Intel® FPGA IP Subsystem for PCI Express* IP User Guide

Updated for Intel® Quartus® Prime Design Suite: **23.1**

Version: **1.0**

Table of Contents

1. Introduction

Intel® FPGA IP Subsystem for PCI Express* allows users to implement PCI Express in their design using Intel's technology leading PCIe hardened protocol stack where the physical, data link, and transaction layers are hardened blocks within the device. The IP includes transaction, data link and physical layers, and includes optional blocks, such as data movers and single root I/O virtualization (SR-IOV) for applications requiring high bandwidth data transfer to/from host memory and virtualization. This document introduces the various Intel FPGA PCI Express IP offerings and details the Intel FPGA IP Subsystem for PCI Express, including features and functional description of the various blocks within the IP. This document also describes the design flow requirements and guidelines, IP parameters, interfaces, and signals available for users, when using the Intel FPGA IP Subsystem for PCI Express.

1.1 Goal of the Intel FPGA IP Subsystem for PCI Express User Guide

The goal of the Intel FPGA IP Subsystem for PCI Express User Guide is for the user to:

- Understand the features supported by this IP
- Parameterize the IP for a specific application
- Understand the IP interfaces and how to connect them to the user logic
- Learn how to drive clock and reset inputs to the IP
- Compile the IP standalone (with interface stubs) or within a larger module

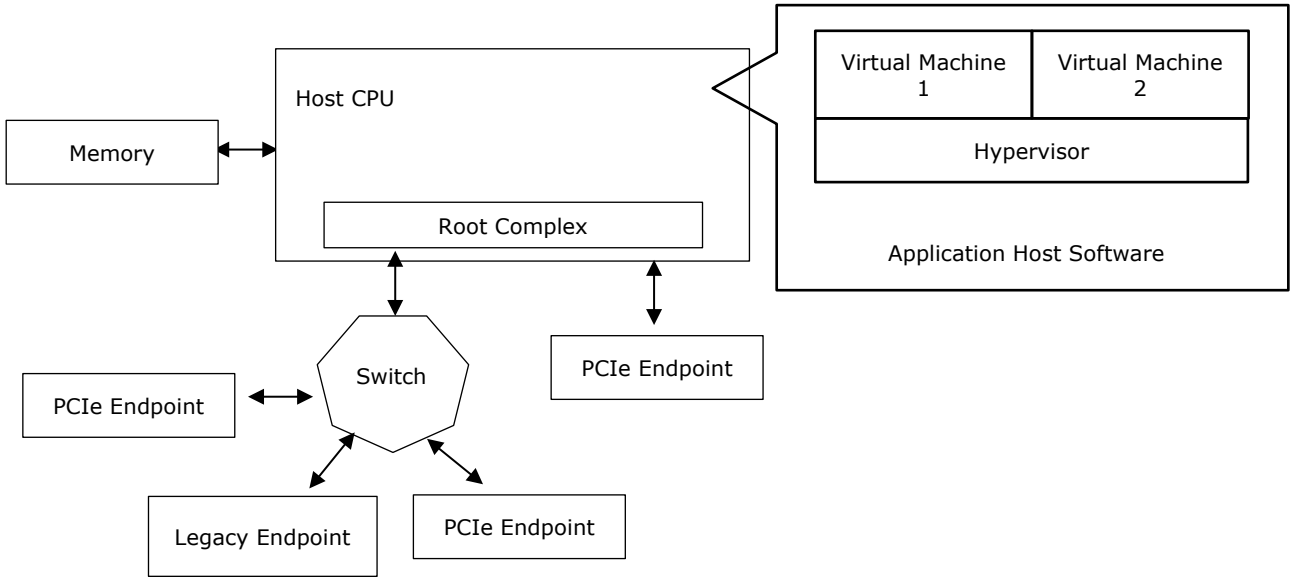
1.2 Intended Audience for Intel FPGA IP Subsystem for PCI Express

This guide is to be used by FPGA designers who will be implementing PCIe using the Intel FPGA IP Subsystem for PCI Express

1.3 What is PCI Express?

PCI Express is a point-to-point, serial interconnect bus with protocol stack that includes Transaction, Data Link, and Physical Layer. The protocol is scalable – from 1 lane to 32 lanes per link, with data on the link serialized and sent from one device to another. It uses differential signaling with complementary pair of signals for transmit and receive sides and uses packet-based transactions. You can use the Intel FPGA PCI Express IPs available in the Intel Quartus Prime Pro Edition Catalogue to implement PCI Express in your designs.

Figure 1. PCI Express Topology



1.4 What are the Intel FPGA IPs for PCI Express?

The Intel FPGA devices offer a wider variety of IPs for users to implement PCI Express in their designs. Along with the Intel FPGA IP Subsystem for PCIe Express, the table below shows the various Intel IPs that integrate PCIe as part of the IP. Features that are enabled are indicated by an "X" in the table below. If you select an IP below that does not support a required feature, you can implement it in your application logic. For example, TLP Packet Formation in the AVST IP will need to be handled by the application logic.

Avalon Streaming Intel FPGA IP for PCI Express [AVST]

Multi Channel DMA Intel FPGA IP for PCI Express [MC DMA]

Scalable Switch Intel FPGA IP for PCI Express [SEP]

Intel FPGA IP Subsystem for PCI Express [SS]

Table 1. Intel FPGA PCI Express IPs

Features	**AVST**			**MC DMA**			**SEP**			**SS**		
Tile	**P**	**F**	**R**	**P**	**F**	**R**	**P**	**F**	**R**	**P**	**F**	**R**
Device / IP												
Intel Agilex ® 7 device support	X	X	X	X	X	X	X			X		
Intel ® Stratix ® 10 device support	X			X			X					
Simulation support	X	X	X	X	X		X			X		

Hardware support	X	X	X	X	X	X	X			X		
Static port bifurcation	X	X	X	X	X	X	X			X		
Independent Reference clock support	X	X		X	X	X						
Independent PERST support (GPIO)	X	X	X	X	X	X						
Autonomous HIP	X	X	X	X	X					X		
Configuration via Protocol (CvP) (Init, Update)	X	X	X	X	X					X		
TLP packet formation				X	X	X				X		
Link partner credit handling				X	X	X				X		
Transaction ordering	X	X	X	X	X	X	X			X		
Completion reordering				X	X	X				X		
Device-dependent application clock frequency	X	X	X	X	X		X			X		
Avalon streaming interface support	X	X	X	X	X	X	X					
Avalon Memory-Mapped interface support				X	X	X						
AXI-4 (streaming, Memory-Mapped) interface support							X			X		
Application error reporting (Error reporting interface (UR/CA/Completion Timeout/Poison)	X	X	X				X			X		
Completion timeout interface	X	X	X							X		
Configuration intercept interface	X	X	X	X	X	X				X		
Debug toolkit	X	X	X	X	X	X				X		
Design Example Generation	X	X	X	X	X	X						
Design Example Driver support	X	X	X	X	X	X						
PCI Express												
Native Gen3 speed	X	X	X	X	X	X	X			X		
Native Gen4 speed	X	X	X	X	X	X	X			X		
Native Gen5 speed			X			X						
Multi-lane link (x16, x8, x4) (* Only x16 and x8 supported currently)	X	X	X	X	X	X	X			X*		
Native Endpoint	X	X	X	X	X	X				X		
Root Port	X	X	X	X	X	X						

Transaction layer bypass (TL Bypass)	X	X	X									
Separate reference clock with Independent Spread Spectrum Clocking (SRIS)	X	X	X	X	X	X				X		
Separate Reference clock with no Spread Spectrum Clocking (SRNS)	X	X	X	X	X	X				X		
Common reference clock architecture	X	X	X	X	X	X				X		
Advanced error reporting (AER)	X	X	X	X	X	X	X			X		
Up to 512-byte maximum payload size (MPS)	X	X	X	X	X	X	X			X		
Up to 4096-byte (4K) maximum read request size (MRRS)	X	X	X				X			X		
32/64-bit BAR support (prefetchable/non-prefetchable)	X	X	X	X	X	X	X			X		
Expansion ROM BAR support	X	X	X	X	X	X				X		
Single virtual channel (VC)	X	X	X	X	X	X	X			X		
Capability registers (MSI)	X	X	X	X	X		X			X		
Capability registers (MSI-X)	X	X	X	X	X	X	X			X		
Capability registers (PM)	X	X	X				X			X		
Capability registers (PRS)	X	X	X	X	X	X				X		
Capability registers (LTR)	X	X	X							X		
Capability registers (ACS)	X	X	X							X		
Capability registers (Vendor specific)	X	X	X	X	X	X				X		
10-bit tag support	X	X	X	X	X	X	X			X		
MSI-X Table				X	X	X				X		
Address Remapping Between Remote Host and Local Fabric Address Map (Device-ATT) (* Root Port only)				X*	X*	X*				X		
Multi-function and virtualization												
Single root IO virtualization (SR-IOV)	X	X	X	X	X	X	X			X		
Functional level reset (FLR)	X	X	X	X	X	X	X			X		
TLP processing hint (TPH)	X	X	X	X	X	X	X			X		

Alternative Routing-ID Interpretation (ARI)	X	X	X				X			X		
Address Translation Services (ATS)	X	X	X	X	X	X	X			X		
Process Address Space ID (PasID)	X	X	X				X					
VirtIO	X	X	X				X			X		

Refer to the Intel FPGA PCI Express IP Support Center for details on each IP.

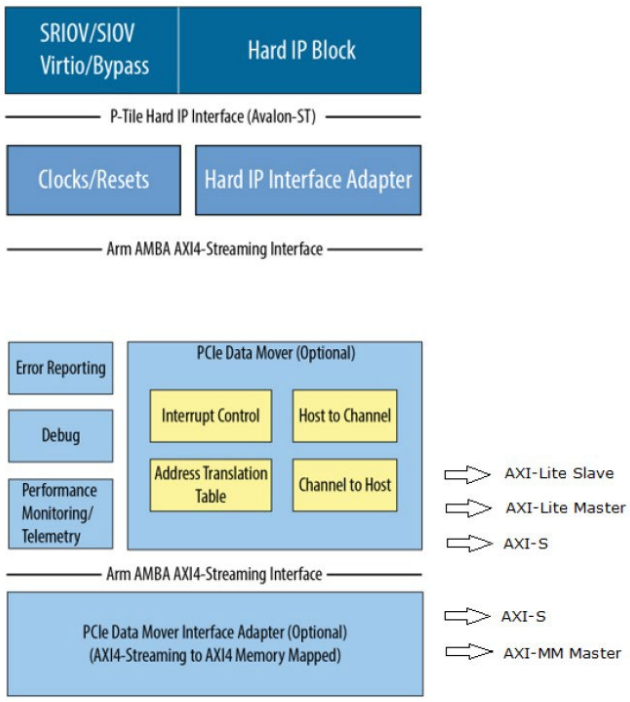
1.5 What is the Intel FPGA IP Subsystem for PCI Express?

The Intel FPGA IP Subsystem for PCI Express is a Subsystem IP that supports PCI Express* Gen3 and Gen4 in Endpoint mode. It includes the PCIe Hard IP (HIP), HIP Interface Adaptor and PCIe Data mover. It allows you to choose these blocks based on the application requirement – e.g., enable Data Mover mode for DMA type of application etc. The Subsystem IP provides users flexibility and control over the transaction layer packets by providing parametrization capabilities, functional modes, optional interfaces, error reporting and debug capabilities. It implements basic telemetry functionality as well.

The figure below shows the block diagram of the Intel FPGA IP Subsystem for PCI Express. This document covers functional mode description, parameterization of Subsystem, and interface definitions for the PCIe Subsystem and its various modes.

Note: Throughout this User Guide, the terms IP Subsystem, PCIe Subsystem, PCIe SS may be used as an abbreviation for the Intel FPGA IP Subsystem for PCI Express.

Figure 2. Intel FPGA IP Subsystem for PCI Express Block Diagram



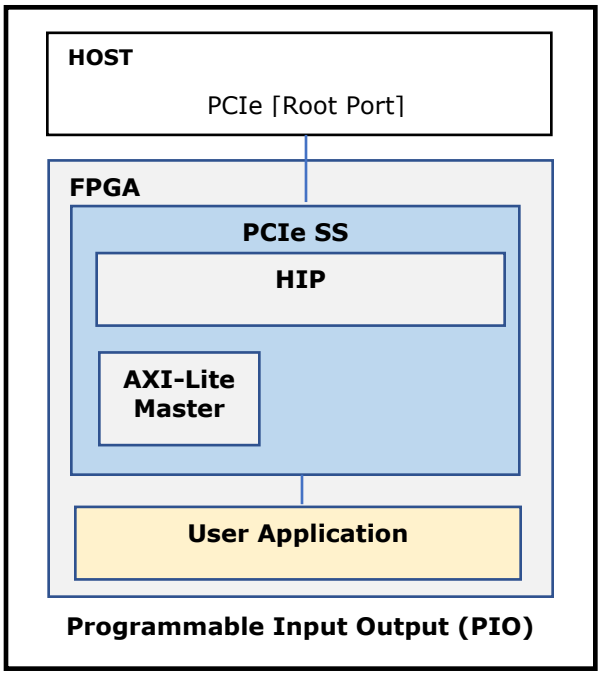
Refer to IP Architecture and Functional Description chapter for details on each of the blocks.

1.6 Example Use Models

The Intel FPGA IP Subsystem for PCI Express can be used in various applications such as an endpoint, root port, virtualization, inline processing, lookaside memory processing, etc. to move data between the source and destination.

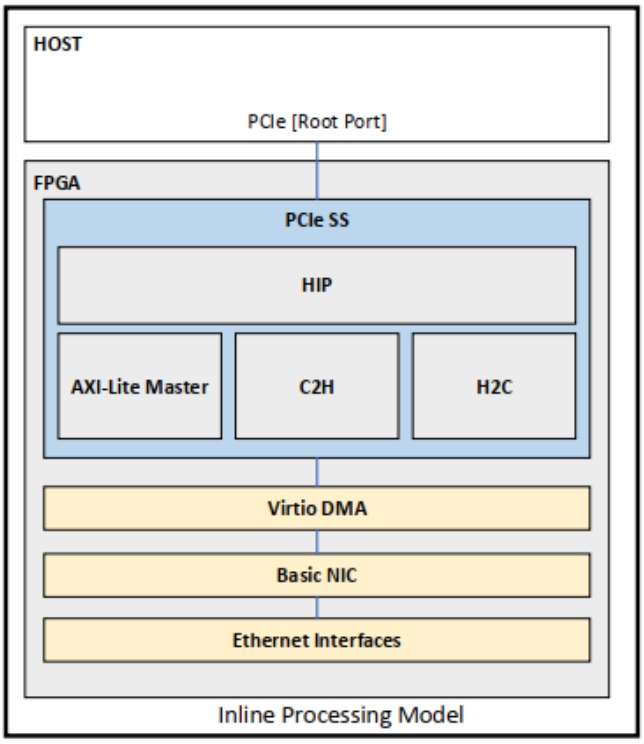
Figure 3 below shows a simple block diagram of the Subsystem in Endpoint mode connected to a root port on a host. The user can run an application like Programmable Input Output (PIO) to perform writes/reads to the host memory. The Intel FPGA IP Subsystem for PCI Express can also be used as a root port, when connected to a System-On-Chip (SoC) / processor similar to the Host in the below example

Figure 3. Example of the Programmable Input Output (PIO) Example Using the Intel FPGA IP Subsystem for PCI Express



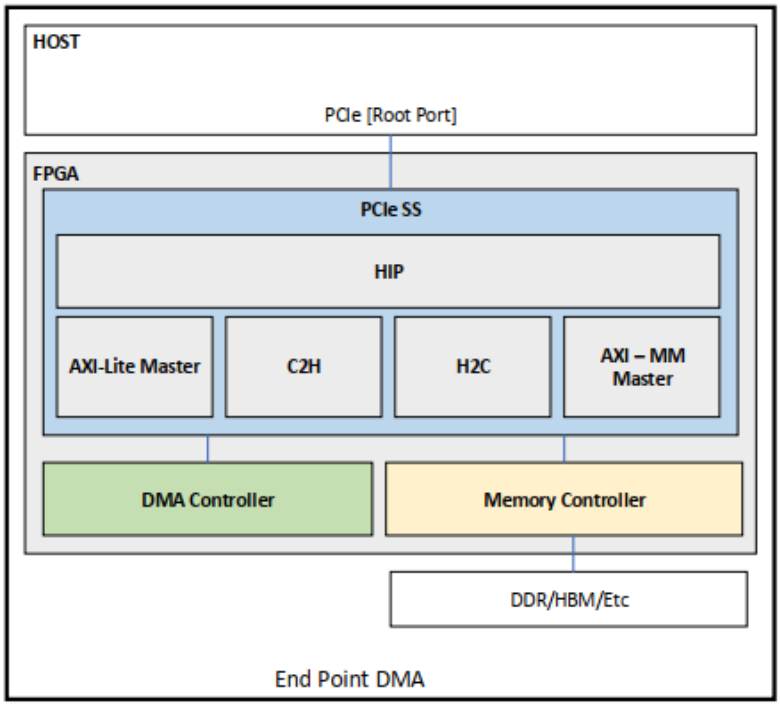
The figure 4 below shows the Subsystem connected to a processing engine directly using AXI streaming interface. The processing engine can be custom user logic implemented in the FPGA fabric, and can perform functions like DMA, e.g., VirtIO DMA connected to a BaseNIC. It receives data from HOST over AXI streaming interface and sends data towards HOST over AXI streaming interface. The Card to Host (C2H) block handles transactions coming from application logic. The Host to Card (H2C) block forwards all type of requests from HOST to application on the AXI streaming interface. The AXI-Lite Master block provides access to Control and Status Registers in the design or registers in user logic.

Figure 4. Example of the Inline Processing Model Using the Intel FPGA IP Subsystem for PCI Express



The Intel FPGA IP Subsystem for PCI Express can also be used in applications like lookaside memory processing. The figure 5 below shows an example of PCIe Subsystem interfacing with a Memory controller using an AXI-MM interface. The processing engine can be custom user logic implemented in the FPGA fabric and can perform functions like DMA. The PCIe Subsystem stores data coming from HOST into external memory. The processing engine then reads this data from external memory and performs required operations. Similarly, data coming from other IO devices are stored by the processing engine into external memory. If the processing engine wishes to transfer this data to HOST, it requests the PCIe Subsystem to read data from external memory and send it to HOST using the command interface.

Figure 5. Example of Lookaside Memory Processing Model Using the Intel FPGA IP Subsystem for PCI Express



1.7 Design Flow Requirements

1.7.1 Design Software

Intel Quartus Prime Pro Edition Software Requirements for the Subsystem FPGA IP are the following:

- Intel® Quartus Prime Pro Edition 23.1
    - Prior to getting started with the Intel FPGA IP Subsystem for PCI Express the user is required to install any required patches in conjunction with Quartus 23.1 software. Please contact your Intel Sales Representative for access to the Quartus patch installer.
- Synopsys VCS Simulator version T-2022.06-SP1-1. Other simulators may be supported in a future release.

1.7.2 Hardware

Hardware Requirements for the Subsystem FPGA IP are the following:

- Intel® Agilex™ FPGA with P-Tile (e.g., AGFB014R24B2E2V)

Standards and Specifications Compliance

Table 2. PCIe Subsystem Standards and Specifications Revision/Version

Standard	Revision/Version
PCI Express Base Specification	4.0
Single Root I/O Virtualization and Sharing Specification	1.1
Address Translation Services	1.1
Virtual I/O Device (VIRTIO)	1.0
AMBA Stream Protocol Specification	AXI-4

2. Features

2.1. Supported Features

The Intel FPGA IP Subsystem for PCI Express supports the following features:

PCIe* Features:

- Complete protocol stack including the Transaction, Data Link, and Physical Layers implemented as Hard IP.
- Configurations supported:
-

Table 3. Configurations Supported by the Intel FPGA IP Subsystem for PCI Express

	Gen3/Gen4 1x16	Gen4 1x8	Gen3/Gen4 2x8
Endpoint (EP)	Yes	Yes	Yes

Note: Currently supported with Intel Agilex ® 7 devices with P-tile (e.g., AGFB014R24B2E2V)

Note: Gen1/Gen2 configurations are supported via link down-training.

- Static port bifurcation: two x8s endpoints
- Separate reference clock with independent spread spectrum clocking (SRIS)
  - Separate reference clock with no spread spectrum clocking (SRNS)
  - Common reference clock architecture
- Single Virtual Channel (VC)
- Capability Registers:
  - Message Signaled Interrupt (MSI)
  - Message Signal Interrupt Extended (MSI-X)
  - Advanced Error Reporting (AER) (PF only)
  - Power Management (PM – D0 and D3 PCIe power states) (PF only)
  - Alternative Routing ID (ARI)
  - Address Translation Services (ATS)
  - Page Request Service (PRS)
  - Transaction Processing Hints (TPH) ("No Steering Tag (ST)" mode only)
  - Access Control Services (ACS) (For ACS, only ports 0 and 1 are supported)

  - Latency Tolerance Reporting (LTR)
  - Process Address Space ID (PASID)
  - Vendor Specific Capability
- PCI Express Advanced Error Reporting (AER) (PF only)
- Supports up to 512-byte maximum payload size (MPS)
- Supports up to 4096-byte (4 KB) maximum read request size (MRRS)
- 32/64-bit BAR support (Prefetchable/Non-Prefetchable)
- Expansion ROM BAR support
- Number of tags – 32, 64, 128, 256, 512
- Application error handling (UR/CA/Completion Timeout/Poison)
-

Multifunction and Virtualization Features (Optional):

- SR-IOV support (Maximum 8PFs, 2048 VFs across all Endpoints in a design)
- Supports single TLP prefix per TLP
- Supports VIRTIO PCI Configuration Registers
- Function Level Reset (FLR) – communicated to application through separate interface
-

User Interface Features:

- AXI4 (Streaming, Memory-Mapped, Lite) user interface for data and control signals
  - Note: AXI-Memory-Mapped user interface may be available in a future release
- AXI Streaming Source Interface
  - The AXI Streaming Source Interface comprises the master signals, and provides the start of the transaction
  - Single Stream Interface
- AXI Streaming Sink Interface
  - The AXI Streaming Sink Interface comprises the slave signals, and provides the response to the transaction from the source
  - Support for basic bare metal mode (e.g., Single physical function, AER etc.) and virtualization mode (e.g., Multiple physical functions, function level reset etc)

Table 4. Functional Mode Description

Functional Mode	Description
Power User mode	• Provides user complete control over PCIe HIP, by providing user with finer control over PCIe Transaction Layer Packet (TLP), credit handling and various modes directly to the application layer. • Sends TLPs received from the Link to user side with some additional information like BAR; number and function number • Single Interface
AXI Streaming (AXI-ST) Data Mover	• Allows high bandwidth data transfer to/from HOST memory • Hides complexity of handling PCIe TLPs, by providing a simple interface for reading and writing to Host Memory. • Checks link partner credits before transmitting packets, also provides MSI-X interrupt generation capability
AXI Memory-Mapped (AXI-MM) Data Mover	Allows users to use the AXI-MM interface for data transfer using AXI-MM protocol. **Note:** The AXI Data Mover with Memory-Mapped Interface may be available in a future release of Quartus.

- AXI-ST Bandwidth
  - Selects application's AXI streaming data bus width. The interface width is defined in terms of number of Bytes.
  - Scalable Data Bus Width and Frequency
  - 32, 64, 128-byte width

    Note: The 128-byte width support may be available in a future release of Quartus.
  - The operating frequency selection options of 250, 350, 400 or 470 MHz Refer to *Table 6. Clock Domains in Intel FPGA IP Subsystem for PCI Express* for the valid combinations of data bus width and frequencies
- AXI Lite Responder Interface
  - This is the Control and Status Register Interface to access registers implemented in Subsystem modules, including PCI/PCIe Configuration Registers of all Functions.
  - 32-bit or 64-bit @ 100-250 MHz

- Configuration Extension Interface provided to extend the configuration capabilities beyond the PCI/PCIe capabilities and implement Customer Specific Capabilities.
- Configuration Intercept Interface - The Configuration Intercept Interface (CII) allows the application logic to detect the occurrence of a Configuration (CFG) request on the link and to modify its behavior.
- Supports Application Error Reporting: The Subsystem implements Application Error Reporting registers. These registers allow user to indicate various errors. The Subsystem logic then forwards this error information to HIP block (UR/CA/Completion Timeout/Poison).
- Completion reordering (Data Mover mode only) (Optional) - The completion reordering feature instantiates reordering buffer and enables sending completion in the same order as corresponding request is received. The Subsystem will monitor the occupancy of reordering buffer apart from Non-Posted credits before sending any read request towards Host.
    - Supports max size of 16MB for completion reorder combining
    - Supports completion reorder packet cut through mode (packet store and forward mode not supported)
- Device address translation table (Data Mover mode only) (Optional) - The device address translation table (Device-ATT) is an optional feature which provides address translation of MMIO address to local FPGA memory address. Supports address remapping between remote host and local fabric address map (MMIO Address to Local AXI Fabric Address support for AXI-ST RX Req).
- MSI-X Table in Subsystem (Maximum 4096 across all Endpoints in a design) (Data Mover mode only)
    - Supports dynamic run time allocation. This feature is not supported in the current release
- Supports Link Partner Credits (exposed via credit interface) (Power User mode only) - The Power User mode exposes link partner credit to user in Transmit direction. The credits will be advertised as limit value specified in PCIe spec. User must check the availability of credits for transmitting the TLP. The Receive side of Power User mode will operate on AXI Streaming ready-valid handshake
- Transaction ordering, deadlock avoidance
    - For Power User mode, user must implement transaction ordering in user application logic.
    - For Data Mover modes, the transaction ordering is handled by the IP.
- Control Shadow Interface provided to shadow the control information from control / command registers (Optional).
- Completion timeout interface (Optional) - The PCIe Subsystem can optionally track outgoing non-posted packets to report completion timeout information to the application.

- Supports Autonomous Hard IP mode - This mode allows the PCIe Hard IP to communicate with the Host before the FPGA configuration and entry into User mode are complete.

  Note: Unless Readiness Notifications mechanisms are used, the Root Complex or system software must allow at least 1.0s after a Conventional Reset of a device before it may determine that a device that fails to return a Successful Completion status for a valid Configuration Request is a broken device. This period is independent of how quickly Link training completes.

- FPGA core configuration via Protocol (CvP Init and CvP Update) (Optional)

  Note: For Gen3 and Gen4 x16 variants, Port 0 (corresponding to lanes 0 - 15) supports the CvP features. For Gen3 and Gen4 x8 variants, only Port 0 (corresponding to lanes 0 - 7) supports the CvP features. Port 1 (corresponding to lanes 8 - 15) does not support CvP.

- Debug Toolkit for register accesses and debug (Optional)
- Software Driver support
  - Available along with the Intel Open FPGA Stack reference design

Reference Documents

P-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide

AMBA AXI4-Stream Protocol Specification

2.2. Device Family Support

The Intel FPGA IP Subsystem for PCI Express currently supports only Intel Agilex 7 devices with P-Tile.

The following table presents the resource utilization of the Subsystem IP. These results come from the compilation of the hardware-test design examples created through the IP Parameter Editor Pro for the device AGFB014R24B2E2V.

Note: The Quartus generated design examples may be available in a future release

Table 5. PCIe Subsystem Intel FPGA IP Resources Utilization – Power User Mode

PCIe	IP Core Variation	Logic Utilization (in ALMs)	Dedicated Logic Registers	M20K RAM Blocks
|pcie_wrapper|	Gen4x16	14,871/487200 (3%)	30058	101/7110 (1%)
|pcie_wrapper|	Gen4x8	10,417/487200 (2%)	21246	90/7110 (2%)
|pcie_wrapper|	Gen3x16	14,010/487200 (3%)	32107	88/7110 (1%)
|pcie_wrapper|	Gen3x8	10,772/487200 (2%)	22920	92/7110 (1%)

Table 6. PCIe Subsystem Intel FPGA IP Resources Utilization – Data Mover Mode

PCIe	IP Core Variation	Logic Utilization (in ALMs)	Dedicated Logic Registers	M20K RAM Blocks
|pcie_wrapper|	Gen4x16	39,688 / 487,200 (8 %)	95987	190/7110 (3%)
|pcie_wrapper|	Gen4x8	36,051/487200 (7%)	78476	179/7110 (3%)
|pcie_wrapper|	Gen3x16	39688/487200 (8%)	95987	190/7110 (3%)
|pcie_wrapper|	Gen3x8	36916/487200 (8%)	86058	188/7110 (3%)

Note: The above numbers are obtained with a design containing 5 physical functions and 4 virtual functions.

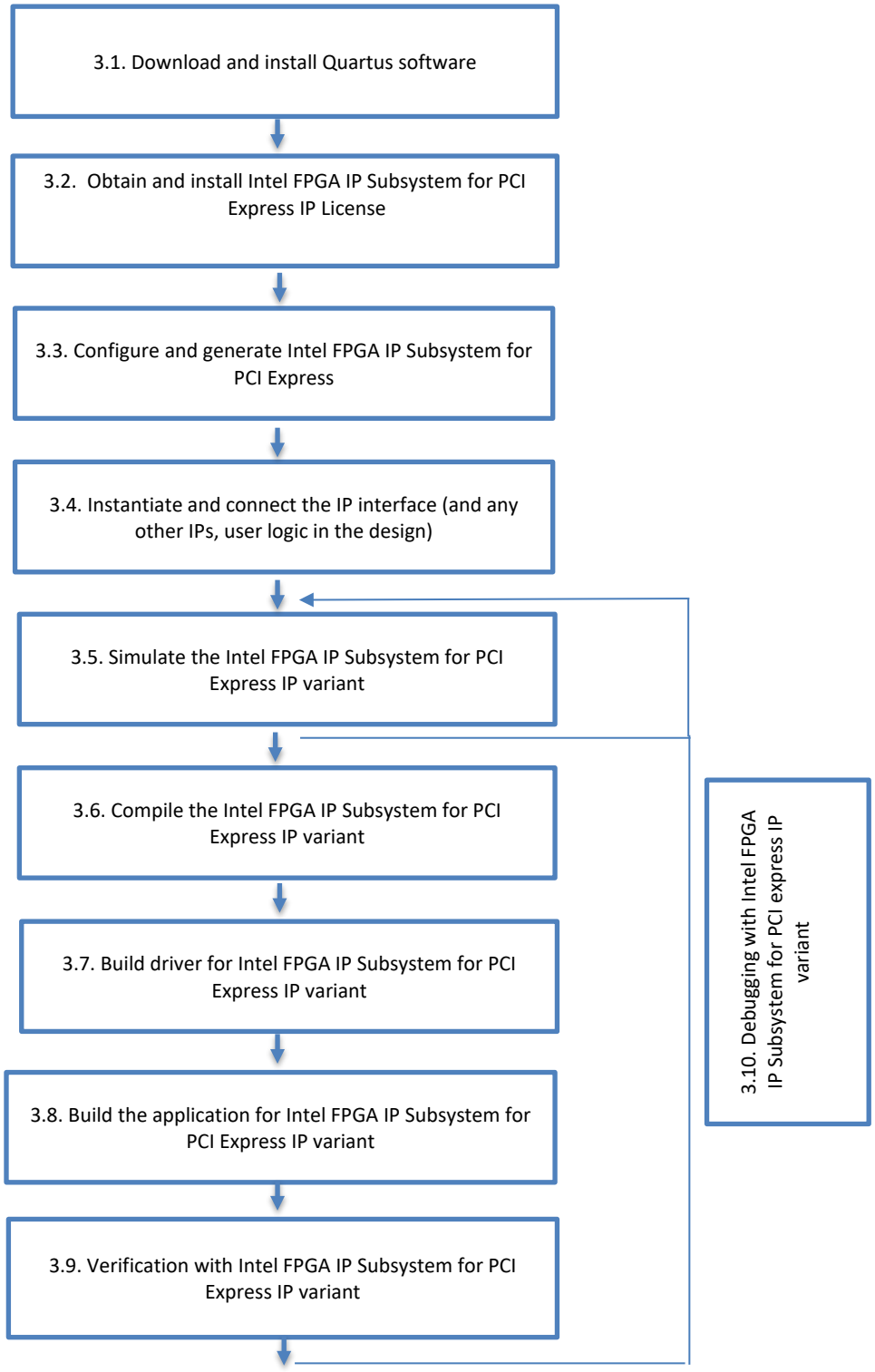
2.3. Performance

This information may be available in a future release.

3. Getting Started with Intel FPGA IP Subsystem for PCI Express

This chapter provides the steps to get started with the Intel FPGA IP Subsystem for PCI Express, from installing the required software, to instantiating the IP to simulating and compiling the IP(s) and verifying the functionality.

Figure 6. Flow diagram for getting started with Intel FPGA IP Subsystem for PCI Express


3.1. Download and install Quartus software
3.2. Obtain and install Intel FPGA IP Subsystem for PCI Express IP License
3.3. Configure and generate Intel FPGA IP Subsystem for PCI Express
3.4. Instantiate and connect the IP interface (and any other IPs, user logic in the design)
3.5. Simulate the Intel FPGA IP Subsystem for PCI Express IP variant
3.6. Compile the Intel FPGA IP Subsystem for PCI Express IP variant
3.7. Build driver for Intel FPGA IP Subsystem for PCI Express IP variant
3.8. Build the application for Intel FPGA IP Subsystem for PCI Express IP variant
3.9. Verification with Intel FPGA IP Subsystem for PCI Express IP variant
3.10. Debugging with Intel FPGA IP Subsystem for PCI express IP variant

3.1. Download and Install Quartus Software

Refer to the Intel Quartus Prime Pro Edition User Guide for details on downloading and installing the Quartus software and the necessary patches. Refer to the section *Design Flow Requirements* for details on the patches

3.2. Obtain and Install Intel FPGA IP Subsystem for PCI Express IP License

Obtain and install the Intel FPGA IP Subsystem for PCI Express License from RDC (Resource Design Centre)

3.3. Configure and generate Intel FPGA IP Subsystem for PCI Express

You can generate the Intel FPGA IP Subsystem for PCI Express as below

Table 7. Generating Intel FPGA IP Subsystem for PCI Express

Intel FPGA IP Subsystem for PCI Express in	**Description**
Standalone mode	Refer to the steps listed below to generate the Subsystem IP in standalone mode
Design example	Refer to the steps listed below to generate the Subsystem IP as part of the design example **Note:** The Quartus generated design examples may be available in a future release
Intel Open FPGA Stack (OFS) reference design	You can use the Intel pre-designed and verified system level shell designs (e.g., Intel Open FPGA Stack (IOFS)) with Subsystem IP and other Intel FPGA IPs like VirtIO, etc., and software stack to run example workloads **Note:** For examples on connecting the various interfaces of the Intel FPGA IP Subsystem for PCI Express, please contact your Intel Sales Representative for access to the Intel OFS design repository.

Below is the procedure to generate the Intel FPGA IP Subsystem for PCI Express and bring up a PCI Express link using Intel Quartus Prime Pro Edition software in standalone mode

1. Use the Intel Quartus Prime Pro Edition software to create a Quartus Project & choose the device. Currently, the Intel FPGA IP Subsystem for PCI Express is only supported on Intel Agilex 7 devices with P-tile (e.g., AGFB014R24B2E2V)

2. The Intel FPGA IP Subsystem for PCI Express parameter editor allows you to quickly configure your custom IP variation. Use the following steps to specify IP core options and parameters in the Intel Quartus Prime Pro Edition software.

    a. Invoke IP Parameter Editor by entering the following command:

```
$ qsys-edit --1=<Your_Subsystem_Name.ip> --pro --new-component-
type=pcie_ss --family=Agilex --part=<Agilex Part number>
```

The **Create New IP Variant** window appears. You can create a new Quartus project or leave it as **None**.

    b. Specify a top-level name for your new custom IP variation. The parameter editor saves the IP variation settings in a file named <your_ip>.ip
    c. Click **OK**. The parameter editor appears.
    d. Specify the parameters for your IP core variation. Refer to Chapter 5.1 Parameter Editor Parameters, for information about specific IP core parameters.

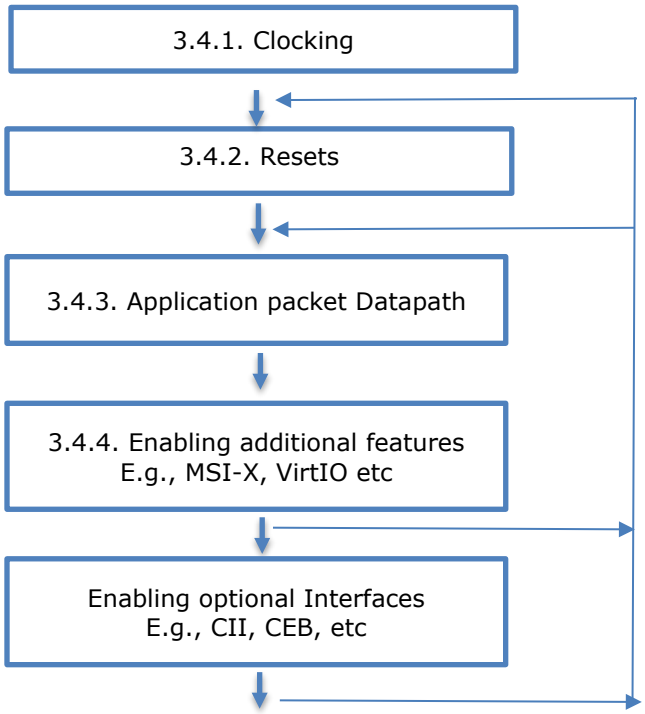
3. Generate the Subsystem IP

    a. **Generation** dialog box appears from the previous step. This allows you to generate a Subsystem IP in the stand-alone mode
    b. Specify output file generation options, and then click **Generate**. The IP variation files are generated according to your specifications.
    c. Click **Close**. The parameter editor adds the top-level.ip file to the current project automatically. If you are prompted to manually add the .ip file to the project, click **Project** > **Add/Remove Files** in Project to add the file.
    d. Use the **Generate Design Example** box to generate the Subsystem IP as part of a Quartus generated dynamic design example which instantiates the Subsystem IP with the chosen parameters, along with a basic application and software drivers to run Programmable Input Output (PIO) type traffic tests.
    Note: The Quartus generated Design examples may be available in a future release

3.4. Instantiate and Connect Intel FPGA IP Subsystem for PCI Express Interfaces

Add any additional IPs, user logic required in the design and connect the IPs and user logic. You can use Quartus platform designer and IPs in the IP catalog, and/or use RTL to design. Also, make appropriate pin assignments to connect ports and set any appropriate per-instance RTL parameters

Figure 7. Flow Diagram for Instantiating and Connecting the Interfaces in Intel FPGA IP Subsystem for PCI Express



3.4.1. Clocking

Refer to section *4.1 Clocks and Resets* for details on clocking architecture and guidelines
Refer to section *5.1 Parameter Editor Parameters* for details on parameters available in the IP parameter editor
Refer to section 6.2 Clocks and Resets for details on the interfaces and signals available in the IP

3.4.2. Resets

Refer to section *4.1 Clocks and Resets* for details on clocking architecture and guidelines
Refer to section *5.1 Parameter Editor Parameters* for details on parameters available in the IP parameter editor
Refer to section *6.2 Clocks and Resets* for details on the interfaces and signals available in the IP

3.4.3. Application Packet Datapath

Refer to section *4.2 Power User mode, 4.3 AXI Data Mover Mode* for details on application Datapath architecture and guidelines
Refer to section *5.1 Parameter Editor Parameters* for details on parameters available in the IP parameter editor
Refer to section *6.3 Application Packet Interface* for details on the interfaces and signals available in the IP

3.4.4. Enabling Additional Features

Refer to section *4.2 Power User mode, 4.3 AXI Data Mover Mode* for details on the architecture and guidelines for additional features like MSI-X, VirtIO,SR-IOV etc
Refer to section *5.1 Parameter Editor Parameters* for details on parameters available in the IP parameter editor
Refer to section *6.10 to 6.14* for details on the interfaces and signals available in the IP

3.4.5. Enabling Optional Interfaces

Refer to section *4.2 Power User mode, 4.3 AXI Data Mover Mode* for details on the architecture and guidelines for optional interfaces like CII, Debug Toolkit and Reconfiguration interface, etc
Refer to section *5.1 Parameter Editor Parameters* for details on parameters available in the IP parameter editor
Refer to sections 6.4 *to 6.9* for details on the interfaces and signals available in the IP

Note: For examples on connecting the various interfaces of the Intel FPGA IP Subsystem for PCI Express, please contact your Intel Sales Representative for access to the Intel OFS design repository.

3.5. Simulate the Intel FPGA IP Subsystem for PCI Express IP Variant

Simulate the design to verify functionality of the IP and design. The IP simulation files for the Subsystem IP can be found in the /ip/sim/<simulator> folder in the Quartus project directory.

You can also use 3rd party Bus Functional Models (BFMs), Verification IPs (VIPs) to verify the IP in simulation

Note: For examples on simulating the Intel FPGA IP Subsystem for PCI Express, please contact your Intel Sales Representative for access to the Intel OFS design repository.

3.6. Compile the Intel FPGA IP Subsystem for PCI Express IP Variant

1. Use the Quartus Prime Pro software -> Processing menu to select Start Compilation. Timing can be verified using the TimeQuest Timing Analyzer of the Quartus Prime Pro. Use the assembler to generate the configuration bit stream as a .sof (or .pof) file. This file is what you download to a board to perform hardware verification.

2. Download the bit stream onto the device and bring up the PCIe link(s) in the design. Ensure that your device is linked up and enumerated in the PCI Express topology. You can use utilities like lspci, setpci to obtain general information of the device like link speed, link width etc.

For example, to read the negotiated link speed for the given device in a system, you can use the following commands:

```
sudo lspci –s $bdf -vvv
```

`-s` refers to "slot" and is used with the bus/device/function number (bdf) information. Use this command if you know the bdf of the device in the system topology

```
sudo lspci –d :$did -vvv
```

`-d` refers to device and is used with the device ID as configured in the parameter settings of the PCIe IP subsystem (vid:did). Use this command to search using the device ID.

3.7. Software drivers for Intel FPGA IP Subsystem for PCI Express IP Variant

The software drivers for the Intel FPGA IP Subsystem for PCI Express vary when the IP is configured in different modes (e.g., Root Port vs Endpoint mode). You must ensure the correct software drivers are used in each of these modes for proper functionality of the IP. You can download, customize, build, and install the software drivers for the Subsystem IP as below

Table 8. Software drivers for Intel FPGA IP Subsystem for PCI Express

Intel FPGA IP Subsystem for PCI Express in	Description
Standalone mode	You must create your own software drivers based on the application requirement.
Design example	The software drivers for the IP are generated as part of the design example generation **Note:** The Quartus generated Design examples may be available in a future release.
Intel OFS reference design	Software drivers for the IP and example workloads are provided as part of the reference design

	Note: For examples on software drivers of the Intel FPGA IP Subsystem for PCI Express, please contact your Intel Sales Representative for access to the Intel OFS design repository.

3.8. Build the Application for Intel FPGA IP Subsystem for PCI Express IP Variant

Based on the previous step, configure the PCIe link to run applications for traffic tests, Read/Write transactions, measure performance etc.

Table 9. Application for Intel FPGA IP Subsystem for PCI Express

Intel FPGA IP Subsystem for PCI Express in	Description
Standalone mode	You must create your own software application based on the application requirement
Design example	Example application for the IP is generated as part of the design example generation. **Note:** The Quartus generated Design examples may be available in a future release.
Intel OFS reference design	Example application for the IP and example workloads are provided as part of the reference design **Note:** For examples on application and workloads of the Intel FPGA IP Subsystem for PCI Express, please contact your Intel Sales Representative for access to the Intel OFS design repository.

3.9. Verification with Intel FPGA IP Subsystem for PCI Express IP Variant

Table 10. Verification with Intel FPGA IP Subsystem for PCI Express

Intel FPGA IP Subsystem for PCI Express in	Description
Standalone mode	You must create your own verification test suite based on the application requirement.

Design example	Example testbench for the IP are generated as part of the design example generation. **Note:** The Quartus generated Design examples may be available in a future release.
Intel OFS reference design	Example verification suite for the IP and example workloads are provided as part of the reference design. **Note:** For examples on verification suite of the Intel FPGA IP Subsystem for PCI Express, please contact your Intel Sales Representative for access to the Intel OFS design repository.

3.10. Debugging with Intel FPGA IP Subsystem for PCI Express IP Variant

As you bring up your PCI Express system, you may face issues related to FPGA configuration, link training, BIOS enumeration, data transfer, and so on. This chapter suggests some strategies to resolve the common issues that occur during bring-up. You can additionally use the Intel Quartus SignalTapII Analyzer, In Systems Sources and Probes (ISSP) tools, Intel FPGA IP Subsystem for PCI Express IP Debug Toolkit to identify the issues.

3.10.1. Hardware

Typically, PCI Express link-up involves the following steps:

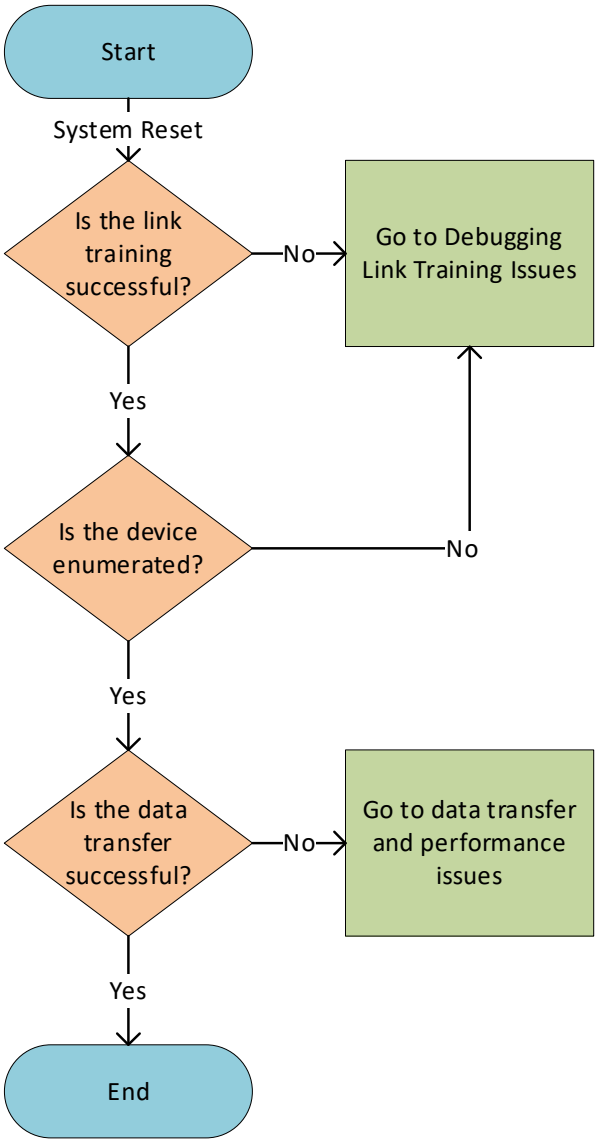
1. Link training
2. BIOS enumeration and data transfer

The following sections describe the flow to debug link issues during the hardware bring-up. Intel recommends a systematic approach to diagnosing issues as illustrated in the following figure.

Additionally, you can use the IP Debug Toolkit for debugging the PCIe links when using the Intel FPGA IP Subsystem for PCI Express. The Debug Toolkit includes the following features:

- Protocol and link status information
- Basic and advanced debugging capabilities including register read access and Eye viewing capability
- System Console based interface to access status registers of the Intel FPGA IP Subsystem for PCI Express IP using scripts

Figure 8. PCI Express Debug Flow Chart



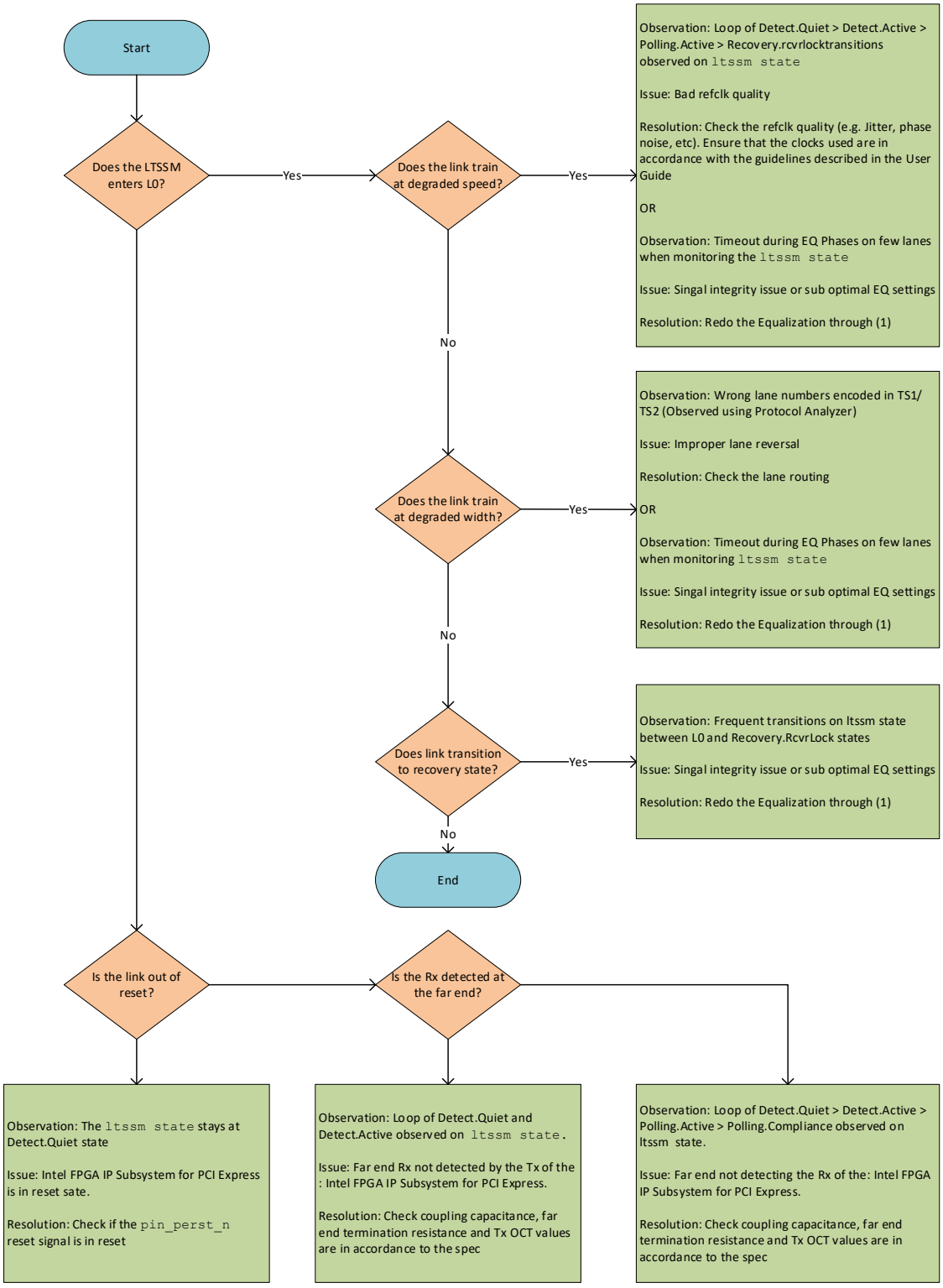
3.10.1.1. Debugging Link Training Issues

The Physical Layer automatically performs link training and initialization without software intervention. This is a well-defined process to configure and initialize the device's Physical Layer and link so that PCIe packets can be transmitted. Some examples of link training issues include:

- Link fails to negotiate to expected link speed
- Link fails to negotiate to the expected link width
- LTSSM fails to reach/stay stable at L0

Use the flow chart below to identify the potential cause of the issue seen during link training when using the Intel FPGA IP Subsystem for PCI Express.

Figure 9. Link Training Debugging Flow



Note: (*) Redo the equalization using the Link Equalization Request 8.0 GT/s bit of the Link Status 2 register for 8.0 GT/s or Link Equalization Request 16.0 GT/s bit of the 16.0 GT/s Status Register.

Use the following debug tools for debugging link training issues observed on the PCI Express link when using the Intel FPGA IP Subsystem for PCI Express.

3.10.1.1.1. Operating System Tools and Utilities

You can use Linux* utilities like lspci, setpci to obtain general information of the device like link speed, link width etc. To read the negotiated link speed for the Subsystem device in a system, you can use the following commands:

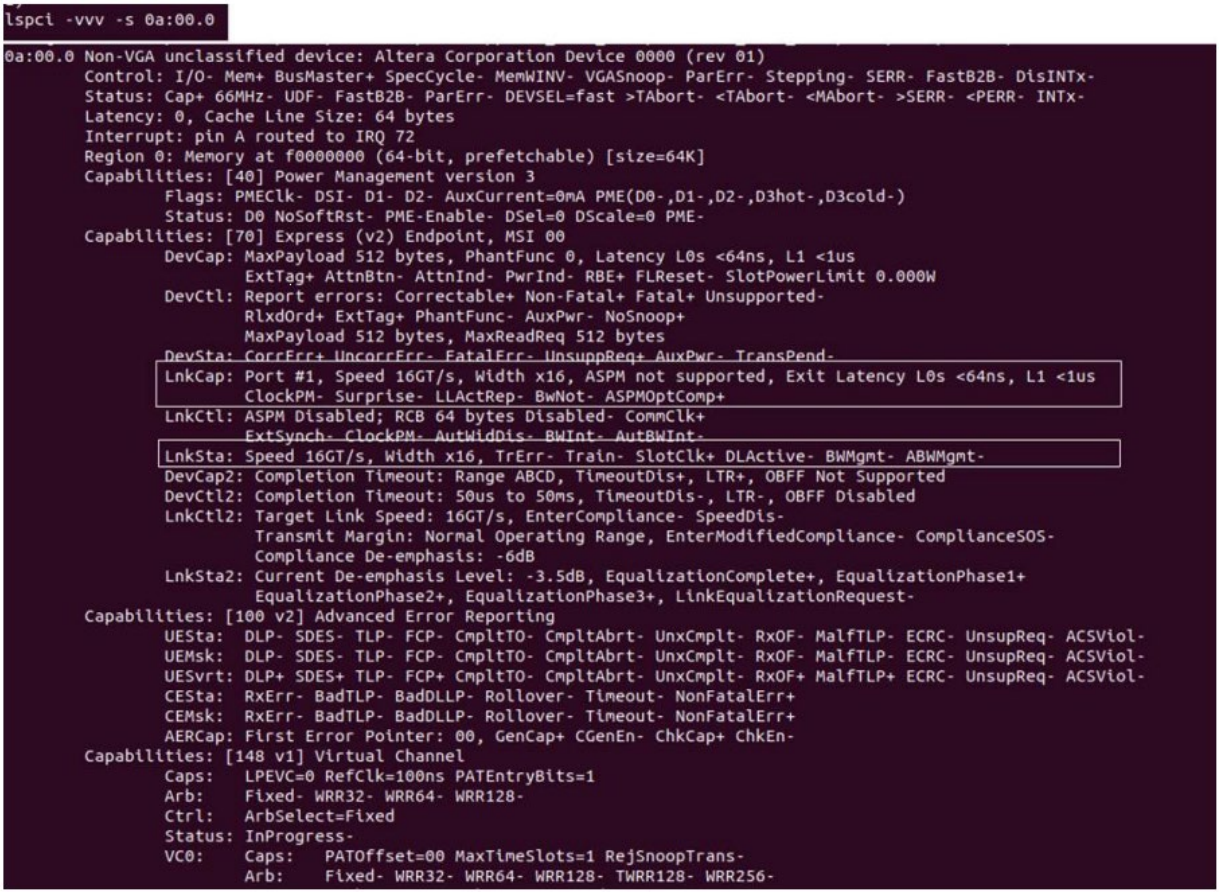
```
sudo lspci –s $bdf -vvv
```

`-s` refers to "slot" and is used with the bus/device/function number (bdf) information. Use this command if you know the bdf of the device in the system topology

```
sudo lspci –d :$did -vvv
```

`-d` refers to device and is used with the device ID as configured in the parameter settings of the PCIe IP subsystem (vid:did). Use this command to search using the device ID.

Figure 10. lspci Output



The **LnkCap** under Capabilities indicates the advertised link speed and width capabilities of the device. The **LnkSta** under Capabilities indicates the negotiated link speed and width of the device.

You can rescan the PCIe bus using the following commands. You must have root privileges to perform the below commands:

```
#To detach the device from the tree:
% echo 1 > /sys/bus/pci/devices/0a:00.0/remove

#To rescan the bus:
% echo 1 > /sys/bus/pci/rescan
```

3.10.1.1.2. Signal Tap Logic Analyzer

Using the Signal Tap Logic Analyzer, you can monitor the following top-level signals from the Intel FPGA IP Subsystem for PCI Express to confirm the failure symptom for any port type (Root Port, Endpoint) and configuration (Gen4/Gen3).

Table 11. Top-Level Signals to be Monitored for Debugging

Signal	Description	Expected value for successful link up
p<n>_pin_perst_n where n = 0, 1, 2, 3	Active-low asynchronous output signal from the PCIe Subsystem IP. It is derived from the pin_perst_n input signal.	1'b1
ninit_done	Active-low asynchronous output signal from the Reset Release Intel FPGA IP. High indicates that the FPGA device is not yet fully configured, and low indicates the device has been configured and is in normal operating mode. For more details on the Reset Release Intel FPGA IP.	1'b0
p<n>_reset_status_n where n = 0, 1, 2, 3	Active-low output signal from the PCIe Subsystem IP, synchronous to coreclkout_hip_toapp of the Subsystem IP. The reset_status_n output of HIP drives this signal. Held low until pin_perst_n is deasserted and the PCIe Hard IP comes out of reset. When port bifurcation is used, there is one such signal for each port. The application logic can use this signal to drive its reset network.	1'b1
p<n>_ss_app_dlup where n = 0, 1, 2, 3	Indicates the data link layer is up. Synchronous to coreclkout_hip_toapp clock of the Subsystem IP	1'b1
p<n>_ss_app_serr where n = 0, 1, 2, 3	Indicates system error is detected. Synchronous to coreclkout_hip_toapp clock of the Subsystem IP EP mode: Asserted when the P-Tile PCIe Hard IP sends a message of correctable/non-fatal/fatal error.	1'b0
link_up_o	Active-high output signal from the PCIe Hard IP, synchronous to coreclkout_hip clock of the Hard IP. Indicates that the Physical Layer link is up. (This signal is currently available at the Hard IP interface)	1'b1
dl_up_o	Active-high output signal from the PCIe Hard IP, synchronous to coreclkout_hip of the Hard IP. Indicates that the Data Link Layer is active.	1'b1
ltssm_state_o[5:0]	Indicates the LTSSM state, synchronous to coreclkout_hip of the Hard IP. (This signal is currently available at the Hard IP interface) • 6'h00: S_DETECT_QUIET • 6'h01: S_DETECT_ACT • 6'h02: S_POLL_ACTIVE • 6'h03: S_POLL_COMPLIANCE • 6'h04: S_POLL_CONFIG • 6'h05: S_PRE_DETECT_QUIET • 6'h06: S_DETECT_WAIT • 6'h07: S_CFG_LINKWD_START	6'b11 (L0)

	• 6'h08: S_CFG_LINKWD_ACCEPT • 6'h09: S_CFG_LANENUM_WAIT • 6'h0A: S_CFG_LANENUM_ACCEPT • 6'h0B: S_CFG_COMPLETE • 6'h0C: S_CFG_IDLE • 6'h0D: S_RCVRY_LOCK • 6'h0E: S_RCVRY_SPEED • 6'h0F: S_RCVRY_RCVRCFG • 6'h10: S_RCVRY_IDLE • 6'h11: S_L0 • 6'h12: S_L0S • 6'h13: S_L123_SEND_EIDLE • 6'h14: S_L1_IDLE • 6'h15: S_L2_IDLE • 6'h16: S_L2_WAKE • 6'h17: S_DISABLED_ENTRY • 6'h18: S_DISABLED_IDLE • 6'h19: S_DISABLED • 6'h1A: S_LPBK_ENTRY • 6'h1B: S_LPBK_ACTIVE • 6'h1C: S_LPBK_EXIT • 6'h1D: S_LPBK_EXIT_TIMEOUT • 6'h1E: S_HOT_RESET_ENTRY • 6'h1F: S_HOT_RESET • 6'h20: S_RCVRY_EQ0 • 6'h21: S_RCVRY_EQ1 • 6'h22: S_RCVRY_EQ2 • 6'h23: S_RCVRY_EQ3	
ss_app_surprise_down_err	Active high asynchronous output signal. Indicates that a surprise down event is occurring in the HardIP controller. This error event is triggered when the PHY layer reports to the Data Link Layer that the link is down.	1'b0
ss_app_rx_par_err	Indicates a parity error detected at the input of the HIP'S RX buffer. Asserts for a single cycle. Synchronous to the axi_st_clk clock Note: Application must reset the HardIP if this occurs because parity errors can leave the HardIP in an unknown state.	1'b0
ss_app_tx_par_err	Indicates a parity error during TX TLP transmission at the HIP. Asserts for a single cycle. Synchronous to the axi_st_clk clock	1'b0

3.10.1.1.3. Additional Debug Tools

Use the Debug Toolkit, Control and Status Responder Interfaces (lite_csr) of the Intel FPGA IP Subsystem for PCI Express to access additional registers like the Subsystem debug registers (for example, receiver detection, etc.).

3.10.1.2. Debugging Data Transfer and Performance Issues

There are many possible reasons causing the PCIe link to stop transmitting data. The PCI Express base specification defines three types of errors, outlined in the table below:

Table 12. Error Types Defined by the PCI Express Base Specification

Type	Responsible Agent	Description
Correctable	Hardware	While correctable errors may affect system performance, data integrity is maintained.
Uncorrectable, non-fatal	Device software	Uncorrectable, non-fatal errors are defined as errors in which data is lost, but system integrity is maintained. For example, the fabric may lose a particular TLP, but it still works without problems.
Uncorrectable, fatal	System software	Errors generated by a loss of data and system failure are considered uncorrectable and fatal. Software must determine how to handle such errors: whether to reset the link or implement other means to minimize the problem.

3.10.1.2.1. Advanced Error Reporting (AER)

Each PCI Express compliant device must implement a basic level of error management and can optionally implement advanced error management. The PCI Express Advanced Error Reporting Capability is an optional Extended Capability that may be implemented by PCI Express device functions supporting advanced error control and reporting.

The Intel FPGA IP Subsystem for PCI Express implements both basic and advanced error reporting. Error handling for a Root Port is more complex than that of an Endpoint. In this IP, the Physical Functions (PFs) are always capable of AER (enabled by default). There is no AER implementation for Virtual Functions (VFs).

Use the AER capability of the IP to identify the type of error and the protocol stack layer in which the error may have occurred. Refer to the PCI Express Capability Structures section of the Configuration Space Registers appendix for the AER Extended Capability Structure and the associated registers.

Table 13. Correctable Error Status Register (AER)

Observation	Issue	Resolution
Receiver error bit set	Physical layer error which may be due to a PCS error when a lane is in L0, or a Control symbol being received in the wrong lane, or signal Integrity issues where the link may transition from L0 to the Recovery state.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error. Also refer to the flow chart in *Debugging Link Training Issues* to obtain more information about the error.
Bad DLLP bit set	Data link layer error which may occur when a CRC verification fails.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error.
Bad TLP bit set	Data link layer error which may occur when an LCRC verification fails or when a sequence number error occurs.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error.
Replay_num_rollover bit set	Data link layer error which may be due to TLPs sent without success (no ACK) four times in a row.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error.
replay timer timeout status bit set	Data link layer error which may occur when no ACK or NAK was received within the timeout period for the TLPs transmitted.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space

		registers, Subsystem Debug Registers to obtain more information about the error.
Advisory non-fatal	Transaction layer error which may be due to higher priority uncorrectable error detected.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error.
Corrected internal error bits set	Transaction layer error which may be due to an ECC error in the internal Hard IP RAM.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error.

Table 14. Uncorrectable Error Status Register (AER)

Observation	Issue	Resolution
Data link protocol error	Data link layer error which may be due to transmitter receiving an ACK/NAK whose Seq ID does not correspond to an unacknowledged TLP or ACK sequence number.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error.
Surprise down error	Data link layer error which may be due to link going down during L0, indicating the physical layer link is going down unexpectedly.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error
Flow control protocol error	Transaction layer error which can be due to the receiver reporting more than the allowed credit limit. This error occurs	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space

	when a component does not receive updated flow control credits with the 200 µs limit.	registers, Subsystem Debug Registers, TX/RX flow control credit interfaces to obtain more information about the error.
Poisoned TLP received	Transaction layer error which can be due to a received TLP with the EP bit set.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error
Completion timeout	Transaction layer error which can be due to a completion not received within the required amount of time after a non-posted request was sent.	Use the Control and Status Register Responder Interface (lite_csr), completion timeout interface to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error
Completer abort	Transaction layer error which can be due to a completer being unable to fulfill a request due to a problem with the requester or a failure of the completer.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error
Unexpected completion	Transaction layer error which can be due to a requester receiving a completion that doesn't match any request awaiting a completion. The TLP is deleted by the Hard IP and not presented to the Application Layer.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error
Receiver overflow	Transaction layer error which can be due to a receiver receiving more TLPs than the available receive buffer space. The	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space

	TLP is deleted by the Hard IP and not presented to the Application Layer.	registers, Subsystem Debug Registers, TX/RX flow control credit interfaces to obtain more information about the error.
Malformed TLP	Transaction layer error which can be due to errors in the received TLP header. The TLP is deleted by the Hard IP and not presented to the Application Layer	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error
ECRC error	Transaction layer error which can be due to an ECRC check failure at the receiver despite the fact that the TLP is not malformed and the LCRC check is valid. The Hard IP block handles this TLP automatically. If the TLP is a nonposted request, the Hard IP block generates a completion with a completer abort status. The TLP is deleted by the Hard IP and not presented to the Application Layer.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error
Unsupported request	Transaction layer error which can be due to the completer being unable to fulfill the request. The TLP is deleted in the Hard IP block and not presented to the Application Layer. If the TLP is a non-posted request, the Hard IP block generates a completion with Unsupported Request status.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error
ACS violation	Transaction layer error which can be due to access control error in the received posted or non-posted request.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem

		Debug Registers to obtain more information about the error
Uncorrectable internal error	Transaction layer error which can be due to an internal error that cannot be corrected by the hardware.	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error
Atomic egress blocked		Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error
TLP prefix blocked	EP or RP only	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error
Poisoned TLP egress blocked	EP or RP only	Use the Control and Status Register Responder Interface (lite_csr) to access the PCIe configuration space registers, Subsystem Debug Registers to obtain more information about the error

3.10.1.2.2. Second-Level Debug Tools

Use the following debug tools for second-level debug of any issue observed on the PCI Express link when using the Intel FPGA IP Subsystem for PCI Express:

Using the Completion Timeout Interface

- Refer to Sections 4.2.12, 6.9 for details on this interface.

Using the Control Shadow Interface

• Refer to Sections 4.2.6, 6.7 for details on this interface.

Using the Configuration Intercept Interface
• Refer to Sections 4.2.7, 6.5 for details on this interface

Using the Configuration Space Extension Interface
• Refer to Sections 4.2.5, 6.4 for details on this interface

Using the Flow Control Credit Handling Interfaces
• Refer to Sections 4.2.11, 6.8, for details on these interfaces

Using the Control and Status Register Responder Interface
• Refer to Section 6.11. Control and Status Register Responder Interface (Lite_csr) Interface for details on this interface

Using the Application Error Interface
• Refer to Section 4.2.3 for details on this interface

Using the Debug toolkit
• Refer to Section 4.2.4 for details on this interface

Reference Documents

P-tile Avalon Streaming Intel FPGA IP for PCI Express User Guide

AMBA AXI4-Stream Protocol Specification

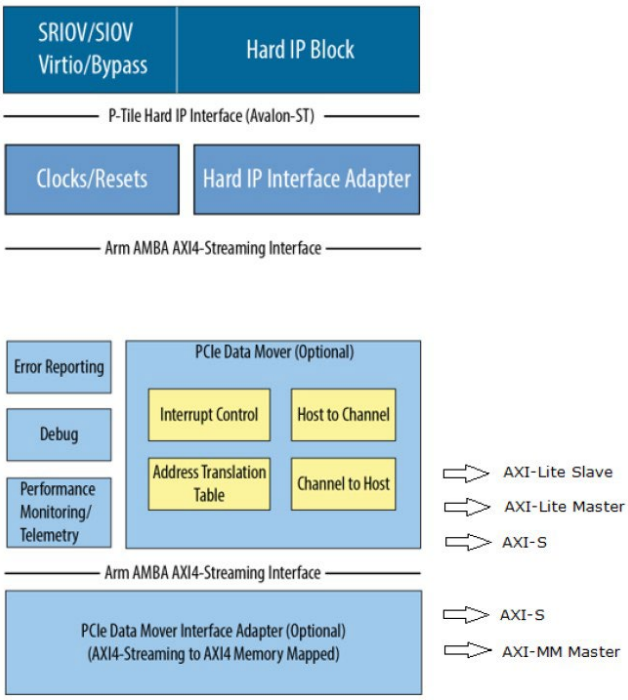
Intel Quartus Prime Pro Edition User Guide

Note: For examples on connecting the various interfaces of the Intel FPGA IP Subsystem for PCI Express, please contact your Intel Sales Representative for access to the Intel OFS design repository.

4. IP Architecture and Functional Description

This chapter describes the architecture details of the Subsystem IP and details the various blocks and modes available in the IP that you can use. The figure below displays the Subsystem IP block diagram, showing important blocks and their connections.

Figure 11. Intel FPGA IP Subsystem for PCI Express Block Diagram



The PCIe Subsystem supports three main functional modes:

Table 15. PCIe Subsystem Functional Modes

Functional Mode	Description
Power User	• Allows user complete control over PCIe HIP. • The user can implement functionality of interest with finer control over PCIe Transaction Layer Packet (TLP), credit handling and various modes provided by HIP.
AXI4 Streaming Data Mover	• Allows high bandwidth data transfer to/from HOST memory. • Hides complexity of handling PCIe TLPs. • Provides simple interface for reading and writing to Host Memory.

	• Data Mover checks link partner credits before transmitting packets. • Also provides MSI-X interrupt generation capability.
AXI4 Memory-Mapped Data Mover	• Allows users to use the AXI-MM interface for data transfer using AXI-MM protocol. **Note:** The AXI-MM Data Mover mode may be available in future release.

4.1. Clocks and Resets

The Intel FPGA IP Subsystem for PCI Express has the following clock domains to drive the various interfaces.

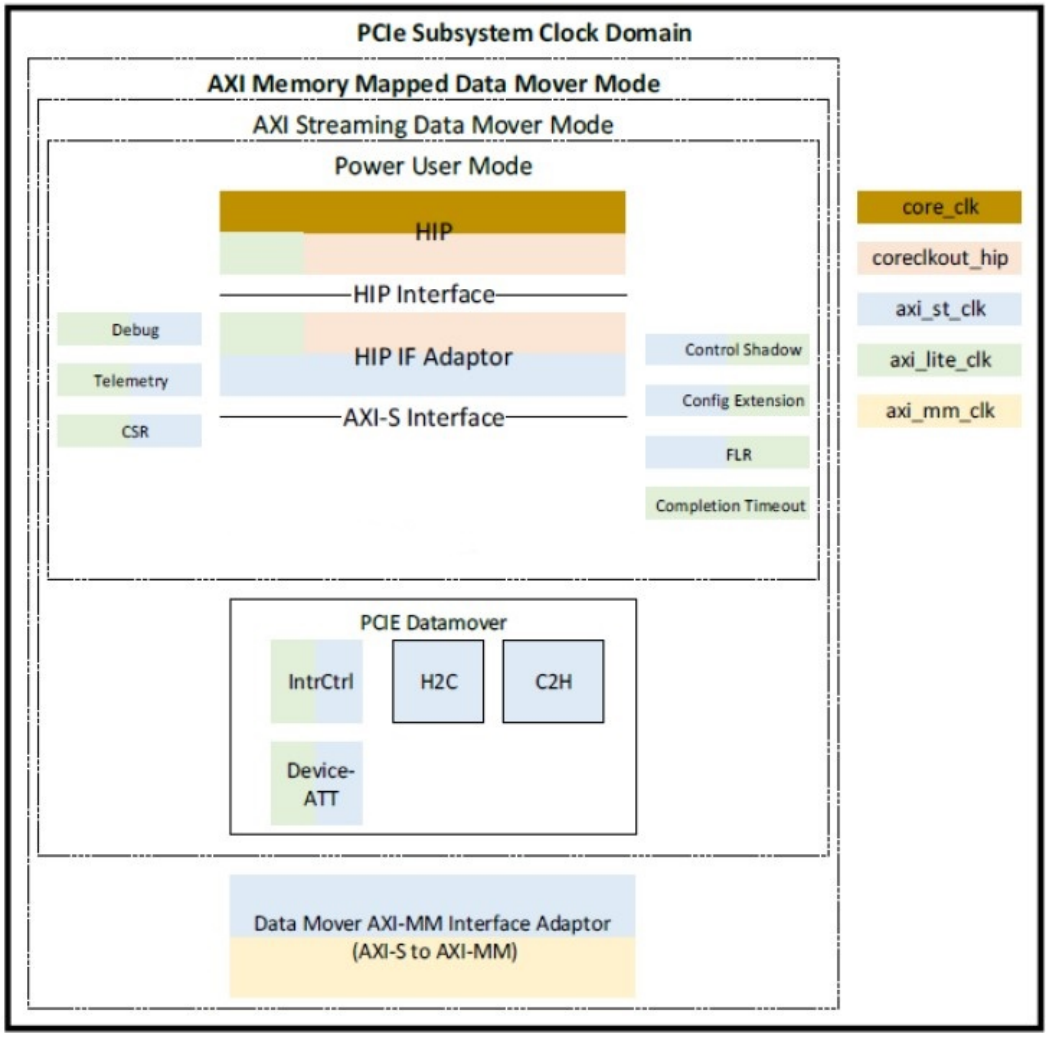
Table 16. Clock domains in Intel FPGA IP Subsystem for PCI Express

Clock Domain	Description
core_clk	This clock is synchronous to the SerDes parallel clock Gen4: 1000MHz Gen3: 500MHz Gen2: 250MHz Gen1: 125MHz
coreclkout_hip_toapp	The coreclkout_hip output of the HIP drives this clock. Application can use this clock to generate PCIe Subsystem clocks. Gen4: 500MHz Gen3: 250MHz Gen2/Gen1: Gen1/Gen2 is supported only via link down-training and not natively. Hence, the coreclkout_hip_toapp clock frequency depends on the configuration you choose in the IP Parameter Editor. For example, if you choose a Gen3 configuration, the application clock frequency is 250MHz.
axi_st_clk	This global clock signal is an input to the IP. This clock is used to clock the AXI-ST Datapath interfaces (TX, RX) to the application logic. All signals of the AXI-ST Datapath interface are sampled on the rising edge of axi_st_clk. Gen4: 470/400/350/250 MHz (32-, 64- byte width) Gen3: 470/400/350/250 MHz (32-, 64- byte width)
axi_lite_clk	This global clock signal is an input to the IP. This clock is used to clock the sideband interfaces, e.g., control and status register interface, completion timeout

	interface, etc. All signals are sampled on the rising edge of axi_lite_clk. Frequency: 100-250MHz (Default 250MHz)
axi_mm_clk	This global clock signal is an input to the IP. This clock is used to clock the application AXI Memory-Mapped interfaces, e.g., APP AXI MM Initiator Interface. All signals are sampled on the rising edge of axi_mm_clk Gen4: 350/400 MHz Gen3: 250 MHz

The figure below shows clock domains in the PCIe Subsystem. All the clocks must be always on for the correct functioning of a design.

Figure 12. Clock Domains in Intel FPGA IP Subsystem for PCI Express



The Intel FPGA IP Subsystem for PCI Express has two types of resets:

- Bus Resets - The bus resets are AXI specification defined reset signals, which are used to reset the logic in Subsystem interfacing with AXI fabric.
- IP Resets - The IP reset signals perform cold/warm reset sequences.

The Intel FPGA IP Subsystem for PCI Express has the following reset domains in Power User mode to drive the various interfaces.

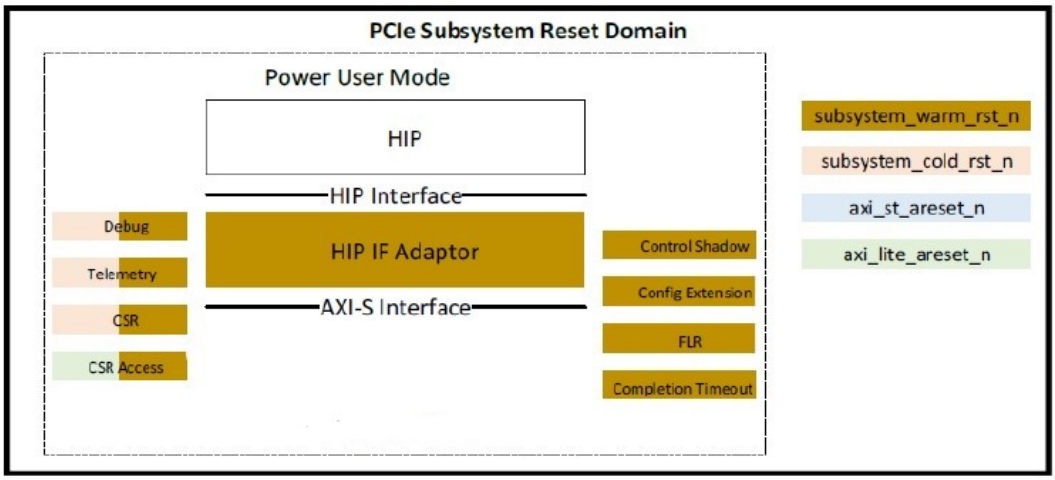
Table 17. Reset Domains in Intel FPGA IP Subsystem for PCI Express

Reset Domain	Type	Description
Cold reset	IP reset	A Fundamental Reset following the application of main power This will reset the following: • Bus resets (AXI-ST/AXI-MM/AXI-Lite) • Hard IP • Sticky registers of configuration space • When cold reset is triggered, warm reset and bus resets must be asserted Refer to *PCI Express Base Specification Revision 4.0 for more details on cold reset*
Warm reset	IP reset	A Fundamental Reset without cycling main power This will reset the following: • Bus resets (AXI-ST/AXI-MM/AXI-Lite) • Hard IP • The warm reset can be triggered multiple times by user without going through cold reset sequence. • When warm reset is triggered, Bus resets must be asserted Refer to *PCI Express Base Specification Revision 4.0 for more details on warm reset*
AXI-ST reset	Bus reset	This will reset the AXI-ST main data path interface (e.g., AXI-ST TX/RX)
AXI-Lite reset	Bus reset	This will reset the AXI-Lite sideband interfaces (e.g., Completion timeout, control and status register)

AXI-MM reset	Bus reset	This will reset the AXI-Memory-Mapped interface. **Note:** The AXI-MM Data Mover mode may be available in future release.

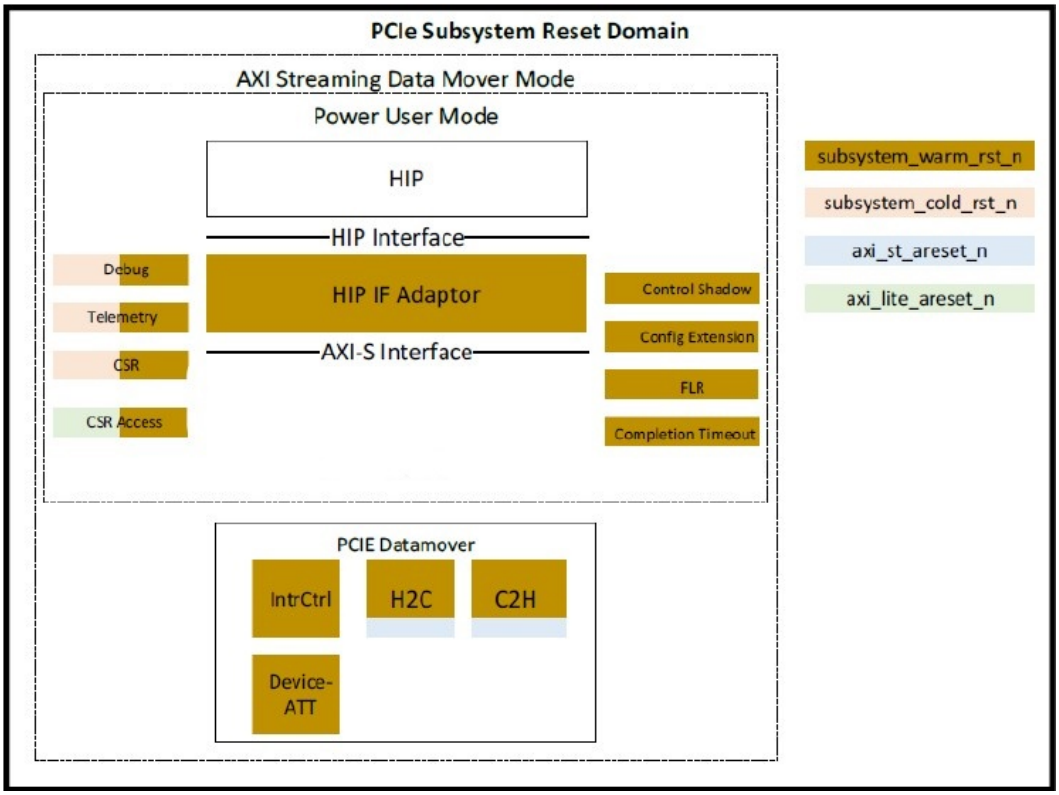
The figure below indicates the reset domains when the IP is configured in Power User and Data Mover modes.

Figure 13. Reset Domains in Power User mode



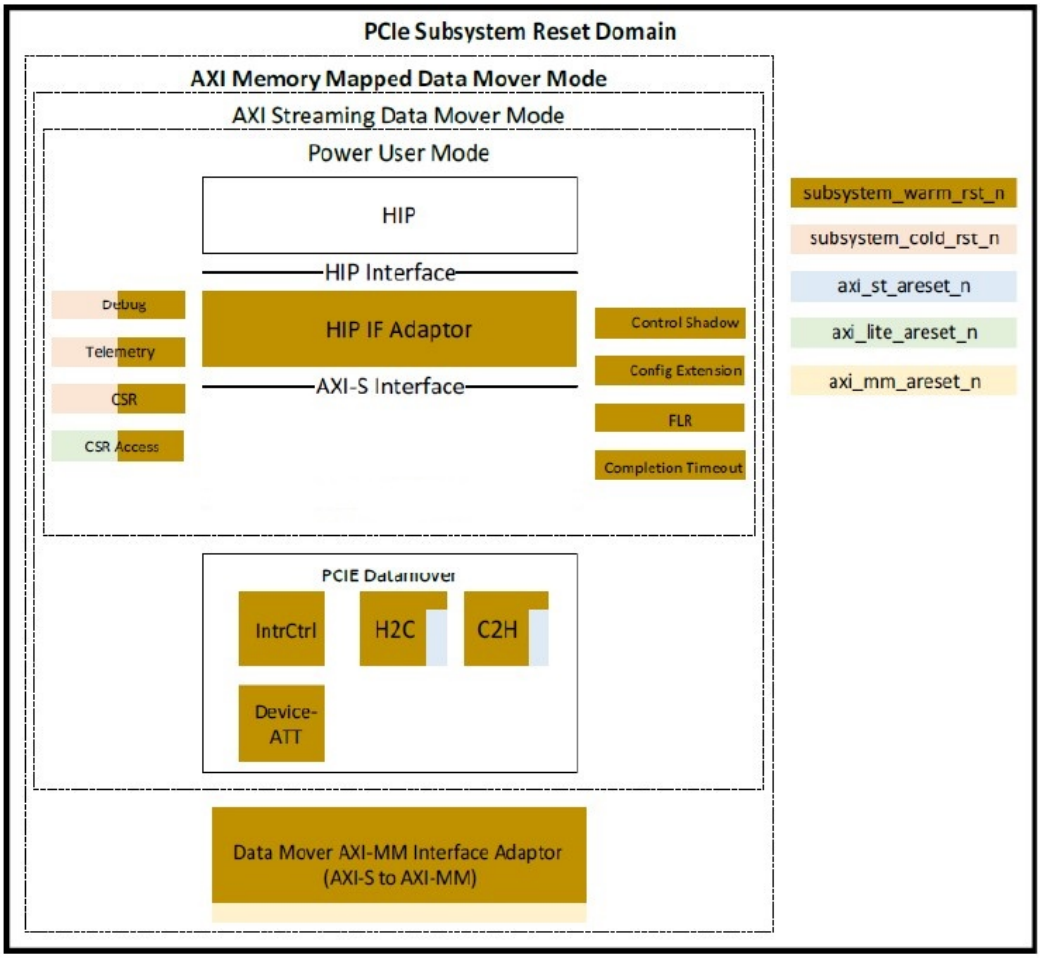
The figure below shows reset domains in PCIe Subsystem design in AXI-ST Data Mover mode

Figure 14. Reset Domains in AXI-ST Data Mover mode



The figure below shows reset domains in the PCIe Subsystem design in AXI-MM Data Mover Mode

Figure 15. Reset Domains in AXI-MM Data Mover Mode



AXI-MM Data Mover Mode will be available in a future release of Quartus

The list below specifies reset sequencing handshake requirement. The user must implement the reset sequencer in application user logic and follow the assertion and deassertion sequence for graceful entry and exit from reset.

Reset assertion:

- Assertion happens concurrently for all.

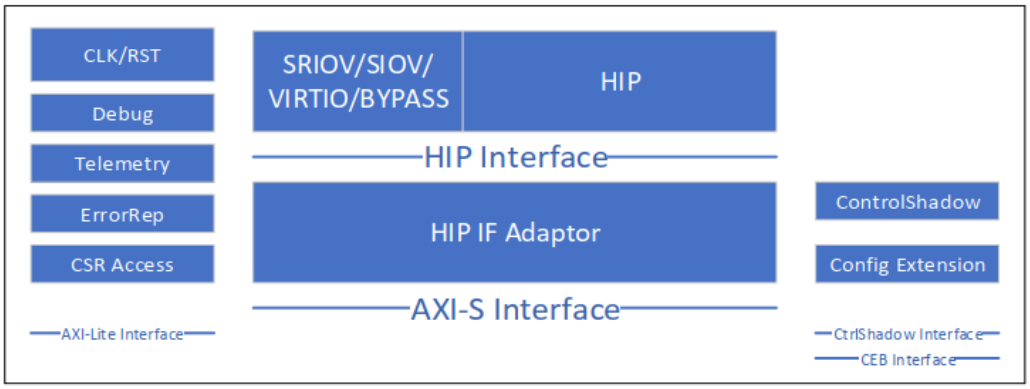
Reset deassertion:

- Cold reset
  - Req/Rdy handshake is used for graceful reset entry and exit.
- Warm reset
  - Req/Rdy handshake is used for graceful reset entry and exit.
- AXI4-Lite reset
- AXI-ST/MM reset

4.2. Power User mode

The following figure shows the block diagram of the PCIe Subsystem in Power User mode.

Figure 16. Basic Power User mode Block Diagram



On transmit side, the Power User mode forwards TLPs received from application to the Link. Your user logic is responsible for constructing TLPs as per PCIe rules, and for implementing credit management logic using the credit interface provided by HIP. The tag allocation and management are also done by your user logic.

On receive side, the Power User mode sends TLPs received from Link to user side with some additional information like BAR number and function number. Apart from forwarding received TLPs, additional side interfaces are provided for error reporting, reading, and writing registers in Hard IP and reset handshake.

This mode also provides basic telemetry and debug functionality blocks.

The following table shows the various profiles available when using the Power User functional mode.

Table 18. Power User Functional Mode Profiles

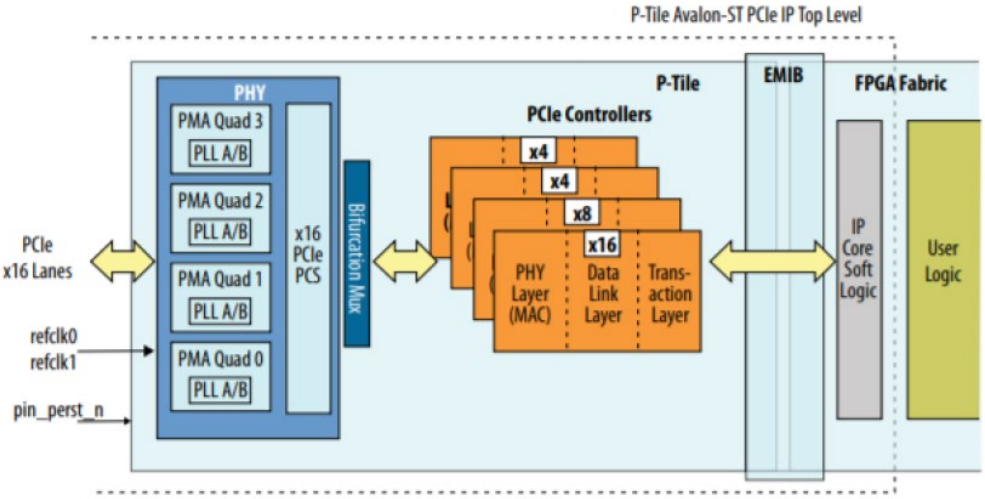
Functional Mode	Profile	Features
Power User mode	Basic	• One PF in an endpoint • One 64-bit BAR of 64 kB • AER enabled • Max payload size of 512 bytes • Maximum read request size (MRRS)=max supported by tile • Tags=max supported by tile

	Basic +	All features from Basic Profile plus: • VirtIO PCIe Capability present • FLR Enabled • MSI-X with 256 vectors
	Virtual	• Two Physical Functions (PF) in an Endpoint • PF0 has no Virtual Functions (Supervisory Role) • 128 Virtual Functions per PF • AER Enabled • FLR Enabled • All functions have one 64-bit BAR of 64Kbyte • MSI-X enabled with 4 vectors per function • ATS, TPH, PASID capabilities Enabled • Maximum Payload size is 512 Bytes • MRRS=Max Supported by Tile • Tags = Max Supported by Tile
	Virtual+	All features from Virtual profile plus: • VirtIO PCI Capability Present • Four PFs in an endpoint

4.2.1. PCIe Hard IP (HIP)

The PCIe Hard IP implements the functionality of the PCIe protocol. The HIP implements Physical, Data Link and Transaction Layers of the protocol. The HIP handles link training, DLLP exchanges, credit handling, BAR decode, and error handling in normal mode. It also implements SRIOV functionality for handling virtualization. The HIP's main data path interface is the Avalon Streaming interface.

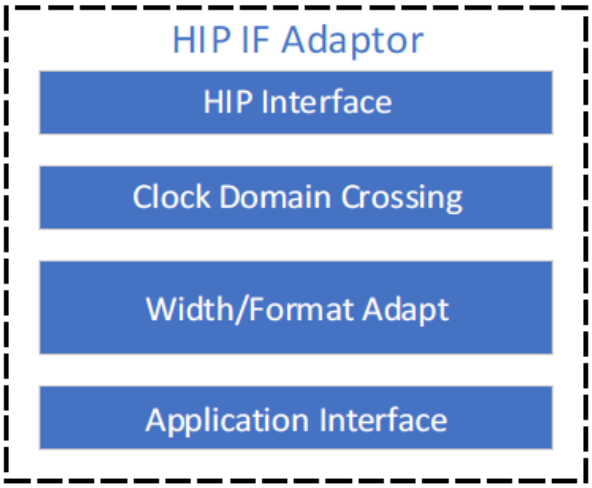
Figure 18. PCIe Hard IP in P-Tile



4.2.2. HIP Interface (IF) Adaptor

The Hard IP (HIP) interfaces with the HIP IF Adaptor in the PCIe Subsystem. The HIP IF adaptor acts as an interface between the HIP and the downstream logic. The HIP IF Adaptor provides a standardized interface to the downstream logic by performing the required width and format adaptation depending on the tile's AVST and sideband interfaces. The clock domain crossing module will allow downstream logic to run at different frequencies

Figure 19. Hard IP IF Adaptor



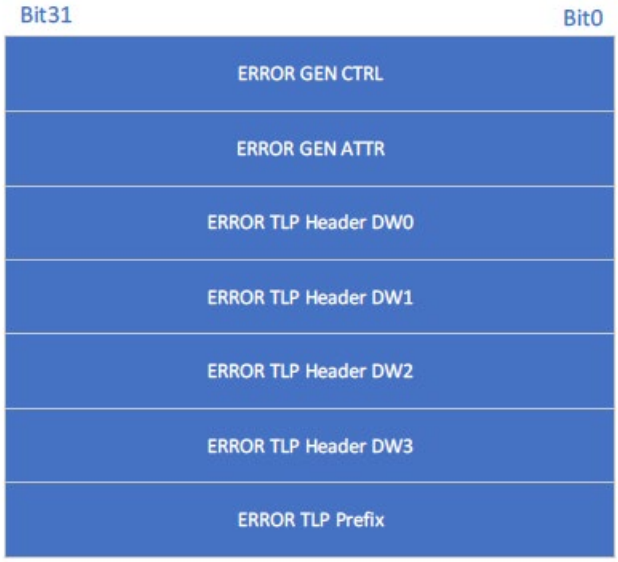
4.2.3. Application Error Reporting

The Subsystem implements Application Error Reporting registers to allow users to indicate various errors. The Subsystem logic then forwards this error information to the HIP block. The HIP block then responds to the assertion of these error bits by performing the following:

- Logging the status in the error reporting registers of the Function
- Sending error messages as per Basic Error Reporting policies or as per AER policies.

The figure below shows registers implemented in Subsystem register space for error reporting. Refer to Chapter 6 Register Descriptions for more details on the AER registers

Figure 20. Error Reporting



Note: In current Quartus release, VF related errors detected in application layer cannot be logged in HIP status registers as HIP's Application Error Interface doesn't provide error handling down to VF granularity. Additionally, the Hard IP Reconfiguration Interface doesn't provide access to set Error Status registers. In summary, VF related errors reported through the Application Error Reporting registers will not have any effect in Power User mode.

4.2.4. Debug Toolkit and Hard IP (HIP) Reconfiguration Interface

The PCIe Subsystem instantiates the Debug Toolkit module to provide debug functionality. The Debug Toolkit can access the link related debug information from the Hard IP (using Hard IP reconfiguration interface) available as tabs on the Debug Toolkit's graphical user interface. The Debug Toolkit can also access the Subsystem's soft register space for control and status information per core (e.g., core_x16, core_x8, core_x4, core_x4) through the system console

The Debug Toolkit provides the following features:

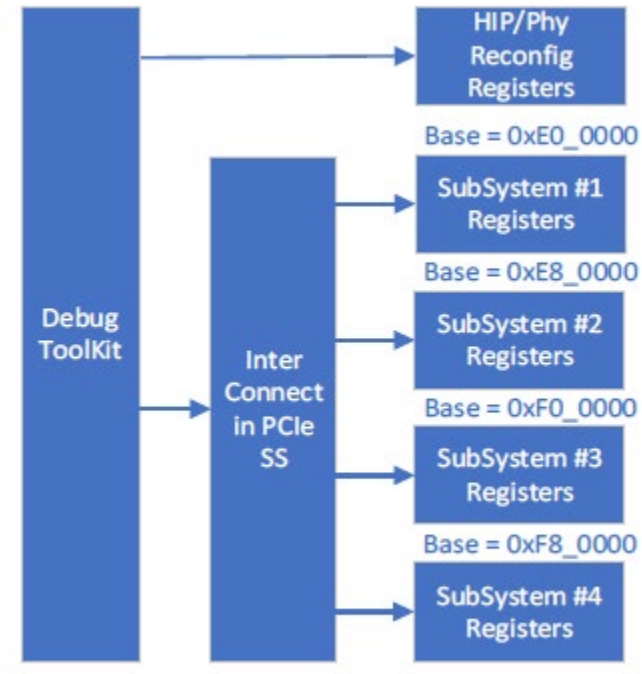
- Real time monitoring of physical layer
- View of Protocol and Link status information
- View of PLL and per-channel status of link
- Indicates presence of a re-timer connected between link partners
- Basic and advance debugging capabilities including PMA register access and Eye margining capability
- Subsystem's soft register space for control and status information using system console

Refer to the P-tile Avalon Streaming Intel FPGA IP for PCI Express User Guide for more details on the Hard IP registers that can be accessed using the Debug Toolkit.

To access the Subsystem's soft control and status registers, you must assign a value of '111' on address [23:21]. For the lower 20-bit address offsets, use the address map defined in *Chapter 6. Register Descriptions* to the respective PCIe Subsystem's base address above. The Subsystem's soft register space for each core will start from the base addresses below:

- PCIe Subsystem Instance #1 (core_x16) - 0xE0_0000
- PCIe Subsystem Instance #2 (core_x8) - 0xE8_0000
- PCIe Subsystem Instance #3 (core_x4) - 0xF0_0000
- PCIe Subsystem Instance #4 (core_x4) - 0xF8_0000

Figure 21. Debug Toolkit and Reconfiguration Interfaces



4.2.5. Configuration Space Extension

The HIP implements mandatory PCI and PCIe capabilities. The PCIe Subsystem (SS) provides the Configuration Extension Bus (CEB) interface to extend the configuration capabilities available in a Subsystem's protocol stack HIP block.

- The configuration TLPs with a destination address not matching with internally implemented registers are routed to the configuration extension interface.
- The application is responsible for returning data on read.
- The application returns zero if a transaction targets unimplemented address space.
- The write access to unimplemented address is dropped by application.
- Maximum one outstanding read request is allowed.
- The next pointer field of the last capability structure within HIP is set by the external capability pointer parameter.
    - Separate parameters are provided for PCI Compatible Region of Physical Function (PF) and Virtual Function (VF).

- Separate parameters are provided for PCIe extended capability region of Physical Function (PF) and Virtual Function (VF).

- The Subsystem implements timeout mechanism for request issued on CEB interface.
  - The timeout value is configurable, and user can set this value during compilation.
  - The Subsystem sends completion back to host with "SC" status and data as all zeros in case application failed to return data before timeout counter expires.

Note: The CEB interface and the CII interface are mutually exclusive. Hence, both cannot be enabled at the same time.

4.2.6. Control Shadow

The control shadow interface is used to bring out the settings of the various configuration register fields of the function. These fields are often required in designing the control path of the application layer logic. The application logic decodes information provided on this interface to create a shadow copy. The interface provides updates for primary control signals only. The application logic must read extra information required through the lite_csr interface by reading the configuration register of interest.

4.2.7. Configuration Intercept Interface

The Configuration Intercept Interface (CII) allows the application logic to detect the occurrence of a Configuration (CFG) request on the link and to modify its behavior. The application logic should detect the CFG request at the assertion of ss_app_st_ciireq_tvalid on the ss_app_st_ciireq* interface.

The application logic can use the CII to:

- Delay the processing of a CFG request by the controller. This allows the application to perform any housekeeping task first. This can be achieved by withholding the assertion of app_ss_st_ciireq_tready.
- Overwrite the data payload of a CfgWr request. The application logic can also overwrite the data payload of a CfgRd completion TLP. This can be achieved using the app_ss_st_ciiresp* interface.

Note: This interface is provided so that PCIe SS is backward compatible to legacy application logic that relies on CII for their functionality. Newly defined application logic should avoid using the CII interface and move to the CEB interface.

Note: The CEB interface and the CII interface are mutually exclusive. Hence, both cannot be enabled at the same time.

4.2.8. Power Management

The power management register allows users to manage power management messages. The generation of power management messages based on user input is not supported. The Subsystem exposes this register only if base hardware supports this capability.

4.2.9. Legacy Interrupt

The Legacy Interrupt register controls generation of assert and deassert messages. This register allows user to generate legacy interrupt through a side band interface instead of sending it over the main band AXI-ST interface.

4.2.10. Credit Handling

The Power User mode exposes link partner credit to user in Transmit direction. The credits will be advertised as limit value specified in PCIe spec. Apart from AXI Streaming ready-valid handshake user must check the availability of credits for transmitting the TLP.

The Receive side of Power User mode will operate on AXI Streaming ready-valid handshake.

4.2.11. Completion Timeout

The Subsystem communicates completion timeout events to the application logic through the dedicated interface.

4.2.12. Transaction Ordering

The Power User mode doesn't have separate receiving queues to handle PCIe transaction ordering or prevent deadlocks. The application logic needs to ensure the transactions adhere to PCIe ordering rules that prevent deadlocks, namely:

- Allow posted writes to pass blocked read transactions.
- Allow posted writes to pass blocked configuration write transactions.
- Allow completion to pass blocked read transactions.
- Allow completion to pass blocked configuration write transactions.

4.2.13. Page Request Service (PRS) Events

The Page Request Service (PRS) Control register allows PCIe Subsystem to generate events on the HIP’s PRS interface. This register allows user logic to generate these PRS events through a side band interface.

4.2.14. TX Non-Posted Metering Requirement on Application

PCIe HIP implements a finite number of RX Completion Buffers for its header and data. However, in endpoint mode, it advertises infinite credit to the host. Hence in Power User mode, application logic needs to implement metering logic on its TX Non-Posted requests such that it doesn't issue more requests than allowed to avoid overflowing the completion buffer. Refer to the Appendix sections for detailed completion buffer sizes for each different tile.

4.2.15. MSI Pending

The MSI Pending registers (MSI PENDING CTRL and MSI PENDING) allow application layer to indicate their MSI Pending bits to the HIP

4.2.16. D-State Status

The D-State Status register (D-State STS) allows application to read the D-State value of each function from the HIP

4.2.17. Configuration Retry Status Enable

The application layer can use the Configuration Retry Control register (CFG RETRY CTRL) to update the per PF Configuration Retry Status Enable controls (CRS En Controls) driven to the HIP. All VFs will share the same control as their parent PF. When the corresponding PF's CRS En Control is asserted, HIP will respond to Configuration TLPs with a CRS (Configuration Retry Status) if it has not already responded to a Configuration TLP with non-CRS status since the last reset. User can use this to hold off on enumeration.

4.2.18. Device Feature List (DFL) Vendor Specific Extended Capability

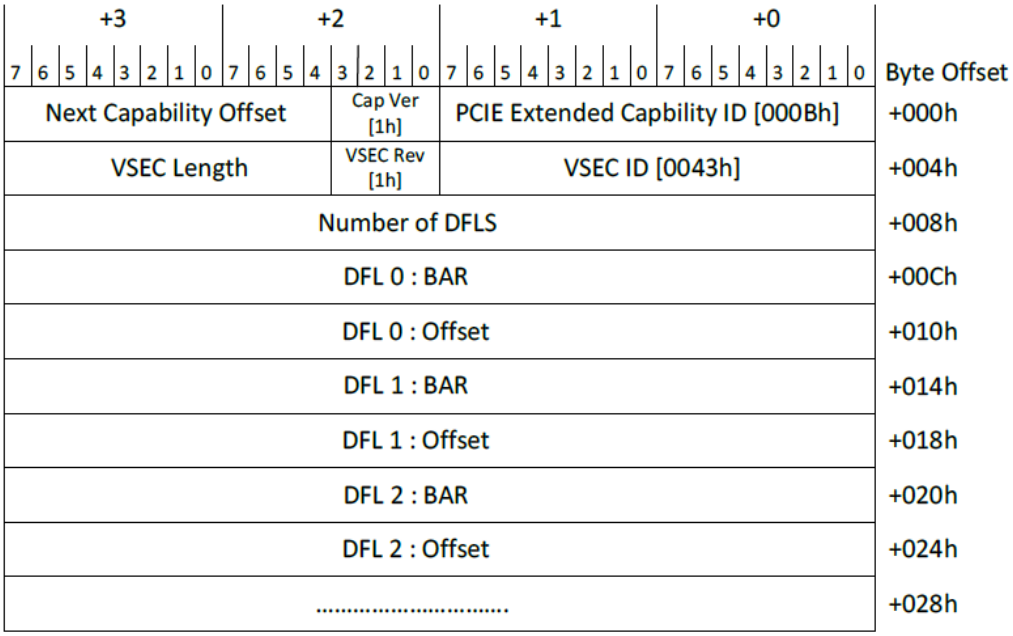
Applications can use these interfaces to configure, enumerate, open and access FPGA accelerators on platforms which implement the DFL in the device memory. Besides this, the DFL framework enables system level management functions such as FPGA reconfiguration.

Device Feature List (DFL) defines a linked list of feature headers within the device MMIO space to provide an extensible way of adding features. Software can walk through these predefined data structures to enumerate FPGA features.

The DFL Vendor Specific Extended Capability (VSEC) capability allows software to identify how many DFLs are implemented in a MMIO region of physical function '0' and their location. To specify location, DFL VSEC capability implements two 32-bit registers per DFL in a design. The first register indicates BAR number, and second register indicates offset within that BAR from where DFL structure starts.

The design will implement DFL VSEC capability with fields as shown in the figure below:

Figure 22. DFL VSEC Capability



Note: In case MSI-X BAR and DFL BAR are the same, the Offset value should be chosen in such a way that it does not overlap with the offset of MSI-X and the Offset value must be greater than MSI-X Offset + MSI-X Size.

The table below describes the register bit definition of fields in DFL VSEC capability. The attributes defined in this table are based on PCIe configuration read/write transactions.

When registers in this capability are accessed via AXI Lite interface, all the register bits must be treated as R/W.

Table 19. Device Feature List Vendor Specific Capabilities

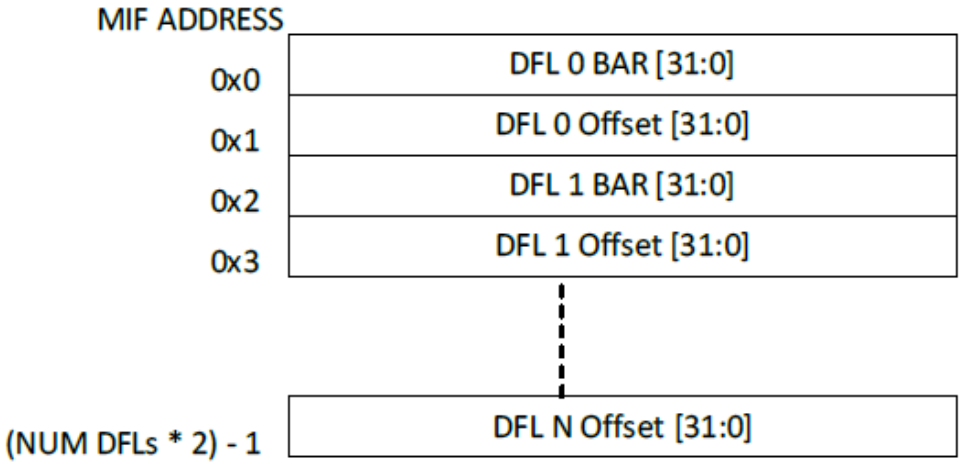
Register	Field	Bit Location	Description	Attributes
Vendor Specific Extended Capability Header	PCI Express Extended Capability ID	15:0	This field is set to 000Bh to indicate its vendor specific extended capability	RO
	Capability Version	19:16	This field is set to 1h	RO
	Next Capability Offset	31:20	Set by GUI parameter if CEB is enabled	RO

			else Set by design automatically. It points to the next capability in the linked list. Set to Null/Zero if it is the last capability in the linked list Default: Null	
Vendor Specific Header	VSEC ID	15:0	This field is set to 0043h	RO
	VSEC Rev	19:16	This field is set to 1h	RO
	VSEC Length	31:20	This field indicates the number of bytes in the entire VSEC structure, including the Vendor-Specific Extended Capability Header, the Vendor-Specific Header, and the vendor-specific registers. Set to 12+(8* Number of DFLS)	
Vendor Specific Register – Number of DFLS	Number of DFLS	31:0	Indicates Number of DFL structures in a design. Set by GUI parameter	RO
Vendor Specific Register – DFL BAR	DFL N BAR (N = 0,1, 2...)	31:0	The DFL BAR indicates MMIO BAR in physical function '0' where DFL structure resides. The number of DFL BAR registers equals to Number of DFLS These registers are implemented using MLAB/M20K The value of this field is initialized using Memory Initialization File (MIF).	RO

Vendor Specific Register – DFL Offset	DFL N Offset (N = 0,1, 2...)	31:0	The DFL Offset indicates offset within BAR (indicated by DFL N BAR) from where DFL structure starts. The number of DFL Offset registers equals to Number of DFLS These registers are implemented using MLAB/M20K The value of this field is initialized using Memory initialization File.	RO

The figure below shows the layout of MIF entries for DFL BAR and DFL Offset registers.

Figure 23. MIF Entries for DFL BAE and DFL Offset Registers



4.2.19. AXI-Streaming Interface

The Subsystem uses an AXI4 Streaming interface for transporting header and data information. The header is presented in line with data instead of a separate interface. The PF Number, VF Number, BAR number and Prefix information are presented in line with data. The PCIe Header and these side signals are grouped as a 32-byte header on the AXI Streaming interface.

4.2.19.1. Header Format

The table below lists header fields, their byte positions and bit positions on the Tdata bus.

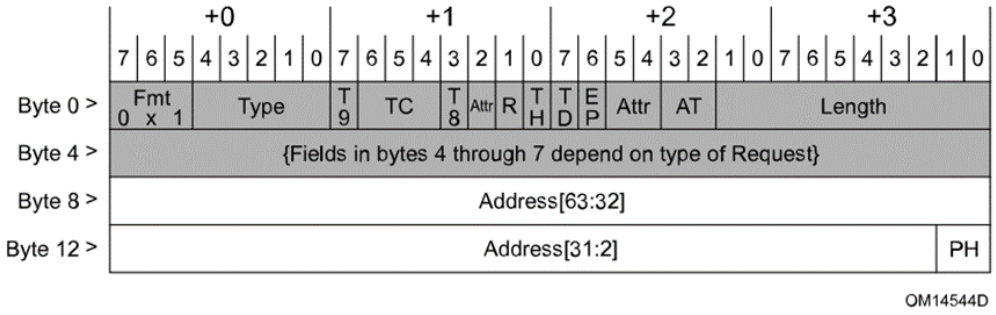
Table 20. Power User mode Header Format

Tdata Header Byte Index	Header Fields	Bits	Tdata Bit Position End	Tdata Bit Position Start
Byte 15 - Byte 0	PCIe Header	128	127	0
Byte 19 - Byte 16	Prefix	24	151	128
	Prefix Type	5	156	152
	Prefix Present	1	157	157
	Reserved	2	159	158
Byte 23 - Byte 20	PF Number	3	162	160
	VF Number	11	173	163
	VF Active	1	174	174
	BAR number	4	178	175
	Slot number	5	183	179
	Reserved*	8	191	184
Byte 31 - Byte 24	Reserved	64	255	192

*__Note__: Bits [186:185] are reserved for future use

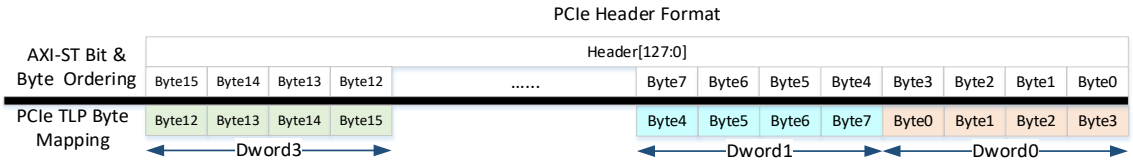
The figure below shows a standard PCIe header format.

Figure 24. PCIe Header for Memory TLP with 64Bit Addressing (4DW Header)



The PCI specification standard header format is mapped to an AXI4-Streaming Tdata interface as shown in figure below:

Figure 25. 4DW PCIe Header Mapping on Tdata Bus



The 3DW PCIe Header will be mapped to AXI-ST Tdata bit[95:0], the bit[127:96] are considered don't care in this case.

Figure . PCIe Header for Memory TLP with 64Bit Addressing (3DW Header)

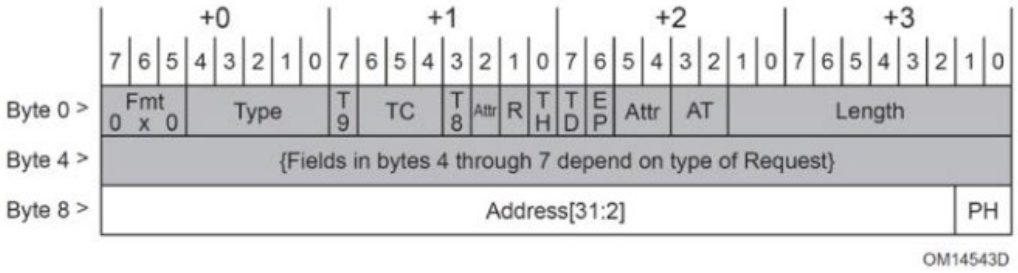
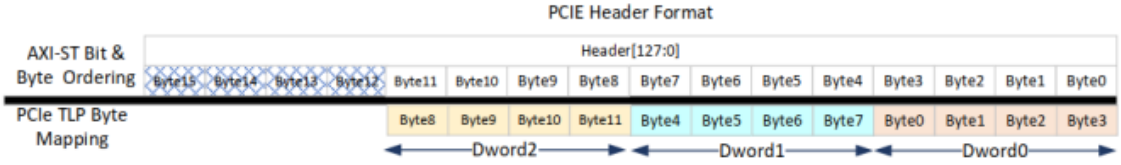


Figure . 3DW PCIe Header Mapping on Tdata Bus



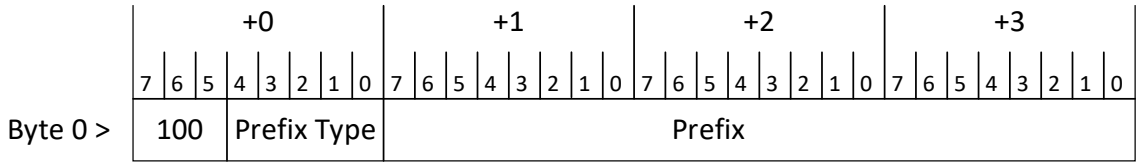
4.2.19.2. Prefix Format

The Prefix and Prefix Type fields carry the prefix information of the current TLP. The Prefix Present field indicates presence of prefix information. The Prefix Present field '0' indicates current TLP has no prefix associated with it. It is recommended to drive zero on Prefix and Prefix Type fields when Prefix Present is equal to '0'.

Note: The design supports one DW prefix per TLP in the current release.

The figure below shows PCIe Prefix TLP Format.

Figure 26. PCIe TLP Prefix



The table below shows the mapping of PCIe TLP Prefix shown in the figure above on to the Tdata bus.

Table 21. PCIe TLP Prefix Mapping to Tdata Header Field

PCIe TLP Prefix Bytes	AXI-ST Tdata Header Byte Index
Prefix - Byte 3	Byte 16
Prefix - Byte 2	Byte 17
Prefix - Byte 1	Byte 18
Prefix Type - Byte 0 [Bits 4:0]	Byte 19 [Bits 4:0]

4.2.19.3. Function Number Format

The PF Number, VF Number and VF Active fields specify the function number of a TLP. The VF Active high indicates the request/completion is targeting a virtual function. The VF Active low indicates the request/completion is targeting a physical function in a device.

The Slot number indicates which endpoint in a design the transaction is targeting. The slot number field is always zero when the Switch functionality is not present in the design.

The definitions of these function fields apply not only to the header but also for all occurrences in this document, for example on interface pins, register fields, and register names.

Table 22. Function Number Field Description

Function Number Fields	Description
Slot Number	Indicates target slot number of received TLP
PF Number	Indicates target physical function number of received TLP
VF Number	Indicates target virtual function number of received TLP
VF Active	When asserted high, indicates access is for Virtual Function. When low, indicates access is for Physical Function

4.2.19.4. BAR Number Format

The BAR Number indicates matching BAR for MMIO transaction coming from HOST side.

Table 23. BAR Number Field Description

BAR Number	Description
0000	BAR 0 (when configured as 32-bit BAR), or BAR 0-1 (when configured as 64-bit BAR)
0001	BAR 1 (when configured as 32-bit BAR); reserved when BAR 1 is combined with BAR 0 to form a 64-bit BAR
0010	BAR 2 (when configured as 32-bit BAR), or BAR 2-3 (when configured as 64-bit BAR)
0011	BAR 3 (when configured as 32-bit BAR); reserved when BAR 2 is combined with BAR 3 to form a 64-bit BAR
0100	BAR 4 (when configured as 32-bit BAR), or BAR 4-5 (when configured as 64-bit BAR)
0101	BAR 5 (when configured as 32-bit BAR); reserved when BAR 4 is combined with BAR 5 to form a 64-bit BAR
0110	Expansion ROM BAR
all others	Reserved

4.2.19.5. Data and Header Packing Schemes

The Power User mode supports the following packing schemes – simple

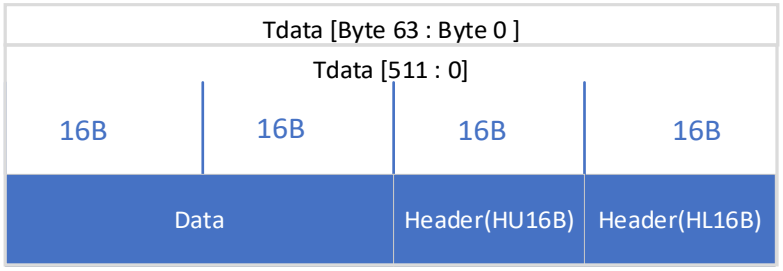
4.2.19.5.1. Simple Data and Header Packing Scheme [Inline Header]

The simple packing scheme inserts a header starting at a fixed location. The header always starts from Byte Index 0. The rule of header starting on Byte Index 0 constrains the design to send one packet per cycle.

The figure below shows the mapping of header and data on 512-bits wide Tdata bus.

Figure 27. Simple packing scheme with 512-bit Tdata bus

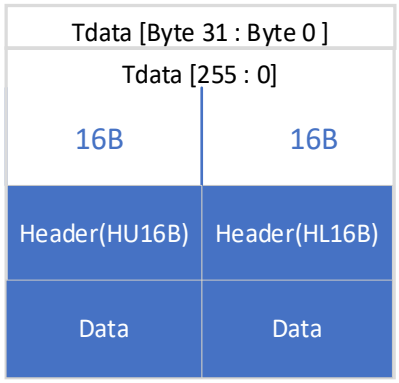
- HL16B - Header's Lower 16 bytes
- HU16B - Header's Upper 16 bytes



The figure below shows the mapping of header and data on 256-bits wide Tdata bus.

Figure 28. Simple packing scheme with 256-bit Tdata bus

- HL16B - Header's Lower 16 bytes
- HU16B - Header's Upper 16 bytes



The figure below shows some packing examples with the above restriction applied:

1st Command with Data - Payload 16 Bytes

2nd Command with Data - Payload 64 Bytes

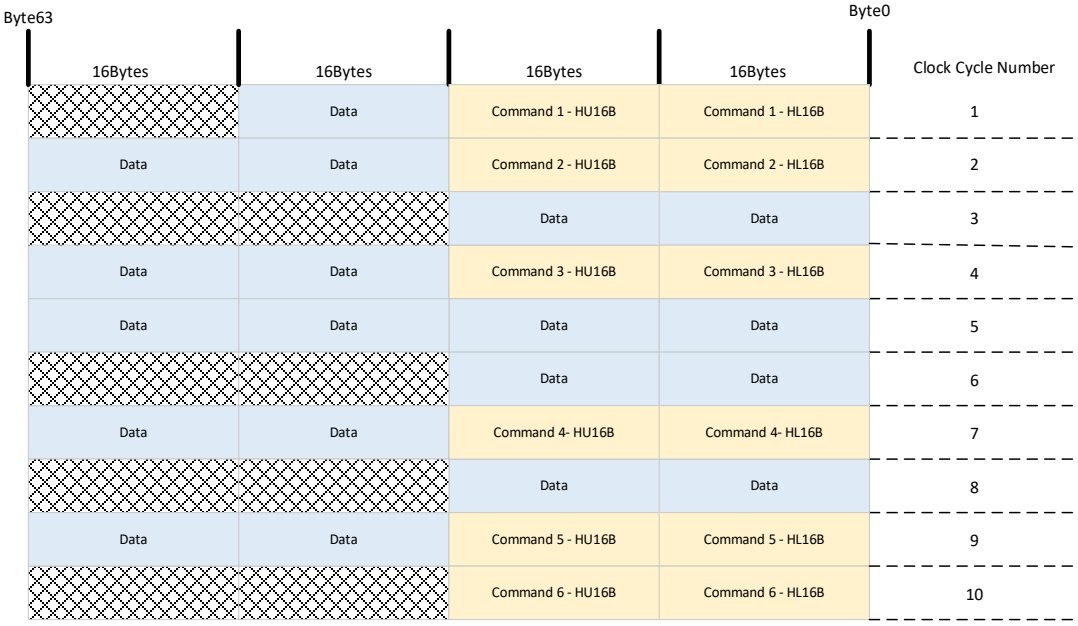
3rd Command with Data - Payload 128 Bytes

4th Command with Data - Payload 60 Bytes

5th Command with Data - Payload 32 Bytes

6th Command without Data

Figure 29. Simple Packing Scheme



The Tdata bus width will vary based on link width and link speed to meet specific bandwidth requirement. The table below lists width and fmax requirements to meet the link bandwidth goal.

Table 24. Simple Packing Data width and Fmax Options

Mode	Tdata Width Bits	Freq (MHz)	Number of Streams
Gen3X8 / Gen4X4	256	350	1
Gen3X16 / Gen4X8 (Gen4x8x8)	512	350	1
Gen4X16 (512-bit, 256 bit)	1024	400	1

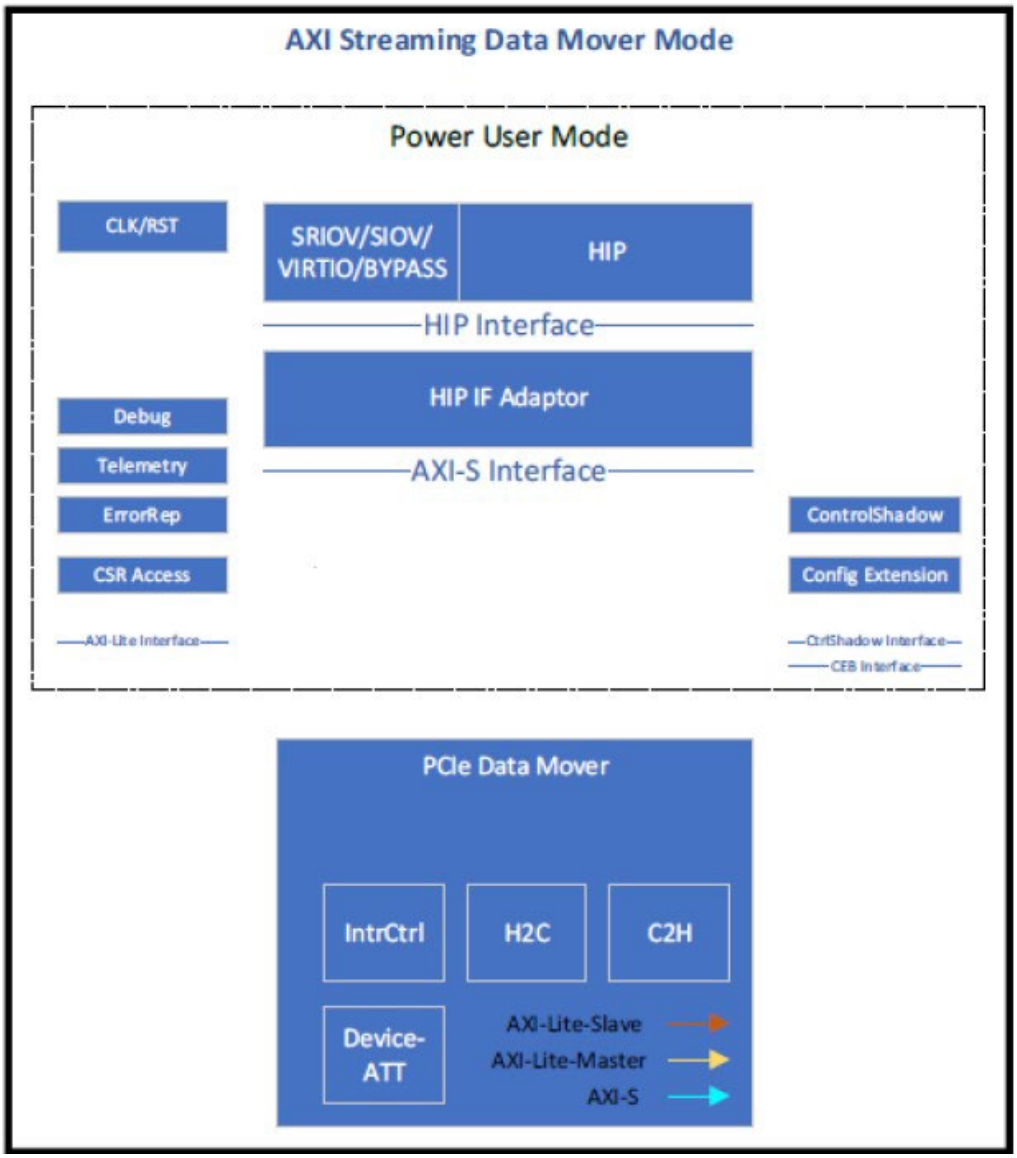
4.3. AXI Data Mover Mode

The AXI4 Data Mover Mode is built on top of Power User mode. This mode helps you implement the application logic without worrying about PCIe TLP rules. The Data Mover block constructs TLPs as per PCIe rules and is responsible for credit management and tag management. This mode allows user to transfer 16 MB size of data. The configuration extension, debugging and managing multiple streaming interfaces functionality remain the same as in Power User mode.

The optional AXI-MM interface allows data transfer using AXI-MM protocol. The Data Mover mode also implements optional MSI-X table supporting up to 4K vectors in total.

The figure below shows a block diagram of the AXI Data Mover Mode with an AXI Streaming Interface.

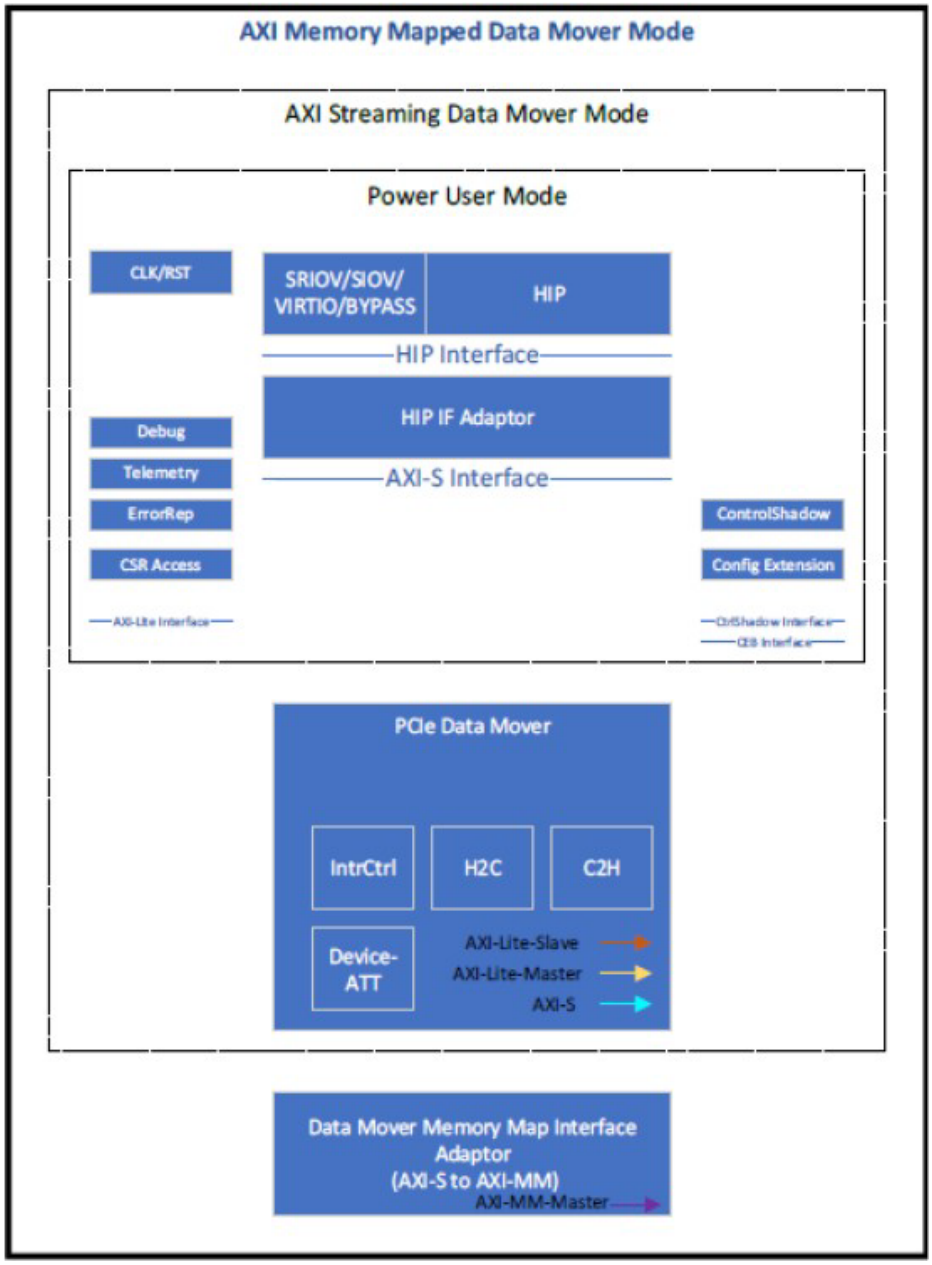
Figure 34. AXI Data Mover Mode with AXI Streaming Interface Block Diagram



The figure below shows block diagram of AXI Data Mover Mode with AXI Memory-Mapped Interface.

Note: The AXI-MM interface may be available in a future release of the Quartus Prime software.

Figure 35. AXI Data Mover Mode with AXI Memory-Mapped Interface Block Diagram



The table below lists the AXI-ST Data Mover mode's profiles and features enabled in each profile. This mode adds additional capabilities on top of features provided by Power User mode. The AXI-MM Data Mover will have the same feature set as the AXI-ST Data Mover mode. This mode will add the additional functionality of moving data over an AXI-MM interface instead of AXI-ST interface.

Table 27. Data Mover Mode Functional Mode Profiles

Functional Mode	Profile	Features
AXI-ST Data Mover mode	Basic	All features from Power User mode Basic Profile plus following additions: -PCIe Data Mover Block for Host to Card and Card to Host Data Transfer
	Basic+	All features from AXI-ST Data Mover mode Basic Profile plus following additions: -FLR Enabled -MSI-X Table -VIRTIO PCI Capability Present -Device Address Translation Table
	Virtual	All features from Power User mode Virtual Profile plus following additions: -PCIe Data Mover Block for Host to Card and Card to Host Data Transfer
	Virtual+	All features from AXI-ST Data Mover mode Virtual Profile plus following additions: -4 PF in an Endpoint -MSI-X Table -VIRTIO PCI Capability Present -Device Address Translation Table

4.3.1. Card to Host (C2H)

The C2H block handles transactions coming from application logic. This block provides two separate streaming interfaces (st_tx and st_txreq). The st_txreq interface is restricted to Data Mover read (DMRd) and Data Mover Interrupt (DMIntr) commands only. This additional interface allows these commands to pass a pending Posted transaction with huge payload on the st_tx interface. On the other hand, the st_tx interface allows transactions with both Power User and Data Mover header format.

For transactions with Power User header format, the application logic is responsible for ensuring that all PCIe TLP rules are met first. For transactions with Data Mover header format, this block will additionally process the transactions from application logic prior to sending them to the link. The DMRd transaction received from application is divided into multiple read TLPs based on MRRS (PF0) set by HOST. The Data Mover Write (DMWr) transaction received from application is divided into

multiple write TLPs based on MPS (PF0) set by HOST. The attributes provided with requests are applied to all TLPs generated for that request. This block ensures that all TLP fields are set as per PCIe TLP rules.

Note: Data Mover will only use MRRS/MPS value from PF0 for the TLP processing above. If different PFs have different MRRS/MPS values, it is required that PF0's value is
set to cater to all other PFs requirement.

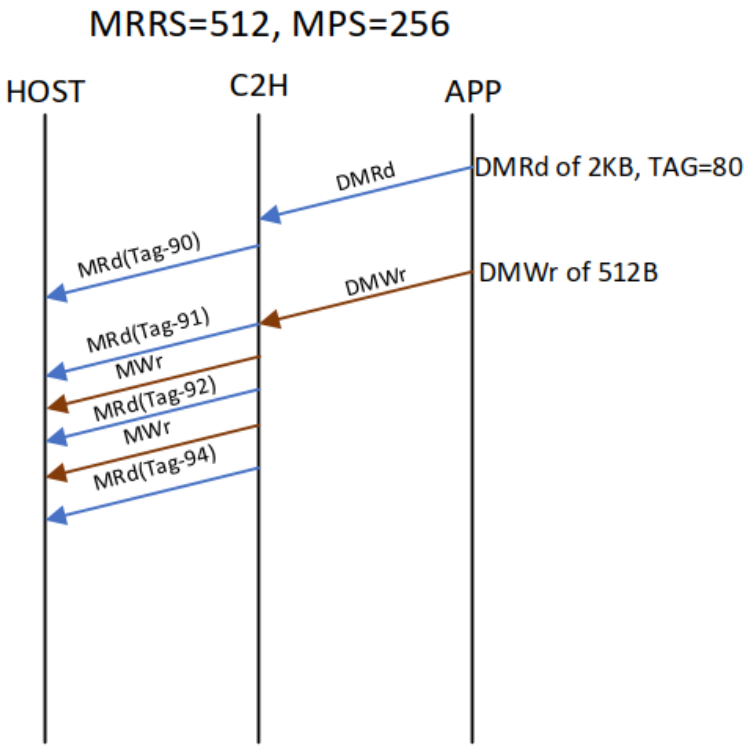
Note: If PF0 is being FLR-ed, the MRRS value (prior to FLR) will continue to be used by all other functions until PF0's MRRS is being programmed again after FLR is complete.
The C2H block also checks that enough credits are available for transmission before transmitting any TLP towards HOST. This applies to transactions with both Power User and Data Mover header formats.

Note: If the application logic has concern on the non-posted packet queued behind a large DMWr when st_txreq interface is not enabled, the application logic is required to split the DMWr to smaller packets and have non-posted packets sent in between.

The C2H block also checks enough credits are available for transmission before transmitting any TLP towards HOST. This applies to transactions with both power user and data mover header format.

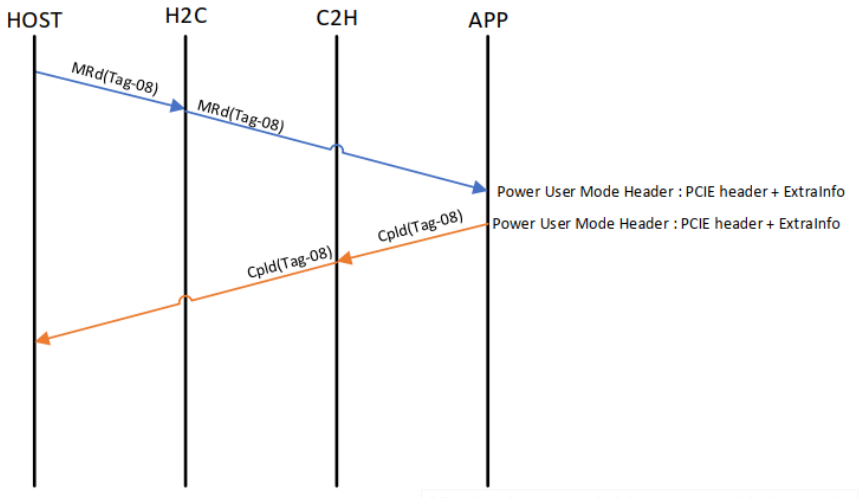
The figure below shows the splitting of incoming application request into multiple requests towards HOST:

Figure 36. C2H Command Splitting Example



The figure below shows the memory read from HOST being forwarded to the application using a PCIe Header. The application sends a completion back in response to this read command using another PCIe Header. The C2H block forwards this completion to HOST without any modification.

Figure 37. C2H Completion



As a requester the C2H block supports commands listed in Table 16. The C2H block translates
DMRd, DMWr and DMIntr commands to PCIe commands before sending them to HOST.

Table 28. C2H Commands as a Requester

Commands Supported as a Requester
DMRd
DMWr
DMIntr
PCIe AtomicOp
PCIe Messages

Commands supported as requester	Description
DMRd	Data Mover read. The DMRd command will read data from HOST memory specified by Host Address and write it to local FPGA memory specified by Local Address.
DMWr	Data Mover write. The DMWr command will read data from local FPGA memory specified by Local Address and writes it to HOST memory specified by Host Address. The application must not send data associated with DMWr command on streaming interface when MM Mode bit is set. The data Mover will drop this data
DMIntr	Data Mover interrupt
DMCpl	Data Mover completion. Completion associated with a Data Mover read
PCIe AtomicOp	PCIe AtomicOps commands as described in the spec
PCIe Messages	PCIe messages as described in the spec

As a completer the C2H block supports commands listed in Table 28

Table 29. C2H Commands as a Completer

Commands Supported as a Completer
PCIe Cpl/CplD
PCIe Messages

The application must avoid sending any other commands as a requester/completer. The application must ensure all PCIe TLP rules are met when sending PCIe commands using power user header format.

4.3.1.1. C2H PF/Slot Segment Distributions

For card to host transactions, when multi segment is enabled, the application must send the

traffic based on the PF or Slot number mapping to the segment number. The physical number or slot number will be mapped in ascending order to segment number and will roll over at the beginning of segment number back to segment zero. For example, in a scenario of 8 PF functions endpoint with 4 segment support, PF0 and PF4 will map to Segment 0, PF1 and PF5 will map to Segment 1, PF2 and PF6 will map to Segment 2 and PF3 and PF7 will then map to segment 3. If SRIOV is enabled, the VF transactions will map accordingly to its corresponding PF number.

4.3.2. Host2Card (H2C)

The H2C block forwards all types of requests from HOST to the application on a single AXI streaming interface. The requests coming from HOST are sent on this interface using the Power User mode header format. The completions received from HOST are checked for correctness before being sent to the application. Every completion received from HOST is checked against an entry in the completion tracker. The block inserts the associated function number and meta data information in the completion header.

As a requester the H2C block supports commands listed in Table 29:

Table 30. H2C Commands as a Requester

Commands Supported as a Completer
PCIe MMIO
PCIe AtomicOp
PCIe Messages

As a completer the H2C block supports commands listed in Table 19. The H2C block translates PCIe Cpl/CplD received from HOST to DMCpl before sending it to the application. Exception is given when the application issued Non-Posted request using Power User header format (in the C2H direction) whereby the PCIe Cpl/CplD will be returned to the application instead.

Table 31. H2C Commands as a Completer

Commands Supported as a Completer
DMCpl
PCIe Messages

4.3.2.1. H2C PF/Slot Segment Distributions

When multiple streams are enabled, the PCIe SS can distribute the traffic based on the PF umber or Slots number for the embedded endpoint and switch the integrated embedded endpoint respectively. The PF number or slot number will be ascendingly map to segment number and will roll over at celling of segment number back to segment zero. Example of 8 PFs in an endpoint with 4 segment support, PF0 and PF4 will map to Segment 0, PF1 and PF5 will map to Segment 1, PF2 and PF6 will map to Segment 2 and PF3 and PF7 will then map to segment 3. If SRIOV is enabled, the VF transactions will map accordingly to its corresponding PF number.

4.3.2.2. Completion Reordering

The Data Mover block supports the option to enable/disable the reordering buffer:

- The Subsystem (SS) will monitor the occupancy of reordering buffer apart from NP credits before
  sending any read request towards Host
- The C2H module can generate multiple requests towards HOST for a single request from application. It is possible that the completer in response sends completions for these requests out of order. When Completion Reorder Buffer is enabled, the H2C block arranges all these completions in order before sending them back to the application.
- The SS starts sending completion out on RX streaming interface as soon as the first set of data for the read request is received from the Host.
- When Completion Reorder Buffer is enabled, the downstream completions will be strictly returned in the order of the request coming from the application layer from the upstream, regardless of the transaction ID (requester ID + application tag)

  Note: Multiple applications that share the same Completion Reorder Buffer must take into consideration the potential QoS performance deterioration. If multiple applications have a QoS requirement, it is strongly recommended to have their own Completion Reorder Buffer in the applications, and not use the PCIe SS Completions Reorder Buffer.
- When Completion Reorder Buffer is enabled, the H2C block will return the entire completion as requested in the original "read request" issued by the application.

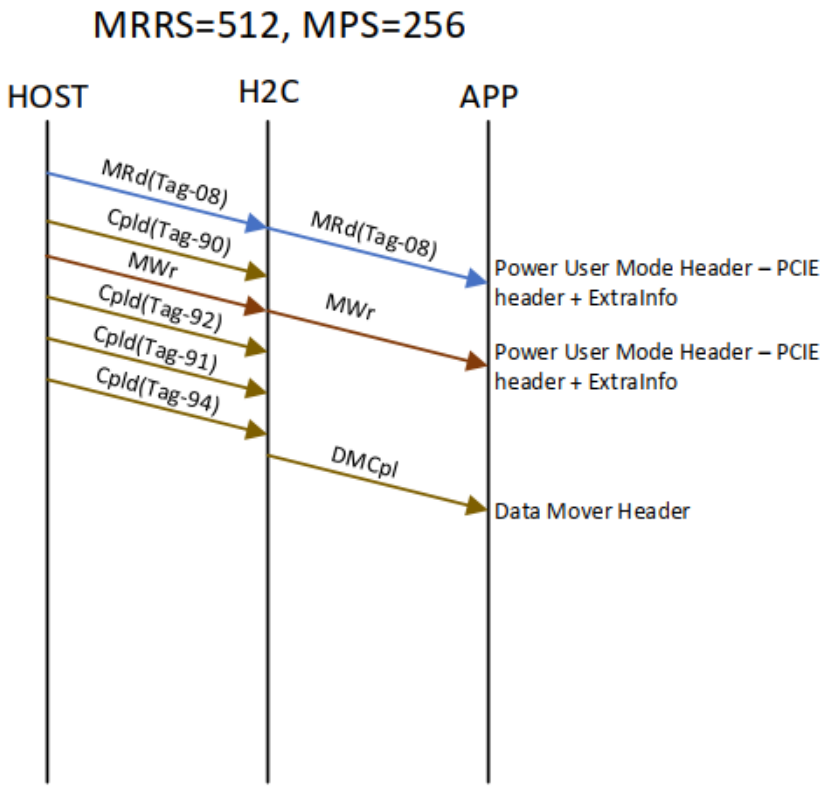
- When Completion Reorder Buffer is disabled, the PCIe SS will return the completions from the host without reordering them. The PCIe SS will remap the PCIe Tag to Application Tag accordingly. The application is required to use the Application Tag and lower address to reorder the completions.

  Note: The Expansion ROM transactions are expected to be completed during boot time and the large completions (greater than 4K bytes) are not expected during boot time.

The figure below shows:

- Sending requests from HOST in PU Mode Header format without modifying PCIe header to Application with Additional info like Function number, BAR number etc.
- H2C block reordering completion received from HOST before sending it to application

Figure 38. H2C Request and Completion Example



When reordering buffer feature is not present:

- The SS will monitor the occupancy of completion buffer in Hard IP before sending any read request towards Host.

- The H2C block sends DMCpl commands towards application in the same order in which it has received completions from the Host.
- The SS will start sending a completion as soon as it has received a completion from the Host.
- The completion size will be the same as the completion size received from the Host.

4.3.2.3. Completion Error Handling

In the scenario where H2C block doesn't receive completion from HOST before completion timeout timer expires, the H2C block informs application by sending completion timeout indication on the st_cplto interface (if enabled). The user can report an error through the application error reporting mechanism described in section 3.1.6.
However, if "Completion Timeout through AXI-ST" option is enabled instead, the H2C block is required to synthesize a completion packet with CA status for each completion timeout event. For completion timeout related to Non-Posted cycles with Power User header format, PCIe Cpl will be returned. For completion timeout related to Non-Posted cycles with Data Mover header format, DMCpl will be returned instead together with its corresponding metadata.

When reordering buffer is not present, the completion with UR/CA/Poisoned status received from the HOST is forwarded to application. When reordering buffer is enabled, there is a possibility that SS will experience a timeout/UR/CA error condition in the middle of a completion transaction. In this scenario:

- The SS will terminate current completion by asserting Tlast
- The SS will assert tuser abort signal for one clock along with tlast
- In case SS receives remaining completions for that same application request tag, the completions of subsequent completions received will be dropped.

  Note: The PCIe SS can only track the request that is split by the PCIe SS, hence, each of the requests with a unique tag number from the applications are tracked as a unique request. The PCIe SS can't associate multiple application tags as a single request from the application.
- The SS logic waits for either timeout or unsuccessful completion from Host to release internal tags associated with that request

4.3.2.4. Data Flow with MM Mode (Completion Handling)

The C2H block stores the MM mode flag and local address in the completion tracker structure when it sends a read request to the Host. The H2C block reads the completion tracker structure when it receives a completion from the Host. If the MM Mode bit is set for a given read request, the C2H block sends a write command to

the DM-MMIA block. The write data is a completion data received from the Host and the starting address of a write command is the address provided by a completion tracker structure.

Note: This is not supported in the current Quartus release

4.3.3. Interrupt Controller (IntCtrl)

The Interrupt Controller is an optional block in the Data Mover engine. The interrupt controller
implements an MSI-X Table and a PBA Table. The MSI-X table size is parameterizable and can be selected at compile time. The design will support a maximum of 4K vectors in total. By default, these 4K vectors will be distributed evenly across all functions.

Note: If only one PF and zero VF is selected, MSI-X table size is restricted to 2047 per PCIe Base Specification.

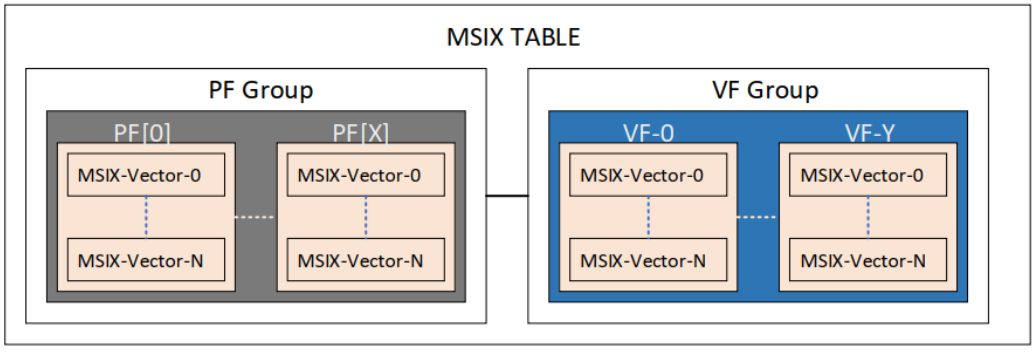
For DM interrupt that send through AXI-Lite Response interface or AXI-ST RX Request interface, the MSI Interrupt will not order against AXI-ST RX interface.

For MSI that send in through AXI-ST RX interface, the MSI will obey posted - posted ordering. PCIe SS will make sure that MSI posted will not overtake posted ahead of the MSI,
as well as will not permit subsequent posted to overtake MSI request.

4.3.3.1. MSI-X Vector Allocation

The figure below shows the allocation of MSI-X vectors and their organization in the internal memory. The "N" represents vectors per function and is the same across all functions in a design. The Physical Function and Virtual Function in Subsystem will be allocated with equal number of vectors. The vector allocation starts from left to right as shown in figure below. The vectors are allocated to physical functions first and then to virtual functions.

Figure 39. MSI-X Vector Allocation for Physical and Virtual Functions



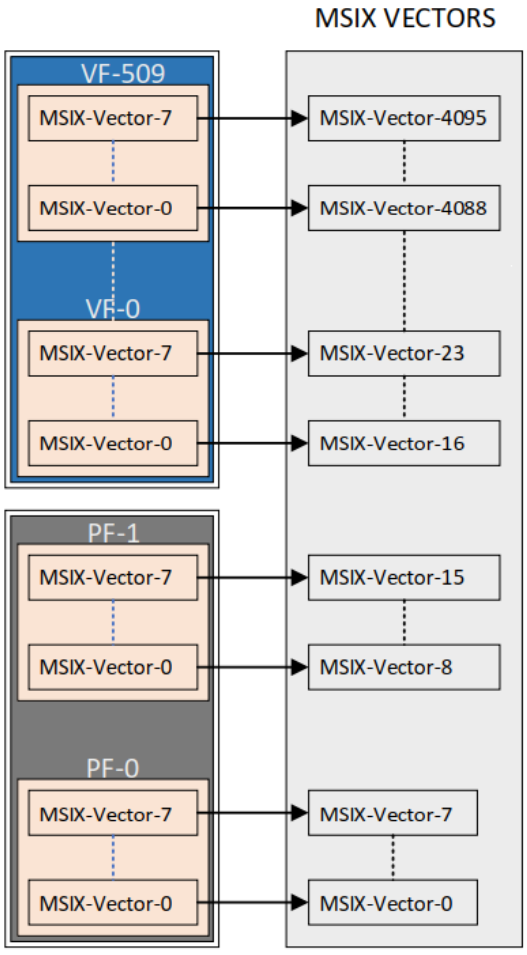
In the figure above PF[0]-PF[X] represent physical functions in the design. The VF[0]-VF[Y] represent virtual functions in the design.

Note: If MSI-X Vector Allocation Policy is set to "Dynamic", the value N will be set to the maximum supported value which is 2047. This feature is not supported in the current release

The figure below shows an example of the mapping of PF and VF vectors to MSI-X Vectors. In this example, the design has a total of 2 physical functions and 510 virtual functions. Every function requires 8 interrupt vectors.

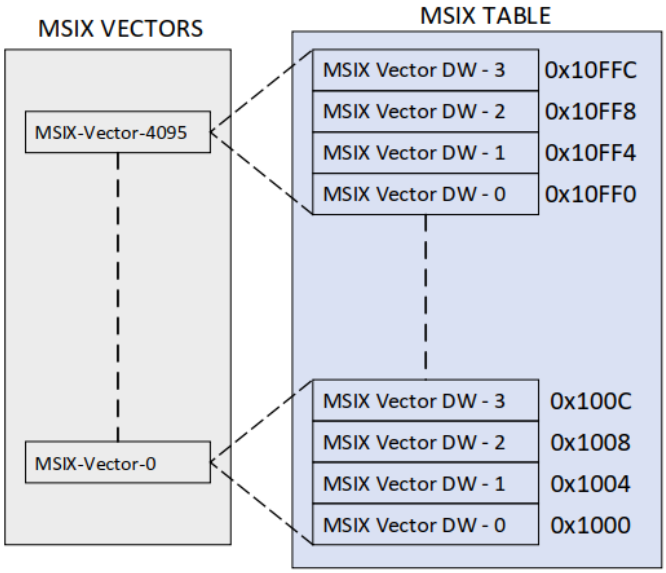
Note: In this example, the MSI-X Vector Allocation Policy is set to "Static."

Figure 40. Example MSI-X Vector Allocation



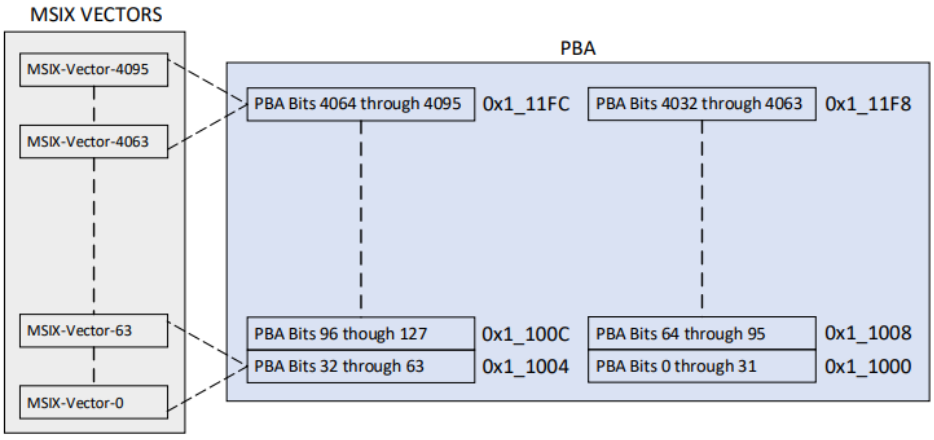
The figure below shows the MSI-X Vector mapping with MSI-X Table in the Subsystem. The user can access the vector information programmed by the Host using the mapping shown in figure below, through the AXI-Lite interface.

Figure 41. MSI-X Vector Mapping with MSI-X TABLE



The figure below shows the MSI-X Vectors mapping with MSI-X PBA in the Sub-system, one PBA bit per MSI-X Vector. The user can access the pending information through AXI-Lite Interface.

Figure 42. MSI-X Vector mapping with MSI-X PBA TABLE



4.3.3.2. MSI-X BAR Indicator, Offset and Size

The Subsystem will provide an option to select the MSI-X BAR indicator (BAR), offset, and table size. All the functions in the design will share the same value of these parameters. The user must choose a table size such that the overall table size does not cross 4096 vectors.

The default values of these parameters are:

- BAR Indicator – 5
- BAR Offset – 0
- Table Size – 2048

If the user selects an offset other than '0', then all the MMIO transactions with an address less than the offset will be forwarded to the application logic or to other BAR decoded ranges in the PCIe Subsystem.

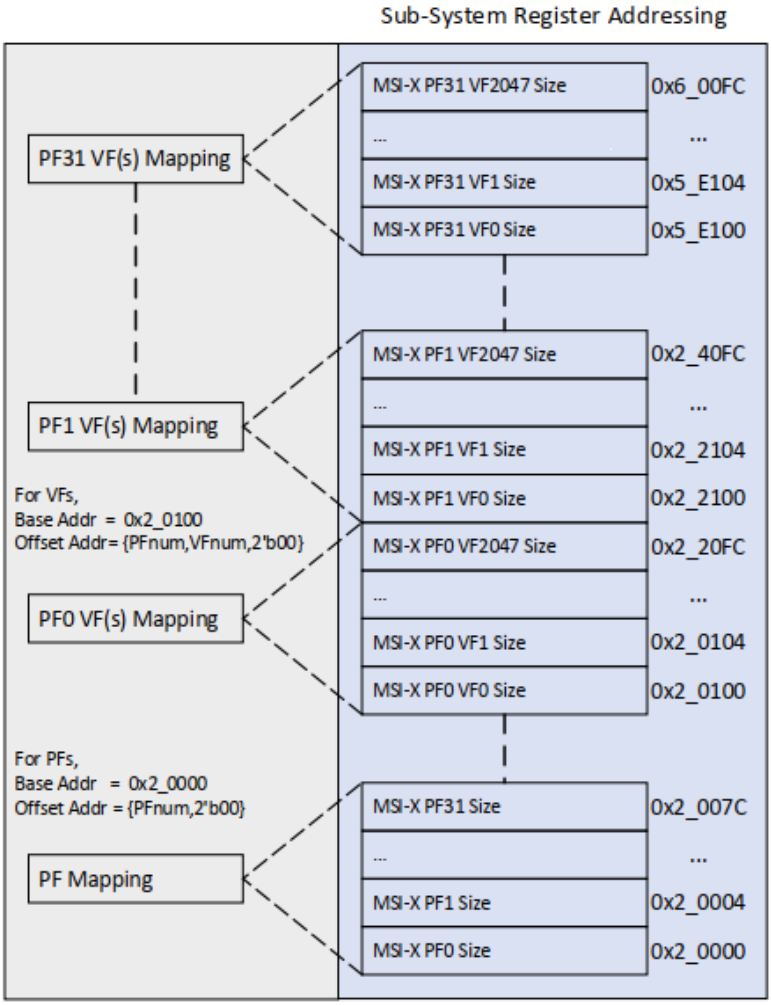
The Pending Bit Array (PBA) will follow immediately after MSI-X table.

Note: If MSI-X Vector Allocation Policy is set to "Dynamic", the Table Size for each function will be set to the maximum supported value which is 2048. However, the actual
implemented MSI-X vectors is still hardware limited to a value of 4096 for all PFs/VFs combined. The actual MSI-X vector(s) allocation to each PF/VF is dynamically managed using the MSI-X PF/VF dynamic sizing table registers by the management software.

4.3.3.3. MSI-X PF/VF Dynamic Sizing Table

When MSI-X Vector Allocation Policy is set to "Dynamic", the MSI-X Vector management
software needs to program the MSI-X PF/VF dynamic sizing table to indicate the number of MSI-X Vectors required for each PF/VF accordingly. The figure below shows the MSI-X PF/VF size registers w.r.t PCIe Subsystem's register addressing. These MSI-X PF/VF size registers are read/write accessible by the Host or by the application logic through AXI-Lite interface. They should be programmed correctly prior to enabling MSI-X feature.

Figure 43. MSI-X PF/VF Dynamic Sizing Table



Note: The address space for Virtual Function Size table is defined considering every physical function can have a maximum of 2048 virtual functions. The design will implement size table registers equal to the number of VFs attached to a particular PF parameter. For example, if "PF - 0" has 16 Virtual functions then the design will implement a table starting from address "0x2_0100" to "0x2_013F". The rest of the address space from "0x2_140" to "0x2_00FC" for PF0-VF is considered unimplemented. If the next PF (PF - 1) has VFs then the design will implement the next set of size registers starting from offset "0x2_2100."

4.3.3.4. MSI-X PF/VF Sizing Table Register Definition

When MSI-X Vector Allocation Policy is set to "Dynamic", this register is used to define the number of vectors needed per function. The Subsystem implements only one set of MSI-X PF/VF Sizing Table registers. It applies to EP0 only.

Note: Usage of these control registers are applicable only in Data Mover Mode
Default Value: 0x0000_0000

Table 32. MSI-X PF/VF Offset Map Register

Register Name	Bits	Attribute User Side	Description
MSI-X Size	11-0	RW	NUM_VECTOR Number of MSI-X vectors required for the function.
	31-12	RsvdZ	Reserved

4.3.3.5. MSI-X PF/VF Parameter Configurable Options

When MSI configuration mode is set to Parameter Configurable Options, the MSI distributions for PF / VF are exactly same as MSI-X PF/VF Dynamic Sizing Allocation, except instead of the allocations are program through registers, it is configurable through parameter.

4.3.3.6. Interrupt Triggering

The application can trigger an interrupt in two ways:

- DMIntr command over AXI streaming interface (both *st_tx* and *st_txreq*)
- Writing into MSIX GEN CTRL register using *lite_csr* interface

Note: DMIntr sent via *st_txreq* will not be ordered against other cycles on *st_tx*.

Note: Similarly, MSI-X triggered via *lite_csr* will not be ordered against other cycles on AXI-ST interface(s).

Note: MSI-X that has ordering requirements must not use *st_txreq* or *lite_csr* interface.

4.3.3.7. AXI Lite Triggering Model

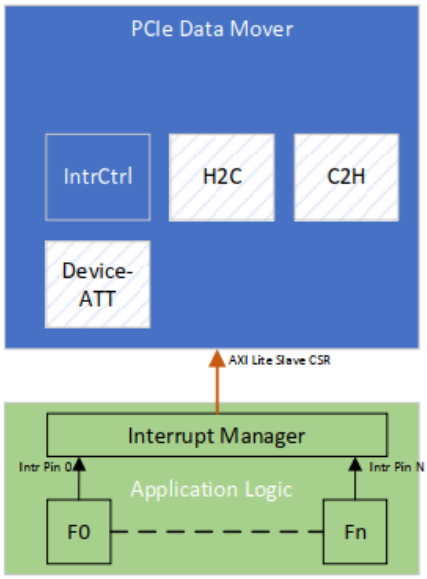
The recommended flow to generate MSI-X using "lite_csr" interface is as follows:

- Read MSIX GEN CTRL register bit[0]
- If bit[0] = '1' then go back to step 1 else
- Write the register with vector number of interrupt and set the bit[0]='1'

- The interrupt controller reads corresponding entry from MSI-X table and generates
  memory write transaction towards HOST
- The interrupt controller clears Bit[0] and vector number field.

The figure below shows how multiple interrupt sources can be connected to interrupt controller block in Data Mover

Figure 44. Interrupt Controller AXI Lite Usage Model



4.3.4. Device Address Translation Table (ATT)

The MMIO transactions generally target responder devices attached to AXI Fabric in the FPGA. The address map of these responder devices is not visible to HOST. These devices are in local address domain. The device address translation table (Device-ATT) is an optional block which provides address translation of MMIO addresses to local FPGA memory address.

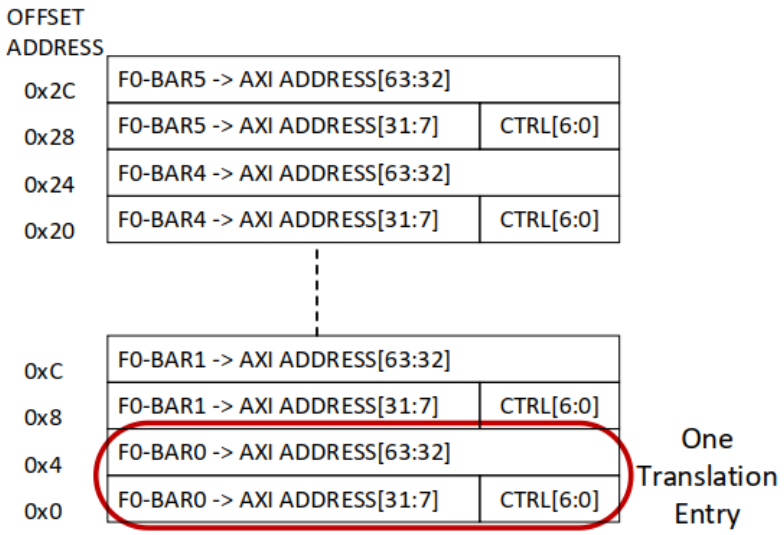
The block implements a set of registers in table form for transaction routing and optional address translation. The table is 32-bit wide. The two consecutive entries in a table format one translation entry of 64-bit AXI address. The lower seven bits of the AXI Address are assumed to be zero. These lower seven bits of translation entry controls translation functionality. One translation entry is mapped to one BAR in a function. BAR0-BAR5 are supported. The Device ATT will also support Expansion ROM BAR Decode. Refer to BAR Number Field Descriptions for details bar number mapping. When Device ATT is enabled, the application can use Device ATT to route the Posted/Non-Posted transactions to AXI-MM or AXI-ST respectively.

When Device ATT is disabled, the PCIe SS will default to route the Posted/Non-Posted transaction to the AXI-ST_RX_REQ interface.

Note: AXI-MM is limited to memory cycles within BAR or Expansion ROM BAR decoding, hence message is unable to detour to AXI-MM. Message can either continue to be routed to AXI-ST_RX_REQ (ordering does not maintain across AXI-MM initiator and AXIST_RX_REQ), or the parameter drops all the messages at the PCIe SS.

The figure below shows the table entry for Function-0 (F0).

Figure 45. Address Translation Entry



The table below lists the CTRL bits and their definitions.

Table 33. Device ATT Control Register

Translation Entry	Bit	Attribute User Side	Default Value	Description
CTRL	0	R/W	1	Entry Valid 1 - Indicates Entry is Valid 0 - Indicates Entry is Invalid
	1	R/W	0	Perform Translation 1 - Perform Address Translation 0 - No Translation Required
	3 - 2	R/W	01	MMIO Destination Interface 00 – Reserved (Not supported)

				01 - AXI Streaming 10 - AXI MM 11 - Reserved
	6 - 4	R/W	000	Reserved

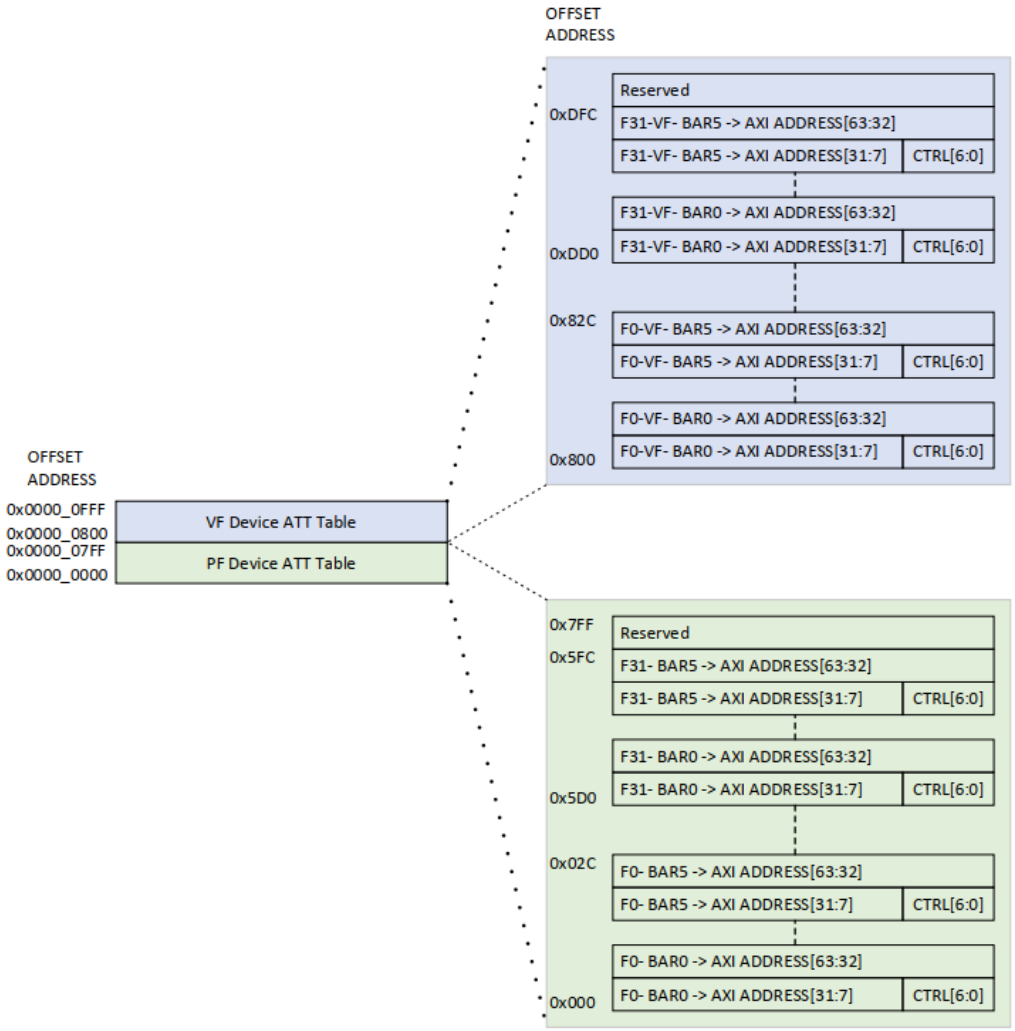
The Data Mover block drops MMIO write if Entry is marked invalid. The MMIO read in this case will return completion with "UR". The Data Mover block also reports an error via Power User mode blocks. The MMIO transactions can be routed to three different interfaces based on the CTRL[3:2] setting.

Note: If MMIO cycles are routed to "Destination Interface" other than the main data path interface, these MMIO cycles will no longer be ordered against other non-MMIO cycles. For example, completion(s) from Host (still on main data path) will not be able to push MMIO writes that get routed to "AXI-MM" interface since these are both orthogonal interfaces.

Note: The application must not program MMIO Destination Interface that is Reserved.
Otherwise, it is subject to undefined behavior.

The design will maintain a table for 32 physical functions and virtual functions associated with these physical functions in a design. The figure below shows the address map of the Device ATT table.

Figure 46. Address Map of Device ATT Table



The PF BAR address/Expansion ROM BAR translation calculation steps are:

1. Select MMIO address whose lower address bits are equal to BAR size.
2. Select PF entry/Expansion ROM entry from Device ATT table pointed to by BAR decode signal and PF Number.

   Note: Expansion ROM Base Address is sizeable between bits [31:11].

   Note: Host only allows to map Expansion ROM Base Address below 4GB range (address 31:0) as defined by the PCI Express Specification. Device ATT has the flexibility to map it to any memory range (64 bits address) within the Device ATT range.
3. Add '1' and '2' to form local AXI Address

The VF BAR address translation calculation steps are:

1. Select MMIO address whose lower address bits are equal to BAR size
2. Select VF entry from the Device ATT table pointed by to the BAR decode signal, PF Number and VF Number
3. Multiply BAR Size with VF number
4. Add '3' to '2' to form VF address offset in local AXI address space
5. Add '1' and '4' to form local AXI Address

The following requirement must be satisfied by SOC while deciding AXI Local address map for child VFs attached to parent PF. This requirement is based on VF address translation scheme implemented by Device ATT in PCIe Data Mover block.

- One child VF occupies address space equal to BAR size in AXI address space
- All Child VFs occupies contiguous address spaces in AXI address space
- First child VF starts at offset '0'
- The starting address of second and higher child VFs equals to previous child VF offset + BAR Size

4.3.5. Address Aligned Data

The address aligned data feature allows user logic to send and receive data (payload) aligned with byte address boundary on AXI streaming transmit/receive interface

The Subsystem expects valid starting byte of payload is aligned as per lower order address bits in a request. The DMWr and CplD commands in upstream direction on tx_st interface must be issued as per this requirement. In downstream direction (Host2Card-H2C) memory write transactions are modified to align data on rx_st interface based on lower address bits of memory write command.

When this feature is enabled, Subsystem aligns completion data on rx_st interface based on lower address field in DMCpl command. The lower address indicates starting byte position in current completion.

The AXI-ST interface width determines the number of lower order address bits used for
deciding starting byte position. The design uses log2 {AXI-ST Interface Width in Bytes} bits of lower address to decide starting byte position. Two separate parameters will be used to enable this feature: one for Data Mover related commands (DMWr and DMCpl) and one for MMIO write and completion for MMIO read.

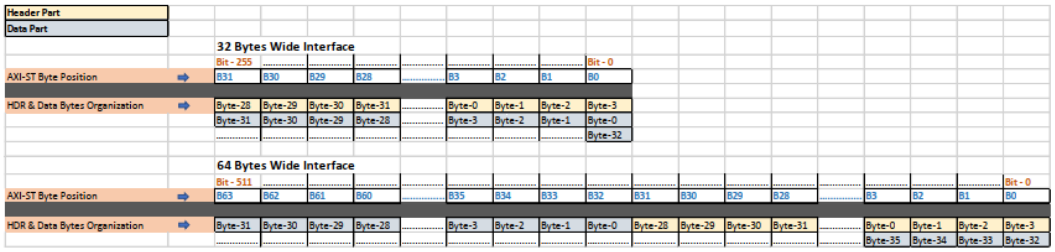
1. Parameter DM_ADR_ALIGN - Data Mover Commands

2. Parameter MMIO_ADR_ALIGN - MMIO Write and Completion of MMIO Read

Note: The ability to dynamically disable this feature for Data Mover commands may be available in a future release

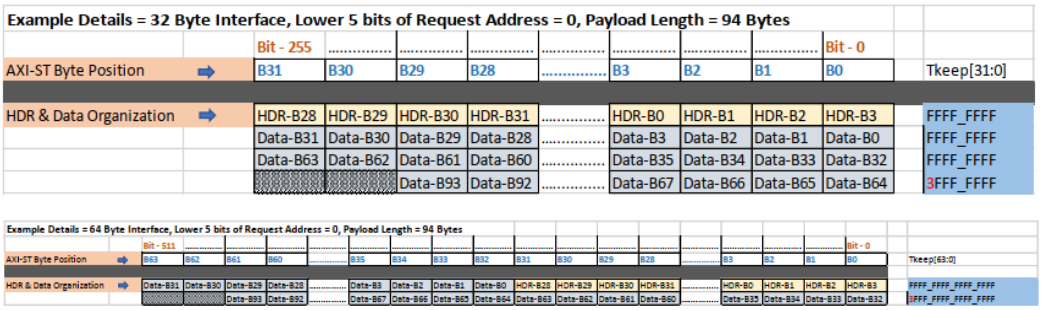
The following section describes the layout of AXI ST interface, header, and data part. The section will cover some examples to show how data will be aligned based on the address in a request.

Figure 47. Address-Aligned Data



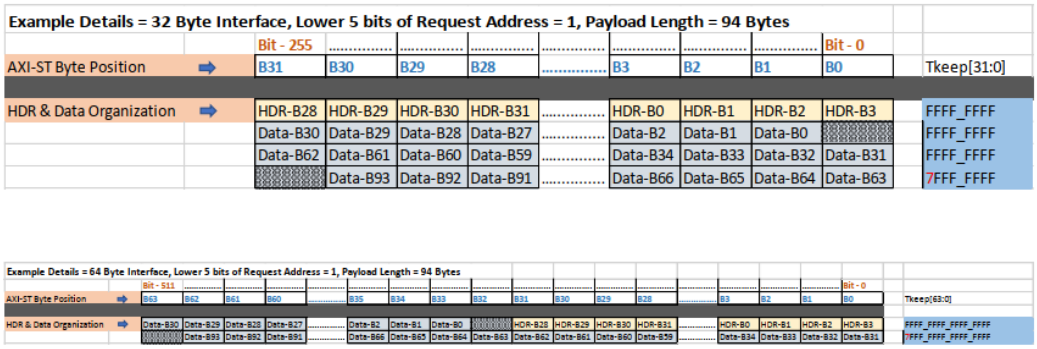
Note: Notice the position of the Byte-0 of Data Part, with 32-bytes wide interface it starts on byte 0 of AXI ST interface and with 64-bytes wide interface it starts on Byte 32 of AXI ST interface The Address-Aligned Data feature will consider Byte-0 of Data Part is aligned to address 0, Byte- 1 aligns to address 1 and so on. The example below shows data alignment when lower 5 bits of address is "00000b".

Figure 48. Example of Address-Aligned Data with Lower 5 Bits 0



The example below shows data alignment when the lower 5 bits of address are "00001b". Note the Data-B0 is starting on Byte-1 of Data Part.

Figure 49. Example of Address-Aligned Data with Lower 5 Bits = 1

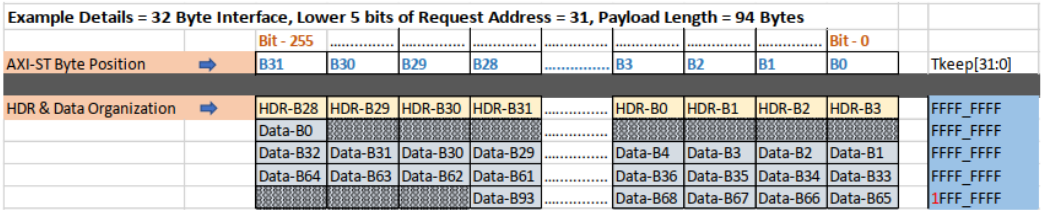


Example Details = 32 Byte Interface, Lower 5 bits of Request Address = 1, Payload Length = 94 Bytes									
	Bit - 255							Bit - 0	
AXI-ST Byte Position ➡	B31	B30	B29	B28	.....	B3	B2	B1	B0	Tkeep[31:0]
HDR & Data Organization ➡	HDR-B28	HDR-B29	HDR-B30	HDR-B31	.....	HDR-B0	HDR-B1	HDR-B2	HDR-B3	FFFF_FFFF
	Data-B30	Data-B29	Data-B28	Data-B27	.....	Data-B2	Data-B1	Data-B0		FFFF_FFFF
	Data-B62	Data-B61	Data-B60	Data-B59	.....	Data-B34	Data-B33	Data-B32	Data-B31	FFFF_FFFF
		Data-B93	Data-B92	Data-B91	.....	Data-B66	Data-B65	Data-B64	Data-B63	7FFF_FFFF

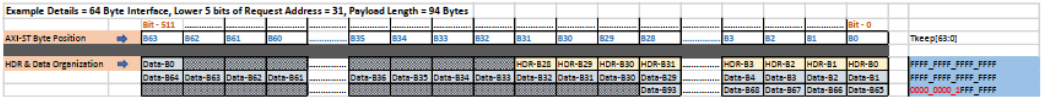
Example Details = 64 Byte Interface, Lower 5 bits of Request Address = 1, Payload Length = 94 Bytes																	
	Bit - 511															Bit - 0	
AXI-ST Byte Position ➡	B63	B62	B61	B60	.....	B35	B34	B33	B32	B31	B30	B29	B28	.....	B3	B2	B1	B0	Tkeep[63:0]
HDR & Data Organization ➡	Data-B30	Data-B29	Data-B28	Data-B27	.....	Data-B2	Data-B1	Data-B0		HDR-B28	HDR-B29	HDR-B30	HDR-B31	.....	HDR-B0	HDR-B1	HDR-B2	HDR-B3	FFFF_FFFF_FFFF_FFFF
	Data-B93	Data-B92	Data-B91	.....	Data-B66	Data-B65	Data-B64	Data-B63	Data-B62	Data-B61	Data-B60	Data-B59	.....	Data-B34	Data-B33	Data-B32	Data-B31	7FFF_FFFF_FFFF_FFFF	

The example below shows data alignment when the lower 5 bits of address are "11111b". Note the Data-B0 is starting on Byte-31 of Data Part

Figure 50. Example of Address-Aligned Data with Lower 5 Bits = 31



Example Details = 32 Byte Interface, Lower 5 bits of Request Address = 31, Payload Length = 94 Bytes									
	Bit - 255							Bit - 0	
AXI-ST Byte Position ➡	B31	B30	B29	B28	.....	B3	B2	B1	B0	Tkeep[31:0]
HDR & Data Organization ➡	HDR-B28	HDR-B29	HDR-B30	HDR-B31	.....	HDR-B0	HDR-B1	HDR-B2	HDR-B3	FFFF_FFFF
	Data-B0				.....					FFFF_FFFF
	Data-B32	Data-B31	Data-B30	Data-B29	.....	Data-B4	Data-B3	Data-B2	Data-B1	FFFF_FFFF
	Data-B64	Data-B63	Data-B62	Data-B61	.....	Data-B36	Data-B35	Data-B34	Data-B33	FFFF_FFFF
				Data-B93	.....	Data-B68	Data-B67	Data-B66	Data-B65	1FFF_FFFF



Example Details = 64 Byte Interface, Lower 5 bits of Request Address = 31, Payload Length = 94 Bytes																	
	Bit - 511															Bit - 0	
AXI-ST Byte Position ➡	B63	B62	B61	B60	.....	B35	B34	B33	B32	B31	B30	B29	B28	.....	B3	B2	B1	B0	Tkeep[63:0]
HDR & Data Organization ➡	Data-B0				.....					HDR-B28	HDR-B29	HDR-B30	HDR-B31	.....	HDR-B3	HDR-B2	HDR-B1	HDR-B0	FFFF_FFFF_FFFF_FFFF
	Data-B64	Data-B63	Data-B62	Data-B61	.....	Data-B36	Data-B35	Data-B34	Data-B33	Data-B32	Data-B31	Data-B30	Data-B29	.....	Data-B4	Data-B3	Data-B2	Data-B1	FFFF_FFFF_FFFF_FFFF
													Data-B93	.....	Data-B68	Data-B67	Data-B66	Data-B65	0000_0000_1FFF_FFFF

4.3.6. Flow Control Mechanisms

4.3.6.1. Tiles TX Interface Flow Control

At transmit interface, there are 2 mechanisms which control the packet transmission to HIP
interface:

1. The Metering logic -- Used to track RX Completion queues availability, prior to the non-posted transaction to be granted to the HIP interface, so that the completion
   space is guaranteed upon receiving the completions. The metering check is only applicable for
   upstream non-posted transactions that are targeting the egress port. When completion reorder is disabled, the metering logics check the HIP RX Completion

Queue Size. When completion
reorder is enabled, metering logics check the PCIe SS RX Completion queue size, with the assumption that the PCIE SS can always make forward progress of accepting completions without throttling the tvalid to HIP interface.

2. The HIP TX Flow control credit – The PCIe SS is checking the credit availability of the HIP TX, prior to granting the transactions respectively. This credit is applicable to all 3 streams (Posted, Non-Posted, Completions). On top of the PCIe Ordering requirement, the respective stream transactions cannot be sent to egress port if the respective stream does not meet the flow control requirement. Since there is no TX Flow control interface to application, in the event of PCIe SS Transmit queue filled up for any of the stream, PCIe SS will backpressure application through tready deassertion.

4.3.6.2. Tiles RX Interface Flow Control

At the receive interface, the PCIe SS implements the posted/non-posted flow control credit advertise to the HIP, so that the HIP uses the flow control to advertise the transactions to the PCIe SS. This will prevent overflowing of the posted/non-posted RX queues in the PCIe SS. There is no completion flow control back to the HIP interface (advertise as infinite completion credit), as there is metering logic at the TX side to pre-allocate the RX completion queues, hence the RX completions queue will never overflow. Although there are RX flow controls implemented in the PCIe SS, there are no RX flow controls at application interface, hence if application needs to backpressure, application will deassert tvalid indication. The tvalid deassertion should be temporary and application must make sure can make forward progress of posted, non-posted and completions
eventually, otherwise the tvalid deassertion may cause backpressure all the way to the HIP via unavailable credit.

Note: Blocking of posted transaction to make forward progress that back pressure to HIP, will
resultant HIP unable to make forward progress of completions as well, as there could be a
requirement of completions to push posted ordering.

4.3.6.3. Application TX Credit Flow Control

The implementation required a parameter that would allow to enable/disable the conditional instantiation of the TX Credit Flow Control Logic. Application is strongly encouraged to use the TX Credit Flow Control to avoid death lock conditions, unless applications have its maintenance mechanisms to prevent death lock conditions to occurs. Proprietary death lock avoidance is outside of the scope of the PCIe SS Data Mover implementation. When this parameter is enabled, PCIe SS Data Mover will advertise TX Flow Control Credit to Application. In the current version of PCIe SS, Data Mover will advertise data credit as infinite, and header credit of finite credit. The PCIe SS will ensure that the Data Buffers allocations are sufficient for max payload size of header credit for PCIe transactions, but the data buffer will not size for max size of DM write transactions as the DM write transactions can go as large as 16 MB payload size. Note that the Data Buffer are shared between DM Write and PCIe Write transactions, hence the data buffer is guaranteed not to overflow only when the application has sent all the PCIe Write. If DM Writes are in queue, the data

buffer may overflow for temporary period of time, and a global back pressure of tready will be deasserted to AXI-ST RX and AXI-ST RX Req interface. This event is temporary, until HIP make forward progress and drain the transactions.

4.3.6.4. Application RX Credit Flow Control

The implementation required a parameter that would allow to enable/disable the conditional instantiation of the RX Credit Flow Control Logic. When the parameter is enabled, the PCIe SS DM is required to track Application RX advertise credit, prior to sending the cycle to the applications. When completion reorder is enabled, the application must always advertise infinite completion credit to the PCIe SS Data Mover. Application can optionally advertise finite or infinite posted/non posted credits to the PCIe SS Data Mover. Note that the credit is managed per transaction base with assumptions that each header entry is capable to hold up to max payload size of the data
payload supported.

PCIe SS Data Mover RX Path to Applications, will not have arbiter to arbitrate the posted/non
posted requests, hence the PCIe SS Data Mover will manipulate the Application RX Credit Versus
the PCIe SS internal RX buffer credit, prior to advertise the credit to the Tiles RX interface. Only when both PCIe SS internal RX buffer and Application credit is available, the PCIe SS will advertise
the respective credit to HIP RX credit interface.

4.3.7. Data Mover Memory Map Interface Adaptor (DM-MMIA)

The PCIe Data Mover block provides ability to interface with AXI MM responder agents with this block. The module supports AXI MM initiator interface. The primary function of this block is to convert AXI streaming commands to AXI Memory-Mapped commands.

The read command from AXI streaming interface is converted to AXI-MM read command. The data returned on AXI-MM Read Data channel is sent back to requester as a completion command on AXI streaming interface. The write command from AXI streaming interface is converted to AXI-MM write command.

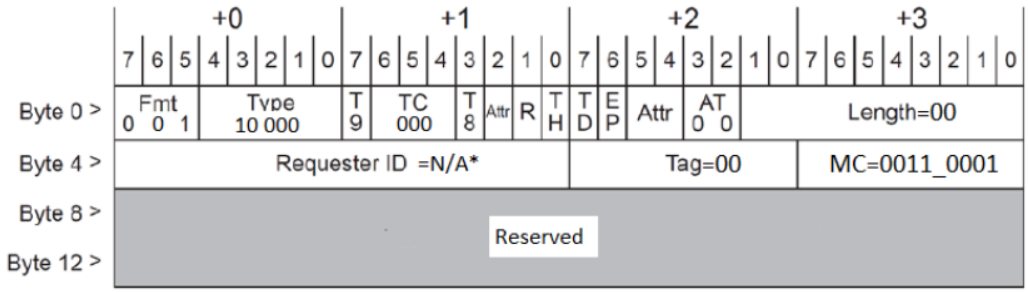
Note: The AXI-MM interface may be available in a future release of the Quartus Prime software.

4.3.8. Application Error Reporting

In Data mover mode, the Subsystem utilizes the same Application Error Reporting registers described in Section 2.1.4. However, there's an additional responsibility that the Subsystem needs to handle during Data Mover Mode. It needs to generate ERROR messages (as shown in Figure 42 below) for errors related to VF. These PCIe error messages are subjected to the standard PCIe Spec's baseline error reporting requirements.

Note: In the current Quartus release, VF related errors detected in application layer cannot be logged in HIP status registers as HIP's Application Error Interface doesn't provide error handling down to VF granularity. Additionally, the Hard IP Reconfiguration Interface doesn't provide access to set Error Status registers also. Hence VF related errors reported through the Application Error Reporting registers will not have any effect

Figure 51. PCIe Error Message (ERR_NONFATAL Message)



* Requester ID will be formed by HIP. PCIe SS needs to populate corresponding vf_active, vf_num and pf_num fields correctly in the header bus towards HIP.

4.3.9. Transaction Ordering

In Data Mover Mode, PCIe Subsystem will support the same PCIe transaction ordering rules that Power User mode supports, with below exceptions:

As there's an additional *st_txreq* interface for Data Mover Mode, PCIe Subsystem has additional rules to adhere to. It is important to allow Non-Posted cycles from the *st_txreq* interface to bypass a pending Posted cycle on the application's AXI interface (*st_tx*). This feature allows Non-Posted cycle and DMIntr cycle (on the *st_txreq* interface) to bypass a pending Posted cycle with huge data payload (e.g., 16MB), provided that the PCIe SS has yet to accept the cycles on the *st_tx* interface.

Note: A pending Posted cycle is defined as a cycle that has been started on the AXI interface (*st_tx*) but has not been completed yet (i.e., no *tlast* yet)

Note: If a transaction is started on the *st_tx* and *st_txreq* at the same time, the transaction on the *st_txreq* will be treated as the earlier transaction of the two. The same consideration applies when a Posted transaction completes on the *st_tx* interface at the same time as a Non-Posted transaction starts on the *st_txreq* interface.

Shown below is an example of 3 Posted cycles and 4 Non-Posted cycles entering PCIe SS Data Mover Mode through both *st_tx* and *st_txreq* interfaces.

Table 34: Example Transaction Ordering

*Ordering	P1	NP2	P3	NP4	P4	NP5	NP6
P1	*N/A*	Yes	Yes	Yes	Yes	Yes	Yes
NP2	No	*N/A*	Yes	Y/N	Yes	Y/N	Y/N
P3	No	Yes	*N/A*	Yes	Yes	Yes	Yes
NP4	No	Y/N	No	*N/A*	Yes	Y/N	Y/N
P4	No	Yes	No	Yes	*N/A*	Yes	Yes
NP5	No	Y/N	No	Y/N	Yes	*N/A*	Y/N
NP6	No	Y/N	No	Y/N	Yes	Y/N	*N/A*

*Number suffix shows the transactions' timing w.r.t the AXI interface. Transactions with smaller numbers indicates they arrive earlier compared to transactions with larger numbers.
Shown in the table below is the ordering requirements based on the example above.

Table 35. Example of Transaction Ordering Requirements

st_tx (Interface)	***st_txreq (Interface)***
P4 (End)	NP6 (Start & End)
P4 (…)	NP5 (Start & End)
P4 (Start)	NP4 (Start & End)
P3 (End)	
P3 (Start)	
NP2 (Start & End)	
P1 (End)	
P1 (Start)	

*Ordering Rules - Row can bypass Column

Note: Y/N means there's no ordering architecture requirements between different Non-Posted cycles. However, it is highly advisable for implementation to ensure Non-Posted cycles received within an individual interface to be processed in a FIFO manner.

Although the DM Interrupt can send over AXI-ST_TX_REQ interface, the application must make sure
that this interrupt has no dependency towards the AXI-ST_TX interface. If the application would want
to have the DM interrupt order with the posted memory write, it is mandatory to send the DM Interrupt over AXI-ST_TX interface.

For the downstream transaction, regardless of if completion reorder buffer is enabled/disabled, since
completions will go over the AXI-ST_RX interface, while the Posted/Non-Posted will go over the AXIST_RX_REQ interface (optionally can be routed to the AXI-MM initiator interface through Device ATT programming), the AXI-ST_RX interface has no ordering relationship maintenance over the AXIST_RX_REQ as well as the AXI-MM Initiator interface. While for the

AXI-ST_RX_REQ, the Posted and Non-Posted will be strictly ordered per its incoming sequence.

Note: For applications that would want to maintain a producer consumer ordering, the application
has to be aware of the data mover order handling and make wise judgement of the application implementation to maintain ordering. An example could be sending a posted MSI after the Posted Mem Write on same channel, to push all the posted mem write to system and with MSI to notify system that the mem write data producing had completed. Alternatively, sending a zero-length read after the memory write transactions, wait for the completion of zero length read return, prior setting producer done
status in device.

4.3.10. AXI-Streaming Interface

4.3.10.1. Header Format

The header will be presented in line with data instead of a separate Side Interface. The PF Number, VF Number, BAR decode, and Prefix information will be presented in line with data. The PCIe Header and these side signals will be grouped as 32-byte header on AXI Streaming interface

The AXI Data Mover mode supports TLPs with two types of commands, PCIe Commands and Subsystem specific commands. The Subsystem specific commands will use Data Mover command header. The PCIe Commands will follow Power User mode header Format specified in previous sections.

- The tuser_vendor[0] bit will be used to differentiate between two headers.
- tuser_vendor[0]=0 indicates Power User mode header
- tuser_vendor[0]=1 indicates Data Mover header

4.3.10.2. Data Mover Header

This section describes Data Mover header format. The table below shows Data Mover header format

Table 36. Data Mover Header Format

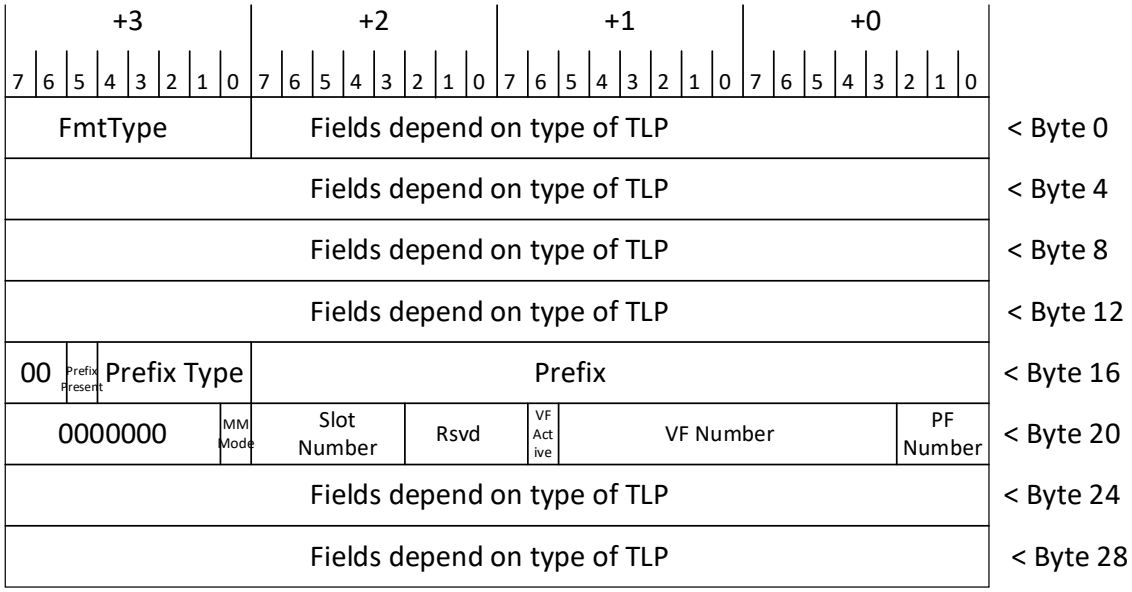
Tdata Header Byte Index	Header Fields	Bits	Tdata Bit Position End	Tdata Bit Position Start
Byte 15 - Byte 0	Data Mover Header	128	127	0
Byte 19 - Byte 16	Prefix	24	151	128
	Prefix Type	5	156	152
	Prefix Present	1	157	157
	Reserved	2	159	158
Byte 23 - Byte 20	PF Number	3	162	160

	VF Number	11	173	163
	VF Active	1	174	174
	Reserved	4	178	175
	Slot number	5	183	179
	MM Mode	1	184	184
	Reserved*	7	191	185
Byte 31 - Byte 24	Local Address/Meta Data	64	255	192

*Note: Bits[186:185] are reserved for future use

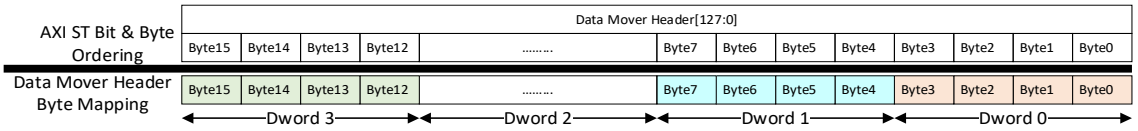
The figure below shows Data Mover header format.

Figure 52. Data Mover Header Format



The Figure below shows Data Mover header mapping to AXI ST Tdata bus.

Figure 53. Data Mover Header Mapping to AXI-ST Tdata



4.3.10.3. Format and Type (Fmt/Type)

Table 37. Data Mover Header Format/Type Field Description

Fmt/Type	Subsystem Command
'h20	Data Mover Read [DMRd]
'h60	Data Mover Write [DMWr]
'h30	Data Mover Interrupt [DMIntr]
'h4A	Data Mover Completion [DMCpl]

4.3.10.4. Prefix Information

Refer to section 2.1.21.2

4.3.10.5. Function Number

Refer section 2.1.21.3

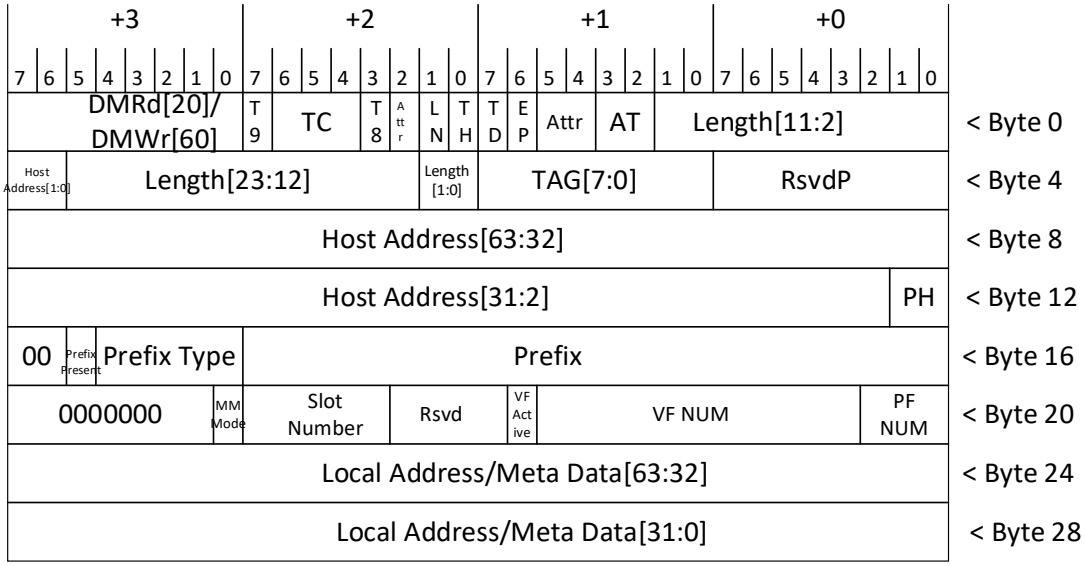
4.3.10.6. Data and Header Packing Schemes

The AXI Data Mover mode streaming interface will implement all packing schemes defined under section 2.2.

4.3.10.7. Data Mover Read/Write Command TLP Header [DMRd / DMWr]

The figure below shows Data Mover Read/Write command TLP header format.

Figure 54. Data Mover Read/Write Command TLP Header Format



4.3.10.7.1. Length

The length field indicates length of transfer in bytes. The length field supports transfer sizes up to 16MB. The transfer length of "0" is legal. The zero-length read and write can be initiated by setting length field equal to zero.

Table 38. TLP Header Length Field

Length[23:0]	Corresponding Data payload Size
24'h000000	0 Byte
24'h000001	1 Byte
24'h000002	2 Bytes
……	….
24'hFFFFFF	16777215 Bytes

4.3.10.7.2. Attribute Bits

The attribute bits (AT, NoSnoop, RelaxOrder, TH, IDOrder and PH) provided in the request are forwarded on the outgoing TLP. The PCIe Subsystem doesn't implement any functionality related to these attributes. The attributes of read request with transfer size greater than MRRS (PF0) will be applied to all TLPs generated for that particular read request. The attributes of write request with transfer size greater than MPS (PF0) will be applied to all TLPs generated for that particular write request.

4.3.10.7.3. TAG

The tag field in packet header is used to represent application tag for each non-posted request. This field must be unique for all outstanding requests regardless of physical functions and virtual functions. The Application must only send a 10-bit tag when the PCIe link is supporting a 10-bit tag, otherwise customer must only use 8-bit tag.

The PCIe SS may remap the application tag to the PCIe tag prior sending upstream to PCIe Link. The PCIe SS will remap back the completions with the application tag prior to returning to the applications.

Please refer to the Completion Reorder Handling section for more information on the Completion Reorder Handling behavior.

4.3.10.7.4. Host Address

This field indicates the 64-bit byte address of HOST memory for DMRd and DMWr commands.

4.3.10.7.5. Local Address / Meta Data

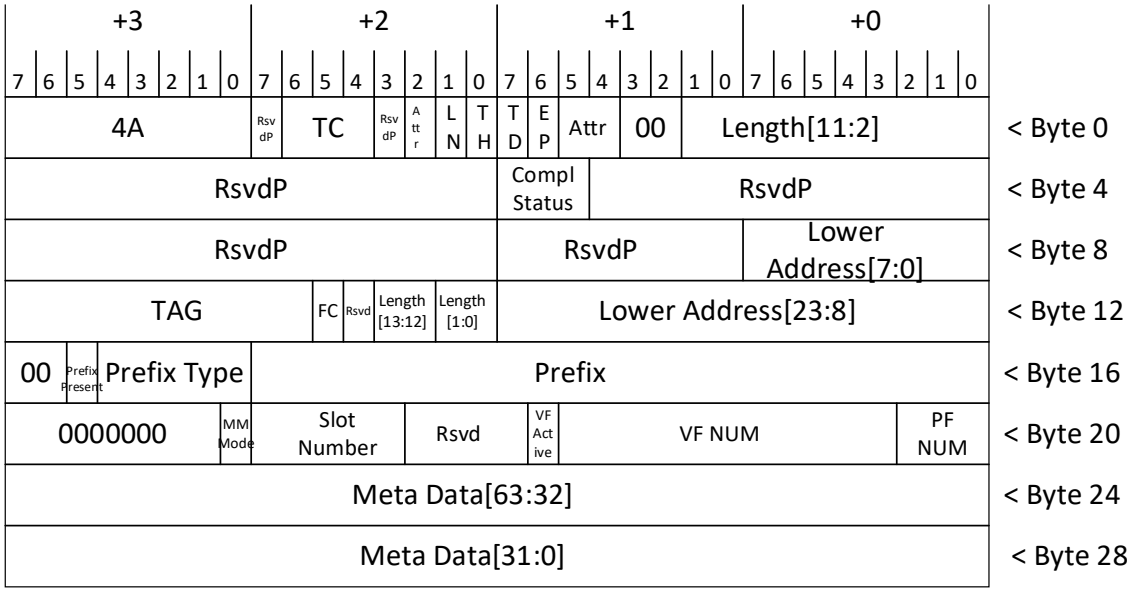
This field indicates the 64-bit byte address of local (FPGA) memory for DMRd and DMWr commands when MM Mode bit is set.

The user can send metadata information with DMRd command when MM Mode bit is clear. This meta data information is sent back to the user with a completion. The DMWr command with MM Mode bit clear drops this information in the core.

4.3.10.8. Data Mover Completion Command TLP Header [DMCpl]

The figure below shows completion header for DMRd command. The fields indicating RsvdP are preserved from original completion received from HOST. The user logic can use the first 12 bytes of header for error reporting purpose.

Figure 55. Data Mover Completion Command TLP Header



4.3.10.8.1. Length

The length indicates number of bytes returned with current completion.

Table 39. Mapping of Length and Data Payload Sizes

Length [13:0]	**Corresponding Data Payload Size**
14'h0000	0 Byte
14'h0001	1 Byte
14'h0002	2 Bytes
……	….
14'h3FFF	16383 Bytes

4.3.10.8.2. Attribute Bits

The attribute bits received in original completion from HOST are preserved and forwarded to application.

4.3.10.8.3. Lower Address [23:0]

The Lower Address indicates the lower 24 bits of starting byte address in current completion.

When completion reorder is enabled, the design is expected to return the entire completion up to the request size of 16MB. The upper 10 bits [23:14] of lower address field are repurposed to indicate length [23:14].

When completion reordering buffer is disabled, the application is required to reorder the completions.

4.3.10.8.4. Final Completion [FC]

This field indicates the current completion is the final completion for the request.

4.3.10.8.5. Completion Status

Table 40. Data Mover Header Completion Status Field Description

Completion Status	Description
000	Successful Completion
001	Unsupported Request
010	Reserved
100	Completer Abort
all others	Reserved

4.3.10.8.6. Meta Data

The user logic can send metadata information with DMRd command when MM Mode bit is clear. The meta data received with request is sent back in this field.

5. PCIe Subsystem Intel FPGA IP Parameters

The PCIe Subsystem Intel FPGA IP parameter editor provides the parameters you can set to configure your PCIe Subsystem Intel FPGA IP variation and simulation and hardware design examples.

5.1. Parameter Editor Parameters

The PCIe Subsystem Intel FPGA IP parameter has one tab, the **PCIe Interfaces** tab.

Table 41. PCIe Subsystem Intel FPGA IP Parameters: PCIe Subsystem – PCIe Interfaces Tab

Parameter	Default Setting	Parameter Description
PCI Interfaces		
Interfaces 0		

Table 42. PCIe Subsystem Intel FPGA IP Parameters: PCIe Subsystem – PCIe Interface 0 Settings

Parameter	Default Setting	Parameter Description
PCIE Interfaces 0 Settings		
PCIe Functional Mode	Power User	Select operational mode of PCIe Subsystem Power User AXI-ST Data Mover
PCIe Profile	Basic	Select functional features based on profile like virtualization, additional interfaces, number of endpoints, etc. - Basic - Basic+ - Virtual - Virtual+
PCIe Mode	Gen4 2x8	Selects the width of the data interface between the transaction layer and the application layer implemented in the PLD fabric, the lane data rate and the lane rate Gen4 1x16 Gen4 1x8 Gen4 2x8 Gen3 1x16 Gen3 2x8
Enable TLP-Bypass Mode	Disabled	Enable the TLP Bypass mode.
Port Mode	Native Endpoint	Selects Port Mode. For Endpoint mode with multiple ports, all ports are set to endpoint mode
Enable PHY Reconfiguration	Disabled	When on, creates an Avalon-MM slave interface that software can drive to update Transceiver reconfiguration registers

PLD Clock Frequency	400MHz	Select the PLD Clock Frequency 500MHz (Gen4) 450MHz (Gen4) 400MHz (Gen4) 350MHz (Gen4) 250MHz (Gen3)
Enable SRIS Mode	Disabled	Enable separate reference clock with independent Spread Spectrum Clocking (SSC)
Enable PCS and controller user reset	Disabled	Enable PCS and controller user reset in Endpoint and Bypass modes
Enable CVP (Intel VSEC)	Disabled	Enablement of CVP for single tile only
Enable Debug Toolkit	Disabled	Enable Debug Toolkit
Data Mover Mode Features Interfaces 0 (Data Mover Mode only)		
Enable Device Address Translation Table	Disabled	Enable Device Address Translation Table in Data Mover Mode
Enable completion timeout through AXI-ST interface	Disabled	Synthesizes completion packet with CA status if completion timeout event happens. When this feature is enabled, the Completion Timeout Interface option should be disabled
Enable Completion Reordering	Disabled	Enable Completion Reordering in Data Mover Mode. When this feature is enabled, the subsystem instantiates reordering buffer and enables sending completion in same order as corresponding request is received. When Completion Reordering is enabled, PCIe0 Device Address Translation Table must also be enabled
PCIe0 Reorder Buffer Size (in KB)	64	Size of reordering buffer in Kilo Bytes. Supports 32, 64, 128,256 KB sizes
Optional Side Interfaces 0		
Enable Control Shadow Interface	Disabled	Enable Control Shadow Interface. Host write to specific PCIe configuration space register's bit is indicated through this interface.
Enable Completion Timeout Interface	Disabled	Enable Completion Timeout Interface. Completion Timeout event is indicated through this interface
Enable Configuration Extension Bus Interface	Disabled	Enable Configuration Extension Bus Interface. User can add additional PCIe capabilities using this interface
PCIe0 Standard next address pointer for PF	0x00000000	This parameter is available when “Enable Configuration Extension Bus Interface” parameter is enabled. Enable CEB pointer address for PF (DW address in Hex). Valid range from 0x000 to 0x03F.

PCIe0 Extended next address pointer for PF	0x00000000	This parameter is available when "Enable Configuration Extension Bus Interface" parameter is enabled. Enable CEB pointer address for PF (DW address in Hex). Valid range from 0x040 to 0x3FF.
PCIe0 Standard next address pointer for VF	0x00000000	This parameter is available when "Enable Configuration Extension Bus Interface" parameter is enabled. Enable CEB pointer address for PF (DW address in Hex). Valid range from 0x000 to 0x03F.
PCIe0 Extended next address pointer for VF	0x00000000	This parameter is available when "Enable Configuration Extension Bus Interface" parameter is enabled. Enable CEB pointer address for PF (DW address in Hex). Valid range from 0x040 to 0x3FF.
PCIe0 CEB REQ to ACK Latency Timeout value	100	This parameter is available when "Enable Configuration Extension Bus Interface" parameter is enabled. Enable CEB REQ to ACK Latency Timeout value (in Clock Cycles). Valid range from 1 to 256
Enable PCIe0 Configuration Intercept Interface	Disabled	Enable Configuration Intercept Interface. User can intercept PCIe Configuration cycles using this interface (Power User Mode only)
PCIe0 CII REQ to ACK Latency Timeout value	100	This parameter is available when "Enable Configuration Intercept Interface" parameter is enabled. Enable CII REQ to ACK Latency Timeout value (in Clock Cycles). Valid range from 1 to 256.
Enable PCIe0 VirtIO PCI CFG Interface	Disabled	HOST read and write accesses to VIRTIO PCI Config Access Data Register will use this interface for its alternate access functionality
Enable PCIe0 PTM Interface	Disabled	Provide application with an interface to perform PTM request or receive PTM updates Note: This feature is not supported in the current Quartus release
MSI-X Table Feature Interfaces 0 (Data Mover Mode Only)		
Enable PCIe Subsystem Port0 MSI-X Table	Disabled	Indicates if PCIe Subsystem MSI-X and PBA table feature is enabled or disabled Note: The PCIe subsystem MSI-X table will become the superset of P-tile PCIe MSI-X table, when enabled
Port 0 MSI-X Table Size	4	Indicates PCIe Subsystem MSI-X Table size per Function. User must choose a value such that overall MSI-X Table size across all functions in P-Tile PCIe doesn't cross 4096 vectors except when "MSI-X Vector Allocation Policy" is Dynamic. This is only applicable if PCIe Subsystem MSI-X Table is Enable.

MSI-X BIR	5	Indicates the PCIe Subsystem BAR Indicator to be used for MSI-X and PBA Table in P-Tile PCIe. This is only applicable if PCIe Subsystem MSI-X Table is enabled. Note: P-Tile PCIe Table size = (PCIe Subsystem Table size - 1)
MSI-X BAR Offset	0x0000_0000_0000_0000	Indicates the PCIe Subsystem BAR Offset to be used for MSI-X and PBA Table in P-Tile PCIe. Offset value must be QW aligned (i.e., bit[2:0] should be all 0's). This is only applicable if PCIe Subsystem MSI-X Table is enabled. Note 1: P-Tile PCIe Table offset = (PCIe Subsystem Table offset >> 3) Note 2: P-Tile PCIe PBA offset = (PCIe Subsystem Table offset + (PCIe Subsystem Table size * 16) >> 3)
MSI-X Vector Allocation Policy	Static	Indicates the PCIe Subsystem MSI-X vector allocation policy. Static - Evenly distributed across all functions in P-Tile PCIe Dynamic - Dynamically assigned for each PF/VF based on MSI-X PF/VF Offset table in P-Tile PCIe. This is only applicable if PCIe Subsystem MSI-X Table is enabled. **Note:** When Dynamic MSI-X feature is enabled, the interrupt numbers reported in the configuration space may not match the actual number of interrupts setup by the management software. You must use the management software to handle the configuration of the interrupts after initial bootup. This feature is not supported in the current release
DFL VSEC and DFH CSR Interfaces 0 Settings		
Port 0 DFL VSEC Interfaces 0		
Enable PCIe0 DFL VSEC Capability	Disabled	Indicates if DFL VSEC Capability Exist.
Number of PCIe0 DFL	1	Indicates number of DFLs in a design. Valid range 1-32
Port 0 DFH CSR Interfaces 0		
PCIe0 DFH FID	0x00000000	Sets DFH Feature ID Field in DFH Header. Valid range 0x00000000-0x00000fff
PCIe0 DFH MAJOR VER	0x00000000	Sets Major version Field in DFH Header. Valid range 0x00000000-0x0000000f
PCIe0 DFH NEXT BYTE OFFSET	0x00000000	Sets Next Byte Offset Field of DFH Header. Valid range 0x00000000-0x00ffffff
PCIe0 DFH END	0	Sets End of List bit of DFH Header. Valid range 0-1

PCIe0 DFH MINOR REV	0x00000000	Sets Minor revision Field of DFH Header. Valid range 0x00000000-0x0000000f
PCIe0 DFH VER	0x00000000	Sets DFH Version Field of DFH Header. Valid range 0x00000000-0x000000ff
PCIe0 DFH FEATURE TYPE	0x00000000	Sets Feature Type field of DFH Header. Valid range 0x00000000-0x0000000f
PCIe0 INST ID	0x00000000	Sets Instance ID field of X_FEATURE_CSR_SIZE_GROUP Register. Valid range 0x00000000-0x0000ffff
PCIe Interfaces 0 Ports Settings		
Extend PCIe Ports Settings	Off	Extend PCIe Subsystem Settings. When enabled, PCIe IP setting and can be further configured.
Port 0		
***PCIe0 Avalon Settings when* Extend PCIe Ports Settings=On**		
Enable byte parity Port on Avalon-ST interface	On	Enables or disables parity Port on Avalon-ST interface. When on, the application layer must provide valid byte parity in the Avalon-ST TX direction.
Enable Power Management Interface and Hard IP Status Interface	On	When selected, Power Management interface and Hard IP Status Interface will be exported.
Power management interface: Enable p0_apps_ready_entr_l23_i port	Off	When selected, input port p0_apps_ready_entr_l23_i is exposed in the Power Management interface. The application logic asserts this signal to indicate that it is ready to enter L2/L3 Ready state. It is provided for applications that must control L2/L3 Ready entry (in case certain task must be performed before going into L2/L3 Ready).
Power management interface: Enable p0_app_xfer_pending_i port	Off	When selected, input port p0_app_xfer_pending_i is exposed in the Power Management interface. This port is used to prevent the entry to L1 or initiates the exit from L1.
Enable Legacy Interrupt	On	When selected, Power Management interface will be exported.
Enable Completion Timeout Interface	On	Select to enable completion timeout interface.
Enable Configuration Intercept Interface	On	Select to enable configuration intercept interface.
Enable PRS Event	On	Select to enable PRS Event Interface.

Enable Error Interface	On	Select to enable Error Interface.
Enable 10-bit tag support Interface	Off	Enable 10-bit tag support enable interface
PCIe0 Configuration, Debug and Extension Options		
Gen 3 Requested equalization far-end TX preset vector	0x00000004	Specifies the Gen 3 requested phase 2/3 far-end TX preset vector. Choosing a value different from the default is not recommended for most designs.
Gen 4 Requested equalization far-end TX preset vector	0x00000270	Specifies the Gen 4 requested phase 2/3 far-end TX preset vector. Choosing a value different from the default is not recommended for most designs.
Enable Rx Buffer Limit Ports	Disabled	When selected, RX buffer limit ports will be exported for User to control RX Posted, Non-Posted and CplD Packets. Else Max Buffer Size will be used.
Bypass Posted Rx Buffer Limit	Disabled	When selected, RX buffer limit selected for Posted packets will be bypassed.
Bypass Non-Posted Rx Buffer Limit	Disabled	When selected, RX buffer limit selected for Non-Posted packets will be bypassed.
Bypass CplD Rx Buffer Limit	Enabled (Data Mover mode), Disabled (Power User Mode)	When selected, RX buffer limit selected for CplD packets will be bypassed.
Enable HIP dynamic reconfiguration of PCIe registers	Enabled	When on, creates an Avalon-MM slave interface that software can drive to update global configuration registers which are read-only at run time.
PCIe0 Base Address Registers		
PCIe0 PF0 BAR Configuration		
PCIe0 PF0 BAR		
BAR0 Type	64-bit prefetchable memory	Sets the BAR type
BAR0 Size	64 Kbytes – 16 bits	Sets from 7-64 bits per base address register (BAR).
BAR1 Type	Disabled	Sets the BAR type.
BAR2 Type	Disabled	Sets the BAR type.
BAR3 Type	Disabled	Sets the BAR type.
BAR4 Type	Disabled	Sets the BAR type.
BAR5 Type	Disabled	Sets the BAR type.

Expansion ROM Size	Disabled	Specifies an expansion ROM from 4 KBytes - 16 MBytes when enabled.
PCIe0 PF0 VF BAR		
BAR0 Type	64-bit prefetchable memory	Sets the BAR type
BAR0 Size	64 Kbytes – 16 bits	Sets from 7-64 bits per base address register (BAR).
BAR1 Type	Disabled	Sets the BAR type.
BAR2 Type	Disabled	Sets the BAR type.
BAR3 Type	Disabled	Sets the BAR type.
BAR4 Type	Disabled	Sets the BAR type.
BAR5 Type	Disabled	Sets the BAR type.
Expansion ROM Size	Disabled	Specifies an expansion ROM from 4 KBytes - 16 MBytes when enabled.
PCIe0 PF1 BAR Configuration		
PCIe0 PF1 BAR		
BAR0 Type	64-bit prefetchable memory	Sets the BAR type
BAR0 Size	64 Kbytes – 16 bits	Sets from 7-64 bits per base address register (BAR).
BAR1 Type	Disabled	Sets the BAR type.
BAR2 Type	Disabled	Sets the BAR type.
BAR3 Type	Disabled	Sets the BAR type.
BAR4 Type	Disabled	Sets the BAR type.
BAR5 Type	Disabled	Sets the BAR type.
Expansion ROM Size	Disabled	Specifies an expansion ROM from 4 KBytes - 16 MBytes when enabled.
PCIe0 PF1 VF BAR		
BAR0 Type	64-bit prefetchable memory	Sets the BAR type
BAR0 Size	64 Kbytes – 16 bits	Sets from 7-64 bits per base address register (BAR).
BAR1 Type	Disabled	Sets the BAR type.
BAR2 Type	Disabled	Sets the BAR type.
BAR3 Type	Disabled	Sets the BAR type.
BAR4 Type	Disabled	Sets the BAR type.
BAR5 Type	Disabled	Sets the BAR type.
Expansion ROM Size	Disabled	Specifies an expansion ROM from 4 KBytes - 16 MBytes when enabled.
PCIe0 PF2 BAR Configuration		
PCIe0 PF2 BAR		
BAR0 Type	64-bit prefetchable memory	Sets the BAR type

BAR0 Size	64 Kbytes – 16 bits	Sets from 7-64 bits per base address register (BAR).
BAR1 Type	Disabled	Sets the BAR type.
BAR2 Type	Disabled	Sets the BAR type.
BAR3 Type	Disabled	Sets the BAR type.
BAR4 Type	Disabled	Sets the BAR type.
BAR5 Type	Disabled	Sets the BAR type.
Expansion ROM Size	Disabled	Specifies an expansion ROM from 4 KBytes - 16 MBytes when enabled.
PCIe0 PF2 VF BAR		
BAR0 Type	64-bit prefetchable memory	Sets the BAR type
BAR0 Size	64 Kbytes – 16 bits	Sets from 7-64 bits per base address register (BAR).
BAR1 Type	Disabled	Sets the BAR type.
BAR2 Type	Disabled	Sets the BAR type.
BAR3 Type	Disabled	Sets the BAR type.
BAR4 Type	Disabled	Sets the BAR type.
BAR5 Type	Disabled	Sets the BAR type.
Expansion ROM Size	Disabled	Specifies an expansion ROM from 4 KBytes - 16 MBytes when enabled.
PCIe3 PF1 BAR Configuration		
PCIe3 PF1 BAR		
BAR0 Type	64-bit prefetchable memory	Sets the BAR type
BAR0 Size	64 Kbytes – 16 bits	Sets from 7-64 bits per base address register (BAR).
BAR1 Type	Disabled	Sets the BAR type.
BAR2 Type	Disabled	Sets the BAR type.
BAR3 Type	Disabled	Sets the BAR type.
BAR4 Type	Disabled	Sets the BAR type.
BAR5 Type	Disabled	Sets the BAR type.
Expansion ROM Size	Disabled	Specifies an expansion ROM from 4 KBytes - 16 MBytes when enabled.
PCIe3 PF1 VF BAR		
BAR0 Type	64-bit prefetchable memory	Sets the BAR type
BAR0 Size	64 Kbytes – 16 bits	Sets from 7-64 bits per base address register (BAR).
BAR1 Type	Disabled	Sets the BAR type.
BAR2 Type	Disabled	Sets the BAR type.
BAR3 Type	Disabled	Sets the BAR type.
BAR4 Type	Disabled	Sets the BAR type.

BAR5 Type	Disabled	Sets the BAR type.
Expansion ROM Size	Disabled	Specifies an expansion ROM from 4 KBytes - 16 MBytes when enabled.
PCIe0 Device Identification Registers		
PCIe0 PF0 IDs		
Vendor ID	0x00001172	
Device ID	0x00000000	Sets the read-only value of the Device ID register.
Revision ID	0x00000001	Sets the read-only value of the Revision ID register.
Class code	0x00FF0000	Sets the read-only value of the Class code register.
Subsystem Vendor ID	0x00000000	Sets the read-only value of the Subsystem Vendor ID register.
Subsystem Device ID	0x0000000	Sets the read-only value of the Subsystem Device ID register.
PCIe0 PF0 VF IDs		
Device ID	0x00000000	
Subsystem ID	0x00000000	Sets the read-only value of the Subsystem ID register for the virtual functions
PCIe0 PF1 IDs		
Vendor ID	0x00000000	Sets the read-only value of the Vendor ID register.
Device ID	0x00000000	Sets the read-only value of the Device ID register.
Revision ID	0x00000001	Sets the read-only value of the Revision ID register.
Class code	0x00FF0000	Sets the read-only value of the Class code register.
Subsystem Vendor ID	0x00000000	Sets the read-only value of the Subsystem Vendor ID register.
Subsystem Device ID	0x0000000	Sets the read-only value of the Subsystem Device ID register.
PCIe0 PF1 VF IDs		
Device ID	0x00000000	Sets the read-only value of the Device ID register for the virtual functions
Subsystem ID	0x00000000	Sets the read-only value of the Subsystem ID register for the virtual functions
PCIe0 PF2 IDs		
Vendor ID	0x00000000	Sets the read-only value of the Vendor ID register.
Device ID	0x00000000	Sets the read-only value of the Device ID register.
Revision ID	0x00000001	Sets the read-only value of the Revision ID register.
Class code	0x00FF0000	Sets the read-only value of the Class code register.
Subsystem Vendor ID	0x00000000	Sets the read-only value of the Subsystem Vendor ID register.
Subsystem Device ID	0x0000000	Sets the read-only value of the Subsystem Device ID register.

PCIe0 PF2 VF IDs		
Device ID	0x00000000	Sets the read-only value of the Device ID register for the virtual functions
Subsystem ID	0x00000000	Sets the read-only value of the Subsystem ID register for the virtual functions
PCIe0 PF3 IDs		
Vendor ID	0x00000000	Sets the read-only value of the Vendor ID register.
Device ID	0x00000000	Sets the read-only value of the Device ID register.
Revision ID	0x00000001	Sets the read-only value of the Revision ID register.
Class code	0x00FF0000	Sets the read-only value of the Class code register.
Subsystem Vendor ID	0x00000000	Sets the read-only value of the Subsystem Vendor ID register.
Subsystem Device ID	0x0000000	Sets the read-only value of the Subsystem Device ID register.
PCIe0 PF3 VF IDs		
Device ID	0x00000000	Sets the read-only value of the Device ID register for the virtual functions
Subsystem ID	0x00000000	Sets the read-only value of the Subsystem ID register for the virtual functions
PCIe0 PCI Express / PCI Capabilities		
PCIe0 Device		
Maximum payload size supported	512 Bytes	Sets the read-only value of the max payload size of the Device Capabilities register and optimizes for this payload size.
Support Extended Tag Field	Enabled	Sets the Extended Tag Field Supported bit in Configuration Space Device Capabilities Register.
PCIe0 Multifunction and SR-IOV System Settings		
Enable multiple physical functions	On	Enables multiple physical functions.
Total physical functions (PFs)	4	Sets the number of physical functions.
Enable SR-IOV support	On	Enable SR-IOV.
Total virtual functions of physical function 0 (PF0 VFs)	0	Sets the number of VFs to be assigned to Physical Function 0.
Total virtual functions of physical function 1 (PF1 VFs)	128	Sets the number of VFs to be assigned to Physical Function 1.
Total virtual functions of physical function 2 (PF2 VFs)	128	Sets the number of VFs to be assigned to Physical Function 2.

Total virtual functions of physical function 3 (PF3 VFs)	128	Sets the number of VFs to be assigned to Physical Function 3.
PCIe0 Link		
Link port number (Root Port only)	1	Sets the read-only value of the port number field in the Link Capabilities register.
Slot clock configuration	On	Sets the read-only value of the slot clock configuration bit in the link status register.
PCIe0 Legacy Interrupt Pin Register		
PCIe0 PF0 Int Pin		
Set Interrupt Pin for PF0	NO INT	Sets interrupt Pin for PF0
PCIe0 PF1 Int Pin		
Set Interrupt Pin for PF0	NO INT	Sets interrupt Pin for PF1
PCIe0 PF2 Int Pin		
Set Interrupt Pin for PF0	NO INT	Sets interrupt Pin for PF2
PCIe0 PF3 Int Pin		
Set Interrupt Pin for PF0	NO INT	Sets interrupt Pin for PF3
PCIe0 LTR		
PCIe0 Enable LTR	Disabled	LTR (Latency Tolerance Reporting) New Mechanism that enables Endpoints to send information about their latency requirements for memory reads/writes and interrupts.
PCIe0 MSI		
PCIe0 PF0 MSI		
PF0 Enable MSI	Disabled	Enables or disables MSI capability for PF0.
PF0 MSI 64-bit addressing	Off	Enables or disables MSI 64-bit addressing for PF0.
PF0 MSI extended data capable	Off	Enables or disables MSI extended data capability for PF0
PF0 Number of MSI messages requested	1	Sets the number of messages that the application can request in the multiple messages capable field of the Message Control register 1 2 4 8 16 32
PCIe0 PF1 MSI		
PF1 Enable MSI	Disabled	Enables or disables MSI capability for PF1.
PF1 MSI 64-bit addressing	Off	Enables or disables MSI 64-bit addressing for PF1.

PF1 MSI extended data capable	Off	Enables or disables MSI extended data capability for PF1
PF1 Number of MSI messages requested	1	Sets the number of messages that the application can request in the multiple messages capable field of the Message Control register 1 2 4 8 16 32
PCIe0 PF2 MSI		
PF2 Enable MSI	Disabled	Enables or disables MSI capability for PF2.
PF2 MSI 64-bit addressing	Off	Enables or disables MSI 64-bit addressing for PF2.
PF2 MSI extended data capable	Off	Enables or disables MSI extended data capability for PF2
PF2 Number of MSI messages requested	1	Sets the number of messages that the application can request in the multiple messages capable field of the Message Control register 1 2 4 8 16 32
PCIe0 PF3 MSI		
PF3 Enable MSI	Disabled	Enables or disables MSI capability for PF3.
PF3 MSI 64-bit addressing	Off	Enables or disables MSI 64-bit addressing for PF3.
PF3 MSI extended data capable	Off	Enables or disables MSI extended data capability for PF3
PF3 Number of MSI messages requested	1	Sets the number of messages that the application can request in the multiple messages capable field of the Message Control register 1 2 4 8 16 32
PCIe0 MSI-X		
PCIe0 PF MSI-X		
PCIe0 PF0 MSI-X		
Enable MSI-X	On	Enables or disables the MSI-X capability

Table size	4	Sets the number of entries in the MSI-X table.
Table offset	0x0000000000000000	Sets the read-only base address of the MSI-X table. The low-order 3 bits are automatically set to 0.
Table BAR indicator	0	Specifies which one of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the MSI-X table into memory space. This field is read-only.
Pending bit array (PBA) offset	0x0000000000000000	Specifies the offset from the address stored in one of the function's base address registers that points to the base of the MSI-X PBA. This field is read-only.
PBA BAR indicator	0	Indicates which of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the function's MSI-X PBA into memory space. This field is read-only.
PCIe0 PF1 MSI-X		
Enable MSI-X	On	Enables or disables the MSI-X capability
Table size	4	Sets the number of entries in the MSI-X table.
Table offset	0x0000000000000000	Sets the read-only base address of the MSI-X table. The low-order 3 bits are automatically set to 0.
Table BAR indicator	0	Specifies which one of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the MSI-X table into memory space. This field is read-only.
Pending bit array (PBA) offset	0x0000000000000000	Specifies the offset from the address stored in one of the function's base address registers that points to the base of the MSI-X PBA. This field is read-only.
PBA BAR indicator	0	Indicates which of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the function's MSI-X PBA into memory space. This field is read-only.
PCIe0 PF2 MSI-X		

Enable MSI-X	On	Enables or disables the MSI-X capability
Table size	4	Sets the number of entries in the MSI-X table.
Table offset	0x0000000000000000	Sets the read-only base address of the MSI-X table. The low-order 3 bits are automatically set to 0.
Table BAR indicator	0	Specifies which one of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the MSI-X table into memory space. This field is read-only.
Pending bit array (PBA) offset	0x0000000000000000	Specifies the offset from the address stored in one of the function's base address registers that points to the base of the MSI-X PBA. This field is read-only.
PBA BAR indicator	0	Indicates which of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the function's MSI-X PBA into memory space. This field is read-only.
PCIe0 PF3 MSI-X		
Enable MSI-X	On	Enables or disables the MSI-X capability
Table size	4	Sets the number of entries in the MSI-X table.
Table offset	0x0000000000000000	Sets the read-only base address of the MSI-X table. The low-order 3 bits are automatically set to 0.
Table BAR indicator	0	Specifies which one of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the MSI-X table into memory space. This field is read-only.
Pending bit array (PBA) offset	0x0000000000000000	Specifies the offset from the address stored in one of the function's base address registers that points to the base of the MSI-X PBA. This field is read-only.

PBA BAR indicator	0	Indicates which of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the function's MSI-X PBA into memory space. This field is read-only.
PCIe0 VF MSI-X		
PCIe0 PF0 VF MSI-X		
Enable VF MSI-X	Enabled	Enables or disables the MSI-X capability.
Table size	4	Sets the number of entries in the MSI-X table.
Table offset	0x0000000000000000	Sets the read-only base address of the MSI-X table. The low-order 3 bits are automatically set to 0.
Table BAR indicator	0	Specifies which one of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the MSI-X table into memory space. This field is read-only.
Pending bit array (PBA) offset	0x0000000000000000	Specifies the offset from the address stored in one of the function's base address registers that points to the base of the MSI-X PBA. This field is read-only.
PBA BAR indicator	0	Indicates which of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the function's MSI-X PBA into memory space. This field is read-only.
PCIe0 PF1 VF MSI-X		
Enable VF MSI-X	Enabled	Enables or disables the MSI-X capability.
Table size	4	Sets the number of entries in the MSI-X table.
Table offset	0x0000000000000000	Sets the read-only base address of the MSI-X table. The low-order 3 bits are automatically set to 0.
Table BAR indicator	0	Specifies which one of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the MSI-X table into memory space. This field is read-only.

Pending bit array (PBA) offset	0x0000000000000000	Specifies the offset from the address stored in one of the function's base address registers the points to the base of the MSI-X PBA. This field is read-only.
PBA BAR indicator	0	Indicates which of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the function's MSI-X PBA into memory space. This field is read-only.
PCIe0 PF2 VF MSI-X		
Enable VF MSI-X	Enabled	Enables or disables the MSI-X capability.
Table size	4	Sets the number of entries in the MSI-X table.
Table offset	0x0000000000000000	Sets the read-only base address of the MSI-X table. The low-order 3 bits are automatically set to 0.
Table BAR indicator	0	Specifies which one of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the MSI-X table into memory space. This field is read-only.
Pending bit array (PBA) offset	0x0000000000000000	Specifies the offset from the address stored in one of the function's base address registers that points to the base of the MSI-X PBA. This field is read-only.
PBA BAR indicator	0	Indicates which of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the function's MSI-X PBA into memory space. This field is read-only.
PCIe0 PF3 VF MSI-X		
Enable VF MSI-X	Enabled	Enables or disables the MSI-X capability.
Table size	4	Sets the number of entries in the MSI-X table.
Table offset	0x0000000000000000	Sets the read-only base address of the MSI-X table. The low-order 3 bits are automatically set to 0.

Table BAR indicator	0	Specifies which one of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the MSI-X table into memory space. This field is read-only.
Pending bit array (PBA) offset	0x0000000000000000	Specifies the offset from the address stored in one of the function's base address registers that points to the base of the MSI-X PBA. This field is read-only.
PBA BAR indicator	0	Indicates which of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the function's MSI-X PBA into memory space. This field is read-only.
PCIe0 PASID		
PCIe0 PF0 PASID		
PCIE0 PF0 Enable PASID	Enabled	PASID (Process Address Space ID) Optional feature which allows a single endpoint to be shared by multiple processes by providing each a virtual 64-bit address space.
PCIe0 PF0 Enable Execute Permission Support	Disabled	Enables or disables PASID Execute Permission Support for PCIe0 PF0.
PCIe0 PF0 Enable Privileged Mode Support	Disabled	Enables or disables PASID Privileged Mode Support for PCIe0 PF0.
PCIe0 PF0 Max PASID Width	0	Sets the Max PASID Width for PCIe0 PF0.
PCIe0 PF1 PASID		
PCIE0 PF1 Enable PASID	Enabled	PASID (Process Address Space ID) Optional feature which allows a single endpoint to be shared by multiple processes by providing each a virtual 64-bit address space.
PCIe0 PF1 Enable Execute Permission Support	Disabled	Enables or disables PASID Execute Permission Support for PCIe0 PF1.
PCIe0 PF1 Enable Privileged Mode Support	Disabled	Enables or disables PASID Privileged Mode Support for PCIe0 PF1.
PCIe0 PF1 Max PASID Width	0	Sets the Max PASID Width for PCIe0 PF1.
PCIe0 PF2 PASID		

PCIE0 PF2 Enable PASID	Enabled	PASID (Process Address Space ID) Optional feature which allows a single endpoint to be shared by multiple processes by providing each a virtual 64-bit address space.
PCIe0 PF2 Enable Execute Permission Support	Disabled	Enables or disables PASID Execute Permission Support for PCIe0 PF2.
PCIe0 PF2 Enable Privileged Mode Support	Disabled	Enables or disables PASID Privileged Mode Support for PCIe0 PF2.
PCIe0 PF2 Max PASID Width	0	Sets the Max PASID Width for PCIe0 PF2.
PCIe0 PF3 PASID		
PCIE0 PF3 Enable PASID	Enabled	PASID (Process Address Space ID) Optional feature which allows a single endpoint to be shared by multiple processes by providing each a virtual 64-bit address space.
PCIe0 PF3 Enable Execute Permission Support	Disabled	Enables or disables PASID Execute Permission Support for PCIe0 PF3.
PCIe0 PF3 Enable Privileged Mode Support	Disabled	Enables or disables PASID Privileged Mode Support for PCIe0 PF3.
PCIe0 PF3 Max PASID Width	0	Sets the Max PASID Width for PCIe0 PF3.
PCIe0 DEV SER		
Enable Device Serial Number Capability	Off	Enables Device Serial Number Capability (DEVSER) optional extended capability is a 64-bit value that is unique for any given PCIe device.
Device Serial Number (DW1)	Disabled	Set the lower 32 bits of IEEE 64-bit Device Serial Number (DW1)
Device Serial Number (DW2)	Disabled	Set the upper 32 bits of IEEE 64-bit Device Serial Number (DW2)
PCIe0 PRS		
PCIe0 PF0 PRS		
PF0 Enable PRS	Off	Enable PF0 Page Request Service (PRS)
PCIe0 PF1 PRS		
PF1 Enable PRS	Off	Enable PF1 Page Request Service (PRS)
PCIe0 PF2 PRS		
PF2 Enable PRS	Off	Enable PF2 Page Request Service (PRS)
PCIe0 PF3 PRS		
PF3 Enable PRS	0ff	Enable PF3 Page Request Service (PRS)

PCIe0 Power Management		
Endpoint L0s acceptable latency	Maximum of 64ns	Sets the read-only value of the endpoint L0s acceptable latency field of the Device Capabilities register. This value should be based on the latency that the application layer can tolerate. This setting is disabled for root ports.
Endpoint L1s acceptable latency	Maximum of 1 us	Sets the acceptable latency that an endpoint can withstand in the transition from the L1 to L0 state. It is an indirect measure of the endpoint internal buffering. This setting is disabled for root ports.
PCIe0 VSEC		
Vendor Specific Extended Capability	Off	Enables Vendor Specific Extended Capability (VSEC). Note: Please enable Configuration Intercept Interface (CII) when using Vendor Specific Extended Capability
User ID register from the Vendor Specific Extended Capability	0x00000000	Sets the read-only value of the 16-bit User ID register from the Vendor Specific Extended Capability
Drops Vendor Type0 Messages	Off	When selected, received Vendor MSG Type0 will be dropped as an Unsupported Request(UR). Otherwise, received Vendor MSG Type0 will not be dropped, but visible on RX AVST interface. This option will not be applicable for TLP Bypass mode. In TLP Bypass mode, received Vendor MSG Type0 will always be visible on RX AVST interface.
Drops Vendor Type1 Messages	0ff	When selected, received Vendor MSG Type1 will be dropped silently. Otherwise, received Vendor MSG Type1 will not be dropped, but visible on RX AVST interface. This option will not be applicable for TLP Bypass mode. In TLP Bypass mode, received Vendor MSG Type1 will always be visible on RX AVST interface.
Note: Please enable PCIe0 Configuration Intercept Interface (CII) when using PCIe0 Vendor Specific Extended Capability.		
PCIe0 ATS		
PCIe0 ATS for Physical Functions		
PCIe0 ATS for PF0		

Enable Address Translation Services (ATS)	On	When Address Translation Services (ATS) is enabled, senders can request and cache translated addresses using the RP memory space for later use
PCIe0 ATS for PF1		
Enable Address Translation Services (ATS)	On	When Address Translation Services (ATS) is enabled, senders can request and cache translated addresses using the RP memory space for later use
PCIe0 ATS for PF2		
Enable Address Translation Services (ATS)	On	When Address Translation Services (ATS) is enabled, senders can request and cache translated addresses using the RP memory space for later use
PCIe0 ATS for PF3		
Enable Address Translation Services (ATS)	On	When Address Translation Services (ATS) is enabled, senders can request and cache translated addresses using the RP memory space for later use
PCIe0 ATS for Virtual Functions		
PCIe0 ATS for PF0 VF		
Enable Address Translation Services (ATS)	On	When Address Translation Services (ATS) is enabled, senders can request and cache translated addresses using the RP memory space for later use
PCIe0 ATS for PF1 VF		
Enable Address Translation Services (ATS)	On	When Address Translation Services (ATS) is enabled, senders can request and cache translated addresses using the RP memory space for later use
PCIe0 ATS for PF2 VF		
Enable Address Translation Services (ATS)	On	When Address Translation Services (ATS) is enabled, senders can request and cache translated addresses using the RP memory space for later use

PCIe0 ATS for PF3 VF		
Enable Address Translation Services (ATS)	On	When Address Translation Services (ATS) is enabled, senders can request and cache translated addresses using the RP memory space for later use
PCIe0 TPH		
PCIe0 TPH for Physical Functions		
PCIe0 TPH for PF0		
Enable TLP Processing Hints (TPH)	On	Using TLP Processing Hints (TPH) may improve latency and traffic congestion
PCIe0 TPH for PF1		
Enable TLP Processing Hints (TPH)	On	Using TLP Processing Hints (TPH) may improve latency and traffic congestion
PCIe0 TPH for PF2		
Enable TLP Processing Hints (TPH)	On	Using TLP Processing Hints (TPH) may improve latency and traffic congestion
PCIe0 TPH for PF3		
Enable TLP Processing Hints (TPH)	On	Using TLP Processing Hints (TPH) may improve latency and traffic congestion
PCIe0 TPH for Virtual Functions		
PCIe0 TPH for PF0 VF		
Enable TLP Processing Hints (TPH)	On	Using TLP Processing Hints (TPH) may improve latency and traffic congestion
PCIe0 TPH for PF1 VF		
Enable TLP Processing Hints (TPH)	On	Using TLP Processing Hints (TPH) may improve latency and traffic congestion
PCIe0 TPH for PF2 VF		
Enable TLP Processing Hints (TPH)	On	Using TLP Processing Hints (TPH) may improve latency and traffic congestion
PCIe0 TPH for PF3 VF		
Enable TLP Processing Hints (TPH)	On	Using TLP Processing Hints (TPH) may improve latency and traffic congestion
PCIe0 ACS Capabilities		
PCIe0 ACS for Physical Functions		
PCIe0 ACS for PF0		
Enable Access Control Service (ACS)	Off	ACS defines a set of control points within a PCI Express topology to determine whether a TLP is to be routed normally, blocked, or redirected.

Enable ACS P2P Traffic Support	Off	Indicates if the component supports Peer to Peer Traffic
Enable ACS P2P Egress Control	Off	Indicates if the component implements ACS P2P Egress Control
PCIe0 ACS for PF1		
Enable Access Control Service (ACS)	Off	ACS defines a set of control points within a PCI Express topology to determine whether a TLP is to be routed normally, blocked, or redirected.
Enable ACS P2P Traffic Support	Off	Indicates if the component supports Peer to Peer Traffic
Enable ACS P2P Egress Control	Off	Indicates if the component implements ACS P2P Egress Control
PCIe0 ACS for PF2		
Enable Access Control Service (ACS)	Off	ACS defines a set of control points within a PCI Express topology to determine whether a TLP is to be routed normally, blocked, or redirected.
Enable ACS P2P Traffic Support	Off	Indicates if the component supports Peer to Peer Traffic
Enable ACS P2P Egress Control	Off	Indicates if the component implements ACS P2P Egress Control
PCIe0 ACS for PF3		
Enable Access Control Service (ACS)	Off	ACS defines a set of control points within a PCI Express topology to determine whether a TLP is to be routed normally, blocked, or redirected.
Enable ACS P2P Traffic Support	Off	Indicates if the component supports Peer to Peer Traffic
Enable ACS P2P Egress Control	Off	Indicates if the component implements ACS P2P Egress Control
PCIe0 ACS for Virtual Functions		
PCIe0 ACS for PF0 VF		
Enable Access Control Service (ACS)	Off	ACS defines a set of control points within a PCI Express topology to determine whether a TLP is to be routed normally, blocked, or redirected.
PCIe0 ACS for PF1 VF		
Enable Access Control Service (ACS)	Off	ACS defines a set of control points within a PCI Express topology to determine whether a TLP is to be routed normally, blocked, or redirected.
PCIe0 ACS for PF2 VF		

Enable Access Control Service (ACS)	Off	ACS defines a set of control points within a PCI Express topology to determine whether a TLP is to be routed normally, blocked, or redirected.
PCIe0 ACS for PF3 VF		
Enable Access Control Service (ACS)	Off	ACS defines a set of control points within a PCI Express topology to determine whether a TLP is to be routed normally, blocked, or redirected.
PCIe0 VIRTIO		
Enable VIRTIO support	On	If set, enable VIRTIO Capabilities for PFs and VFs
PCIe0 PF0 VIRTIO STRUCTURES		
Enable VIRTIO Capabilities for PF0	Off	Exposes VIRTIO Capabilities for VIRTIO Capable Devices
Enable Device Specific Capability for PF0	On	Enables Device Specific Capability for VIRTIO Device on PF0
PCIe0 PF0 COMMON CONFIGURATION STRUCTURE		
BAR Indicator	Disabled	Indicates BAR holding the Common Configuration Structure
Offset within BAR	Disabled	Indicates starting position of Common Config Structure in given BAR
Structure Length in Bytes	Disabled	Indicates length of Common Config Structure
PCIe0 PF0 NOTIFICATION STRUCTURE		
BAR Indicator	Disabled	Indicates BAR holding the Notification Structure
Offset within BAR	Disabled	Indicates starting position of Notification Structure in given BAR
Structure Length in Bytes	Disabled	Indicates length of Notification Structure
Notify Off Multiplier	Disabled	Indicates Multiplier for queue_notify_off
PCIe0 PF0 ISR STATUS STRUCTURE		
BAR Indicator	Disabled	Indicates BAR holding the ISR STATUS Structure
Offset within BAR	Disabled	Indicates starting position of ISR STATUS Structure in given BAR
Structure Length in Bytes	Disabled	Indicates length of ISR STATUS Structure
PCIe0 PF0 PCI CONFIGURATION ACCESS STRUCTURE		
BAR Indicator	Disabled	Indicates BAR holding the PCI Configuration Access Structure
Offset within BAR	Disabled	Indicates starting position of PCI Configuration Access Structure in given BAR
Structure Length in Bytes	Disabled	Indicates length of PCI Configuration Access Structure
Note: BAR Indicator, Offset within BAR and Structure Length are 0 by default.		

PCIe0 PF1 VIRTIO STRUCTURES		
Enable VIRTIO Capabilities for PF1	On	Exposes VIRTIO Capabilities for VIRTIO Capable Devices
Enable Device Specific Capability for PF1	On	Enables Device Specific Capability for VIRTIO Device on PF1
PCIe0 PF1 COMMON CONFIGURATION STRUCTURE		
BAR Indicator	Disabled	Indicates BAR holding the Common Configuration Structure
Offset within BAR	Disabled	Indicates starting position of Common Configuration Structure in given BAR
Structure Length in Bytes	Disabled	Indicates length of Common Configuration Structure
PCIe0 PF1 NOTIFICATION STRUCTURE		
BAR Indicator	Disabled	Indicates BAR holding the Notification Structure
Offset within BAR	Disabled	Indicates starting position of Notification Structure in given BAR
Structure Length in Bytes	Disabled	Indicates length of Notification Structure
Notify Off Multiplier	Disabled	Indicates Multiplier for queue_notify_off
PCIe0 PF1 ISR STATUS STRUCTURE		
BAR Indicator	Disabled	Indicates BAR holding the ISR STATUS Structure
Offset within BAR	Disabled	Indicates starting position of ISR STATUS Structure in given BAR
Structure Length in Bytes	Disabled	Indicates length of ISR STATUS Structure
PCIe0 PF1 PCI CONFIGURATION ACCESS STRUCTURE		
BAR Indicator	Disabled	Indicates BAR holding the PCI Configuration Access Structure
Offset within BAR	Disabled	Indicates starting position of PCI Configuration Access Structure in given BAR
Structure Length in Bytes	Disabled	Indicates length of PCI Configuration Access Structure
Note: BAR Indicator, Offset within BAR and Structure Length are 0 by default.		
PCIe0 PF2 VIRTIO STRUCTURES		
Enable VIRTIO Capabilities for PF2	On	Exposes VIRTIO Capabilities for VIRTIO Capable Devices
Enable Device Specific Capability for PF2	On	Enables Device Specific Capability for VIRTIO Device on PF2
PCIe0 PF2 COMMON CONFIGURATION STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Common Configuration Structure
Offset within BAR	0x00000000	Indicates starting position of Common Configuration Structure in given BAR

Structure Length in Bytes	0x00000000	Indicates length of Common Configuration Structure
PCIe0 PF2 NOTIFICATION STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Notification Structure
Offset within BAR	0x00000000	Indicates starting position of Notification Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Notification Structure
Notify Off Multiplier	0x00000000	Indicates Multiplier for queue_notify_off
PCIe0 PF2 ISR STATUS STRUCTURE		
BAR Indicator	0	Indicates BAR holding the ISR STATUS Structure
Offset within BAR	0x00000000	Indicates starting position of ISR STATUS Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of ISR STATUS Structure
PCIe0 PF2 DEVICE SPECIFIC STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Device Specific Structure
Offset within BAR	0x00000000	Indicates starting position of Device Specific Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Device Specific Structure
PCIe0 PF2 PCI CONFIGURATION ACCESS STRUCTURE		
BAR Indicator	0	Indicates BAR holding the PCI Configuration Access Structure
Offset within BAR	0x00000000	Indicates starting position of PCI Configuration Access Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of PCI Configuration Access Structure
Note: BAR Indicator, Offset within BAR and Structure Length are 0 by default.		
PCIe0 PF3 VIRTIO STRUCTURES		
Enable VIRTIO Capabilities for PF3	On	Exposes VIRTIO Capabilities for VIRTIO Capable Devices
Enable Device Specific Capability for PF3	On	Enables Device Specific Capability for VIRTIO Device on PF3
PCIe0 PF3 COMMON CONFIGURATION STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Common Configuration Structure
Offset within BAR	0x00000000	Indicates starting position of Common Configuration Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Common Configuration Structure
PCIe0 PF3 NOTIFICATION STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Notification Structure

Offset within BAR	0x00000000	Indicates starting position of Notification Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Notification Structure
Notify Off Multiplier	0x00000000	Indicates Multiplier for queue_notify_off
PCIe0 PF3 ISR STATUS STRUCTURE		
BAR Indicator	0	Indicates BAR holding the ISR STATUS Structure
Offset within BAR	0x00000000	Indicates starting position of ISR STATUS Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of ISR STATUS Structure
PCIe0 PF3 DEVICE SPECIFIC STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Device Specific Structure
Offset within BAR	0x00000000	Indicates starting position of Device Specific Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Device Specific Structure
PCIe0 PF3 PCI CONFIGURATION ACCESS STRUCTURE		
BAR Indicator	0	Indicates BAR holding the PCI Configuration Access Structure
Offset within BAR	0x00000000	Indicates starting position of PCI Configuration Access Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of PCI Configuration Access Structure
Note: BAR Indicator, Offset within BAR and Structure Length are 0 by default.		
PCIe0 PF0 VFs VIRTIO STRUCTURES		
Enable VIRTIO Capabilities for PF0 VFs	On	Exposes VIRTIO Capabilities for VIRTIO Capable Devices
Enable Device Specific Capability for PF0 VFs	On	Enables Device Specific Capability for VIRTIO Device on PF1 VFs
PCIe0 PF0 VFs COMMON CONFIGURATION STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Common Configuration Structure
Offset within BAR	0x00000000	Indicates starting position of Common Configuration Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Common Configuration Structure
PCIe0 PF0 VFs NOTIFICATION STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Notification Structure
Offset within BAR	0x00000000	Indicates starting position of Notification Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Notification Structure

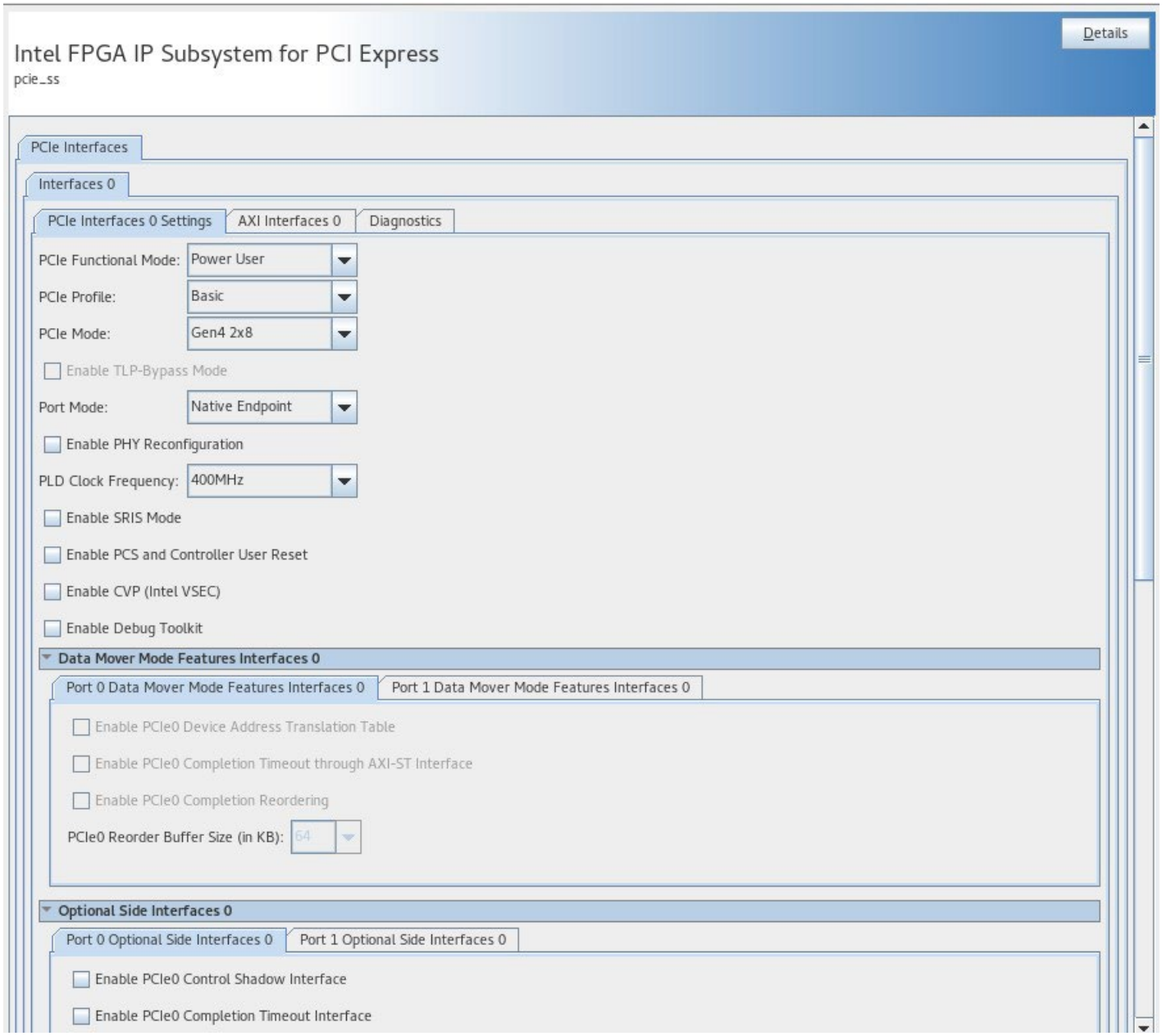
Notify Off Multiplier	0x00000000	Indicates Multiplier for queue_notify_off
PCIe0 PF0 VFs ISR STATUS STRUCTURE		
BAR Indicator	0	Indicates BAR holding the ISR STATUS Structure
Offset within BAR	0x00000000	Indicates starting position of ISR STATUS Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of ISR STATUS Structure
PCIe0 PF0 VFs DEVICE SPECIFIC STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Device Specific Structure
Offset within BAR	0x00000000	Indicates starting position of Device Specific Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Device Specific Structure
PCIe0 PF0 VFs PCI CONFIGURATION ACCESS STRUCTURE		
BAR Indicator	0	Indicates BAR holding the PCI Configuration Access Structure
Offset within BAR	0x00000000	Indicates starting position of PCI Configuration Access Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of PCI Configuration Access Structure
Note: BAR Indicator, Offset within BAR and Structure Length are 0 by default.		
PCIe0 PF1 VFs VIRTIO STRUCTURES		
Enable VIRTIO Capabilities for PF1	On	Exposes VIRTIO Capabilities for VIRTIO Capable Devices
Enable Device Specific Capability for PF1	On	Enables Device Specific Capability for VIRTIO Device on PF2 VFs
PCIe0 PF1 VFs COMMON CONFIGURATION STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Common Configuration Structure
Offset within BAR	0x00000000	Indicates starting position of Common Configuration Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Common Configuration Structure
PCIe0 PF1 VFs NOTIFICATION STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Notification Structure
Offset within BAR	0x00000000	Indicates starting position of Notification Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Notification Structure
Notify Off Multiplier	0x00000000	Indicates Multiplier for queue_notify_off
PCIe0 PF1 Vfs ISR STATUS STRUCTURE		
BAR Indicator	0	Indicates BAR holding the ISR STATUS Structure

Offset within BAR	0x00000000	Indicates starting position of ISR STATUS Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of ISR STATUS Structure
PCIe0 PF1 VFs DEVICE SPECIFIC STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Device Specific Structure
Offset within BAR	0x00000000	Indicates starting position of Device Specific Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Device Specific Structure
PCIe0 PF1 VFs PCI CONFIGURATION ACCESS STRUCTURE		
BAR Indicator	0	Indicates BAR holding the PCI Configuration Access Structure
Offset within BAR	0x00000000	Indicates starting position of PCI Configuration Access Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of PCI Configuration Access Structure
Note: BAR Indicator, Offset within BAR and Structure Length are 0 by default.		
PCIe0 PF2 Vfs VIRTIO STRUCTURES		
Enable VIRTIO Capabilities for PF2	On	Exposes VIRTIO Capabilities for VIRTIO Capable Devices
Enable Device Specific Capability for PF2	On	Enables Device Specific Capability for VIRTIO Device on PF3 VFs
PCIe0 PF2 VFs COMMON CONFIGURATION STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Common Configuration Structure
Offset within BAR	0x00000000	Indicates starting position of Common Configuration Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Common Configuration Structure
PCIe0 PF2 VFs NOTIFICATION STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Notification Structure
Offset within BAR	0x00000000	Indicates starting position of Notification Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Notification Structure
Notify Off Multiplier	0x00000000	Indicates Multiplier for queue_notify_off
PCIe0 PF2 VFs ISR STATUS STRUCTURE		
BAR Indicator	0	Indicates BAR holding the ISR STATUS Structure
Offset within BAR	0x00000000	Indicates starting position of ISR STATUS Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of ISR STATUS Structure
PCIe0 PF2 VFs DEVICE SPECIFIC STRUCTURE		

BAR Indicator	0	Indicates BAR holding the Device Specific Structure
Offset within BAR	0x00000000	Indicates starting position of Device Specific Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Device Specific Structure
PCIe0 PF2 VFs PCI CONFIGURATION ACCESS STRUCTURE		
BAR Indicator	0	Indicates BAR holding the PCI Configuration Access Structure
Offset within BAR	0x00000000	Indicates starting position of PCI Configuration Access Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of PCI Configuration Access Structure
Note: BAR Indicator, Offset within BAR and Structure Length are 0 by default.		
PCIe0 PF3 Vfs VIRTIO STRUCTURES		
Enable VIRTIO Capabilities for PF3	On	Exposes VIRTIO Capabilities for VIRTIO Capable Devices
Enable Device Specific Capability for PF3	On	Enables Device Specific Capability for VIRTIO Device on PF3 VFs
PCIe0 PF3 VFs COMMON CONFIGURATION STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Common Configuration Structure
Offset within BAR	0x00000000	Indicates starting position of Common Configuration Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Common Configuration Structure
PCIe0 PF3 VFs NOTIFICATION STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Notification Structure
Offset within BAR	0x00000000	Indicates starting position of Notification Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of Notification Structure
Notify Off Multiplier	0x00000000	Indicates Multiplier for queue_notify_off
PCIe0 PF3 VFs ISR STATUS STRUCTURE		
BAR Indicator	0	Indicates BAR holding the ISR STATUS Structure
Offset within BAR	0x00000000	Indicates starting position of ISR STATUS Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of ISR STATUS Structure
PCIe0 PF3 VFs DEVICE SPECIFIC STRUCTURE		
BAR Indicator	0	Indicates BAR holding the Device Specific Structure
Offset within BAR	0x00000000	Indicates starting position of Device Specific Structure in given BAR

Structure Length in Bytes	0x00000000	Indicates length of Device Specific Structure
PCIe0 PF3 VFs PCI CONFIGURATION ACCESS STRUCTURE		
BAR Indicator	0	Indicates BAR holding the PCI Configuration Access Structure
Offset within BAR	0x00000000	Indicates starting position of PCI Configuration Access Structure in given BAR
Structure Length in Bytes	0x00000000	Indicates length of PCI Configuration Access Structure
Note: BAR Indicator, Offset within BAR and Structure Length are 0 by default.		

Figure 56. PCIe Interfaces 0 Settings with Power User Functional Mode with Extended PCIe Port Settings Disabled



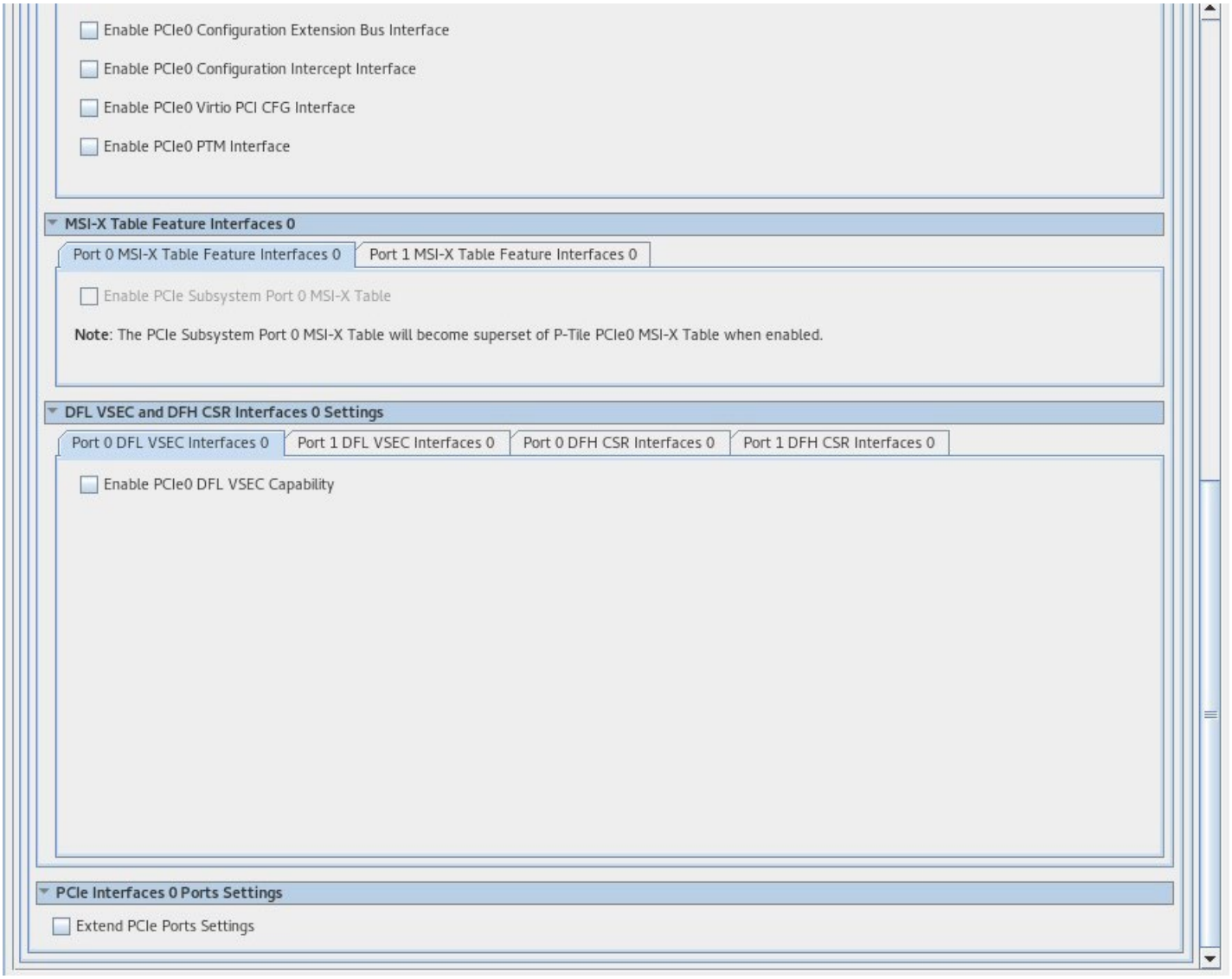

Enable PCIe0 Configuration Extension Bus Interface
Enable PCIe0 Configuration Intercept Interface
Enable PCIe0 Virtio PCI CFG Interface
Enable PCIe0 PTM Interface
MSI-X Table Feature Interfaces 0
Port 0 MSI-X Table Feature Interfaces 0
Port 1 MSI-X Table Feature Interfaces 0
Enable PCIe Subsystem Port 0 MSI-X Table
Note: The PCIe Subsystem Port 0 MSI-X Table will become superset of P-Tile PCIe0 MSI-X Table when enabled.
DFL VSEC and DFH CSR Interfaces 0 Settings
Port 0 DFL VSEC Interfaces 0
Port 1 DFL VSEC Interfaces 0
Port 0 DFH CSR Interfaces 0
Port 1 DFH CSR Interfaces 0
Enable PCIe0 DFL VSEC Capability
PCIe Interfaces 0 Ports Settings
Extend PCIe Ports Settings

Figure 57. PCIe Interfaces 0 Settings with Power User Functional Mode with Extended PCIe Port Settings Enabled – Port 0 Tab

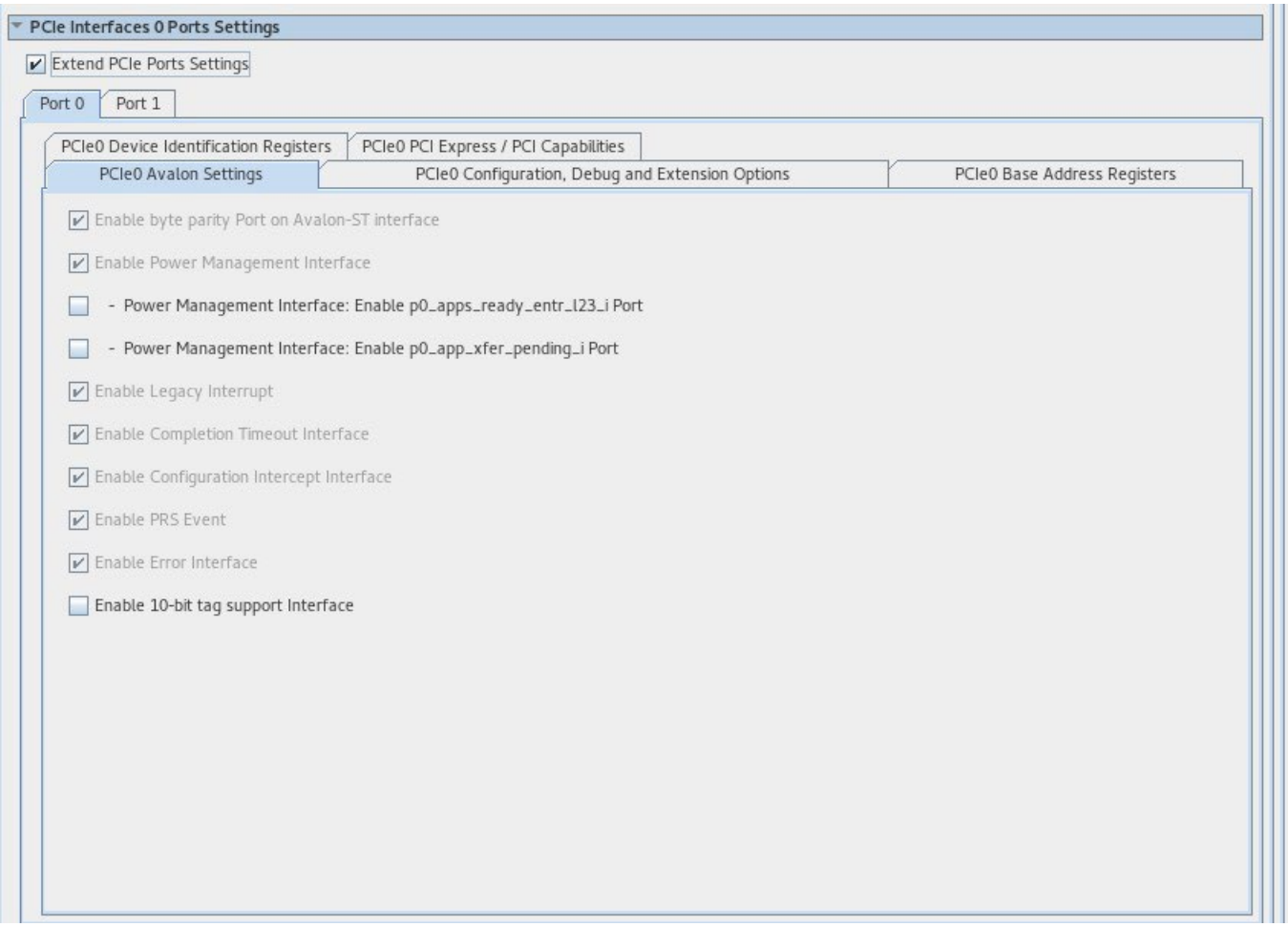
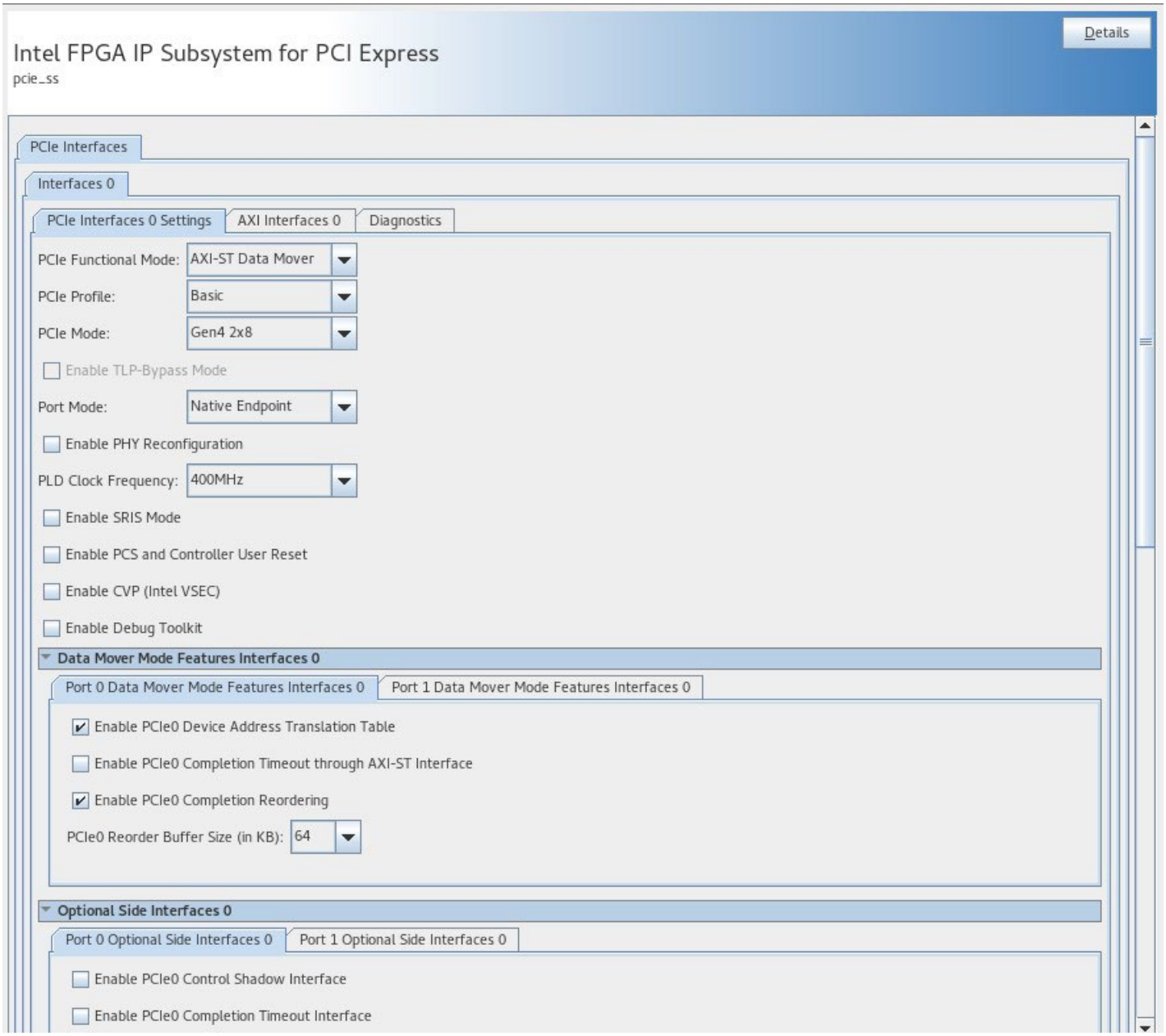


Figure 58. PCIe Interfaces 0 Settings with AXI-ST Data Mover Functional Mode with Extended PCIe Port Settings Disabled



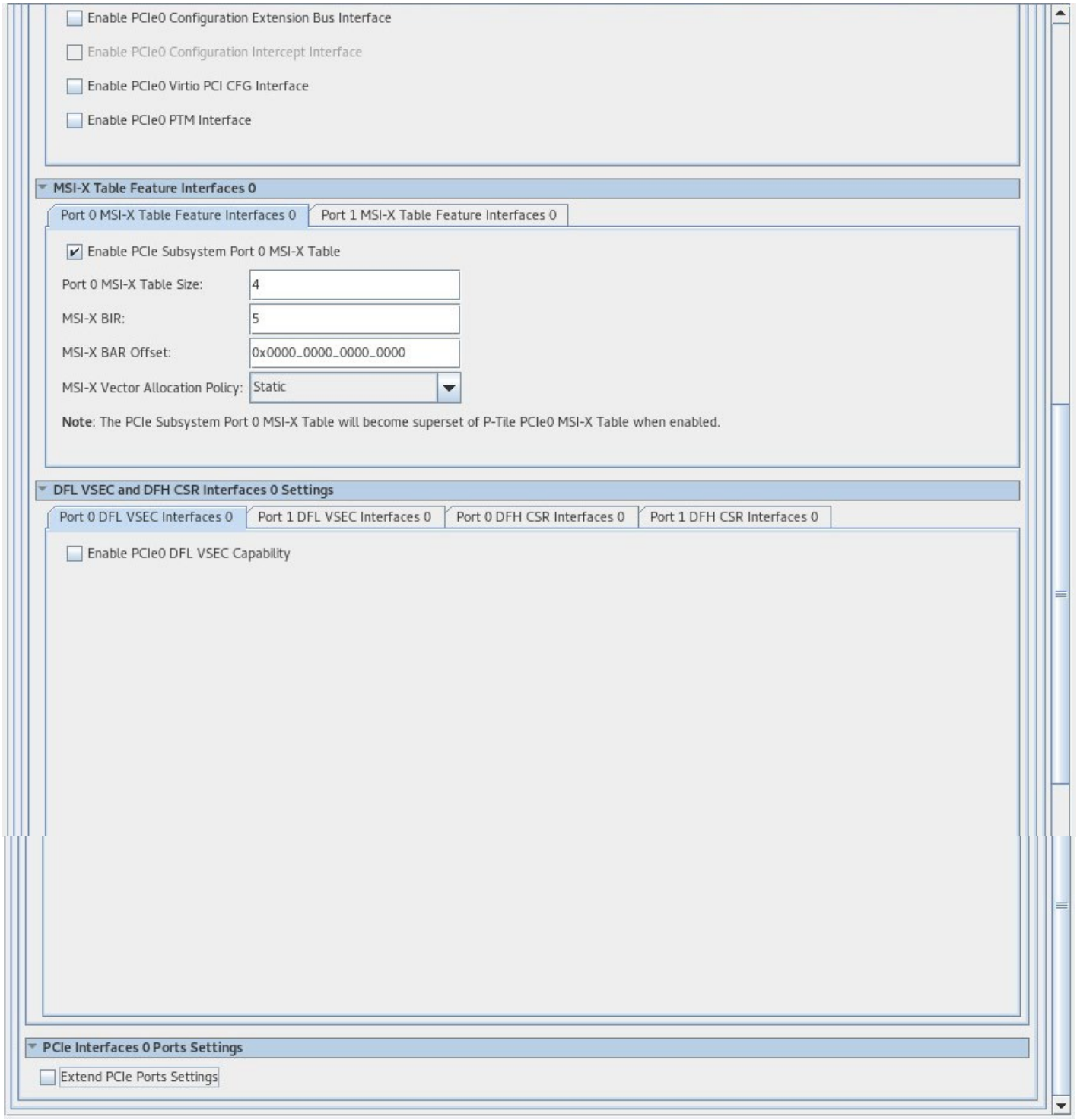

Enable PCIe0 Configuration Extension Bus Interface
Enable PCIe0 Configuration Intercept Interface
Enable PCIe0 Virtio PCI CFG Interface
Enable PCIe0 PTM Interface
MSI-X Table Feature Interfaces 0
Port 0 MSI-X Table Feature Interfaces 0
Port 1 MSI-X Table Feature Interfaces 0
Enable PCIe Subsystem Port 0 MSI-X Table
Port 0 MSI-X Table Size:
4
MSI-X BIR:
5
MSI-X BAR Offset:
0x0000_0000_0000_0000
MSI-X Vector Allocation Policy:
Static
Note: The PCIe Subsystem Port 0 MSI-X Table will become superset of P-Tile PCIe0 MSI-X Table when enabled.
DFL VSEC and DFH CSR Interfaces 0 Settings
Port 0 DFL VSEC Interfaces 0
Port 1 DFL VSEC Interfaces 0
Port 0 DFH CSR Interfaces 0
Port 1 DFH CSR Interfaces 0
Enable PCIe0 DFL VSEC Capability
PCIe Interfaces 0 Ports Settings
Extend PCIe Ports Settings

Figure 59. PCIe Interfaces 0 Settings with AXI-ST Data Mover Functional Mode and Extended PCIe Port Settings Enabled – Port 0 Tab

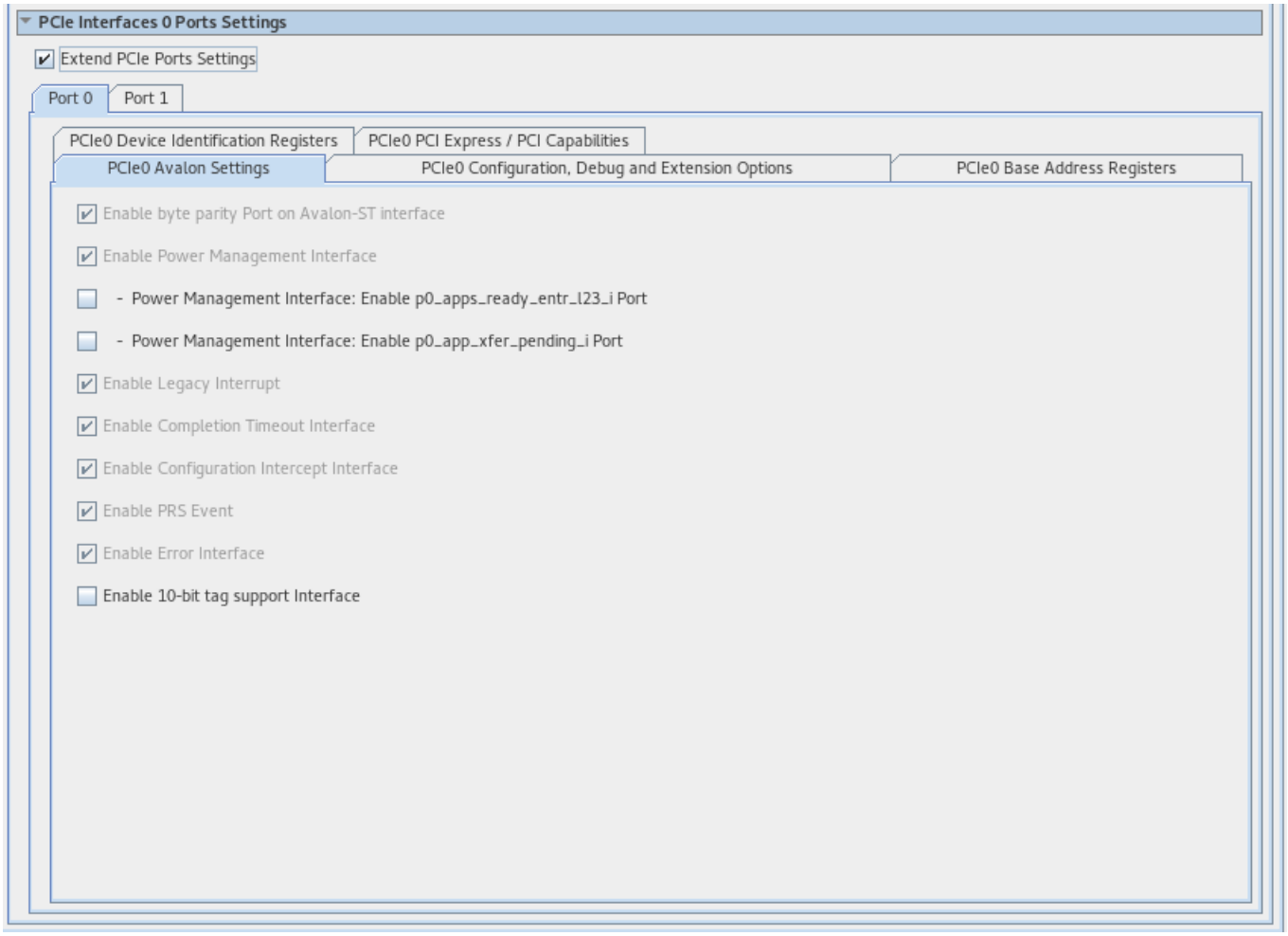


Table 43. PCIe Subsystem Intel FPGA IP Parameters: PCIe Subsystem – AXI Interfaces 0 Tab

Parameter	Default Setting	Parameter Description
AXI Streaming Configuration		
AXI Interfaces 0		
AXI-Lite Clock Frequency (in MHz)	250	Selects the PCIe Subsystem AXI-Lite Operating Clock Frequency
AXI-ST Clock Frequency (in MHz)	470	Selects the PCIe Subsystem AXI-ST Operating Clock Frequency 470 (Gen4) 400 (Gen4) 350 (Gen4) 250 (Gen3)
AXI-ST Interfaces 0		
Port 0 AXI-ST Interface 0 Settings		
PCIe0 AXI-ST Interface Data Bus Width (in Byte)	64	Selects application's AXI streaming data bus width. The interface width is defined in terms of number of Bytes. - 32 - 64 - 128 (not supported)
PCIe0 AXI-ST Interface Data Packing Scheme	Simple	Indicates packing scheme user wants to select for AXI-ST interface - Simple (Currently only supports simple packing mode)
PCIe0 AXI-ST Segment Size (in Byte)	16	Indicates smallest fragment of Data bus when it is divided into multiple segments. This allows start of packet at different positions on data bus - 16
AXI-Lite Interfaces 0		
Port 0 AXI-Lite Interface 0 Settings		
PCIe0 AXI-Lite Responder Interface Data Bus Width (in Byte)	4	Selects application's AXI-Lite Responder interface's data bus width. The interface width is defined in terms of number of Bytes 4 8

Figure 60. AXI Interface 0 and AXI-ST Interface 0 Settings Tab

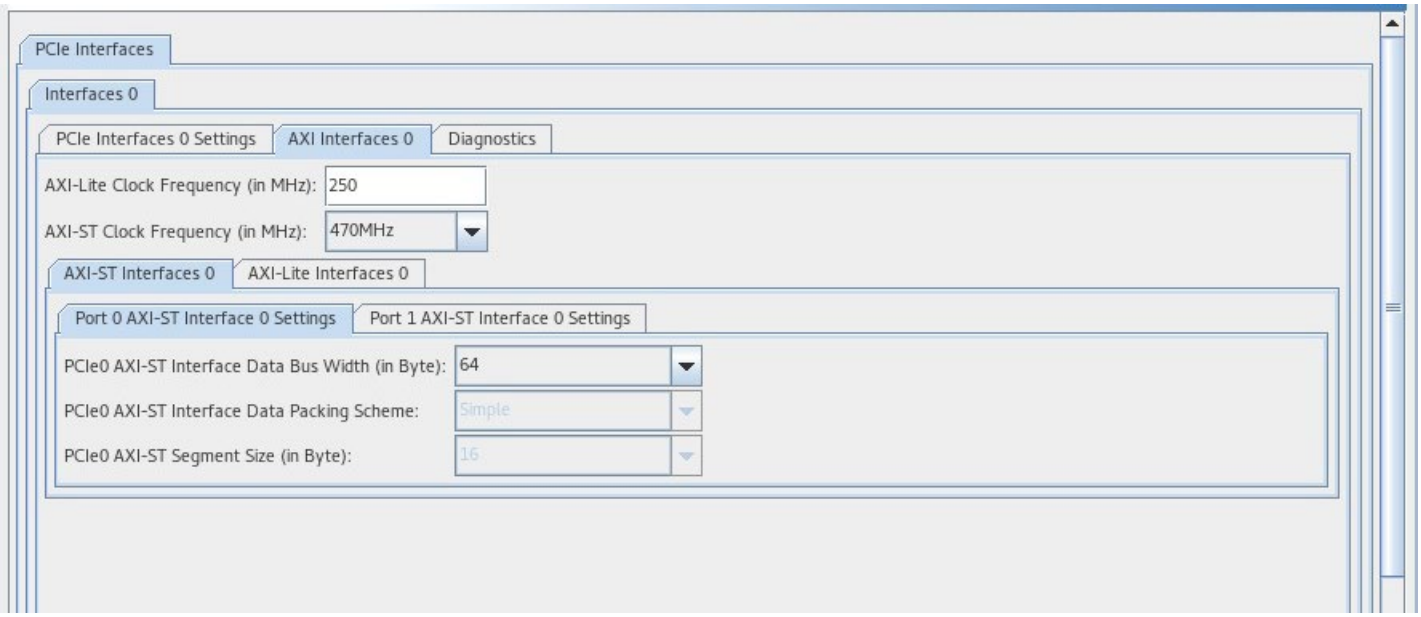


Figure 61. AXI Interface 0 and AXI-Lite Interface 0 Settings Tab

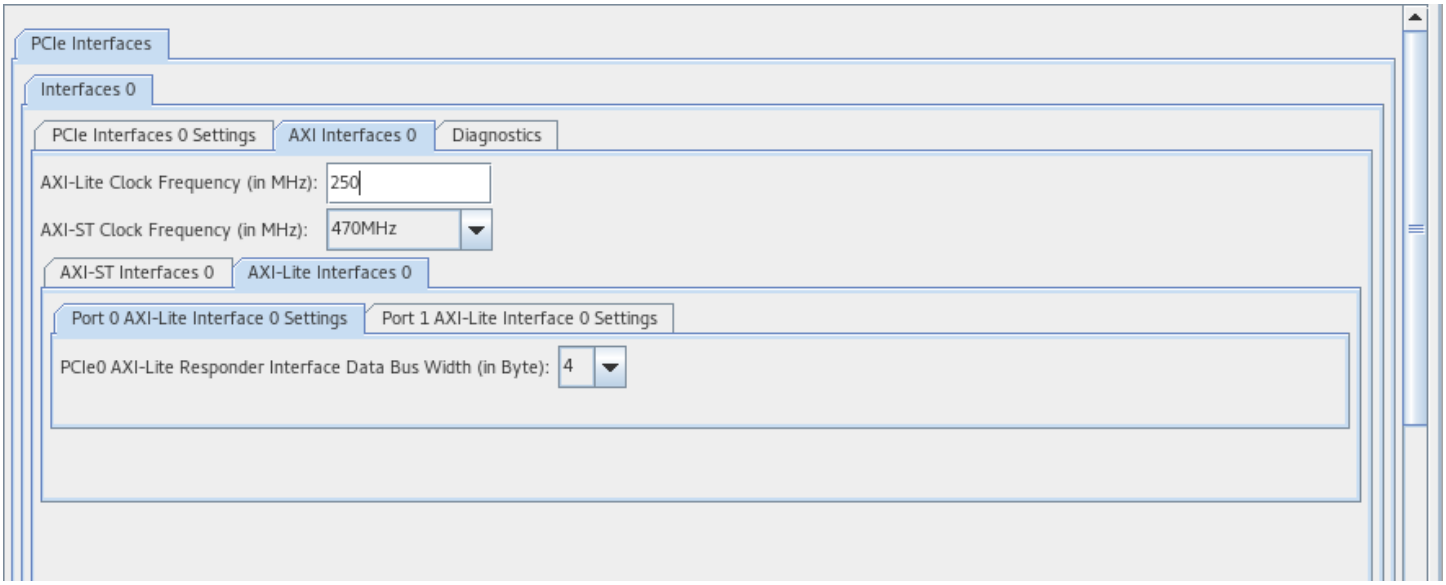
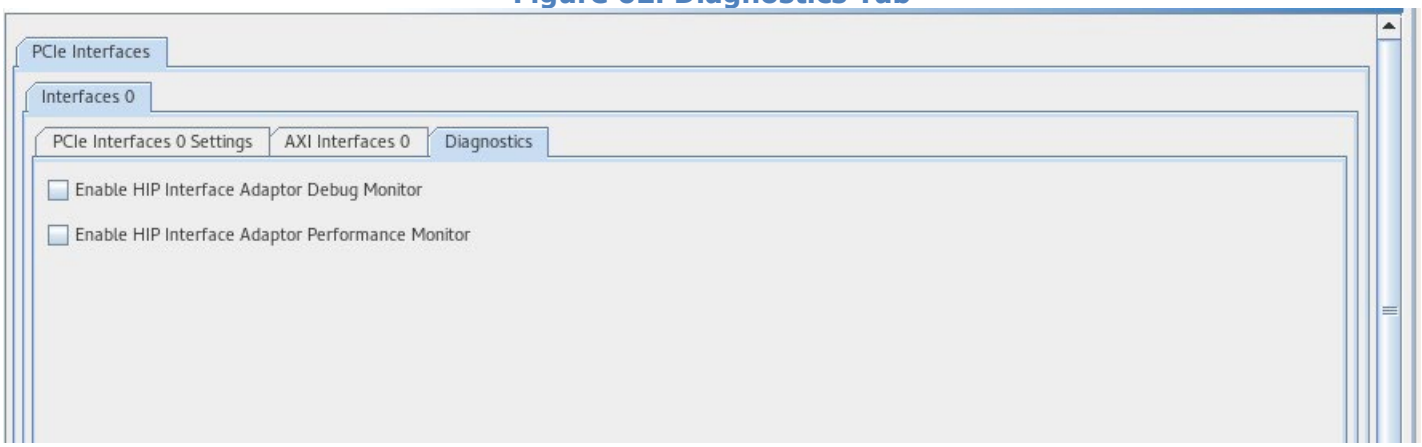


Table 43. PCIe Subsystem Intel FPGA IP Parameters: PCIe Subsystem – Diagnostics Tab

Parameter	Default Setting	Parameter Description
Enable HIP Interface Adaptor Debug Monitor	False	Selects the PCIe Subsystem AXI-Lite Operating Clock Frequency True False
Enable HIP Interface Adaptor Performance Monitor	False	Selects the PCIe Subsystem AXI-ST Operating Clock Frequency True False

Figure 62. Diagnostics Tab



6. Interfaces and Signals

This section focuses mainly on the signal interfaces that the P-Tile Subsystem IP for PCIe uses to communicate with the Application Layer in the FPGA fabric core. It also briefly covers the Serial Data Interface, which allows the IP to communicate with the link partner across the PCIe link.

6.1. Overview

You can determine which core each interface in this section belongs to by looking at the prefixes in the signal names:

p0: x16 core

p1: x8 core

p2: x4 core

p3: x4 core

The Intel FPGA IP Subsystem for PCI Express Top-Level Signals shows the top-level signals of this IP. Note that the signal names in the figure will get the appropriate prefix p<n> (where n = 0, 1, 2, 3) depending on which of the supported configurations (1x16, 2x8 or 4x4) the PCIe SS IP is in.

As an example, the ss_app_st_rx_tdata bus can take on the following names:

In the 1x16 configuration, only the x16 core is active. In this case, this bus appears as p0_ss_app_st_rx_tdata [511:0].

In the 2x8 configuration, both the x16 core and x8 core are active. In this case, this bus is split into p0_ss_app_st_rx_tdata[511:0] and p1_ss_app_st_rx_tdata [511:0].

The only cases where the interface signal names do not get the pn prefixes are the interfaces that are common for all the cores, like the PHY reconfiguration interface, clocks and resets. For example, there is only one xcvr_reconfig_clk that is shared by all the cores.

You can enable the PHY reconfiguration interface from the PCIe Interface Settings in the Parameter editor.

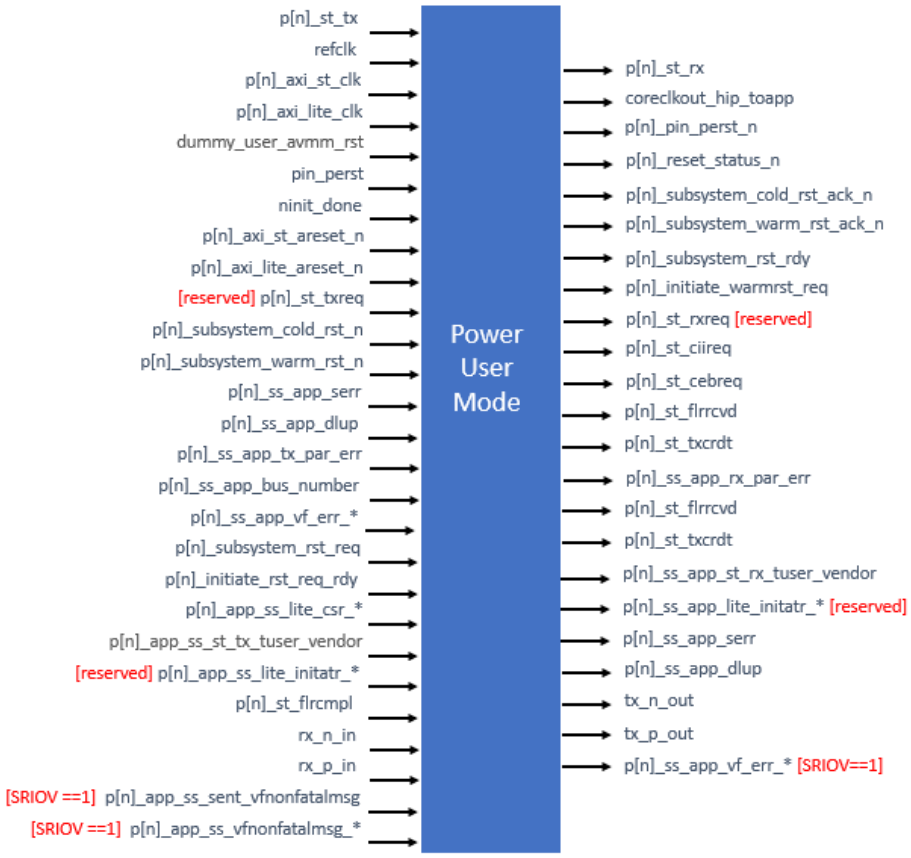
Each of the cores has its own AXI-ST interface to the user logic. The number of IP-to-User Logic interfaces exposed to the FPGA fabric are different based on the configuration modes:

Table 44. IP to FPGA Fabric Interfaces Summary

Mode	AXI-ST Interface Count	Data Width in Bytes (each Interface)	AXI-ST Clock Frequency (MHz)	PLD Clock Frequency (coreclkout_hip_toapp)(MHz)
Gen4 1x16	1	32 64 128*	470 400 350 250	500
Gen4 1x8	1	32 64 128*	470 400 350 250	500
Gen4 2x8	2	32 64 128*	470 400 350 250	500
Gen3 1x16	1	32 64 128*	470 400 350 250	250
Gen3 2x8	2	32 64 128*	470 400 350 250	250

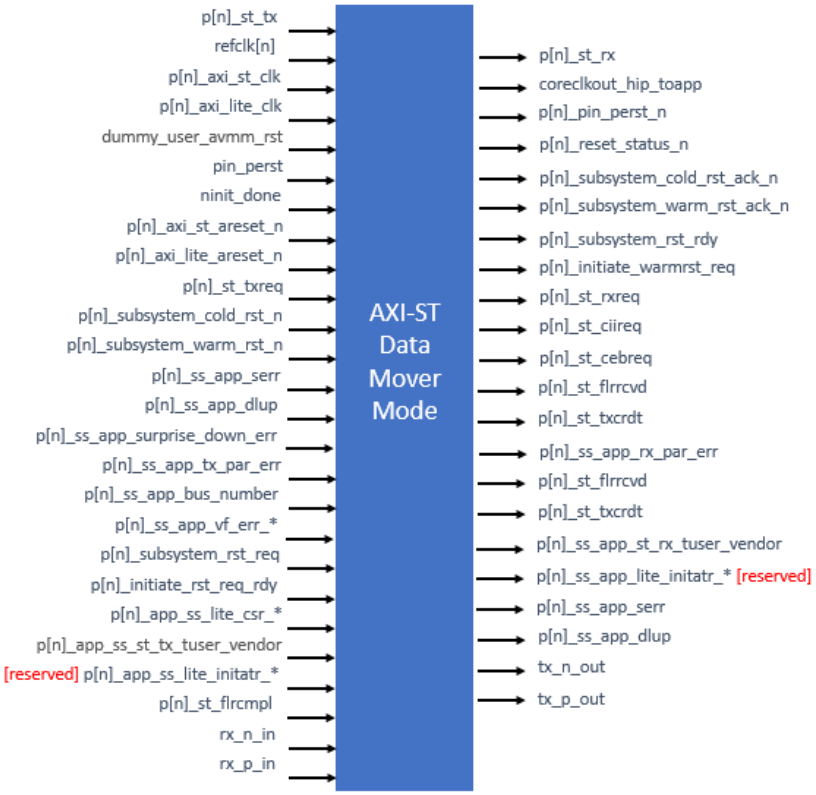
*May be available in future Quartus release

Figure 62. Intel FPGA IP Subsystem for PCI Express Top-Level Signals in Power User mode



Note: Reserved signals are not supported in the current release.

Figure 63. Intel FPGA IP Subsystem for PCI Express Top-Level Signals in Data Mover Mode



Note: Reserved signals are not supported in the current release.

The table below shows the variables that are used to define the bus indices for top level signal busses shown in the top-level block diagram above. The values of these variables change depending on which configuration is active (1x16, 2x8).

Table 45. Variables Used in the Bus Indices

Variable	1 x16 configuration	1 x8 configuration	2 x8 configuration
n	1	1	2
b	16	8	16
w	4	2	2
p	6	3	3

6.2. Clocks and Resets

6.2.1. Interface Clock Signals

Table 46. Interface Clock Signals

Signal Name	Direction	EP/RP/BP	Description
refclk[1:0]	I	EP/RP/BP	These are the input reference clocks for the IP core. These clocks must be free-running. For more details on how to connect these clocks, refer to the section Clock Sharing in Bifurcation Modes. EP/RP/BP 100 MHz ± 300 ppm
p<n>_axi_st_clk	I	EP/RP/BP	Global clock signal for AXI -ST interface. All AXI-ST signals are sampled on the rising edge of this clk. This clock drives main data path Gen4: 470/400/350/250 MHz (32-, 64 – byte width) Gen3: 470/400/350/250 MHz (32-, 64- byte width)
axi_lite_clk	I	EP/RP/BP	The global clock signal for AXI-Lite interface. All AXI-Lite signals are sampled on the rising edge of p<n>_axi_lite_clk. This clock drives sideband and CSR interfaces in design. Frequency: 100-250MHz (250MHz default)
p<n>_axi_mm_clk	I	EP/RP/BP	Global clock signal for AXI -MM interface. All AXI-MM signals are sampled on the rising edge of this clk. This clock drives AXI MM data path.
Coreclkput_hip_toapp	O	EP/RP/BP	The coreclkout_hip output of Hard IP drives this clock. Application can use this clock to generate PCIe SS clocks. Gen4: 500MHz Gen3: 250MHz Gen2/Gen1: Gen1/Gen2 is supported only via link down-training and not natively. Hence, the coreclkout_hip clock frequency depends on the configuration you choose in the IP Parameter Editor. For example, if you choose a Gen3 configuration, the application clock frequency is 250MHz.

6.2.2. Interface Reset Signals

Table 47. Interface Reset Signals

Signal Name	Direction	Type	Description
p<n>_Subsystem_cold_rst_n	I	Could be implemented as synchronous or asynchronous reset.	Subsystem global reset. Resets sticky register bits. Active low.
p<n>_Subsystem_warm_rst_n	I	Could be implemented as synchronous or asynchronous reset.	Subsystem warm reset. Does not reset sticky register bits. Active low.
p<n>_Subsystem_cold_rst_ack_n	O	Asynchronous	Handshake signal. Indicates cold reset action is completed by Subsystem.
p<n>_Subsystem_warm_rst_ack_n	O	Asynchronous	Handshake signal. Indicates warm reset action is completed by Subsystem.
p<n>_axi_st_areset_n	I	The reset signal can be asserted asynchronously, but deassertion must be synchronous after the rising edge of axi_st_clk.	AXI-Streaming main datapath reset. Active-LOW reset signal. Used to reset the AXI-ST datapath interface
p<n>_axi_lite_areset_n	I	The reset signal can be asserted asynchronously, but deassertion must be synchronous after the rising edge of axi_lite_clk.	AXI-Lite reset. Active-LOW reset signal. Used to reset the AXI Lite interface
p<n>_axi_mm_areset_n	I	The reset signal can be asserted asynchronously, but deassertion must be synchronous after the rising edge of axi_mm_clk.	AXI-MM reset. Active-LOW reset signal. Used to reset the AXI-MM interface

Signal Name	Direction	Type	Description
p<n>_Subsystem_rst_req	I	Asynchronous	Reset entry indication from Central Reset Sequencer block in SOC Subsystem quiesces the blocks in design upon receiving this request and sends acknowledgement back when block is ready for reset entry.
p<n>_Subsystem_rst_rdy	O	Asynchronous	Ready signal for reset entry indication from Subsystem to Central Reset Sequencer block
p<n>_initiate_warmrst_req	O	Asynchronous	Warm Reset entry required indication from SIP SRC block to Central Reset Sequencer Initiator block cannot issue new reset entry request until previous reset sequence (entire reset operation) is completed
p<n>_initiate_rst_req_rdy	I	Asynchronous	Indicates Central Reset Sequencer block has accepted initiation request and will start issuing resets
Reset_status_n	O	Synchronous to coreclkout_hip of Hard IP	Active low signal. When asserted, indicates HARD IP is in reset state. When asserted, will continue to stay asserted until pin perstn is deasserted and HARD IP is out of reset state. The application logic can use this signal to drive its reset network. The reset_status_n output of HIP drives this signal.
dummy_user_avmm_rst	I	-	Reset signal. You must tie it to ground

Note: The user must implement the user reset sequencer in application user logic and follow the assertion and deassertion sequence for graceful entry and exit for each of the resets (cold, warm etc). Below table indicates the signals/blocks used for each type of reset

Table 48. Signals and blocks used for reset type

Reset Type	Signals/Blocks under reset
Cold Reset	• *Subsystem_cold_rst_n* and *Subsystem_warm_rst_n* will be asserted • Bus resets (AXI-ST/AXI-MM/AXI-Lite) will be asserted • HIP will undergo reset
Warm Reset (e.g., LTSSM Hot reset)	• *Subsystem_cold_rst_n* will not be asserted • Bus resets (AXI-ST/AXI-MM/AXI-Lite) will be asserted also • HIP will undergo reset also

Expected reset isolation requirement for reset domain crossings are shown in the table below:

Table 49. Reset Isolation requirement for reset domain crossings

Column: Source Row: Destination	Cold Reset	HIP Reset	Warm Reset	AXI-ST/MM Reset	AXI-Lite Reset
Cold Reset	N/A	No	Yes	No	No
HIP Reset	No	N/A	No	No	No
Warm Reset	No	No	N/A	No	No
AXI-ST/MM Reset	No	No	No	N/A	No
AXI-Lite Reset	No	No	No	No	N/A

No: No reset isolation required for Column->Row Reset Domain Crossing
Yes: Reset isolation required for Column->Row Reset Domain Crossing
N/A: Not applicable since same Reset Domain Crossing

Note: In Endpoint mode, Subsystem warm reset could be asserted without Subsystem cold reset in scenarios such LTSSM Hot Reset

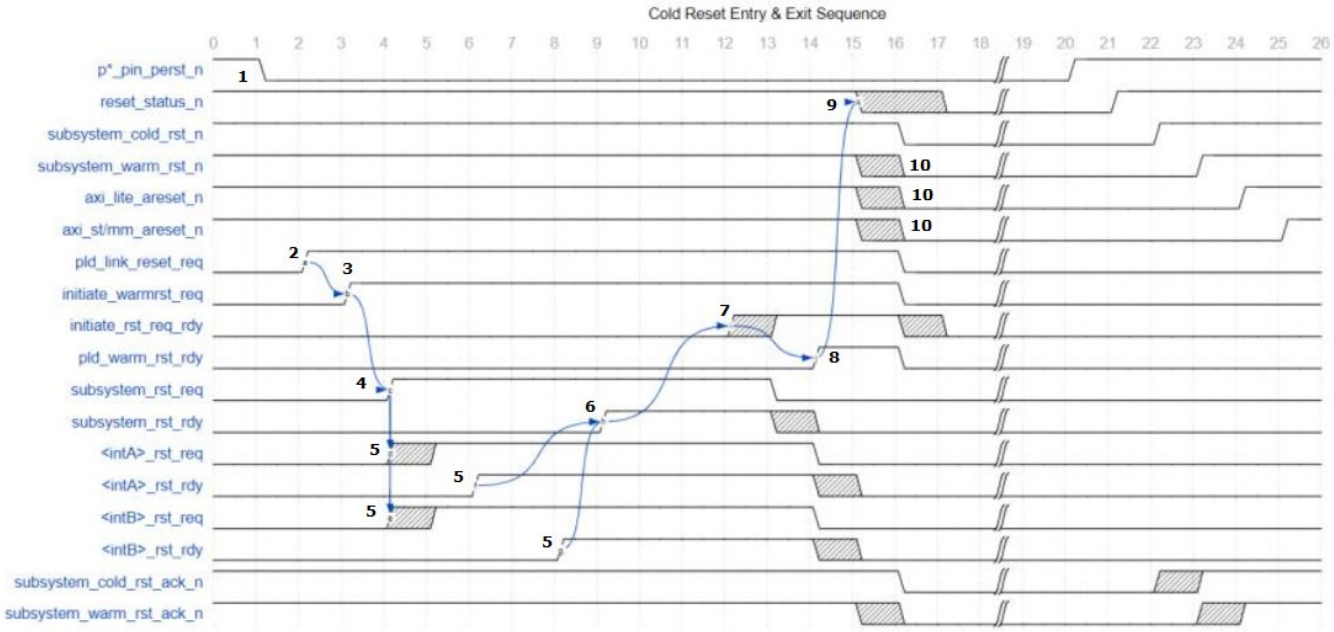
Cold Reset Entry and Exit Sequence

Cold Reset Entry sequence is shown below.

1. Cold reset is initiated by the assertion of Hard IP's *p*_pin_perst_n*
2. Hard IP asserts *pld_link_reset_req* to the Subsystem
3. Subsystem notifies the user reset sequencer by asserting *initiate_warmrst_req*
4. User reset sequencer will then assert *Subsystem_rst_req*

5. Subsystem will sequence its internal blocks for reset entry (intA_rst_req, intB_rst_req, intA_rst_rdy, intB_rst_rdy, …)
6. Subsystem asserts *Subsystem_rst_rdy* to user reset sequencer, indicating the Subsystem internal blocks are ready for reset
7. User reset sequencer acknowledges to Subsystem that it is ready for reset by asserting *initiate_rst_req_rdy*
8. Subsystem then asserts pld_warm_rst_rdy to Hard IP
9. Hard IP asserts *reset_status_n* indicating the application logic needs to be in reset
10. User reset sequencer asserts *Subsystem_cold_rst_n, Subsystem_warm_rst_n* and AXI bus resets

Figure 64. Cold reset entry and exit sequence timing diagram



Warm Reset Entry & Exit Sequence

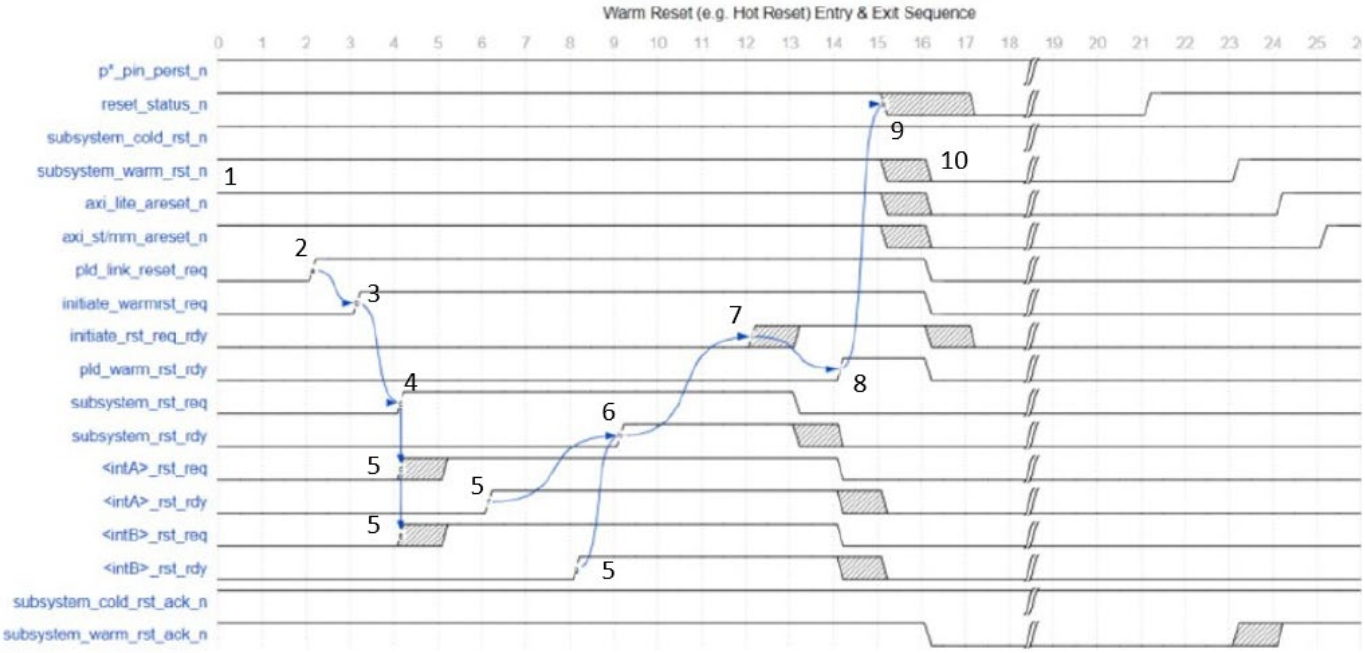
Warm Reset Entry sequence is shown below:

1. Warm reset is initiated by the assertion of Hard IP event, e.g., Hot reset
2. Hard IP asserts *pld_link_reset_req* to the Subsystem
3. Subsystem notifies the user reset sequencer by asserting *initiate_warmrst_req*
4. User reset sequencer will then assert *Subsystem_rst_req*
5. Subsystem will sequence its internal blocks for reset entry (intA_rst_req, intB_rst_req, intA_rst_rdy, intB_rst_rdy, …)

6. Subsystem asserts *Subsystem_rst_rdy* to user reset sequencer, indicating the Subsystem internal blocks are ready for reset
7. User reset sequencer acknowledges to Subsystem that it is ready for reset by asserting *initiate_rst_req_rdy*
8. Subsystem then asserts pld_warm_rst_rdy to Hard IP
9. Hard IP asserts *reset_status_n* indicating the application logic needs to be in reset
10. User reset sequencer asserts *Subsystem_warm_rst_n* and AXI bus resets

Note: Warm Reset flow is similar to Cold Reset flow, with the exception that p*_pin_perst_n and Subsystem_cold_rst_n are not asserted for warm reset

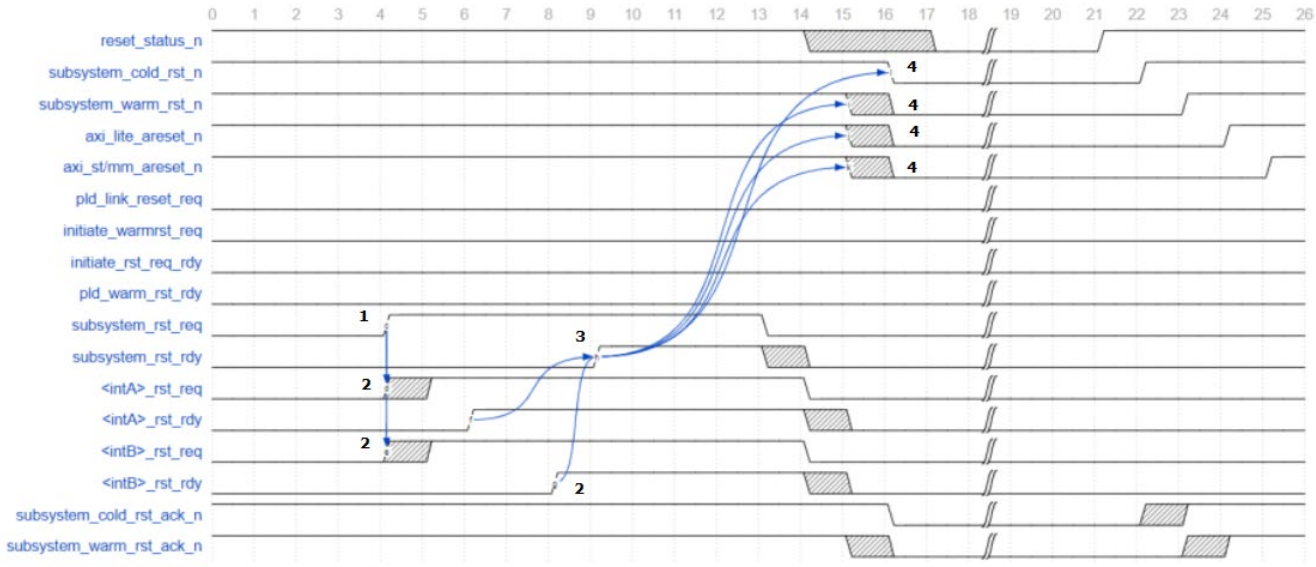
Figure 65. Warm reset entry and exit sequence timing diagram



User reset sequencer initiated Cold Reset Entry and Exit Sequence

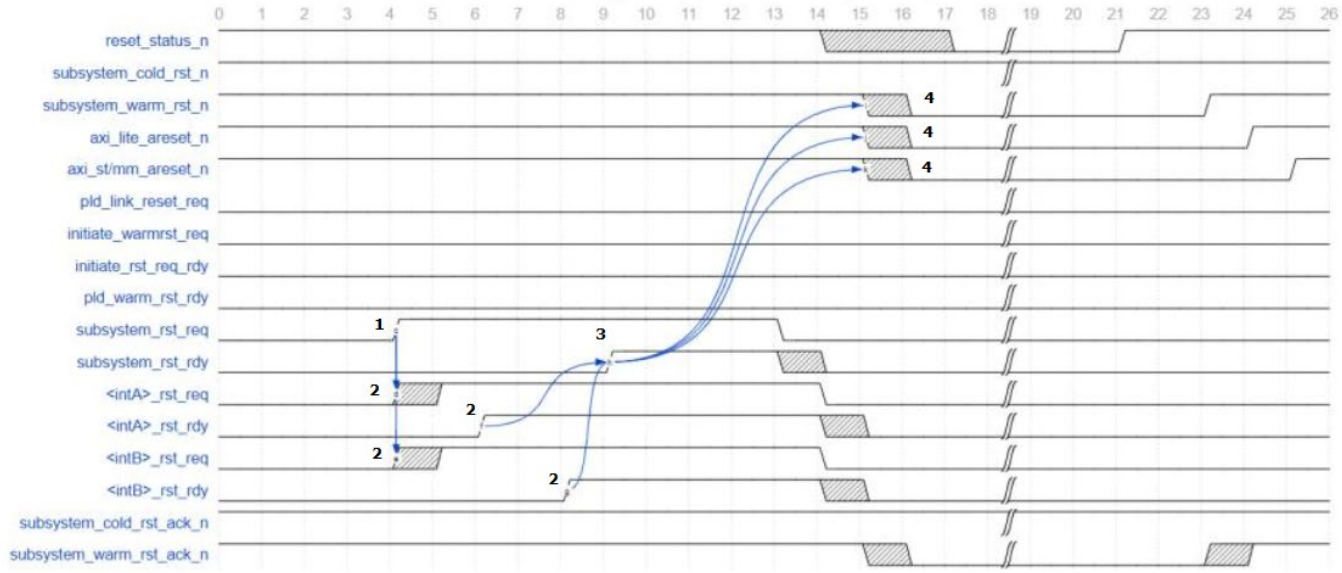
1. Cold reset is initiated by the user reset sequencer by the assertion of the Subsystem_rst_req.
2. Subsystem sequences its internal blocks for reset entry (intA_rst_req, intB_rst_req, intA_rst_rdy, intB_rst_rdy, …)
3. Subsystem asserts *Subsystem_rst_rdy* to user reset sequencer, indicating the Subsystem internal blocks are ready for reset
4. User reset sequencer asserts *Subsystem_cold_rst_n, Subsystem_warm_rst_n* and AXI bus resets

Figure 66. User reset sequencer initiated cold reset entry and exit sequence timing diagram



User reset sequencer triggered Warm Reset flow is the same as the user reset sequencer triggered Cold Reset flow, with the exception that the *Subsystem_cold_rst_n* will not be asserted for this flow

Figure 67. User reset sequencer initiated warm reset entry and exit sequence timing diagram



6.3. Application Packet Interface

6.3.1. Application Packet Receive Interface (st_rx)

Application Packet Receive Interface (st_rx)

The packet received from the link is presented to application logic on this interface. The interface supports data width of 32 bytes (256 bits), 64 bytes (512 bits) and 128 bytes (1024 bits). The BAR, function number of TLP and prefix signals are sent in line with data.

For downstream transactions, data mover will forward the completions on AXI-ST_RX interface and posted and non-posted on AXI-ST_RXREQ interface for the application that use the data mover facility. For application that does not use the data mover facility, all the cycles will pass down at AXI-ST_REQ interface in order. This is to avoid data mover completions creates blocking to non-data mover transactions.

Note: The 128-bytes (1024 bits) width support may be available in a future release of Quartus

Table 50. APP AXI ST RX Interface

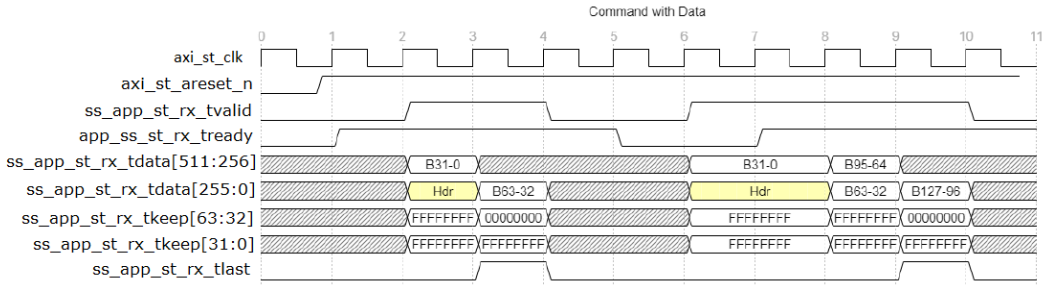
Signal Name	Direction	Clock Domain	Description
ss_app_st_rx_tvalid	Output	axi_st_clk	ss_app_st_rx_tvalid indicates that the source is driving a valid transfer
app_ss_st_rx_tready	Input	axi_st_clk	app_ss_st_rx_tready indicates that the sink can accept a transfer in the current cycle

			readyLatency parameter defined in Avalon spec shall be supported. By default, the value is '0'.
ss_app_st_rx_tdata[(DWIDTH-1):0]	Output	axi_st_clk	Data interface with configurable width specified by DWIDTH parameter. Default DWIDTH : 512 **Note:** The 128-bytes (1024 bits) width support may be available in a future release of Quartus
ss_app_st_rx_tkeep[(DWIDTH/8-1):0]	Output	axi_st_clk	A byte qualifier used to indicate whether the content of the associated byte is valid The invalid bytes are allowed only during ss_app_st_tx_tlast cycle The sparse ss_app_st_tx_tkeep is not allowed
ss_app_st_rx_tlast	Output	axi_st_clk	Indicates End of Data/Command Transmission
ss_app_st_rx_tuser_last_segment<Number>	Output	axi_st_clk	Indicates Packet End position on tdata bus Only applicable with Variable Header Position and Compact Packing scheme Number of ports equals to NUM_OF_SEG <Number> equals to segment number Note: Variable and compact packing not currently supported
ss_app_st_rx_tuser_vendor[1:0]	Output	axi_st_clk	Vendor Specific Tuser bits Indicates Header Format. Don't Care in Power User mode [0] - Indicates Header format of First packet in a cycle 0 - Power User mode header format 1 - Data Mover header format [1] - Indicates Header format of Second packet in a cycle 0 - Power User mode header format 1 - Data Mover header format This signal is never asserted when the TLP has no payload. (Dont care in Power User mode)

The timing diagrams provided in the following sections are for the simple packing scheme only. The figure below shows a timing diagram for command with data. The completion, memory write, messages and configuration write commands fall under the command with data category. The first command transfers a payload of 64 Bytes. The sink is ready to accept a command at clock cycle 1 but the source does not have any command to transfer in that same cycle. The source starts transfer in the next cycle.

The second command transfers a payload of 128 Bytes. Here sink is not ready to accept a command when source has asserted valid. The source holds the information on the bus until it observes ready from the sink.

Figure 68. Timing Diagram for Simple Packing Scheme - Command With Data

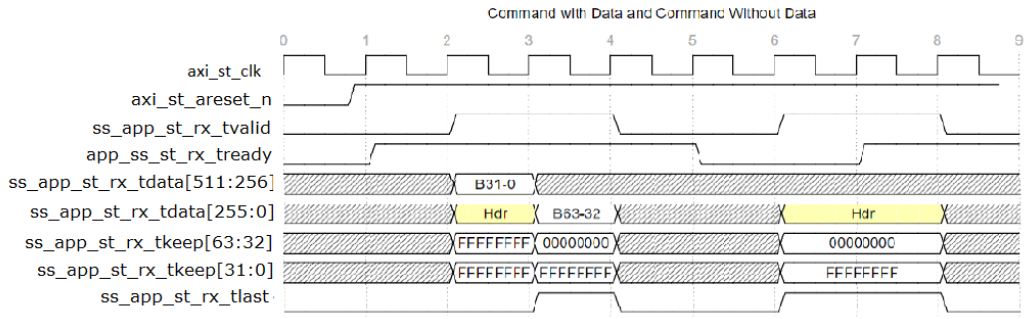


The figure below shows timing diagram for command with data followed by command without data. The completion, memory write, messages and configuration write commands fall under command with data category. The memory read, configuration read, messages without data and completion without data fall under command without data category.

The first command transfers a payload of 64 Bytes. The sink is ready to accept a command at clock cycle 1 but the source does not have any command to transfer in that same cycle. The source starts transfer in the next cycle.

The second command is a command without data. Here sink is not ready to accept a command when source has asserted valid. The source holds the information on the bus until it observes ready from the sink.

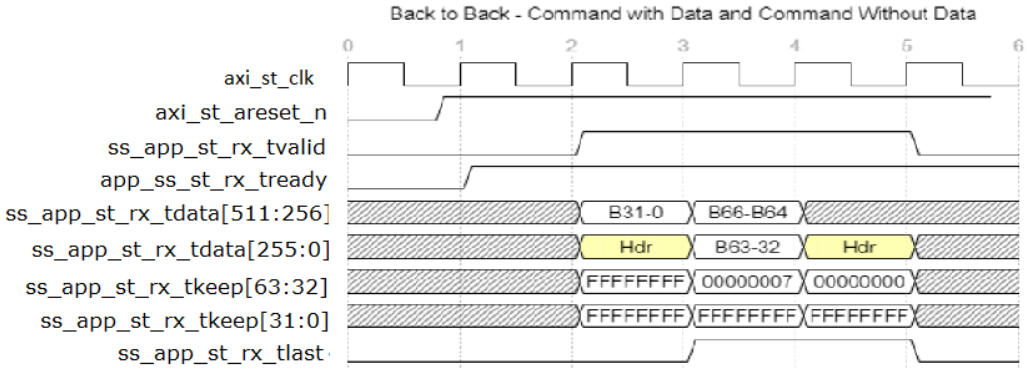
Figure 69. Timing Diagram for Simple Packing Scheme - Command With Data Followed by Command Without Data



The first command transfers the payload of 67 Bytes. Note tkeep during tlast has partial ones, but these ones are contiguous, sparse tkeep is not allowed. The partial tkeep is allowed only on tlast cycle.

The second command is a command without data.

Figure 70. Timing Diagram for Simple Packing Scheme – Back-to-Back Commands With Data and Without Data



6.3.2. Application Request Interface (st_rxreq)

This interface is available in AXI Data Mover mode only. For downstream transactions, data mover will forward the completions on AXI-ST_RX interface and posted and non-posted on AXI-ST_RXREQ interface for the application that use the data mover facility. For application that does not use the data mover facility, all the cycles will pass down at AXI-ST_REQ interface in order. This is to avoid data mover completions creates blocking to non-data mover transactions.

Table <>. APP AXI ST Request TX Interface

Signal Name	Direction	Clock Domain	Description

ss_app_st_rxreq_tvalid	Output	axi_st_clk	ss_app_st_rx_tvalid indicates that the source is driving a valid transfer
app_ss_st_rxreq_tready	Input	axi_st_clk	app_ss_st_rx_tready indicates that the sink can accept a transfer in the current cycle readyLatency parameter defined in AXI Spec shall be supported. By default, the value is '0'.
ss_app_st_rxreq_tdata [255:0]	Output	axi_st_clk	Data interface
ss_app_st_rxreq_tkeep[31:0]	Output	axi_st_clk	Byte qualifier used to indicate whether the content of the associated byte is valid The invalid bytes are allowed only during ss_app_st_tx_tlast cycle The sparse ss_app_st_tx_tkeep is not allowed
ss_app_st_rxreq_tlast	Output	axi_st_clk	Indicates End of Data/Command Transmission
ss_app_st_rxreq_tuser_last_segment<number>	Output	axi_st_clk	Indicates Packet End position on tdata bus Only applicable with Variable Header Position and Compact Packing scheme Number of ports equals to NUM_OF_SEG <Number> equals to segment number Note: Variable and compact packing not currently supported
ss_app_st_rxreq_tuser_vendor[1:0]	Output	axi_st_clk	Vendor Specific Tuser bits - Header Format Indicates Header Format. Don’t Care in Power user Mode Bit[0] - Indicates Header format of First packet in a cycle

			0 - Power user mode header format 1 - Data mover header format Bit[1] - Indicates Header format of Second packet in a cycle 0 - Power user mode header format
ss_app_st_rxreq_tuser_vendor[2]	Output	axi_st_clk	Vendor Specific Tuser bits - Transaction Abort Indicates current transmission is terminated abnormally because of error condition It's one clock cycle pulse and goes high with Tlast This signal is never asserted when the TLP has no payload. Note: Don't care in Power User Mode

6.3.3. Application Packet Transmit Interface (st_tx)

The outbound packet towards the link from application side is transmitted through this interface. The interface supports data widths of 32 bytes (256 bits), 64 bytes (512 bits) and 128 bytes (1024 bits). The function number of TLP and prefix signals are sent in line with data.

Note: The 128-bytes (1024 bits) width support may be available in a future release of Quartus.

Table 51. APP AXI ST TX Interface

Signal Name	Direction	Clock Domain	Description
app_ss_st_tx_tvalid	Input	axi_st_clk	app_ss_st_tx_valid indicates that the source is driving a valid transfer
ss_app_st_tx_tready	Output	axi_st_clk	app_axi_st_tx_ready indicates that the sink can accept a transfer in the current cycle

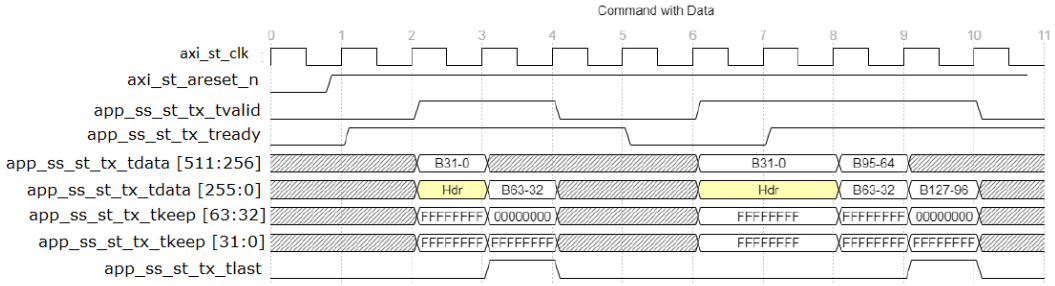
			readyLatency parameter defined in Avalon spec shall be supported. By default, the value is '0'.
app_ss_st_tx_tdata[(DWIDTH-1):0]	Input	axi_st_clk	Data Interface with configurable width specified by DWIDTH parameter. Default DWIDTH : 512 **Note:** The 128-bytes (1024 bits) width support may be available in a future release of Quartus
app_ss_st_tx_tkeep[(DWIDTH/8-1):0]	Input	axi_st_clk	A byte qualifier used to indicate whether the content of the associated byte is valid The invalid bytes are allowed only during app_axi_st_tx_tlast cycle The sparse app_axi_st_tx_tkeep is not allowed
app_ss_st_tx_tlast	Input	axi_st_clk	Indicates End of Data/Command Transmission
ss_app_st_tx_tuser_last_segment<Number>	Input	axi_st_clk	Indicates Packet End position on tdata bus Only applicable with Variable Header Position and Compact Packing scheme Number of ports equals to NUM_OF_SEG <Number> equals to segment number Note: Variable and compact packing not currently supported
ss_app_st_tx_tuser_vendor[1:0]	Input	axi_st_clk	Vendor Specific Tuser bits Indicates Header Format. Don't Care in Power User mode Bit 0 - Indicates Header format of First packet in a cycle 0 - Power User mode header format 1 - Data Mover header format Bit 1 - Indicates Header format of Second packet in a cycle 0 - Power User mode header format 1 - Data Mover header format.

The timing diagrams provided in the following sections are for the simple packing scheme only. The figure below shows timing diagram for command with data. The completion, memory write, messages and configuration write commands fall under

command with data category. The first command transfers a payload of 64 Bytes. The sink is ready to accept command at clock cycle 1 but source does not have any command to transfer in that same cycle. The source starts transfer in the next cycle.

The second command transfers a payload of 128 Bytes. Here sink is not ready to accept command when source has asserted valid. The source holds information on the bus till it observes ready from sink.

Figure 71. Timing Diagram for Simple Packing Scheme - Command With Data

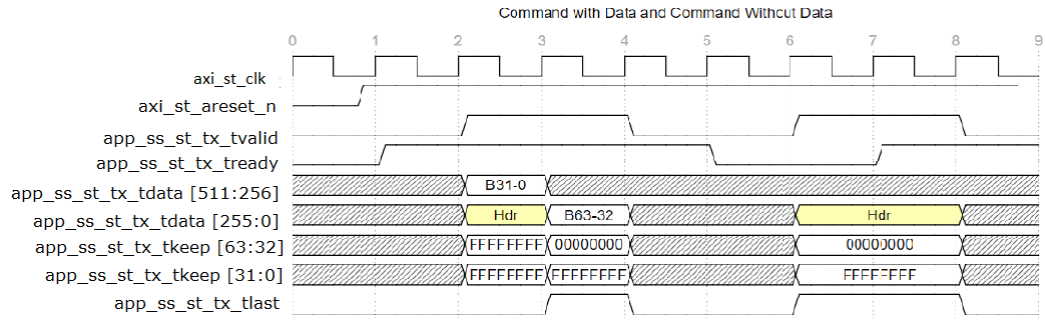


The figure below shows timing diagram for command with data followed by command without data. The completion, memory write, messages and configuration write commands fall under command with data category. The memory read, configuration read, messages without data and completion without data fall under command without data category.

The first command transfers a payload of 64 Bytes. The sink is ready to accept command at clock cycle 1 but source does not have any command to transfer in that same cycle. The source starts transfer in the next cycle.

The second command is a command without data. Here sink is not ready to accept command when source has asserted valid. The source holds information on the bus till it observes ready from sink.

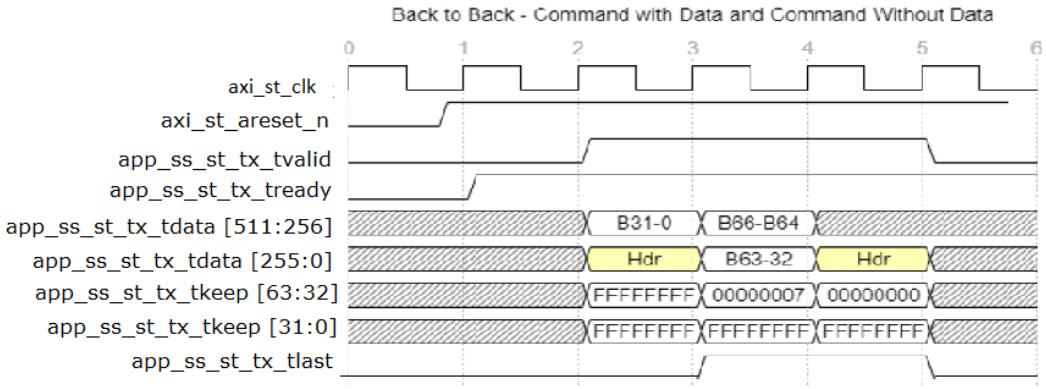
Figure 72. Timing Diagram for Simple Packing Scheme - Command With Data and Command Without Data



The first command transfers the payload of 67 Bytes. Note tkeep during tlast has partial ones, but these ones are contiguous, sparse tkeep is not allowed. The partial

tkeep is allowed only on tlast cycle. The second command is a command without data.

Figure 73. Timing Diagram for Simple Packing Scheme – Back-to-Back Commands With and Without Data



6.3.4. Application Request Interface (st_txreq)

This interface is available in AXI Data Mover mode only. The Data Mover mode provides this additional interface for you to send requests without data. You can send only DMRd or DMIntr commands through this interface.

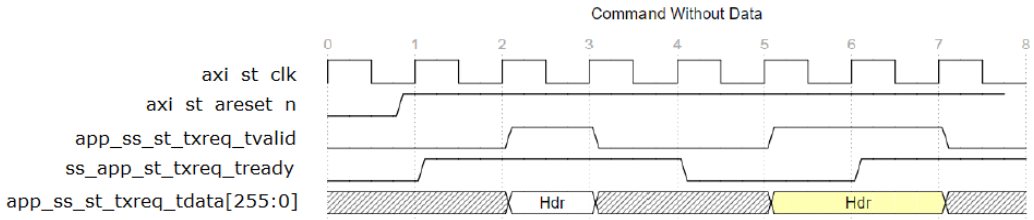
Table 52. APP AXI ST Request TX Interface

Signal Name	Direction	Clock Domain	Description
app_ss_st_txreq_tvalid	Input	axi_st_clk	app_ss_st_txreq_tvalid indicates that the source is driving a valid transfer
ss_app_st_txreq_tready	Output	axi_st_clk	ss_app_st_txreq_tready indicates that the sink can accept a transfer in the current cycle readyLatency parameter defined in Avalon spec shall be supported. By default, the value is '0'.
app_ss_st_txreq_tdata[255:0]	Input	axi_st_clk	Carries Data Mover Mode read command and Interrupt command.
app_ss_st_txreq_tlast	Input	axi_st_clk	Indicates End of Data/Command Transmission

The figure below shows timing diagram of command without data. The first command sends DMRd command. The sink is ready to accept command at clock cycle 1 but source does not have any command to transfer in that same cycle. The source starts transfer in the next cycle. The second command is DMIntr.

Here sink is not ready to accept command when source has asserted valid. The source holds information on the bus till it observes ready from sink.

Figure 74. Timing Diagram for AXI Data Mover Mode - Command Without Data



6.3.5. Application AXI MM Initiator Interface (mm_initatr)

This interface is available in the Data Mover mode only. The Data Mover mode provides an AXI MM master interface for high bandwidth data transfer to and from memory resources in FPGA logic. The interface supports data widths of 32 bytes (256 bits), 64 bytes (512 bits) and 128 bytes (1024 bits).

Table 54. APP AXI MM Initiator Interface

Signal Name	Direction	Registered	Clock Domain	Description
Write Address Channel				
ss_app_mm_initatr_awvalid	Output	Yes	axi_mm_clk	Indicates that the write address channel signals are valid
app_ss_mm_initatr_awready	Input	NA	axi_mm_clk	Indicates that a transfer on the write address channel can be accepted
ss_app_mm_initatr_awaddr[MMAWD-1:0]	Output	Yes	axi_mm_clk	The address of the first transfer in a write transaction The default value of

				MMAWD = 64
ss_app_mm_initatr_awlen[MMBLWD-1:0]	Output	Yes	axi_mm_clk	Length, the exact number of data transfers in a write transaction
ss_app_mm_initatr_awsize[2:0]	Output	Yes	axi_mm_clk	The maximum number of bytes to transfer in each data transfer, or beat, in a burst
ss_app_mm_initatr_awprot	Output	Yes	axi_mm_clk	Protection attributes of a write transaction: privilege, security level, and access type
Write Data Channel				
ss_app_mm_initatr_wvalid	Output	Yes	axi_mm_clk	Indicates that the write data channel signals are valid
ss_app_mm_initatr_wlast	Output	Yes	axi_mm_clk	Indicates whether this is the last data transfer in a write transaction
app_ss_mm_initatr_wready	Input	NA	axi_mm_clk	Indicates that a transfer on the write data channel can be accepted

ss_app_mm_initatr_wdata[MMDWD-1:0]	Output	Yes	axi_mm_clk	Write Data the default value of MMDWD = 512
ss_app_mm_initatr_wstrb[MMDWD/8-1:0]	Output	Yes	axi_mm_clk	Write strobes, indicate which byte lanes hold valid data
Write Response Channel				
app_ss_mm_initatr_bvalid	Input	NA	axi_mm_clk	Indicates that the write response channel signals are valid
ss_app_mm_initatr_bready	Output	Yes	axi_mm_clk	Indicates that a transfer on the write response channel can be accepted
app_ss_mm_initatr_bresp[1:0]	Input	NA	axi_mm_clk	Write response, indicates the status of a write transaction
Read Address Channel				
ss_app_mm_initatr_arvalid	Output	Yes	axi_mm_clk	Indicates that the read address channel signals are valid
app_ss_mm_initatr_arready	Input	NA	axi_mm_clk	Indicates that a transfer on the read address channel can be accepted

ss_app_mm_initatr_araddr[MMAWD-1:0]	Output	Yes	axi_mm_clk	The address of the first transfer in a read transaction The default value of MMAWD = 64
ss_app_mm_initatr_arlen[MMAWD-1:0]	Output	Yes	axi_mm_clk	Length, the exact number of data transfers in a read transaction
ss_app_mm_initatr_arsize[2:0]	Output	Yes	axi_mm_clk	The maximum number of bytes to transfer in each data transfer, or beat, in a burst
ss_app_mm_initatr_arprot	Output	Yes	axi_mm_clk	Protection attributes of a read transaction: privilege, security level, and access type
Read Data Channel				
app_ss_mm_initatr_rvalid	Input	NA	axi_mm_clk	Indicates that the read data channel signals are valid
app_ss_mm_initatr_rlast	Input	NA	axi_mm_clk	Indicates whether this is the last data transfer in a read transaction
ss_app_mm_initatr_rready	Output	Yes	axi_mm_clk	Indicates that a

				transfer on the read data channel can be accepted
app_ss_mm_initatr_rdata[MMDWD-1:0]	Input	NA	axi_mm_clk	Read data The default value of MMDWD = 512
app_ss_mm_initatr_rresp[1:0]	Input	NA	axi_mm_clk	Read response, indicates the status of a read transfer

6.4. Configuration Extension Bus Interface

6.4.1. Configuration Extension Bus Request Interface (st_cebreq)

The Subsystem sends configuration read and configuration write requests using this interface. The interface follows AXI Streaming interface protocol with ready valid handshake. The interface will support a maximum of one outstanding read request.

This interface is mutually exclusive with the Configuration Intercept Request Interface.

Table 55. Configuration Extension Bus Request Interface

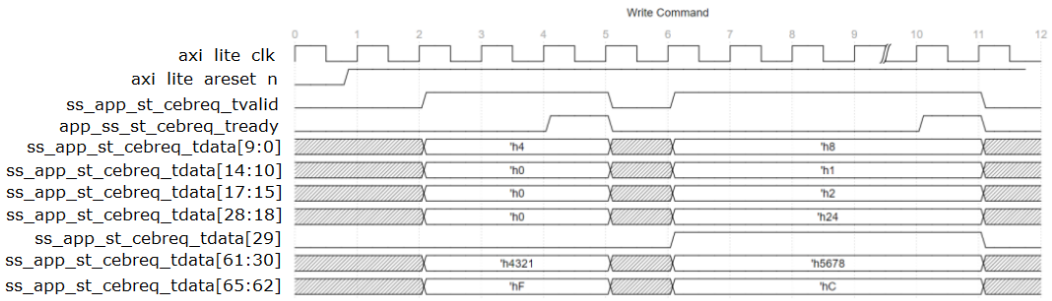
Signal Name	Direction	Clock Domain	Description
ss_app_st_cebreq_tvalid	Output	axi_lite_clk	When asserted, indicates a valid Configuration Extension access cycle. Deasserted when app_ss_st_cebreq_tready is asserted.
app_ss_st_cebreq_tready	Input	axi_lite_clk	Application asserts this signal for one clock to acknowledge ss_app_st_cebreq_tvalid is seen by responder.
ss_app_st_cebreq_tdata[9:0]	Output	axi_lite_clk	DWORD Address of register being accessed
ss_app_st_cebreq_tdata[14:10]	Output	axi_lite_clk	The slot Number of register access

ss_app_st_cebreq_tdata[17:15]	Output	axi_lite_clk	The PF Number of register access
ss_app_st_cebreq_tdata[28:18]	Output	axi_lite_clk	Indicates child VF Number of parent PF indicated by ss_app_st_cebreq_tdata[17:15]
ss_app_st_cebreq_tdata[29]	Output	axi_lite_clk	Indicates access is for Virtual Function implemented in slot's physical function
ss_app_st_cebreq_tdata[61:30]	Output	axi_lite_clk	Write data for write access
ss_app_st_cebreq_tdata[65:62]	Output	axi_lite_clk	Indicates the configuration register access type, read or write. For writes, indicates the byte enables: The following encodings are defined: 4'b0000: Read 4'b0001: Write byte 0 4'b0010: Write byte 1 4'b0100: Write byte 2 4'b1000: Write byte 3 4'b1111: Write all bytes. Combinations of byte enables, for example,4'b 0101b are also valid.

The figure below shows timing diagram for write command; the first command sends write for all four bytes of register located at address=4. The ss_app_st_cebreq_tdata[29] low indicates the access is for a physical function.

The second command sends write for byte3 and byte2 of register located at address =8. The ss_app_st_cebreq_tdata[29] high indicates access is for a virtual function.

Figure 75. Timing Diagram for Configuration Extension Bus Request Interface



6.4.2. Configuration Extension Bus Response Interface (st_cebresp)

The application returns read data for requests received from "st_cebreq" interface using "st_cebresp" interface. The Subsystem will be always ready to accept responses from the application. The application will provide response data with valid qualifier.

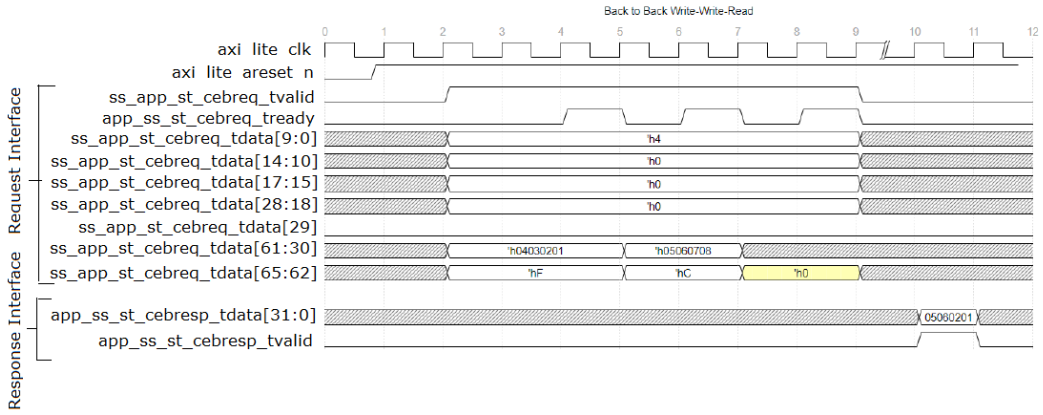
This interface is mutually exclusive with the Configuration Intercept Response Interface.

Table 56. Configuration Extension Bus Response Interface

Signal Name	Direction	Clock Domain	Description
app_ss_st_cebresp_tvalid	Input	axi_lite_clk	Application assert this signal for one clock to indicate that valid data is driven on app_ss_st_cebresp_tdata bus.
app_ss_st_cebresp_tdata[31:0]	Input	axi_lite_clk	Response data from application for read request issued using "st_cebreq interface"

The figure below shows timing diagram for back-to-back write and read command; the first command sends write for all four bytes of register located at address=4. The second command sends write for byte3 and byte2 of same register. The third command sends read for same register. Upon receiving the read command on st_cebreq interface, the application returns data on st_cebresp interface. Note the data returned is 5621. The upper two bytes were modified by the second write.

Figure 76. Timing Diagram for Configuration Extension Bus Response Interface



6.5. Configuration Intercept Interface

The Subsystem allows application logic to intercept configuration read and configuration write requests using this interface. The interface follows AXI Streaming interface protocol with ready valid handshake. The interface will support a maximum

of one outstanding request at a time. The Subsystem provides st_ciireq and st_ciiresp interfaces for intercepting packets.

Note: This interface is provided so that PCIe Subsystem is backward compatible to legacy application logic that relies on CII for their functionality. Newly defined application logic should avoid using the CII interface and move to the CEB interface.

Note: This interface is mutually exclusive with the Configuration Extension Bus Request Interface.

This interface is applicable only when operating as Endpoint in Power User mode. This interface is not available when operating in Data Mover mode.

6.5.1. Configuration Intercept Request Interface (st_ciireq)

Table 57. Configuration Intercept Request Interface

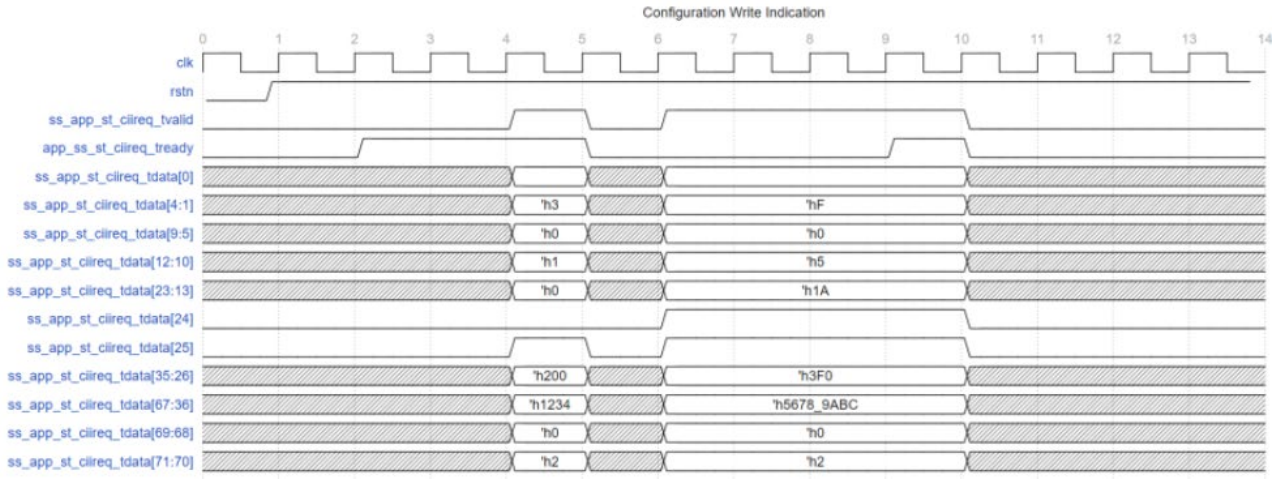
Signal Name	Direction	Clock Domain	Description
ss_app_st_ciireq_tvalid	Output	axi_lite_clk	When asserted, indicates a valid CFG request cycle is waiting to be intercepted. Deasserted when app_ss_st_ciireq_tready is asserted.
app_ss_st_cciireq_tready	Input	axi_lite_clk	Application asserts this signal for one clock to acknowledge ss_app_st_ciireq_tvalid is seen by responder.
ss_app_st_ciireq_tdata[0]	Output	axi_lite_clk	hdr_poisoned: The poisoned bit in the received TLP header on the CII.
ss_app_st_ciireq_tdata[4:1]	Output	axi_lite_clk	hdr_first_be: The first dword byte enable field in the received TLP header on the CII.
ss_app_st_ciireq_tdata[9:5]	Output	axi_lite_clk	slot_num: The slot number in the received TLP header on the CII.
ss_app_st_ciireq_tdata[12:10]	Output	axi_lite_clk	func_num: The PF number in the received TLP header on the CII.
ss_app_st_ciireq_tdata[23:13]	Output	axi_lite_clk	vf_num: The child VF number of parent PF in the received TLP header on the CII.

ss_app_st_ciireq_tdata[24]	Output	axi_lite_clk	vf_active: Indicates VF number is valid in the received TLP header on the CII.
ss_app_st_ciireq_tdata[25]	Output	axi_lite_clk	wr: Indicates a configuration write request detected in the received TLP header on the CII. Also indicates that st_app_st_ciireq_tdata[67:36] is valid.
ss_app_st_ciireq_tdata[35:26]	Output	axi_lite_clk	addr: The double word register address in the received TLP header on the CII.
ss_app_st_ciireq_tdata[67:36]	Output	axi_lite_clk	dout: Received TLP payload data from the link partner to your application client. The data is in little endian format. The first received payload byte is in [43:36].

The figure below shows timing diagram for configuration write request indication to the
application when intercept feature is not enabled.

The first command is a configuration write to PF1 byte-1 and byte-0 at address=0x200. The tvalid ishigh for 1 clock cycle as the application is ready to accept the packet.

The second command is a full dword configuration write to VF=26 of PF5 at address=0x3F0. As the
application is not ready to accept the packet, sub-system holds the information till tready is
seen.



6.5.2. Configuration Intercept Response Interface (st_ciiresp)

The application must return the response for request received on "st_ciireq" interface using "st_ciiresp" interface. The Subsystem will be always ready to accept responses from the application. The application will provide response data with valid qualifier.

This interface is applicable only when operating as Endpoint in Power User mode. This interface is not available when operating in Data Mover mode. This interface is mutually exclusive with the Configuration Extension Bus Response Interface.

Table 58. Configuration Intercept Response Interface

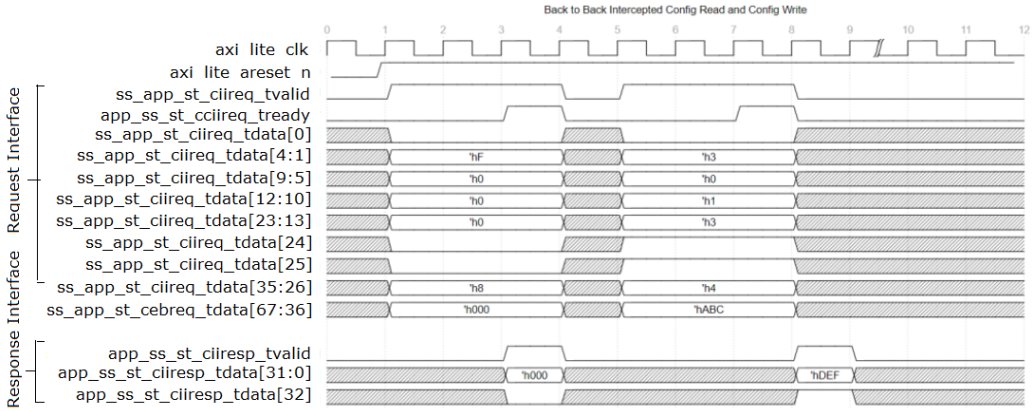
Signal Name	Direction	Clock Domain	Description
app_ss_st_ciiresp_tvalid	Input	axi_lite_clk	Application asserts this signal for one clock to indicate that valid data is driven on app_ss_st_ciiresp_tdata bus.
app_ss_st_ciiresp_tdata[31:0]	Input	axi_lite_clk	Override data from application for the intercepted configuration request on "st_ciireq interface" For CfgWr: override the write data to the Configuration register with data supplied by the application logic. For CfgRd: override the data payload of the completion TLP with data supplied by the application logic.
app_ss_st_ciiresp_tdata[32]	Input	axi_lite_clk	Override Data Enable: Application assert this signal to override the CfgWr payload or CfgRd completion using the data supplied by the application logic on app_ss_st_ciiresp_tdata[31:0] bus.

The figure below shows timing diagram for back-to-back read and write request; the first request sends configuration read on st_ciireq interface for all four bytes of register located at address=0x8. Application decides not to intercept this configuration read and hence return tvalid=1 together with tdata[32]=0 on the st_ciiresp interface.

After receiving the first response on the st_ciiresp interface, the second request sends configuration write for byte0, byte1 and byte2 of register located at address=0x4 with data value of 0xABC. Application decides to intercept this configuration write and hence return tvalid=1 together with tdata[32]=1 on the st_ciiresp interface. Additionally, application provides the data (i.e., 0xDEF) to be

used for the intercepted configuration write on the st_ciiresp interface through tdata[31:0].

Figure 77. Timing Diagram for Configuration Intercept Interface



6.6. Function Level Reset Interface

Each of the functions in the Subsystem can be individually reset through function level reset interface. The Subsystem provides st_flrrcvd and st_flrcmpl interfaces for FLR handshake.

6.6.1. Function Level Reset Received Interface (st_flrrcvd)

The st_flrrcvd interface provides indication to the application for which function FLR is received from HOST. The application responds with FLR completion using st_flrcmpl interface, indicating FLR process is completed. The Subsystem does not implement any timeout for this handshake, so it is important that application sends FLR completion for FLR request received on st_flrrcvd interface.

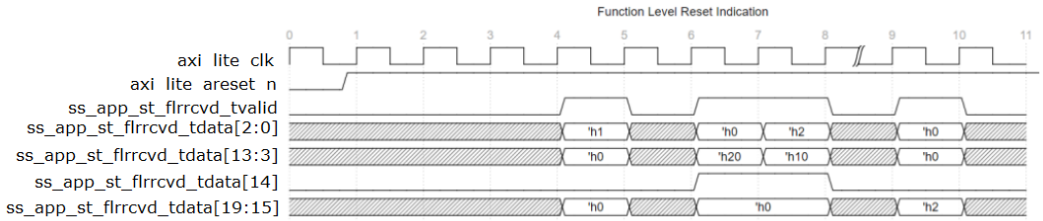
Table 59. Function Level Reset Received Interface

Signal Name	Direction	Clock Domain	Description
ss_app_st_flrrcvd_tvalid	Output	axi_lite_clk	When asserted, indicates a FLR request received from HOST. The signal is valid for one clock cycle
ss_app_st_flrrcvd_tdata[19:0]	Output	axi_lite_clk	[2:0] - The PF Number of FLR Request [13:3] - Indicates child VF Number of parent PF indicated by PF Number [14] - Indicates request is for Virtual Function

			implemented in slot's physical function [19:15] - The slot Number of FLR Request

The figure below shows timing diagram for function level reset indication to application. The first command indicates FLR for Physical Function =1 on slot = 0. The second and third back-to-back indications are for VF, the ss_app_st_flrrcvd_tdata[14] high indicates FLR is received for Virtual function. The fourth command signals FLR for PF=0 and slot=2.

Figure 78. Timing Diagram for Function Level Reset Receive Interface



6.6.2. Function Level Reset Completion Interface (st_flrcmpl)

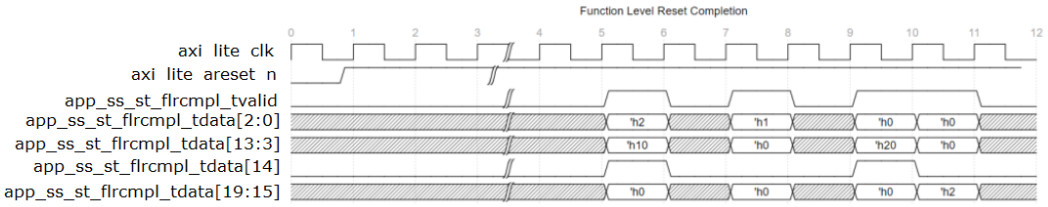
The st_flrrcvd interface provides indication to the application for which function FLR is received from HOST. The application responds with FLR completion using st_flrcmpl interface, indicating FLR process is completed. The Subsystem does not implement any timeout for this handshake, so it is important that application sends FLR completion for FLR request received on st_flrrcvd interface.

Table 60. Function Level Reset Completion Interface

Signal Name	Direction	Clock Domain	Description
app_ss_st_flrcmpl_tvalid	Input	axi_lite_clk	When asserted, indicates a FLR request completed by application. The signal is valid for one clock cycle
app_ss_st_flrcmpl_tdata[19:0]	Input	axi_lite_clk	[2:0] - The PF Number of FLR Completion [13:3] - Indicates child VF Number of parent PF indicated by PF Number [14] - Indicates completion is from Virtual Function implemented in slot's physical function [19:15] - The slot Number of FLR completion

The figure below shows timing diagram for function level reset completion from application. The first completion indicates FLR completion for Virtual Function =0x10, the app_ss_flrcmpl_tdata[14] high indicates FLR completion from Virtual function. The second completion indicates FLR completion for Physical Function =0x1. The third completion indicates FLR completion for Virtual Function =0x20, the app_ss_flrcmpl_tdata[14] high indicates FLR completion from Virtual function. The fourth completion indicates FLR completion for Physical Function =0x0 in "slot = 2"

Figure 79. Timing Diagram for Function Level Reset Completion Interface



6.7. Control Shadow Interface (st_ctrlshadow)

The control shadow interface is used to bring out the settings of the various configuration register fields of the function. These fields are often required in designing the control path of the application layer logic.

The application logic decodes information provided on this interface to create a shadow copy. The interface provides update to primary control signals only. The application logic must read extra information required through lite_csr interface by reading configuration register of interest. The sections below explain additional information required by application.

Bus Master Enable

The application logic requires the BME information to determine if it can generate request for a particular function. Each function in the application logic cannot generate bus master requests unless its corresponding BME is set. The application logic monitors control shadow interface for BME event for this purpose. Since PCIe SS doesn't autonomously generate bus master request by itself, it will not qualify the transmit path with BME settings and solely relies on application.

MSI Enable

The application logic requires MSI Address and MSI Data information from MSI capability to generate MSI interrupt. The application logic monitors control shadow interface for MSI Enable event to read this additional information from MSI capability.

VF Enable

The Virtual Function in application logic cannot generate any traffic unless they are enabled by HOST. The number of VFs enabled can be different than number of VFs advertised as initial VFs. The application logic can find number of VFs visible by reading NumVFs register in SRIOV capability. The read of this register must be triggered after VF Enable bit is set by HOST.

TPH Req Enable

The function can support all operational modes of TPH, no ST mode, interrupt vector mode or device specific mode. The HOST communicates mode of operation by writing ST Mode Select bits in TPH requester control register. The application reads this register and generates traffic only when TPH requester enable bit is set.

ATS Enable

The function can read Smallest Translation Unit field from ATS control register when ATS enable bit is set.

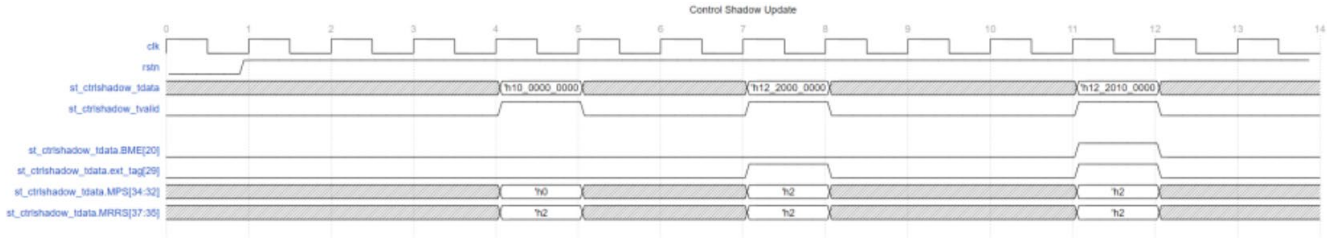
Table 61. Control Shadow Interface

Signal Name	Direction	Clock Domain	Description
ss_app_st_ctrlshadow_tvalid	Output	axi_lite_clk	The Subsystem asserts this output for one clock cycle when there is an update to the register fields being monitored, because of a Configuration Write performed by the Root Comple.g., The user can copy the new settings of the register fields from the tdata bus.
ss_app_st_ctrlshadow_tdata[39:0]	Output	axi_lite_clk	When app_ctrl_shadow_tvalid has been asserted, this output provides the current settings of the register fields of the associated Function [2:0] - Identifies the physical function Number of configuration register [13:3] - Identifies the virtual function Number of configuration register [14] - Indicates information is for Virtual Function implemented in slot's physical function [19:15] - Identifies the slot Number of configuration register [20] - Bus Master Enable [21] - MSI-X Mask [22] - MSI-X Enable [23] - Mem Space Enable [24] - ExpRom Enable [25] - TPH Req Enable

			[26] - ATS Enable [27] - MSI Enable [28] - MSI Mask [29] - Extended Tag [30] - 10Bit Tag Req Enable [31] - PTM Enable [34:32] - MPS Size [37:35] - MRRS Size [38] - VF Enable [39] - Page Request Enable

ATS Enable Timing Diagram

The figure below shows output on the Control Shadow interface when there is an update to the
control shadow bits in the HIP configuration register.



6.8. Transmit Flow Control Credit Interface (st_txcrdt)

The link partner's receive buffer space information is provided to application through the Transmit Flow Control Credit Interface. The application transmits packet only when link partner receive buffer has enough space to accept the TLP. The interface provides Posted, Non-Posted, Completion data and header credit information. One data credit is equal to four dwords (DWs) and one header credit is equal to the max size header plus optional digest field. The credits are advertised as limit values.

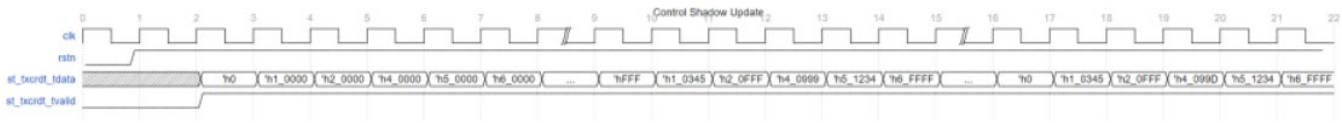
Table 62. Transmit Flow Control Credit Interface

Signal Name	Direction	Clock Domain	Description
ss_app_st_txcrdt_tvalid	Output	axi_st_clk	tvalid indicates that the credit information on tdata is valid
ss_app_st_txcrdt_tdata[18:0]	Output	axi_st_clk	Carries credit limit information and type of credit [15:0] - Credit Limit Value

			[18:16] - Credit Type • 3'b000 - Posted Header Credit • 3'b001 - Non-Posted Header Credit • 3'b010 - Completion Header Credit • 3'b011 - Reserved • 3'b100 - Posted Data Credit • 3'b101 - Non-Posted Data Credit • 3'b110 - Completion Data Credit • 3'b111 - Reserved

The figure below shows the credit limit update on the Transmit Flow Control Credit Interface.
The credit limit is first initialized to 0 for all the credit types.

In the example below, updated credit limit is output from cycle 9 to cycle 14. When the HOST
returns the credit after receiving the packet, credit limit is incremented by the number of
credits returned. At cycle 16, one Posted Header credit is returned at cycle 19, four Posted
Data credit is returned.



6.9. Completion Timeout Interface (st_cplto)

The completion timeout interface indicates completion timeout event to application. The interface provides the function number and tag number of the outstanding request timed out.

Table 63. Completion Timeout Interface

Signal Name	Direction	Clock Domain	Description
ss_app_st_cplto_tvalid	Output	axi_lite_clk	tvalid indicates that the completion timeout received for outstanding NP request

ss_app_st_cplto_tdata[29:0]	Output	axi_lite_clk	Carries completion Timeout Information [9:0] - Tag Number [12:10] - PF Number, indicates parent PF number of VF when VF Active is high else PF Number of function [23:13] - VF Number, indicates VF number when VF Active js high [24] - VF Active, Indicates timeout is for VF [29:25] always 0. Reserved

6.10. Miscellaneous Signals

Table 64. Miscellaneous Signals

Signal Name	Direction	Clock Domain	Description
ss_app_serr	Output	axi_st_clk	Indicates System Error is detected In TLP Bypass Mode indicates PL/DL/TL layer error detected by HardIP
ss_app_dlup	Output	axi_st_clk	Indicates Data Link Layer is UP
ss_app_int_status	Output	axi_st_clk	This signal drives legacy interrupts to the Application Layer. The source of the interrupt will be logged in the Root Port Interrupt Status registers in the Port Configuration and Status registers. Note: Applicable only in Root Port Mode.
ss_app_surprise_down_err	Output	Async	Indicates that a surprise down event is occurring in the HardIP controller.
ss_app_rx_par_err	Output	axi_st_clk	Indicates a parity error detected at the input of the HIP'S RX buffer. Asserts for a single cycle. Note: Application must reset the HardIP if this occurs because

			parity errors can leave the HardIP in an unknown state.
ss_app_tx_par_err	Output	axi_st_clk	Indicates a parity error during TX TLP transmission at the HIP. Asserts for a single cycle.
ss_app_bus_number[31:0]	Output	axi_lite_clk	Indicates content of Bus Number Register [4:0] - PF Number [17:5] - Reserved [18] - Bus Number is Valid [23:19] - Reserved [31:24] - Bus Number

6.11. Control and Status Register Responder Interface (lite_csr)

This interface is available in power user and data mover modes. The Subsystem provides a Control and Status Register Interface to access registers implemented in Subsystem modules. The user can access PCI/PCIe Configuration Registers of all Functions through this interface as well as private register implemented in design. The interface follows AXI4-Lite protocol.

The Subsystem does not differentiate between non-secure and secure accesses. All accesses are considered secure.

Table 65. Control and Status Register Responder Interface

Signal Name	Direction	Clock Domain	Description
Write Address Channel			
app_ss_lite_csr_awvalid	Input	axi_lite_clk	Indicates that the write address channel signals are valid
ss_app_lite_csr_awready	Output	axi_lite_clk	Indicates that a transfer on the write address channel can be accepted
app_ss_lite_csr_awaddr [LiteSlvAWD-1:0]	Input	axi_lite_clk	The address of the first transfer in a write transaction The default value of LiteSlvAWD = 18
Write Data Channel			

app_ss_lite_csr_wvalid	Input	axi_lite_clk	Indicates that the write data channel signals are valid
ss_app_lite_csr_wready	Output	axi_lite_clk	Indicates that a transfer on the write data channel can be accepted
app_ss_lite_csr_wdata[LiteSlvDWD-1:0]	Input	axi_lite_clk	Write Data The default value of LiteSlvDWD=32
app_ss_lite_csr_wstrb[LiteSlvDWD/8-1:0]	Input	axi_lite_clk	Write strobes, indicate which byte lanes hold valid data
Write Response Channel			
ss_app_lite_csr_bvalid	Output	axi_lite_clk	Indicates that the write response channel signals are valid
app_ss_lite_csr_bready	Input	axi_lite_clk	Indicates that a transfer on the write response channel can be accepted
ss_app_lite_csr_bresp[1:0]	Output	axi_lite_clk	Write response, indicates the status of a write transaction
Read Address Channel			
app_ss_lite_csr_arvalid	Input	axi_lite_clk	Indicates that the read address channel signals are valid
ss_app_lite_csr_arready	Output	axi_lite_clk	Indicates that a transfer on the read address channel can be accepted
app_ss_lite_csr_araddr[LiteSlvAWD-1:0]	Input	axi_lite_clk	The address of the first transfer in a read transaction The default value of LiteSlvAWD = 18
Read Data Channel			
ss_app_lite_csr_rvalid	Output	axi_lite_clk	Indicates that the read data channel signals are valid
app_ss_lite_csr_rready	Input	axi_lite_clk	Indicates that a transfer on the read data channel can be accepted
ss_app_lite_csr_rdata[LiteSlvDWD:0]	Output	axi_lite_clk	Read data The default value of LiteSlvDWD=32
ss_app_lite_csr_rresp[1:0]	Output	axi_lite_clk	Read response, indicates the status of a read transfer

6.12. VF Error Flag Interface (vf_err/sent_vfnonfatalmsg)

This interface is available in power user mode only. When SRIOV is enabled, the PCIe Subsystem provides a passage for the HIP's VF Error Flag Interface to application logic.

In the absence of AER and Error Message Generation support for VF in the HIP, the generation of VF's Non-Fatal Error messages relies on the user application logic. It is up to the user application logic to generate appropriate PCIe error messages when specific error conditions occur (as indicated by this interface).

Note: VF Non-Fatal errors reported through this interface would have their error status logged in the HIP registers already

This interface exists when SRIOV is enabled only. N/A to PCIe device type is Root Port.

Table 66. VF Error Flag Interface

Signal Name	Direction	Clock Domain	Description
ss_app_vf_err_poisonedwrreq_<w>	Output	axi_lite_clk	Indicates a Poisoned Write Request is received.
ss_app_vf_err_poisonedcompl_<w>	Output	axi_lite_clk	Indicates a Poisoned Completion is received.
ss_app_vf_err_ur_postedreq_<w>	Output	axi_lite_clk	Indicates the IP core received a Posted UR request.
ss_app_vf_err_ca_postedreq_<w>	Output	axi_lite_clk	Indicates the IP core received a Posted CA request.
ss_app_vf_err_vf_num[10:0]_<w>	Output	axi_lite_clk	Indicates the VF number for which the error is detected.
ss_app_vf_err_func_num[2:0]_<w>	Output	axi_lite_clk	Indicates the physical function number associated with the VF that has the error.
/ss_app_vf_err_overflow	Output	axi_lite_clk	Indicates a VF error FIFO overflow and a loss of an error report. The overflow can happen when axi_lite_clk is slower than the coreclkout_hip clock. It can also overflow internally in the HIP.
app_ss_sent_vfnonfatalmsg	Input	axi_lite_clk	Indicates the user application sent a non-fatal error message in response to an error detected.
app_ss_vfnonfatalmsg_vf_num[10:0]	Input	axi_lite_clk	Indicates the VF number for which the error message was generated. This bus is valid when app_ss_sent_vfnonfatalmsg is high.

app_ss_vfnonfatalmsg_func_num[2:0]	Input	axi_lite_clk	Indicates the PF number associated with the VF with the error. This bus is valid when app_ss_sent_vfnonfatalmsg is high.

6.13. VIRTIO PCI Configuration Access Interface

The VIRTIO PCI Configuration Access Interface is provided to allow application to implement the VIRTIO PCI Configuration Access Data register functionality. The VIRTIO specification allows software to use the VIRTIO PCI Configuration Access capability register as an alternative method to access VIRTIO device region. When this interface is enabled, PCIe Subsystem provides a passage for the HIP's VIRTIO PCI Configuration Access Interface to application logic. When this interface is disabled, PCIe Subsystem will internally drop writes from HIP's VIRTIO PCI Configuration Access Interface and return 0's for reads (per the requested byte length).

Note: Only the first 3 bits of QHIP's virtio_pcicfg_length_o[31:0] will be used since length is restricted by VIRTIO specification to be 1, 2 or 4 only.

Note: The QHIP's virtio_pcicfg_appvfnum_i and virtio_pcicfg_apppfnum_i are not used by QHIP. The PCIe SS can tie-off these to 0's.

Note: The QHIP's virtio_pcicfg_rdbe_i[3:0] needs to be internally driven by PCIe SS based on the pending read length on the st_virtio_pcicfgreq interface.

Note: When the QHIP is not instantiated, the VirtIO capability structure will be included in the SS.

6.13.1. VIRTIO PCI Config Access Request Interface (stvirtio_pcicfgreq)

Table 67. VIRTIO PCI Configuration Access Request Interface

Signal Name	Direction	Clock Domain	Description
ss_app_virtio_pcicfgreq_tvalid	Output	axi_lite_clk	When asserted, indicates a VIRTIO PCI Configuration Access Request received from HOST. The signal is valid for one clock cycle
ss_app_virtio_pcicfgreq_tdata[95:0]	Output	axi_lite_clk	[0] - When set, the request is a write request. Else, the request is a read request. [1] - Indicates request is for Virtual Function

			implemented in slot's physical function [12:2] - Indicates child VF Number of parent PF indicated by PF Number [15:13] - The PF Number of the Request [20:16] - The Slot Number of the Request [28:21] - The BAR value to be used for the Request [60:29] - The BAR Offset value to be used for the Request [63:61] - The Length value to be used for the Request [95:64] - The Data value to be used for the Write Request. N/A for Read Request.

6.13.2. VIRTIO PCI Config Access Completion Interface (stvirtio_pcicfgcmpl)

Table 68. VIRTIO PCI Configuration Access Completion Interface

Signal Name	Direction	Clock Domain	Description
app_ss_virtio_pcicfgcmpl_tvalid	Input	axi_lite_clk	When asserted, indicates a VIRTIO PCI Configuration Access Completion to be returned to HOST. The signal is valid for one clock cycle
app_ss_virtio_pcicfgcmpl_tdata[31:0]	Input	axi_lite_clk	[31:0] - The completion Data value.

6.14. Serial Data Signals

The Intel FPGA Subsystem IP for PCI Express natively supports 4, 8, or 16 PCIe lanes. Each lane includes a TX differential pair and an RX differential pair. Data is striped across all available lanes. Refer to the table *Variables Used in the Bus Indices* for more details on bus indices

The following table shows the signals of the Serial Interface of the PCIe IP Subsystem.

Table 69. Serial Data Signals

Signal Name	Direction	Clock Domain	Description
tx_p_out[<b>-1:0], tx_n_out[<b>-1:0]	Output	N/A	Transmit serial data outputs using the High-Speed Differential I/O standard.
rx_p_in[<b>-1:0], rx_n_in[<b>-1:0]	Input	N/A	Receive serial data inputs using the High-Speed Differential I/O standard.

7. Register Descriptions

The subsequent sections describe the PCIe Subsystem registers in detail. Please refer to the table below for the definition of the acronyms used in the "Attribute User Side" column.

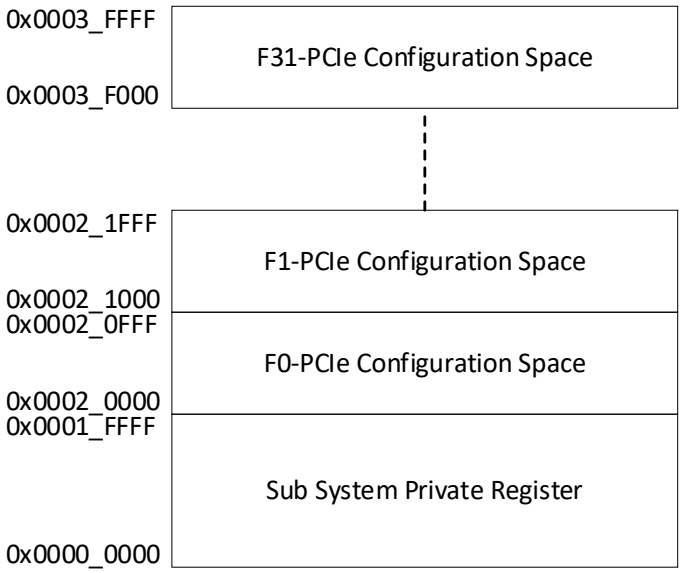
Table 70. Register Attribute Definition

Attribute	Definition
RW	Read Write
RWS	Read Write Sticky
RO	Read Only
ROS	Read Only Sticky
WO	Write Only
RW1S	Read Write 1 to Set. Clear by Hardware
RW1C	Read Write 1 to Clear. Set by Hardware
RW1CS	Read Write 1 to Clear sticky
RsvdZ	Reserved, Return 0
Hwinit	Hardware Initiate

7.1. Register Address Map

The application can access registers of Subsystem as well as PCIe configuration space registers of physical functions present in design. The figure below shows the address map of registers when accessing them from the AXI Lite csr Interface (lite_csr). Refer to the next sections for details on the register addresses and bit mappings for each register space

Figure 80. PCIe Subsystem Address Map



All AXI-Lite accesses are completed with appropriate response (BRESP, RRESP) so that the bus does not stall.

- AXI-Lite access to address ranges defined in Register Address Map shall be completed successfully with BRESP/RRESP="OK". This includes access to unimplemented, i.e., Reserved, register offsets within the valid address range.
  • Read to reserved register shall return value of all zeroes.
  • Write to register location containing any number of RO or RO/V bit, design shall return write response BRESP=OKAY. Write is dropped for the RO or RO/V bit location(s). This is not an error condition.
- AXI-Lite access to address beyond the register map shall be completed gracefully with BRESP/RRESP="DECERR". Read returns all zeroes and write is dropped.

7.2. PCIe Configuration Space

7.2.1. PCIe Configuration Space Registers

Refer to the *Appendix A – Table PCIe Configuration Space registers for x16/x8/x4 controllers* of the *P-tile Avalon Streaming Intel FPGA IP For PCI Express User Guide* for register details

Reference Documents

P-tile Avalon Streaming Intel FPGA IP for PCI Express User Guide

7.2.2. Fn PCIe Configuration Space Address Map

The physical functions in design can be spread across different slots. These physical functions in different slots will be mapped to Fn-PCIe Configuration Space. The mapping will follow rules specified below:

- One physical function in the slot will be mapped to one Function in the address map
- All the physical functions in a given slot will occupy contiguous function numbers in lexicographical order
- If Slot does not have any physical function, no address space will be allocated to that slot
  The Address Space which is not mapped to any Physical Function is considered as RsvdZ
  Note: This Direct Access method can be used for up to 32 PFs across all slots.
  Note: For configurations that require more than 32 PFs or require VFs, the Indirect Access
  method needs to be used.

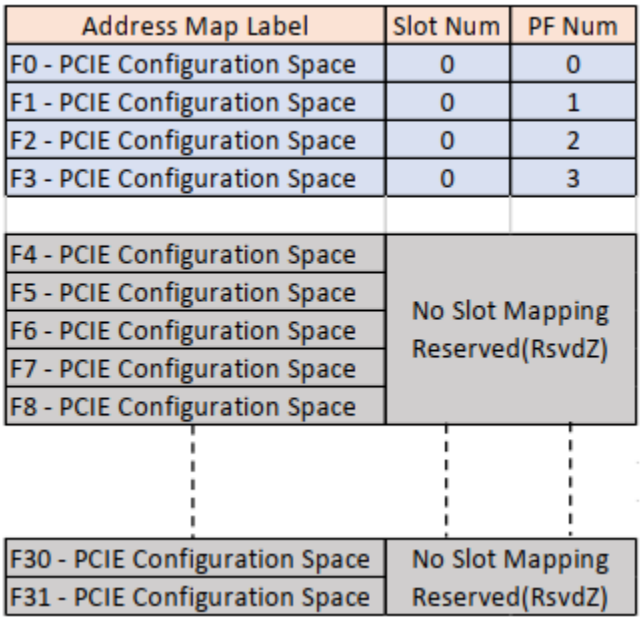
The example below shows the address map allocation for 32 slots with 1 Physical Function

Figure 81. Address map allocation for 32 slots with 1 Physical function

Address Map Label	Slot Num	PF Num
F0 - PCIE Configuration Space	0	0
F1 - PCIE Configuration Space	1	0
F2 - PCIE Configuration Space	2	0
F3 - PCIE Configuration Space	3	0
F4 - PCIE Configuration Space	4	0
F5 - PCIE Configuration Space	5	0
F6 - PCIE Configuration Space	6	0
F7 - PCIE Configuration Space	7	0
F8 - PCIE Configuration Space	8	0
⋮	⋮	⋮
F30 - PCIE Configuration Space	30	0
F31 - PCIE Configuration Space	31	0

The example below shows address space allocation for one slot with 4 Physical Function

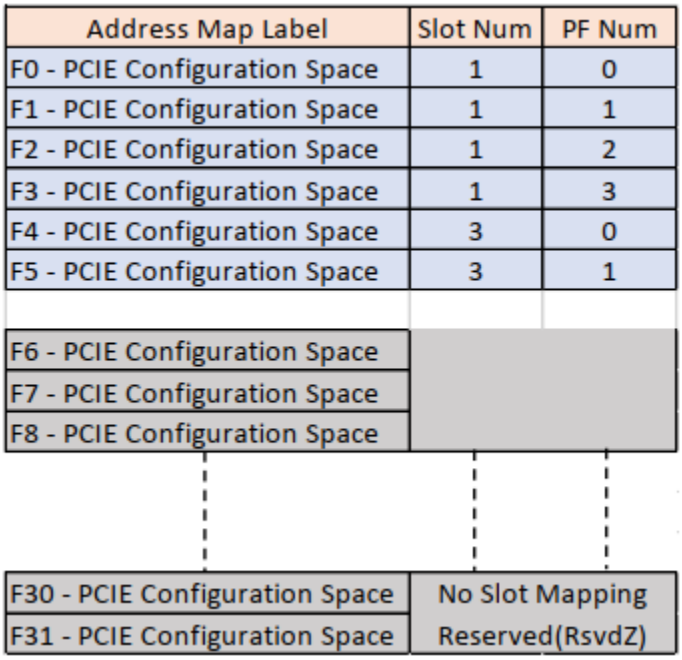
Figure 82. Address map allocation for 1 slot with 4 Physical functions



The example below shows address space allocation for four slots:

- Slot 0 has no physical functions
- Slot 1 has 4 physical functions
- Slot 2 has no physical functions
- Slot 3 has 2 Physical Functions

Figure 83. Address map allocation for 4 slots

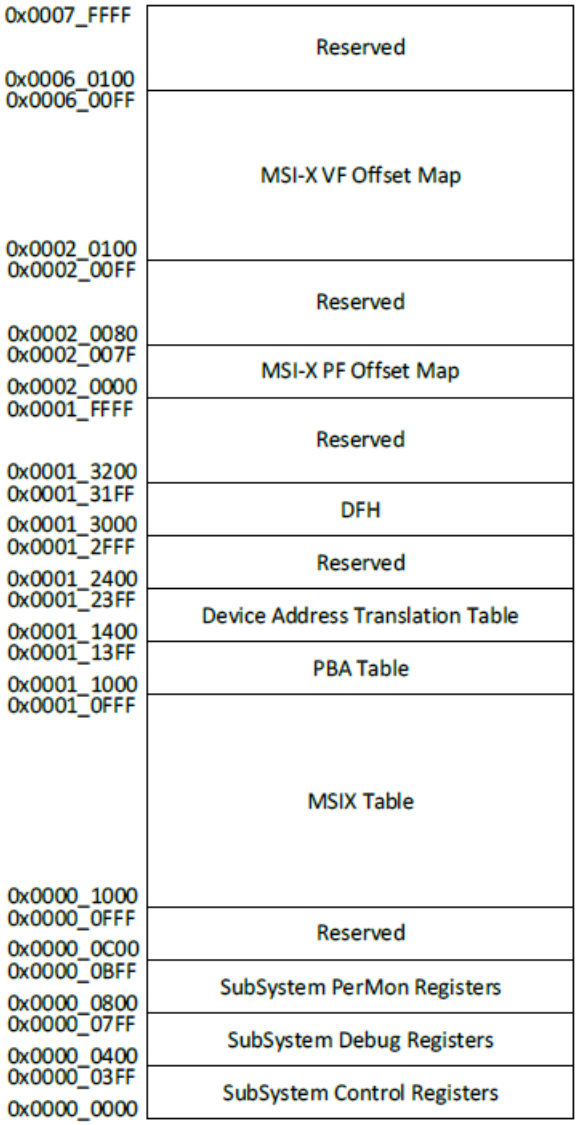


Address Map Label	Slot Num	PF Num
F0 - PCIE Configuration Space	1	0
F1 - PCIE Configuration Space	1	1
F2 - PCIE Configuration Space	1	2
F3 - PCIE Configuration Space	1	3
F4 - PCIE Configuration Space	3	0
F5 - PCIE Configuration Space	3	1
F6 - PCIE Configuration Space		
F7 - PCIE Configuration Space		
F8 - PCIE Configuration Space		
F30 - PCIE Configuration Space	No Slot Mapping	
F31 - PCIE Configuration Space	Reserved(RsvdZ)	

7.3. Subsystem Soft Register Address Map

The Subsystem address map is shown in the figure below

Figure 84. Subsystem Soft Register Address Map



7.3.1. Subsystem Control Registers

The table below lists the control registers implemented by the Subsystem. The Subsystem Control register starts from Base Address = 0x0

Table 71. Subsystem Control Register Address Map

Register Name	Offset
Subsystem Version	0x0000_0000
Subsystem Features	0x0000_0004
Subsystem Interface Attributes	0x0000_0008
Reserved	0x0000_000C
ERROR GEN CTRL	0x0000_0010
ERROR GEN ATTR	0x0000_0014
ERROR TLP Header DW0	0x0000_0018
ERROR TLP Header DW1	0x0000_001C
ERROR TLP Header DW2	0x0000_0020
ERROR TLP Header DW3	0x0000_0024
ERROR TLP Prefix	0x0000_0028
HOT PLUG GEN CTRL	0x0000_002C
POWER MANAGEMENT CTRL	0x0000_0030
LEGACY INTERRUPT CTRL	0x0000_0034
MSIX GEN CTRL EP 0	0x0000_0038
MSIX GEN CTRL EP 1	0x0000_003C
……	……
MSIX GEN CTRL EP 31	0x0000_00B4
MSI CTRL	0x0000_00B8
MSI ADDRESS LOWER	0x0000_00BC
MSI ADDRESS UPPER	0x0000_00C0
MSI DATA	0x0000_00C4
CFG REG IA CTRL	0x0000_00C8
CFG REG IA FN NUM	0x0000_00CC
CFG REG IA FN WRDATA	0x0000_00D0
CFG REG IA FN RDDATA	0x0000_00D4
PRS CTRL	0x0000_00D8
MSI PENDING CTRL	0x0000_00DC
MSI PENDING	0x0000_00E0
D-STATE STS	0x0000_00E4
CFG RETRY CTRL	0x0000_00E8
BUS_NUMBER	0x0000_00EC

The DFH CSR implements Device Feature Header version 1.0 compatible register set. The DFH CSR starts from offset 0X13000 in Subsystem memory space address map.

Table 72. DFH CSR Address Map

Register Name	Offset
X_FEATURE_DFH	0x0000_0000
X_FEATURE_GUID_L	0x0000_0008
X_FEATURE_GUID_H	0x0000_0010
X_FEATURE_CSR_ADDR	0x0000_0018
X_FEATURE_CSR_SIZE_GROUP	0x0000_0020
Reserved	0x0000_0028
Reserved	0x0000_0030
Reserved	0x0000_0038-0x0000_00FF

Note: 0x0000_0028 reserved for X_PARAM_HEADER
Note: 0x0000_0030 reserved for X_PARAM_Y
Note: 0x0000_0060 - 0x0000_00FF reserved for DFH CSR

7.3.1.1. Subsystem Version

The register indicates the Subsystem major and minor versions.

Default Value: 0x0100_0000

Table 73. Subsystem Version Register

Register Name	Bit	Attribute User Side	Description
Subsystem Version	7-0	RO	Reserved
	15-8	RO	Indicates Minor Version Number
	31-16	RO	Indicates Major Version Number

7.3.1.2. Subsystem Features

The register indicates features enabled in the Subsystem during compile time.

Default Value: Set As per Parameter Settings

Table 74. Subsystem Feature Registers

Register Name	Bit	Attribute User Side	Description
Subsystem Features 1	1-0	RO	Reflects Functional Mode parameter value 00 - Power User mode 01 - AXI-ST Data Mover mode 10 - AXI-MM Data Mover Mode

			11 - Reserved
	2	RO	Reserved
	3	RO	Indicates presence of Debug Toolkit block in a design 0 - Debug Toolkit not Present 1 - Debug Toolkit Present
	4	RO	Indicates presence of Device ATT Table in a design 0 - Device ATT not Present 1 - Device ATT Present
	5	RO	Indicates presence of MSI-X Table in a design 0 – MSI-X Table not Present 1 – MSI-X Table Present
	8-6	RO	Multiple AXI Stream Support 000 - Single Stream Present 001 - Two Stream Present All Others - Reserved
	10-9	RO	AXI-ST Header and Data Packing Scheme 00 - Simple Packing
	31-11	RO	Reserved

7.3.1.3. Subsystem Interface Attributes

The register indicates Subsystem interface attributes settings during compile time.

Default Value: Set As per Parameter Settings

Table 75. Subsystem Interface attributes

Register Name	Bit	Attribute User Side	Description
Subsystem Interface Attributes	3-0	RO	Reflects AXI-ST Initiator Interface ready_latency setting
	7-4	RO	Reflects AXI-Lite Initiator Interface ready_latency setting
	11-8	RO	Reflects AXI-MM Initiator Interface ready_latency setting
	14-12	RO	Indicates AXI-ST Interface Width 000 - 32 bits 001 - 64 Bits 010 - 128 Bits

			011 - 256 Bits 100 - 512 Bits 101 - 1024 Bits All others - Reserved
	17-15	RO	Indicates AXI-Lite Interface Width 000 - 32 bits 001 - 64 Bits All others - Reserved
	20-18	RO	Indicates AXI-MM Interface Width 000 - 32 bits 001 - 64 Bits 010 - 128 Bits 011 - 256 Bits 100 - 512 Bits 101 - 1024 Bits All others - Reserved
	31-21	RO	Indicates AXI-MM Interface Burst Length The supported burst length = 2 power (number in this field)

7.3.1.4. ERROR GEN CTRL

The table below lists details of ERROR GEN CTRL from Application Error Reporting Registers. If header and prefix logging is required, then you must set the ERROR GEN CTRL register after ERROR TLP Header DWn and ERROR TLP Prefix registers are populated with required information.

Default Value: 0x0000_0000

Table 76. Error Gen Control Register

Register Name	Bit	Attribute User Side	Description
ERROR GEN CTRL	0	RW1S	Error Gen Trigger This bit will trigger Error Handling based on application error attributes at ERROR GEN ATTR, ERROR TLP Header DWn and ERROR TLP Prefix registers. Hence this bit should only be set once the application error attributes

			are programmed with the correct values. Once this bit is set, the application error attributes shouldn't be changed until the operation is completed. The Subsystem will report these error(s) by utilizing the application error reporting mechanism. The Subsystem will clear this bit when the requested operation is complete.
	1	RW	Log header If header logging is required, this bit must be set. The header must be supplied using Header Register Note: TLP Header error logging feature is not available for VF related errors (since AER is not supported for VF)
	2	RW	Log Prefix If prefix logging is required, this bit must be set. The prefix must be supplied using Prefix Register Note: TLP Prefix error logging feature is not available for VF related errors (since AER is not supported for VF)
	6-3	RsvdZ	Reserved
	7	RW	VF Active Indicates error reporting function is for a Virtual Function Note: Not supported in the current Quartus release
	12-8	RW	PF Number Indicates PF Number of function reporting an error Note: Current Quartus release supports up to 8 PFs only
	13	RsvdZ	Reserved
	24-14	RW	VF Number

			Indicates VF Number of function reporting an error Note: Valid only when VF Active is set Note: Not supported in the current release.
	25	RsvdZ	Reserved
	30-26	RW	Slot Number Indicates Slot Number of function reporting an error
	31	RsvdZ	Reserved

7.3.1.5. ERROR GEN ATTR

The table below lists details of ERROR GEN ATTR from Application Error Reporting Registers. When ERROR GEN CTRL operation is triggered and pending, the ERROR GEN ATTR shouldn't be programmed with new error values, otherwise newly updated error(s) might not be treated as new errors. It might also lead to pending error(s) being reported incorrectly.

Default Value: 0x0000_0000

Table 77. Error Gen Attribute Register

Register Name	Bit	Attribute User Side	Description
ERROR GEN ATTR	0	RW	Advisory Indicates application error (if applicable) is an advisory error. Examples when application can assert this are: - Treating a poisoned TLP as normal TLP - Detected Completion Timeout but intends to resend the request - Returning UR completion for a request that is treated as Unsupported Request - Returning CA completion for a request that is treated as Completer Abort - Receiving an unexpected completion
	1	RW	Unexpected Completion Error This bit should be set when an Application Layer master block detects an unexpected Completion. The Subsystem responds to this by reporting a Fatal or Correctable Error to the Root Complex, depending on the severity of the Unexpected Completion Received error programmed in the AER Uncorrectable Error Severity Register of the Function (if not masked).
	2	RW	Completer Abort Error This bit should be set when Application Layer has treated a request as a Completer Abort (CA). The Subsystem responds to this by reporting a Fatal or Non-Fatal/Correctable Error to the Root Complex, depending on the severity of the Completer Abort error programmed in the AER Uncorrectable Error Severity Register of the Function (if not masked).

	3	RW	Completion Timeout Error This bit should be set when a master-like interface has transmitted a Non-Posted request that never receives a corresponding Completion from the link and the error is not correctable. The Subsystem responds to this by reporting a Fatal or Non-Fatal/Correctable Error to the Root Complex, depending on the severity of the Completion Timeout error programmed in the AER Uncorrectable Error Severity Register of the Function (if not masked).
	4	RW	Unsupported Request Error This bit should be set when Application Layer has treated a request as an Unsupported Request. The Subsystem responds to this by reporting a Fatal or Non-Fatal/Correctable Error to the Root Complex, depending on the severity of the Unsupported Request error programmed in the AER Uncorrectable Error Severity Register of the Function (if not masked).
	5	RW	Poisoned TLP Received Error This bit should be set when Application Layer has treated a request as poisoned. The Subsystem responds to this by reporting a Fatal or Non-Fatal/Correctable Error to the Root Complex, depending on the severity of the Poisoned TLP Received error programmed in the AER Uncorrectable Error Severity Register of the Function (if not masked).
	6	RW	ECRC Check Failed Error

			This bit should be set when Application Layer has detected ECRC Check Failed error. The Subsystem responds to this by reporting a Fatal or Non-Fatal/Correctable Error to the Root Complex, depending on the severity of the ECRC Check Failed error programmed in the AER Uncorrectable Error Severity Register of the Function (if not masked). Note: Not supported in the current Quartus release.
	7	RW	AtomicOp Egress Blocked Error This bit should be set when Application Layer has encountered AtomicOp Egress Blocked error. The Subsystem responds to this by reporting a Fatal or Non-Fatal/Correctable Error to the Root Complex, depending on the severity of the AtomicOp Egress Blocked error programmed in the AER Uncorrectable Error Severity Register of the Function (if not masked). Note: Not supported in the current Quartus release.
	8	RW	Uncorrectable Internal Error This bit should be set when Application Layer has encountered Uncorrectable Internal error. The Subsystem responds to this by reporting a Fatal or Non-Fatal Error to the Root Complex, depending on the severity of the Uncorrectable Internal error programmed in the AER Uncorrectable Error Severity Register of the Function (if not masked). Note: Not supported in the current Quartus release.

	9	RW	Corrected Internal Error This bit should be set when Application Layer has corrected an internal error. The Subsystem responds to this by reporting a Correctable Error to the Root Complex (if not masked). Note: Not supported in the current Quartus release.
	10	RW	TLP Prefix Blocked Error This bit should be set when Application Layer has encountered TLP Prefix Blocked error. The Subsystem responds to this by reporting a Fatal or Non-Fatal/Correctable Error to the Root Complex, depending on the severity of the TLP Prefix Blocked error programmed in the AER Uncorrectable Error Severity Register of the Function (if not masked). Note: Not supported in the current Quartus release.
	11	RW	ACS Violation Error This bit should be set when Application Layer has encountered ACS Violation error. The Subsystem responds to this by reporting a Fatal or Non-Fatal/Correctable Error to the Root Complex, depending on the severity of the ACS Violation error programmed in the AER Uncorrectable Error Severity Register of the Function (if not masked). Note: Not supported in the current Quartus release
	12	RW	MC Blocked TLP Error This bit should be set when an Application layer detected a MC

			Blocked TLP error. The Subsystem responds to this by reporting a Fatal or Non-Fatal Error to the Root Complex, depending on the severity of the MC Blocked TLP error programmed in the AER Uncorrectable Error Severity Register of the Function (if not masked). Note: Not supported in the current Quartus release
	13	RW	Poisoned TLP Egress Blocked Error This bit should be set when an Application layer detected a Poisoned TLP Egress Blocked error. The Subsystem responds to this by reporting a Fatal or Non-Fatal Error to the Root Complex, depending on the severity of the Poison TLP Egress Blocked error programmed in the AER Uncorrectable Error Severity Register of the Function (if not masked). Note: Not supported in the current Quartus release
	31-14	RsvdZ	Reserved

7.3.1.6. ERROR TLP Header DW0-3

This register holds PCIe TLP Header information of the error packet reported using ERROR GEN CTRL register. When ERROR GEN CTRL operation is triggered and pending, the ERROR TLP Header DWn shouldn't be programmed with new values, otherwise wrong TLP Header values might be used.

Default Value: 0x0000_0000

Table 78. Error TLP Header DW Register

Register Name	Bit	Attribute User Side	Description
Header DWn	31 - 0	RW	Holds TLP Header DWn of the reported error packet.

			n indicates Header DWORD index

7.3.1.7. ERROR TLP Prefix

This register holds PCIe TLP Prefix information of the error packet reported using ERROR GEN CTRL register. When ERROR GEN CTRL operation is triggered and pending, the ERROR TLP Prefix shouldn't be programmed with new values, otherwise wrong TLP Prefix values might be used.

Default Value: 0x0000_0000

Table 79. Error TLP Prefix Register

Register Name	Bit	Attribute User Side	Description
Prefix	31 - 0	RW	Holds TLP Prefix of the reported error packet.

7.3.1.8. HOT PLUG GEN CTRL

The table below lists details of HOT PLUG GEN CTRL Register.

Default Value: 0x0000_0000

Table 80. Hot Plug Control Register

Register Name	Bit	Attribute User Side	Description
HOT PLUG GEN CTRL	0	RW	Attention Button Pressed Indicates that the system attention button was pressed The Subsystem pass on this information to HIP block and clears this bit indicating requested operation complete.
	1	RW	Power Fault Detection Indicates the power controller detected a power fault at this slot The Subsystem pass on this information to HIP block and clears this bit indicating requested operation complete.
	2	RW	MRL Sensor Changed

			Indicates that the state of the MRL sensor has changed The Subsystem pass on this information to HIP block and clears this bit indicating requested operation complete.
	3	RW	Presence Detect Changed. Indicates that the state of the card presence detector has changed. The Subsystem pass on this information to HIP block and clears this bit indicating requested operation complete.
	4	RW	Command Completed Indicates that the Hot Plug controller completed a command. The Subsystem pass on this information to HIP block and clears this bit indicating requested operation complete.
	5	RW	Data Link Layer State Change Indicates the state of Data Link Layer Link Active bit of the Link Status register is changed The Subsystem pass on this information to HIP block and clears this bit indicating requested operation complete.
	6	RW	MRL Sensor State This bit reports the status of the MRL sensor 0 - MRL Closed 1 - MRL Open
	7	RW	Presence Detect State

			0 - Slot Empty 1 - Adaptor Present
	8	RW	Electromechanical Interlock Status Indicates whether the system electromechanical interlock is engaged 0 - Electromechanical Interlock Disengaged 1 - Electromechanical Interlock Engaged
	13-9	RW	Slot Number Indicates Slot Number of Function Generating Hot Plug Even
	31 - 14	RsvdZ	Reserved

7.3.1.9. POWER MANAGEMENT CTRL

Default Value: 0x0000_0000

Table 81. Power Management Control Register

Register Name	Bit	Attribute User Side	Description
POWER MANAGEMENT CTRL	0	RW	Generate PME Message Wake Up. If PME is enabled and PME support is configured for current PMCSR D-state asserting this signal will cause the controller to wake from either L1 or L2 state. When the controller has transitioned back to the L0 state it will transmit a PME message and set the PME_Status. Upon receiving the PME message the root complex should clear the PME_Status and change the D-state back to D0 The Subsystem pass on this information to HIP block and clears this bit indicating requested operation complete.
	1	RW	Generate PME Turnoff Message Only Available in RC Mode The Subsystem pass on this information to HIP block and clears this bit indicating requested operation complete.

	2	RW	Delay PM_Enter_L23 response Indication from application that it is ready to enter the L23 state. The controller sends PM_Enter_L23 in response to PM_Turn_Off when this bit is set. Application that do not require this feature hardware initialize bit[3]. If bit[3] is set this bit is don't care from hardware point of view The Subsystem pass on this information to HIP block and clears this bit indicating requested operation complete.
	3	Hwinit	Autonomous PM_Enter_L23 response The controller sends PM_Enter_L23 in response to PM_Turn_Off
	15-4	RsvdZ	Reserved
	20-16	RW	PF Number Indicates PF Number of Function generating PME Note: Current Quartus release limits to max 8 PFs only
	31-21	RsvdZ	Reserved

7.3.1.10. LEGACY INTERRUPT CTRL

Default Value : 0x0000_0000

Table 82. Legacy Interrupt Control Register

Register Name	Bit	Attribute User Side	Description
LEGACY INTERRUPT CTRL	0	RW	Assert Message

			The application sets this bit when it wants to send assert message The Subsystem pass on this information to HIP block and clears this bit indicating requested operation complete.
	1	RW	Deassert Message The application sets this bit when it want to send deassert message The Subsystem pass on this information to HIP block and clears this bit indicating requested operation complete.
	15-4	RsvdZ	Reserved
	20-16	RW	PF Number Indicates PF Number of Function generating Assert or Deassert message Note: Current Quartus release limits to max 8 PFs only
	31-21	RsvdZ	Reserved

7.3.1.11. MSI-X GEN CTRL EP 0-31

The Subsystem reads MSI-X address and data information from the table pointed to by the vector number in this register and generates the MSI-X interrupt.

The Subsystem implements one MSIX GEN CTRL register for each endpoint in a system.

Usage of this control register is applicable only in Data Mover Mode.

Default Value: 0x0000_0000

Table 83. MSI-X Generation Control Register

Register Name	Bit	Attribute User Side	Description
MSIX GEN CTRL	0	RW	Generate Interrupt Writing '1' to this bit Triggers Interrupt. Write to this bit is ignored if bit is already set The Subsystem generates interrupt and clears this bit indicating requested operation complete.
	5-1	RW	PF Number Indicates Physical Function Number of Interrupt Note: Current Quartus release limits to max 8 PFs only
	6	RsvdZ	Reserved
	17-7	RW	VF Number Indicates Virtual Function Number of Interrupt
	18	RsvdZ	Reserved
	19	RW	VF Active Indicates Virtual Function is generating Interrupt
	31-20	RW	Vector Number Indicates Vector Number of Interrupt

7.3.1.12. MSI GEN CTRL

The Subsystem reads MSI Address and Data register and generates the MSI interrupt. The application must write this register after writing to MSI Address and Data register.

Usage of this control register is applicable only in Data Mover Mode.

Default Value: 0x0000_0000

Table 84. MSI Generation Control Register

Register Name	Bit	Attribute User Side	Description
MSI GEN CTRL	0	RW	Generate Interrupt Writing '1' to this bit Triggers Interrupt. Write to this bit is ignored if bit is already set The Subsystem generates interrupt and clears this bit indicating requested operation complete.
	5-1	RW	PF Number Indicates Physical Function Number of Interrupt Note: Current Quartus release limits to max 8 PFs only
	6	RsvdZ	Reserved
	17-7	RW	VF Number Indicates Virtual Function Number of Interrupt
	18	RsvdZ	Reserved
	19	RW	VF Active Indicates Virtual Function is generating Interrupt
	24-20	RW	Slot Number Indicates Slot Number of function generating Interrupt
	31-25	RsvdZ	Reserved

7.3.1.13. MSI Lower Address

Default Value: 0x0000_0000

Table 85. MSI Lower Address Register

Register Name	Bit	Attribute User Side	Description
MSI Address Lower	31 - 0	RW	Indicates lower 32 bits of MSI Address

7.3.1.14. MSI Upper Address

Default Value: 0x0000_0000

Table 86. MSI Upper Address Register

Register Name	Bit	Attribute User Side	Description
MSI Address Upper	31 - 0	RW	Indicates upper 32 bits of MSI Address

7.3.1.15. MSI Data

Default Value: 0x0000_0000

Table 87. MSI Data Register

Register Name	Bit	Attribute User Side	Description
MSI Data	31 - 0	RW	Indicates Data to be sent with MSI command

7.3.1.16. CFG REG IA CTRL

The table below lists details of configuration register indirect access control register.

Default Value: 0x0000_0000

Table 88. Configuration Register Indirect Access Control

Register Name	Bit	Attribute User Side	Description
CFG REG IA CTRL	0	RW	Initiate Access This bit should be set when a master-like interface wants to read from or write to PCIe configuration space register. The Subsystem performs read or write operation to a function pointed to by IA_FN_NUM register when this bit is set and clears this bit indicating requested operation is complete. Master cannot initiate new transaction if this bit is set

	1	RW	Access Type Indicates access type of operation. 0 - Read Operation 1 - Write Operation
	5-2	RW	Byte Enables Indicates Byte Enables of Write Operations 4'b0001: Write byte 0 4'b0010: Write byte 1 4'b0100: Write byte 2 4'b1000: Write byte 3 4'b1111: Write all bytes. Any Combinations of byte enables are valid, e.g., 1010, 1011 etc.
	14-5	RW	Register Address DWORD Address of Register
	31-15	RsvdZ	Reserved

7.3.1.17. CFG REG IA FN NUM

This register points to the function number of the configuration register the master is accessing through indirect access mechanism.

Default Value: 0x0000_0000

Table 89. Configuration Register Indirect Access Function Number

Register Name	Bit	Attribute User Side	Description
CFG REG IA FN NUM	2-0	RW	Function Type 000 - Indicates Access Physical Function 001 - Indicates Access Virtual Function All others - Reserved
	7-3	RW	PF Number The PF Number of register access Note: Current Quartus release limits to max 8 PFs only
	8	RsvdZ	Reserved
	19-9	RW	VF Number

			The VF Number of register access
	20	RsvdZ	Reserved
	25-21	RW	Slot Number The slot Number of register access
	31-26	RsvdZ	Reserved

7.3.1.18. CFG REG IA WRDATA

This register points to the function number of the configuration register the master is accessing through indirect access mechanism.

Default Value: 0x0000_0000

Table 90. Configuration Register Indirect Access Write Data

Register Name	Bit	Attribute User Side	Description
CFG REG IA WRDATA	31-0	RW	Write Data Data to be written into configuration register with write access Master writes this register with required Data before Initiating Write Access

7.3.1.19. CFG REG IA RDDATA

This register holds read data from read operation initiated by the master using indirect access mechanism.

Default Value: 0x0000_0000

Table 91. Configuration Register Indirect Access Write Data

Register Name	Bit	Attribute User Side	Description
CFG REG IA RDDATA	31-0	RO	Read Data Data read from configuration register with read access Master reads this register after read operation completion indicated by Initiate Access bit in CFG IA CTRL register

7.3.1.20. PRS CTRL

The Subsystem generates Page Request Service (PRS) events to the QHIP based on the settings of this PRS CTRL register.

Usage of this control register is applicable only when operating as Endpoint and with TLP Bypass disabled.

Default Value: 0x0000_0000

Table 92. Page Request Service (PRS) Control Register

Register Name	Bit	Attribute User Side	Description
MSI GEN CTRL	0	RW	Generate Page Request Service (PRS) Event Writing '1' to this bit triggers PRS event. Write to this bit is ignored if bit is already set The Subsystem generates PRS event and clears this bit indicating requested operation complete.
	5-1	RW	PF Number Indicates Physical Function Number of the PRS event Note: Current Quartus release limits to max 8 PFs only
	7-6	RsvdZ	Reserved
	8	RW	Response Failure: Indicate that the function has received a PRG response failure.
	9	RW	Unexpected Page Request Group Index: Indicate that the function has received a response with Unexpected Page Request Group Index
	10	RW	Stopped: Indicate that the function has completed all previously issued page requests and that it has stopped requests for additional pages. Only valid when the PRS enable bit is clear.
	19-11	RsvdZ	Reserved
	24-20	RW	Slot Number

			Indicates Slot Number of function generating Interrupt
	31-25	RsvdZ	Reserved

7.3.1.21. MSI PENDING CTRL

The application layer uses both MSI PENDING CTRL and MSI PENDING registers to update the MSI Pending Bits for each function.

Default Value: 0x0000_0000

Table 93. MSI Pending Control Register

Register Name	Bit	Attribute User Side	Description
MSI PENDING CTRL	0	RW	Update MSI Pending Bits Writing '1' to this bit causes an update to the MSI Pending Bits based on MSI PENDING value. Write to this bit is ignored if bit is already set The Subsystem clears this bit indicating the requested update is complete
	5-1	RW	PF Number Indicates Physical Function Number of the update request Note: Current Quartus release limits to max 8 PFs only
	6	RsvdZ	Reserved
	17-7	RW	VF Number Indicates Virtual Function Number of the update request
	18	RsvdZ	Reserved
	19	RW	VF Active Indicates Virtual Function is the target of the update request
	24-20	RW	Slot Number Indicates Slot Number of the update request
	31-25	RsvdZ	Reserved

7.3.1.22. MSI PENDING

Default Value: 0x0000_0000

Table 94. MSI Pending Register

Register Name	Bit	Attribute User Side	Description

MSI PENDING	31 - 0	RW	Indicates MSI Pending Bits value to be updated

7.3.1.23. D-STATE STS

The application layer can use this register to obtain the D-State values of each function. It
should write appropriate values to the PF Number, VF Number, VF Active and Slot Number fields first before issuing a read to obtain the D-State value of the corresponding function.

Default Value: 0x0000_0000

Table 95. D-State Status Register

Register Name	Bit	Attribute User Side	Description
D-STATE STS	4-0	RW	PF Number Indicates Physical Function Number of Interrupt Note: Current Quartus release limits to max 8 PFs only
	5	RsvdZ	Reserved
	16-6	RW	VF Number Indicates Virtual Function Number of Interrupt
	17	RsvdZ	Reserved
	18	RW	VF Active Indicates Virtual Function is generating Interrupt
	23-19	RW	Slot Number Indicates Slot Number of function generating Interrupt
	27-24	RsvdZ	Reserved
	31-28	RO	D-State Value Power Management D-State for each function per PF, VF, VF Active and Slot Number settings above. 0000: Uninitialized or Invalid 0001: D0 0010: D1 0100: D2 1000: D3

7.3.1.24. CFG RETRY CTRL

The application layer can use this register to update the per PF Configuration Retry Status Enable controls (CRS En Controls) driven to the Hard IP Controller. All VFs will share the same control as their parent PF. When the corresponding PF's CRS En Control is asserted, HardIP Controller will respond to configuration TLPs with a CRS (Configuration Retry Status) if it has not already responded to a Configuration TLP with non-CRS status since the last

reset. You can use this to hold off on enumeration.

Note: This register control allows the application layer to update 8 PFs at one time. If more than 8 PFs are used, the application layer needs to perform multiple updates by changing the PF Index field to point to the respective 8 PFs.

Default Value: 0x0000_0000

Table 96. Configuration Retry Control Register

Register Name	Bit	Attribute User Side	Description
CFG RETRY CTRL	0	RW	Update CRS En Control Writing '1' to this bit causes the Subsystem to update the corresponding CRS En Controls indicated by "Slot Number", "PF Index" and "PF Number". The Subsystem clears this bit when the update is complete. Write to this bit is ignored if bit is already set
	5-1	RW	Slot Number Indicates Slot Number of the CRS En Controls to be updated
	7-6	RsvdZ	Reserved
	9-8	RW	PF Index Indicates which 8 Physical Functions of the CRS En Controls to be updated 00 - PF7:PF0 01 - PF15:PF8 10 - PF23:PF16 11 - PF31:PF24 Note: Current Quartus release limits to max 8 PFs only
	15-10	RsvdZ	Reserved
	23-16	RW	PF Number Indicates up-to 8 Physical Functions (one-hot) of the CRS En Controls to be updated
	31-24	RsvdZ	Reserved

7.3.1.25. BUS NUMBER

The application layer can use this register to obtain the bus number of each function. The default value of PF Number field is "0" which indicates "PF 0", the bus number field will reflect bus number of PF 0 by default and will set Bus Number Valid to '1' when content of Bus Number is valid. To obtain the bus number of a particular function, the application will update the PF Number field with the required function number. In response SS will update the bus number field with the bus number of that function. Application must check the "Bus Number Valid" field to determine the validity

of requested information. Application must not change the PF Number field before obtaining result for previous request. Failure to do so may result in an erroneous result.

Table 97. Bus Number Status Register

Register Name	Bit	Attribute User Side	Description
BUS NUMBER	4-0	RW	PF Number Indicates Physical Function Number of request Note: Current Quartus release limits to max 8 PFs only
	17-5	RsvdZ	Reserved
	18	RO	Bus Number Valid 1. Bus Number Field has valid information for requested function 2. This field will be cleared by SS when application writes this register
	23-19	RsvdZ	Reserved [Use for Device Number in future]
	31-24	RO	Bus Number

7.3.1.26. DFH CSR - X_FEATURE_DFH

Default Value: 0x0000_0000_0000_00C0

Table 98. X_FEATURE_DFH Register

Register Name	Bit	Attribute User Side	Description
X_FEATURE_DFH	11-0	RO	Feature ID Set by GUI Parameter DFH_FID
	15-12	RO	DFH Major Version Number Field Set by GUI Parameter DFH_MAJOR_VER
	39-16	RO	Next DFH Byte Offset Set by GUI Parameter DFH_NEXT_BYTE_OFFSET
	40	RO	End of List Set by GUI Parameter DFH_END, if true set to '1' else set to '0'
	47-41	Rsvd	Reserved
	51-48	RO	DFH Minor Revision Set by GUI Parameter DFH_MINOR_REV
	59-52	RO	DFH Version Set by GUI Parameter DFH_VER
	63-60	RO	Feature Type Set by GUI Parameter DFH_FEATURE_TYPE

7.3.1.27. DFH CSR - X_FEATURE_GUID_L

Default Value: 0x9fc4_21cd_df63_cd30

Table 99. X_FEATURE_GUID_L Register

Register Name	Bit	Attribute User Side	Description
X_FEATURE_GUID_L	63-0	RO	Lower 64 bits of GUID Set by invisible GUI parameter

7.3.1.28. DFH CSR - X_FEATURE_GUID_H

Default Value: 0xb563_2d82_3368_4a28

Table 100. X_FEATURE_GUID_H Register

Register Name	Bit	Attribute User Side	Description
X_FEATURE_GUID_H	63-0	RO	Upper 64 bits of GUID Set by invisible GUI parameter

7.3.1.29. DFH CSR - X_FEATURE_CSR_ADDR

Default Value: 0x0000_0000_0000_0C00

Table 101. X_FEATURE_CSR_ADDR Register

Register Name	Bit	Attribute User Side	Description
X_FEATURE_CSR_ADDR	63-1	RO	CSR Address set to 60h
	0	RO	0- Relative Address 1 - Absolute Address Set to 0b

7.3.1.30. DFH CSR - X_FEATURE_CSR_SIZE_GROUP

Default Value: 0x0000_0000_0000_0000

Table 102. X_FEATURE_CSR_SIZE_GROUP Register

Register Name	Bit	Attribute User Side	Description

X_FEATURE_CSR _SIZE_GROUP	63-32	RO	CSR Size set to 0h
	31	RO	Parameter Exist or Not Set to 0b
	30-16	RO	Groups Features and Interfaces Set to 0h
	15-0	RO	Instance ID Set by GUI parameter INST_ID

7.3.2. Subsystem Debug Registers

The table below lists the debug registers implemented by the Subsystem.

The Subsystem debug registers starts from Base Address = 0x400

Table 103. Subsystem Debug Registers Address Map

Register Name	Offset
HIP Status	0x0000_0000
HIA Debug 1	0x0000_0004
HIA Debug 2	0x0000_0008
WR DM Debug 1	0x0000_000C
WR DM Debug 2	0x0000_0010
RD DM Debug 1	0x0000_0014
RD DM Debug 2	0x0000_0018
CPL TRK Debug 1	0x0000_001C
CPL TRK Debug 2	0x0000_0020
AXIMM Bridge Debug 1	0x0000_0024
AXIMM Bridge Debug 2	0x0000_0028
HIP BP CYCLES	0x0000_002C
HIA BP CYCLES	0x0000_0030
DM BP CYCLES	0x0000_0034
DMRD BP CYCLES	0x0000_0038
APP BP CYCLES	0x0000_003C
HIA RX BP CYCLES	0x0000_0040
DM RX BP CYCLES	0x0000_0044

7.3.2.1. HIP Status

Default Value: 0x0000_0000

Table 104. Hard IP Status Registers

Register Name	Bit	Attribute User Side	Description
HIP Status	0	ROS	Link Up Indication 0 - Link Down

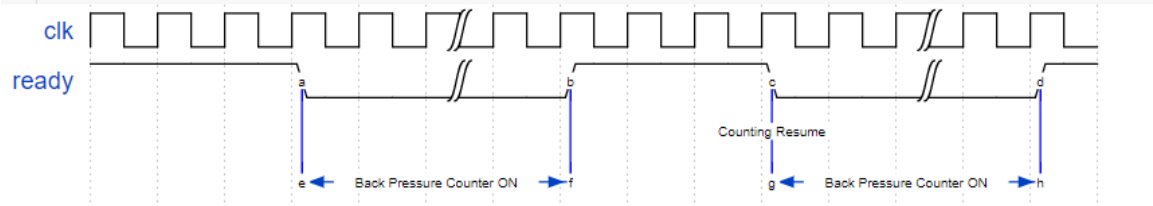
			1 - Link up
	1	ROS	Data Link Layer Active Indication 0 - DL not Active 1 - DL Active
	7:2	ROS	Indicates LTSSM State 6'h00: S_DETECT_QUIET 6'h01: S_DETECT_ACT 6'h02: S_POLL_ACTIVE 6'h03: S_POLL_COMPLIANCE 6'h04: S_POLL_CONFIG 6'h05: S_PRE_DETECT_QUIET 6'h06: S_DETECT_WAIT 6'h07: S_CFG_LINKWD_START 6'h08: S_CFG_LINKWD_ACCEPT 6'h09: S_CFG_LANENUM_WAIT 6'h0A: S_CFG_LANENUM_ACCEPT 6'h0B: S_CFG_COMPLETE 6'h0C: S_CFG_IDLE 6'h0D: S_RCVRY_LOCK 6'h0E: S_RCVRY_SPEED 6'h0F: S_RCVRY_RCVRCFG 6'h10: S_RCVRY_IDLE 6'h11: S_L0 6'h12: S_L0S 6'h13: S_L123_SEND_EIDLE 6'h14: S_L1_IDLE 6'h15: S_L2_IDLE 6'h16: S_L2_WAKE 6'h17: S_DISABLED_ENTRY 6'h18: S_DISABLED_IDLE 6'h19: S_DISABLED 6'h1A: S_LPBK_ENTRY 6'h1B: S_LPBK_ACTIVE 6'h1C: S_LPBK_EXIT 6'h1D: S_LPBK_EXIT_TIMEOUT 6'h1E: S_HOT_RESET_ENTRY 6'h1F: S_HOT_RESET 6'h20: S_RCVRY_EQ0 6'h21: S_RCVRY_EQ1 6'h22: S_RCVRY_EQ2 6'h23: S_RCVRY_EQ3
	8	ROS	User Mode

			0 - HIP is in non-user mode 1 - HIP in User Mode
	9	ROS	HIP PLD Interface Ready Indication 0 - HIP PLD Interface not Ready 1 - HIP PLD Interface Ready
	10	ROS	HIP Entered in Error Mode 0 - Normal Operation 1 - RAM ECC Error Detected by HIP
	31-11	ROS	Reserved

7.3.2.2. HIP BP CYCLES

The register indicates back pressure cycles observed because HIP transmit interface was not ready to accept transaction.

Figure 85. HIP BP Cycles



Default Value: 0x0000_0000

Table 105. Subsystem Interface attributes

Register Name	Bit	Attribute User Side	Description
HIP BP CYCLES	30-0	RW1C	Back Pressure Cycle Count
	31	RW1C	Indicates Overflow, cycle count reached 31'h7FFFFFFF

7.3.2.3. HIA BP CYCLES

The register indicates back pressure cycles observed because HIP interface adaptor transmit interface was not ready to accept transactions.

Default Value: 0x0000_0000

Table 106. HIA BP Registers

Register Name	Bit	Attribute User Side	Description
HIA BP CYCLES	30-0	RW1C	Back Pressure Cycle Count

	31	RW1C	Indicates Overflow, cycle count reached 31'h7FFFFFFF

7.3.2.4. DM BP CYCLES

The register indicates back pressure cycles observed because C2H transmit interface was not ready to accept transactions from application logic.

Default Value: 0x0000_0000

Table 107. DM BP Registers

Register Name	Bit	Attribute User Side	Description
DM BP CYCLES	30-0	RW1C	Back Pressure Cycle Count
	31	RW1C	Indicates Overflow, cycle count reached 31'h7FFFFFFF

7.3.2.5. DMRD BP CYCLES

The register indicates back pressure cycles observed because C2H read transmit interface was not ready to accept transaction from application logic.

Default Value: 0x0000_0000

Table 108. DMRD BP Registers

Register Name	Bit	Attribute User Side	Description
DMRD BP CYCLES	30-0	RW1C	Back Pressure Cycle Count
	31	RW1C	Indicates Overflow, cycle count reached 31'h7FFFFFFF

7.3.2.6. APP BP CYCLES

The register indicates back pressure cycles observed because application logic connected to H2C block was not ready to accept transactions.

Default Value: 0x0000_0000

Table 109. APP BP Registers

Register Name	Bit	Attribute User Side	Description
APP BP CYCLES	30-0	RW1C	Back Pressure Cycle Count
	31	RW1C	Indicates Overflow, cycle count reached 31'h7FFFFFFF

7.3.2.7. HIA RX BP CYCLES

The register indicates back pressure cycles observed because HIP interface adaptor receive interface was not ready to accept transactions.

Default Value: 0x0000_0000

Table 110. HIA RX BP Registers

Register Name	Bit	Attribute User Side	Description
HIA RX BP CYCLES	30-0	RW1C	Back Pressure Cycle Count
	31	RW1C	Indicates Overflow, cycle count reached 31'h7FFFFFFF

7.3.2.8. DM RX BP CYCLES

The register indicates back pressure cycles observed because C2H receive interface was not ready to accept transactions from HIA logic.

Default Value: 0x0000_0000

Table 111. DM RX BP Registers

Register Name	Bit	Attribute User Side	Description
DM RX BP CYCLES	30-0	RW1C	Back Pressure Cycle Count
	31	RW1C	Indicates Overflow, cycle count reached 31'h7FFFFFFF

7.3.3. Subsystem Performance Monitor Registers

The table below lists the performance registers implemented by the Subsystem. The Subsystem performance monitor registers starts from Base Address = 0x800

Table 112. Subsystem Performance Register Address Map

Register Name	Offset
PERFMON CTRL	0x0000_0000
TX MRD TLP	0x0000_0004
TX MWR TLP	0x0000_0008
TX MSG TLP	0x0000_000C
TX CFGWR TLP	0x0000_0010
TX CFGRD TLP	0x0000_0014
RX MRD TLP	0x0000_0018
RX MWR TLP	0x0000_001C
RX MSG TLP	0x0000_0020
RX CFGWR TLP	0x0000_0024
RX CFGRD TLP	0x0000_0028
TX MEM DATA	0x0000_002C
TX CPL DATA	0x0000_0030
RX MEM DATA	0x0000_0034
RX CPL DATA	0x0000_0038

7.3.3.1. PERFMON CTRL

The register controls various performance monitor counters in the Subsystem.

Default Value: 0x0000_0000

Table 113. PERFMON CTRL Registers

Register Name	Bit	Attribute User Side	Description
PERFMON CTRL	0	RW	Global Enable 1 - Turn On All Performance Counters 0 - Turn Off All Performance Counters
	1	RW	Clear Counters 1 - Clears all performance counters in a system 0 - No effect
	10-2	RW	Counting Duration in seconds 9'h000 - No Limit 9'h001 - 1 seconds 9'h002 - 2 seconds ……. 9'h1FF - 511 seconds
	31-11	RsvdZ	Reserved

7.3.3.2. TX MRD TLP

The register indicates number of memory read TLPs transmitted by the Subsystem.

Default Value: 0x0000_0000

Table 114. TX MRD TLP Registers

Register Name	Bit	Attribute User Side	Description
TX MRD TLP	31-0	RW1C	Number of Memory Read TLPs

7.3.3.3. TX MWR TLP

The register indicates number of memory write TLPs transmitted by the Subsystem.

Default Value: 0x0000_0000

Table 115. TX MWR TLP Registers

Register Name	Bit	Attribute User Side	Description
TX MWR TLP	31-0	RW1C	Number of Memory Write TLPs

7.3.3.4. TX MSG TLP

The register indicates the number of message TLPs transmitted by the Subsystem.

Default Value: 0x0000_0000

Table 116. TX MSG TLP Registers

Register Name	Bit	Attribute User Side	Description
TX MSG TLP	31-0	RW1C	Number of Message Write TLPs

7.3.3.5. TX CFGWR TLP

The register indicates the number of configuration write TLPs transmitted by the Subsystem.

Default Value: 0x0000_0000

Table 117. TX CFGWR TLP Registers

Register Name	Bit	Attribute User Side	Description
TX CFGWR TLP	31-0	RW1C	Number of Configuration Write TLPs

7.3.3.6. TX CFGRD TLP

The register indicates the number of configuration read TLPs transmitted by the Subsystem.

Default Value: 0x0000_0000

Table 118. TX CFGRD TLP Registers

Register Name	Bit	Attribute User Side	Description
TX CFGRD TLP	31-0	RW1C	Number of Configuration Read TLPs

7.3.3.7. RX MRD TLP

The register indicates the number of memory read TLPs received by the Subsystem.

Default Value: 0x0000_0000

Table 119. RX MRD TLP Registers

Register Name	Bit	Attribute User Side	Description
RX MRD TLP	31-0	RW1C	Number of Memory Read TLPs

7.3.3.8. RX MWR TLP

The register indicates the number of memory write TLPs received by the Subsystem.

Default Value: 0x000AXI0_0000

Table 120. RX MWR TLP Registers

Register Name	Bit	Attribute User Side	Description
RX MWR TLP	31-0	RW1C	Number of Memory Write TLPs

7.3.3.9. RX MSG TLP

The register indicates the number of message TLPs received by the Subsystem.

Default Value: 0x0000_0000

Table 121. RX MSG TLP Registers

Register Name	Bit	Attribute User Side	Description
RX MSG TLP	31-0	RW1C	Number of Message Write TLPs

7.3.3.10. RX CFGWR TLP

The register indicates the number of configuration write TLPs received by the Subsystem.

Default Value: 0x0000_0000

Table 122. RX CFGWR TLP Registers

Register Name	Bit	Attribute User Side	Description
RX CFGWR TLP	31-0	RW1C	Number of Configuration Write TLPs

7.3.3.11. RX CFGRD TLP

The register indicates the number of configuration read TLPs received by the Subsystem.

Default Value: 0x0000_0000

Table 123. RX CFGRD TLP Registers

Register Name	Bit	Attribute User Side	Description
RX CFGRD TLP	31-0	RW1C	Number of Configuration Read TLPs

7.3.3.12. TX MEM DATA

The register indicates data transmitted by a Subsystem for memory write operation.

Default Value: 0x0000_0000

Table 124. TX MEM Registers

Register Name	Bit	Attribute User Side	Description
TX MEM DATA	31-0	RW1C	Bytes Transferred 32'h00000000 - No bytes 32'h00000001 - 1 KB 32'h00000002 - 2 KB ……. 32'hFFFFFFFF - 4 TB

7.3.3.13. TX CPL DATA

The register indicates completion data transmitted by a Subsystem.

Default Value: 0x0000_0000

Table 125. TX CPL DATA Registers

Register Name	Bit	Attribute User Side	Description
TX CPL DATA	31-0	RW1C	Bytes Transferred 32'h00000000 - No bytes 32'h00000001 - 1 KB 32'h00000002 - 2 KB ……. 32'hFFFFFFFF - 4 TB

7.3.3.14. RX MEM DATA

The register indicates data received by a Subsystem for memory write operation.

Default Value: 0x0000_0000

Table 126. RX MEM DATA Registers

Register Name	Bit	Attribute User Side	Description
RX MEM DATA	31-0	RW1C	Bytes Transferred 32'h00000000 - No bytes 32'h00000001 - 1 KB 32'h00000002 - 2 KB ……. 32'hFFFFFFFF - 4 TB

7.3.3.15. RX CPL DATA

The register indicates completion data received by a Subsystem.

Default Value: 0x0000_0000

Table 127. RX CPL DATA Registers

Register Name	Bit	Attribute User Side	Description
RX CPL DATA	31-0	RW1C	Bytes Transferred 32'h00000000 - No bytes 32'h00000001 - 1 KB 32'h00000002 - 2 KB ……. 32'hFFFFFFFF - 4 TB

8. Appendix

8.1. P-tile Completion Buffer Size

P-tile PCIe Hard IP implements Completion Buffers for Header and Data for each PCIe core/port. In Endpoint mode, when Completion credits are infinite, user application needs to manage the number of outstanding requests according to the buffer size to prevent overflow and lost Completions packets.

Table <>. P-tile Completion Buffer Size

Completion Buffer	Depth	Width
Port 0 Cpl header	1144	NA
Port 0 Cpl data	1444	256
Port 1 Cpl header	572	NA
Port 1 Cpl data	1444	128
Port 2 Cpl header	286	NA
Port 2 Cpl data	1444	64
Port 3 Cpl header	286	NA
Port 3 Cpl data	1444	64

Please refer to section 4.4.8.1 of the P-Tile Avalon® Streaming Intel® FPGA IP for PCI Express* User Guide.

8.2. Power Management

Please refer to section 4.9 of the P-Tile Avalon® Streaming Intel® FPGA IP for PCI Express* User Guide.

8.3. MSI and MSI-X

Please refer to section 4.6.2 and 4.6.3 of the P-Tile Avalon® Streaming Intel® FPGA IP for PCI Express* User Guide.

9. Release Notes

The IP version (X.Y.Z) number may change from one Intel® Quartus® Prime software version to another. A change in:

- X indicates a major revision of the IP. If you update your Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Subsystem Intel FPGA IP for PCI Express IP Core v3.0.0

Table x. v3.0.0 2023.03.06

Intel Quartus Prime Version	Description	Impact
23.1	Removed SEP block	The SEP block is no longer embedded within the Subsystem but will be connected separately to the standalone Switch IP.
	Updated Synopsys supported simulator version.	Synopsys simulator version has changed from Q-2020.03-SP2 to Q-2022.06-SP1-1.
	Replaced one of the operating frequency selection options.	Removed 500 MHz operating frequency selection option and replaced with 470 MHz
	Replaced AXI-Lite Initiator	The Subsystem will no longer support AXI-Lite initiator as it has been upgraded from AXI-Lite initiator to AXI-Streaming in Data Mover Mode.
	Added MSI-X Table in Subsystem.	The Subsystem MSI-X Table will be maximum of 4096 across all endpoints while in Data Mover Mode.
	Hot Plug capability no longer supported.	With SEP block removed, Hot Plug is no longer supported.
	Enhanced Completion Reordering.	General enhancements have been made to the previous completion reordering.

	Enhance Device Address Translation Table.	General enhancements have been made to the previous Device Address Translation Table.
	Added Flow Control Mechanisms	Support added for Flow Control Mechanisms.

Table X. PCIe Subsystem IP Support Matrix

EP = Endpoint, RP = Root Port, BP = TLP Bypass. Support level keys: S = Simulation, C = Compilation, T = Timing, H = Hardware, N/A = Configuration not supported.

Configuration	PCIe IP Support			Design Example Support		
	EP	RP	BP	EP	RP	BP
Gen 4 x16 512-bit	S C T H	N/A	N/A	N/A	N/A	N/A
Gen4 x8/x8 256-bit	S C T H	N/A	N/A	N/A	N/A	N/A

10. Document Revision History

Date	Quartus Prime Pro Version	Changes
2021.12.03	v21.3	Initial release.
2022.12.12	V22.3	Initial release.
2023.05.11	V23.1	Updated release