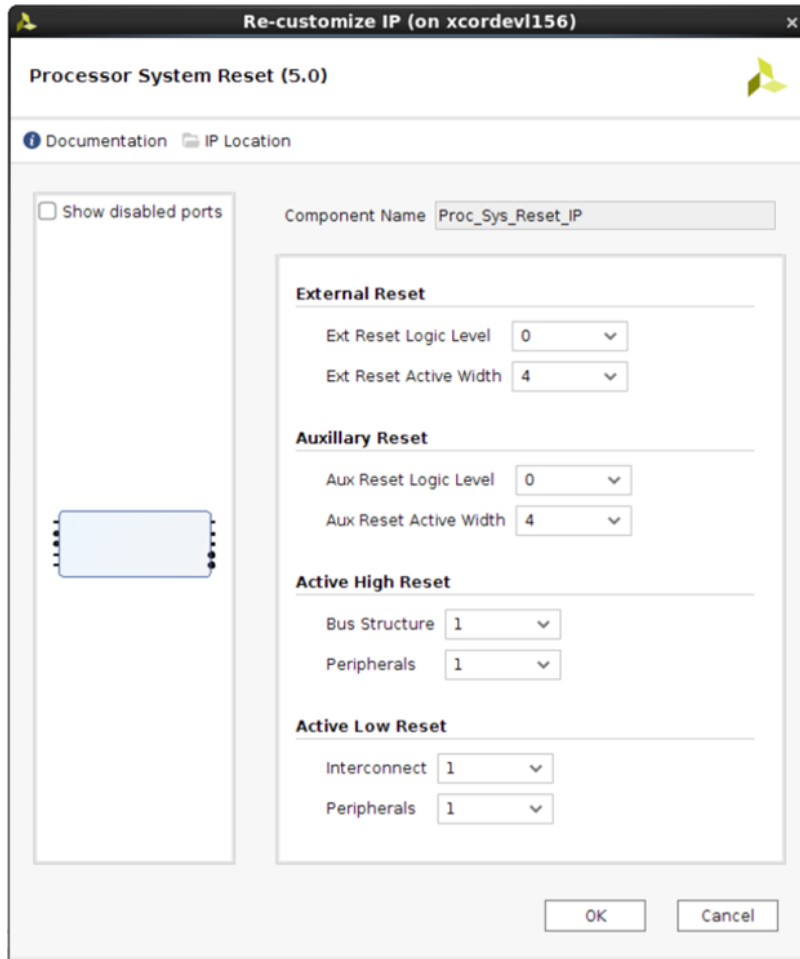
3. Add the CIPS IP to the design or configure the existing CIPS IP and include the following configuration options. These options will enable an AXI Master, clock, and reset that can be connected to the Debug Hub IP. To do so:
 - a. Select **PS-PMC** → **Clock Configuration** → **Output Clocks** → **PMC Domain Clocks** → **PL Fabric Clocks** selection enable a 100 MHz or similar output clock.



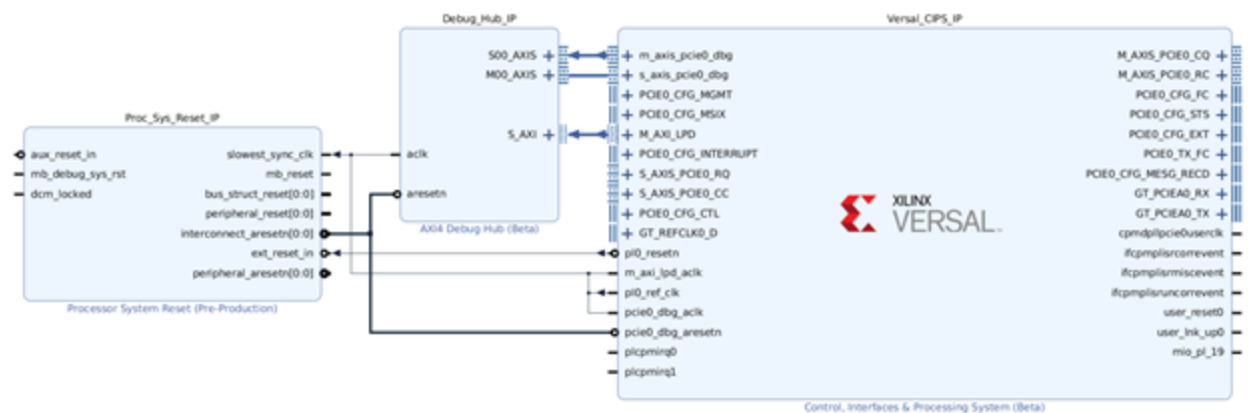
- b. Select **PS-PMC** → **PL-PS Interfaces**, and enable at least one PL reset in **Number of PL Resets**, and the **M_AXI_LPD** AXI master.



- 4. Add and configure the Processor System Reset IP.



5. Connect the IPs as shown in the following figures. This may need to be customized to fit with existing design connectivity.



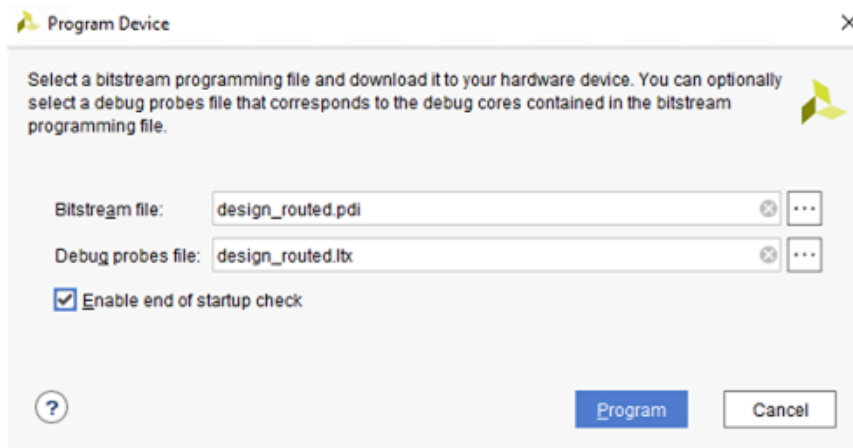
After the debug connections have been added to an Vivado IP integrator design, as shown above, PCIe Link debug is enabled in the generated `.pdi` image. The connections shown above should be added to a full design and are not sufficient to create a working design alone. The PCIe IP ports and the remainder of the design must be created and configured as per the desired operation of the PCIe-enabled IP.

Connecting to PCIe Link Debug in Vivado

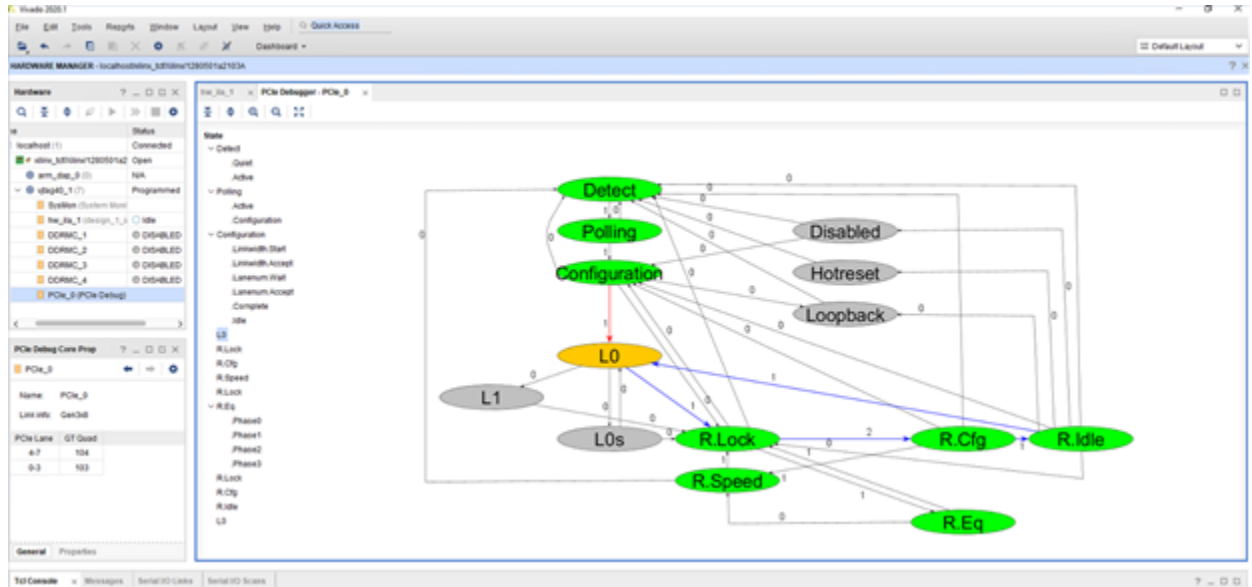
Use the following steps to connect Vivado Hardware Manager to the FPGA device and associated PCIe Link Debug enabled design.

1. Open the Hardware Manager.
2. Select the device from the **Tools** → **Program Device...** drop-down menu.
3. Select the `.pdi` and `.ltx` files for programming the device, and select **Program**.

Note: You should not load the `.ltx` file and refresh the target until after the `.pdi` file has been programmed.



4. Select the PCIe Debug core in the Hardware window. You will see three main views that include the PCIe Debug Core Properties, PCIe Link LTSSM State Trace, and the PCIe Link LTSSM State Diagram with transitions.



Using this view, you can observe the active PCIe link status and state transitions. In the PCIe Debug Core Properties window, you can see the name of the PCIe debug core (PCIe_0), the current link status (Gen3x8), and the connected GTs (Quads 103 and 104). The PCIe LTSSM State Trace view shows a hierarchical view of the PCIe LTSSM state machine transitions. The PCIe LTSSM State Diagram provides a graphical display of the PCIe LTSSM states transitions that were traversed during the PCIe link up process. Visited LTSSM states are shown in green, the final or current LTSSM state is shown in yellow and the number of times each transition was traversed is identified on the arcs between states.

In addition to the graphical display, the `report_hw_pcie` command can be used to generate a console text report that contains the PCIe debug information. This information can be shared with others to aid in debugging PCIe Link issues. For this example, the name of the debug core is PCIe_0, and is inserted into the command.

```
report_hw_pcie PCIe_0
```

This information helps determine where in the PCIe link-up process an issue occurred and can guide further debug of link related issues.

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 Versal™ ACAP CPM Mode for PCIe, 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 Versal™ ACAP CPM Mode for PCIe. 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.

Solution Centers

See the [Xilinx Solution Centers](#) for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

The Solution Center specific to the Versal™ ACAP CPM Mode for PCIe Versal™ ACAP CPM Mode for PCIe is listed below.

- [Xilinx Solution Center for PCI Express](#)

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 Versal™ ACAP CPM Mode for PCIe 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

AR [75350](#).

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](#).

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see [Xilinx Support](#).

Documentation Navigator and Design Hubs

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

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

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

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the [Design Hubs](#) page.

Note: For more information on DocNav, see the [Documentation Navigator](#) page on the Xilinx website.

References

These documents provide supplemental material useful with this guide:

1. *Versal ACAP CPM DMA and Bridge Mode for PCI Express Product Guide* (PG347)
2. *Versal ACAP Integrated Block for PCI Express LogiCORE IP Product Guide* (PG343)
3. *Versal ACAP Technical Reference Manual* (AM011)
4. *Versal ACAP CPM CCIX Architecture Manual* (AM016)
5. *PCI Express Base Specification 4.0* (<http://www.pcisig.com/specifications>)
6. *PCI Express Base Specification 5.0* (<http://www.pcisig.com/specifications>)
7. *Cache Coherent Interconnect for Accelerators (CCIX) Transport Specification* (available at <http://www.ccixconsortium.com/>; membership required).
8. *Control Interface and Processing System IP Product Guide* (PG352)
9. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
10. *Vivado Design Suite User Guide: Designing with IP* (UG896)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
07/24/2020 Version 1.0	
Initial Xilinx release.	N/A

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the

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

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

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